Gust-Front Web Application

A Prototype Web-Map Interface For The Neuro-Fuzzy Gust Front Detection Algorithm

CSCI 483 Interdisciplinary Project Capstone Spring 2026

Authors:
Robert McAbee
Eoghan Perkins
William Elder
Coleton Flesner

1 - Introduction

Gust fronts are sudden wind boundaries formed when thunderstorm downdrafts reach the ground and spread outward. These rapidly moving air masses create hazardous conditions when they cause erratic wildfire behavior. As a result, firefighters and field personnel safety are at high risk when these gust fronts occur.

Due to these risks, timely detection of these gust fronts is crucial. Existing tools to analyze weather data are often not easily accessible or tailored to gust front detection and firefighters in the field.The Neuro-Fuzzy Gust Front Detection Algorithm (NFGDA) was developed by researchers at the University of Oklahoma and collaborating institutions and is currently maintained for fire applications by the US Forest Service's Missoula Fire Sciences Laboratory ("Fire Lab"). The NFGDA detects and predicts incoming gust fronts using data from NEXRAD radar stations across the United States. However, the algorithm's usefulness in the field is limited by its command-line-only interface.

This project focuses on bridging the gap between research capabilities and field operations by developing a web-based application that makes the NFGDA algorithm accessible through an intuitive interface. By integrating real-time weather station data with an interactive map, the system enables firefighters and other field users to easily visualize and monitor gust front activity without requiring technical expertise. The goal is to transform a powerful but inaccessible research tool into a practical, user-friendly application for real-world deployment.

2 - Qualifications

The team members' resumes can be found on the following pages.

ROBERT MCABEE

robertmcabee.netlify.app
(406) 334-9106
robertthemcabee@gmail.com

SUMMARY

I'm a self-taught React developer turned computer science student. I'm passionate about clean code and elegant design.

PERSONAL PROJECTS

Gust Front Detection Web App (Senior Capstone Sp 2026)
A containerized web interface for the Neuro-Fuzzy Gust Front Detection Algorithm, built in collaboration with the USFS. Uses a Python backend with a React frontend.

Easyhabit (Personal Project 2022) - A full-stack cloud-based habit tracker app. Users can track their own data over time in a beautiful, responsive interface. Built from a Supabase backend with TypeScript, React and Tailwind CSS.

EXPERIENCE

Single Barrel — *Inventory Manager*

OCTOBER 2022 - CURRENT

Inventory Management: Led various inventory management tasks, including ordering inventory, and correcting product information.

Training and Mentoring: Provided training and mentoring to new hires to ensure they were equipped with the necessary procedures and tools to succeed.

Security Title Company — *Document Review & Database Maintenance*

JULY 2018 - JAN 2022

Created Tools - Programmed a JavaScript utility and created reference documents, which are currently used department-wide on a daily basis.

Research Skills - Maintained an accurate SQL database, researching out-of-date, incomplete or incorrect information.

Detail-Oriented - Reviewed legal documents such as deeds and mortgages to verify accuracy and insure legal requirements were met.

Professionalism - Directly worked with clients such as attorneys and lenders to meet industry-specific and local requirements.

Collaboration - Coordinated with a small team to effectively share tasks and meet tight deadlines.

EDUCATION

B.S., Computer Science

Comp. Engineering Minor

Montana State University

3.94 GPA

EXPECTED DEC 2026

B.A., Biblical Studies

Montana Bible College

AUG 2014 - MAY 2018

TECHNOLOGIES

C

Java

Git

React

JavaScript

Python

FPGAs (VDHL)

Eoghan Perkins

perkinseoghan@gmail.com • (503) 926 - 3136 • www.github.com/Eoghan-Perkins

TECHNICAL LITERACY

Programming Languages: Go, Python, Java, SQL, Dart
Databases & Data: PostgreSQL, PostGIS, MySQL, Redis, MongoDB, AWS S3/Aurora
Tools & Technologies: Git/GitHub, Docker, Apache Airflow, JupyterLab, NumPy, pandas, QGIS

EDUCATION

Bachelor of Science in Computer Science Graduating May 2026
Montana State University, Bozeman MT

- Relevant Coursework: Data Structures and Algorithms, Artificial Intelligence, Databases, Networks, Advanced Software Engineering, Discrete Mathematics, Computer Systems, Computer Science Theory

Bachelor of Science in Biological Sciences Completed May 2022
Montana State University, Bozeman MT

- Notable Courses: Intro to GIS, Advanced GIS and GPS Systems, Biometry, Argument-Based Research

PROJECTS

GlenCairn – A Social Oriented Scotch Whisky Review Application

- Designed and implemented a full-stack application using Go, MySQL, and React/TypeScript, following MVC architecture principles
- Built RESTful APIs supporting authentication, CRUD operations, and cloud deployment

Includify – An Accessibility Application catered to Organizations with Sprawling Campuses

- Designed a mapping application that centralizes campus accessibility data, improving information discoverability across large, distributed environments
- Authored requirements specifications and system design documentation, translating user needs into technical solutions

WORK EXPERIENCE

Land ID, Bozeman MT - *Software Engineering Intern* Jan. 2025 - Current

- Designed and deployed a cloud-native, distributed data pipeline using Apache Airflow with custom DAGs to orchestrate Dockerized, idempotent services processing nationwide datasets (100M+ records)
- Ran idempotent ETL jobs on AWS Fargate, enabling scalable execution with safe retries and consistent runtime environments
- Implemented comprehensive unit tests, structured logging, QA instrumentation, and data validation to ensure correctness, observability, and reliable cross-team consumption
- Optimized end-to-end ETL execution, reducing processing time by 70%+ while improving overall system reliability and maintainability

Montana State University Vertebrate Museum – *Team Lead & Intern* 2021 - 2022

- Directed digitization project and workflow that enabled creation of over 500,000 data points
- Optimized digital database for ongoing cross-collaboration of dozens of Ecology department employees and students

OTHER ACCOLADES

- Early acceptance into the **Accelerated M.S. in Computer Science Program** at Montana State University
- **Consecutive Semesters** on the College of Engineering **Dean's List** (Fall 2023 – Spring 2025)
- **1st Place Winner** - Montana State University **Big Idea Challenge 2024 (Inlcudify)**
- 2025 - Awarded the **Ressmeyer Scholarship for Computer Science Advancement**

WILLIAM ELDER

williamxelder@gmail.com | (808)859-4938

COMPUTER SCIENCE MAJOR WITH MINOR IN MATH; VETERAN

Former US Army Military Satellite Communications Operator and Maintainer with 4 years of experience. Sec+, Net+ and Linux+ Certified. Current Montana State University student.

Active Security Clearance

MONTANA STATE UNIVERSITY

- **Computer Science** **Expected Graduation December 2026**
  - B.S. in Computer Science, Minor in Mathematics — GPA: 3.6
  - Emphasis on probability algorithms and their implementations in C, Python or Java
  - Developed a web application displaying gust front data for the U.S. Forest Service as part of a collaborative team
  - GitHub username: Williamj48
- **Mathematics**
  - Completion of four semesters of calculus, physics II, linear algebra and discrete math
  - Focus on applied math with the use of MATLAB

MILITARY EXPERIENCE

U.S. Army Signal Corps **3/2020- 01/2024**
Army Satellite Communications Operator-Maintainer (MOS 25S)
Active Security Clearance: Secret

- **Professional Designations**
  - Certified to operate GATR inflatable satellite antenna systems
  - Security+, Linux+ and Network+ certified
- **Global Customer Focus**
  - Worked jointly with Armed Forces of allied nations, including overseas deployment
  - Provided tactical communications and technical support for over 400 soldiers
  - Enforced strict adherence to Department of Defense OPSEC guidelines
- **Equipment/Network Design**
  - Provided encrypted communications using TACLANEs
  - Connected to satellites using a satellite transportable terminal and other communications equipment
  - Responsible for all technical aspects of satellite equipment installation, configuration and maintenance
- **Systems Troubleshooting**
  - Diagnosed satellite and network equipment within physical, data link, network, and transport layers (1-4) of OSI Model
  - Helped end users solve problems related to connectivity and device configuration on mission critical devices
- **Operator**
  - Served as lead operator on my team
  - Troubleshot equipment to maintain services with less than 1% downtime

Coleton Flesner

ColetonFlesner@ProtonMail.com

https://www.linkedin.com/in/coleton-flesner-a26b1029a
(253)-941-2815

EDUCATION

Bachelor of Science, Computer Science (Interdisciplinary Option)
Minor in Mechatronics
Montana State University, Bozeman, MT
Expected Date of Graduation: December 2026

Specialized Skills

- Languages: Java, Python, C
- Tools: VSCode, Eclipse, IntelliJ

Relevant Courses

- CSCI 331 Web Development
- CSCI 232 Data Structures and Algorithms
- CSCI 305 Concepts/Programming languages
- ESOF 322 Software Engineering

Relevant Projects

- Python project to track moving objects
- Java project to guess animals by characteristics
- Website to post and interact with user made polls

AWARDS

- Dean's List – 2023, 2024

RELATED EXPERIENCE

Montana State University
CLS 460 **2019 – 2020**

- Teaching internship. Helped students adapt to college and overcome logistical challenges to ensure success, graded and led discussions on class material.

OTHER EXPERIENCE

Re-Compute: **2025**

- Communicating with customers about the extent of their computers needs
- Repaired software and hardware in personal computers and other devices

References

- Hunter Lloyd, Professor and Advisor
  - hunterl@montana.edu (406) 994-5959
- Dave Tirrell, Owner of Re-Compute
  - recompute@hotmail.com (406) 522-9049

3 - Background

3.1 Gust Fronts:
Gust fronts, or outflow boundaries, are high speed wind boundaries formed when thunderstorm downdrafts reach the ground and spread outward. Gust fronts can be identified visually by thin lines spreading out from storm systems on radar reflectivity maps.

These gust fronts can create hazardous conditions when wildfires are present in the area, dramatically changing fire behavior. These gust fronts can come suddenly and pose a massive risk to field workers like firefighters. This can have tragic results, as in the case of the 2013 Yarnell Hill Fire which killed 19 firefighters.

3.2 Existing Tools:
Several tools are used to analyze weather conditions and atmospheric events. One of the most widely used systems is provided by the National Weather Service, which delivers forecasts and alerts for hazardous weather events. This system is highly reliable, but is designed for large-scale forecasting and does not provide gust front detections.

Another resource is data provided by the National Centers for Environmental Information under NOAA. This resource offers both realtime and atmospheric data available for identifying weather patterns. However, the interface and dataset is designed for researchers and meteorologists.

Research-based approaches such as the Automatic Detection of Gust Fronts procedure (1986) and later algorithms like the MIT Lincoln Laboratory's MIGFA demonstrated that gust fronts can be detected algorithmically from radar data. However, these algorithms exist primarily as research artifacts as they are not used outside of research environments.

3.3 NFGDA:
The Neuro-Fuzzy Gust Front Detection Algorithm (NFGDA) is a radar-based gust front detection algorithm developed at the University of Oklahoma. It was designed to take advantage of dual-polarization upgrade to the national network of WSR-88D radar systems, which provide richer atmospheric measurements than the single-polarization Doppler data used by earlier gust front detectors.

The algorithm extracts a set of signatures from polarimetric WSR-88D data to characterize a gust front: medium reflectivity, apparent thin line feature in reflectivity, the motion of reflectivity, high differential reflectivity, low copolar cross-correlation coefficient, apparent convergence, and large standard deviation of differential phase. These signatures are inherently fuzzy. Rarely will an actual gust front match all of these perfectly. The NFGDA combines them using a fuzzy logic system, evaluating the presence and strength of each and aggregates them into an overall detection confidence. The system parameters are then tuned through neural network training, which is where the “neuro-fuzzy” terminology comes from.

For this project, the NFGDA serves as the detection backbone: our system feeds real-time WSR-88D data and surfaces its output through a map-based interface oriented toward fire field operations. The algorithm

itself is treated as an external dependency. Our project is the integration, accessibility, and operational framing around it.

3.4 Differentiation:
The technologies previously listed are either general purpose weather products that are accessible to field users, but lack automated gust front detections, or gust front detection algorithms that perform the technical task well, but are packaged as research software. Neither of these categories caters to the specific needs of wildland fire field users.

The NFGDA and its predecessors require command-line proficiency, radar data pipelines, and meteorological background to operate. This project bridges the gap and wraps the Missoula Fire Sciences Laboratory's NFGDA implementation in a web-based application that overlays the algorithms outputs atop an interactive map, with NEXRAD station selection, real-time and historical playback, and confidence colored detection and prediction of gust fronts. This interface is designed to be operable from a phone or laptop without command-line tools and is oriented specifically towards wildland fire field personnel. To our knowledge, no existing system combines the gust front detection and prediction like the NFGDA with a field-deployable interface.

4 - Work Schedule

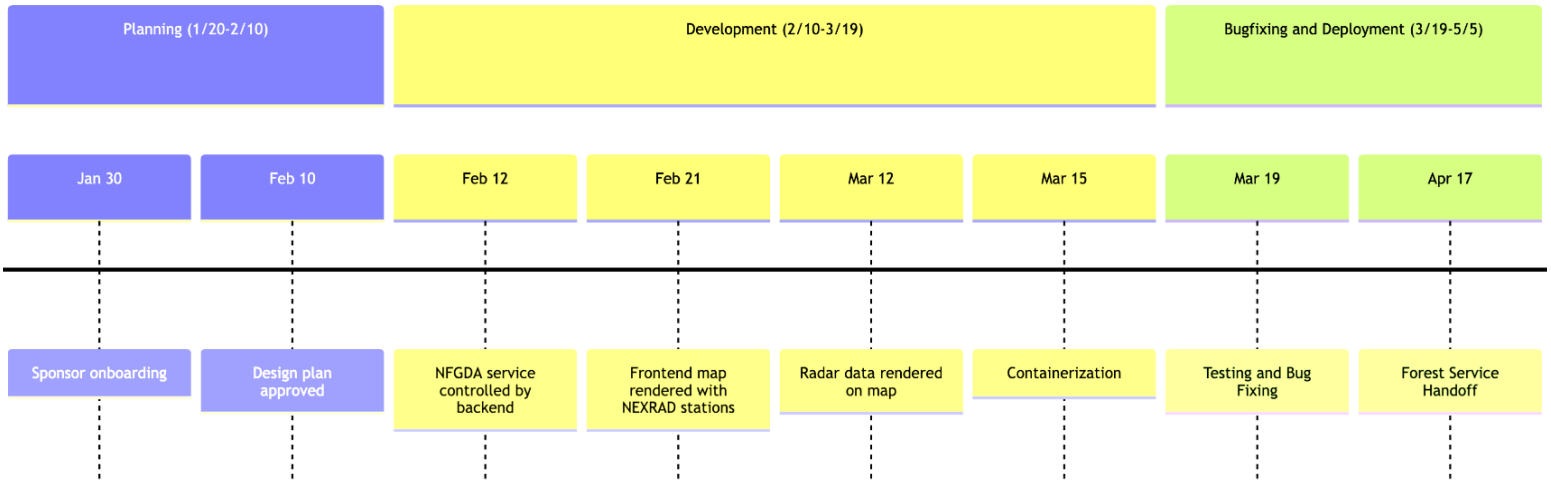
4.1 Team Members and Responsibilities

Member	Role	Primary Responsibilities
Robert McAbee	Front End	Map integration, GeoTiff rendering, UI Design
Coleton Flesner	Front End	NEXRAD Radar Station Display, Frontend Containerization
William Elder	Back End	Flask API, Backend Containerization
Eoghan Perkins	Back End	Redis Job Queue, Backend API Endpoints, NFGDA integration

4.2 Development Lifecycle

The team adopted an agile methodology, working in short cycles driven by weekly meetings with the USFS to determine what was needed for the product. Because the project depended on integrating a third party research algorithm (NFGDA) with no documentation, the capabilities were discovered during development, and thus an agile methodology suited the development process.

4.3 Project Timeline



#	Milestone	Date	Description
1	Sponsor onboarding	January 30	Initial meeting with the USFS Fire Lab. Defined scope of work.
2	Design plan approved	February 10	Written plan and structural/functional UML diagrams approved by USFS Missoula Fire Lab.
3	NFGDA service controlled by backend	February 12	NFGDA accepts job requests from the backend and writes detection output to a shared volume.
4	Frontend map rendered with NEXRAD stations	February 21	React/Leaflet map displays all continental US NEXRAD stations as interactive markers, delivered via backend API
5	Radar data rendered on map	March 12	NFGDA detection output processed into GeoTIFFs, reprojected and rendered on basemap
6	Containerization	March 15	All four services (frontend, backend, NFGDA service, redis instance) cross-communicate and initialize together
8	Testing and Bug Fixing	March 19	Core functionality complete, begin stress testing and optimization of UI and backend functionality
9	Forest Service Handoff	April 17	Begin work with forest service team on deployment to their server infrastructure and coordinate product handoff

5 - Project Technical Information

5.1 Functional Requirements

FR-1 The system shall display an interactive map showing all continental U.S. NEXRAD radar stations as clickable markers.
FR-2 The user shall be able to select a radar station from the map and from a dropdown menu to initiate analysis.
FR-3 The system shall execute the Neuro-Fuzzy Gust Front Detection Algorithm (NFGDA) on the data from the user-selected station.
FR-4 The system shall display an animation of the radar data.
FR-5 The map shall overlay NFGDA outputs including detected gust fronts and predicted gust front locations
FR-6 The user shall be able to select a timezone.
FR-7 The system shall interface with external radar sources, including NEXRAD metadata, GIS layers, and data stored on AWS
FR-8 The application shall be compatible with the Weather and Climate Toolkit (WCT) for debugging and data validation purposes

5.2 Non-Functional Requirements

NFR-1 All code shall be hosted in a GitHub repository belonging to USFS Firelab
NFR-2 A typical NFGDA job covering a one-hour radar window shall complete processed GeoTIFF assets to the user within 5 minutes
NFR-3 The server-side logic must be developed using Python (Flask)
NFR-4 The user interface must be built using React
NFR-5 Geospatial visualizations must be handled via the Leaflet library
NFR-6 The application must be containerized to ensure consistent development across environments

5.3 Performance Requirements

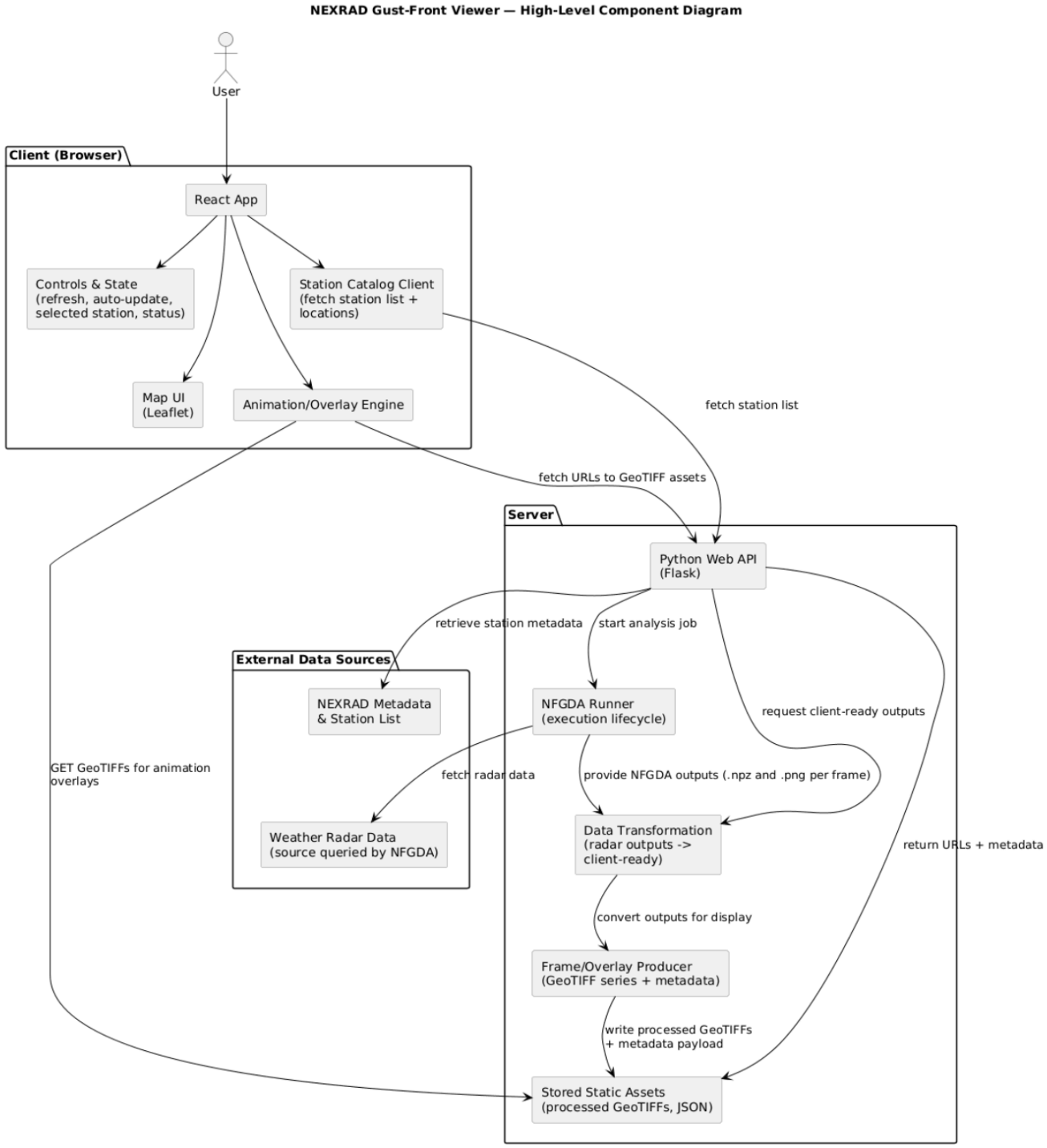
PR-1 The system shall process a one-hour radar window for a selected NEXRAD station and make generated GeoTIFF outputs available to the user within 5 minutes under normal operating conditions.
PR-2 The system shall load the interactive map and NEXRAD station markers within 2 seconds on a standard broadband connection.
PR-3 The system shall display radar animation frames with no more than 1 second of delay between frame transitions during playback.
PR-6 The system shall support at least 2 concurrent users/jobs without exceeding the defined processing time limit.

5.4 Interface Requirements

IR-1 The user interface shall provide an interactive Leaflet map displaying all continental U.S. NEXRAD radar stations as selectable markers.
IR-2 The user interface shall provide a dropdown menu containing all available NEXRAD radar stations.

IR-3 The user interface shall allow the user to select a radar station from either the map marker or the dropdown menu.
IR-4 The user interface shall provide controls for selecting the radar analysis time window.
IR-5 The user interface shall provide controls for selecting the display timezone.
IR-6 The user interface shall provide controls for starting an NFGDA analysis job.
IR-7 The user interface shall display processing status, including queued, running, completed, and failed states.
IR-8 The user interface shall display radar animation controls, including play, pause, restart, and frame navigation.
IR-9 The user interface shall display NFGDA output layers, including detected gust fronts and predicted gust front locations.
IR-10 The backend API shall expose an endpoint for retrieving available NEXRAD radar station metadata.
IR-11 The backend API shall expose an endpoint for submitting an NFGDA processing request.
IR-12 The backend API shall expose an endpoint for checking processing job status.
IR-13 The backend API shall expose an endpoint for retrieving generated radar animation assets and NFGDA output layers.
IR-14 The system shall display error messages when radar data cannot be retrieved, processing fails, or outputs cannot be displayed.

5.5 Architectural Design Documents



pp

5.6 Development Standards and Tools Used

Our team followed a number of holistic and consistent development standards to ensure the code's quality and reliability during the project lifecycle. We utilized git for version control, and sequestered new work into feature branches that were quality checked automatically via integration testing and basic linting workflows that were triggered on pull requests via GitHub actions. Each pull request was required to pass these checks before it could be merged into our main branch. This testing acted as a validation process that confirmed the core functionality of our product had not been compromised in between consecutive code pushes, enabled by a baseline test suite early in the development process. Python code quality was enforced using the Ruff linting tool, which ensured consistent formatting and helped identify common errors early in the development process.

Validation was performed both manually and automatically, with the integration testing pipeline confirming baseline functionality of the entire system from a high level, and the interactive map interface allowing developers to directly observe and evaluate the results of each request. This provided immediate visual confirmation that the GeoTiff frames produced as output of the NFGDA algorithm and subsequent data processing contained the desired weather reflectivity and gust front detection data.

Our team emphasized clarity and traceability from the very beginning of the development lifecycle. including descriptive comments consistently and uniformly throughout the codebase to explain code logic, design choices, and implementation details, especially within the more complex components of the product. Pull request descriptions came with the expectation that they were clearly detailed as to their purpose, with clear logs of changes, design decisions, and links back to notable issues. These practices were ultimately extremely beneficial to the development lifecycle, as it reduced the cognitive load and temporal delay associated with having to discern the purpose of a groupmates' pull request without sufficient documentation. We got to witness firsthand what a structured and disciplined development environment can do when collaborating on a complex software system.

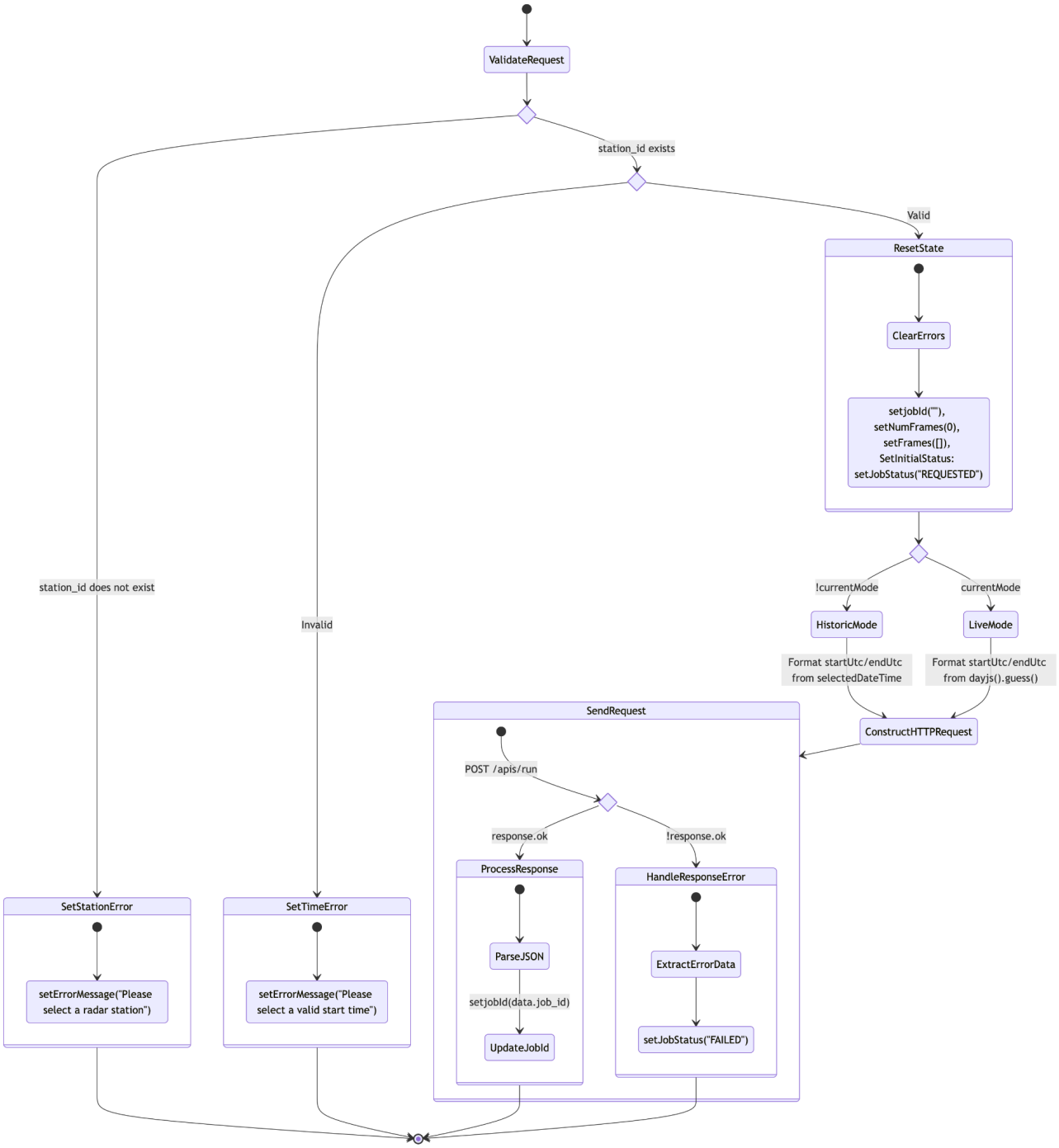
From the beginning of the project, we set out to leverage a modern, heavily decoupled system architecture separated into two main parts: a Python-based backend and a React/Typescript powered user interface. On the backend, the core web API is powered by Python's Flask library, providing a reliable out-of-the-box framework for routing requests sent from the user interface. We leveraged an in-memory data cache (Redis) to manage, queue, and coordinate these tasks asynchronously. This allows the main web server to remain serviceable for other frontend requests while the sequestered and dedicated background service (NFGDA Service) is responsible for running the resource-intensive NFGDA algorithm developed by students at the University of Oklahoma. The user-facing application was constructed using the React framework. The core component of the user interface is the Leaflet web-map module, which empowers the application's ability to present the GeoTiff frame products of individual job requests. The visual design and layout of the application are constructed using Material UI, which provides a clean, modern interface for the web-map application controls.

It should be noted that there is some functionality deliberately left un-discussed in this document and out of the attached code appendix and technical detail in this document. The NFGDA algorithm related code itself provided by the University of Oklahoma, and a GeoTiff processing script provided to us by the US Forest Service. Both of these components will factor into the following technical design diagrams.

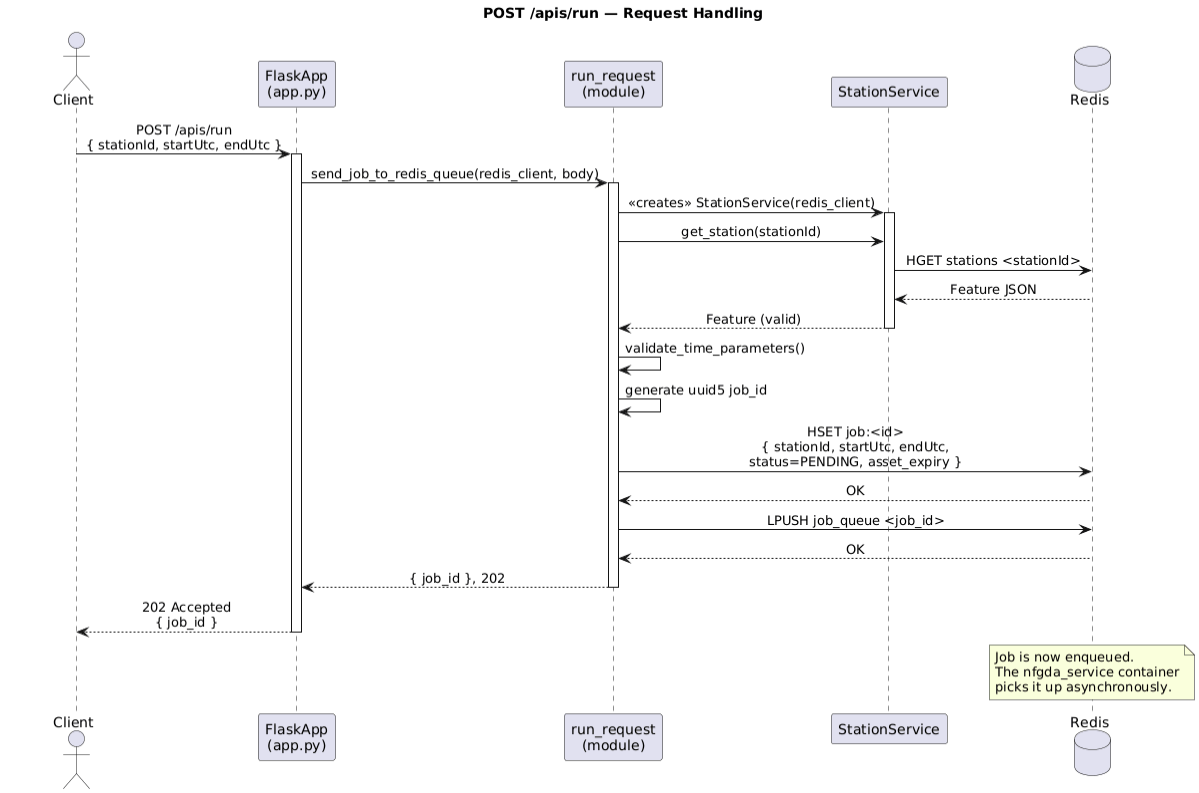
6 - Methodology

6.1 UML Diagrams

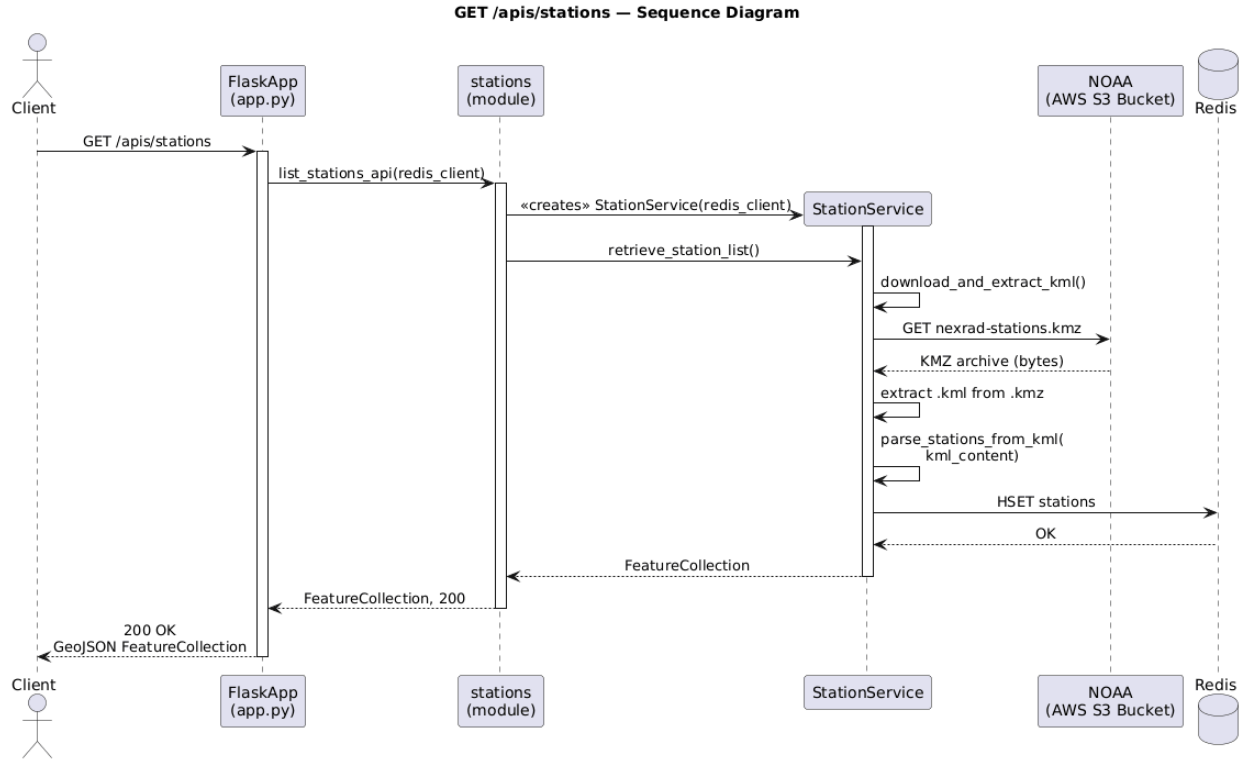
Sending HTTP Job Request to Backend



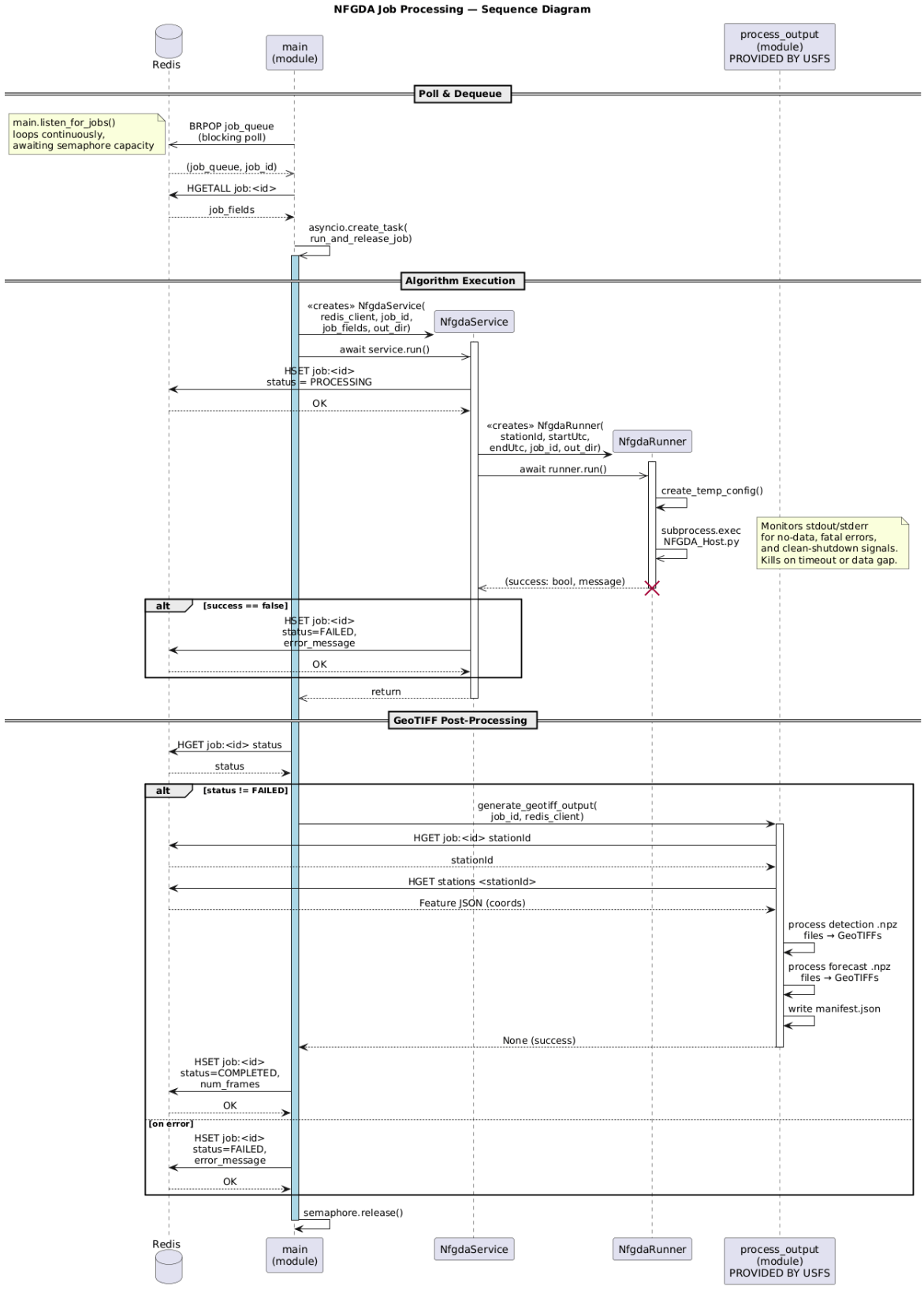
Processing job in backend container (*run_request()* route)



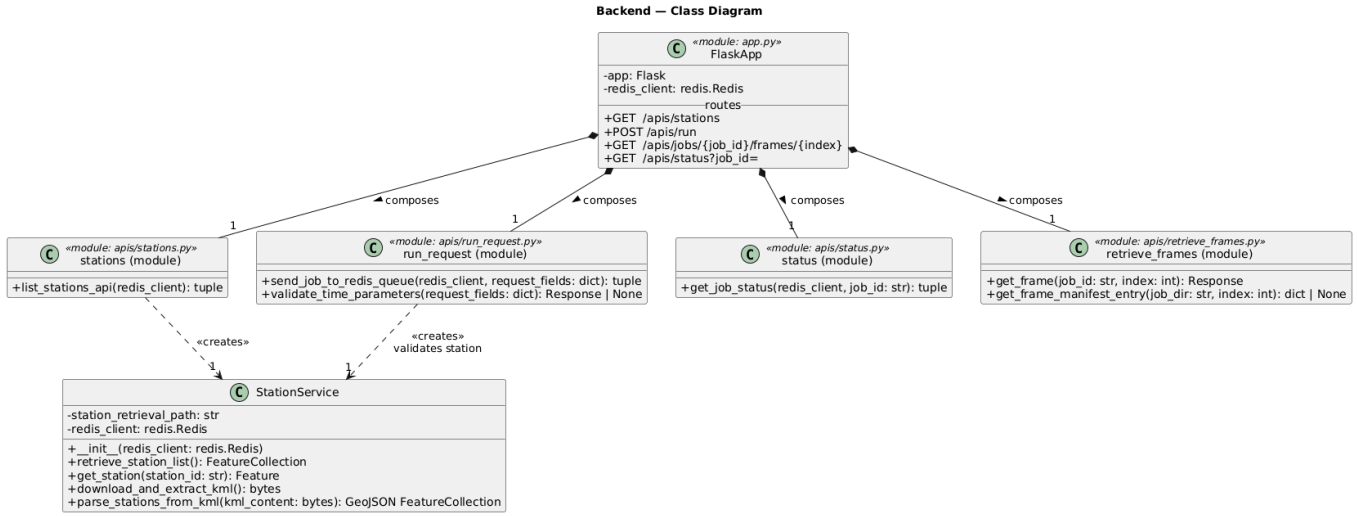
Request for list of stations



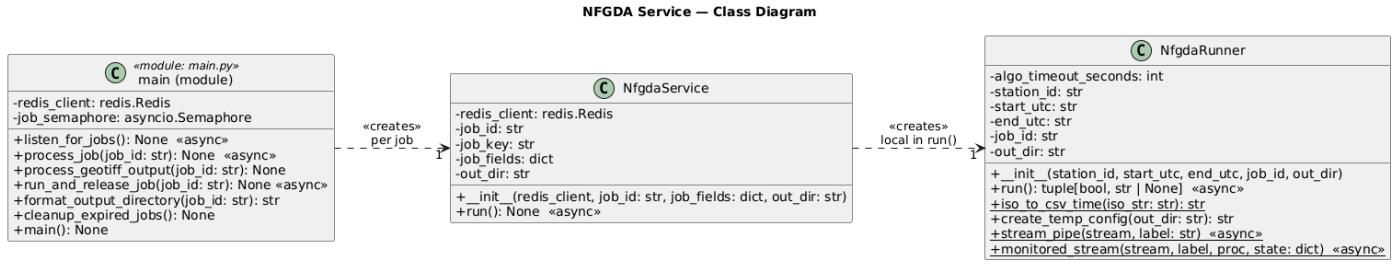
NFGDA Job Processing



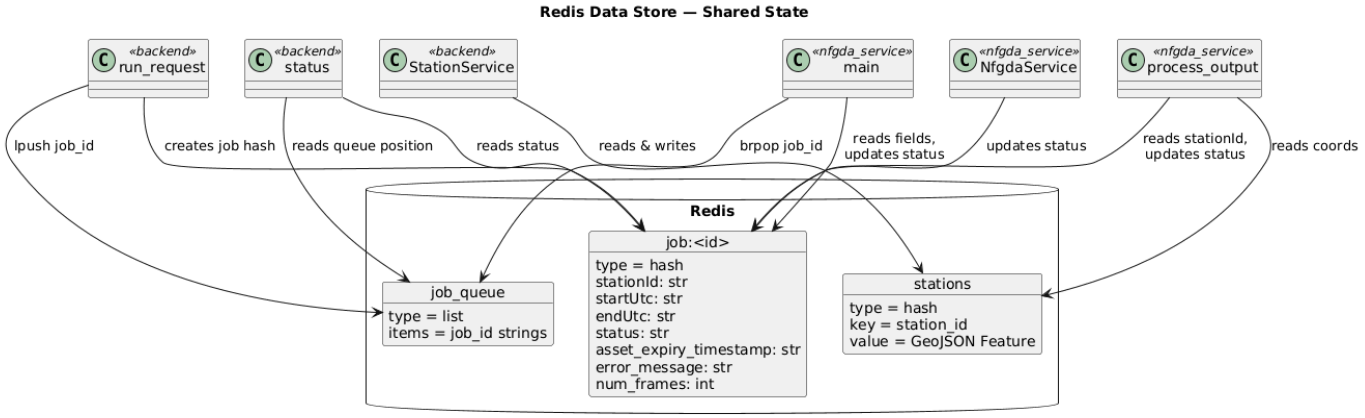
Backend Class/Module Diagram



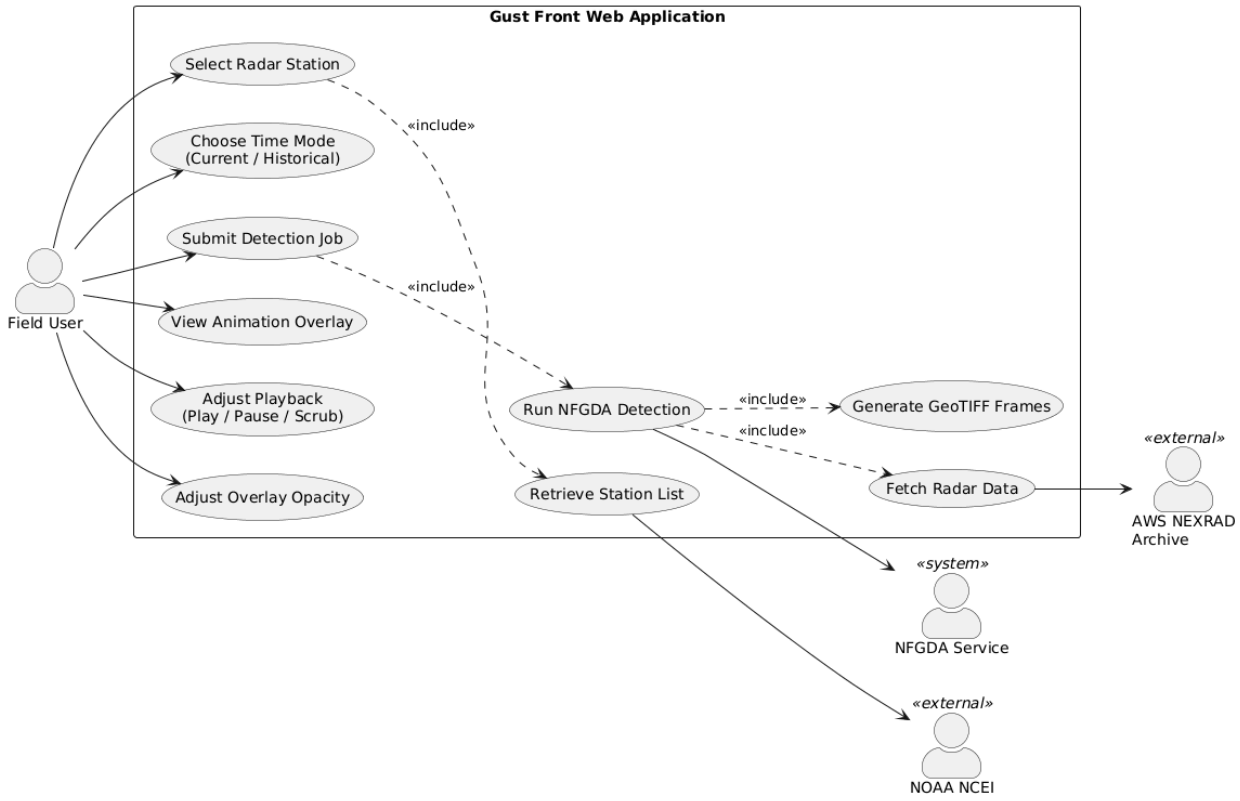
NFGDA Service Class/Module Diagram



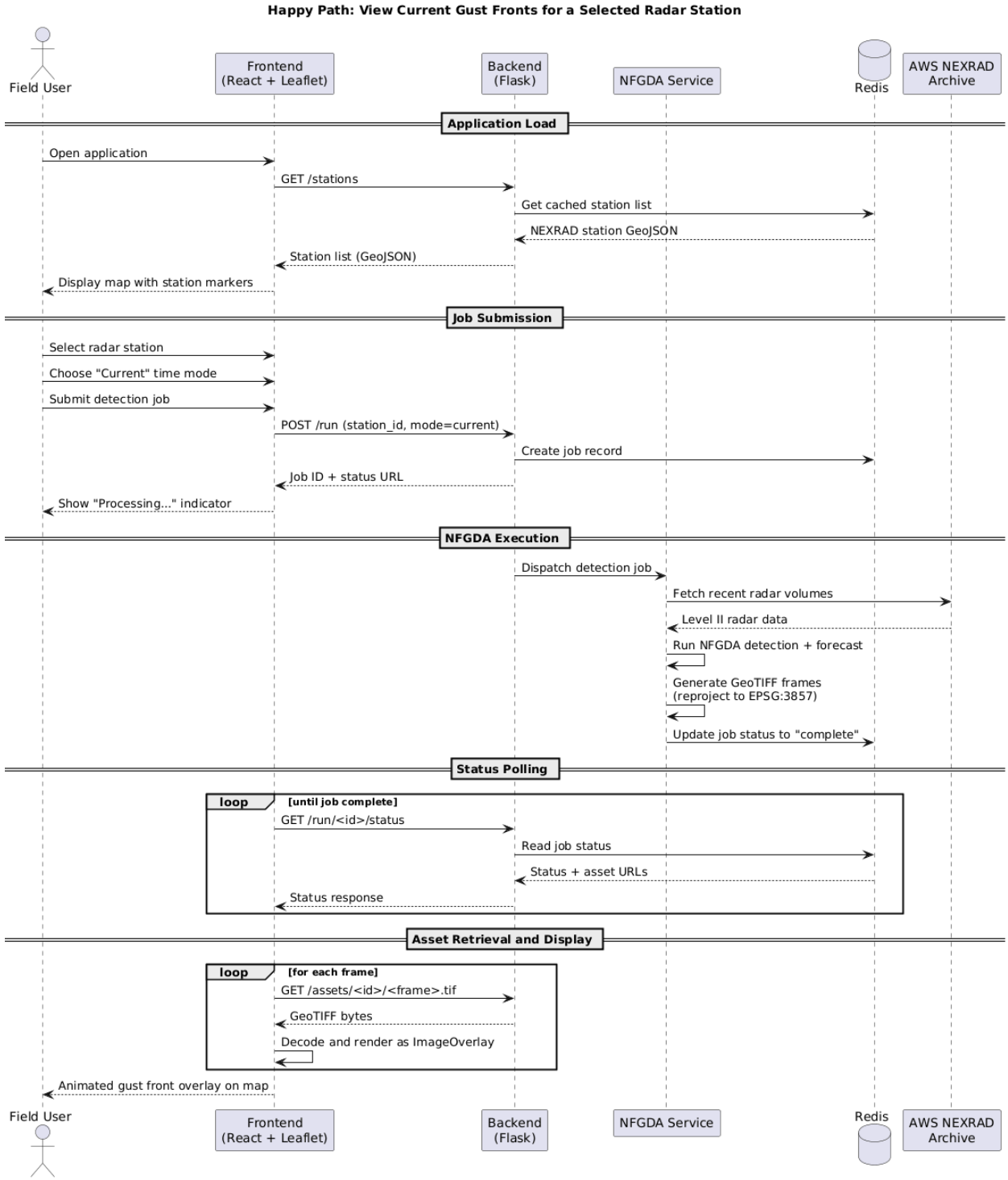
Redis Instance (mock) Class Diagram



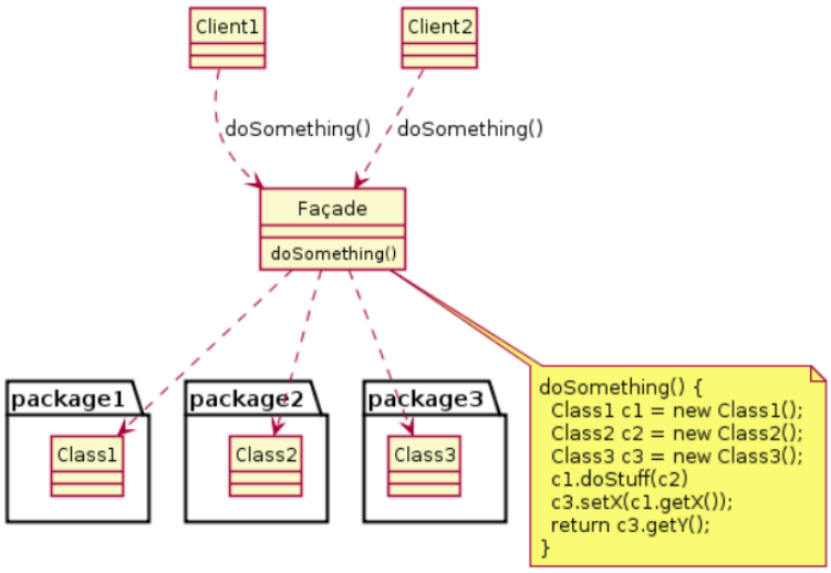
Main use case Diagram



High Level Sequence Diagram



6.2 Design Patterns Used - The Facade Pattern



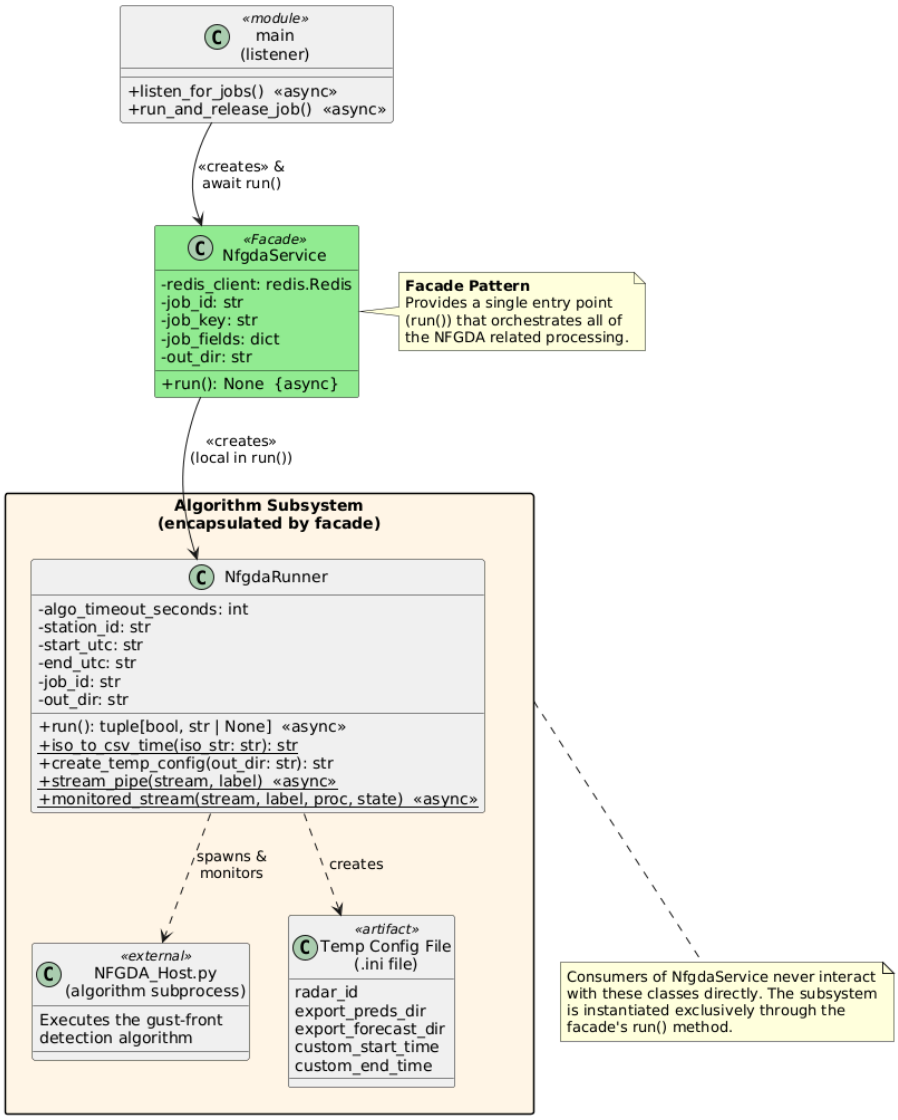
The Facade Pattern is a structural design pattern that provides a simplified, unified interface to a complex subsystem, shielding the caller from its internal workings. Rather than exposing the details of how individual components operate, a facade presents a single, clean entry point that delegates responsibility to the underlying classes. (“GeeksForGeeks”, 2024).

In our application, this pattern is implemented by way of the NfgdaService class. Consumers of this class are not asked to understand or orchestrate the behavior of the classes defined beneath it. They are designed to adhere to the facade pattern architecture and be instantiated and managed exclusively through NfgdaService.

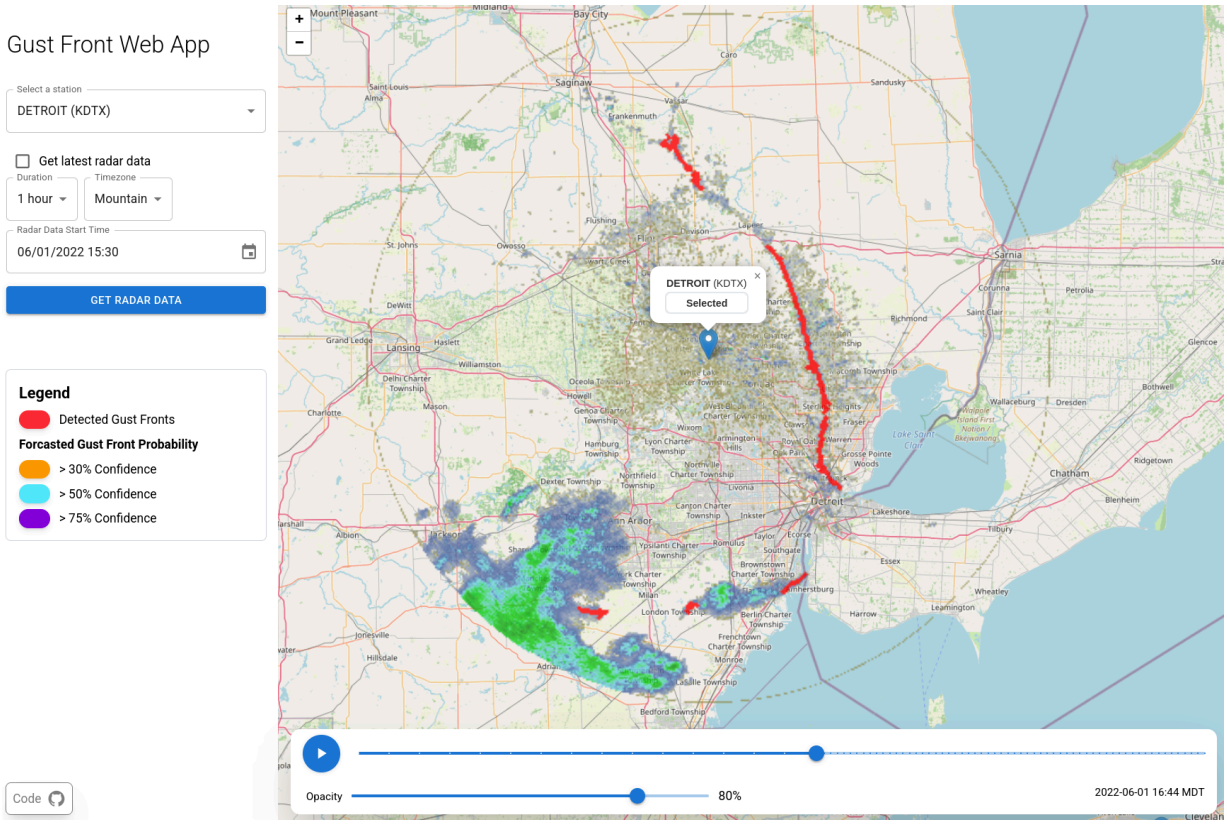
This is a classic example of the facade pattern architecture, because the NfgdaService class completely abstracts away the functionality of the NFGDA algorithm and the classes and methods required to make it run. This separation allows a single execution pathway for the encapsulated classes to run, while keeping the interface clean and intentional by hiding the complexity of the underlying subsystem.

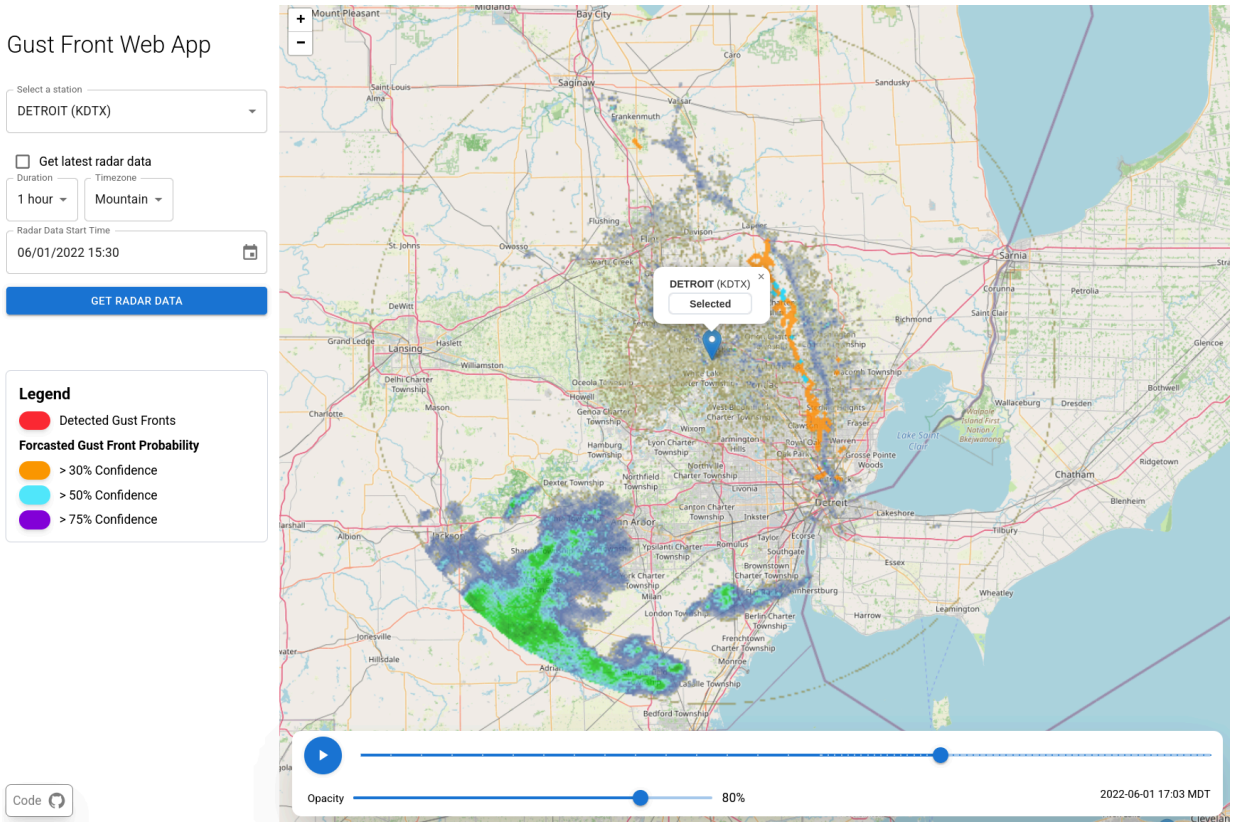
In our case, the single execution path is called via the primary method provided by the NfgdaService class “run()”. The routing middleware between the API endpoint /run_request that acts as an upstream method call and NfgdaService's run() method is more complicated than one method simply calling another: it relies on job metadata passed between a Redis cache and one of the classes within the encapsulated subsystem (NfgdaRunner). However, the lone execution pathway between the API endpoint and the run() method qualifies this scheme to be described via the facade pattern. Our implementation of this pattern can be visualized via the following figure:

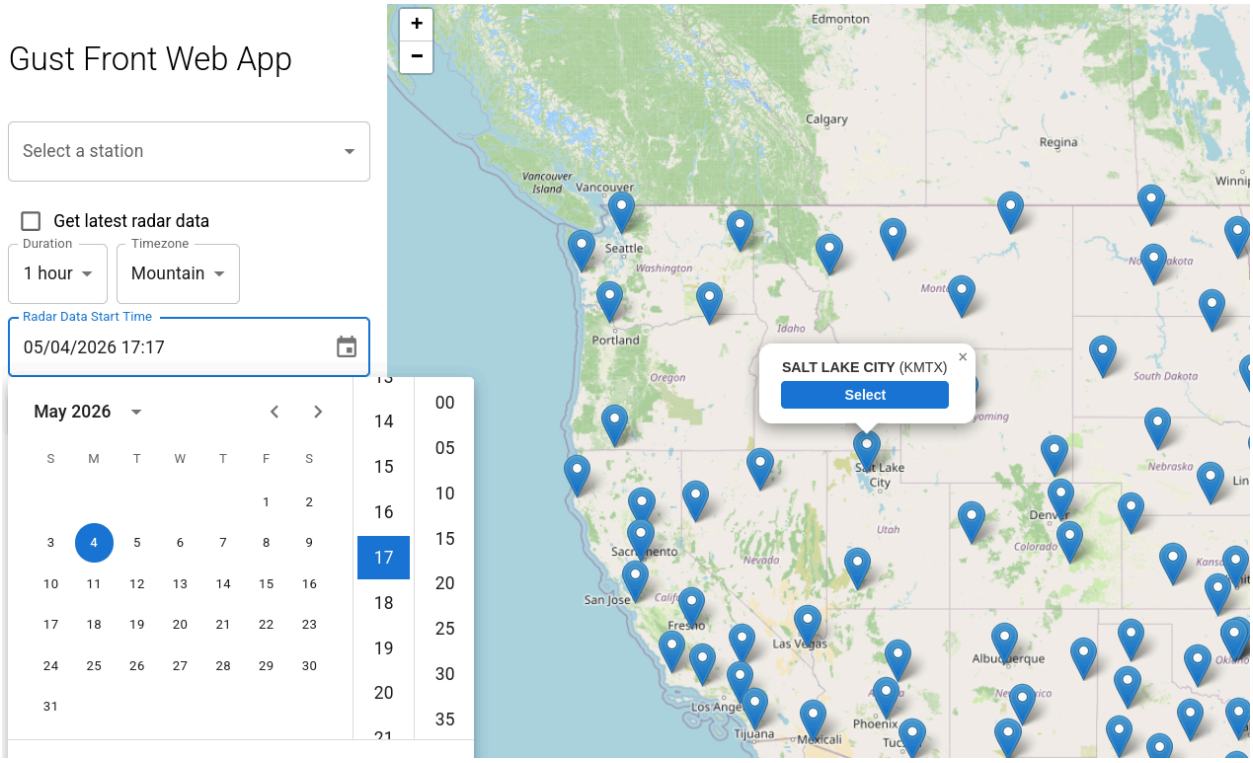
Facade Design Pattern — NfgdaService



6.3 Example Usage






Gust Front Web App
Select a station
Get latest radar data
Duration
1 hour
Timezone
Mountain
Radar Data Start Time
05/04/2026 17:17
May 2026
S M T W T F S
1 2
3 4 5 6 7 8 9
10 11 12 13 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29 30
31
14 15 16 17 18 19 20
00 05 10 15 20 25 30 35
Edmonton
Calgary
Regina
Vancouver Island
Vancouver
Seattle
Washington
Portland
Oregon
Idaho
SALT LAKE CITY (KMTX)
Select
Salt Lake City
Utah
Nevada
Colorado
Nebraska
South Dakota
San Jose
Las Vegas
Albuquerque
Phoenix
Tijuana

References

"Hwang, Yunsung, et al. "Neuro-Fuzzy Gust Front Detection Algorithm with S-Band Polarimetric Radar | IEEE Journals & Magazine | IEEE Xplore." *Neuro-Fuzzy Gust Front Detection Algorithm With S-Band Polarimetric Radar*, 12 Dec. 2016, ieeexplore.ieee.org/document/7781638/.

Uyeda, Hiroshi, and Dusan S. Zrnić. "Automatic Detection of Gust Fronts in: Journal of Atmospheric and Oceanic Technology Volume 3 Issue 1 (1986)." *Automatic Detection of Gust Fronts*, American Meteorological Society, 1 Mar. 1986, journals.ametsoc.org/view/journals/atot/3/1/1520-0426_1986_003_0036_adogf_2_0_co_2.xml.

"What Is a Gust Front?" *National Environmental Satellite, Data, and Information Service*, NOAA, www.nesdis.noaa.gov/about/k-12-education/severe-weather/what-gust-front. Accessed 26 Apr. 2026.

"GeeksforGeeks. (2024). *Facade Design Pattern – Introduction*." www.geeksforgeeks.org/system-design/facade-design-pattern-introduction/. Accessed 4 May 2026.