

Milestone 3 Report

Engicoders

H. Bird, B. Karacelik, J. Peters, J. Ropotar, A. Rybka

Department of Computer Science, University of British Columbia

COSC310: Software Engineering

Dr. Gema Rodriguez-Perez

March 11, 2024

Sequence Diagrams

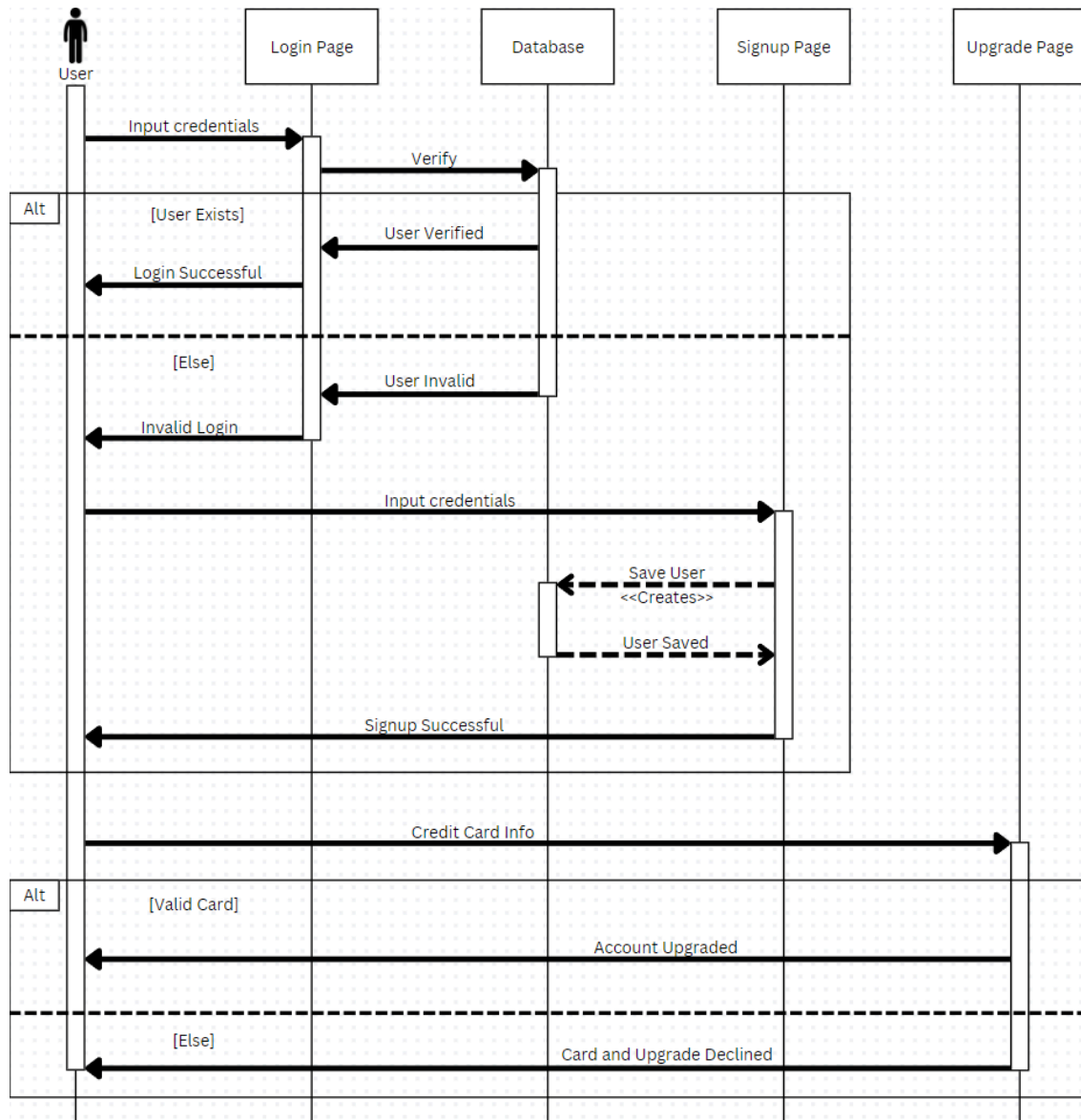


Figure 1. User Login/Signup/Upgrade Account Sequence Diagram

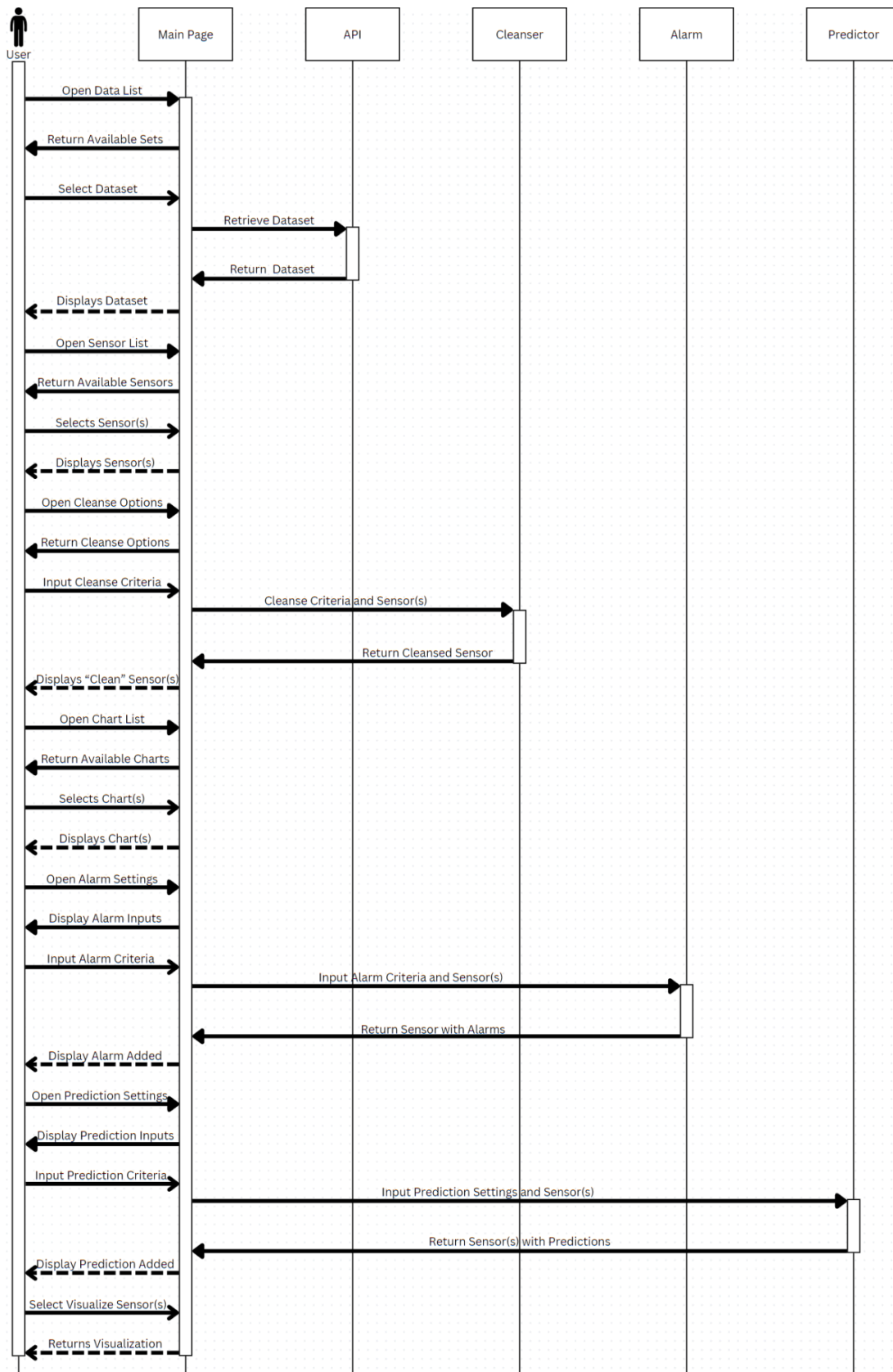


Figure 2. User Main Page Sequence Diagram

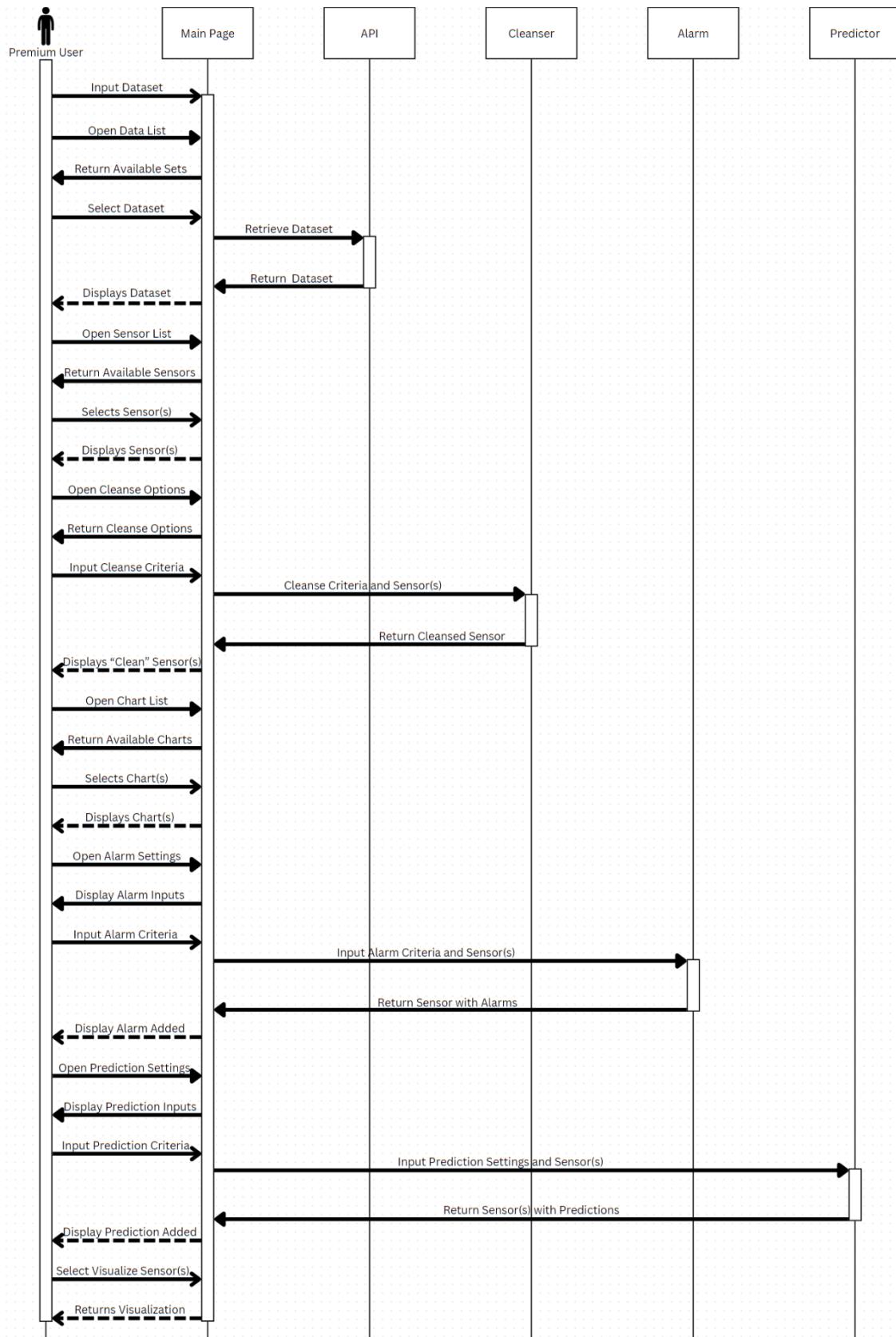


Figure 3. Premium User Main Page Sequence Diagram

Shown above in Figure 1-3 are the sequence diagrams of our project. To start, Figure 1 is the typical login page sequence. The user must input their username and password which will be verified against the database. If the user exists, they will gain access to the main page. If the user does not exist, they will be prompted to input their username and password into the sign up screen which will then save their credentials into the database for future use. The new (or existing) user can then choose to upgrade the account, which will prompt them to input their credit card info. If the credit card is valid, the account will be upgraded, if not their account will remain unchanged.

Figure 2 illustrates the typical main page sequence of a common user. Once on the main page (after logging in), the user will open a drop-down list of available datasets and once open, they will select one to use. This will then get the dataset from the api and show the user it has been selected. Next, the user will open a list of sensors within the dataset and select the ones they want to use. If the user wants to cleanse the sensor(s), they will open the cleanse option and input their requirements which will send the sensor and requirements to the cleanser and return a message with the new sensor object. The user will then open up the available charts and select which one they want to use. If the user wants alarms, they will open up the alarm settings, input the specifications they want for the alarms, and send the data. The sensor will then be returned with the added alarm. If the user wants to add a prediction to the alarm, they can open the prediction tab and input their prediction requirements. Once input, the data will be sent to the predictor and the sensor with the predictions will be returned. Finally, the user can select to visualize the data they manipulated.

Figure 3 shows the typical sequence diagram for premium users on the main page. This diagram is the exact same as Figure 2 for the common user and the main page, but it allows the premium user to input their own dataset before selecting datasets. This allows the premium user to use the software on their own data along with the data from the api.

Class Diagrams

Words

Test Plan

words

Design Patterns

Observer:

For our project, we have chosen to implement the observer pattern. The observer pattern is useful for the IoT web application because it allows for dynamic updating and communication between objects in the GUI and event-driven systems. This is particularly useful not only as sensor values change but as sensors are swapped. For this application there will be an abstract

Subject class where a sensor object is the subject. For each subject the abstract Observer class will maintain a list of observers. These observers will include Alarm, Visualization, and Prediction classes. When the subject, or sensor, updates or is replaced the observers will then have an update() method that will be called. An illustration of this pattern is shown below in Figure 5.

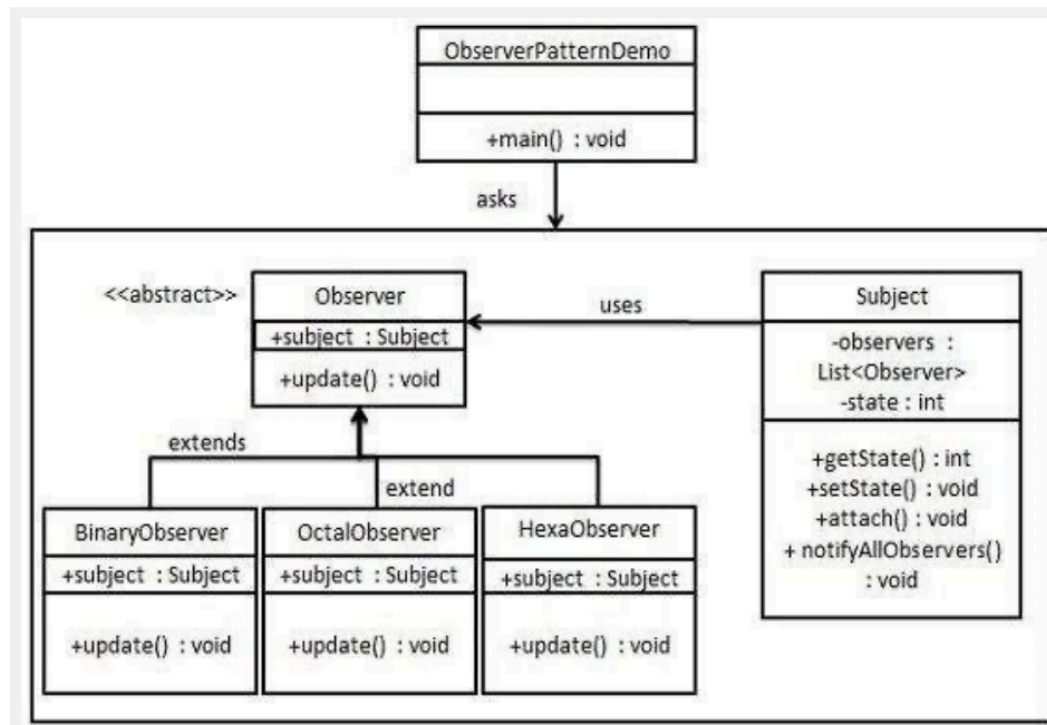


Figure 5. Observer Pattern Example

MVC Pattern:

Another design pattern our project will integrate is the Model-View-Controller pattern (MVC); which is commonly applied for web applications. In our use case, it has the ability to separate the applications data, UI, and control logic into manageable modules. This means our UI will be built using flasks which will be separate from our data. Our data will be a separate class representing the queries and GET REQUEST interactions with thinkspeak. Finally, our control module will manage the sensors, alarms, and visualizations. By using MVC these separate concerns can be developed, tested, and maintained easier. A diagram of the MVC pattern can be found below in Figure 6.

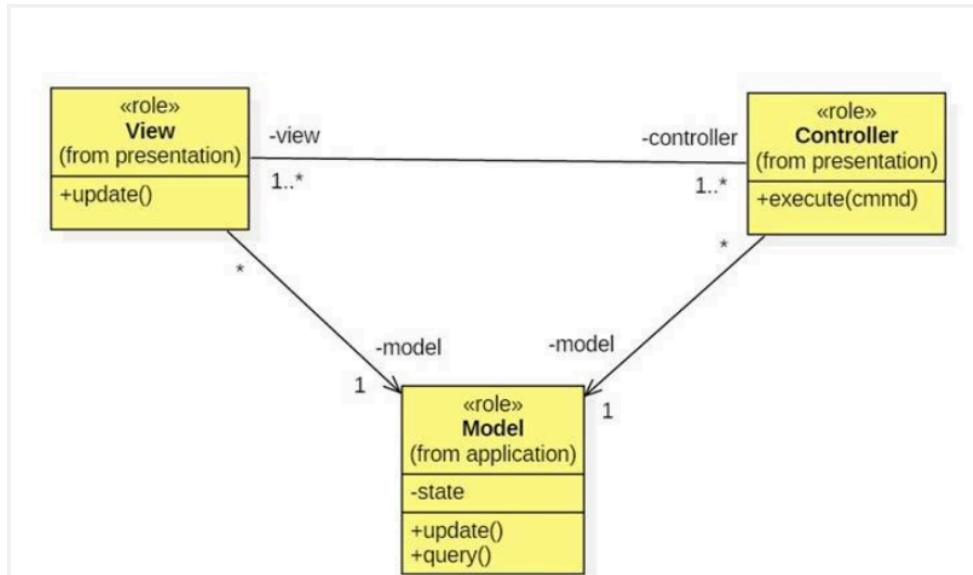


Figure 6. MVC Pattern Example