Software Requirements

And Design Specification

for

Railway Reservation

Version 2.0 approved

Prepared by: Kenya Holland, Lance Blum, Dylan Morris, Priya Pilla, Evan Jones

Group 4

12/10/2020

Table of Contents

1. Introduction

1.1 Description

The Railway Reservation System (the product) will likely be a replacement for existing systems, as an upgrade to existing railway ticketing systems that are fairly dated, especially in parts of the Northeastern United States. This product contains a graphical interface and database component for a service in which users can look up train routing information, including: price, routing information, and other services. The working components include a GUI for booking a train ticket and a database in order to write, read, update, and delete both user data and train data. This would give both customers and administrators a new, easier to use system than ones currently available.

1.2 Team Contributions

Kenya Holland: Drew up initial and final versions of the train map and routes. Made the main menu page of GUI, styled the ticket booking page of GUI, and made adjustments to the ticket booking page as well as confirmation page. Assisted in establishing SQL database. Helped hook up the database to GUI. Calculated prices for tickets and displayed on the confirmation page. Refactored and troubleshot major changes in project code to follow OOP principles in GUI. Improved administration access processes. Completed most major parts of this document including diagrams and explanations. Tested functions that update the database. Finalized software requirements document.

Lance Blum: Assisted in base creation of train route structures and map. Established base file I/O, train information and classes. Assisted in establishing SQL database and troubleshooting. Created GUI for admin and backOffice pages, and utilized existing database queries for displaying trains in the database.

Dylan Morris: Assisted in brainstorming route map ideas and initial concepts for system structure. Established base Maven project, and established SQL database for storage of train information. Worked with teammates to style and set up GUI, and hooked the database into GUI for proper access to available trains and options. Helped troubleshoot File I/O and manually merged github conflicts when needed. Assisted in fixing refactoring issues and incompatibilities, as well as styling new GUI components, and improving existing features. Completed most major parts of this document, including diagrams and explanations. Did the final presentation. Added JUnit dependency and some tests for the program.

Priya Pilla: Created GUI for both the ticketing and confirmation pages, and worked with main JavaFX components. Also worked with existing train conventions to ensure no conflicts between GUI and backend arose. Major refactoring of the project to break up GUI classes to follow OOP principles. Assisted in troubleshooting the refactoring process. Did the final presentation.

Evan Jones: Helped to create the original train route and decide how trains should behave and operate. Assisted with file IO and csv files, along with other program troubleshooting. Worked on

the admin page to include login protection, GUI setup for backroom, and utilizing existing database queries. Sprint 2 presentation. Added junit tests. Did the final presentation.

1.3 Document Conventions

There are currently no standards or typographical conventions for this document, however there will possibly be new standards added during the next sprint.

2. Overview of Project Characteristics

2.1 User Classes and Characteristics

There are two ways a user can go into this program, the customer route and an administration route. The program stores train information and the only users that will have direct access to this information is the administration. Administration includes: security, customer service and ticketing staff, locomotive handler/motorman, engineers, etc. Administration is expected to have high levels of experience in ticketing systems and administration. Administration has the highest privilege and security levels, for they can change many aspects of train information. Customer user classes include anyone that needs to book a ticket for a train. Customers have low privilege levels when it comes to accessing and managing train information. They will only have access to their own information, for the ticket that they are booking. This includes top level information such as train routes, meals and types of seats offered, prices, and days that they will travel on said train.

2.2 Operating Environment

This program will run on Java version 11.0.8. It was created using Maven for dependency management, for the use of JavaFX in GUI creation. Derby framework is used for the implementation of our SQL database for this program. Junit-Jupiter API will be used as a unit testing framework.

2.3 Assumptions and Dependencies

Assumptions:

- Users will have access to some form of electronic device such as a phone or computer.
- Users will have access to an internet connection.
- Customers will be able to properly input the information for their trip.
- Stations will have kiosks for ticket sales and reservation information.
- Employees will be well trained on our system.
- Users will have a bank account and/or debit/credit card that they can pay online with.

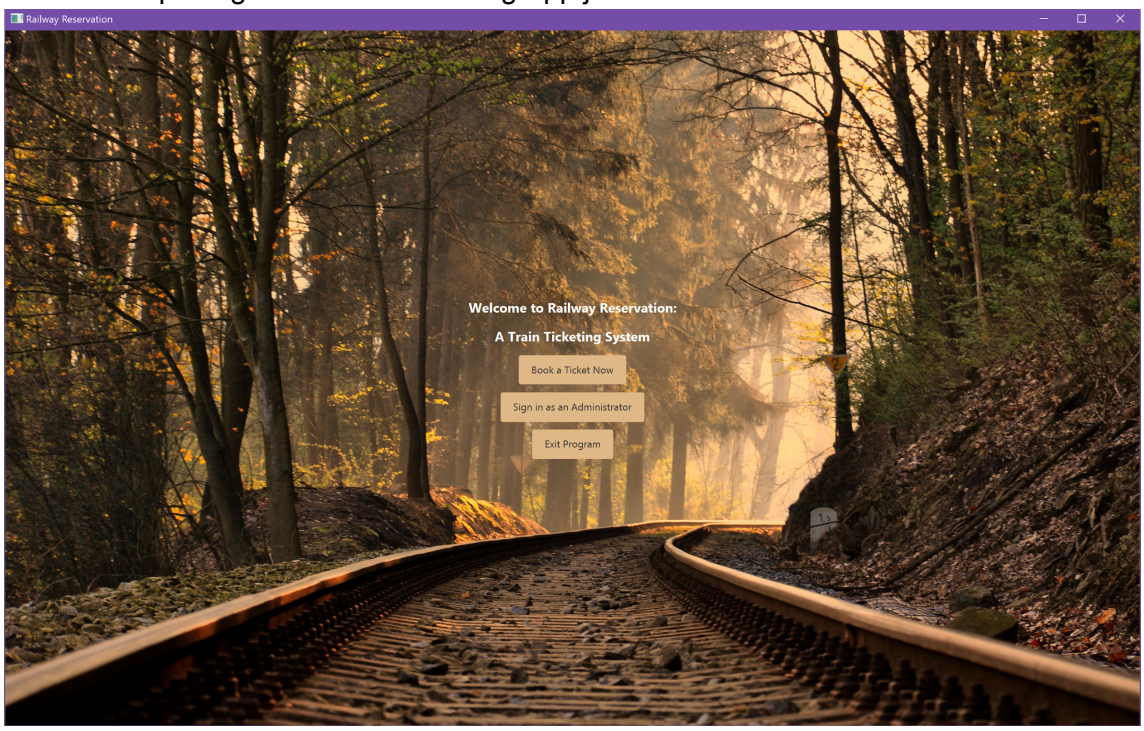
Dependencies:

- The way a user can pay will depend on an external interface provided by a company such as PayPal, that will allow for secure payment for both parties.
- The program depends on Maven to manage dependencies.
- The program depends on Derby to implement an SQL database.

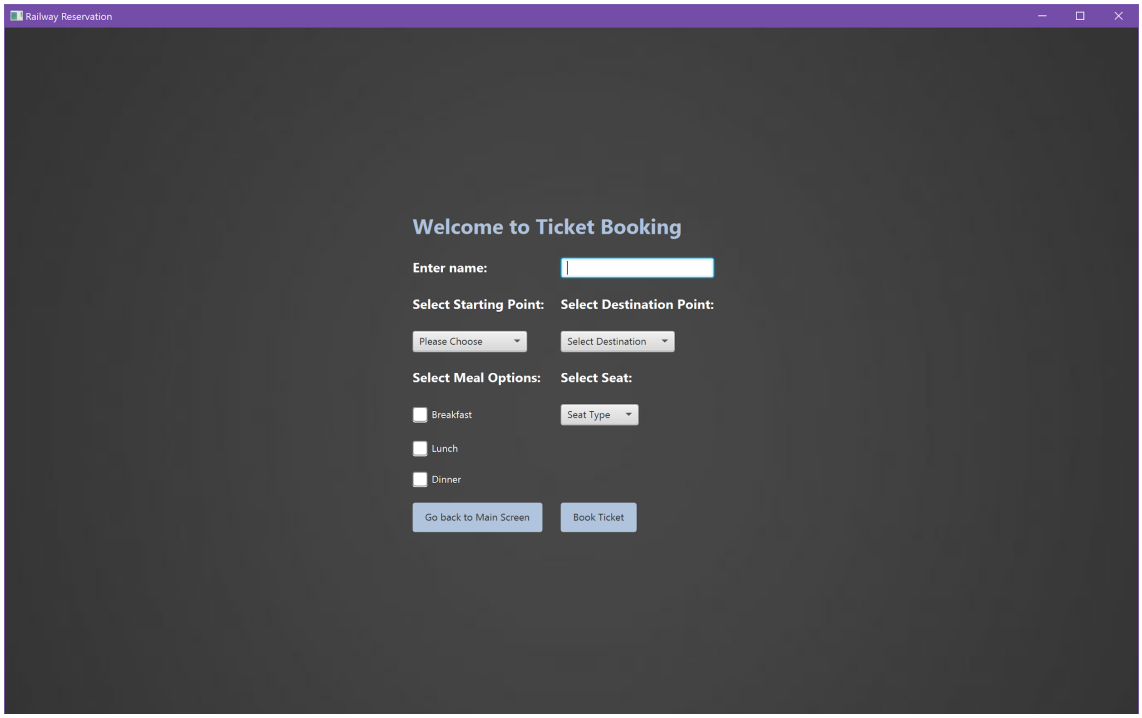
- The program depends on Junit-Jupiter to implement unit testing.

3. User Interfaces

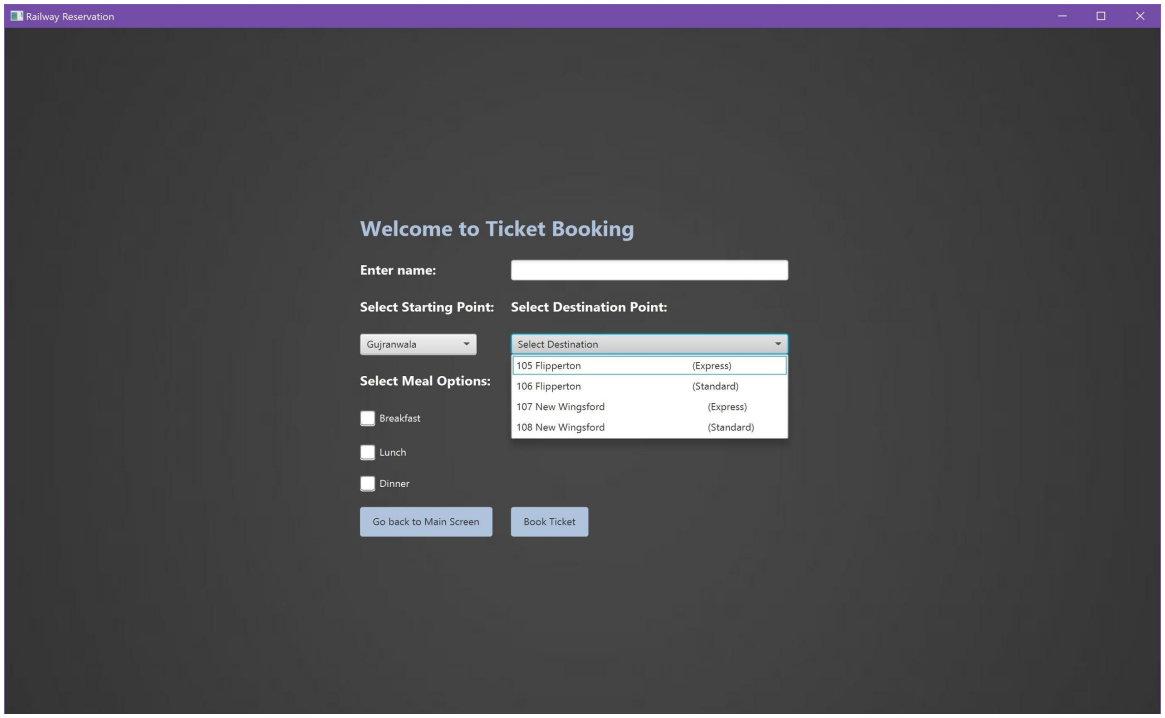
This is the opening scene when running App.java.



If you click Exit Program, the program will terminate.
If you click Book a Ticket Now, you will be redirected to the following page.

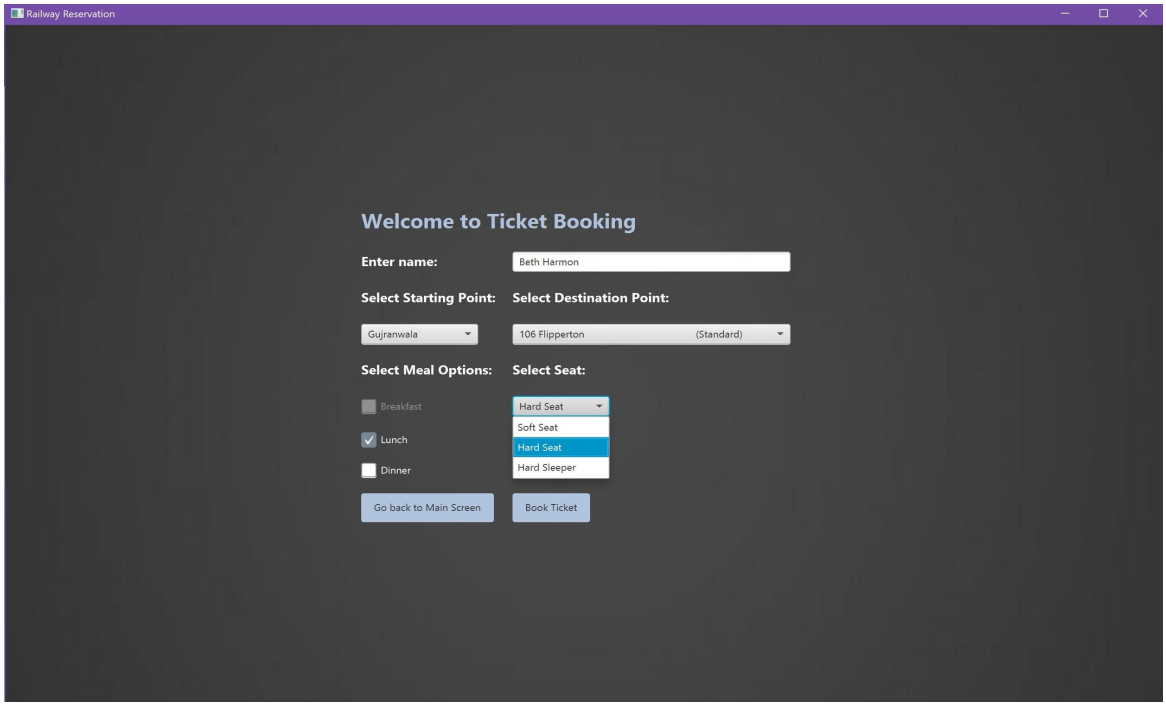


On this screen, you can enter your name, select your starting point, and be presented with the following destination points as shown in the following picture.

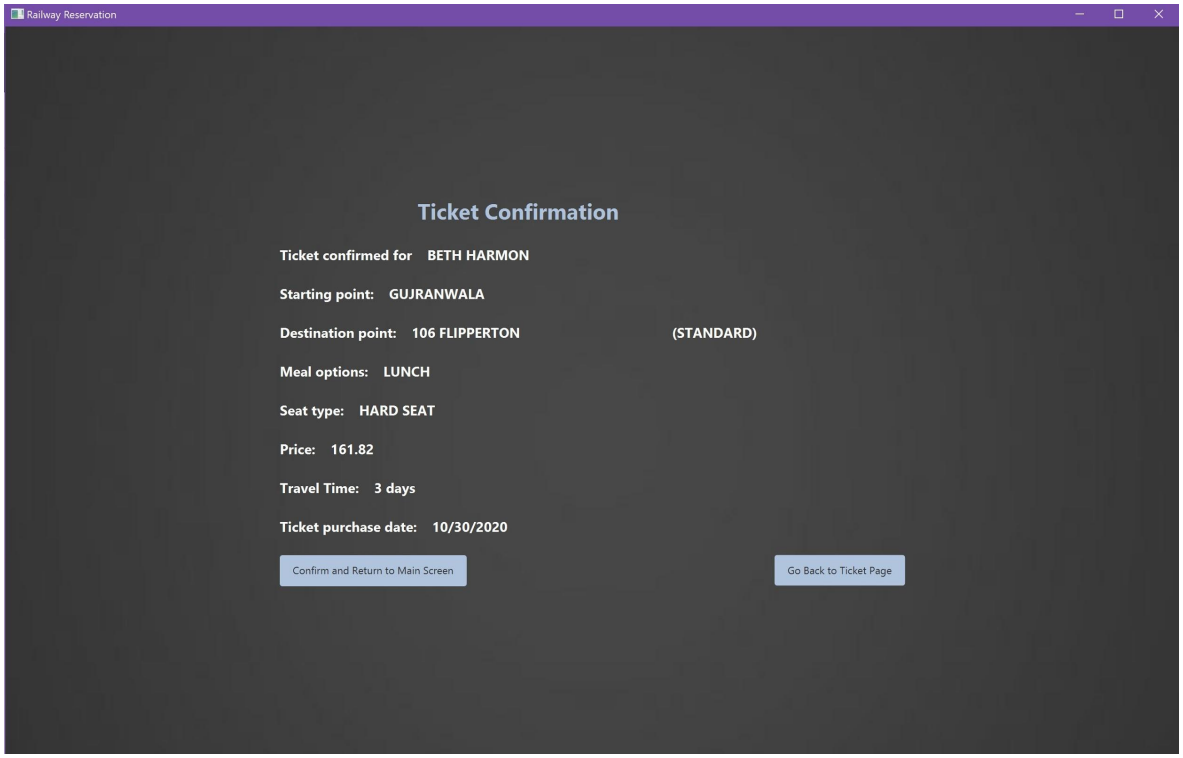


Here, you can see the two types of trains you can select. Express trains will not stop at any intermediate points, and Standard trains have destinations points in-between. (These intermediate points have not yet been added).

Once a starting and a destination point is selected, you may choose what meals and what seating options are available for that train. Some trains don't have all options, as shown below.



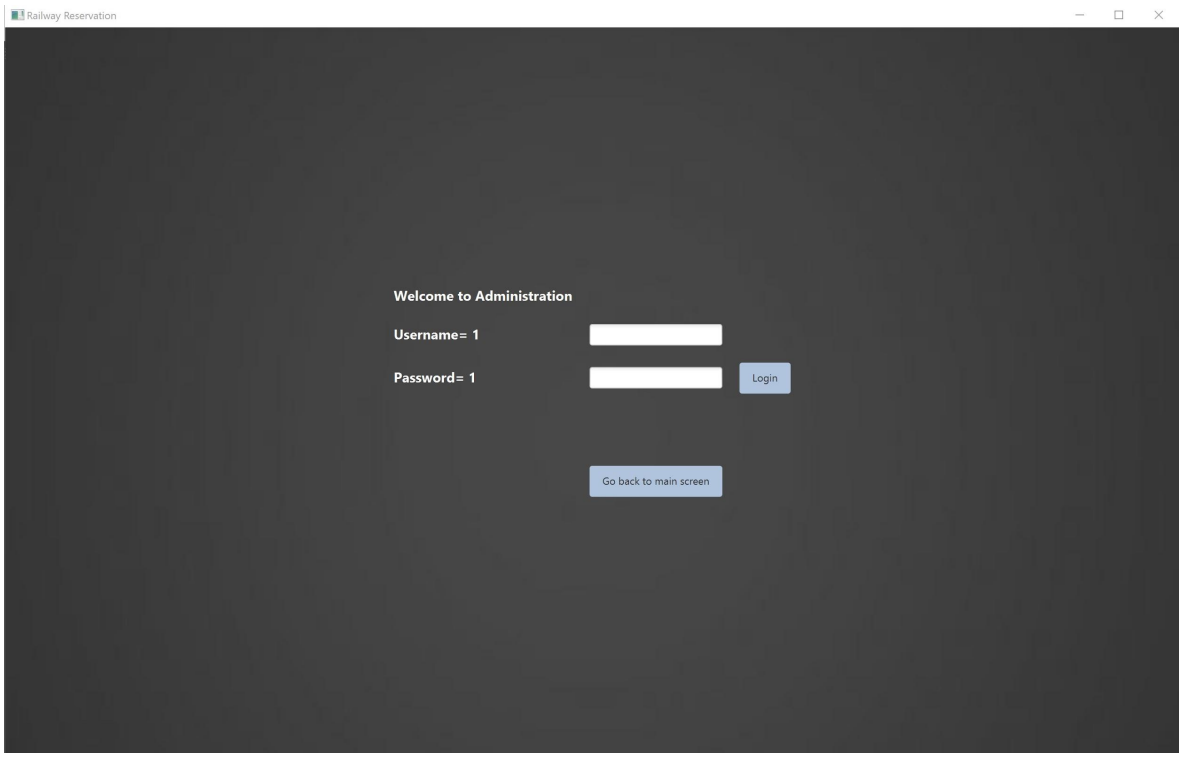
After all options have been selected by the user, the user can either cancel their ticket booking by returning to the main screen, which returns them to the first image above, or they can go ahead and book a ticket. If they choose to book a ticket, they will be redirected to the following page.



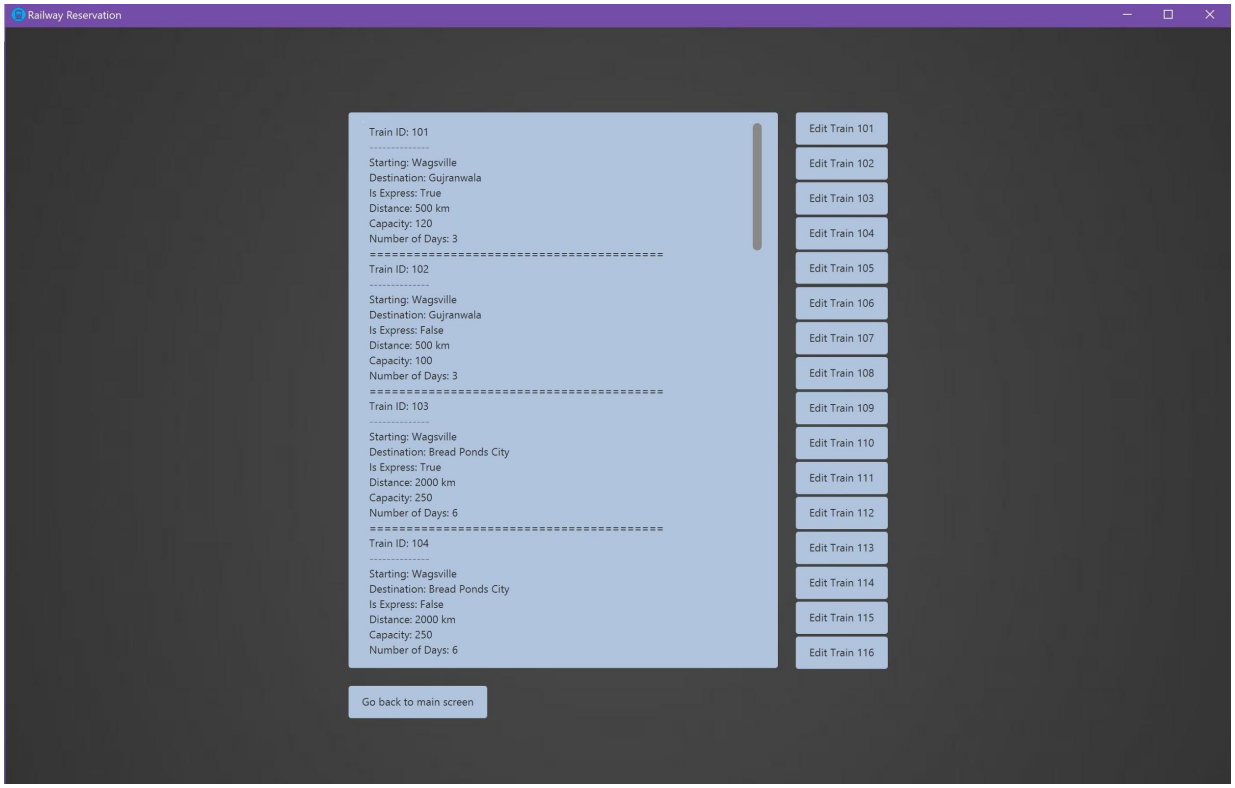
Here, all of the information that the user chose will be presented to them. If they made a mistake, they can return to the ticket booking page by selecting "Go Back to Ticket Page." Otherwise, if they

are done, they may confirm their ticket and return to the main screen by pressing “Confirm and Return to Main Screen.”

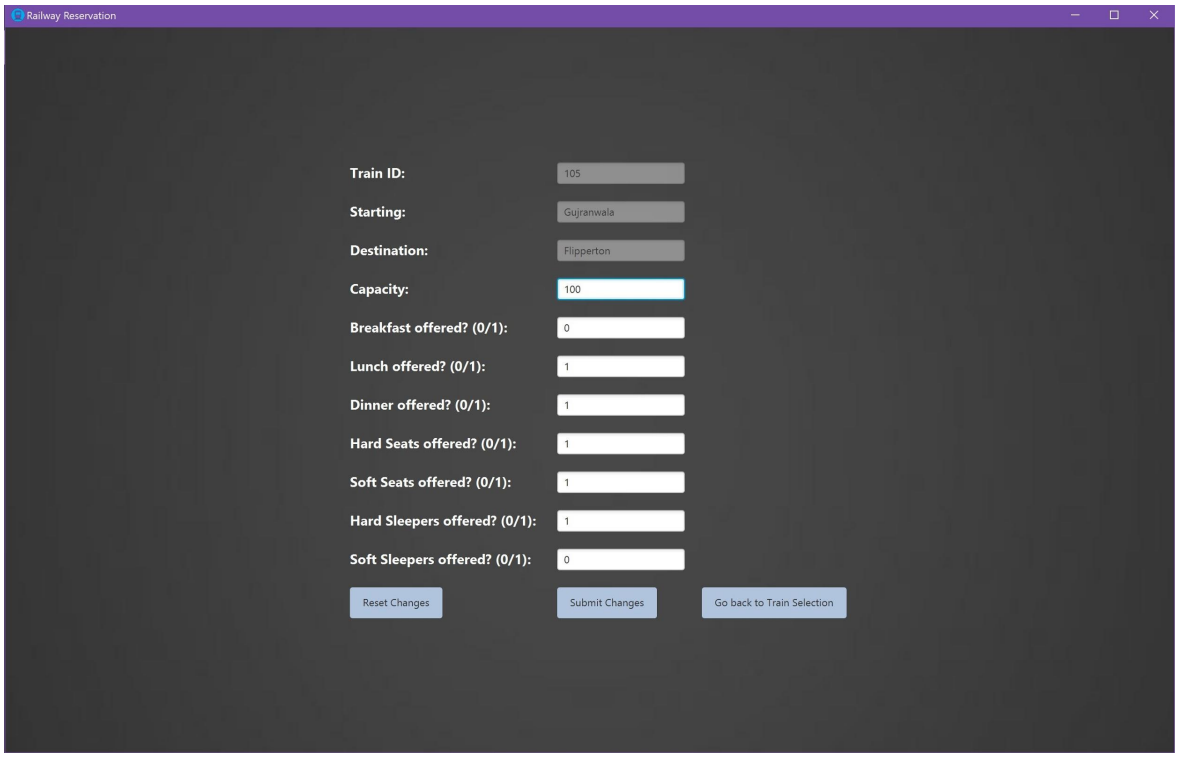
From the main screen, when you select “Sign in as An Administrator,” you will be redirected to a login screen. For the purpose of easy testing, the username and password are given.



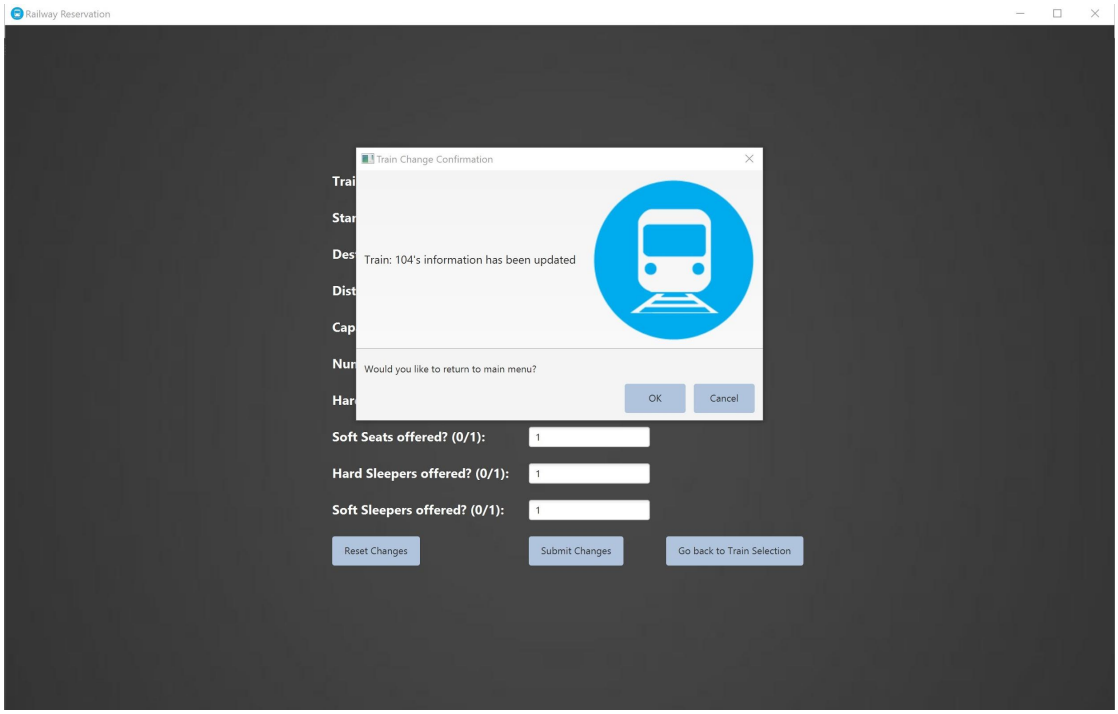
Once the correct login is inputed, you may select the login button. This will redirect you to the back office page. The back office page consists of a list of trains that coincide with a button.



Select the button of the train you wish to edit and you will be redirected to an editing page. Some train characteristics will have text boxes and others won't. This is because some characteristics about a train cannot be changed, such as its route and ID. This is meant to allow for more meals and seats or less meals and seats.

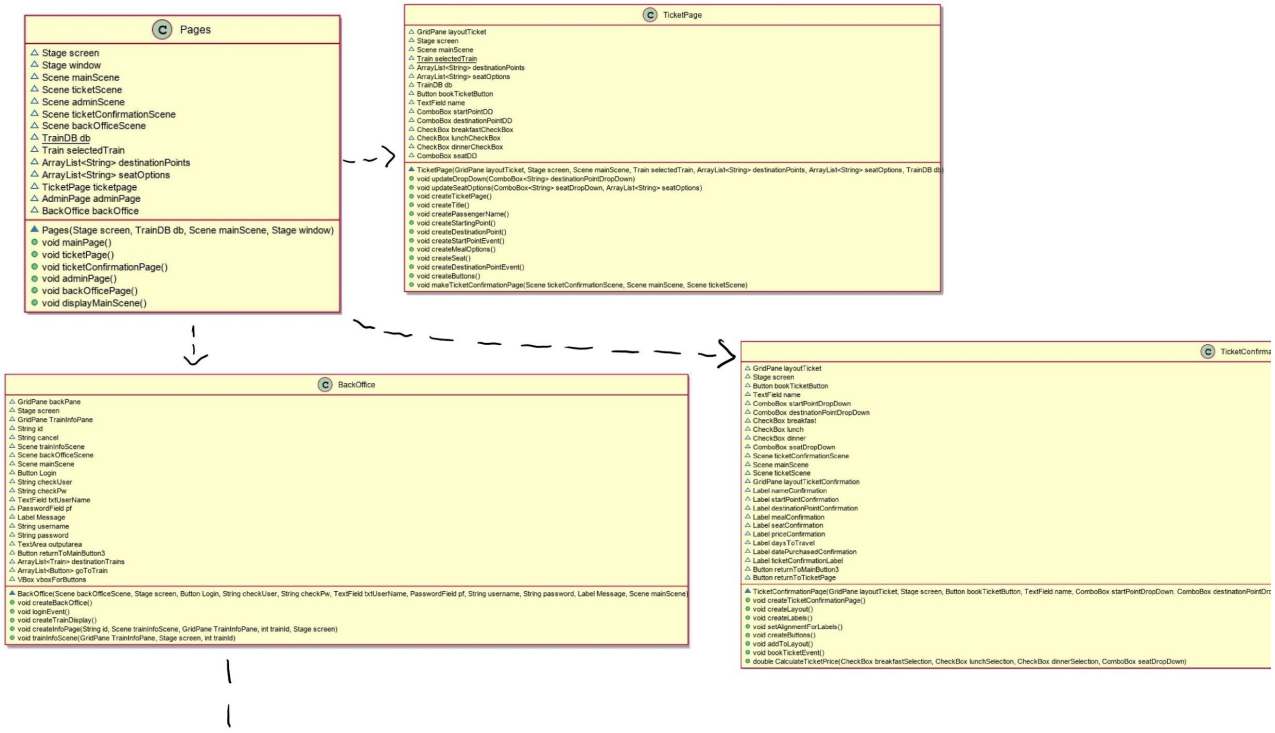


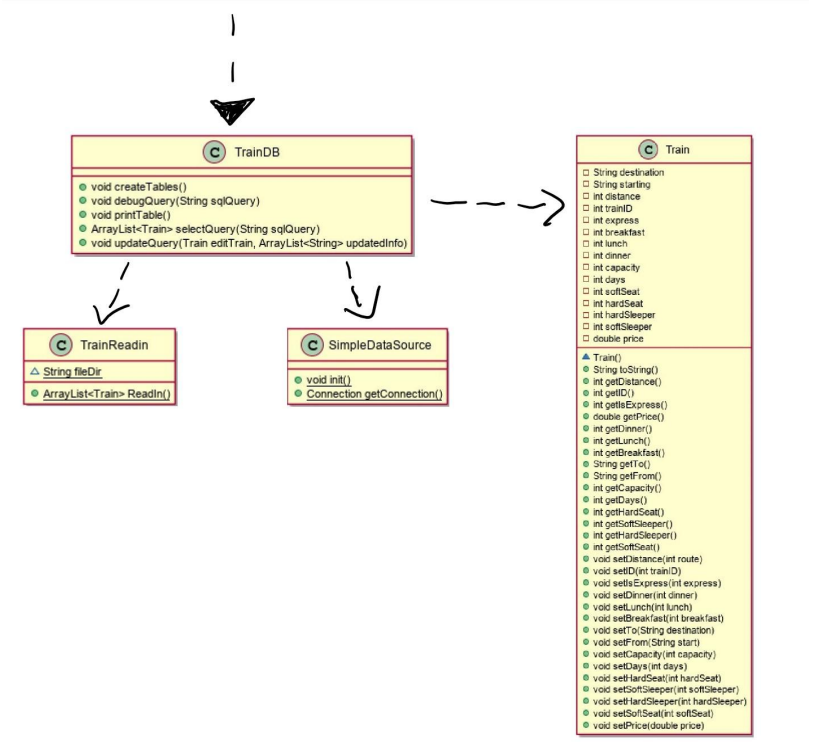
When a change is made, a confirmation pop-up will appear. The “Ok” button will return you to the main screen and the “Cancel” button will return you to the back office page.



4. Class Design

4.1 Class Diagram





4.2 Responsibility of Each Class

Class Name	Class Responsibilities
App.java	Holds Main that creates the database and launches GUI.
Pages.java	Responsible for creating or calling class functions that create scenes including their labels, buttons, layouts, etc.
TicketPage.java	Creates a ticket booking menu and sends user input information to the confirmation page class.
TicketConfirmationPage.java	Takes information from the ticket booking page and displays the users selections as well as the calculated ticket price.
AdminPage.java	Middle screen that requires a valid login to access the “back office” of the program
BackOffice.java	Displays list of trains with correlating buttons that the admin can click to edit a certain train. Creates a new scene for train editing.
Train.java	Object class used to create train objects.

TrainDB.java	Contains definitions for creating the Trains table, and holds methods for querying the table.
TrainReadin.java	Class used for reading in from a CSV to create the Trains table in TrainDB.java
TrainTest.java	Tests functionality of program using JUnit

5. System Features

5.1 Booking a train ticket

5.1.1 Description

Customer users will be able to book a ticket on their desired train, with their options of starting point, destination, seat type and meals, and express or standard trip.

5.1.2 Related User Classes

The users for this system feature will be train passengers.

5.1.3 Functional Characteristics

- *The system allows the user to choose their starting point and destination. (Complete)*
- *The system allows for the user to choose an express or standard train. (Complete)*
- *The system allows for the user to choose available meals on their train. (Complete)*
- *The system allows the user to choose from available seating options. (Complete)*
- *The system allows the user to choose their starting point and destination. (Complete)*
- *The system allows for the user to choose an express or standard train. (Complete)*
- *The system allows for the user to choose available meals on their train. (Complete)*
- *The system allows the user to choose from available seating options. (Complete)*

5.1.4 Non-Functional Characteristics

- *The user will not be able to choose a route that does not have an existing train available to it. (Complete)*
- *The user will not be able to choose seat options or meals not available on the chosen train. (Complete)*
- *The user should receive immediate confirmation after booking their ticket. (Complete)*
- *The user's private information (name) will not be stored in the database. (Complete)*

5.1.4 Testing

- *Tests have been implemented to test the functionality and ensure that everything is working as expected.*

5.2 Administrator edit trains

5.2.1 Description

*Administrator users will be able to view all train information on trains that are currently housed in the database. The user will then also be able to edit **non-critical** information on the train.*

5.2.2 Related User Classes

The users for this system feature will be train administrators.

5.2.3 Functional Characteristics

- *The system allows the user to access an admin side of the program. (Complete)*
- *The system allows for the user to enter administrator login credentials. (Complete)*
- *The system allows for secure administrative access. (Complete)*
- *The system allows the user to see all trains currently in the database. (Complete)*
- *The system allows the user to choose to edit a train. (Complete)*
- *The system allows for the user to edit dining options available on the train. (Complete)*
- *The system allows for the user to edit seating options available on the train. (Complete)*
- *The system allows the user to reset, cancel or confirm changes. (Complete)*

5.2.4 Non-Functional Characteristics

- *The user will not be able to add trains. (Complete)*
- *The user will not be able change train ID, route distance or capacity. (Complete)*
- *The user should receive immediate confirmation after submitting changes. (Complete)*

5.2.4 Testing

- *Testing has been done to ensure that updating a train's characteristics will update the train database accordingly.*

6. Data Storage and Schema

6.1 Data Sources

src/main/java

- App.java
- SimpleDataSource.java
- SystemInfo.java

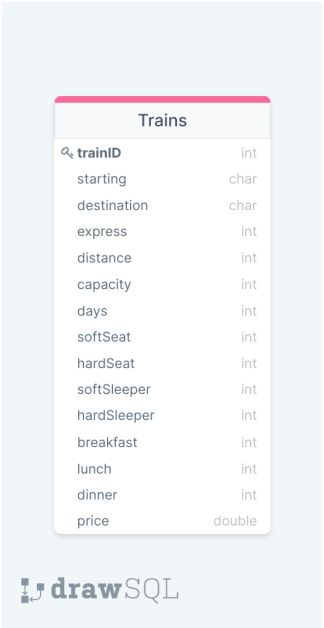
- Train.java
- TrainDB.java
- TrainReadIn.java
- TicketPage.java
- TicketConfirmationPage.java
- AdminPage.java
- BackOffice.java
- Pages.java
- Inventory.csv
- Stops.csv
- mainmenu.css
- ticketpage.css
- adminpage.css
- main_background.jpg
- ticket_background.jpg
- trainIcon.png

src/test/java

- TrainTest.java

6.2 Formats and Schema

SQL database schema:



7. Known Issues and Bugs

- Medium priority: When a user attempts to book a ticket without selecting an item from every dropdown, an error occurs, but the GUI still functions. (CURRENT ISSUE)

- Low priority: When a user attempts to book a ticket and changes their starting point dropdown after choosing a destination, an error occurs, but the GUI still functions.(FIXED)
    - We need to add error messages for users if they do not select all options needed to book a ticket, such as not selecting a seat. (CURRENT ISSUE)
    - We need to impliment error handling for ensuring that a user selects a starting and destination point before they select meal options and a seating option.(FIXED)
- Medium priority: Not all destinations are currently listed.(CURRENT ISSUE)
- Low priority: When a user returns to the main menu after filling out the ticketing screen, the choices are saved. These choices should be wiped any time a user leaves the ticketing screen. (CURRENT ISSUE)

Appendix A: GitHub Information

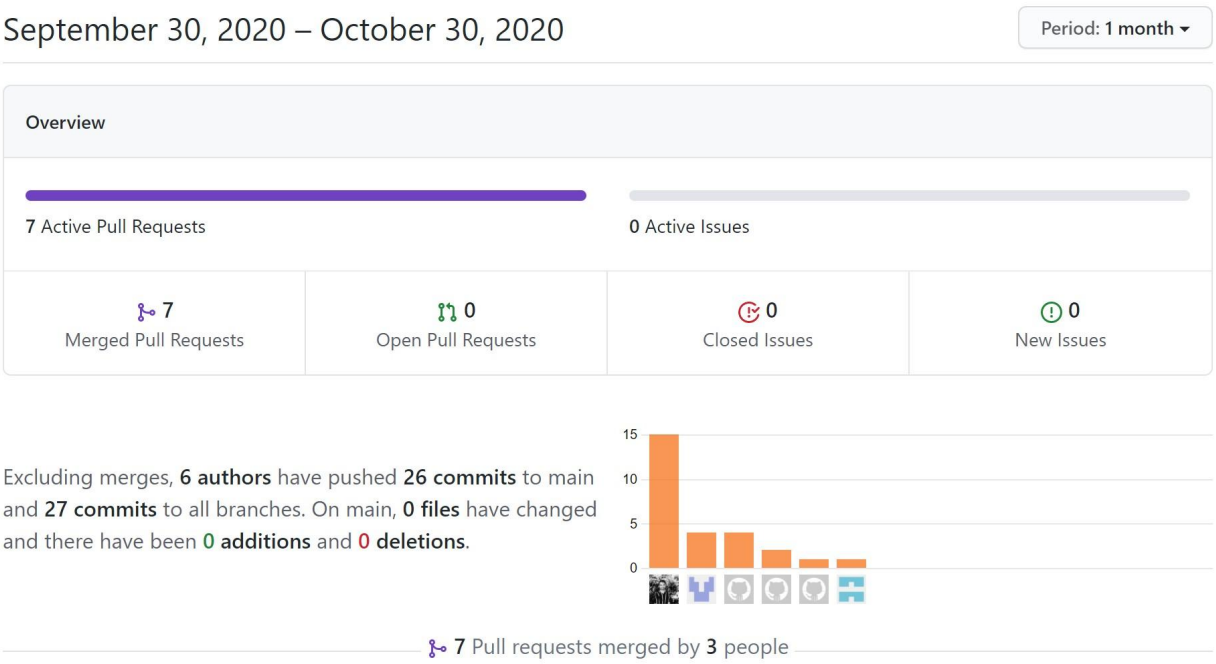
cen3031-80066-group4: **https://github.com/UWF-HMCSE-CS/cen3031-80066-group4**

Video Demonstration: https://www.youtube.com/watch?v=5HVNqJVuz4E

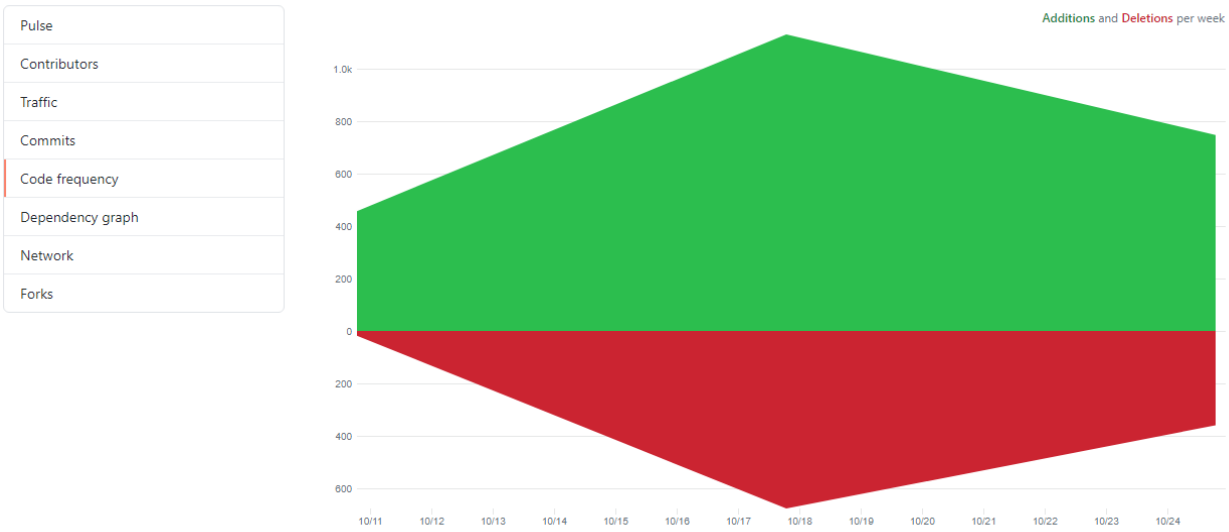
The team has not used any issue tracking systems or wiki as of yet, but will likely implement both in the future.

Contributions Screen:

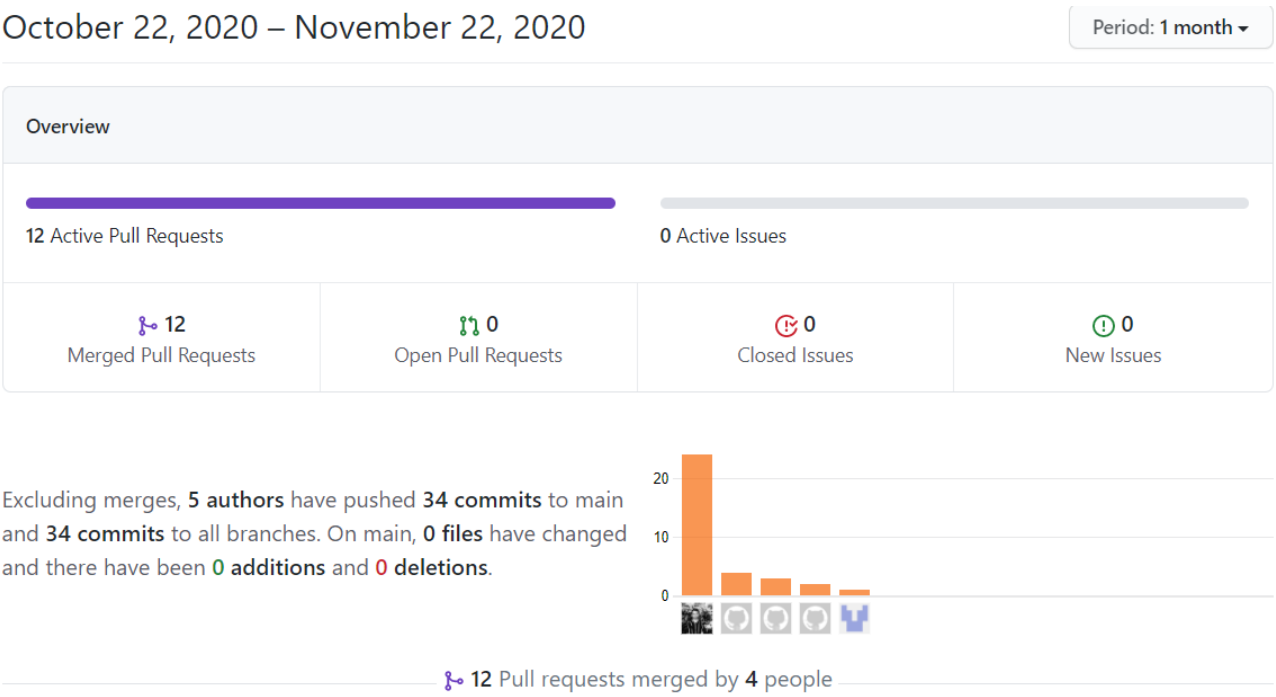
Sprint 1:



Code Frequency:

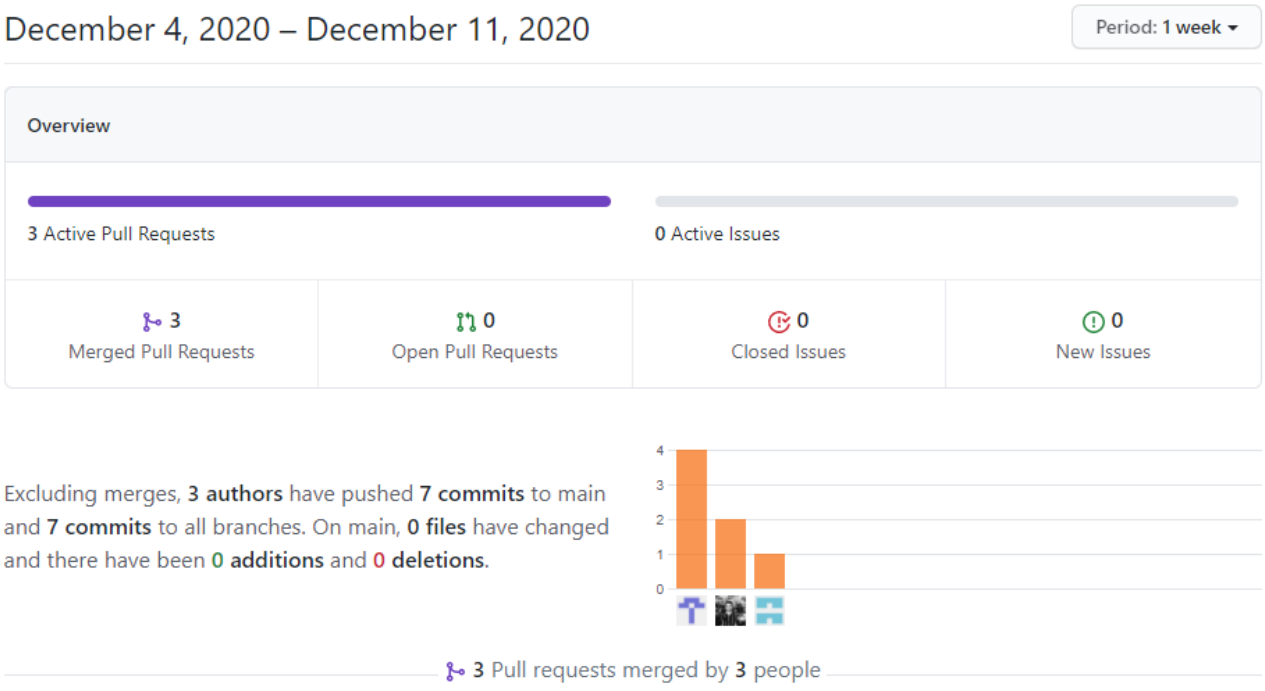


Sprint 2:



Code frequency:

Sprint 3



Code frequency:

Appendix B: Required Reports (Optional)

The software does not generate reports.

Appendix C: Glossary

There are not any terms which need to be defined in order to interpret this document.