Montana State University Computer Science Department

Senior Team Portfolio

Section 1: Program.

Project Link: https://github.com/423s23/G4-DrRx

Program specifications

The program specifications, as provided to us by the client, are as follows:

- This software shall read data containing patient results on various mental health screenings (ASRS, GAD-7, etc.) from some (unknown) 3rd party
- A user can query over this data to search for a specific patient
- A queried patient's information will be displayed, listing some of their personal information as well as their scores, and whether they are at risk for certain mental health conditions
- The patient's scores are examined by an "algorithm" and the recommended treatment (information which was not provided to the team) is displayed along with the patient's data

Section 2: Teamwork.

Team Member 1:

- Implemented CSV reading/formatting
- Implemented patient query by last name
- Implemented Builder Design Pattern
- ASRS screening result
- Implemented 'User Not Found' feedback

- All Developer Documentation

Team Member 2:

- Executable Creation Research (How to make .jar files to run from Desktop)
- Read and Recommend Format to Implement Testing
- GAD-7/PHQ-9/Columbia Testing screening result
- Charts (Backlogs/Burndown) and their upkeep
- GUI Remodel (Creating several panels for easier formatting)
- All User Documentation / All Documentation Formatting

Team Member 3:

- Initial Github Setup
- Test Data (Creating test patients with varying inputs for Screenings)
- Data Security (Making sure that input is read as text rather than code)
- Columbia Testing
- Formatting (Adding quality of life, and aesthetic GUI changes)

Team Member 4:

- Initial code file structure work
- Set up the ISI test and its recommendation
- GUI work setting up search bar and button
- GUI work full screen functionality
- GUI work displaying results of patient search

Backlog items and time estimate info

Section 3: Design pattern.

The design pattern used in this project was the Builder Design Pattern, which can be found within the buildLabels() method (located at line 230) within Questionnaire_Application.java.

This Design Pattern was used to streamline the process for creating the labels used to display patient data. Because eleven labels were required, initializing them individually would have caused there to be a large block of repetitive code wherein the same kind of label is created and modified with the proper format over and over. By using the Builder Pattern, this same process was simplified and performs the same task with much less code. Additionally, by using this pattern, we could very easily change the number of labels created, if needed, without needing to write any extra code.

Section 4: Technical writing.

Developer Documentation:

Obtaining Source Code

Source code for the latest version of this software can be obtained by downloading the Project folder in its entirety. All source code and .exe files can be found under the Project/JavaCode directory. Documentation, artifacts and version history can be found under the Tracking directory. No additional build instructions are currently associated with this software.

Updating Test Data

The test data .CSV file can be found under Project > testData > sampleinput.csv. Each line in the file represents an individual patient, and data is formatted the following way:

FirstName,LastName,DOB,PHQ-9,#,#,#,#,#,#,#,#,#,GAD-7,#,#,#,#,#,#,#,ISI,#,#,#,#,#,#,#,ASRS,#,#,#,#,#,#,CSS,#,#,#,#,#,#,#

With each # pertaining to a patient's numeric answer for a screening question. I.e.: PHQ-9,2,3,1,2,3,1,0,1,1 indicates that a patient answered 2 for question 1 of the PHQ-9 screening, 3 for question 2, 1 for question 3 and so on. DOB should be formatted as month_day_year (i.e. 1_1_2000.)

Unit Testing

A test suite is available for this software and can be found under the Test directory. Testing may be performed by running the Questionnaire_ApplicationTest.java and Check_DataTest.java files.

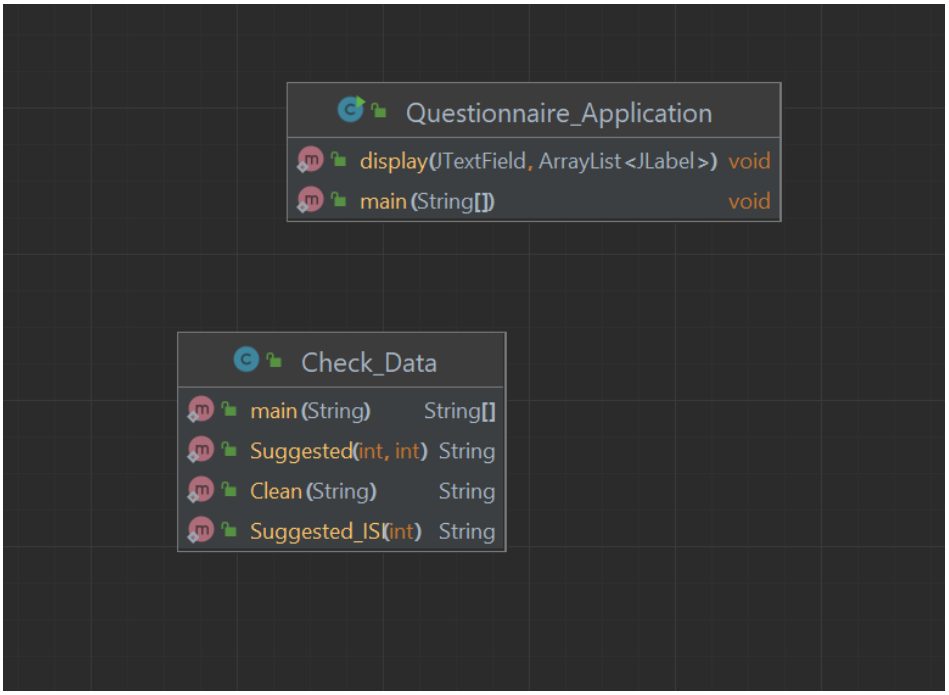
Bug/Issue Tracking

All bugs/issues are tracked and archived under the Issues tab on the Github repository. Known issues are considered prioritized over new features.

Versions

When a new version of the software is created, documentation and artifacts will be updated and the executable file will be rebuilt to run the most recent version of the software.

Section 5: UML.



Section 6: Design trade-offs.

One trade-off decision the team made was the decision to have the project expect only a basic .csv file format as user input. Having been advised initially that the input data would come from some unnamed 3rd party source, the team had to make certain assumptions as to what that data could look like due to the fact that the team had little to no communication with the project client in order for the project to somewhat resemble the desired product.

Because of this decision, the team also opted to forego a database, exchanging maximum accessibility between computers for easy functionality in order to read user data in a way that is guaranteed to function according to the project specifications provided to us.

Section 7: Software development life cycle model.

To develop this project, the team used an agile development model, and followed a two week sprint pattern wherein we would meet 2-3 times a week. This methodology proved very effective, as during weekly scrum meetings the team had a valuable chance to discuss, brainstorm and plan out the project, and spend the remaining weekly meetups working towards implementing ideas that were polished and sufficiently discussed.

However, there were also challenges associated with this method, most notably the difficulties the team faced when quantifying how long certain tasks would take to complete, which occasionally caused the team to encounter unexpected problems and fall short on feature deadlines.