# WindNinja Graphical User Interface Redesign

**Austen Harrell** AUSTEN.HARRELL@STUDENT.MONTANA.EDU
**Jack Hayward** JACK.HAYWARD@STUDENT.MONTANA.EDU
**Henry Jacobson** HENRY.JACOBSON@STUDENT.MONTANA.EDU
**Cody Matz** CODY.MATZ@STUDENT.MONTANA.EDU
*Montana State University*
*Bozeman, MT 59715, USA*

**Editor:** Natalie Wagenbrenner and Jason Forthofer
**Advisor:** Dr. Clemente Izurieta

## Abstract

WindNinja is a wind analysis tool created by the USDA Forest Service Missoula Fire Sciences Laboratory for managing wildfires. WindNinja predicts wind patterns through various geographic features such as mountains, ridges, and valleys. These features alter the surface elevation wind vectors against their higher elevation atmospheric vectors. Amalgamated together, this software is an improvement over NOAA's operational weather prediction models that are run over vast areas and are unable to resolve local terrain features. This data is pertinent because wildfires are highly sensitive to wind conditions. WindNinja offers information for predicting fire behavior and enhancing firefighter safety. While the WindNinja application has several interfaces, the graphical user interface was developed in a now deprecated C++ framework over fifteen years ago. In addition, data results from running WindNinja cannot be viewed natively; users have to view results in external third-party applications such as ArcGIS or Google Earth. This project addressed these issues by updating the graphical user interface using a modern framework and allowing users to view georeferenced wind data on a map-based layer natively. The updated application is fully cross-platform between Windows, MacOS, and Linux operating systems. Lastly, WindNinja remains free and open source, in accordance with Title 17 Section 105 of the United States Code for software written by a government agency.

## 1 Introduction

In 2023, over 55,000 wildfires were reported in the United States alone, burning nearly 2.6 million acres of land, destroying over 4300 structures, and causing hundreds of billions of dollars in damages [1, 2, 3]. The top five states at the highest risk for property damage, including California, Colorado, and Idaho, have heavily forested mountainous regions, which make fire management exceptionally difficult [4]. According to Idaho Firewise, weather, fuel, and topography are the three key factors in determining fire behavior, and mountainous forests create ideal fire conditions based on all three factors [5]. Fuel and topography are relatively easy factors to account for since both are constant. However, weather conditions are a difficult variable to consider. The National Oceanic and Atmospheric Administration (NOAA) operational weather prediction models work well over large, flat areas but are poor predictors when more granularity is required in a smaller area, especially over local terrain features.

WindNinja is a free and open-source wind analysis tool developed in 2004 by the USDA Forest Service Missoula Fire Sciences Laboratory to solve the shortcomings of NOAA weather prediction models. WindNinja predicts wind patterns through various geographic features such as mountains, ridges, and valleys. These features alter the surface elevation wind vectors against their higher elevation atmospheric vectors. By using local topographical data combined with high-elevation wind data, WindNinja uses sophisticated fluid dynamics modeling to simulate and predict low-elevation wind behavior. Access to these complex models allows firefighters and local governments to more accurately predict wildfire behavior, allowing for advanced safety measures and wildfire containment strategies. In addition to the firefighting advantages offered by WindNinja, the software is also used in fields such as wind power generation, hydrology modeling, and snow avalanche prediction. Since its creation in 2004, WindNinja has been maintained by the Missoula Fire Sciences Laboratory team and continues to support Windows and Linux operating systems. The application includes several interfaces, including a command line interface (CLI) and a graphical user interface (GUI), however, the GUI hasn't been updated since the application's early inception. The GUI is built on a now-deprecated Qt framework, which poses several challenges for upgrades, improvements, maintenance, and build processes today. In addition, data results from running WindNinja cannot be viewed natively; users must view results in external third-party applications such as ArcGIS or Google Earth. The goal of this effort is to update the graphical user interface using a modern framework and allow users to natively view georeferenced wind data on a map-based layer. The application must remain cross-platform between Windows, MacOS, and Linux operating systems.

## 2 Work Schedule

### 2.1 Milestones and Timeline

**1. Initial UI Design (January 19 – February 9)**
Draft wireframes and mockups, define key user flows and component requirements, and review designs with stakeholders.

*Deliverables:* High-level mockups, component lists, UX goals documentation.

**2. Layout Implementation and Development Environment Setup (February 9 – March 16)**
Develop the base window structure in Qt6 and implement static UI elements. Set up Oracle Virtualbox virtual environment for development work.

*Deliverables:* Functional static layout and early feedback report.

**3. Fix WindNinja API (March 16 – April 6)**
Complete the outstanding WindNinja API tasks. Add missing methods, fix broken methods, and use the API with the existing (legacy) GUI.

*Deliverables:* Functional API.

**4. Core Features Development (March 2 – April 6)**
Connect the UI to the WindNinja API, implement user input handling, embed the Leaflet map viewer, and display basic simulation outputs on the map.

*Deliverables:* Functional simulation execution, visualized output, updated documentation.

**5. Integration and Finalization (March 23 – April 17)**
Finalize GUI and backend integration, complete output visualization features, and conduct cross-platform testing and user acceptance tests.

*Deliverables:* Finalized application, cross-platform builds, testing reports, usability feedback.

### 2.2 Lifecycle Approach and Key Elements

The WindNinja GUI redesign project will follow an incremental and iterative lifecycle. Development will be divided into overlapping phases to allow for frequent deliverables, early feedback, and continuous integration of improvements after meetings with the client. Weekly SCRUM stand-up meetings will ensure effective communication, task distribution, and steady progress from week to week. Task tickets are assigned weekly based on availability. This approach provides the flexibility to adapt quickly to feedback from the Fire Sciences Laboratory team and mitigates risk by continuously testing developments instead of waiting until the final stages. In addition, and when feasible, backend and frontend development will proceed in parallel to maximize efficiency. Communication will be maintained through weekly SCRUM meetings and feedback sessions with the Fire Sciences Laboratory team to ensure alignment with project goals.

## 3 Qualifications–Resumes Below

# Cody Matz

Computer Science

115 Michael Grove,
Bozeman, MT 59718
**(360) 219-8869**
**Codymm03@gmail.com**

## OBJECTIVE

Hard-working current undergraduate computer science student with a passion for computing and problem-solving, seeking new experience to utilize and build on my academic knowledge and programming skills in a professional environment. Along with educational experience in Computing/ development, I also have professional experience working with people, supporting their IT needs.

## EDUCATION

### Montana State — *Computer Science with Geographic Information Systems Minor*

**August 2021 - Present**

Currently a Senior in standing, with a 3.98 GPA. While working toward my degree, I have taken courses in Software Engineering, Computer Systems, Data Structures and Algorithms which have created growth in my understanding of the professional aspects of computers. Along with Advanced Geographic Information Systems, and GPS Mapping, I have knowledge in the utilization of computing to connect with my passion for nature.

To view some of my GIS work go to https://codymatz.myportfolio.com/

### W. F. West High School

**Graduated June 2021**

- Valedictorian for Class of 2021 with 4.0 GPA
- Honors in STEM and Humanities
- Participated in Varsity Math Clb, Leadership, Knowledge Bowl, and Robotics Team

## WORK EXPERIENCE

### Basin Electric Power Cooperative— *Operational Technology*

**August 2023 - May 2024**

At BEPC, I worked with the operational technology team gaining skills in the data managment, system administration, and management of critical infrastructure as it relates to control systems in power generation and transmission to maintain reliability and security.

### Montana State University Auxiliary Services— *ResNet IT Help Desk*

**August 2023 - May 2024**

While working for ResNet IT Help Desk, I worked as customer service with students and staff. Which provided technical assistance with networking, hardware and software support. Also, I supported the installation of network systems on the Montana State University campus.

## SKILLS

Software Development

Geographic Information Systems Experience

Creative

Objective Oriented

## AWARDS

**Valedictorian**
W. F. West Class of 2021

**W. F. West Scholarships**
Chehalis foundation scholarship recipient of Migratory Student, Rainier Connect, and the W.F. West Scholarship, when graduating High School

**Paxton Family Scholarship**
awarded while attending Montana State University

**Montana State University Presidents and Deans List**
awarded every semester I have attended Montana State University

## NOTABLE COURSEWORK

**Web Development - CSCI331**

**UI Design - CSCI443**

**Software Engineering - ESOF322**

**Computer Systems - CSCI366**

**Applied GIS & Spatial Analysis - GPHy484R**

# Austen Harrell

github.com/aaharrell linkedin.com/in/austen-harrell harrell1st@gmail.com 801-633-8183

## Education

**Montana State University** May 2025
*Bachelor of Science in Computer Science - Mechatronics Minor* *Overall GPA: 3.4/4.0*

**Montana State University** May 2020
*Bachelor of Science in Mechanical Engineering*

## Skills

**Languages**: C/C++, C#, Rust, Java, Python, HTML/CSS
**Tools**: Git/GitHub, Unix Shell, AutoCAD, SolidWorks, ArcGIS, ANSYS, MS Excel

## Experience

**Ulteig Engineering, Inc.** | *Transmission Design Engineer* October 2023 – Present

- Working as a consulting design engineer to oversee and design high voltage transmission projects from the initial scoping level through to project construction.
- Create and upkeep detailed project schedules to meet key deadlines throughout project life cycle.
- Write comprehensive estimates at various stages of the project development with accuracy ranging from 20% at project initiation to 2% at project construction.
- Design complex power transmission systems ranging from 69kV to 500kV using 3D modeling software PLS-CADD.
- Select, order, and coordinate the material necessary to construct the design, often with challenging and problematic material lead times.
- Demonstrate project management ability by coordinating the efforts of construction crews, land rights agents, and other client contractors.
- Provide timely and effective construction support during the building process to address any unanticipated constructability issues.
- Lead large meetings with the client, supporting engineers, vendors, construction crews, land rights agents, and contractors to coordinate project efforts to facilitate project milestones.

**HDR, Inc.** | *Transmission Engineer* February 2021 – October 2023

- Working as a consulting design engineer to oversee and design high voltage transmission projects from the initial scoping level through to project construction.
- Create and upkeep detailed project schedules to meet key deadlines throughout project life cycle.
- Write comprehensive estimates at various stages of the project development with accuracy ranging from 20% at project initiation to 2% at project construction.
- Design complex power transmission systems ranging from 69kV to 500kV using 3D modeling software PLS-CADD.
- Select, order, and coordinate the material necessary to construct the design, often with challenging and problematic material lead times.
- Demonstrate project management ability by coordinating the efforts of construction crews, land rights agents, and other client contractors.
- Provide timely and effective construction support during the building process to address any unanticipated constructability issues.
- Lead large meetings with the client, supporting engineers, vendors, construction crews, land rights agents, and contractors to coordinate project efforts to facilitate project milestones.

**Montana State University** | *SmartyCats Tutor* Present

- Schedule meetings with and tutor MSU students in various computer science, mathematics, and engineering courses.

**Enterprise Holdings** | *Customer Service Agent* 2019-2020

- Rented vehicles to customers at the front desk, addressed customer service complaints, and performed inventory/fleet management.

# *Jack Hayward*

512 ½ S Grand Ave • Bozeman, MT • (303)-562-5575 → flapjack90@gmail.com

## EDUCATION

Montana State University, BS Computer Science - Minor in Geographic Information Systems
2025 Expected graduation

## SKILLS

Java, Python, Technical Support, Linguistics, Creative Problem Solving, Spatial Thinking

## WORK HISTORY

**Ski Bootfitter/Sales representative,** *Roundhouse Ski and Sports 7/2020 - present*

- Custom ski boot fitting, which requires a high level of one on one collaboration with customers. Each fit is catered specifically for an individual.
- Expert industry knowledge, in a constantly evolving industry loaded with jargon and buzzwords, I've developed the skill of keeping up with the trends.

**Sales Representative,** *Cellular Plus - Verizon, Bozeman MT, 8/2019 - 7/2020*

- Selling smartphones, tablets, and connected devices. As well as selling verizon service, insurance, and internal marketing
- Technical Support for a wide range of software and hardware
- Management of serialized inventory

## ACTIVITIES

- Freeride World Qualifier, 2019-present
- Mountain Biking
- Rock Climbing
- High School Alpine Ski Team, 2015-2018
- Boy Scouts, 2010-2018

## HONORS/AWARDS

- Earned the rank of Eagle Scout, 2018

## LEADERSHIP EXPERIENCE

- Traveled to Japan on behalf of the United States and the city/county of Broomfield, CO as a youth delegate
- Elected "Outdoors Manager" for 7 consecutive terms in Boy Scout Troop
- Low Brass Section Leader in High School Marching Band

# Henry C. Jacobson

Bozeman, Montana || +1 (253)-326-1575 || henrycjmsu@gmail.com


## Education

**Montana State University**, Bozeman, MT GPA: 3.96 August 2021 – May 2025
Data Science B.S., Asbjornson College of Engineering || Highest Distinction Honors
**Languages:** English, Danish
**Relevant Coursework**: Machine Learning, Data Science for Sustainability, Algorithms I & II, Discrete Mathematics, Linear Algebra, Business Management & Information Systems, Honors Leadership, Databases, Upper Divisional Statistics, Calc I - III
**Upcoming Coursework:** Data Mining, Artificial Intelligence, Data Science Capstone, Honors Research Thesis

## Professional Experience

**Université Lumière Lyon 2**, France May 2025 — June 2025
*Course Assistant, CSCI 493 –Motivating Data Science* *Faculty: Dr. John Paxton, john.paxton@montana.edu*
• Facilitate collaborative data science projects between Montana State and French students, providing instructional support and fostering cross-cultural teamwork..
**Montana State University**, Bozeman, MT January 2022 — May 2025
*Teaching Assistant *(CS145RA—Web Dev|| CSCI107—Joy and Beauty of Data || HONR210—Mentoring Gifted Students)*
• Lead a lab section in developing web pages with HTML and CSS code. || Assist professor in answering student questions, grading python code, and hosting office hours. || Mentor seminar discussions, plan and implement projects, and evaluate mentee's academic formation. *Respectively
*Honors Fellowship, HONR250—Text and Critics*
• Moderate class discussions, encourage participation; observe, evaluate, and optimize student performance.

## Leadership Experience

**Mary Bridge Children's Hospital and Seattle Children's Hospita**l, Washington May 2013—August 2021
*Philanthropist & Music Therapist*
• Raised over $50,000 for Seattle's Uncompensated Care Program and Tacoma's Greatest Need program by selling my music compositions and hosting benefit concerts; provided music therapy to pediatric oncology patients.

## Academic Projects

**Københavns Universitet,** Copenhagen, Denmark January 2024 — May 2024
*Quantitative Finance Research* *Advisor: Dr. Rolf Poulsen, rolf@math.ku.dk*
• Replication and robustness study; implementation of applied mathematics and machine learning to investigate established publication from Bespoke Investment Group.

## Academic Awards / Nominations

• **Montana State University Presidential Scholar** August 2021 — May 2025
The most distinguished academic award–recognizes scholastic achievement, leadership, and unique personal qualities. Full tuition.
• **Student Representative for Gianforte Hall Project Work Group** February 2023 - May 2025
Selected by the Director of the Gianforte School of Computing to offer current student perspectives at bi-weekly board meetings for the forthcoming building.
• **Student Finalist for Regeneron International Science & Engineering Fair** May 2021
Twenty-seven young entrepreneurs, innovators, and scientists were selected from over 1,800 applicants Nationwide. Title: *Avalanche Snow Probes versus Conventional Snow Pit in Determining Avalanche Danger*

## 4 Background

Various tools and research initiatives tackle wind flow simulation, each with specific applications and unique limitations. One example is OpenFOAM, which serves as a comprehensive open-source CFD toolbox. It is widely utilized in disciplines such as aerodynamics and heat transfer [5]. Despite its utility, it lacks specialization in terrain-adjusted wind modeling. Another example is FluidX3D, which represents a highly developed approach to fluid simulations, implementing the Lattice Boltzmann machine learning methodology optimized for high-performance computing [6]. While this software excels in general fluid dynamics and offers impressive visualization capabilities, it remains too broad as an application; we need to create a simulation of terrain-influenced wind prediction within Windninja. Further, FluidX3D's research-oriented nature makes it less suitable for operational firefighting scenarios where ease of use is crucial [1, 2]. Another research-based platform, NOAA's Weather Prediction software, provides more comprehensive atmospheric display capabilities at various scaling [8]. While the models excel at large-scale weather prediction, they are much too detailed for smaller-scale models [3]. The models' complexity and operational overhead create barriers to practical field deployment and practical runtimes. Lastly, WindSim has established itself as a leading CFD software solution, particularly within wind energy [7]. It is sophisticated in its own right, making it ideal for wind farm optimization and planning; however, the primary focus is on energy applications, combined with its proprietary nature and outlandish cost, which limits its applicability for copious types of operations–its complex interface makes it less suitable for the user [4]. Our modernization of WindNinja's GUI directly addresses all these limitations by combining specialized terrain-adjusted wind modeling with contemporary visualization capabilities through Leaflet integration and Qt Framework [9] architecture.

## 5 Proposal Statement

Our addition to WindNinja is to provide a new and improved user experience by utilizing modern frameworks and incorporating existing technologies to convey information in a useful and efficient manner. This includes developing a new cross-platform interface using QT6. In addition to this upgrade, we are integrating a built-in visualization tool powered by Leaflet, enabling the team to view georeferenced wind simulation outputs directly within the application itself. This would eliminate any need for third-party software, streamlining the workflow for the users and maintaining the government standards the team upholds. With these advancements in place, we aim to modernize the Missoula Fire Laboratory's tool in order to create a more dynamic and accessible software.

### 5.1 Functional Requirements

F-1. An appropriate architecture shall be utilized for development of a new GUI.

Acceptance criteria: Given that the user compiles the application on the current development standard, the program should compile successfully without requiring any legacy software.

User stories: As a developer, I want to utilize WindNinja in another application so I can include it in my projects. As a wildland firefighter, I want the system to be up to date so I can accurately predict wind directions while fighting fires.

F-2. The application shall include the same features as the old GUI.

Acceptance criteria: Given that a user is creating a simulation, when they open the application, the interface should appear familiar and be easy to use.

User stories: As a researcher, I want to be able to use the application for my studies so I can analyze wind and fire patterns without issues. As a wildland firefighter, I want the system to be quickly usable so I do not have to learn a completely new interface.

F-3. The application shall utilize the WindNinja C API.

Acceptance criteria: Given that a developer is working with the program, when they make changes to the API, the application should reflect and utilize those changes accordingly.

User stories: As a developer, I want to be able to modify the WindNinja algorithm to provide accurate simulations for users. Additionally, I want the GUI to be easily extensible using the WindNinja API so I can introduce new features without having to alter existing code significantly.

F-4. A GUI shall include a graphics window to display surface wind output.

Acceptance criteria: Given that a user runs a simulation, when the program completes the simulation, the application should display the results in an integrated graphics window.

User stories: As a wildland firefighter, I want the simulation results to be easily visible so I can make quick decisions in the field. As a researcher, I want the results clearly presented so I can validate them without relying on another tool. As a developer, I want the results view to be built-in so I don't need to worry about external data pipelines.

F-5. The application shall be cross-platform and open source.

Acceptance criteria: Given that a user is on Linux, macOS, or Windows, when the application is deployed, it should be fully accessible and operable across all these platforms without requiring a license.

User stories: As a macOS user, I want to be able to generate simulations using the application. As a Windows user, I want the same access and capabilities

### 5.2 Non-Functional Requirements

Efficiency: The application must operate efficiently on standard-issue laptops without requiring specialized hardware.

User Familiarity: The interface must remain intuitive for new users while preserving familiarity for existing WindNinja users to minimize retraining.

Maintainability: The codebase must be modular, clearly documented, and structured according to open-source best practices to support future development.

### 5.3 Performance Requirements

Cross-Platform Resources: Performance and system resource usage must remain consistent across Windows, MacOS, and Linux platforms, ensuring a uniform user experience.

Scalability: The application must be capable of scaling up to handle larger datasets or longer simulations without significant changes to the core system design.

System Impact: The application must not monopolize CPU or memory resources during simulations, allowing other standard applications to continue operating normally on the user's system.

### 5.4 Interface Requirements

Parameter Input: The GUI must present all necessary WindNinja simulation parameters through organized and clearly labeled input fields.

Visualization Pane: An embedded Leaflet-based map interface must support zooming, panning, and toggling between simulation layers.

Input Validation: The system must validate user inputs at the GUI level to prevent invalid simulation configurations.

Error Handling: Errors and important system messages must be presented clearly and non-intrusively through notification dialogs integrated within the GUI.

### 5.5 Architectural Design

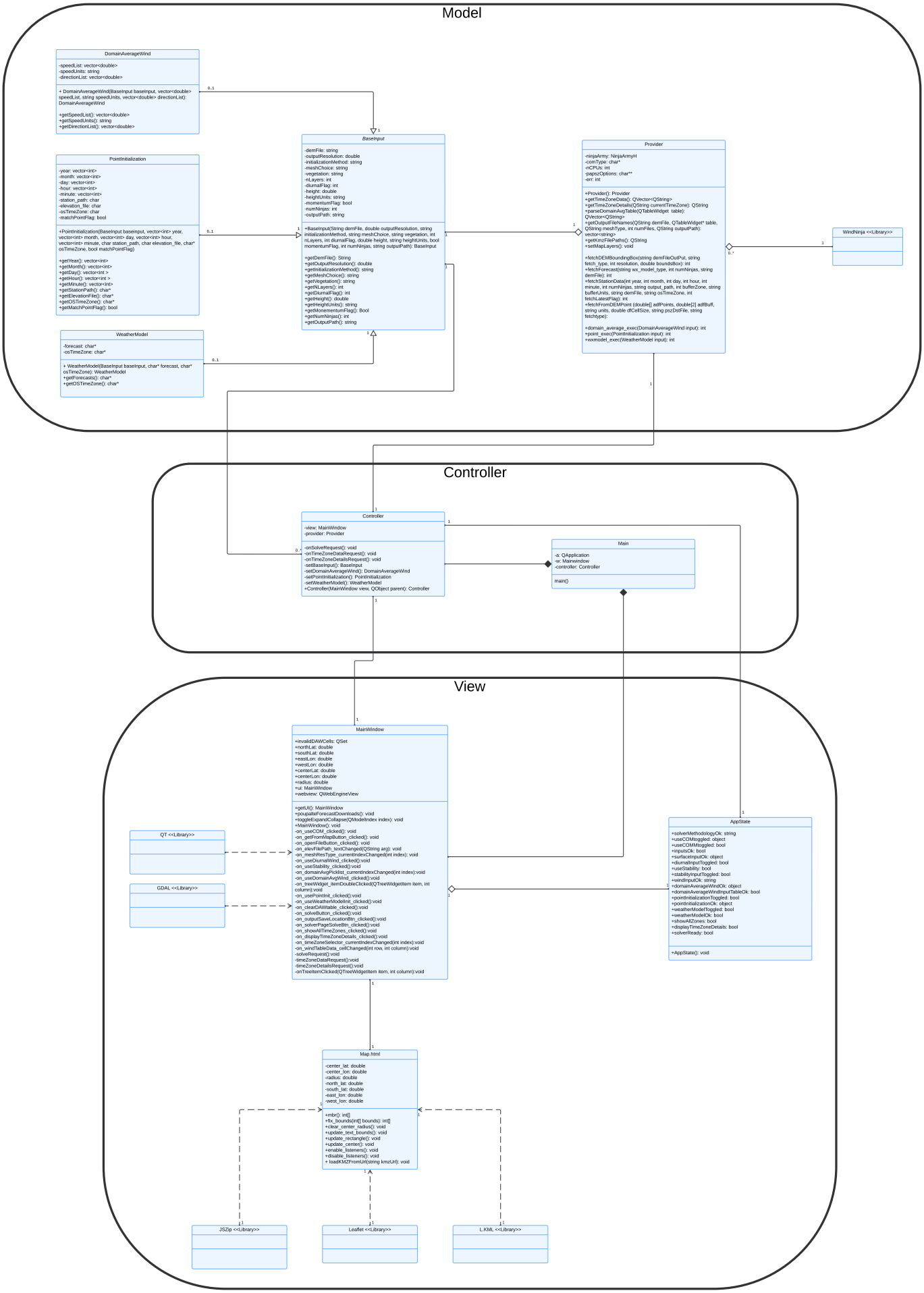
The WindNinja GUI employs a Model-View-Controller (MVC) architecture, where the Model manages simulation logic and interfaces with the WindNinja C API, the View handles user interactions and visualizes results through Qt6 and Leaflet, and the Controller coordinates data flow between them. Further architectural details are provided in the Methodology section.

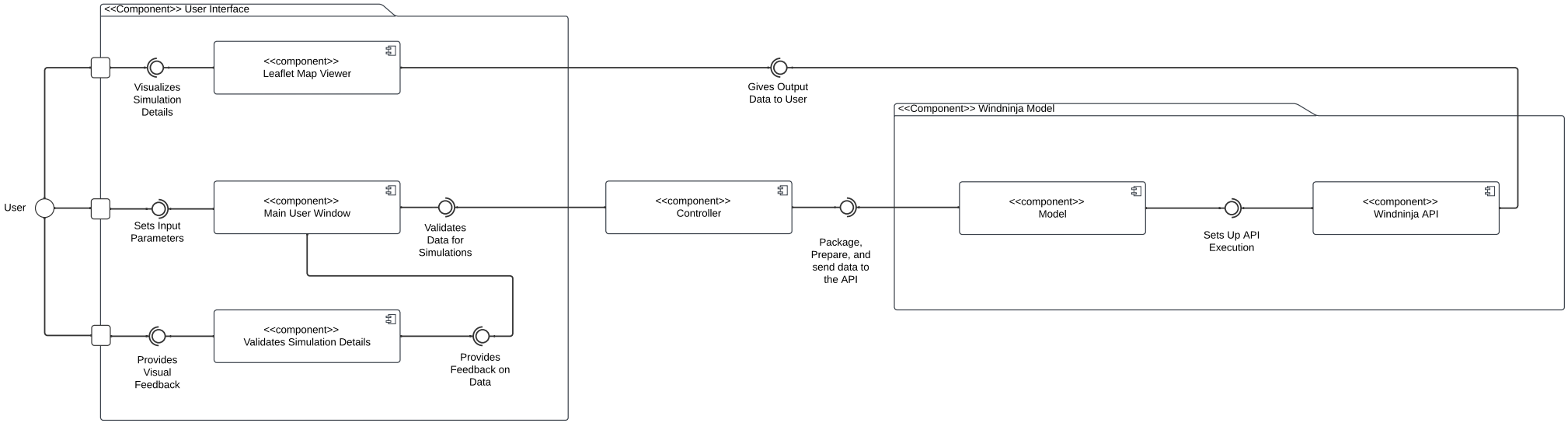
### 5.6 Development Standards and Tools Used

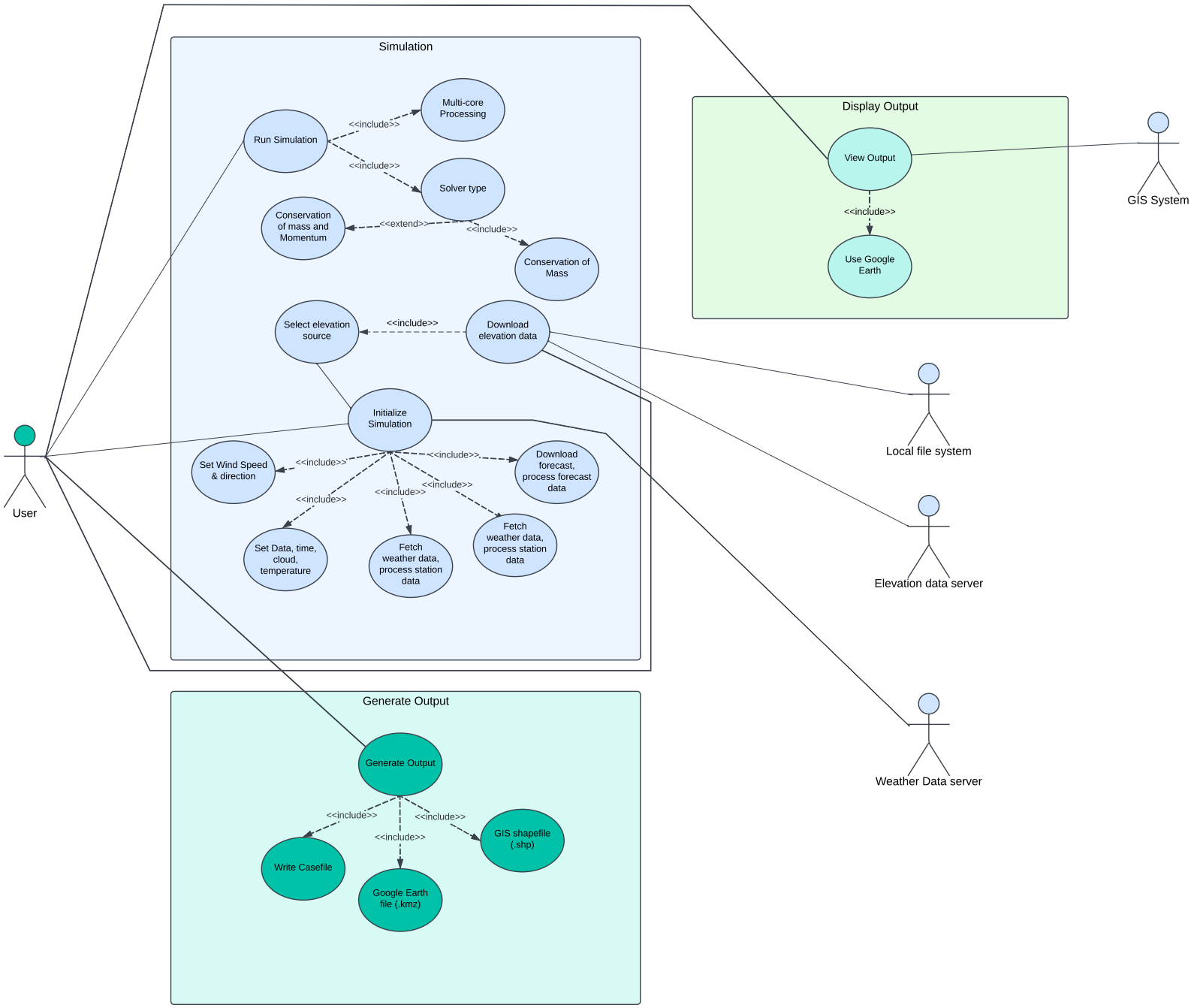
After setting up Ubuntu 22.04 using Windows Subsystem for Linux (WSL), we used Qt6 for GUI development and Leaflet.js for integrated geospatial visualization. For programming languages, C was used for core application development, with extensions interfacing with the team's computational fluid dynamics API. Our build system utilized Cmake for cross-platform build management. All of the commits were used with Git and our weekly SCRUM meetings and task management was done with a Kanban-style board.

## 6 Methods

### 6.1 UML Diagrams – See Below


Model
DomainAverageWind
-speedList: vector<double>
-speedUnits: string
-directionList: vector<double>
+ DomainAverageWind(BaseInput baseInput, vector<double> speedList, string speedUnits, vector<double> directionList): DomainAverageWind
+getSpeedList(): vector<double>
+getSpeedUnits(): string
+getDirectionList(): vector<double>
0..1
1
BaseInput
-demFile: string
-outputResolution: double
-initializationMethod: string
-meshChoice: string
-vegetation: string
-nLayers: int
-diurnalFlag: int
-height: double
-heightUnits: string
-momentumFlag: bool
-numNinjas: int
-outputPath: string
+BaseInput(String demFile, double outputResolution, string initializationMethod, string meshChoice, string vegetation, int nLayers, int diurnalFlag, double height, string heightUnits, bool momentumFlag, int numNinjas, string outputPath): BaseInput
+getDemFile(): String
+getOutputResolution(): double
+getInitializationMethod(): string
+getMeshChoice(): string
+getVegetation(): string
+getNLayers(): int
+getDiurnalFlag(): int
+getHeight(): double
+getHeightUnits(): string
+getMonementumFlag(): Bool
+getNumNinjas(): int
+getOutputPath(): string
PointInitialization
-year: vector<int>
-month: vector<int>
-day: vector<int>
-hour: vector<int>
-minute: vector<int>
-station_path: char
-elevation_file: char
-osTimeZone: char
-matchPointFlag: bool
+PointInitialization(BaseInput baseInput, vector<int> year, vector<int> month, vector<int> day, vector<int> hour, vector<int> minute, char station_path, char elevation_file, char* osTimeZone, bool matchPointFlag)
+getYear(): vector<int>
+getMonth(): vector<int>
+getDay(): vector<int >
+getHour(): vector<int >
+getMinute(): vector<int>
+getStationPath(): char*
+getElevationFile(): char*
+getOSTimeZone(): char*
+getMatchPointFlag(): bool
0..1
1
Provider
-ninjaArmy: NinjaArmyH
-comType: char*
-nCPUs: int
-papszOptions: char**
-err: int
+Provider(): Provider
+getTimeZoneData(): QVector<QString>
+getTimeZoneDetails(QString currentTimeZone): QString
+parseDomainAvgTable(QTableWidget table): QVector<QString>
+getOutputFileNames(QString demFile, QTableWidget* table, QString meshType, int numFiles, QString outputPath): vector<string>
+getKmzFilePaths(): QString
+setMapLayers(): void
+fetchDEMBoundingBox(string demFileOutPut, string fetch_type, int resolution, double boundsBox): int
+fetchForecast(string wx_model_type, int numNinjas, string demFile): int
+fetchStationData(int year, int month, int day, int hour, int minute, int numNinjas, string output_path, int bufferZone, string bufferUnits, string demFile, string osTimeZone, int fetchLatestFlag): int
+fetchFromDEMPoint (double[] adfPoints, double[2] adfBuff, string units, double dfCellSize, string pszDstFile, string fetchtype):
+domain_average_exec(DomainAverageWind input): int
+point_exec(PointInitialization input): int
+wxmodel_exec(WeatherModel input): int
0..*
1
WindNinja <<Library>>
WeatherModel
-forecast: char*
-osTimeZone: char*
+ WeatherModel(BaseInput baseInput, char* forecast, char* osTimeZone): WeatherModel
+getForecast(): char*
+getOSTimeZone(): char*
0..1
1
Controller
Controller
-view: MainWindow
-provider: Provider
-onSolveRequest(): void
-onTimeZoneDataRequest(): void
-onTimeZoneDetailsRequest(): void
-setBaseInput(): BaseInput
-setDomainAverageWind(): DomainAverageWind
-setPointInitialization(): PointInitialization
-setWeatherModel(): WeatherModel
+Controller(MainWindow view, QObject parent): Controller
0..*
1
Main
-a: QApplication
-w: Mainwindow
-controller: Controller
main()
1
View
MainWindow
+invalidDAWCells: QSet
+northLat: double
+southLat: double
+eastLon: double
+westLon: double
+centerLat: double
+centerLon: double
+radius: double
+ui: MainWindow
+webview: QWebEngineView
+getUI(): MainWindow
+populateForecastDownloads(): void
+toggleExpandCollapse(QModelIndex index): void
+MainWindow(): void
-on_useCOM_clicked(): void
-on_getFromMapButton_clicked(): void
-on_openFileButton_clicked(): void
-on_elevFilePath_textChanged(QString arg): void
-on_meshResType_currentIndexChanged(int index): void
-on_useDiurnalWind_clicked():void
-on_useStability_clicked():void
-on_domainAvgPicklist_currentIndexChanged(int index):void
-on_useDomainAvgWind_clicked():void
-on_treeWidget_itemDoubleClicked(QTreeWidgetItem item, int column):void
-on_usePointInit_clicked():void
-on_useWeatherModelInit_clicked():void
-on_clearDAWtable_clicked():void
-on_solveButton_clicked():void
-on_outputSaveLocationBtn_clicked():void
-on_solverPageSolveBtn_clicked():void
-on_showAllTimeZones_clicked():void
-on_displayTimeZoneDetails_clicked():void
-on_timeZoneSelector_currentIndexChanged(int index):void
-on_windTableData_cellChanged(int row, int column):void
-solveRequest():void
-timeZoneDataRequest():void
-timeZoneDetailsRequest():void
-onTreeItemClicked(QTreeWidgetItem item, int column):void
1
1
AppState
+solverMethodologyOk: string
+useCOMtoggled: object
+useCOMNtoggled: bool
+inputsOk: bool
+surfaceInputOk: object
+diurnalInputToggled: bool
+useStability: bool
+stabilityInputToggled: bool
+windInputOk: string
+domainAverageWindOk: object
+domainAverageWindInputTableOk: bool
+pointInitializationToggled: bool
+pointInitializationOk: object
+weatherModelToggled: bool
+weatherModelOk: bool
+showAllZones: bool
+displayTimeZoneDetails: bool
+solverReady: bool
+AppState(): void
1
0..*
QT <<Library>>
GDAL <<Library>>
Map.html
-center_lat: double
-center_lon: double
-radius: double
-north_lat: double
-south_lat: double
-east_lon: double
-west_lon: double
+mbr(): int[]
+fix_bounds(int[] bounds): int[]
+clear_center_radius(): void
+update_text_bounds(): void
+update_rectangle(): void
+update_center(): void
+enable_listeners(): void
+disable_listeners(): void
+ loadKMZFromUrl(string kmzUrl): void
1
1
1
1
1
1
JSZip <<Library>>
Leaflet <<Library>>
L.KML <<Library>>
1
1
1
1
1
1


<<Component>> User Interface
<<component>>
Leaflet Map Viewer
Visualizes
Simulation
Details
Gives Output
Data to User
<<Component>> Windninja Model
User
<<component>>
Main User Window
Sets Input
Parameters
Validates
Data for
Simulations
<<component>>
Controller
Package,
Prepare, and
send data to
the API
<<component>>
Model
Sets Up API
Execution
<<component>>
Windninja API
<<component>>
Validates Simulation Details
Provides
Visual
Feedback
Provides
Feedback on
Data


Simulation
Multi-core Processing
Run Simulation
<<include>>
<<include>>
Solver type
Conservation of mass and Momentum
<<extend>>
<<include>>
Conservation of Mass
Select elevation source
<<include>>
Download elevation data
Initialize Simulation
Set Wind Speed & direction
<<include>>
<<include>>
Download forecast, process forecast data
<<include>>
<<include>>
<<include>>
Set Data, time, cloud, temperature
Fetch weather data, process station data
Fetch weather data, process station data
Display Output
View Output
<<include>>
Use Google Earth
GIS System
Local file system
Elevation data server
Weather Data server
User
Generate Output
Generate Output
<<include>>
<<include>>
<<include>>
Write Casefile
Google Earth file (.kmz)
GIS shapefile (.shp)

### 6.2 Design Patterns

We utilized two unique patterns for our WindNinja GUI redesign, these were the Model-View-Controller pattern which separates the applications data, presentation, and controller code. The model for the pattern is the WindNinja simulation code, which can also come from the casefile. The View component of the MVC pattern is being handled more by QT6 and Leaflet. The window classes are associated with each view. These components handle the view and can be individually updated to better suit. They do not contain much functional code. Finally, the Controller at the center of this project is the main function. It handles the data interface between the model and the views. This pattern is particularly applicable for our redesign as it allows us to completely overhaul the user interface without impacting other functional sections of code when applied. This will allow us to work through each stage without disrupting previous work.

### 6.3 Design Tradeoffs and Decisions

One of the biggest challenges in our project involves balancing the new updates with previous usability. Additionally, while the design patterns will provide excellent maintainability, they will produce additional overhead we have to take into consideration. It would be prudent to try implementing lean interfaces between components and carefully managing unnecessary updates. Additionally, as stated previously, we must balance the modernization the interface with maintaining familiarity for its existing users. Any dramatic changes to the user interface could lead to disrupting the current users workflows. To address these challenges, we will focus on maintaining a clean, simple base interface, that will allow users to enable/disable complex features as needed. Supporting Windows, MacOS, and Linux while maintaining a consistent experience presents additional challenges, which we will handle using QT6 cross-platform software. This will allow us to focus on core functionality that works consistently across all user platforms.

## 7 Expected Results

We expect that our redesign of the WindNinja GUI project will create an interface that's much more user friendly and modern for a wide range of users. By including a built-in feature that allows users to visualize wind data without the use of external tools like ArcGIS or Google Earth within the application itself, the user experience will be elevated and modernized. The revamped GUI should be compatible with Windows, MacOS, and Linux operating systems to cater to a diverse user base. In addition, the updated interface will retain its familiarity for the more seasoned users while presenting a fresh look and simplified navigation experience. By incorporating the original WindNinja API into the redesign, we expect to make changes and introduce additional functionalities. The update will also add more flexibility for future enhancements to the GUI with strategic design and thoughtful code.

## References


[1] Joint Economic Committee. Climate-Exacerbated Wildfires Cost As Much as $893 Billion Per Year. *Top-end Annual Total Costs and Losses (Billions $)*, 2023. Available at: https://www.jec.senate.gov/public/_cache/files/9220abde-7b60-4d05-ba0a-8cc20df44c7d/jec-report-on-total-costs-of-wildfires.pdf

[2] The Qt Company. Qt Framework — One Framework to Rule All! *www.qt.io*, n.d. Available at: https://www.qt.io/product/framework

[3] National Interagency Fire Center. National Interagency Coordination Center Wildland Fire Summary and Statistics Annual Report 2023. 2023. Available at: https://www.nifc.gov/sites/default/files/NICC/2-Predictive%20Services/Intelligence/Annual%20Reports/2023/annual_report_2023_0.pdf

[4] NOAA National Centers for Environmental Information. Annual 2023 Wildfires Report. 2023. Available at: https://www.ncei.noaa.gov/access/monitoring/monthly-report/fire/202313

[5] Idaho Firewise. Fire Ecology and Management: Wildfire Ignition, Behavior, and Effects. 2023. Available at: https://idahofirewise.org/fire-ecology-and-management/wildfire-ignition-behavior-and-effects/

[6] OpenFOAM Foundation. OpenFOAM: The Open Source Computational Fluid Dynamics (CFD) Toolbox. Available at: https://openfoam.org

[7] ProjectPhysX. FluidX3D: Fast Fluid Simulation for 3D Fluids. GitHub Repository. Available at: https://github.com/ProjectPhysX/FluidX3D

[8] WindSim AS. WindSim: Simulation Software for Wind Resource Assessment and Wind Farm Design. Available at: https://windsim.com/software/

[9] Verisk Analytics. Verisk Wildfire Risk Analysis. October 2, 2017. Available at: https://www.verisk.com/resources/campaigns/location-fireline-state-risk-report/

[10] National Weather Service. Weather Prediction Center (WPC) Home Page. 2002. Available at: https://www.wpc.ncep.noaa.gov/

# 8 Appendix

## 8.1 Source Code - AppState

```cpp
#ifndef APPSTATE_H
#define APPSTATE_H

class AppState
{
public:
    static AppState& instance() {
        static AppState s;
        return s;
    }

    // Solver Methodology input states
    bool solverMethodologyOk = false;
    bool useCOMtoggled = false;
    bool useCOMMtoggled = false;

    // Input states
    bool inputsOk = false;
    bool surfaceInputOk = false;
    bool diurnalInputToggled = false;
    bool stabilityInputToggled = false;

    // Wind Input States
    bool windInputOk = false;
    bool domainAverageWindToggled = false;
    bool domainAverageWindInputTableOk = true;
    bool domainAverageWindOk = false;
    bool pointInitializationToggled = false;
    bool pointInitializationOk = false;
    bool weatherModelToggled = false;
    bool weatherModelOk = false;
    bool showAllZones = false;
    bool displayTimeZoneDetails = false;

    // All Inputs Ok
    bool solverReady = false;


private:
    AppState() {}
    AppState(const AppState&) = delete;
    AppState& operator=(const AppState&) = delete;
};

#endif // APPSTATE_H
```

### 8.2 Source Code - Controller (header)

```cpp
#ifndef CONTROLLER_H
#define CONTROLLER_H

#include <QObject>
#include <vector>
#include "appstate.h"
#include "mainwindow.h"
#include "modeldata.h"
#include "provider.h"

class Controller : public QObject {
    Q_OBJECT
public:
    Controller(MainWindow* view, QObject* parent = nullptr);

private:
  MainWindow* view;
  Provider provider;
  BaseInput setBaseInput();
  DomainAverageWind setDomainAverageWind();
  PointInitialization setPointInitialization();
  WeatherModel setWeatherModel();

  void onSolveRequest();
  void onTimeZoneDataRequest();
  void onTimeZoneDetailsRequest();
  void onGetDEMrequest(std::array<double, 4> boundsBox, QString outputFile);
  void onGetWeatherForecast();
};

#endif // CONTROLLER_H
```

### 8.3 Source Code - Controller (c++)

```cpp
#include "controller.h"
#include <string>
#include <vector>

using  namespace std;

Controller::Controller(MainWindow* view, QObject* parent)
    : QObject(parent), view(view)
{
    connect(view, &MainWindow::solveRequest, this, &Controller::onSolveRequest);
    connect(view, &MainWindow::timeZoneDataRequest, this,
&Controller::onTimeZoneDataRequest);
    connect(view, &MainWindow::timeZoneDetailsRequest, this,
&Controller::onTimeZoneDetailsRequest);
    connect(view, &MainWindow::getDEMrequest, this, &Controller::onGetDEMrequest);
    connect(view, &MainWindow::getWeatherForecast, this,
&Controller::onGetWeatherForecast);
}

// Listens for solve request; facilitates model creation and provider passing
void Controller::onSolveRequest() {
  // Alias app state, used to determine which type of solution to run
  AppState& state = AppState::instance();

  // Determine which run to perform
  if (state.domainAverageWindOk) {
    DomainAverageWind domainAvgWind = setDomainAverageWind();
    provider.domain_average_exec(domainAvgWind);
  }else if(state.pointInitializationOk){
    PointInitialization pointInit = setPointInitialization();
    provider.point_exec(pointInit);
  }else if(state.weatherModelOk){
    WeatherModel weatherModel = setWeatherModel();
    provider.wxmodel_exec(weatherModel);
  }


  vector<string> outputFileList = provider.getOutputFileNames(
    view->getUi()->elevFilePath->text(),
    view->getUi()->windTableData,
    view->getUi()->meshResValue->text(),
    provider.parseDomainAvgTable(view->getUi()->windTableData).size(),
    view->getUi()->outputDirectory->toPlainText());

  view->loadMapKMZ(outputFileList);
}

// Get time zone list from provider
void Controller::onTimeZoneDataRequest() {
  // Call provider to get 2D vector with timezone data
  bool showAllZones = view->getUi()->showAllTimeZones->isChecked();
  QVector<QVector<QString>> timeZoneData = provider.getTimeZoneData(showAllZones);

  // Clear timezone list
```

```
  view->getUi()->timeZoneSelector->clear();

  // Populate timezone list
  for (const QVector<QString>& zone : timeZoneData) {
    if (!zone.isEmpty()) {
      view->getUi()->timeZoneSelector->addItem(zone[0]);
    }
  }

  // Default to America/Denver
  view->getUi()->timeZoneSelector->setCurrentText("America/Denver");
}

// Get time zone details from provider
void Controller::onTimeZoneDetailsRequest() {
  QString currentTimeZone = view->getUi()->timeZoneSelector->currentText();
  QString timeZoneDetails = provider.getTimeZoneDetails(currentTimeZone);

  // Set value in ui
  view->getUi()->timeZoneDetails->setText(timeZoneDetails);
}

void Controller::onGetDEMrequest(std::array<double, 4> boundsBox, QString
outputFile) {

  // Get correct fetch type
  // TODO: set correct string for landscape files in else condition
  int fetchIndex = view->getUi()->fetchType->currentIndex();
  string fetchType;
  if (fetchIndex == 0) {
    fetchType = "srtm";
  } else if (fetchIndex == 1) {
    fetchType = "gmted";
  } else {
    fetchType = "land";
  }

  double resolution = view->getUi()->meshResValue->value();

  provider.fetchDEMBoundingBox(outputFile.toStdString(), fetchType, resolution,
boundsBox.data());
  view->getUi()->elevFilePath->setText(outputFile);
}

void Controller::onGetWeatherForecast() {
  string modelType = view->getUi()->weatherDataSelector-
>currentText().toStdString();
  int numNinjas = 1;
  string demFile = view->getUi()->elevFilePath->text().toStdString();

  provider.fetchForecast(modelType, numNinjas, demFile);
}

/*
```

```
 * Helper functions that construct the API input models
 */

BaseInput Controller::setBaseInput() {
  QString demPath = view->getUi()->elevFilePath->text();
  double outputResolution = view->getUi()->meshResValue->value();
  QString initMethod;
  if (view->getUi()->useDomainAvgWind->isChecked()) {
    initMethod = "domain_average";
  } else if (view->getUi()->usePointInit->isChecked()) {
    initMethod = "point";
  } else {
    initMethod = "wxmodel";
  }
  QString meshType = view->getUi()->meshResType->currentText().toLower();
  QString vegetation = view->getUi()->vegetationType->currentText().toLower();
  int nLayers = 20;
  int diurnalFlag = view->getUi()->useDiurnalWind->isChecked() ? 1 : 0;

  double height = view->getUi()->windHeightValue->value();
  QString heightUnits;
  if (view->getUi()->windHeightFeet) {
    heightUnits = "ft";
  } else {
    heightUnits = "m";
  }

  bool useMomentum = view->getUi()->useCOMM->isChecked() ? 1 : 0;
  int numNinjas = 1;
  // Count the number of ninjas, depending on the wind method being used
  QVector<QVector<QString>> domainAvgTable = provider.parseDomainAvgTable(view-
>getUi()->windTableData);

  if (view->getUi()->useDomainAvgWind->isChecked()) {
    if (domainAvgTable.size() > 0) {
      numNinjas = domainAvgTable.size();
    }
  } else if (view->getUi()->usePointInit->isChecked()) {
    //TODO
      //numNinjas = view->getUi()->pointInitStepsValue->value();
  } else {
    //Todo wxmodel
  }

  QString outputPath = view->getUi()->outputDirectory->toPlainText();

  return BaseInput (
    demPath.toStdString(),
    outputResolution,
    initMethod.toStdString(),
    meshType.toStdString(),
    vegetation.toStdString(),
    nLayers,
    diurnalFlag,
```

```
    height,
    heightUnits.toStdString(),
    useMomentum,
    numNinjas,
    outputPath.toStdString()
  );
}

DomainAverageWind Controller::setDomainAverageWind() {
  BaseInput baseInput = setBaseInput();

  // Get all wind data
  QVector<QVector<QString>> windData = provider.parseDomainAvgTable(view->getUi()-
>windTableData);

  // Get speed and direction lists
  vector<double> speedList;
  vector<double> directionList;
  for (int i = 0; i < windData.size(); i++) {
    speedList.push_back(windData[i][0].toDouble());
    directionList.push_back(windData[i][1].toDouble());
  }
  QString speedUnits = view->getUi()->speedUnits->currentText();

  return DomainAverageWind (
    baseInput,
    speedList,
    speedUnits.toStdString(),
    directionList
  );
}

PointInitialization Controller::setPointInitialization() {
  BaseInput baseInput = setBaseInput();

  vector<int> year;
  vector<int> month;
  vector<int> day ;
  vector<int> hour ;
  vector<int> minute ;
  //ToDo Understand QT setting and  receivingS
  char* station_path = "NULL";
  char* osTimeZone= view->getUi()->timeZoneSelector->currentText().toUtf8().data();;
  bool matchPointFlag = true;
  int numNinjas = baseInput.getNumNinjas();

  QDateTime startTime = QDateTime::fromString("2025-04-22 00:00", "yyyy-MM-dd
HH:mm");
  QDateTime endTime = QDateTime::fromString("2025-04-22 12:00", "yyyy-MM-dd HH:mm");

  //divides times and setsup simulations
  QList<QDateTime> ninjaTimes;

  qint64 totalSeconds = startTime.secsTo(endTime);
```

```
  qint64 step = totalSeconds / (numNinjas - 1);

  for (int i = 0; i < numNinjas; ++i) {
    QDateTime timePoint = startTime.addSecs(i * step);
    ninjaTimes.append(timePoint);
  }

  for (const QDateTime& dt : ninjaTimes) {
    QDate date = dt.date();
    QTime time = dt.time();

    year.push_back(date.year());
    month.push_back(date.month());
    day.push_back(date.day());
    hour.push_back(time.hour());
    minute.push_back(time.minute());
  }

  return PointInitialization (
      baseInput,
      year,
      month,
      day,
      hour,
      minute,
      station_path,
      osTimeZone,
      matchPointFlag
      );
}


WeatherModel Controller::setWeatherModel() {
  //Todo Implement WeatherModel
  BaseInput baseInput = setBaseInput();

  char* forecast = "NULL";
  char* osTimeZone= "NULL";

  return WeatherModel (
      baseInput,
      forecast,
      osTimeZone
      );
}
```

### 8.4 Source Code - Main (c++)

```cpp
#include "mainwindow.h"
#include "../../ninja/windninja.h"
#include <QApplication>
#include <QTimer>
#include "controller.h"


int main(int argc, char *argv[]) {

  QApplication a(argc, argv);
  MainWindow w;

  Controller controller(&w);

  // Immediately pull timezone data
  QTimer::singleShot(0, &w, &MainWindow::timeZoneDataRequest);

  w.show();
  return a.exec();
}
```

### 8.5 Source Code - MainWindow (header)

```cpp
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QFutureWatcher>
#include <QtConcurrent/QtConcurrentRun>
#include <QProgressDialog>
#include <QMainWindow>
#include <QMessageBox>
#include <QTreeWidgetItem>
#include <QtWebEngineWidgets/qwebengineview.h>
#include "ui_mainwindow.h"
#include "gdal/gdal_utils.h"
#include "gdal/gdal_priv.h"
#include <vector>
#include <string>

QT_BEGIN_NAMESPACE
namespace Ui {
class MainWindow;
}
QT_END_NAMESPACE

class MainWindow : public QMainWindow {
  Q_OBJECT

public:
  Ui::MainWindow* getUi() const { return ui; }
  void populateForecastDownloads();
  void toggleExpandCollapse(const QModelIndex &index);
  void loadMapKMZ(const std::vector<std::string>& input);

  // GDAL Values
  QString GDALDriverName, GDALDriverLongName;
  std::string GDALProjRef;
  bool hasGDALCenter;
  double GDALCenterLat;
  double GDALCenterLon;
  int GDALXSize, GDALYSize;
  double GDALCellSize, GDALNoData;
  double GDALMaxValue, GDALMinValue;


  MainWindow(QWidget *parent = nullptr);
  ~MainWindow();

private slots:
  void on_useCOM_clicked();
  void on_useCOMM_clicked();

  void on_getFromMapButton_clicked();

  void on_openFileButton_clicked();

  void on_elevFilePath_textChanged(const QString &arg1);
```

```
    void on_meshResType_currentIndexChanged(int index);

    void on_useDiurnalWind_clicked();

    void on_useStability_clicked();

    void on_domainAvgPicklist_currentIndexChanged(int index);

    void on_useDomainAvgWind_clicked();

    void on_treeWidget_itemDoubleClicked(QTreeWidgetItem *item, int column);

    void on_usePointInit_clicked();

    void on_useWeatherModelInit_clicked();

    void on_clearDAWtable_clicked();

    void on_solveButton_clicked();

    void on_outputSaveLocationBtn_clicked();

    void on_solverPageSolveBtn_clicked();

    void on_showAllTimeZones_clicked();

    void on_displayTimeZoneDetails_clicked();

    void on_timeZoneSelector_currentIndexChanged(int index);

    void on_windTableData_cellChanged(int row, int column);

    void on_meshResMeters_toggled(bool checked);

    void on_meshResFeet_toggled(bool checked);

    void on_weatherStationTimeComboBox_currentIndexChanged(int index);

    void on_downloadWeatherData_clicked();

signals:
    void solveRequest();
    void timeZoneDataRequest();
    void timeZoneDetailsRequest();
    void getDEMrequest(std::array<double, 4> boundsBox, QString outputFile);
    void getWeatherForecast();

private:
    void onTreeItemClicked(QTreeWidgetItem *item, int column);
    QSet<QPair<int, int>> invalidDAWCells;

    // DEM inputs
    double northLat;
```

```
  double southLat;
  double eastLon;
  double westLon;
  double centerLat;
  double centerLon;
  double radius;

  Ui::MainWindow *ui;
  QWebEngineView *webView;
};
#endif // MAINWINDOW_H
```

### 8.6 Source Code - MainWindow (c++)

```cpp
#include "mainwindow.h"
#include "appstate.h"
#include "./ui_mainwindow.h"
#include <QDir>
#include <QDirIterator>
#include <QDateTime>
#include <QDebug>
#include <QFileDialog>
#include <QFileInfo>
#include <QFileSystemModel>
#include <QSortFilterProxyModel>
#include <QSplitter>
#include <QStandardItemModel>
#include <QStandardPaths>
#include <QTextEdit>
#include <QTextStream>
#include <QThread>
#include <QTimer>
#include <QTreeWidget>
#include <QtWebEngineWidgets/qwebengineview.h>
#include <QWebEngineProfile>
#include <QWebEngineSettings>
#include <vector>
#include <string>

// Menu filtering class
class DirectoryFilterModel : public QSortFilterProxyModel {
protected:
  bool filterAcceptsRow(int source_row, const QModelIndex &source_parent) const
override {
    QFileSystemModel *fsModel = qobject_cast<QFileSystemModel *>(sourceModel());
    if (!fsModel) return false;

    QModelIndex index = fsModel->index(source_row, 0, source_parent);
    if (!index.isValid()) return false;

    // Define the download path
    QFileInfo fileInfo = fsModel->fileInfo(index);
    QString downloadsPath =
QStandardPaths::writableLocation(QStandardPaths::DownloadLocation);

    // Keep the Downloads root directory
    if (fileInfo.absoluteFilePath() == downloadsPath) {
      return true;
    }

    // Ensure filtering applies only inside Downloads
    if (!fileInfo.absoluteFilePath().startsWith(downloadsPath)) {
      return false;
    }

    // Allow `WXSTATIONS-*` directories
    if (fileInfo.isDir() && fileInfo.fileName().toLower().startsWith("wxstations"))
{
```

```
      return true;
    }

    // Allow files **inside** `WXSTATIONS-*`
    QModelIndex parentIndex = index.parent();
    if (parentIndex.isValid()) {
      QFileInfo parentInfo = fsModel->fileInfo(parentIndex);
      if (parentInfo.isDir() &&
parentInfo.fileName().toLower().startsWith("wxstations")) {
        return true;
      }
    }

    return false;
  }
};

MainWindow::~MainWindow() { delete ui; }

/*
 * Click tree item helper function
 */

void MainWindow::onTreeItemClicked(QTreeWidgetItem *item, int column) {
  int pageIndex = item->data(column, Qt::UserRole).toInt();
  if (pageIndex >= 0) {
    ui->stackedInputPage->setCurrentIndex(pageIndex);
  }
}

// Recursive function to add files and directories correctly with Name and Date
columns
void addFilesRecursively(QStandardItem *parentItem, const QString &dirPath) {
  QDir dir(dirPath);
  QFileInfoList entries = dir.entryInfoList(QDir::Files | QDir::Dirs |
QDir::NoDotAndDotDot);
  for (const QFileInfo &entry : entries) {
    QStandardItem *nameItem = new QStandardItem(entry.fileName());
    QStandardItem *dateItem = new QStandardItem(entry.lastModified().toString("yyyy-
MM-dd HH:mm:ss"));
    nameItem->setEditable(false);
    dateItem->setEditable(false);
    parentItem->appendRow({nameItem, dateItem});
    if (entry.isDir()) {
      addFilesRecursively(nameItem, entry.absoluteFilePath());
    }
  }
}

// Function to populate forecastDownloads with .tif parent directories and all
nested contents
void MainWindow::populateForecastDownloads() {
  QString downloadsPath =
QStandardPaths::writableLocation(QStandardPaths::DownloadLocation);
```

```
  QDir downloadsDir(downloadsPath);

  if (!downloadsDir.exists()) return;

  QStandardItemModel *model = new QStandardItemModel(this);
  model->setHorizontalHeaderLabels({"Name", "Date Modified"});

  QDirIterator it(downloadsPath, QDir::Dirs | QDir::NoDotAndDotDot);
  while (it.hasNext()) {
    QString dirPath = it.next();
    if (dirPath.endsWith(".tif", Qt::CaseInsensitive)) {
      QStandardItem *parentItem = new QStandardItem(QFileInfo(dirPath).fileName());
      parentItem->setEditable(false);
      addFilesRecursively(parentItem, dirPath);
      model->appendRow(parentItem);
    }
  }

  ui->forecastDownloads->setModel(model);
  ui->forecastDownloads->header()->setSectionResizeMode(QHeaderView::Stretch);

  // Disable editing and enable double-click expansion
  ui->forecastDownloads->setEditTriggers(QAbstractItemView::NoEditTriggers);
  ui->forecastDownloads->setExpandsOnDoubleClick(true);
}

/*
 * Dynamic UI handling
 */

/*
 * Helper function to refresh the ui state of the app
 * Called on every user input action
 */
static void refreshUI(const Ui::MainWindow* ui)
{
  // Alias the AppState
  AppState& state = AppState::instance();

  // Define state icons
  QIcon tickIcon(":/tick.png");
  QIcon xIcon(":/cross.png");
  QIcon bulletIcon(":/bullet_blue.png");

  // Enable mouse tracking on tree
  ui->treeWidget->setMouseTracking(true);

  // Update Solver Methodology UI
  if (state.useCOMtoggled != state.useCOMMtoggled) {
    state.solverMethodologyOk = true;
    ui->treeWidget->topLevelItem(0)->setIcon(0, tickIcon);
    ui->treeWidget->topLevelItem(0)->setToolTip(0, "");
  } else if (state.useCOMtoggled && state.useCOMMtoggled) {
    state.solverMethodologyOk = false;
```

```
    ui->treeWidget->topLevelItem(0)->setIcon(0, xIcon);
    ui->treeWidget->topLevelItem(0)->setToolTip(0,"Requires exactly one selection:
currently too many selections.");
  } else {
    state.solverMethodologyOk = false;
    ui->treeWidget->topLevelItem(0)->setIcon(0, xIcon);
    ui->treeWidget->topLevelItem(0)->setToolTip(0,"Requires exactly one selection:
currently no selections.");
  }

  if (state.useCOMtoggled) {
    ui->treeWidget->topLevelItem(0)->child(0)->setIcon(0, tickIcon);
  } else {
    ui->treeWidget->topLevelItem(0)->child(0)->setIcon(0, bulletIcon);
  }

  if (state.useCOMMtoggled) {
    ui->treeWidget->topLevelItem(0)->child(1)->setIcon(0, tickIcon);
  } else {
    ui->treeWidget->topLevelItem(0)->child(1)->setIcon(0, bulletIcon);
  }

  /*
   * Primary state machine for inputs (surface, wind, etc.)
   */

  // Update surface input state
  if (ui->elevFilePath->text() != "") {
    state.surfaceInputOk = true;
    ui->treeWidget->topLevelItem(1)->child(0)->setIcon(0, tickIcon);
    ui->treeWidget->topLevelItem(1)->child(0)->setToolTip(0, "");
  } else {
    state.surfaceInputOk = false;
    ui->treeWidget->topLevelItem(1)->child(0)->setIcon(0, xIcon);
    ui->treeWidget->topLevelItem(1)->child(0)->setToolTip(0, "No DEM file
detected.");
  }

  // Update diurnal input state
  if (state.diurnalInputToggled) {
    ui->treeWidget->topLevelItem(1)->child(1)->setIcon(0, tickIcon);
  } else {
    ui->treeWidget->topLevelItem(1)->child(1)->setIcon(0, bulletIcon);
  }

  // Update stability input state
  if (state.stabilityInputToggled) {
    ui->treeWidget->topLevelItem(1)->child(2)->setIcon(0, tickIcon);
  } else {
    ui->treeWidget->topLevelItem(1)->child(2)->setIcon(0, bulletIcon);
  }

  // Update domain average wind state
  if (state.domainAverageWindToggled && state.domainAverageWindInputTableOk) {
```

```
    ui->treeWidget->topLevelItem(1)->child(3)->child(0)->setIcon(0, tickIcon);
    ui->treeWidget->topLevelItem(1)->child(3)->child(0)->setToolTip(0, "");
    state.domainAverageWindOk = true;
  } else if (state.domainAverageWindToggled && !state.domainAverageWindInputTableOk)
{
    ui->treeWidget->topLevelItem(1)->child(3)->child(0)->setIcon(0, xIcon);
    ui->treeWidget->topLevelItem(1)->child(3)->child(0)->setToolTip(0, "Bad wind
inputs; hover over red cells for explanation.");
    state.domainAverageWindOk = false;
  } else {
    ui->treeWidget->topLevelItem(1)->child(3)->child(0)->setIcon(0, bulletIcon);
    ui->treeWidget->topLevelItem(1)->child(3)->child(0)->setToolTip(0, "");
    state.domainAverageWindOk = false;
  }

  // Update point initialization state
  if (state.pointInitializationToggled) {
    ui->treeWidget->topLevelItem(1)->child(3)->child(1)->setIcon(0, tickIcon);
    state.pointInitializationOk = true;
  } else {
    ui->treeWidget->topLevelItem(1)->child(3)->child(1)->setIcon(0, bulletIcon);
    state.pointInitializationOk = false;
  }

  // Update weather model state
  if (state.weatherModelToggled) {
    ui->treeWidget->topLevelItem(1)->child(3)->child(2)->setIcon(0, tickIcon);
    state.weatherModelOk = true;
  } else {
    ui->treeWidget->topLevelItem(1)->child(3)->child(2)->setIcon(0, bulletIcon);
    state.weatherModelOk = false;
  }

  //  Update wind input
  if (state.domainAverageWindOk || state.pointInitializationOk ||
state.weatherModelOk) {
    ui->treeWidget->topLevelItem(1)->child(3)->setIcon(0, tickIcon);
    state.windInputOk = true;
  } else {
    ui->treeWidget->topLevelItem(1)->child(3)->setIcon(0, xIcon);
    state.windInputOk = false;
  }

  // Update overall input UI state
  if (state.surfaceInputOk && state.windInputOk) {
    state.inputsOk = true;
    ui->treeWidget->topLevelItem(1)->setIcon(0, tickIcon);
    ui->treeWidget->topLevelItem(1)->setToolTip(0, "");
  } else if (!state.surfaceInputOk && !state.windInputOk) {
    state.inputsOk = false;
    ui->treeWidget->topLevelItem(1)->setIcon(0, xIcon);
    ui->treeWidget->topLevelItem(1)->setToolTip(0, "Bad surface and wind inputs.");
  } else if (!state.surfaceInputOk) {
    state.inputsOk = false;
```

```
    ui->treeWidget->topLevelItem(1)->setIcon(0, xIcon);
    ui->treeWidget->topLevelItem(1)->setToolTip(0, "Bad surface input.");
  } else if (!state.windInputOk) {
    state.inputsOk = false;
    ui->treeWidget->topLevelItem(1)->setIcon(0, xIcon);
    ui->treeWidget->topLevelItem(1)->setToolTip(0, "Bad wind input.");
  }

  // Update solve state
  if (state.solverMethodologyOk && state.inputsOk) {
    ui->solveButton->setEnabled(true);
    ui->solverPageSolveBtn->setEnabled(true);
    ui->solveButton->setToolTip("");
    ui->solverPageSolveBtn->setToolTip("");
  } else {
    ui->solveButton->setEnabled(false);
    ui->solverPageSolveBtn->setEnabled(false);
    ui->solveButton->setToolTip("Solver Methodology and Inputs must be passing to
solve.");
    ui->solverPageSolveBtn->setToolTip("Solver Methodology and Inputs must be
passing to solve.");
  }
}


/*
 * Signal and slot handlers
 */

// Use selects Conservation of Mass
void MainWindow::on_useCOM_clicked()
{
  AppState& state = AppState::instance();

  // Only allow CoM or CoMM to be toggled
  if (state.useCOMMtoggled) {
    ui->useCOMM->setChecked(false);
    state.useCOMMtoggled = ui->useCOMM->isChecked();
  }

  // Update app states
  state.useCOMtoggled = ui->useCOM->isChecked();
  refreshUI(ui);
}


// User selects Conservation of Mass and Momentum
void MainWindow::on_useCOMM_clicked()
{
  AppState& state = AppState::instance();

  // Only allow CoM or CoMM to be toggled
  if (state.useCOMtoggled) {
    ui->useCOM->setChecked(false);
```

```
    state.useCOMtoggled = ui->useCOM->isChecked();
  }

  // Update app states
  state.useCOMMtoggled = ui->useCOMM->isChecked();
  refreshUI(ui);
}


// User selects an elevation input file (by file)
void MainWindow::on_elevFilePath_textChanged(const QString &arg1)
{
  // Get GDAL data information on DEM input
  QString fileName = ui->elevFilePath->text();
  double adfGeoTransform[6];
  GDALDataset *poInputDS;
  poInputDS = (GDALDataset*)GDALOpen(fileName.toStdString().c_str(), GA_ReadOnly);

  // Set driver info
  GDALDriverName = poInputDS->GetDriver()->GetDescription();
  GDALDriverLongName = poInputDS->GetDriver()->GetMetadataItem(GDAL_DMD_LONGNAME);

  // get x and y dimensions
  GDALXSize = poInputDS->GetRasterXSize();
  GDALYSize = poInputDS->GetRasterYSize();

  // Calculate cell size
  if (poInputDS->GetGeoTransform(adfGeoTransform) == CE_None) {
    int c1, c2;
    c1 = adfGeoTransform[1];
    c2 = adfGeoTransform[5];
    if (abs(c1) == abs(c2)) {
      GDALCellSize = abs(c1);
    } else {
      GDALClose((GDALDatasetH)poInputDS);
    }
  }

  // Get GDAL min/max values
  GDALRasterBand* band = poInputDS->GetRasterBand(1);
  int gotMin = 0, gotMax = 0;
  double minVal = band->GetMinimum(&gotMin);
  double maxVal = band->GetMaximum(&gotMax);

  if (!gotMin || !gotMax) {
    band->ComputeStatistics(false, &minVal, &maxVal, nullptr, nullptr, nullptr,
nullptr);
  }

  GDALMinValue = minVal;
  GDALMaxValue = maxVal;

  // Close
  GDALClose((GDALDatasetH)poInputDS);
```

```
  // Run mesh calculator
  MainWindow::on_meshResType_currentIndexChanged(ui->meshResType->currentIndex());

  refreshUI(ui);
}


void MainWindow::on_openFileButton_clicked()
{
  QString downloadsPath =
QStandardPaths::writableLocation(QStandardPaths::DownloadLocation);
  QString filePath = QFileDialog::getOpenFileName(this,
                                                   "Select a file",                //
Window title
                                                   downloadsPath,                //
Starting directory
                                                   "(*.tif);;All Files (*)"  //
Filter
                                                   );
  ui->elevFilePath->setText(filePath);
  ui->elevFilePath->setToolTip(filePath);
}


// User selects an elevation input file (by map import)
void MainWindow::on_getFromMapButton_clicked()
{
  // We have to use batching since the Javascript part is async
  struct JSFieldBatch {
    QMap<QString, QVariant> results;
    int expected = 7;
    std::function<void(QMap<QString, QVariant>)> onReady;
  };

  auto batch = new JSFieldBatch();
  batch->onReady = [this, batch](QMap<QString, QVariant> fields) {
    northLat = fields["north_lat"].toDouble();
    southLat = fields["south_lat"].toDouble();
    eastLon  = fields["east_lon"].toDouble();
    westLon  = fields["west_lon"].toDouble();
    centerLat = fields["center_lat"].toDouble();
    centerLon = fields["center_lon"].toDouble();
    radius = fields["radius"].toDouble();

    delete batch;  // clean up
  };

  auto run = [this, batch](QString fieldId) {
    webView->page()-
>runJavaScript(QString("document.getElementById('%1').value;").arg(fieldId),
                                    [fieldId, batch](const QVariant &result) {
                                      batch->results[fieldId] = result;
                                      if (batch->results.size() == batch->expected) {
```

```
                                        batch->onReady(batch->results);
                                    }
                                });
    };

    // Kick off all the field reads
    run("north_lat");
    run("south_lat");
    run("east_lon");
    run("west_lon");
    run("center_lat");
    run("center_lon");
    run("radius");

            // Verify input validity and write file
    if (northLat != 0 && southLat != 0 && eastLon != 0 && westLon != 0) {
      QString defaultName = "demDownload.tif";
      QString filter = "TIF Files (*.tif)";

              // Get downloads path and join to filename
      QString downloadsPath =
QStandardPaths::writableLocation(QStandardPaths::DownloadLocation);
      QDir dir(downloadsPath);
      QString fullPath = dir.filePath(defaultName);

              // Open save window
      QString fileName = QFileDialog::getSaveFileName(this,
                                                      "Save DEM File",
                                                      fullPath,
                                                      filter);

      if (fileName != "") {
        std::array<double, 4> coordsCopy = { northLat, eastLon, southLat, westLon };
        QString fileNameCopy = fileName;

        QtConcurrent::run([coordsCopy, fileNameCopy, this]() {
          emit getDEMrequest(coordsCopy, fileNameCopy);
        });
      }
    }
}

    // User changes the mesh resolution spec for surface input
void MainWindow::on_meshResType_currentIndexChanged(int index)
{
    // Set value box enable for custom/other
    if (index == 3) {
      ui->meshResValue->setEnabled(true);
    } else {
      ui->meshResValue->setEnabled(false);
    }

    int coarse = 4000;
    int medium = 10000;
```

```
  int fine = 20000;
  double meshResolution = 200.0;

  int targetNumHorizCells = fine;
  switch (index) {
  case 0:
    targetNumHorizCells = coarse;
    break;
  case 1:
    targetNumHorizCells = medium;
    break;
  case 2:
    targetNumHorizCells = fine;
    break;
  case 3:
    targetNumHorizCells = meshResolution;
  }

  double XLength = GDALXSize * GDALCellSize;
  double YLength = GDALYSize * GDALCellSize;
  double nXcells = 2 * std::sqrt((double)targetNumHorizCells) * (XLength / (XLength
+ YLength));
  double nYcells = 2 * std::sqrt((double)targetNumHorizCells) * (YLength / (XLength
+ YLength));

  double XCellSize = XLength / nXcells;
  double YCellSize = YLength / nYcells;

  meshResolution = (XCellSize + YCellSize) / 2;

  ui->meshResValue->setValue(meshResolution);

}

void MainWindow::on_meshResMeters_toggled(bool checked)
{
  if (checked) {
    ui->meshResValue->setValue(ui->meshResValue->value() * 0.3048);
  }
}

void MainWindow::on_meshResFeet_toggled(bool checked)
{
  if (checked) {
    ui->meshResValue->setValue(ui->meshResValue->value() * 3.28084);
  }
}

// User selects a new time zone
void MainWindow::on_timeZoneSelector_currentIndexChanged(int index)
{
  emit timeZoneDetailsRequest();
}
```

```
// User toggles show all time zones
void MainWindow::on_showAllTimeZones_clicked()
{
  AppState& state = AppState::instance();

  // Update show all zones state
  state.showAllZones = ui->showAllTimeZones->isChecked();

  emit timeZoneDataRequest();
}

// User toggles show time zone details
void MainWindow::on_displayTimeZoneDetails_clicked()
{
  AppState& state = AppState::instance();

  // Update time zone details state
  state.displayTimeZoneDetails = ui->displayTimeZoneDetails->isChecked();

  // Update visibility of details pane
  ui->timeZoneDetails->setVisible(state.displayTimeZoneDetails);

}

// User selects Diurnal Input
void MainWindow::on_useDiurnalWind_clicked()
{
  AppState& state = AppState::instance();

  // Update UI state
  state.diurnalInputToggled = ui->useDiurnalWind->isChecked();

  // Change the domain average input table based on diurnal wind
  QTableWidget* table = ui->windTableData;
  if (!ui->useDiurnalWind->isChecked()) {
    table->hideColumn(2);
    table->hideColumn(3);
    table->hideColumn(4);
    table->hideColumn(5);
    ui->windTableData->horizontalHeader()-
>setSectionResizeMode(QHeaderView::Stretch);
  } else {
    table->showColumn(2);
    table->showColumn(3);
    table->showColumn(4);
    table->showColumn(5);
    ui->windTableData->horizontalHeader()-
>setSectionResizeMode(QHeaderView::Stretch);
  }

  refreshUI(ui);
}

// User selects Stability Input
```

```cpp
void MainWindow::on_useStability_clicked()
{
  AppState& state = AppState::instance();

  // Update UI state
  state.stabilityInputToggled = ui->useStability->isChecked();
  refreshUI(ui);
}

/*
 * Wind Inputs
 */

// Domain Average Wind

// User selects Domain Average Wind
void MainWindow::on_useDomainAvgWind_clicked()
{
  AppState& state = AppState::instance();

  // Update the domain average wind state
  state.domainAverageWindToggled = ui->useDomainAvgWind->isChecked();

  // Only allow one wind methodology to be used
  if (state.domainAverageWindToggled) {
    ui->usePointInit->setChecked(false);
    ui->useWeatherModelInit->setChecked(false);
    state.pointInitializationToggled = ui->usePointInit->isChecked();
    state.weatherModelToggled = ui->useWeatherModelInit->isChecked();
  }

  // Update app state
  refreshUI(ui);
}

// User changes Domain Average Wind -> Input Wind Height
void MainWindow::on_domainAvgPicklist_currentIndexChanged(int index)
{
  switch(index) {
  case 0:
    ui->windHeightValue->setValue(20.00);
    ui->windHeightValue->setEnabled(false);
    ui->windHeightFeet->setChecked(true);
    ui->windHeightFeet->setEnabled(false);
    ui->windHeightMeters->setEnabled(false);
    break;

  case 1:
    ui->windHeightValue->setValue(10.00);
    ui->windHeightValue->setEnabled(false);
    ui->windHeightMeters->setChecked(true);
    ui->windHeightFeet->setEnabled(false);
    ui->windHeightMeters->setEnabled(false);
    break;
```

```
    case 2:
      ui->windHeightValue->setEnabled(true);
      ui->windHeightFeet->setEnabled(true);
      ui->windHeightMeters->setEnabled(true);
      break;
    }
}

// User clears the domain average wind input table
void MainWindow::on_clearDAWtable_clicked()
{
    ui->windTableData->clearContents();
    invalidDAWCells.clear();
    AppState::instance().domainAverageWindInputTableOk = true;
    refreshUI(ui);
}

// User changes a value in the domain average wind input table
void MainWindow::on_windTableData_cellChanged(int row, int column)
{
    QTableWidget* table = ui->windTableData;
    QTableWidgetItem* item = table->item(row, column);
    if (!item) return;

    QString value = item->text().trimmed();
    bool valid = false;
    QString errorMessage;

    // Allow empty input
    if (value.isEmpty()) {
        valid = true;
    } else {
        switch (column) {
        case 0: {
            double d = value.toDouble(&valid);
            if (!valid || d <= 0)
                valid = false;
                errorMessage = "Must be a positive number";
            break;
        }
        case 1: {
            int i = value.toDouble(&valid);
            if (!valid || i < 0 || i > 359.9) {
                valid = false;
                errorMessage = "Must be a number between 0 and 359";
            }
            break;
        }
        case 2: {
            QTime t = QTime::fromString(value, "hh:mm");
            valid = t.isValid();
            if (!valid) errorMessage = "Must be a valid 24h time (hh:mm)";
            break;
```

```
    }
    case 3: {
      QDate d = QDate::fromString(value, "MM/dd/yyyy");
      valid = d.isValid();
      if (!valid) errorMessage = "Must be a valid date (MM/DD/YYYY)";
      break;
    }
    case 4: {
      int i = value.toDouble(&valid);
      if (!valid || i < 0 || i > 100) {
        valid = false;
        errorMessage = "Must be a number between 0 and 100";
      }
      break;
    }
    case 5: {
      value.toInt(&valid);
      if (!valid) errorMessage = "Must be an integer";
      break;
    }
    default:
      valid = true;
    }
  }

  QPair<int, int> cell(row, column);
  if (!valid) {
    invalidDAWCells.insert(cell);
    item->setBackground(Qt::red);
    item->setToolTip(errorMessage);
  } else {
    invalidDAWCells.remove(cell);
    item->setBackground(QBrush());  // Reset to default
    item->setToolTip("");
  }

  AppState::instance().domainAverageWindInputTableOk = invalidDAWCells.isEmpty();
  refreshUI(ui);
}

// User selects Point Initialization wind model
void MainWindow::on_usePointInit_clicked()
{
  AppState& state = AppState::instance();

  // Update the domain average wind state
  state.pointInitializationToggled = ui->usePointInit->isChecked();

  // Only allow one wind methodology to be used
  if (state.pointInitializationToggled) {
    ui->useDomainAvgWind->setChecked(false);
    ui->useWeatherModelInit->setChecked(false);
    state.domainAverageWindToggled = ui->useDomainAvgWind->isChecked();
    state.weatherModelToggled = ui->useWeatherModelInit->isChecked();
```

```
  }

  // Update app state
  refreshUI(ui);
}

// User selects point init download type
void MainWindow::on_weatherStationTimeComboBox_currentIndexChanged(int index)
{
  if (index == 0) {
    ui->downloadTimeData->setCurrentIndex(0);
  } else {
    ui->downloadTimeData->setCurrentIndex(1);
  }
}

// User selects Weather Model Initialization model
void MainWindow::on_useWeatherModelInit_clicked()
{
  AppState& state = AppState::instance();

  // Update the domain average wind state
  state.weatherModelToggled = ui->useWeatherModelInit->isChecked();

  // Only allow one wind methodology to be used
  if (state.weatherModelToggled) {
    ui->useDomainAvgWind->setChecked(false);
    ui->usePointInit->setChecked(false);
    state.domainAverageWindToggled = ui->useDomainAvgWind->isChecked();
    state.pointInitializationToggled = ui->usePointInit->isChecked();
  }

  // Update app state
  refreshUI(ui);
}

// User downloads weather forecast
void MainWindow::on_downloadWeatherData_clicked()
{
  if (ui->elevFilePath->text() != "") {

  }
}

// User selects a new output location
void MainWindow::on_outputSaveLocationBtn_clicked()
{
  QString currentPath = ui->outputDirectory->toPlainText();
  QString downloadsPath =
QStandardPaths::writableLocation(QStandardPaths::DownloadLocation);
  QString dirPath = QFileDialog::getExistingDirectory(this,
                                                      "Select a directory",  //
Window title
                                                      currentPath,         //
```

```
Starting location
                                                    QFileDialog::ShowDirsOnly |
QFileDialog::DontResolveSymlinks);

  if (!dirPath.isEmpty()) {
    ui->outputDirectory->setText(dirPath);
    ui->outputDirectory->setToolTip(dirPath);
  }
}

// User selects the solve button on the solver page
void MainWindow::on_solverPageSolveBtn_clicked()
{
  ui->solveButton->click();
}

// User selects the primary solve button
void MainWindow::on_solveButton_clicked()
{
  emit solveRequest();
}

// Enable double clicking on tree menu items
void MainWindow::on_treeWidget_itemDoubleClicked(QTreeWidgetItem *item, int column)
{
  if (item->text(0) == "Conservation of Mass") {
    ui->useCOM->click();
  } else if (item->text(0) == "Conservation of Mass and Momentum") {
    ui->useCOMM->click();
  } else if (item->text(0) == "Diurnal Input") {
    ui->useDiurnalWind->click();
  } else if (item->text(0) == "Stability Input") {
    ui->useStability->click();
  } else if (item->text(0) == "Domain Average Wind") {
    ui->useDomainAvgWind->click();
  } else if (item->text(0) == "Point Initialization") {
    ui->usePointInit->click();
  } else if (item->text(0) == "Weather Model") {
    ui->useWeatherModelInit->click();
  }
}

void MainWindow::loadMapKMZ(const std::vector<std::string>&  input){

  for (const auto& dir : input) {
    QString qDir = QString::fromStdString(dir).replace("\"", "\\\""); // escape
quotes
    QString jsCall = QString("loadKmzFromUrl(\"%1\");").arg(qDir);

    webView->page()->runJavaScript(jsCall);
  }


}
```

```cpp
MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent),
      ui(new Ui::MainWindow)
{
  ui->setupUi(this);

  // Set default window size
  resize(1200, 700);

  // Immediately call a UI refresh to set initial states
  refreshUI(ui);
  // Expand tree UI
  ui->treeWidget->expandAll();

  // Register GDAL drivers
  GDALAllRegister();

  /*
   * Create file handler window for point init screen
   */

  // Get the correct Downloads folder path
  QString downloadsPath =
QStandardPaths::writableLocation(QStandardPaths::DownloadLocation);

  // Enable QFileSystemModel to process directories and files
  QFileSystemModel *model = new QFileSystemModel(this);
  model->setFilter(QDir::NoDotAndDotDot | QDir::AllDirs | QDir::AllEntries);  //
Ensure files appear
  model->setRootPath(downloadsPath);

  // Enable file watching so contents refresh properly
  model->setReadOnly(false);
  model->setResolveSymlinks(true);

  // Create a filtering model
  DirectoryFilterModel *filterModel = new DirectoryFilterModel();
  filterModel->setSourceModel(model);

  // Set the correct root index inside Downloads
  QModelIndex rootIndex = model->index(downloadsPath);
  ui->treeFileExplorer->setModel(filterModel);
  ui->treeFileExplorer->setRootIndex(filterModel->mapFromSource(rootIndex));

  // Ensure folders expand and collapse correctly
  ui->treeFileExplorer->setExpandsOnDoubleClick(true);
  ui->treeFileExplorer->setAnimated(true);
  ui->treeFileExplorer->setIndentation(15);
  ui->treeFileExplorer->setSortingEnabled(true);
  ui->treeFileExplorer->setItemsExpandable(true);
  ui->treeFileExplorer->setUniformRowHeights(true);

  // Show only "Name" and "Date Modified" columns
```

```
  ui->treeFileExplorer->hideColumn(1);  // Hide Size column
  ui->treeFileExplorer->hideColumn(2);  // Hide Type column

  // Optional: Set column headers
  QHeaderView *header = ui->treeFileExplorer->header();
  header->setSectionResizeMode(0, QHeaderView::Interactive);  // Name fits content
  header->setSectionResizeMode(3, QHeaderView::Stretch);           // Date Modified
stretches
  model->setHeaderData(0, Qt::Horizontal, "Name");
  model->setHeaderData(3, Qt::Horizontal, "Date Modified");

  // Force model to reload children
  ui->treeFileExplorer->expandAll();  // Force expand all to check visibility

  /*
   * Functionality for the map widget
   */

  // Enable remote content
  QWebEngineProfile::defaultProfile()->settings()-
>setAttribute(QWebEngineSettings::LocalContentCanAccessRemoteUrls, true);
  QWebEngineProfile::defaultProfile()->settings()-
>setAttribute(QWebEngineSettings::LocalContentCanAccessFileUrls, true);

  // Resolve the map file path
  QString filePath = QString(MAP_PATH);

  //Load HTML file with Leaflet
  webView = new QWebEngineView(ui->mapPanelWidget);
  QUrl url = QUrl::fromLocalFile(filePath);
  webView->setUrl(url);

  // Set up layout
  QVBoxLayout *layout = new QVBoxLayout();
  layout->setContentsMargins(0, 0, 0, 0);
  layout->addWidget(webView);

  // Apply
  ui->mapPanelWidget->setLayout(layout);

  /*
   * Connect tree items to stacked tab window
   */

  // Top-level items
  ui->stackedInputPage->setCurrentIndex(0);
  ui->treeWidget->topLevelItem(0)->setData(0, Qt::UserRole, 1);  // Solver
Methodology (Page 0)
  ui->treeWidget->topLevelItem(1)->setData(0, Qt::UserRole, 4);  // Inputs (Page 5)
  ui->treeWidget->topLevelItem(2)->setData(0, Qt::UserRole, 12); // Inputs (Page 13)

  // Sub-items for Solver Methodology
  ui->treeWidget->topLevelItem(0)->child(0)->setData(0, Qt::UserRole, 2);  //
Conservation of Mass (Page 1)
```

```
  ui->treeWidget->topLevelItem(0)->child(1)->setData(0, Qt::UserRole, 3);  //
Conservation of Mass and Momentum (Page 2)

  // Sub-items for Inputs
  ui->treeWidget->topLevelItem(1)->child(0)->setData(0, Qt::UserRole, 5);  //
Surface Input (Page 6)
  ui->treeWidget->topLevelItem(1)->child(1)->setData(0, Qt::UserRole, 6);  //
Dirunal Input (Page 7)
  ui->treeWidget->topLevelItem(1)->child(2)->setData(0, Qt::UserRole, 7);  //
Stability Input (Page 8)
  ui->treeWidget->topLevelItem(1)->child(3)->setData(0, Qt::UserRole, 8);  // Wind
Input (Page 9)

  // Sub-sub-items for Wind Input
  QTreeWidgetItem *windInputItem = ui->treeWidget->topLevelItem(1)->child(3);
  windInputItem->child(0)->setData(0, Qt::UserRole, 9);  // Domain Average Wind
(Page 9)
  windInputItem->child(1)->setData(0, Qt::UserRole, 10); // Point Init (Page 10)
  windInputItem->child(2)->setData(0, Qt::UserRole, 11); // Weather Model (Page 11)

  connect(ui->treeWidget, &QTreeWidget::itemClicked, this,
&MainWindow::onTreeItemClicked);

  /*
   * Downloaded Forecast explorer
   */

  populateForecastDownloads();

  /*
   * Basic initial setup steps
   */

  // Surface Input window
  // Set icons
  ui->openFileButton->setIcon(QIcon(":/folder.png"));
  ui->getFromMapButton->setIcon(QIcon(":/swoop_final.png"));

  // Solver window
  // Update processor count and set user input default value & upper bound
  int cpuCount = QThread::idealThreadCount();
  ui->availableProcessors->setPlainText("Available Processors:  " +
QString::number(cpuCount));
  ui->numProcessorsSpinbox->setMaximum(cpuCount);
  ui->numProcessorsSpinbox->setValue(cpuCount);

  // Wind Input -> Point Init window
  ui->downloadPointInitData->setIcon(QIcon(":/application_get"));

  // Populate default location for output location
  ui->outputDirectory->setText(downloadsPath);
  ui->outputSaveLocationBtn->setIcon(QIcon(":/folder.png"));

  // Set initial visibility of time zone details
```

```
  ui->timeZoneDetails->setVisible(false);

  // Set initial formatting of domain average input table
    ui->windTableData->hideColumn(2);
    ui->windTableData->hideColumn(3);
    ui->windTableData->hideColumn(4);
    ui->windTableData->hideColumn(5);
    ui->windTableData->horizontalHeader()-
>setSectionResizeMode(QHeaderView::Stretch);

}
```

### 8.7 Source Code - Map (html)

```
<!DOCTYPE html>
<html>
<meta charset="UTF-8">
<head>
 <style>
 body {
   padding: 0 ;
   margin:  0 ;
 }
 html, body {
    height: 100% ;
    width:  100% ;
    margin: 0;
    padding: 0;
    overflow: hidden;
    box-sizing: border-box;
 }
 #map {
    height: 70% ;
    width: 100% ;
    max-width: 100%;
    box-sizing: border-box;
 }
 #data_entry {
    height: 30%;
    width: 100%;
    max-width: 100%;
    box-sizing: border-box;
    overflow: auto;
 }
 #data_entry form {
    box-sizing: border-box;
    width: 100%;
    height: 50%;
    padding: 0;
 }
 .option_hdr {
   font-size: 0.9em ;
 }
 .option_hdr2 {
   font-size: 0.7em ;
 }
 label, input {
   font-size: 0.8em;
 }
 #helpBlock {
       font-size: x-small;
         display: block;
 }
 .input-help {
       vertical-align: top;
         display: inline-block;
 }
 .adjust-line-height {
```

```
        line-height: 0em;
 }


 </style>
    <title>Leaflet.draw drawing and editing tools</title>

    <link rel="stylesheet" href="leaflet/leaflet.css"/>
    <link rel="stylesheet" href="leaflet/draw/src/leaflet.draw.css"/>

    <script src="https://cdnjs.cloudflare.com/ajax/libs/jszip/3.10.1/
jszip.min.js"></script>


    <!-- TODO(kyle): prune dependencies -->
    <script src="./leaflet/leaflet-src.js"></script>
    <script src='./leaflet/L.KML.js'></script>
    <script src="./leaflet/jszip.min.js"></script>

    <script src="./leaflet/draw/src/Leaflet.draw.js"></script>
    <script src="./leaflet/draw/src/Leaflet.Draw.Event.js"></script>

    <script src="./leaflet/draw/src/edit/handler/Edit.Poly.js"></script>
    <script src="./leaflet/draw/src/edit/handler/Edit.SimpleShape.js"></script>
    <script src="./leaflet/draw/src/edit/handler/Edit.Rectangle.js"></script>
    <script src="./leaflet/draw/src/edit/handler/Edit.Marker.js"></script>
    <script src="./leaflet/draw/src/edit/handler/Edit.CircleMarker.js"></script>
    <script src="./leaflet/draw/src/edit/handler/Edit.Circle.js"></script>

    <script src="./leaflet/draw/src/draw/handler/Draw.Feature.js"></script>
    <script src="./leaflet/draw/src/draw/handler/Draw.Polyline.js"></script>
    <script src="./leaflet/draw/src/draw/handler/Draw.Polygon.js"></script>
    <script src="./leaflet/draw/src/draw/handler/Draw.SimpleShape.js"></script>
    <script src="./leaflet/draw/src/draw/handler/Draw.Rectangle.js"></script>
    <script src="./leaflet/draw/src/draw/handler/Draw.Circle.js"></script>
    <script src="./leaflet/draw/src/draw/handler/Draw.Marker.js"></script>
    <script src="./leaflet/draw/src/draw/handler/Draw.CircleMarker.js"></script>

    <script src="./leaflet/draw/src/ext/TouchEvents.js"></script>
    <script src="./leaflet/draw/src/ext/LatLngUtil.js"></script>
    <script src="./leaflet/draw/src/ext/GeometryUtil.js"></script>
    <script src="./leaflet/draw/src/ext/LineUtil.Intersect.js"></script>
    <script src="./leaflet/draw/src/ext/Polyline.Intersect.js"></script>
    <script src="./leaflet/draw/src/ext/Polygon.Intersect.js"></script>

    <script src="./leaflet/draw/src/Control.Draw.js"></script>
    <script src="./leaflet/draw/src/Tooltip.js"></script>
    <script src="./leaflet/draw/src/Toolbar.js"></script>

    <script src="./leaflet/draw/src/draw/DrawToolbar.js"></script>
    <script src="./leaflet/draw/src/edit/EditToolbar.js"></script>
    <script src="./leaflet/draw/src/edit/handler/EditToolbar.Edit.js"></script>
    <script src="./leaflet/draw/src/edit/handler/EditToolbar.Delete.js"></script>
</head>
```

```
<body>
 <div id="map" style="border: 1px solid #ccc"></div>
<script>
    // Get a valid key from our server
    var apiKey = "";
    function getKey() {
     // don't call this
     return
     xhr = new XMLHttpRequest();
     var url = "https://windninja.org/mapkey/";
     xhr.open("GET", url, true);
     xhr.onreadystatechange = function () {
     // readyState == HEADER_RECEIVED or DONE is fine (2 or 4)
     // We need to read the return on the api call, and report a valid
     // error.  These are worthless.
      if (xhr.readyState == 4 && xhr.status == 200) {
       apiKey = JSON.parse(xhr.responseText).key;
       alert(apiKey);
      } else {
       console.log("failed");
      }
     }
     xhr.send(null);
    }
    getKey();

    // invalidate the key, use this static one for now.
    apiKey =
"pk.eyJ1IjoibWFwYm94IiwiYSI6ImNpejY4NXVycTA2emYycXBndHRqcmZ3N3gifQ.rJcFIG214AriISLbB
6B5aw";

    mbAtt = 'Map data &copy; <a href="https://
www.openstreetmap.org/">OpenStreetMap</a> contributors, ' +
     '<a href="https://creativecommons.org/licenses/by-sa/2.0/">CC-BY-SA</a>, ' +
     'Imagery © <a href="https://www.mapbox.com/">Mapbox</a>'


    mbl = L.tileLayer('https://api.mapbox.com/styles/v1/{id}/tiles/{z}/{x}/{y}?
access_token=pk.eyJ1IjoiYm5vcmRncmVuIiwiYSI6ImNsZmxuMHowZzAzaTczeG80ZXR3a2ZnNHEifQ.k
c7P57DJg8tyDMjjP7czuQ', {
     tileSize: 512,
     maxZoom: 18,
     zoomOffset: -1,
     // attribution: mbAtt,
     id: 'mapbox/streets-v11'
    })
    var map = new L.Map('map', {layers: [mbl], center: new L.LatLng(43.62455,
-113.2971), zoom: 8});
    var mbrLayer = new L.FeatureGroup();
    map.addLayer(mbrLayer);
    var txtboxLayer = new L.FeatureGroup();
    map.addLayer(txtboxLayer) ;

    var drawControl = new L.Control.Draw({
```

```
        position: 'topright',
        draw: {
            polyline: false,
            polygon: false,
            circle: false,
            marker: false,
            circlemarker: false,
            rectangle: true
        }
    });
    map.addControl(drawControl);

    map.on(L.Draw.Event.DRAWSTART, function (e) {
        mbrLayer.clearLayers();
        txtboxLayer.clearLayers();
    });

    map.on(L.Draw.Event.CREATED, function (e) {
          var b = e.layer._bounds;
          var bounds = fix_bounds([b._southWest.lng, b._northEast.lng,
b._southWest.lat, b._northEast.lat]);

          disable_listeners() ;
          //set the input boxes at the bottom
          document.getElementById("west_lon").value = bounds[0];
          document.getElementById("east_lon").value = bounds[1];
          document.getElementById("south_lat").value = bounds[2];
          document.getElementById("north_lat").value = bounds[3];
          enable_listeners() ;

          //draw the rectangle
          clear_center_radius();
          update_rectangle();
          console.log(bounds);
    });


    const baseMaps = {};
    const overlayMaps = {};
    const layerControl = L.control.layers(baseMaps, overlayMaps).addTo(map);

    //loadKmzFromUrl("big_butte_5_5_307m.kmz");
    //loadKmzFromUrl("big_butte_10_10_307m.kmz");
    L.control.scale().addTo(map);



    // the function evaluated by the C++ code to retrieve the selected box
    function mbr() {
      var b = [
          document.getElementById("west_lon").valueAsNumber,
          document.getElementById("east_lon").valueAsNumber,
          document.getElementById("south_lat").valueAsNumber,
          document.getElementById("north_lat").valueAsNumber
```

```
  ];
  console.log(b);
  return fix_bounds(b);
}

function fix_bounds(bounds) {
  // make shallow copy of array
  // element ordering should be west, east, south, north
  var b = bounds.slice(0) ;

  // normalize longitudes to -180 - 180.
  b[0] = b[0] % 360;
  if(b[0] < -180) {
    b[0] += 360;
  }
  b[1] = b[1] % 360;
  if(b[1] < -180) {
    b[1] += 360;
  }
  // switch to negative longitudes if needed
  if(b[0] > 180){
    b[0] -= 360;
  }
  if(b[1] > 180){
    b[1] -= 360;
  }
  return b;
}

function clear_center_radius() {
    console.log("clear_center_radius")
    disable_listeners();
    //set the input boxes at the bottom
    document.getElementById("center_lat").value =  null ;
    document.getElementById("center_lon").value =  null ;
    document.getElementById("radius").value = null ;
    enable_listeners() ;
}

function update_text_bounds() {
    console.log("update_text_bounds") ;
    clear_center_radius() ;
    update_rectangle() ;
}


function update_rectangle() {
  console.log("update_rectangle") ;
  var bounds = [ [ document.getElementById("south_lat").valueAsNumber,
                   document.getElementById("west_lon").valueAsNumber],
                 [ document.getElementById("north_lat").valueAsNumber,
                   document.getElementById("east_lon").valueAsNumber] ];

  // if any of the text boxes are empty, disregard.
```

```
      for (var i = 0; i < bounds.length; i++) {
          var corner = bounds[i];
          for (var j = 0; j < corner.length; j++) {
             if (isNaN(corner[j])) {
                 return null ;
             }
          }
      }

      // draw the box
      txtboxLayer.clearLayers() ;
      L.rectangle(bounds).addTo(txtboxLayer);
    }

    function update_center() {
        console.log("update_center");
        var center_lat = document.getElementById("center_lat").valueAsNumber;
        var center_lon = document.getElementById("center_lon").valueAsNumber;
        var radius = document.getElementById("radius").valueAsNumber;

        // if any boxes empty, skip
        if (isNaN(center_lat)) return null ;
        if (isNaN(center_lon)) return null ;
        if (isNaN(radius)) return null ;


        // crude spherical earth distance calculation.
        var delta_lat = radius / ( 2 * Math.PI * 3958.7613 / 360.);
        var lat_rad = 3958.7613 * Math.cos(center_lat * Math.PI/180) ;
        var delta_lon = radius / ( 2 * Math.PI * lat_rad / 360.);

        console.log([delta_lat, delta_lon, lat_rad]);

        //set the input boxes at the bottom
        disable_listeners();
        document.getElementById("west_lon").value = center_lon - delta_lon;
        document.getElementById("east_lon").value = center_lon + delta_lon;
        document.getElementById("south_lat").value = center_lat - delta_lat;
        document.getElementById("north_lat").value = center_lat + delta_lat;
        enable_listeners();

        update_rectangle();
    }

    function enable_listeners() {
        // draw a new box whenever user types in coordinates
        document.getElementById("west_lon").addEventListener("input",
update_text_bounds);
        document.getElementById("east_lon").addEventListener("input",
update_text_bounds);
        document.getElementById("south_lat").addEventListener("input",
update_text_bounds);
        document.getElementById("north_lat").addEventListener("input",
update_text_bounds);
```

```
        // calculate new bounds whenever user types in center and radius
        document.getElementById("center_lat").addEventListener("input",
update_center);
        document.getElementById("center_lon").addEventListener("input",
update_center);
        document.getElementById("radius").addEventListener("input", update_center);
    }
    function disable_listeners() {
        // draw a new box whenever user types in coordinates
        document.getElementById("west_lon").removeEventListener("input",
update_text_bounds);
        document.getElementById("east_lon").removeEventListener("input",
update_text_bounds);
        document.getElementById("south_lat").removeEventListener("input",
update_text_bounds);
        document.getElementById("north_lat").removeEventListener("input",
update_text_bounds);

        // calculate new bounds whenever user types in center and radius
        document.getElementById("center_lat").removeEventListener("input",
update_center);
        document.getElementById("center_lon").removeEventListener("input",
update_center);
        document.getElementById("radius").removeEventListener("input",
update_center);
    }

    async function loadKmzFromUrl(kmzUrl) {
      try {
        const response = await fetch(kmzUrl);
        const arrayBuffer = await response.arrayBuffer();

        const zip = await JSZip.loadAsync(arrayBuffer);
        const kmlFiles = Object.values(zip.files).filter(entry =>
          entry.name.toLowerCase().endsWith('.kml')
        );

        if (kmlFiles.length === 0) {
          alert('No KML files found in this KMZ.');
          return;
        }

        const kmlGroup = L.featureGroup().addTo(map);

        for (const entry of kmlFiles) {
          const kmlText = await entry.async('string');
          const parser = new DOMParser();
          const kml = parser.parseFromString(kmlText, 'text/xml');
          const track = new L.KML(kml);
          kmlGroup.addLayer(track);

          const layerName = entry.name || 'Unnamed Layer';
          layerControl.addOverlay(track, layerName);
```

```
        }

        setTimeout(() => {
          if (kmlGroup.getLayers().length > 0) {
            map.fitBounds(kmlGroup.getBounds());
          }
        }, 1000);

      } catch (err) {
        console.error('Failed to load KMZ from URL:', err);
      }
    }

</script>

<div id="data_entry" style="border: 1px solid #ccc">
  <form id="center_radius" style="border-style: solid;">
    <div class="adjust-line-height">
    <p class="option_hdr">Point and Radius:</p>
    <p style="color: grey" class="option_hdr2"> Enter lat, lon, and radius, then
choose Data Source and click Download File</p>
    </div>
    <label for="center_lat">Center Lat</label>
    <span class="input-help">
    <input type="number" placeholder="44.334" id="center_lat" name="center_lat" />
    <small style="color: grey" id="helpBlock" class="form-text text-muted">e.g.,
44.334</small>
    </span>
    <label for="center_lon">Center Lon</label>
    <span class="input-help">
    <input type="number" placeholder="-112.555" id="center_lon" name="center_lon" />
    <small style="color: grey" id="helpBlock" class="form-text text-muted">e.g.,
-112.555</small>
    </span>
    <label for="radius">Radius (miles)</label>
    <span class="input-help">
    <input type="number" placeholder="10" id="radius" name="radius" />
    <small style="color: grey" id="helpBlock" class="form-text text-muted">e.g., 10
</small>
    </span>
  </form>
  <form id="bounds" style="border-style: solid">
    <p class="option_hdr">Bounding Box Coordinates:</p>
    <label for="north_lat">North</label>
    <input type="number" placeholder="44.334" id="north_lat" name="north_lat" />
    <label for="south_lat">South</label>
    <input type="number" placeholder="43.532" id="south_lat" name="south_lat" />
    <label for="east_lon">East</label>
    <input type="number" placeholder="-112.555" id="east_lon" name="east_lon" />
    <label for="west_lon">West</label>
    <input type="number" placeholder="-113.058" id="west_lon" name="west_lon" />
  </form>
</div>
```

```
<script>
  enable_listeners();
</script>
</body>
</html>
```

### 8.8 Source Code - ModelData (header)

```cpp
#ifndef MODELDATA_H
#define MODELDATA_H

#include <string>
#include <vector>

using namespace std;

class BaseInput {
  private:
    string demFile;
    double outputResolution;
    string initializationMethod;
    string meshChoice;
    string vegetation;
    int nLayers;
    int diurnalFlag;
    double height;
    string heightUnits;
    bool momentumFlag;
    int numNinjas;
    string outputPath;

  public:
    BaseInput() {}

    // Constructor
    BaseInput(string demFile, double outputResolution, string initializationMethod,
string meshChoice, string vegetation,
              int nLayers, int diurnalFlag, double height, string heightUnits, bool
momentumFlag, int numNinjas, string outputPath)
      : demFile(demFile), outputResolution(outputResolution),
initializationMethod(initializationMethod), meshChoice(meshChoice),
          vegetation(vegetation), nLayers(nLayers), diurnalFlag(diurnalFlag),
height(height), heightUnits(heightUnits), momentumFlag(momentumFlag),
numNinjas(numNinjas), outputPath(outputPath) {}

    // Getter methods
    const string& getDemFile() const { return demFile; }
    double getOutputResolution() { return outputResolution; }
    const string& getInitializationMethod() const { return initializationMethod; }
    const string& getMeshChoice() const { return meshChoice; }
    const string& getVegetation() const { return vegetation; }
    int getNLayers() { return nLayers; }
    int getDiurnalFlag() { return diurnalFlag; }
    double getHeight() { return height; }
    const string& getHeightUnits() const { return heightUnits; }
    bool getMomentumFlag() { return momentumFlag; }
    int getNumNinjas() { return numNinjas; }
    const string& getOutputPath() const { return outputPath; }
};

class DomainAverageWind : public BaseInput {
  private:
```

```
    vector<double> speedList;
    string speedUnits;
    vector<double> directionList;

  public:
    // Constructor
    DomainAverageWind(const BaseInput& baseInput,
                      vector<double> speedList, string speedUnits, vector<double>
directionList)
      : BaseInput(baseInput), speedList(speedList), speedUnits(speedUnits),
directionList(directionList) {}

    // Getter methods
    vector<double> getSpeedList() { return speedList; }
    const string& getSpeedUnits() const { return speedUnits; }
    vector<double> getDirectionList() { return directionList; }
};

class PointInitialization : public BaseInput {
  private:
    vector<int> year;
    vector<int> month;
    vector<int> day;
    vector<int> hour;
    vector<int> minute;
    char* station_path;
    char* osTimeZone;
    bool matchPointFlag;

  public:
    // Constructor
    PointInitialization(const BaseInput& baseInput,
                        vector<int> year, vector<int> month, vector<int> day,
vector<int> hour, vector<int> minute, char* station_path, char* osTimeZone, bool
matchPointFlag)
        : BaseInput(baseInput), year(year), month(month), day(day), hour(hour),
minute(minute), station_path(station_path), osTimeZone(osTimeZone),
matchPointFlag(matchPointFlag) {}

    // Getter methods
    vector<int> getYear() { return year; }
    vector<int> getMonth() { return month; }
    vector<int> getDay() { return day; }
    vector<int> getHour() { return hour; }
    vector<int> getMinute() { return minute; }
    char* getStationPath() { return station_path; }
    char* getOSTimeZone() { return osTimeZone; }
    bool getMatchPointFlag() {return matchPointFlag;}
};

class WeatherModel : public BaseInput {
  private:
    char* forecast;
    char* osTimeZone;
```

```
  public:
    // Constructor
    WeatherModel(const BaseInput& baseInput,
                 char* forecast, char* osTimeZone)
      : BaseInput(baseInput), forecast(forecast), osTimeZone(osTimeZone) {}

    // Getter methods
    char* getForecast() { return forecast; }
    char* getOSTimeZone() { return osTimeZone; }
};

class modeldata {
  public:
    modeldata() {}
};

#endif // MODELDATA_H
```

### 8.9 Source Code - Provider (header)

```
#ifndef PROVIDER_H
#define PROVIDER_H

#include "appstate.h"
#include "modeldata.h"
#include <cmath>
#include <QDir>
#include <QFileInfo>
#include <QString>
#include <QTableWidget>
#include <vector>
#include <string>

using namespace std;

class Provider {
  private:
    class NinjaArmyH* ninjaArmy = nullptr;
    const char* comType = "cli";
    const int nCPUs = 1;
    char** papszOptions = nullptr;
    int err = 0;

  public:
    Provider();
    int domain_average_exec(class DomainAverageWind& input);
    QVector<QVector<QString>> getTimeZoneData(bool showAllZones);
    QVector<QVector<QString>> parseDomainAvgTable(QTableWidget* table);
    vector<string> getOutputFileNames(QString demFile, QTableWidget* table, QString
meshType, int numFiles, QString outputPath);
    QString getKmzFilePaths();
    QString getTimeZoneDetails(const QString& currentTimeZone);
    void setMapLayers();
    int point_exec(class PointInitialization& input);
    int wxmodel_exec(class WeatherModel& input);

    int fetchDEMBoundingBox(const string demFileOutPut, const string fetch_type, int
resolution, double* boundsBox);
    int fetchForecast(const string wx_model_type, int numNinjas, const string
demFile);
    int fetchStationData(int year, int month, int day, int hour, int minute, int
numNinjas, const string output_path, int bufferZone, const string bufferUnits, const
string demFile, const string osTimeZone, int fetchLatestFlag);
    int fetchFromDEMPoint (double adfPoints[2], double adfBuff[2], const string
units, double dfCellSize, const string pszDstFile, const string fetchtype);

};

#endif //PROVIDER_H
```

### 8.10 Source Code - Provider (c++)

```cpp
#include "provider.h"
#include "modeldata.h"
#include "../../ninja/windninja.h"

#include <iostream>
#include <list>
#include <QDebug>
#include <QFile>
#include <QStringList>
#include <QTextStream>
#include <vector>

using namespace std;


NinjaArmyH* ninjaArmy = NULL;
const char * comType = "cli";
const int nCPUs = 1;
char ** papszOptions = NULL;
NinjaErr err = 0;

Provider::Provider() {

}

int Provider::domain_average_exec(DomainAverageWind& input) {
  std::vector<double> speedVector = input.getSpeedList();
  const double* speedList = speedVector.data();
  std::vector<double> directionVector = input.getDirectionList();
  const double* directionList = directionVector.data();
  const char * demFile = input.getDemFile().c_str();
  double outputResolution = input.getOutputResolution();
  const char* initializationMethod = input.getInitializationMethod().c_str();
  const char* meshChoice = input.getMeshChoice().c_str();
  const char* vegetation = input.getVegetation().c_str();
  const int nLayers = input.getNLayers();
  const int diurnalFlag = input.getDiurnalFlag();
  const double height = input.getHeight();
  const char* heightUnits = input.getHeightUnits().c_str();
  const char* speedUnits = input.getSpeedUnits().c_str();
  bool momentumFlag = input.getMomentumFlag();
  unsigned int numNinjas = input.getNumNinjas();
  const char* outputPath = input.getOutputPath().c_str();


  /*
   * Setting up the simulation
   */
  err = NinjaInit(papszOptions); //initialize global singletons and environments
(GDAL_DATA, etc.)
  ninjaArmy = NinjaMakeDomainAverageArmy(numNinjas, momentumFlag, speedList,
speedUnits, directionList, papszOptions);

  if(err != NINJA_SUCCESS)
```

```
  {
    printf("NinjaInit: err = %d\n", err);
  }

  /*
   * Create the army
   */
  if( NULL == ninjaArmy )
  {
    printf("NinjaCreateArmy: ninjaArmy = NULL\n");
  }

  err = NinjaInit(papszOptions); //must be called for any simulation
  if(err != NINJA_SUCCESS)
  {
    printf("NinjaInit: err = %d\n", err);
  }

  /*
   * Prepare the army
   */
  for(unsigned int i=0; i<input.getNumNinjas(); i++)
  {
    err = NinjaSetCommunication(ninjaArmy, i, comType, papszOptions);
    if(err != NINJA_SUCCESS)
    {
      printf("NinjaSetCommunication: err = %d\n", err);
    }

    err = NinjaSetOutputPath(ninjaArmy, i, outputPath, papszOptions);
    if(err != NINJA_SUCCESS)
    {
      printf("NinjaSetNumberCPUs: err = %d\n", err);
    }

    err = NinjaSetNumberCPUs(ninjaArmy, i, nCPUs, papszOptions);
    if(err != NINJA_SUCCESS)
    {
      printf("NinjaSetNumberCPUs: err = %d\n", err);
    }

    err = NinjaSetInitializationMethod(ninjaArmy, i, initializationMethod,
papszOptions);
    if(err != NINJA_SUCCESS)
    {
      printf("NinjaSetInitializationMethod: err = %d\n", err);
    }

    err = NinjaSetDem(ninjaArmy, i, demFile, papszOptions);
    if(err != NINJA_SUCCESS)
    {
      printf("NinjaSetDem: err = %d\n", err);
    }
```

```
err = NinjaSetPosition(ninjaArmy, i, papszOptions);
if(err != NINJA_SUCCESS)
{
  printf("NinjaSetPosition: err = %d\n", err);
}


//heighUnits set static

err = NinjaSetInputWindHeight(ninjaArmy, i, height, heightUnits, papszOptions);
if(err != NINJA_SUCCESS)
{
  printf("NinjaSetInputWindHeight: err = %d\n", err);
}

err = NinjaSetOutputWindHeight(ninjaArmy, i, height, heightUnits, papszOptions);
if(err != NINJA_SUCCESS)
{
  printf("NinjaSetOutputWindHeight: err = %d\n", err);
}

err = NinjaSetOutputSpeedUnits(ninjaArmy, i, speedUnits, papszOptions);
if(err != NINJA_SUCCESS)
{
  printf("NinjaSetOutputSpeedUnits: err = %d\n", err);
}

err = NinjaSetDiurnalWinds(ninjaArmy, i, diurnalFlag, papszOptions);
if(err != NINJA_SUCCESS)
{
  printf("NinjaSetDiurnalWinds: err = %d\n", err);
}

err = NinjaSetUniVegetation(ninjaArmy, i, vegetation, papszOptions);
if(err != NINJA_SUCCESS)
{
  printf("NinjaSetUniVegetation: err = %d\n", err);
}

err = NinjaSetMeshResolutionChoice(ninjaArmy, i, meshChoice, papszOptions);
if(err != NINJA_SUCCESS)
{
  printf("NinjaSetMeshResolutionChoice: err = %d\n", err);
}

err = NinjaSetNumVertLayers(ninjaArmy, i, nLayers, papszOptions);
if(err != NINJA_SUCCESS)
{
  printf("NinjaSetNumVertLayers: err = %d\n", err);
}


err = NinjaSetGoogOutFlag(ninjaArmy, i, true, papszOptions);
if(err != NINJA_SUCCESS)
```

```
    {
      printf("NinjaSetGoogOutFlag err = %d\n", err);
    }
  }

  /*
   * Start the simulations
   */
  err = NinjaStartRuns(ninjaArmy, nCPUs, papszOptions);
  if(err != 1) //NinjaStartRuns returns 1 on success
  {
    printf("NinjaStartRuns: err = %d\n", err);
  }

  /*
   * Get the outputs
   */
  const double* outputSpeedGrid = NULL;
  const double* outputDirectionGrid = NULL;
  const char* outputGridProjection = NULL;
  const int nIndex = 0;
  const char* units = "m";

  //Google Maps Output
  outputSpeedGrid = NinjaGetOutputSpeedGrid(ninjaArmy, nIndex, papszOptions);
  if( NULL == outputSpeedGrid )
  {
    printf("Error in NinjaGetOutputSpeedGrid");
  }

  outputDirectionGrid = NinjaGetOutputDirectionGrid(ninjaArmy, nIndex,
papszOptions);
  if( NULL == outputDirectionGrid )
  {
    printf("Error in NinjaGetOutputDirectionGrid");
  }

  outputGridProjection = NinjaGetOutputGridProjection(ninjaArmy, nIndex,
papszOptions);
  if( NULL == outputGridProjection )
  {
    printf("Error in NinjaGetOutputGridProjection");
  }

  /*
   * Clean up
   */
  err = NinjaDestroyArmy(ninjaArmy, papszOptions);
  if(err != NINJA_SUCCESS)
  {
    printf("NinjaDestroyRuns: err = %d\n", err);
  }

  return NINJA_SUCCESS;
```

```
}

int Provider::point_exec(PointInitialization& input) {
  /*
   * Setting up the simulation
   */

  //initialize global singletons and environments (GDAL_DATA, etc.)
  if(err != NINJA_SUCCESS)
  {
    printf("NinjaInit: err = %d\n", err);
  }

  /*
   * Create the army
   */
  vector<int> yearVector = input.getYear();
   int* year = yearVector.data();
  vector<int> monthVector = input.getMonth();
   int* month = monthVector.data();
  vector<int> dayVector = input.getDay();
   int* day = dayVector.data();
  vector<int> hourVector = input.getHour();
   int* hour = hourVector.data();
  vector<int> minuteVector = input.getMinute();
   int* minute = minuteVector.data();
   char* station_path = input.getStationPath();
   char* osTimeZone = input.getOSTimeZone();
  bool matchPointFlag = input.getMatchPointFlag();

   char * demFile = const_cast<char*>(input.getDemFile().c_str());
  double outputResolution = input.getOutputResolution();
  const char* initializationMethod = input.getInitializationMethod().c_str();
  const char* meshChoice = input.getMeshChoice().c_str();
  const char* vegetation = input.getVegetation().c_str();
  const int nLayers = input.getNLayers();
  const int diurnalFlag = input.getDiurnalFlag();
  const double height = input.getHeight();
  const char* heightUnits = input.getHeightUnits().c_str();
  bool momentumFlag = input.getMomentumFlag();
  unsigned int numNinjas = input.getNumNinjas();
  const char * outputPath =  input.getOutputPath().c_str();


  ninjaArmy = NinjaMakePointArmy(year, month, day, hour, minute, numNinjas,
osTimeZone, station_path, demFile, matchPointFlag, momentumFlag, papszOptions);

  if( NULL == ninjaArmy )
  {
    printf("NinjaCreateArmy: ninjaArmy = NULL\n");
  }

  err = NinjaInit(papszOptions); //must be called for any simulation
  if(err != NINJA_SUCCESS)
```

```
  {
    printf("NinjaInit: err = %d\n", err);
  }

  /*
   * Prepare the army
   */
  for(unsigned int i=0; i<numNinjas; i++)
  {

    err = NinjaSetOutputPath(ninjaArmy, i, outputPath, papszOptions);
    if(err != NINJA_SUCCESS)
    {
      printf("NinjaSetOutputPath: err = %d\n", err);
    }

    err = NinjaSetGoogOutFlag(ninjaArmy, i, true, papszOptions);
    if(err != NINJA_SUCCESS)
    {
      printf("NinjaSetGoogleOutPut: err = %d\n", err);
    }

    err = NinjaSetCommunication(ninjaArmy, i, comType, papszOptions);
    if(err != NINJA_SUCCESS)
    {
      printf("NinjaSetCommunication: err = %d\n", err);
    }

    err = NinjaSetCommunication(ninjaArmy, i, comType, papszOptions);
    if(err != NINJA_SUCCESS)
    {
      printf("NinjaSetCommunication: err = %d\n", err);
    }

    err = NinjaSetNumberCPUs(ninjaArmy, i, nCPUs, papszOptions);
    if(err != NINJA_SUCCESS)
    {
      printf("NinjaSetNumberCPUs: err = %d\n", err);
    }

    err = NinjaSetInitializationMethod(ninjaArmy, i, initializationMethod,
papszOptions);
    if(err != NINJA_SUCCESS)
    {
      printf("NinjaSetInitializationMethod: err = %d\n", err);
    }

    err = NinjaSetDem(ninjaArmy, i, demFile, papszOptions);
    if(err != NINJA_SUCCESS)
    {
      printf("NinjaSetDem: err = %d\n", err);
    }

    err = NinjaSetPosition(ninjaArmy, i, papszOptions);
```

```
  if(err != NINJA_SUCCESS)
  {
    printf("NinjaSetPosition: err = %d\n", err);
  }

  err = NinjaSetInputWindHeight(ninjaArmy, i, height, heightUnits, papszOptions);
  if(err != NINJA_SUCCESS)
  {
    printf("NinjaSetInputWindHeight: err = %d\n", err);
  }

  err = NinjaSetOutputWindHeight(ninjaArmy, i, height, heightUnits, papszOptions);
  if(err != NINJA_SUCCESS)
  {
    printf("NinjaSetOutputWindHeight: err = %d\n", err);
  }

        //err = NinjaSetOutputSpeedUnits(ninjaArmy, i, speedUnits, papszOptions);
        //if(err != NINJA_SUCCESS)
        //{
        //  printf("NinjaSetOutputSpeedUnits: err = %d\n", err);
        //}

  err = NinjaSetDiurnalWinds(ninjaArmy, i, diurnalFlag, papszOptions);
  if(err != NINJA_SUCCESS)
  {
    printf("NinjaSetDiurnalWinds: err = %d\n", err);
  }

  err = NinjaSetUniVegetation(ninjaArmy, i, vegetation, papszOptions);
  if(err != NINJA_SUCCESS)
  {
    printf("NinjaSetUniVegetation: err = %d\n", err);
  }

  err = NinjaSetMeshResolutionChoice(ninjaArmy, i, meshChoice, papszOptions);
  if(err != NINJA_SUCCESS)
  {
    printf("NinjaSetMeshResolutionChoice: err = %d\n", err);
  }

  err = NinjaSetNumVertLayers(ninjaArmy, i, nLayers, papszOptions);
  if(err != NINJA_SUCCESS)
  {
    printf("NinjaSetNumVertLayers: err = %d\n", err);
  }
}

/*
 * Start the simulations
 */
err = NinjaStartRuns(ninjaArmy, nCPUs, papszOptions);
if(err != 1) //NinjaStartRuns returns 1 on success
{
```

```
    printf("NinjaStartRuns: err = %d\n", err);
  }

  return NINJA_SUCCESS;
}

int Provider::wxmodel_exec(WeatherModel& input) {
  /*
   * Setting up the simulation
   */

  err = NinjaInit(papszOptions); //initialize global singletons and environments
(GDAL_DATA, etc.)
  if(err != NINJA_SUCCESS)
  {
    printf("NinjaInit: err = %d\n", err);
  }

  /*
   * Set up Weather Model Initialization run
   */

  const char * demFile = input.getDemFile().c_str();
  double outputResolution = input.getOutputResolution();
  const char* initializationMethod = input.getInitializationMethod().c_str();
  const char* meshChoice = input.getMeshChoice().c_str();
  const char* vegetation = input.getVegetation().c_str();
  const int nLayers = input.getNLayers();
  const int diurnalFlag = input.getDiurnalFlag();
  const double height = input.getHeight();
  const char* heightUnits = input.getHeightUnits().c_str();
  bool momentumFlag = input.getMomentumFlag();
  unsigned int numNinjas = input.getNumNinjas();
  const char* outputPath = input.getOutputPath().c_str();


  /* inputs that can vary among ninjas in an army */
  //const char * speedUnits = input.get;

  /*
   * Create the army
   */
  const char * forecast = input.getForecast();
  const char * osTimeZone = input.getOSTimeZone();

  ninjaArmy = NinjaMakeWeatherModelArmy(forecast, osTimeZone, momentumFlag,
papszOptions);
  if( NULL == ninjaArmy )
  {
    printf("NinjaCreateArmy: ninjaArmy = NULL\n");
  }

  err = NinjaInit(papszOptions); //must be called for any simulation
  if(err != NINJA_SUCCESS)
```

```
  {
    printf("NinjaInit: err = %d\n", err);
  }

  /*
   * Prepare the army
   */
  for(unsigned int i=0; i<numNinjas; i++)
  {
    err = NinjaSetOutputPath(ninjaArmy, i, outputPath, papszOptions);
    if(err != NINJA_SUCCESS)
    {
      printf("NinjaSetOutputPath: err = %d\n", err);
    }

    err = NinjaSetGoogOutFlag(ninjaArmy, i, true, papszOptions);
    if(err != NINJA_SUCCESS)
    {
      printf("NinjaSetGoogleOutPut: err = %d\n", err);
    }


    err = NinjaSetCommunication(ninjaArmy, i, comType, papszOptions);
    if(err != NINJA_SUCCESS)
    {
      printf("NinjaSetCommunication: err = %d\n", err);
    }

    err = NinjaSetNumberCPUs(ninjaArmy, i, nCPUs, papszOptions);
    if(err != NINJA_SUCCESS)
    {
      printf("NinjaSetNumberCPUs: err = %d\n", err);
    }

    err = NinjaSetInitializationMethod(ninjaArmy, i, initializationMethod,
papszOptions);
    if(err != NINJA_SUCCESS)
    {
      printf("NinjaSetInitializationMethod: err = %d\n", err);
    }

    err = NinjaSetDem(ninjaArmy, i, demFile, papszOptions);
    if(err != NINJA_SUCCESS)
    {
      printf("NinjaSetDem: err = %d\n", err);
    }

    err = NinjaSetPosition(ninjaArmy, i, papszOptions);
    if(err != NINJA_SUCCESS)
    {
      printf("NinjaSetPosition: err = %d\n", err);
    }

    err = NinjaSetInputWindHeight(ninjaArmy, i, height, heightUnits, papszOptions);
```

```
  if(err != NINJA_SUCCESS)
  {
    printf("NinjaSetInputWindHeight: err = %d\n", err);
  }

  err = NinjaSetOutputWindHeight(ninjaArmy, i, height, heightUnits, papszOptions);
  if(err != NINJA_SUCCESS)
  {
    printf("NinjaSetOutputWindHeight: err = %d\n", err);
  }

        //err = NinjaSetOutputSpeedUnits(ninjaArmy, i, speedUnits, papszOptions);
        //if(err != NINJA_SUCCESS)
        //{
        //  printf("NinjaSetOutputSpeedUnits: err = %d\n", err);
        //}

  err = NinjaSetDiurnalWinds(ninjaArmy, i, diurnalFlag, papszOptions);
  if(err != NINJA_SUCCESS)
  {
    printf("NinjaSetDiurnalWinds: err = %d\n", err);
  }

  err = NinjaSetUniVegetation(ninjaArmy, i, vegetation, papszOptions);
  if(err != NINJA_SUCCESS)
  {
    printf("NinjaSetUniVegetation: err = %d\n", err);
  }

  err = NinjaSetMeshResolutionChoice(ninjaArmy, i, meshChoice, papszOptions);
  if(err != NINJA_SUCCESS)
  {
    printf("NinjaSetMeshResolutionChoice: err = %d\n", err);
  }

  err = NinjaSetNumVertLayers(ninjaArmy, i, nLayers, papszOptions);
  if(err != NINJA_SUCCESS)
  {
    printf("NinjaSetNumVertLayers: err = %d\n", err);
  }
}

/*
 * Start the simulations
 */
err = NinjaStartRuns(ninjaArmy, nCPUs, papszOptions);
if(err != 1) //NinjaStartRuns returns 1 on success
{
  printf("NinjaStartRuns: err = %d\n", err);
}

/*
 * Get the outputs
 */
```

```
  const double* outputSpeedGrid = NULL;
  const double* outputDirectionGrid = NULL;
  const char* outputGridProjection = NULL;
  const int nIndex = 0;
  const char* units = "m";
  outputSpeedGrid = NinjaGetOutputSpeedGrid(ninjaArmy, nIndex, papszOptions);
  if( NULL == outputSpeedGrid )
  {
    printf("Error in NinjaGetOutputSpeedGrid");
  }

  outputDirectionGrid = NinjaGetOutputDirectionGrid(ninjaArmy, nIndex,
papszOptions);
  if( NULL == outputDirectionGrid )
  {
    printf("Error in NinjaGetOutputDirectionGrid");
  }

  outputGridProjection = NinjaGetOutputGridProjection(ninjaArmy, nIndex,
papszOptions);
  if( NULL == outputGridProjection )
  {
    printf("Error in NinjaGetOutputGridProjection");
  }

  /*
   * Clean up
   */
  err = NinjaDestroyArmy(ninjaArmy, papszOptions);
  if(err != NINJA_SUCCESS)
  {
    printf("NinjaDestroyRuns: err = %d\n", err);
  }

  return NINJA_SUCCESS;
}

// Time zone data provider
QVector<QVector<QString>> Provider::getTimeZoneData(bool showAllZones) {
  QVector<QVector<QString>> fullData;
  QVector<QVector<QString>> americaData;

  QFile file(":/date_time_zonespec.csv");

  if (!file.open(QIODevice::ReadOnly | QIODevice::Text)) {
    qDebug() << "Failed to open CSV file.";
    return fullData;
  }

  QTextStream in(&file);
  bool firstLine = true;

  while (!in.atEnd()) {
    QString line = in.readLine();
```

```
    if (firstLine) {
      firstLine = false;
      continue;
    }

    QStringList tokens = line.split(",", Qt::KeepEmptyParts);
    QVector<QString> row;
    for (const QString& token : tokens)
      row.append(token.trimmed().remove('"'));

    if (!row.isEmpty())
      fullData.append(row);

    if (!row.isEmpty()) {
      QStringList parts = row[0].split("/", Qt::KeepEmptyParts);
      if (!parts.isEmpty() && parts[0] == "America" || row[0] == "Pacific/Honolulu")
{
        americaData.append(row);
      }
    }
  }

  file.close();

  if (showAllZones) {
    return fullData;
  } else {
    return americaData;
  }

}

// Provider for getting time zone details
QString Provider::getTimeZoneDetails(const QString& currentTimeZone) {
  QVector<QString> matchedRow;
  QFile file(":/date_time_zonespec.csv");

  if (!file.open(QIODevice::ReadOnly | QIODevice::Text)) {
    qWarning() << "Failed to open date_time_zonespec.csv";
    return "No data found";
  }

  QTextStream in(&file);
  bool firstLine = true;

  while (!in.atEnd()) {
    QString line = in.readLine();

    if (firstLine) {
      firstLine = false;
      continue;  // skip header
    }
```

```
  QStringList tokens = line.split(",", Qt::KeepEmptyParts);
  QVector<QString> row;

  for (const QString& token : tokens)
    row.append(token.trimmed().remove("\""));

  QString fullZone = row.mid(0, 1).join("/");

  if (fullZone == currentTimeZone) {
    matchedRow = row;
    break;
  }
}

file.close();

if (matchedRow.isEmpty()) {
  return "No matching time zone found.";
}

QString standardName = matchedRow.value(2);
QString daylightName = matchedRow.value(4);
QString stdOffsetStr = matchedRow.value(5);  // Already in HH:MM:SS
QString dstAdjustStr = matchedRow.value(6);  // Already in HH:MM:SS

// Function to convert signed HH:MM:SS to total seconds
auto timeToSeconds = [](const QString& t) -> int {
  QString s = t.trimmed();
  bool negative = s.startsWith('-');
  s = s.remove(QChar('+')).remove(QChar('-'));

  QStringList parts = s.split(':');
  if (parts.size() != 3) return 0;

  int h = parts[0].toInt();
  int m = parts[1].toInt();
  int sec = parts[2].toInt();

  int total = h * 3600 + m * 60 + sec;
  return negative ? -total : total;
};

// Convert total seconds back to HH:MM:SS with sign
auto secondsToTime = [](int totalSec) -> QString {
  QChar sign = totalSec < 0 ? '-' : '+';
  totalSec = std::abs(totalSec);

  int h = totalSec / 3600;
  int m = (totalSec % 3600) / 60;
  int s = totalSec % 60;

  return QString("%1%2:%3:%4")
    .arg(sign)
    .arg(h, 2, 10, QChar('0'))
```

```
        .arg(m, 2, 10, QChar('0'))
        .arg(s, 2, 10, QChar('0'));
  };

  int stdSecs = timeToSeconds(stdOffsetStr);
  int dstSecs = timeToSeconds(dstAdjustStr);
  QString combinedOffsetStr = secondsToTime(stdSecs + dstSecs);

  return QString("Standard Name:\t\t%1\n"
                 "Daylight Name:\t\t%2\n"
                 "Standard Offset from UTC:\t%3\n"
                 "Daylight Offset from UTC:\t%4")
      .arg(standardName)
      .arg(daylightName)
      .arg(stdOffsetStr)
      .arg(combinedOffsetStr);
}

// Provider for parsing the domain average wind table
QVector<QVector<QString>> Provider::parseDomainAvgTable(QTableWidget* table) {
  QVector<QVector<QString>> result;

  int rowCount = table->rowCount();
  int colCount = table->columnCount();

  for (int row = 0; row < rowCount; ++row) {
    bool rowComplete = true;
    QVector<QString> rowData;

    for (int col = 0; col < colCount; ++col) {
      if (table->isColumnHidden(col)) {
        continue;  // skip this column entirely
      }

      QTableWidgetItem* item = table->item(row, col);

      if (!item || item->text().trimmed().isEmpty()) {
        rowComplete = false;
        break;
      }

      rowData.append(item->text().trimmed());
    }

    if (rowComplete) {
      result.append(rowData);
    }
  }
  return result;
}

int Provider::fetchDEMBoundingBox(const string demFileOutPut, const string
fetch_type, int resolution, double* boundsBox)
{
```

```
  /*
   * Setting up NinjaArmy
   */

  NinjaArmyH* ninjaArmy = NULL;
  char ** papszOptions = NULL;
  NinjaErr err = 0;
  err = NinjaInit(papszOptions); // must be called for fetching and simulations
  if(err != NINJA_SUCCESS)
  {
    printf("NinjaInit: err = %d\n", err);
  }
  /*
   * Testing fetching from a DEM bounding box
   */
  // Bounding box (north, east, south, west)
  err = NinjaFetchDEMBBox(ninjaArmy, boundsBox, demFileOutPut.c_str(), resolution,
strdup(fetch_type.c_str()), papszOptions);
  if (err != NINJA_SUCCESS){
    printf("NinjaFetchDEMBBox: err = %d\n", err);
  }

  return NINJA_SUCCESS;
}

int Provider::fetchForecast(const string wx_model_type, int numNinjas, const string
demFile){
  /*
   * const char*wx_model_type = "NOMADS-HRRR-CONUS-3-KM";
   * int numNinjas = 2;
   */

  /*
   * Setting up NinjaArmy
   */
  NinjaArmyH* ninjaArmy = NULL;
  char ** papszOptions = NULL;
  NinjaErr err = 0;
  err = NinjaInit(papszOptions); // must be called for fetching and simulations
  if(err != NINJA_SUCCESS)
  {
    printf("NinjaInit: err = %d\n", err);
  }
  /*
   * Testing fetching for a Forecast file (wx_model_type are the names listed in the
weather station download in WindNinja)
   */
  const char* forecastFilename = NinjaFetchForecast(ninjaArmy,
wx_model_type.c_str(), numNinjas, demFile.c_str(), papszOptions);
  if (strcmp(forecastFilename, "exception") == 0){
    printf("NinjaFetchForecast: err = %s\n", forecastFilename);
  }
  else{
    printf("NinjaFetchForecast: forecastFilename = %s\n", forecastFilename);
```

```
  }

  return NINJA_SUCCESS;
}

int Provider::fetchStationData(int year, int month, int day, int hour, int minute,
int numNinjas, const string output_path, int bufferZone, const string bufferUnits,
const string demFile, const string osTimeZone, int fetchLatestFlag){
  /*
   * int year[1] = {2023};
   * int month[1] = {10};
   * int day[1] = {10};
   * int hour[1] = {12};
   * int minute[1] = {60};
   * const char* output_path = "";
   * const char* elevation_file = "data/missoula_valley.tif";
   * const char* osTimeZone = "UTC";
   * bool fetchLatestFlag = 1;
   * /

 /*
  * Setting up NinjaArmy
  */
  NinjaArmyH* ninjaArmy = NULL;
  char ** papszOptions = NULL;
  NinjaErr err = 0;
  err = NinjaInit(papszOptions); // must be called for fetching and simulations
  if(err != NINJA_SUCCESS)
  {
    printf("NinjaInit: err = %d\n", err);
  }

  /*
   * Testing fetching station data from a geotiff file.
   */

  err = NinjaFetchStation(&year, &month, &day, &hour, &minute, numNinjas,
demFile.c_str(), bufferZone, bufferUnits.c_str(), osTimeZone.c_str(),
fetchLatestFlag, output_path.c_str(), papszOptions);
  if (err != NINJA_SUCCESS) {
    printf("NinjaFetchStation: err = %d\n", err);
  } else {
    printf("NinjaFetchStation: success\n");
  }

  return NINJA_SUCCESS;
}

int Provider::fetchFromDEMPoint (double adfPoints[2], double adfBuff[2], const
string units, double dfCellSize, const string pszDstFile, const string fetchtype){
  /*
   * double adfPoint[] = {104.0, 40.07}; // Point coordinates (longitude, latitude)
   * double adfBuff[] = {30, 30}; // Buffer to store the elevation value
   * const char* units = "mi";
```

```
    * double dfCellSize = 30.0; // Cell size in meters
    * char* pszDstFile = "data/dem_point_output.tif";
    * char* fetchType = "gmted";
    */

   /*
    * Setting up NinjaArmy
    */
   NinjaArmyH* ninjaArmy = NULL;
   char ** papszOptions = NULL;
   NinjaErr err = 0;
   err = NinjaInit(papszOptions); // must be called for fetching and simulations
   if(err != NINJA_SUCCESS)
   {
     printf("NinjaInit: err = %d\n", err);
   }


   /*
    * Testing fetching from a DEM point
    */
   err = NinjaFetchDEMPoint(ninjaArmy, adfPoints, adfBuff, units.c_str(), dfCellSize,
strdup(pszDstFile.c_str()), strdup(fetchtype.c_str()), papszOptions);
   if (err != NINJA_SUCCESS) {
     printf("NinjaFetchDemPoint: err = %d\n", err);
   } else {
     printf("NinjaFetchDemPoint: elevation = %f\n", adfBuff[0]);
   }

   return NINJA_SUCCESS;
}

vector<string> Provider::getOutputFileNames(QString demFile, QTableWidget *table,
QString meshValue, int numFiles, QString outputPath) {
   vector<string> outputFiles;
   QDir outDir(outputPath);
   QString demName = QFileInfo(demFile).completeBaseName();
   int meshInt = static_cast<int>(std::round(meshValue.toDouble()));
   QString meshSize = QString::number(meshInt) + "m";

   for (int i = 0; i < numFiles; i++) {
     QString direction = table->item(i, 1)->text().trimmed();
     QString speed = table->item(i, 0)->text().trimmed();
     QString filePath = outDir.filePath(QString("%1_%2_%3_%4.kmz")
                                .arg(demName)
                                .arg(direction)
                                .arg(speed)
                                .arg(meshSize));
     outputFiles.push_back(filePath.toStdString());
   }

   return outputFiles;
}
```

### 8.11 Source Code - MainWindow Styling (XML)

```
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
 <class>MainWindow</class>
 <widget class="QMainWindow" name="MainWindow">
  <property name="geometry">
   <rect>
    <x>0</x>
    <y>0</y>
    <width>1081</width>
    <height>638</height>
   </rect>
  </property>
  <property name="sizePolicy">
   <sizepolicy hsizetype="Minimum" vsizetype="Minimum">
    <horstretch>0</horstretch>
    <verstretch>0</verstretch>
   </sizepolicy>
  </property>
  <property name="minimumSize">
   <size>
    <width>490</width>
    <height>500</height>
   </size>
  </property>
  <property name="baseSize">
   <size>
    <width>1000</width>
    <height>1000</height>
   </size>
  </property>
  <property name="palette">
   <palette>
    <active>
     <colorrole role="WindowText">
      <brush brushstyle="SolidPattern">
       <color alpha="255">
        <red>255</red>
        <green>255</green>
        <blue>255</blue>
       </color>
      </brush>
     </colorrole>
     <colorrole role="Button">
      <brush brushstyle="SolidPattern">
       <color alpha="255">
        <red>60</red>
        <green>60</green>
        <blue>60</blue>
       </color>
      </brush>
     </colorrole>
     <colorrole role="Text">
      <brush brushstyle="SolidPattern">
       <color alpha="255">
```

```
      <red>255</red>
      <green>255</green>
      <blue>255</blue>
     </color>
    </brush>
   </colorrole>
   <colorrole role="ButtonText">
    <brush brushstyle="SolidPattern">
     <color alpha="255">
      <red>255</red>
      <green>255</green>
      <blue>255</blue>
     </color>
    </brush>
   </colorrole>
   <colorrole role="Base">
    <brush brushstyle="SolidPattern">
     <color alpha="255">
      <red>60</red>
      <green>60</green>
      <blue>60</blue>
     </color>
    </brush>
   </colorrole>
   <colorrole role="Window">
    <brush brushstyle="SolidPattern">
     <color alpha="255">
      <red>60</red>
      <green>60</green>
      <blue>60</blue>
     </color>
    </brush>
   </colorrole>
  </active>
  <inactive>
   <colorrole role="WindowText">
    <brush brushstyle="SolidPattern">
     <color alpha="255">
      <red>255</red>
      <green>255</green>
      <blue>255</blue>
     </color>
    </brush>
   </colorrole>
   <colorrole role="Button">
    <brush brushstyle="SolidPattern">
     <color alpha="255">
      <red>60</red>
      <green>60</green>
      <blue>60</blue>
     </color>
    </brush>
   </colorrole>
   <colorrole role="Text">
```

```
   <brush brushstyle="SolidPattern">
    <color alpha="255">
     <red>255</red>
     <green>255</green>
     <blue>255</blue>
    </color>
   </brush>
  </colorrole>
  <colorrole role="ButtonText">
   <brush brushstyle="SolidPattern">
    <color alpha="255">
     <red>255</red>
     <green>255</green>
     <blue>255</blue>
    </color>
   </brush>
  </colorrole>
  <colorrole role="Base">
   <brush brushstyle="SolidPattern">
    <color alpha="255">
     <red>60</red>
     <green>60</green>
     <blue>60</blue>
    </color>
   </brush>
  </colorrole>
  <colorrole role="Window">
   <brush brushstyle="SolidPattern">
    <color alpha="255">
     <red>60</red>
     <green>60</green>
     <blue>60</blue>
    </color>
   </brush>
  </colorrole>
 </inactive>
 <disabled>
  <colorrole role="Button">
   <brush brushstyle="SolidPattern">
    <color alpha="255">
     <red>60</red>
     <green>60</green>
     <blue>60</blue>
    </color>
   </brush>
  </colorrole>
  <colorrole role="Base">
   <brush brushstyle="SolidPattern">
    <color alpha="255">
     <red>60</red>
     <green>60</green>
     <blue>60</blue>
    </color>
   </brush>
```

```
   </colorrole>
   <colorrole role="Window">
    <brush brushstyle="SolidPattern">
     <color alpha="255">
      <red>60</red>
      <green>60</green>
      <blue>60</blue>
     </color>
    </brush>
   </colorrole>
  </disabled>
 </palette>
</property>
<property name="windowTitle">
 <string>WindNinja</string>
</property>
<property name="styleSheet">
 <string notr="true"/>
</property>
<widget class="QWidget" name="centralwidget">
 <property name="enabled">
  <bool>true</bool>
 </property>
 <property name="sizePolicy">
  <sizepolicy hsizetype="Expanding" vsizetype="Minimum">
   <horstretch>0</horstretch>
   <verstretch>0</verstretch>
  </sizepolicy>
 </property>
 <property name="minimumSize">
  <size>
   <width>0</width>
   <height>0</height>
  </size>
 </property>
 <property name="autoFillBackground">
  <bool>false</bool>
 </property>
 <property name="styleSheet">
  <string notr="true"/>
 </property>
 <layout class="QHBoxLayout" name="horizontalLayout_11">
  <property name="leftMargin">
   <number>0</number>
  </property>
  <property name="topMargin">
   <number>0</number>
  </property>
  <property name="rightMargin">
   <number>0</number>
  </property>
  <property name="bottomMargin">
   <number>0</number>
  </property>
```

```
<item>
 <widget class="QSplitter" name="mainWindowSplitter">
  <property name="sizePolicy">
   <sizepolicy hsizetype="Expanding" vsizetype="Expanding">
    <horstretch>0</horstretch>
    <verstretch>0</verstretch>
   </sizepolicy>
  </property>
  <property name="minimumSize">
   <size>
    <width>200</width>
    <height>0</height>
   </size>
  </property>
  <property name="baseSize">
   <size>
    <width>225</width>
    <height>0</height>
   </size>
  </property>
  <property name="palette">
   <palette>
    <active>
     <colorrole role="ButtonText">
      <brush brushstyle="SolidPattern">
       <color alpha="255">
        <red>255</red>
        <green>255</green>
        <blue>255</blue>
       </color>
      </brush>
     </colorrole>
    </active>
    <inactive>
     <colorrole role="ButtonText">
      <brush brushstyle="SolidPattern">
       <color alpha="255">
        <red>255</red>
        <green>255</green>
        <blue>255</blue>
       </color>
      </brush>
     </colorrole>
    </inactive>
    <disabled/>
   </palette>
  </property>
  <property name="frameShape">
   <enum>QFrame::Shape::NoFrame</enum>
  </property>
  <property name="orientation">
   <enum>Qt::Orientation::Horizontal</enum>
  </property>
  <property name="handleWidth">
```

```
 <number>15</number>
</property>
<property name="childrenCollapsible">
 <bool>false</bool>
</property>
<widget class="QWidget" name="verticalLayoutWidget">
 <layout class="QVBoxLayout" name="inputLayout" stretch="2,1,0,0">
  <property name="spacing">
   <number>0</number>
  </property>
  <property name="sizeConstraint">
   <enum>QLayout::SizeConstraint::SetMinimumSize</enum>
  </property>
  <property name="leftMargin">
   <number>5</number>
  </property>
  <property name="topMargin">
   <number>10</number>
  </property>
  <property name="rightMargin">
   <number>5</number>
  </property>
  <property name="bottomMargin">
   <number>0</number>
  </property>
  <item>
   <widget class="QTreeWidget" name="treeWidget">
    <property name="sizePolicy">
     <sizepolicy hsizetype="Maximum" vsizetype="Minimum">
      <horstretch>0</horstretch>
      <verstretch>0</verstretch>
     </sizepolicy>
    </property>
    <property name="minimumSize">
     <size>
      <width>0</width>
      <height>250</height>
     </size>
    </property>
    <property name="maximumSize">
     <size>
      <width>600</width>
      <height>250</height>
     </size>
    </property>
    <property name="palette">
     <palette>
      <active>
       <colorrole role="Base">
        <brush brushstyle="SolidPattern">
         <color alpha="255">
          <red>60</red>
          <green>60</green>
          <blue>60</blue>
```

```
      </color>
     </brush>
    </colorrole>
   </active>
   <inactive>
    <colorrole role="Base">
     <brush brushstyle="SolidPattern">
      <color alpha="255">
       <red>60</red>
       <green>60</green>
       <blue>60</blue>
      </color>
     </brush>
    </colorrole>
   </inactive>
   <disabled/>
  </palette>
 </property>
 <property name="font">
  <font>
   <pointsize>12</pointsize>
  </font>
 </property>
 <property name="cursor" stdset="0">
  <cursorShape>PointingHandCursor</cursorShape>
 </property>
 <property name="styleSheet">
  <string notr="true"/>
 </property>
 <property name="frameShape">
  <enum>QFrame::Shape::StyledPanel</enum>
 </property>
 <property name="autoExpandDelay">
  <number>0</number>
 </property>
 <property name="uniformRowHeights">
  <bool>false</bool>
 </property>
 <property name="itemsExpandable">
  <bool>true</bool>
 </property>
 <property name="allColumnsShowFocus">
  <bool>false</bool>
 </property>
 <property name="wordWrap">
  <bool>true</bool>
 </property>
 <property name="expandsOnDoubleClick">
  <bool>true</bool>
 </property>
 <attribute name="headerVisible">
  <bool>false</bool>
 </attribute>
 <column>
```

```
 <property name="text">
  <string>Configuration</string>
 </property>
</column>
<item>
 <property name="text">
  <string>Solver Methodology</string>
 </property>
 <property name="flags">
  <set>ItemIsSelectable|ItemIsEnabled</set>
 </property>
 <item>
  <property name="text">
   <string>Conservation of Mass</string>
  </property>
  <property name="flags">
   <set>ItemIsSelectable|ItemIsEnabled</set>
  </property>
 </item>
 <item>
  <property name="text">
   <string>Conservation of Mass and Momentum</string>
  </property>
  <property name="flags">
   <set>ItemIsSelectable|ItemIsEnabled</set>
  </property>
 </item>
</item>
<item>
 <property name="text">
  <string>Inputs</string>
 </property>
 <property name="flags">
  <set>ItemIsSelectable|ItemIsEnabled</set>
 </property>
 <item>
  <property name="text">
   <string>Surface Input</string>
  </property>
  <property name="flags">
   <set>ItemIsSelectable|ItemIsEnabled</set>
  </property>
 </item>
 <item>
  <property name="text">
   <string>Diurnal Input</string>
  </property>
  <property name="flags">
   <set>ItemIsSelectable|ItemIsEnabled</set>
  </property>
 </item>
 <item>
  <property name="text">
   <string>Stability Input</string>
```

```
     </property>
     <property name="flags">
      <set>ItemIsSelectable|ItemIsEnabled</set>
     </property>
    </item>
    <item>
     <property name="text">
      <string>Wind Input</string>
     </property>
     <property name="flags">
      <set>ItemIsSelectable|ItemIsEnabled</set>
     </property>
     <item>
      <property name="text">
       <string>Domain Average Wind</string>
      </property>
      <property name="flags">
       <set>ItemIsSelectable|ItemIsEnabled</set>
      </property>
     </item>
     <item>
      <property name="text">
       <string>Point Initialization</string>
      </property>
      <property name="flags">
       <set>ItemIsSelectable|ItemIsEnabled</set>
      </property>
     </item>
     <item>
      <property name="text">
       <string>Weather Model</string>
      </property>
      <property name="flags">
       <set>ItemIsSelectable|ItemIsEnabled</set>
      </property>
     </item>
    </item>
   </item>
   <item>
    <property name="text">
     <string>Solver</string>
    </property>
   </item>
  </widget>
 </item>
 <item>
  <spacer name="verticalSpacer_4">
   <property name="orientation">
    <enum>Qt::Orientation::Vertical</enum>
   </property>
   <property name="sizeType">
    <enum>QSizePolicy::Policy::Fixed</enum>
   </property>
   <property name="sizeHint" stdset="0">
```

```
    <size>
     <width>20</width>
     <height>10</height>
    </size>
   </property>
  </spacer>
 </item>
 <item>
  <widget class="QStackedWidget" name="stackedInputPage">
   <property name="sizePolicy">
    <sizepolicy hsizetype="Expanding" vsizetype="Expanding">
     <horstretch>0</horstretch>
     <verstretch>0</verstretch>
    </sizepolicy>
   </property>
   <property name="minimumSize">
    <size>
     <width>0</width>
     <height>250</height>
    </size>
   </property>
   <property name="maximumSize">
    <size>
     <width>600</width>
     <height>16777215</height>
    </size>
   </property>
   <property name="baseSize">
    <size>
     <width>300</width>
     <height>0</height>
    </size>
   </property>
   <property name="frameShape">
    <enum>QFrame::Shape::StyledPanel</enum>
   </property>
   <property name="frameShadow">
    <enum>QFrame::Shadow::Sunken</enum>
   </property>
   <property name="currentIndex">
    <number>11</number>
   </property>
   <widget class="QWidget" name="blankPage"/>
   <widget class="QWidget" name="solverMethodologyPage">
    <layout class="QHBoxLayout" name="horizontalLayout_3">
     <property name="spacing">
      <number>0</number>
     </property>
     <property name="leftMargin">
      <number>0</number>
     </property>
     <property name="topMargin">
      <number>0</number>
     </property>
```

```
            <property name="rightMargin">
             <number>0</number>
            </property>
            <property name="bottomMargin">
             <number>0</number>
            </property>
            <item>
             <widget class="QTextEdit" name="textEdit_7">
              <property name="font">
               <font>
                <pointsize>10</pointsize>
               </font>
              </property>
              <property name="html">
               <string>&lt;!DOCTYPE HTML PUBLIC &quot;-//W3C//DTD HTML 4.0//EN&quot;
&quot;http://www.w3.org/TR/REC-html40/strict.dtd&quot;&gt;
&lt;html&gt;&lt;head&gt;&lt;meta name=&quot;qrichtext&quot; content=&quot;1&quot; /
&gt;&lt;meta charset=&quot;utf-8&quot; /&gt;&lt;style type=&quot;text/css&quot;&gt;
p, li { white-space: pre-wrap; }
hr { height: 1px; border-width: 0; }
li.unchecked::marker { content: &quot;\2610&quot;; }
li.checked::marker { content: &quot;\2612&quot;; }
&lt;/style&gt;&lt;/head&gt;&lt;body style=&quot; font-family:'Ubuntu'; font-size:
10pt; font-weight:400; font-style:normal;&quot;&gt;
&lt;p style=&quot; margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:
0px; -qt-block-indent:0; text-indent:0px;&quot;&gt;&lt;span style=&quot; font-
family:'Sans Serif';&quot;&gt;A solution requires the selection of one (1) Solver
Methodology:&lt;/span&gt;&lt;/p&gt;
&lt;p style=&quot;-qt-paragraph-type:empty; margin-top:0px; margin-bottom:0px;
margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px; font-
family:'Sans Serif';&quot;&gt;&lt;br /&gt;&lt;/p&gt;
&lt;p style=&quot; margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:
0px; -qt-block-indent:0; text-indent:0px;&quot;&gt;&lt;span style=&quot; font-
family:'Sans Serif';&quot;&gt;    1. Conservation of Mass (faster, less
accurate)&lt;/span&gt;&lt;/p&gt;
&lt;p style=&quot; margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:
0px; -qt-block-indent:0; text-indent:0px;&quot;&gt;&lt;span style=&quot; font-
family:'Sans Serif';&quot;&gt;    2. Conservation of Mass and Momentum (slower, more
accurate)&lt;/span&gt;&lt;/p&gt;
&lt;p style=&quot;-qt-paragraph-type:empty; margin-top:0px; margin-bottom:0px;
margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px; font-
family:'Sans Serif';&quot;&gt;&lt;br /&gt;&lt;/p&gt;
&lt;p style=&quot; margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:
0px; -qt-block-indent:0; text-indent:0px;&quot;&gt;&lt;span style=&quot; font-
family:'Sans Serif';&quot;&gt;Select either option for more detail.&lt;/
span&gt;&lt;/p&gt;&lt;/body&gt;&lt;/html&gt;</string>
              </property>
              <property name="textInteractionFlags">
               <set>Qt::TextInteractionFlag::TextSelectableByKeyboard|
Qt::TextInteractionFlag::TextSelectableByMouse</set>
              </property>
             </widget>
            </item>
           </layout>
```

```
</widget>
<widget class="QWidget" name="comPage">
 <property name="sizePolicy">
  <sizepolicy hsizetype="Expanding" vsizetype="Expanding">
   <horstretch>0</horstretch>
   <verstretch>0</verstretch>
  </sizepolicy>
 </property>
 <layout class="QVBoxLayout" name="verticalLayout_2">
  <property name="leftMargin">
   <number>5</number>
  </property>
  <property name="topMargin">
   <number>5</number>
  </property>
  <property name="rightMargin">
   <number>5</number>
  </property>
  <property name="bottomMargin">
   <number>5</number>
  </property>
  <item>
   <widget class="QGroupBox" name="comGroupBox">
    <property name="title">
     <string/>
    </property>
    <layout class="QHBoxLayout" name="horizontalLayout_9">
     <property name="spacing">
      <number>0</number>
     </property>
     <property name="leftMargin">
      <number>5</number>
     </property>
     <property name="topMargin">
      <number>5</number>
     </property>
     <property name="rightMargin">
      <number>5</number>
     </property>
     <property name="bottomMargin">
      <number>5</number>
     </property>
     <item>
      <widget class="QCheckBox" name="useCOM">
       <property name="cursor">
        <cursorShape>PointingHandCursor</cursorShape>
       </property>
       <property name="text">
        <string>Use Conservation of Mass</string>
       </property>
       <property name="checked">
        <bool>false</bool>
       </property>
      </widget>
```

```
                </item>
               </layout>
              </widget>
             </item>
             <item>
              <widget class="QTextEdit" name="comTextEdit">
               <property name="font">
                <font>
                 <pointsize>8</pointsize>
                </font>
               </property>
               <property name="html">
                <string>&lt;!DOCTYPE HTML PUBLIC &quot;-//W3C//DTD HTML 4.0//EN&quot;
&quot;http://www.w3.org/TR/REC-html40/strict.dtd&quot;&gt;
&lt;html&gt;&lt;head&gt;&lt;meta name=&quot;qrichtext&quot; content=&quot;1&quot; /
&gt;&lt;meta charset=&quot;utf-8&quot; /&gt;&lt;style type=&quot;text/css&quot;&gt;
p, li { white-space: pre-wrap; }
hr { height: 1px; border-width: 0; }
li.unchecked::marker { content: &quot;\2610&quot;; }
li.checked::marker { content: &quot;\2612&quot;; }
&lt;/style&gt;&lt;/head&gt;&lt;body style=&quot; font-family:'Ubuntu'; font-size:
8pt; font-weight:400; font-style:normal;&quot;&gt;
&lt;p style=&quot; margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:
0px; -qt-block-indent:0; text-indent:0px;&quot;&gt;&lt;span style=&quot; font-
family:'Sans Serif'; font-size:9pt;&quot;&gt;This is the native WindNinja solver. It
solves a conservation of mass equation, but not a conservation of momentum equation.
This solver is fast-running, but may give less accurate wind predictions in regions
where momentum effects are important, for example on the lee side of terrain
obstacles.&lt;/span&gt;&lt;/p&gt;&lt;/body&gt;&lt;/html&gt;</string>
               </property>
               <property name="textInteractionFlags">
                <set>Qt::TextInteractionFlag::TextSelectableByKeyboard|
Qt::TextInteractionFlag::TextSelectableByMouse</set>
               </property>
              </widget>
             </item>
             <item>
              <spacer name="verticalSpacer_3">
               <property name="orientation">
                <enum>Qt::Orientation::Vertical</enum>
               </property>
               <property name="sizeHint" stdset="0">
                <size>
                 <width>20</width>
                 <height>40</height>
                </size>
               </property>
              </spacer>
             </item>
            </layout>
           </widget>
           <widget class="QWidget" name="commPage">
            <layout class="QVBoxLayout" name="verticalLayout_4">
             <property name="leftMargin">
```

```
 <number>5</number>
</property>
<property name="topMargin">
 <number>5</number>
</property>
<property name="rightMargin">
 <number>5</number>
</property>
<property name="bottomMargin">
 <number>5</number>
</property>
<item>
 <widget class="QGroupBox" name="commGroupBox">
  <property name="title">
   <string/>
  </property>
  <layout class="QHBoxLayout" name="horizontalLayout_10">
   <property name="spacing">
    <number>0</number>
   </property>
   <property name="leftMargin">
    <number>5</number>
   </property>
   <property name="topMargin">
    <number>5</number>
   </property>
   <property name="rightMargin">
    <number>5</number>
   </property>
   <property name="bottomMargin">
    <number>5</number>
   </property>
   <item>
    <widget class="QCheckBox" name="useCOMM">
     <property name="cursor">
      <cursorShape>PointingHandCursor</cursorShape>
     </property>
     <property name="text">
      <string>Use Conservation of Mass and Momentum</string>
     </property>
    </widget>
   </item>
  </layout>
 </widget>
</item>
<item>
 <widget class="QTextEdit" name="commTextEdit">
  <property name="font">
   <font>
    <pointsize>8</pointsize>
   </font>
  </property>
  <property name="frameShape">
   <enum>QFrame::Shape::NoFrame</enum>
```

```
              </property>
              <property name="html">
               <string>&lt;!DOCTYPE HTML PUBLIC &quot;-//W3C//DTD HTML 4.0//EN&quot;
&quot;http://www.w3.org/TR/REC-html40/strict.dtd&quot;&gt;
&lt;html&gt;&lt;head&gt;&lt;meta name=&quot;qrichtext&quot; content=&quot;1&quot; /
&gt;&lt;meta charset=&quot;utf-8&quot; /&gt;&lt;style type=&quot;text/css&quot;&gt;
p, li { white-space: pre-wrap; }
hr { height: 1px; border-width: 0; }
li.unchecked::marker { content: &quot;\2610&quot;; }
li.checked::marker { content: &quot;\2612&quot;; }
&lt;/style&gt;&lt;/head&gt;&lt;body style=&quot; font-family:'Ubuntu'; font-size:
8pt; font-weight:400; font-style:normal;&quot;&gt;
&lt;p style=&quot; margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:
0px; -qt-block-indent:0; text-indent:0px;&quot;&gt;&lt;span style=&quot; font-
family:'Sans Serif'; font-size:9pt;&quot;&gt;This solver conserves both mass and
momentum. It is based on the OpenFOAM CFD toolbox. This solver should give more
accurate wind predictions in regions where momentum effects are important, such as
on the lee side of terrain obstacles. Because this solver is more computationally
intensive than the conservation of mass solver, simulation times will be longer.
Typical simulation times for this solver range from 10-30 minutes, but will depend
on your domain, resolution, and computational resources. Note that some options
(e.g., point initialization and non-neutral stability) are not available for this
solver at this time. We plan to make these options available in future
releases.&lt;/span&gt;&lt;/p&gt;&lt;/body&gt;&lt;/html&gt;</string>
              </property>
              <property name="textInteractionFlags">
               <set>Qt::TextInteractionFlag::TextSelectableByKeyboard|
Qt::TextInteractionFlag::TextSelectableByMouse</set>
              </property>
             </widget>
            </item>
           </layout>
          </widget>
          <widget class="QWidget" name="inputPage">
           <layout class="QVBoxLayout" name="verticalLayout_12">
            <property name="leftMargin">
             <number>5</number>
            </property>
            <property name="topMargin">
             <number>5</number>
            </property>
            <property name="rightMargin">
             <number>5</number>
            </property>
            <property name="bottomMargin">
             <number>5</number>
            </property>
            <item>
             <widget class="QTextEdit" name="inputTextWindow">
              <property name="font">
               <font>
                <pointsize>8</pointsize>
               </font>
              </property>
```

```
                <property name="frameShape">
                 <enum>QFrame::Shape::NoFrame</enum>
                </property>
                <property name="html">
                 <string>&lt;!DOCTYPE HTML PUBLIC &quot;-//W3C//DTD HTML 4.0//EN&quot;
&quot;http://www.w3.org/TR/REC-html40/strict.dtd&quot;&gt;
&lt;html&gt;&lt;head&gt;&lt;meta name=&quot;qrichtext&quot; content=&quot;1&quot; /
&gt;&lt;meta charset=&quot;utf-8&quot; /&gt;&lt;style type=&quot;text/css&quot;&gt;
p, li { white-space: pre-wrap; }
hr { height: 1px; border-width: 0; }
li.unchecked::marker { content: &quot;\2610&quot;; }
li.checked::marker { content: &quot;\2612&quot;; }
&lt;/style&gt;&lt;/head&gt;&lt;body style=&quot; font-family:'Ubuntu'; font-size:
8pt; font-weight:400; font-style:normal;&quot;&gt;
&lt;p style=&quot; margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:
0px; -qt-block-indent:0; text-indent:0px;&quot;&gt;&lt;span style=&quot; font-size:
10pt;&quot;&gt;This is where input parameters are defined. Analysis requires:&lt;/
span&gt;  &lt;/p&gt;
&lt;p style=&quot; margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:
0px; -qt-block-indent:0; text-indent:0px;&quot;&gt;      &lt;/p&gt;
&lt;p style=&quot; margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:
0px; -qt-block-indent:0; text-indent:0px;&quot;&gt;        &lt;span style=&quot;
font-family:'Sans Serif'; font-size:10pt;&quot;&gt;1. Surface Input (.tif)
file:&lt;/span&gt;&lt;/p&gt;
&lt;p style=&quot; margin-top:0px; margin-bottom:0px; margin-left:30px; margin-
right:0px; -qt-block-indent:1; text-indent:0px;&quot;&gt;&lt;span style=&quot; font-
family:'Sans Serif'; font-size:10pt;&quot;&gt;- Can be uploaded from a computer or
downloaded using the map on the right&lt;/span&gt;        &lt;/p&gt;
&lt;p style=&quot; margin-top:0px; margin-bottom:0px; margin-left:30px; margin-
right:0px; -qt-block-indent:1; text-indent:0px;&quot;&gt;&lt;span style=&quot; font-
family:'Sans Serif'; font-size:10pt;&quot;&gt;- Optional diurnal and stability
analysis&lt;/span&gt;        &lt;/p&gt;
&lt;p style=&quot; margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:
0px; -qt-block-indent:0; text-indent:0px;&quot;&gt;        &lt;span style=&quot;
font-family:'Sans Serif'; font-size:10pt;&quot;&gt;2. Wind Input:&lt;/span&gt;&lt;/
p&gt;
&lt;p style=&quot; margin-top:0px; margin-bottom:0px; margin-left:30px; margin-
right:0px; -qt-block-indent:1; text-indent:0px;&quot;&gt;&lt;span style=&quot; font-
family:'Sans Serif'; font-size:10pt;&quot;&gt;- Requires one (1) of the three (3)
options available: Domain Average Wind, Point Initialization, or Weather Model&lt;/
span&gt;&lt;/p&gt;&lt;/body&gt;&lt;/html&gt;</string>
                </property>
                <property name="textInteractionFlags">
                 <set>Qt::TextInteractionFlag::TextSelectableByKeyboard|
Qt::TextInteractionFlag::TextSelectableByMouse</set>
                </property>
               </widget>
              </item>
             </layout>
            </widget>
            <widget class="QWidget" name="surfaceInputPage">
             <property name="sizePolicy">
              <sizepolicy hsizetype="Preferred" vsizetype="Preferred">
               <horstretch>0</horstretch>
```

```
  <verstretch>0</verstretch>
 </sizepolicy>
</property>
<property name="minimumSize">
 <size>
  <width>0</width>
  <height>0</height>
 </size>
</property>
<layout class="QVBoxLayout" name="verticalLayout_3" stretch="1,1,1,1">
 <property name="sizeConstraint">
  <enum>QLayout::SizeConstraint::SetMinAndMaxSize</enum>
 </property>
 <property name="leftMargin">
  <number>5</number>
 </property>
 <property name="topMargin">
  <number>5</number>
 </property>
 <property name="rightMargin">
  <number>5</number>
 </property>
 <property name="bottomMargin">
  <number>5</number>
 </property>
 <item>
  <widget class="QGroupBox" name="elevInFileBox">
   <property name="sizePolicy">
    <sizepolicy hsizetype="Expanding" vsizetype="Fixed">
     <horstretch>0</horstretch>
     <verstretch>0</verstretch>
    </sizepolicy>
   </property>
   <property name="minimumSize">
    <size>
     <width>0</width>
     <height>70</height>
    </size>
   </property>
   <property name="title">
    <string>Elevation Input File</string>
   </property>
   <layout class="QHBoxLayout" name="horizontalLayout_4">
    <property name="leftMargin">
     <number>5</number>
    </property>
    <property name="topMargin">
     <number>5</number>
    </property>
    <property name="rightMargin">
     <number>5</number>
    </property>
    <property name="bottomMargin">
     <number>5</number>
```

```
</property>
<item>
 <widget class="QLineEdit" name="elevFilePath">
  <property name="sizePolicy">
   <sizepolicy hsizetype="Fixed" vsizetype="Fixed">
    <horstretch>0</horstretch>
    <verstretch>0</verstretch>
   </sizepolicy>
  </property>
  <property name="readOnly">
   <bool>true</bool>
  </property>
  <property name="placeholderText">
   <string>*.tif</string>
  </property>
 </widget>
</item>
<item>
 <widget class="QPushButton" name="openFileButton">
  <property name="cursor">
   <cursorShape>PointingHandCursor</cursorShape>
  </property>
  <property name="text">
   <string>Open File</string>
  </property>
 </widget>
</item>
<item>
 <widget class="QPushButton" name="getFromMapButton">
  <property name="cursor">
   <cursorShape>PointingHandCursor</cursorShape>
  </property>
  <property name="text">
   <string>Import From Map</string>
  </property>
 </widget>
</item>
<item>
 <widget class="QComboBox" name="fetchType">
  <item>
   <property name="text">
    <string>WORLD SRTM (30m)</string>
   </property>
  </item>
  <item>
   <property name="text">
    <string>WORLD GMTED (250m)</string>
   </property>
  </item>
  <item>
   <property name="text">
    <string>Landscape File</string>
   </property>
  </item>
```

```
    </widget>
   </item>
   <item>
    <spacer name="horizontalSpacer_3">
     <property name="orientation">
      <enum>Qt::Orientation::Horizontal</enum>
     </property>
     <property name="sizeHint" stdset="0">
      <size>
       <width>40</width>
       <height>20</height>
      </size>
     </property>
    </spacer>
   </item>
  </layout>
 </widget>
</item>
<item>
 <widget class="QGroupBox" name="vegBox">
  <property name="sizePolicy">
   <sizepolicy hsizetype="Expanding" vsizetype="Fixed">
    <horstretch>0</horstretch>
    <verstretch>0</verstretch>
   </sizepolicy>
  </property>
  <property name="minimumSize">
   <size>
    <width>0</width>
    <height>70</height>
   </size>
  </property>
  <property name="title">
   <string>Vegetation</string>
  </property>
  <layout class="QHBoxLayout" name="horizontalLayout">
   <item>
    <widget class="QComboBox" name="vegetationType">
     <property name="sizePolicy">
      <sizepolicy hsizetype="Fixed" vsizetype="Fixed">
       <horstretch>0</horstretch>
       <verstretch>0</verstretch>
      </sizepolicy>
     </property>
     <property name="cursor">
      <cursorShape>PointingHandCursor</cursorShape>
     </property>
     <item>
      <property name="text">
       <string>Grass</string>
      </property>
     </item>
     <item>
      <property name="text">
```

```
        <string>Brush</string>
       </property>
      </item>
      <item>
       <property name="text">
        <string>Trees</string>
       </property>
      </item>
     </widget>
    </item>
    <item>
     <spacer name="horizontalSpacer">
      <property name="orientation">
       <enum>Qt::Orientation::Horizontal</enum>
      </property>
      <property name="sizeHint" stdset="0">
       <size>
        <width>40</width>
        <height>20</height>
       </size>
      </property>
     </spacer>
    </item>
   </layout>
  </widget>
 </item>
 <item>
  <widget class="QGroupBox" name="meshResBox">
   <property name="sizePolicy">
    <sizepolicy hsizetype="Expanding" vsizetype="Fixed">
     <horstretch>0</horstretch>
     <verstretch>0</verstretch>
    </sizepolicy>
   </property>
   <property name="minimumSize">
    <size>
     <width>0</width>
     <height>70</height>
    </size>
   </property>
   <property name="title">
    <string>Mesh Resolution</string>
   </property>
   <layout class="QHBoxLayout" name="horizontalLayout_5">
    <item>
     <widget class="QComboBox" name="meshResType">
      <property name="sizePolicy">
       <sizepolicy hsizetype="Fixed" vsizetype="Fixed">
        <horstretch>0</horstretch>
        <verstretch>0</verstretch>
       </sizepolicy>
      </property>
      <property name="cursor">
       <cursorShape>PointingHandCursor</cursorShape>
```

```
   </property>
   <item>
    <property name="text">
     <string>Coarse</string>
    </property>
   </item>
   <item>
    <property name="text">
     <string>Medium</string>
    </property>
   </item>
   <item>
    <property name="text">
     <string>Fine</string>
    </property>
   </item>
   <item>
    <property name="text">
     <string>Custom</string>
    </property>
   </item>
  </widget>
 </item>
 <item>
  <widget class="QDoubleSpinBox" name="meshResValue">
   <property name="enabled">
    <bool>false</bool>
   </property>
   <property name="sizePolicy">
    <sizepolicy hsizetype="Fixed" vsizetype="Fixed">
     <horstretch>0</horstretch>
     <verstretch>0</verstretch>
    </sizepolicy>
   </property>
   <property name="cursor">
    <cursorShape>PointingHandCursor</cursorShape>
   </property>
   <property name="readOnly">
    <bool>false</bool>
   </property>
   <property name="minimum">
    <double>0.010000000000000</double>
   </property>
   <property name="maximum">
    <double>10000.000000000000000</double>
   </property>
   <property name="value">
    <double>200.000000000000000</double>
   </property>
  </widget>
 </item>
 <item>
  <widget class="QRadioButton" name="meshResMeters">
   <property name="enabled">
```

```
    <bool>true</bool>
   </property>
   <property name="sizePolicy">
    <sizepolicy hsizetype="Fixed" vsizetype="Fixed">
     <horstretch>0</horstretch>
     <verstretch>0</verstretch>
    </sizepolicy>
   </property>
   <property name="cursor">
    <cursorShape>PointingHandCursor</cursorShape>
   </property>
   <property name="text">
    <string>Meters</string>
   </property>
   <property name="checked">
    <bool>true</bool>
   </property>
  </widget>
 </item>
 <item>
  <widget class="QRadioButton" name="meshResFeet">
   <property name="enabled">
    <bool>true</bool>
   </property>
   <property name="sizePolicy">
    <sizepolicy hsizetype="Fixed" vsizetype="Fixed">
     <horstretch>0</horstretch>
     <verstretch>0</verstretch>
    </sizepolicy>
   </property>
   <property name="cursor">
    <cursorShape>PointingHandCursor</cursorShape>
   </property>
   <property name="text">
    <string>Feet</string>
   </property>
   <property name="checked">
    <bool>false</bool>
   </property>
  </widget>
 </item>
 <item>
  <spacer name="horizontalSpacer_2">
   <property name="orientation">
    <enum>Qt::Orientation::Horizontal</enum>
   </property>
   <property name="sizeHint" stdset="0">
    <size>
     <width>40</width>
     <height>20</height>
    </size>
   </property>
  </spacer>
 </item>
```

```
                </layout>
               </widget>
              </item>
              <item>
               <widget class="QGroupBox" name="timeZoneBox">
                <property name="sizePolicy">
                 <sizepolicy hsizetype="Expanding" vsizetype="Expanding">
                  <horstretch>0</horstretch>
                  <verstretch>0</verstretch>
                 </sizepolicy>
                </property>
                <property name="minimumSize">
                 <size>
                  <width>0</width>
                  <height>0</height>
                 </size>
                </property>
                <property name="title">
                 <string>Time Zone</string>
                </property>
                <layout class="QGridLayout" name="gridLayout_2" rowstretch="0,0,0"
columnstretch="0,0,0,0" rowminimumheight="0,0,0">
                 <item row="0" column="0">
                  <widget class="QComboBox" name="timeZoneSelector">
                   <property name="sizePolicy">
                    <sizepolicy hsizetype="Fixed" vsizetype="Fixed">
                     <horstretch>0</horstretch>
                     <verstretch>0</verstretch>
                    </sizepolicy>
                   </property>
                   <property name="minimumSize">
                    <size>
                     <width>150</width>
                     <height>0</height>
                    </size>
                   </property>
                   <property name="maximumSize">
                    <size>
                     <width>150</width>
                     <height>16777215</height>
                    </size>
                   </property>
                   <property name="cursor">
                    <cursorShape>PointingHandCursor</cursorShape>
                   </property>
                   <property name="currentText">
                    <string/>
                   </property>
                   <property name="currentIndex">
                    <number>-1</number>
                   </property>
                   <property name="maxVisibleItems">
                    <number>10</number>
                   </property>
```

```
                  <property name="sizeAdjustPolicy">
                   <enum>QComboBox::SizeAdjustPolicy::AdjustToContentsOnFirstShow</
enum>
                  </property>
                 </widget>
                </item>
                <item row="2" column="0" colspan="4">
                 <widget class="QTextEdit" name="timeZoneDetails">
                  <property name="sizePolicy">
                   <sizepolicy hsizetype="Expanding" vsizetype="Expanding">
                    <horstretch>0</horstretch>
                    <verstretch>0</verstretch>
                   </sizepolicy>
                  </property>
                  <property name="html">
                   <string>&lt;!DOCTYPE HTML PUBLIC &quot;-//W3C//DTD HTML 4.0//
EN&quot; &quot;http://www.w3.org/TR/REC-html40/strict.dtd&quot;&gt;
&lt;html&gt;&lt;head&gt;&lt;meta name=&quot;qrichtext&quot; content=&quot;1&quot; /
&gt;&lt;meta charset=&quot;utf-8&quot; /&gt;&lt;style type=&quot;text/css&quot;&gt;
p, li { white-space: pre-wrap; }
hr { height: 1px; border-width: 0; }
li.unchecked::marker { content: &quot;\2610&quot;; }
li.checked::marker { content: &quot;\2612&quot;; }
&lt;/style&gt;&lt;/head&gt;&lt;body style=&quot; font-family:'Ubuntu'; font-size:
11pt; font-weight:400; font-style:normal;&quot;&gt;
&lt;p style=&quot; margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:
0px; -qt-block-indent:0; text-indent:0px;&quot;&gt;Standard Name:
Mountain Standard Time&lt;/p&gt;
&lt;p style=&quot; margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:
0px; -qt-block-indent:0; text-indent:0px;&quot;&gt;Daylight Name:
Mountain Daylight Time&lt;/p&gt;
&lt;p style=&quot; margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:
0px; -qt-block-indent:0; text-indent:0px;&quot;&gt;Standard Offset from UTC:
-07:00:00&lt;/p&gt;
&lt;p style=&quot; margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:
0px; -qt-block-indent:0; text-indent:0px;&quot;&gt;Daylight Offset from UTC:
-06:00:00&lt;/p&gt;&lt;/body&gt;&lt;/html&gt;</string>
                  </property>
                  <property name="textInteractionFlags">
                   <set>Qt::TextInteractionFlag::NoTextInteraction</set>
                  </property>
                 </widget>
                </item>
                <item row="0" column="2">
                 <widget class="QCheckBox" name="displayTimeZoneDetails">
                  <property name="sizePolicy">
                   <sizepolicy hsizetype="Fixed" vsizetype="Fixed">
                    <horstretch>0</horstretch>
                    <verstretch>0</verstretch>
                   </sizepolicy>
                  </property>
                  <property name="cursor">
                   <cursorShape>PointingHandCursor</cursorShape>
                  </property>
```

```
        <property name="text">
         <string>Display Time Zone Details</string>
        </property>
       </widget>
      </item>
      <item row="0" column="3">
       <spacer name="horizontalSpacer_4">
        <property name="orientation">
         <enum>Qt::Orientation::Horizontal</enum>
        </property>
        <property name="sizeHint" stdset="0">
         <size>
          <width>40</width>
          <height>20</height>
         </size>
        </property>
       </spacer>
      </item>
      <item row="0" column="1">
       <widget class="QCheckBox" name="showAllTimeZones">
        <property name="sizePolicy">
         <sizepolicy hsizetype="Fixed" vsizetype="Fixed">
          <horstretch>0</horstretch>
          <verstretch>0</verstretch>
         </sizepolicy>
        </property>
        <property name="cursor">
         <cursorShape>PointingHandCursor</cursorShape>
        </property>
        <property name="text">
         <string>Show All Zones</string>
        </property>
       </widget>
      </item>
      <item row="1" column="0">
       <spacer name="verticalSpacer_10">
        <property name="orientation">
         <enum>Qt::Orientation::Vertical</enum>
        </property>
        <property name="sizeHint" stdset="0">
         <size>
          <width>20</width>
          <height>40</height>
         </size>
        </property>
       </spacer>
      </item>
     </layout>
    </widget>
   </item>
  </layout>
 </widget>
 <widget class="QWidget" name="diurnalInputPage">
  <layout class="QVBoxLayout" name="verticalLayout_5">
```

```
<property name="spacing">
 <number>0</number>
</property>
<property name="leftMargin">
 <number>5</number>
</property>
<property name="topMargin">
 <number>5</number>
</property>
<property name="rightMargin">
 <number>5</number>
</property>
<property name="bottomMargin">
 <number>5</number>
</property>
<item>
 <widget class="QGroupBox" name="diurnalBox">
  <property name="sizePolicy">
   <sizepolicy hsizetype="Expanding" vsizetype="Fixed">
    <horstretch>0</horstretch>
    <verstretch>0</verstretch>
   </sizepolicy>
  </property>
  <property name="title">
   <string/>
  </property>
  <layout class="QHBoxLayout" name="horizontalLayout_6">
   <property name="spacing">
    <number>0</number>
   </property>
   <property name="leftMargin">
    <number>5</number>
   </property>
   <property name="topMargin">
    <number>5</number>
   </property>
   <property name="rightMargin">
    <number>5</number>
   </property>
   <property name="bottomMargin">
    <number>5</number>
   </property>
   <item>
    <widget class="QCheckBox" name="useDiurnalWind">
     <property name="sizePolicy">
      <sizepolicy hsizetype="Minimum" vsizetype="Expanding">
       <horstretch>0</horstretch>
       <verstretch>0</verstretch>
      </sizepolicy>
     </property>
     <property name="cursor">
      <cursorShape>PointingHandCursor</cursorShape>
     </property>
     <property name="text">
```

```
          <string>Use Diurnal Wind</string>
         </property>
        </widget>
       </item>
       <item>
        <spacer name="horizontalSpacer_5">
         <property name="orientation">
          <enum>Qt::Orientation::Horizontal</enum>
         </property>
         <property name="sizeHint" stdset="0">
          <size>
           <width>40</width>
           <height>20</height>
          </size>
         </property>
        </spacer>
       </item>
      </layout>
     </widget>
    </item>
    <item>
     <spacer name="verticalSpacer">
      <property name="orientation">
       <enum>Qt::Orientation::Vertical</enum>
      </property>
      <property name="sizeHint" stdset="0">
       <size>
        <width>20</width>
        <height>40</height>
       </size>
      </property>
     </spacer>
    </item>
   </layout>
  </widget>
  <widget class="QWidget" name="stabilityInputPage">
   <layout class="QVBoxLayout" name="verticalLayout_6">
    <property name="spacing">
     <number>0</number>
    </property>
    <property name="leftMargin">
     <number>5</number>
    </property>
    <property name="topMargin">
     <number>5</number>
    </property>
    <property name="rightMargin">
     <number>5</number>
    </property>
    <property name="bottomMargin">
     <number>5</number>
    </property>
    <item>
     <widget class="QGroupBox" name="stabilityBox">
```

```
  <property name="title">
   <string/>
  </property>
  <layout class="QHBoxLayout" name="horizontalLayout_7">
   <property name="spacing">
    <number>0</number>
   </property>
   <property name="leftMargin">
    <number>5</number>
   </property>
   <property name="topMargin">
    <number>5</number>
   </property>
   <property name="rightMargin">
    <number>5</number>
   </property>
   <property name="bottomMargin">
    <number>5</number>
   </property>
   <item>
    <widget class="QCheckBox" name="useStability">
     <property name="cursor">
      <cursorShape>PointingHandCursor</cursorShape>
     </property>
     <property name="text">
      <string>Use Stability</string>
     </property>
    </widget>
   </item>
   <item>
    <spacer name="horizontalSpacer_6">
     <property name="orientation">
      <enum>Qt::Orientation::Horizontal</enum>
     </property>
     <property name="sizeHint" stdset="0">
      <size>
       <width>40</width>
       <height>20</height>
      </size>
     </property>
    </spacer>
   </item>
  </layout>
 </widget>
</item>
<item>
 <spacer name="verticalSpacer_2">
  <property name="orientation">
   <enum>Qt::Orientation::Vertical</enum>
  </property>
  <property name="sizeHint" stdset="0">
   <size>
    <width>20</width>
    <height>40</height>
```

```
                </size>
               </property>
              </spacer>
             </item>
            </layout>
           </widget>
           <widget class="QWidget" name="windInputPage"/>
           <widget class="QWidget" name="domainAvgWndTable">
            <layout class="QVBoxLayout" name="verticalLayout_11" stretch="0">
             <property name="spacing">
              <number>6</number>
             </property>
             <property name="leftMargin">
              <number>5</number>
             </property>
             <property name="topMargin">
              <number>5</number>
             </property>
             <property name="rightMargin">
              <number>5</number>
             </property>
             <property name="bottomMargin">
              <number>5</number>
             </property>
             <item>
              <layout class="QVBoxLayout" name="domainAvgWndLayout"
stretch="1,0,1,1,0">
               <property name="spacing">
                <number>6</number>
               </property>
               <property name="sizeConstraint">
                <enum>QLayout::SizeConstraint::SetMinAndMaxSize</enum>
               </property>
               <property name="topMargin">
                <number>0</number>
               </property>
               <item>
                <layout class="QHBoxLayout" name="domainAvgWindLayout">
                 <property name="sizeConstraint">
                  <enum>QLayout::SizeConstraint::SetDefaultConstraint</enum>
                 </property>
                 <property name="topMargin">
                  <number>0</number>
                 </property>
                 <item>
                  <widget class="QCheckBox" name="useDomainAvgWind">
                   <property name="sizePolicy">
                    <sizepolicy hsizetype="Expanding" vsizetype="Fixed">
                     <horstretch>0</horstretch>
                     <verstretch>0</verstretch>
                    </sizepolicy>
                   </property>
                   <property name="minimumSize">
                    <size>
```

```
        <width>0</width>
        <height>5</height>
       </size>
      </property>
      <property name="cursor">
       <cursorShape>PointingHandCursor</cursorShape>
      </property>
      <property name="text">
       <string>Domain Average Wind</string>
      </property>
     </widget>
    </item>
    <item>
     <spacer name="horizontalSpacer_11">
      <property name="orientation">
       <enum>Qt::Orientation::Horizontal</enum>
      </property>
      <property name="sizeHint" stdset="0">
       <size>
        <width>40</width>
        <height>20</height>
       </size>
      </property>
     </spacer>
    </item>
   </layout>
  </item>
  <item>
   <layout class="QVBoxLayout" name="verticalLayout_13">
    <property name="topMargin">
     <number>10</number>
    </property>
   </layout>
  </item>
  <item>
   <widget class="QGroupBox" name="inputWindHeight">
    <property name="sizePolicy">
     <sizepolicy hsizetype="Preferred" vsizetype="Fixed">
      <horstretch>0</horstretch>
      <verstretch>100</verstretch>
     </sizepolicy>
    </property>
    <property name="title">
     <string>Input Wind Height</string>
    </property>
    <layout class="QHBoxLayout" name="horizontalLayout_2">
     <item>
      <widget class="QComboBox" name="domainAvgPicklist">
       <property name="cursor">
        <cursorShape>PointingHandCursor</cursorShape>
       </property>
       <item>
        <property name="text">
         <string>20ft-US</string>
```

```
   </property>
  </item>
  <item>
   <property name="text">
    <string>10m-SI</string>
   </property>
  </item>
  <item>
   <property name="text">
    <string>Custom</string>
   </property>
  </item>
 </widget>
</item>
<item>
 <widget class="QDoubleSpinBox" name="windHeightValue">
  <property name="enabled">
   <bool>false</bool>
  </property>
  <property name="cursor">
   <cursorShape>PointingHandCursor</cursorShape>
  </property>
  <property name="minimum">
   <double>0.010000000000000</double>
  </property>
  <property name="value">
   <double>20.000000000000000</double>
  </property>
 </widget>
</item>
<item>
 <widget class="QRadioButton" name="windHeightFeet">
  <property name="enabled">
   <bool>false</bool>
  </property>
  <property name="cursor">
   <cursorShape>PointingHandCursor</cursorShape>
  </property>
  <property name="text">
   <string>Feet</string>
  </property>
  <property name="checked">
   <bool>true</bool>
  </property>
 </widget>
</item>
<item>
 <widget class="QRadioButton" name="windHeightMeters">
  <property name="enabled">
   <bool>false</bool>
  </property>
  <property name="cursor">
   <cursorShape>PointingHandCursor</cursorShape>
  </property>
```

```
                    <property name="text">
                     <string>Meters</string>
                    </property>
                   </widget>
                  </item>
                  <item>
                   <spacer name="horizontalSpacer_10">
                    <property name="orientation">
                     <enum>Qt::Orientation::Horizontal</enum>
                    </property>
                    <property name="sizeHint" stdset="0">
                     <size>
                      <width>40</width>
                      <height>20</height>
                     </size>
                    </property>
                   </spacer>
                  </item>
                 </layout>
                </widget>
               </item>
               <item>
                <layout class="QHBoxLayout" name="clearTableLayout"
stretch="1,0,1,4,1">
                 <property name="topMargin">
                  <number>0</number>
                 </property>
                 <property name="bottomMargin">
                  <number>0</number>
                 </property>
                 <item>
                  <widget class="QLineEdit" name="unitTextField">
                   <property name="sizePolicy">
                    <sizepolicy hsizetype="Fixed" vsizetype="Fixed">
                     <horstretch>0</horstretch>
                     <verstretch>0</verstretch>
                    </sizepolicy>
                   </property>
                   <property name="minimumSize">
                    <size>
                     <width>90</width>
                     <height>0</height>
                    </size>
                   </property>
                   <property name="maximumSize">
                    <size>
                     <width>90</width>
                     <height>16777215</height>
                    </size>
                   </property>
                   <property name="text">
                    <string>Table Units:</string>
                   </property>
                   <property name="frame">
```

```
    <bool>false</bool>
   </property>
   <property name="readOnly">
    <bool>true</bool>
   </property>
  </widget>
 </item>
 <item>
  <widget class="QComboBox" name="speedUnits">
   <property name="sizePolicy">
    <sizepolicy hsizetype="Fixed" vsizetype="Fixed">
     <horstretch>0</horstretch>
     <verstretch>0</verstretch>
    </sizepolicy>
   </property>
   <property name="minimumSize">
    <size>
     <width>60</width>
     <height>0</height>
    </size>
   </property>
   <property name="maximumSize">
    <size>
     <width>60</width>
     <height>16777215</height>
    </size>
   </property>
   <property name="cursor">
    <cursorShape>PointingHandCursor</cursorShape>
   </property>
   <item>
    <property name="text">
     <string>mph</string>
    </property>
   </item>
   <item>
    <property name="text">
     <string>m/s</string>
    </property>
   </item>
   <item>
    <property name="text">
     <string>kph</string>
    </property>
   </item>
   <item>
    <property name="text">
     <string>kts</string>
    </property>
   </item>
  </widget>
 </item>
 <item>
  <widget class="QComboBox" name="tempUnits">
```

```
  <property name="sizePolicy">
   <sizepolicy hsizetype="Fixed" vsizetype="Fixed">
    <horstretch>0</horstretch>
    <verstretch>0</verstretch>
   </sizepolicy>
  </property>
  <property name="minimumSize">
   <size>
    <width>40</width>
    <height>0</height>
   </size>
  </property>
  <property name="maximumSize">
   <size>
    <width>40</width>
    <height>16777215</height>
   </size>
  </property>
  <property name="cursor">
   <cursorShape>PointingHandCursor</cursorShape>
  </property>
  <property name="layoutDirection">
   <enum>Qt::LayoutDirection::LeftToRight</enum>
  </property>
  <item>
   <property name="text">
    <string>°F</string>
   </property>
  </item>
  <item>
   <property name="text">
    <string>°C</string>
   </property>
  </item>
 </widget>
</item>
<item>
 <spacer name="horizontalSpacer_12">
  <property name="orientation">
   <enum>Qt::Orientation::Horizontal</enum>
  </property>
  <property name="sizeHint" stdset="0">
   <size>
    <width>40</width>
    <height>20</height>
   </size>
  </property>
 </spacer>
</item>
<item>
 <widget class="QPushButton" name="clearDAWtable">
  <property name="sizePolicy">
   <sizepolicy hsizetype="Fixed" vsizetype="Fixed">
    <horstretch>0</horstretch>
```

```
      <verstretch>0</verstretch>
     </sizepolicy>
    </property>
    <property name="cursor">
     <cursorShape>PointingHandCursor</cursorShape>
    </property>
    <property name="text">
     <string>Clear Table</string>
    </property>
   </widget>
  </item>
 </layout>
</item>
<item>
 <layout class="QVBoxLayout" name="verticalLayout_14" stretch="0">
  <property name="sizeConstraint">
   <enum>QLayout::SizeConstraint::SetMaximumSize</enum>
  </property>
  <property name="topMargin">
   <number>0</number>
  </property>
  <item>
   <widget class="QTableWidget" name="windTableData">
    <property name="sizePolicy">
     <sizepolicy hsizetype="Expanding" vsizetype="Expanding">
      <horstretch>1</horstretch>
      <verstretch>0</verstretch>
     </sizepolicy>
    </property>
    <property name="minimumSize">
     <size>
      <width>100</width>
      <height>0</height>
     </size>
    </property>
    <property name="dragEnabled">
     <bool>true</bool>
    </property>
    <property name="dragDropMode">
     <enum>QAbstractItemView::DragDropMode::DragDrop</enum>
    </property>
    <property name="defaultDropAction">
     <enum>Qt::DropAction::MoveAction</enum>
    </property>
    <property name="wordWrap">
     <bool>true</bool>
    </property>
    <property name="rowCount">
     <number>50</number>
    </property>
    <attribute name="horizontalHeaderCascadingSectionResizes">
     <bool>false</bool>
    </attribute>
    <attribute name="horizontalHeaderMinimumSectionSize">
```

```
 <number>20</number>
</attribute>
<attribute name="horizontalHeaderDefaultSectionSize">
 <number>80</number>
</attribute>
<attribute name="horizontalHeaderStretchLastSection">
 <bool>true</bool>
</attribute>
<attribute name="verticalHeaderCascadingSectionResizes">
 <bool>false</bool>
</attribute>
<attribute name="verticalHeaderMinimumSectionSize">
 <number>20</number>
</attribute>
<attribute name="verticalHeaderDefaultSectionSize">
 <number>20</number>
</attribute>
<attribute name="verticalHeaderStretchLastSection">
 <bool>true</bool>
</attribute>
<row/>
<row/>
<row/>
<row/>
<row/>
<row/>
<row/>
<row/>
<row/>
<row/>
<row/>
<row/>
<row/>
<row/>
<row/>
<row/>
<row/>
<row/>
<row/>
<row/>
<row/>
<row/>
<row/>
<row/>
<row/>
<row/>
<row/>
<row/>
<row/>
<row/>
<row/>
<row/>
<row/>
<row/>
<row/>
<row/>
<row/>
```

```
                    <row/>
                    <row/>
                    <row/>
                    <row/>
                    <row/>
                    <row/>
                    <row/>
                    <row/>
                    <row/>
                    <row/>
                    <row/>
                    <row/>
                    <row/>
                    <row/>
                    <row/>
                    <column>
                     <property name="text">
                      <string>Speed
(Select)</string>
                     </property>
                     <property name="font">
                      <font>
                       <pointsize>10</pointsize>
                      </font>
                     </property>
                    </column>
                    <column>
                     <property name="text">
                      <string>Direction
(degrees)</string>
                     </property>
                     <property name="toolTip">
                      <string extracomment="degrees"/>
                     </property>
                     <property name="font">
                      <font>
                       <pointsize>10</pointsize>
                      </font>
                     </property>
                    </column>
                    <column>
                     <property name="text">
                      <string>Time
(hh:mm)</string>
                     </property>
                     <property name="font">
                      <font>
                       <pointsize>10</pointsize>
                      </font>
                     </property>
                    </column>
                    <column>
                     <property name="text">
```

```
                    <string>Date
(mm/dd/yyyy)</string>
                   </property>
                   <property name="font">
                    <font>
                     <pointsize>10</pointsize>
                    </font>
                   </property>
                  </column>
                  <column>
                   <property name="text">
                    <string>Cloud Cover
(%)</string>
                   </property>
                   <property name="font">
                    <font>
                     <pointsize>10</pointsize>
                    </font>
                   </property>
                  </column>
                  <column>
                   <property name="text">
                    <string>Air Temp
(Select)</string>
                   </property>
                   <property name="font">
                    <font>
                     <pointsize>10</pointsize>
                    </font>
                   </property>
                  </column>
                 </widget>
                </item>
               </layout>
              </item>
             </layout>
            </item>
           </layout>
          </widget>
          <widget class="QWidget" name="pointInitPage">
           <property name="sizePolicy">
            <sizepolicy hsizetype="Expanding" vsizetype="Preferred">
             <horstretch>0</horstretch>
             <verstretch>0</verstretch>
            </sizepolicy>
           </property>
           <property name="maximumSize">
            <size>
             <width>1000</width>
             <height>16777215</height>
            </size>
           </property>
           <layout class="QVBoxLayout" name="verticalLayout_15">
            <property name="leftMargin">
```

```
 <number>5</number>
</property>
<property name="topMargin">
 <number>5</number>
</property>
<property name="rightMargin">
 <number>5</number>
</property>
<property name="bottomMargin">
 <number>5</number>
</property>
<item>
 <widget class="QCheckBox" name="usePointInit">
  <property name="sizePolicy">
   <sizepolicy hsizetype="Expanding" vsizetype="Fixed">
    <horstretch>0</horstretch>
    <verstretch>0</verstretch>
   </sizepolicy>
  </property>
  <property name="minimumSize">
   <size>
    <width>0</width>
    <height>20</height>
   </size>
  </property>
  <property name="maximumSize">
   <size>
    <width>16777215</width>
    <height>20</height>
   </size>
  </property>
  <property name="cursor">
   <cursorShape>PointingHandCursor</cursorShape>
  </property>
  <property name="text">
   <string>Point Initialization</string>
  </property>
 </widget>
</item>
<item>
 <widget class="QGroupBox" name="weatherStationGroup">
  <property name="sizePolicy">
   <sizepolicy hsizetype="Expanding" vsizetype="Preferred">
    <horstretch>1</horstretch>
    <verstretch>0</verstretch>
   </sizepolicy>
  </property>
  <property name="title">
   <string>Select Weather Stations</string>
  </property>
  <layout class="QVBoxLayout" name="verticalLayout_16">
   <property name="leftMargin">
    <number>0</number>
   </property>
```

```
                <property name="topMargin">
                 <number>0</number>
                </property>
                <property name="rightMargin">
                 <number>0</number>
                </property>
                <property name="bottomMargin">
                 <number>0</number>
                </property>
                <item>
                 <layout class="QHBoxLayout" name="selectStationOptions"
stretch="1,0,1">
                  <property name="spacing">
                   <number>0</number>
                  </property>
                  <item>
                   <widget class="QComboBox" name="weatherStationComboBox">
                    <property name="sizePolicy">
                     <sizepolicy hsizetype="Maximum" vsizetype="Fixed">
                      <horstretch>0</horstretch>
                      <verstretch>0</verstretch>
                     </sizepolicy>
                    </property>
                    <property name="maximumSize">
                     <size>
                      <width>200</width>
                      <height>16777215</height>
                     </size>
                    </property>
                    <property name="cursor">
                     <cursorShape>PointingHandCursor</cursorShape>
                    </property>
                    <item>
                     <property name="text">
                      <string>Download from DEM</string>
                     </property>
                    </item>
                    <item>
                     <property name="text">
                      <string>Download by Station ID</string>
                     </property>
                    </item>
                   </widget>
                  </item>
                  <item>
                   <spacer name="horizontalSpacer_13">
                    <property name="orientation">
                     <enum>Qt::Orientation::Horizontal</enum>
                    </property>
                    <property name="sizeHint" stdset="0">
                     <size>
                      <width>40</width>
                      <height>20</height>
                     </size>
```

```
      </property>
     </spacer>
    </item>
    <item>
     <widget class="QComboBox" name="weatherStationTimeComboBox">
      <property name="maximumSize">
       <size>
        <width>250</width>
        <height>16777215</height>
       </size>
      </property>
      <property name="cursor">
       <cursorShape>PointingHandCursor</cursorShape>
      </property>
      <item>
       <property name="text">
        <string>Download Most Recent Data</string>
       </property>
      </item>
      <item>
       <property name="text">
        <string>Download Between Two Dates</string>
       </property>
      </item>
     </widget>
    </item>
   </layout>
  </item>
  <item>
   <layout class="QHBoxLayout" name="selectDataWidget">
    <item>
     <widget class="QStackedWidget" name="downloadSource">
      <property name="minimumSize">
       <size>
        <width>225</width>
        <height>150</height>
       </size>
      </property>
      <property name="maximumSize">
       <size>
        <width>225</width>
        <height>16777215</height>
       </size>
      </property>
      <property name="frameShadow">
       <enum>QFrame::Shadow::Sunken</enum>
      </property>
      <property name="lineWidth">
       <number>0</number>
      </property>
      <property name="currentIndex">
       <number>0</number>
      </property>
      <widget class="QWidget" name="DEM">
```

```
                    <layout class="QVBoxLayout" name="verticalLayout_10">
                     <property name="spacing">
                      <number>0</number>
                     </property>
                     <property name="leftMargin">
                      <number>0</number>
                     </property>
                     <property name="topMargin">
                      <number>0</number>
                     </property>
                     <property name="rightMargin">
                      <number>0</number>
                     </property>
                     <property name="bottomMargin">
                      <number>0</number>
                     </property>
                     <item>
                      <layout class="QGridLayout" name="gridLayout">
                       <item row="0" column="0">
                        <widget class="QGroupBox" name="groupBox">
                         <property name="minimumSize">
                          <size>
                           <width>0</width>
                           <height>50</height>
                          </size>
                         </property>
                         <property name="title">
                          <string>Buffer Around DEM</string>
                         </property>
                         <layout class="QGridLayout" name="gridLayout_3"
columnstretch="0,2,0,0">
                          <property name="leftMargin">
                           <number>0</number>
                          </property>
                          <property name="topMargin">
                           <number>0</number>
                          </property>
                          <property name="rightMargin">
                           <number>0</number>
                          </property>
                          <property name="bottomMargin">
                           <number>0</number>
                          </property>
                          <item row="0" column="1">
                           <widget class="QSpinBox" name="DEMbufferInput">
                            <property name="cursor">
                             <cursorShape>PointingHandCursor</cursorShape>
                            </property>
                           </widget>
                          </item>
                          <item row="0" column="3">
                           <widget class="QComboBox" name="DEMbufferUnits">
                            <property name="cursor">
                             <cursorShape>PointingHandCursor</cursorShape>
```

```
                                </property>
                                <item>
                                 <property name="text">
                                  <string>mi</string>
                                 </property>
                                </item>
                                <item>
                                 <property name="text">
                                  <string>km</string>
                                 </property>
                                </item>
                               </widget>
                              </item>
                              <item row="1" column="1">
                               <widget class="QPushButton"
name="downloadAroundDEMbutton">
                                <property name="text">
                                 <string>Download</string>
                                </property>
                               </widget>
                              </item>
                             </layout>
                            </widget>
                           </item>
                          </layout>
                         </item>
                         <item>
                          <spacer name="verticalSpacer_5">
                           <property name="orientation">
                            <enum>Qt::Orientation::Vertical</enum>
                           </property>
                           <property name="sizeHint" stdset="0">
                            <size>
                             <width>20</width>
                             <height>40</height>
                            </size>
                           </property>
                          </spacer>
                         </item>
                        </layout>
                       </widget>
                       <widget class="QWidget" name="stationID">
                        <layout class="QVBoxLayout" name="verticalLayout_19">
                         <property name="spacing">
                          <number>0</number>
                         </property>
                         <property name="leftMargin">
                          <number>0</number>
                         </property>
                         <property name="topMargin">
                          <number>0</number>
                         </property>
                         <property name="rightMargin">
                          <number>0</number>
```

```
</property>
<property name="bottomMargin">
 <number>0</number>
</property>
<item>
 <layout class="QVBoxLayout" name="verticalLayout_17">
  <property name="spacing">
   <number>0</number>
  </property>
  <item>
   <widget class="QGroupBox" name="groupBox_2">
    <property name="sizePolicy">
     <sizepolicy hsizetype="Preferred" vsizetype="Fixed">
      <horstretch>0</horstretch>
      <verstretch>0</verstretch>
     </sizepolicy>
    </property>
    <property name="minimumSize">
     <size>
      <width>0</width>
      <height>50</height>
     </size>
    </property>
    <property name="maximumSize">
     <size>
      <width>16777215</width>
      <height>50</height>
     </size>
    </property>
    <property name="title">
     <string>Input Station IDs</string>
    </property>
    <layout class="QVBoxLayout" name="verticalLayout_20">
     <property name="spacing">
      <number>0</number>
     </property>
     <property name="leftMargin">
      <number>0</number>
     </property>
     <property name="topMargin">
      <number>0</number>
     </property>
     <property name="rightMargin">
      <number>0</number>
     </property>
     <property name="bottomMargin">
      <number>0</number>
     </property>
     <item>
      <widget class="QLineEdit" name="stationInputID">
       <property name="minimumSize">
        <size>
         <width>0</width>
         <height>20</height>
```

```
                    </size>
                   </property>
                   <property name="placeholderText">
                    <string>KMSO, PNTM8</string>
                   </property>
                   <property name="clearButtonEnabled">
                    <bool>false</bool>
                   </property>
                  </widget>
                 </item>
                 <item>
                  <widget class="QPushButton" name="downloadPointInitData">
                   <property name="minimumSize">
                    <size>
                     <width>0</width>
                     <height>25</height>
                    </size>
                   </property>
                   <property name="maximumSize">
                    <size>
                     <width>140</width>
                     <height>16777215</height>
                    </size>
                   </property>
                   <property name="cursor">
                    <cursorShape>PointingHandCursor</cursorShape>
                   </property>
                   <property name="text">
                    <string>Download Data</string>
                   </property>
                  </widget>
                 </item>
                </layout>
               </widget>
              </item>
              <item>
               <spacer name="verticalSpacer_6">
                <property name="orientation">
                 <enum>Qt::Orientation::Vertical</enum>
                </property>
                <property name="sizeHint" stdset="0">
                 <size>
                  <width>20</width>
                  <height>40</height>
                 </size>
                </property>
               </spacer>
              </item>
             </layout>
            </item>
           </layout>
          </widget>
         </widget>
        </item>
```

```
                    <item>
                     <spacer name="horizontalSpacer_7">
                      <property name="orientation">
                       <enum>Qt::Orientation::Horizontal</enum>
                      </property>
                      <property name="sizeHint" stdset="0">
                       <size>
                        <width>40</width>
                        <height>20</height>
                       </size>
                      </property>
                     </spacer>
                    </item>
                    <item>
                     <widget class="QStackedWidget" name="downloadTimeData">
                      <property name="minimumSize">
                       <size>
                        <width>225</width>
                        <height>0</height>
                       </size>
                      </property>
                      <property name="maximumSize">
                       <size>
                        <width>225</width>
                        <height>16777215</height>
                       </size>
                      </property>
                      <property name="currentIndex">
                       <number>0</number>
                      </property>
                      <widget class="QWidget" name="mostRecent"/>
                      <widget class="QWidget" name="btwnTwoDates">
                       <property name="sizePolicy">
                        <sizepolicy hsizetype="Preferred" vsizetype="Minimum">
                         <horstretch>0</horstretch>
                         <verstretch>0</verstretch>
                        </sizepolicy>
                       </property>
                       <property name="minimumSize">
                        <size>
                         <width>0</width>
                         <height>150</height>
                        </size>
                       </property>
                       <layout class="QVBoxLayout" name="verticalLayout_21"
stretch="1,1,1">
                        <property name="spacing">
                         <number>10</number>
                        </property>
                        <property name="leftMargin">
                         <number>0</number>
                        </property>
                        <property name="topMargin">
                         <number>0</number>
```

```
</property>
<property name="rightMargin">
 <number>0</number>
</property>
<property name="bottomMargin">
 <number>0</number>
</property>
<item>
 <widget class="QGroupBox" name="groupBox_3">
  <property name="sizePolicy">
   <sizepolicy hsizetype="Preferred" vsizetype="Fixed">
    <horstretch>0</horstretch>
    <verstretch>0</verstretch>
   </sizepolicy>
  </property>
  <property name="minimumSize">
   <size>
    <width>0</width>
    <height>70</height>
   </size>
  </property>
  <property name="title">
   <string>Start Time</string>
  </property>
  <layout class="QVBoxLayout" name="verticalLayout_22">
   <item>
    <widget class="QDateTimeEdit" name="dateTimeEdit">
     <property name="cursor">
      <cursorShape>PointingHandCursor</cursorShape>
     </property>
     <property name="calendarPopup">
      <bool>true</bool>
     </property>
    </widget>
   </item>
  </layout>
 </widget>
</item>
<item>
 <widget class="QGroupBox" name="groupBox_4">
  <property name="sizePolicy">
   <sizepolicy hsizetype="Preferred" vsizetype="Fixed">
    <horstretch>0</horstretch>
    <verstretch>0</verstretch>
   </sizepolicy>
  </property>
  <property name="minimumSize">
   <size>
    <width>0</width>
    <height>70</height>
   </size>
  </property>
  <property name="maximumSize">
   <size>
```

```
            <width>16777215</width>
            <height>70</height>
           </size>
          </property>
          <property name="cursor">
           <cursorShape>PointingHandCursor</cursorShape>
          </property>
          <property name="title">
           <string>End Time</string>
          </property>
          <layout class="QVBoxLayout" name="verticalLayout_23">
           <item>
            <widget class="QDateTimeEdit" name="dateTimeEdit_2">
             <property name="calendarPopup">
              <bool>true</bool>
             </property>
            </widget>
           </item>
          </layout>
         </widget>
        </item>
        <item>
         <spacer name="verticalSpacer_7">
          <property name="orientation">
           <enum>Qt::Orientation::Vertical</enum>
          </property>
          <property name="sizeHint" stdset="0">
           <size>
            <width>20</width>
            <height>40</height>
           </size>
          </property>
         </spacer>
        </item>
       </layout>
      </widget>
     </widget>
    </item>
   </layout>
  </item>
  <item>
   <layout class="QHBoxLayout" name="horizontalLayout_17">
    <item>
     <widget class="QGroupBox" name="groupBox_5">
      <property name="title">
       <string>Start Time</string>
      </property>
      <layout class="QHBoxLayout" name="horizontalLayout_18">
       <property name="spacing">
        <number>0</number>
       </property>
       <property name="leftMargin">
        <number>0</number>
       </property>
```

```
    <property name="topMargin">
     <number>0</number>
    </property>
    <property name="rightMargin">
     <number>0</number>
    </property>
    <property name="bottomMargin">
     <number>0</number>
    </property>
    <item>
     <widget class="QDateTimeEdit" name="pointInitStartTime">
      <property name="cursor">
       <cursorShape>PointingHandCursor</cursorShape>
      </property>
     </widget>
    </item>
   </layout>
  </widget>
 </item>
 <item>
  <widget class="QGroupBox" name="groupBox_6">
   <property name="title">
    <string>Stop Time</string>
   </property>
   <layout class="QHBoxLayout" name="horizontalLayout_19">
    <property name="spacing">
     <number>0</number>
    </property>
    <property name="leftMargin">
     <number>0</number>
    </property>
    <property name="topMargin">
     <number>0</number>
    </property>
    <property name="rightMargin">
     <number>0</number>
    </property>
    <property name="bottomMargin">
     <number>0</number>
    </property>
    <item>
     <widget class="QDateTimeEdit" name="pointInitStopTime">
      <property name="cursor">
       <cursorShape>PointingHandCursor</cursorShape>
      </property>
     </widget>
    </item>
   </layout>
  </widget>
 </item>
 <item>
  <widget class="QGroupBox" name="groupBox_7">
   <property name="title">
    <string>Number of Time Steps</string>
```

```
    </property>
    <layout class="QHBoxLayout" name="horizontalLayout_20">
     <property name="spacing">
      <number>0</number>
     </property>
     <property name="leftMargin">
      <number>0</number>
     </property>
     <property name="topMargin">
      <number>0</number>
     </property>
     <property name="rightMargin">
      <number>0</number>
     </property>
     <property name="bottomMargin">
      <number>0</number>
     </property>
     <item>
      <widget class="QSpinBox" name="pointInitTimeSteps">
       <property name="cursor">
        <cursorShape>PointingHandCursor</cursorShape>
       </property>
       <property name="minimum">
        <number>1</number>
       </property>
       <property name="maximum">
        <number>999999</number>
       </property>
       <property name="value">
        <number>24</number>
       </property>
      </widget>
     </item>
    </layout>
   </widget>
  </item>
 </layout>
</item>
<item>
 <widget class="QCheckBox" name="writeStationKML">
  <property name="cursor">
   <cursorShape>PointingHandCursor</cursorShape>
  </property>
  <property name="text">
   <string>Write Station KML</string>
  </property>
 </widget>
</item>
<item>
 <widget class="QTreeView" name="treeFileExplorer">
  <property name="sortingEnabled">
   <bool>true</bool>
  </property>
  <attribute name="headerMinimumSectionSize">
```

```
          <number>10</number>
         </attribute>
         <attribute name="headerDefaultSectionSize">
          <number>300</number>
         </attribute>
         <attribute name="headerShowSortIndicator" stdset="0">
          <bool>true</bool>
         </attribute>
         <attribute name="headerStretchLastSection">
          <bool>false</bool>
         </attribute>
        </widget>
       </item>
      </layout>
     </widget>
    </item>
   </layout>
  </widget>
  <widget class="QWidget" name="weatherModelPage">
   <layout class="QVBoxLayout" name="verticalLayout_24">
    <property name="leftMargin">
     <number>5</number>
    </property>
    <property name="topMargin">
     <number>5</number>
    </property>
    <property name="rightMargin">
     <number>5</number>
    </property>
    <property name="bottomMargin">
     <number>5</number>
    </property>
    <item>
     <widget class="QCheckBox" name="useWeatherModelInit">
      <property name="sizePolicy">
       <sizepolicy hsizetype="Expanding" vsizetype="Fixed">
        <horstretch>0</horstretch>
        <verstretch>0</verstretch>
       </sizepolicy>
      </property>
      <property name="minimumSize">
       <size>
        <width>0</width>
        <height>20</height>
       </size>
      </property>
      <property name="maximumSize">
       <size>
        <width>16777215</width>
        <height>20</height>
       </size>
      </property>
      <property name="cursor">
       <cursorShape>PointingHandCursor</cursorShape>
```

```
   </property>
   <property name="text">
    <string>Weather Model Initialization</string>
   </property>
  </widget>
 </item>
 <item>
  <widget class="QGroupBox" name="downloadWeatherDataGroup">
   <property name="title">
    <string>Download Weather Data</string>
   </property>
   <property name="checkable">
    <bool>false</bool>
   </property>
   <layout class="QVBoxLayout" name="verticalLayout_25" stretch="1,5">
    <property name="leftMargin">
     <number>0</number>
    </property>
    <property name="topMargin">
     <number>0</number>
    </property>
    <property name="rightMargin">
     <number>0</number>
    </property>
    <property name="bottomMargin">
     <number>0</number>
    </property>
    <item>
     <layout class="QHBoxLayout" name="horizontalLayout_8"
stretch="2,1,1">
      <item>
       <widget class="QComboBox" name="weatherDataSelector">
        <property name="cursor">
         <cursorShape>PointingHandCursor</cursorShape>
        </property>
        <item>
         <property name="text">
          <string>UCAR-NDFD-CONUS-2.5-KM</string>
         </property>
        </item>
        <item>
         <property name="text">
          <string>UCAR-NAM-CONUS-12-KM</string>
         </property>
        </item>
        <item>
         <property name="text">
          <string>UCAR-RAP-CONUS-13-KM</string>
         </property>
        </item>
        <item>
         <property name="text">
          <string>UCAR-NAM-ALASKA-11-KM</string>
         </property>
```

```
</item>
<item>
 <property name="text">
  <string>UCAR-GFS-GLOBAL-0.5-DEG</string>
 </property>
</item>
<item>
 <property name="text">
  <string>NOMADS-HIRES-ARW-ALASKA-5-KM</string>
 </property>
</item>
<item>
 <property name="text">
  <string>NOMADS-HIRES-FV3-ALASKA-5-KM</string>
 </property>
</item>
<item>
 <property name="text">
  <string>NOMADS-HIRES-ARW-CONUS-5-KM</string>
 </property>
</item>
<item>
 <property name="text">
  <string>NOMADS-HIRES-FV3-CONUS-5-KM</string>
 </property>
</item>
<item>
 <property name="text">
  <string>NOMADS-HIRES-GUAM-5-KM</string>
 </property>
</item>
<item>
 <property name="text">
  <string>NOMADS-HIRES-HAWAII-5-KM</string>
 </property>
</item>
<item>
 <property name="text">
  <string>NOMADS-HIRES-PUERTO-RICO-5-KM</string>
 </property>
</item>
<item>
 <property name="text">
  <string>NOMADS-NAM-ALASKA-11.25-KM</string>
 </property>
</item>
<item>
 <property name="text">
  <string>NOMADS-NAM-CONUS-12-KM</string>
 </property>
</item>
<item>
 <property name="text">
  <string>NOMADS-NAM-NORTH-AMERICA-32-KM</string>
```

```
   </property>
  </item>
  <item>
   <property name="text">
    <string>NOMADS-NAM-NEST-ALASKA-3-KM</string>
   </property>
  </item>
  <item>
   <property name="text">
    <string>NOMADS-NAM-NEST-CONUS-3-KM</string>
   </property>
  </item>
  <item>
   <property name="text">
    <string>NOMADS-NAM-HAWAII-3-KM</string>
   </property>
  </item>
  <item>
   <property name="text">
    <string>NOMADS-NAM-NEST-PUERTO-RICO-3-KM</string>
   </property>
  </item>
  <item>
   <property name="text">
    <string>NOMADS-HRRR-ALASKA-3-KM</string>
   </property>
  </item>
  <item>
   <property name="text">
    <string>NOMADS-HRRR-CONUS-3-KM</string>
   </property>
  </item>
  <item>
   <property name="text">
    <string>NOMADS-HRRR-CONUS-SUBHOURLY-3-KM</string>
   </property>
  </item>
  <item>
   <property name="text">
    <string>NOMADS-HRRR-ALASKA-SUBHOURLY-3-KM</string>
   </property>
  </item>
  <item>
   <property name="text">
    <string>NOMADS-RAP-CONUS-13-KM</string>
   </property>
  </item>
  <item>
   <property name="text">
    <string>NOMADS-RAP-NORTH-AMERICA-32-KM</string>
   </property>
  </item>
 </widget>
</item>
```

```
   <item>
    <widget class="QSpinBox" name="weatherDataValue">
     <property name="cursor">
      <cursorShape>PointingHandCursor</cursorShape>
     </property>
     <property name="buttonSymbols">
      <enum>QAbstractSpinBox::ButtonSymbols::UpDownArrows</enum>
     </property>
     <property name="specialValueText">
      <string/>
     </property>
     <property name="suffix">
      <string> hour(s)</string>
     </property>
     <property name="minimum">
      <number>1</number>
     </property>
    </widget>
   </item>
   <item>
    <widget class="QPushButton" name="downloadWeatherData">
     <property name="cursor">
      <cursorShape>PointingHandCursor</cursorShape>
     </property>
     <property name="text">
      <string>Download Data</string>
     </property>
    </widget>
   </item>
  </layout>
 </item>
 <item>
  <widget class="QGroupBox" name="downloadedForecastGroup">
   <property name="title">
    <string>Downloaded Forecasts</string>
   </property>
   <layout class="QVBoxLayout" name="verticalLayout" stretch="1,1">
    <property name="spacing">
     <number>5</number>
    </property>
    <property name="leftMargin">
     <number>0</number>
    </property>
    <property name="topMargin">
     <number>0</number>
    </property>
    <property name="rightMargin">
     <number>0</number>
    </property>
    <property name="bottomMargin">
     <number>0</number>
    </property>
    <item>
     <widget class="QTreeView" name="forecastDownloads">
```

```
            <property name="sortingEnabled">
             <bool>true</bool>
            </property>
           </widget>
          </item>
          <item>
           <widget class="QGroupBox" name="timeStepGroupbox">
            <property name="cursor">
             <cursorShape>PointingHandCursor</cursorShape>
            </property>
            <property name="title">
             <string>Select Specific Time Steps</string>
            </property>
            <property name="checkable">
             <bool>true</bool>
            </property>
            <property name="checked">
             <bool>false</bool>
            </property>
           </widget>
          </item>
         </layout>
        </widget>
       </item>
      </layout>
     </widget>
    </item>
   </layout>
  </widget>
 </widget>
 <widget class="QWidget" name="solverPage">
  <layout class="QVBoxLayout" name="verticalLayout_7">
   <property name="sizeConstraint">
    <enum>QLayout::SizeConstraint::SetDefaultConstraint</enum>
   </property>
   <property name="leftMargin">
    <number>5</number>
   </property>
   <property name="topMargin">
    <number>5</number>
   </property>
   <property name="rightMargin">
    <number>5</number>
   </property>
   <property name="bottomMargin">
    <number>5</number>
   </property>
   <item>
    <widget class="QPlainTextEdit" name="availableProcessors">
     <property name="sizePolicy">
      <sizepolicy hsizetype="Expanding" vsizetype="Fixed">
       <horstretch>0</horstretch>
       <verstretch>0</verstretch>
      </sizepolicy>
     </property>
```

```
  <property name="maximumSize">
   <size>
    <width>16777215</width>
    <height>25</height>
   </size>
  </property>
  <property name="baseSize">
   <size>
    <width>0</width>
    <height>0</height>
   </size>
  </property>
  <property name="frameShape">
   <enum>QFrame::Shape::NoFrame</enum>
  </property>
  <property name="verticalScrollBarPolicy">
   <enum>Qt::ScrollBarPolicy::ScrollBarAlwaysOff</enum>
  </property>
  <property name="horizontalScrollBarPolicy">
   <enum>Qt::ScrollBarPolicy::ScrollBarAlwaysOff</enum>
  </property>
  <property name="readOnly">
   <bool>true</bool>
  </property>
 </widget>
</item>
<item>
 <layout class="QHBoxLayout" name="processorLayout">
  <property name="spacing">
   <number>0</number>
  </property>
  <property name="topMargin">
   <number>0</number>
  </property>
  <item>
   <widget class="QPlainTextEdit" name="numberOfProcessorsText">
    <property name="sizePolicy">
     <sizepolicy hsizetype="Fixed" vsizetype="Fixed">
      <horstretch>0</horstretch>
      <verstretch>0</verstretch>
     </sizepolicy>
    </property>
    <property name="maximumSize">
     <size>
      <width>175</width>
      <height>25</height>
     </size>
    </property>
    <property name="baseSize">
     <size>
      <width>0</width>
      <height>0</height>
     </size>
    </property>
```

```
    <property name="frameShape">
     <enum>QFrame::Shape::NoFrame</enum>
    </property>
    <property name="verticalScrollBarPolicy">
     <enum>Qt::ScrollBarPolicy::ScrollBarAlwaysOff</enum>
    </property>
    <property name="horizontalScrollBarPolicy">
     <enum>Qt::ScrollBarPolicy::ScrollBarAlwaysOff</enum>
    </property>
    <property name="readOnly">
     <bool>true</bool>
    </property>
    <property name="plainText">
     <string>Number of Processors:</string>
    </property>
   </widget>
  </item>
  <item>
   <widget class="QSpinBox" name="numProcessorsSpinbox">
    <property name="cursor">
     <cursorShape>PointingHandCursor</cursorShape>
    </property>
    <property name="minimum">
     <number>1</number>
    </property>
   </widget>
  </item>
  <item>
   <spacer name="horizontalSpacer_8">
    <property name="orientation">
     <enum>Qt::Orientation::Horizontal</enum>
    </property>
    <property name="sizeHint" stdset="0">
     <size>
      <width>40</width>
      <height>20</height>
     </size>
    </property>
   </spacer>
  </item>
 </layout>
</item>
<item>
 <widget class="QPlainTextEdit" name="outputDirText">
  <property name="sizePolicy">
   <sizepolicy hsizetype="Expanding" vsizetype="Fixed">
    <horstretch>0</horstretch>
    <verstretch>0</verstretch>
   </sizepolicy>
  </property>
  <property name="maximumSize">
   <size>
    <width>16777215</width>
    <height>25</height>
```

```
    </size>
   </property>
   <property name="frameShape">
    <enum>QFrame::Shape::NoFrame</enum>
   </property>
   <property name="verticalScrollBarPolicy">
    <enum>Qt::ScrollBarPolicy::ScrollBarAlwaysOff</enum>
   </property>
   <property name="horizontalScrollBarPolicy">
    <enum>Qt::ScrollBarPolicy::ScrollBarAlwaysOff</enum>
   </property>
   <property name="readOnly">
    <bool>true</bool>
   </property>
   <property name="plainText">
    <string>Output Directory</string>
   </property>
  </widget>
 </item>
 <item>
  <layout class="QHBoxLayout" name="outputLayout">
   <property name="spacing">
    <number>10</number>
   </property>
   <item>
    <widget class="QTextEdit" name="outputDirectory">
     <property name="sizePolicy">
      <sizepolicy hsizetype="Fixed" vsizetype="Fixed">
       <horstretch>0</horstretch>
       <verstretch>0</verstretch>
      </sizepolicy>
     </property>
     <property name="maximumSize">
      <size>
       <width>300</width>
       <height>30</height>
      </size>
     </property>
     <property name="verticalScrollBarPolicy">
      <enum>Qt::ScrollBarPolicy::ScrollBarAlwaysOff</enum>
     </property>
     <property name="horizontalScrollBarPolicy">
      <enum>Qt::ScrollBarPolicy::ScrollBarAlwaysOff</enum>
     </property>
     <property name="readOnly">
      <bool>true</bool>
     </property>
    </widget>
   </item>
   <item>
    <widget class="QPushButton" name="outputSaveLocationBtn">
     <property name="cursor">
      <cursorShape>PointingHandCursor</cursorShape>
     </property>
```

```
    <property name="text">
     <string>Output Location</string>
    </property>
   </widget>
  </item>
  <item>
   <spacer name="horizontalSpacer_9">
    <property name="orientation">
     <enum>Qt::Orientation::Horizontal</enum>
    </property>
    <property name="sizeHint" stdset="0">
     <size>
      <width>40</width>
      <height>20</height>
     </size>
    </property>
   </spacer>
  </item>
 </layout>
</item>
<item>
 <spacer name="verticalSpacer_9">
  <property name="orientation">
   <enum>Qt::Orientation::Vertical</enum>
  </property>
  <property name="sizeType">
   <enum>QSizePolicy::Policy::Fixed</enum>
  </property>
  <property name="sizeHint" stdset="0">
   <size>
    <width>20</width>
    <height>10</height>
   </size>
  </property>
 </spacer>
</item>
<item>
 <widget class="QPushButton" name="solverPageSolveBtn">
  <property name="enabled">
   <bool>false</bool>
  </property>
  <property name="sizePolicy">
   <sizepolicy hsizetype="Fixed" vsizetype="Fixed">
    <horstretch>0</horstretch>
    <verstretch>0</verstretch>
   </sizepolicy>
  </property>
  <property name="cursor">
   <cursorShape>PointingHandCursor</cursorShape>
  </property>
  <property name="text">
   <string>Solve</string>
  </property>
 </widget>
```

```
    </item>
    <item>
     <spacer name="verticalSpacer_8">
      <property name="orientation">
       <enum>Qt::Orientation::Vertical</enum>
      </property>
      <property name="sizeHint" stdset="0">
       <size>
        <width>20</width>
        <height>40</height>
       </size>
      </property>
     </spacer>
    </item>
   </layout>
  </widget>
 </widget>
</item>
<item>
 <layout class="QHBoxLayout" name="solveButtonLayout" stretch="1,1,1">
  <property name="spacing">
   <number>0</number>
  </property>
  <property name="topMargin">
   <number>3</number>
  </property>
  <property name="bottomMargin">
   <number>3</number>
  </property>
  <item>
   <layout class="QHBoxLayout" name="horizontalLayout_16">
    <property name="spacing">
     <number>0</number>
    </property>
    <item>
     <widget class="QWidget" name="widget" native="true"/>
    </item>
   </layout>
  </item>
  <item>
   <layout class="QHBoxLayout" name="horizontalLayout_13">
    <item>
     <widget class="QPushButton" name="solveButton">
      <property name="enabled">
       <bool>false</bool>
      </property>
      <property name="sizePolicy">
       <sizepolicy hsizetype="Fixed" vsizetype="Fixed">
        <horstretch>0</horstretch>
        <verstretch>0</verstretch>
       </sizepolicy>
      </property>
      <property name="maximumSize">
       <size>
```

```
                <width>16777215</width>
                <height>16777215</height>
               </size>
              </property>
              <property name="cursor">
               <cursorShape>PointingHandCursor</cursorShape>
              </property>
              <property name="layoutDirection">
               <enum>Qt::LayoutDirection::LeftToRight</enum>
              </property>
              <property name="styleSheet">
               <string notr="true">#solveButton {
        padding: 0;
        width: 100px;
        text-align: center;
        padding: 5px;
}</string>
              </property>
              <property name="text">
               <string>Solve</string>
              </property>
             </widget>
            </item>
           </layout>
          </item>
          <item>
           <layout class="QHBoxLayout" name="horizontalLayout_12">
            <item>
             <widget class="QWidget" name="widget_2" native="true">
              <property name="enabled">
               <bool>true</bool>
              </property>
             </widget>
            </item>
           </layout>
          </item>
         </layout>
        </item>
       </layout>
      </widget>
      <widget class="QWidget" name="mapPanelWidget" native="true">
       <property name="sizePolicy">
        <sizepolicy hsizetype="Expanding" vsizetype="Expanding">
         <horstretch>1</horstretch>
         <verstretch>0</verstretch>
        </sizepolicy>
       </property>
       <property name="minimumSize">
        <size>
         <width>100</width>
         <height>0</height>
        </size>
       </property>
       <property name="baseSize">
```

```
        <size>
         <width>0</width>
         <height>0</height>
        </size>
       </property>
       <property name="palette">
        <palette>
         <active>
          <colorrole role="ButtonText">
           <brush brushstyle="SolidPattern">
            <color alpha="255">
             <red>0</red>
             <green>0</green>
             <blue>0</blue>
            </color>
           </brush>
          </colorrole>
         </active>
         <inactive>
          <colorrole role="ButtonText">
           <brush brushstyle="SolidPattern">
            <color alpha="255">
             <red>0</red>
             <green>0</green>
             <blue>0</blue>
            </color>
           </brush>
          </colorrole>
         </inactive>
         <disabled/>
        </palette>
       </property>
       <property name="styleSheet">
        <string notr="true"/>
       </property>
      </widget>
     </widget>
    </item>
   </layout>
  </widget>
  <widget class="QMenuBar" name="menuBar">
   <property name="geometry">
    <rect>
     <x>0</x>
     <y>0</y>
     <width>1081</width>
     <height>22</height>
    </rect>
   </property>
   <property name="styleSheet">
    <string notr="true">QMenuBar {
        background-color: #818181
}
QMenu::item:selected {
```

```
    background: #3c3c3c;
}</string>
   </property>
   <widget class="QMenu" name="menuFile">
    <property name="title">
     <string>File</string>
    </property>
    <addaction name="actionNew_Project"/>
    <addaction name="actionOpen_Project"/>
    <addaction name="actionExport_Solution"/>
    <addaction name="actionClose_Project"/>
    <addaction name="actionExit_WindNinja"/>
   </widget>
   <widget class="QMenu" name="menuOptions">
    <property name="title">
     <string>Options</string>
    </property>
    <addaction name="actionConsole_Output"/>
   </widget>
   <widget class="QMenu" name="menuTools">
    <property name="title">
     <string>Tools</string>
    </property>
    <addaction name="actionWrite_blank_station_file"/>
    <addaction name="actionSet_configuration_option"/>
   </widget>
   <widget class="QMenu" name="menuHelp">
    <property name="title">
     <string>Help</string>
    </property>
    <widget class="QMenu" name="menuDisplaying_Shapefiles">
     <property name="title">
      <string>Displaying Shapefiles</string>
     </property>
     <addaction name="actionHow_to_Display_Shapefiles_in_ArcMap"/>
    </widget>
    <widget class="QMenu" name="menuTutorials">
     <property name="title">
      <string>Tutorials</string>
     </property>
     <addaction name="actionTutorial_1_The_Basics"/>
     <addaction name="actionTutorial_2_Diurnal_Winds_and_Non_Neutral_Stability"/>
     <addaction name="actionTutorial_3_Point_Initialization"/>
     <addaction name="actionTutorial_4_Weather_Model_Initialization"/>
    </widget>
    <widget class="QMenu" name="menuInstructions">
     <property name="title">
      <string>Instructions</string>
     </property>
     <addaction name="actionDEM_Download"/>
     <addaction name="actionfetch_dem_Instructions"/>
     <addaction name="actionCommand_Line_Interface"/>
    </widget>
    <addaction name="menuDisplaying_Shapefiles"/>
```

```
   <addaction name="menuTutorials"/>
   <addaction name="menuInstructions"/>
   <addaction name="actionAbout_WindNinja"/>
   <addaction name="actionCitation"/>
   <addaction name="actionEmail_Us"/>
   <addaction name="actionSubmit_Bug_Report"/>
  </widget>
  <addaction name="menuFile"/>
  <addaction name="menuOptions"/>
  <addaction name="menuTools"/>
  <addaction name="menuHelp"/>
 </widget>
 <action name="actionConsole_Output">
  <property name="text">
   <string>Console Output</string>
  </property>
 </action>
 <action name="actionWrite_blank_station_file">
  <property name="text">
   <string>Write blank station file</string>
  </property>
 </action>
 <action name="actionSet_configuration_option">
  <property name="text">
   <string>Set configuration option</string>
  </property>
 </action>
 <action name="actionDEM_Download_Instructions">
  <property name="text">
   <string>Displaying Shapefiles</string>
  </property>
 </action>
 <action name="actionHow_to_Display_Shapefiles_in_ArcMap">
  <property name="text">
   <string>How to Display Shapefiles in ArcMap</string>
  </property>
 </action>
 <action name="actionTutorial_1_The_Basics">
  <property name="text">
   <string>Tutorial 1: The Basics</string>
  </property>
 </action>
 <action name="actionTutorial_2_Diurnal_Winds_and_Non_Neutral_Stability">
  <property name="text">
   <string>Tutorial 2: Diurnal Winds and Non-Neutral Stability</string>
  </property>
 </action>
 <action name="actionTutorial_3_Point_Initialization">
  <property name="text">
   <string>Tutorial 3: Point Initialization</string>
  </property>
 </action>
 <action name="actionTutorial_4_Weather_Model_Initialization">
  <property name="text">
```

```
   <string>Tutorial 4: Weather Model Initialization</string>
  </property>
 </action>
 <action name="actionDEM_Download">
  <property name="text">
   <string>DEM Download</string>
  </property>
 </action>
 <action name="actionfetch_dem_Instructions">
  <property name="text">
   <string>fetch_dem Instructions</string>
  </property>
 </action>
 <action name="actionCommand_Line_Interface">
  <property name="text">
   <string>Command Line Interface</string>
  </property>
 </action>
 <action name="actionAbout_WindNinja">
  <property name="text">
   <string>About WindNinja</string>
  </property>
 </action>
 <action name="actionCitation">
  <property name="text">
   <string>Citation</string>
  </property>
 </action>
 <action name="actionEmail_Us">
  <property name="text">
   <string>Email Us</string>
  </property>
 </action>
 <action name="actionSubmit_Bug_Report">
  <property name="text">
   <string>Submit Bug Report</string>
  </property>
 </action>
 <action name="actionNew_Project">
  <property name="text">
   <string>New Project</string>
  </property>
 </action>
 <action name="actionOpen_Project">
  <property name="text">
   <string>Open Project</string>
  </property>
 </action>
 <action name="actionExport_Solution">
  <property name="text">
   <string>Export Solution</string>
  </property>
 </action>
 <action name="actionClose_Project">
```

```
   <property name="text">
    <string>Close Project</string>
   </property>
  </action>
  <action name="actionExit_WindNinja">
   <property name="text">
    <string>Exit WindNinja</string>
   </property>
  </action>
 </widget>
 <resources/>
 <connections/>
</ui>
```