

# **CSc-131**

## **Computer Software Engineering**

**Instructor: Doan Nguyen, Ph. D**

**Summer 2018**

### **Deliverable #3: Software Design Document**

Calvin Saechao

Daniil Kistanov

Dennis Dang

Kenneth Godfred Echaluse

Sangwook Park

**Due Date: 07/02/2018**

## **Goal of the Project**

The goal of the project is to design a web application that will take students attendance by visiting a webpage and filling up a form that will be validated in professor's Google sheet. The web application will also have an instructor's page where the instructors can generate a key to be given to students, and be able to update attendance grades with a push of a button.

## **Problem Statement**

Currently, students are not able to submit attendance onto the campus's Canvas LMS attendance tracker and professors would have to individually call student names. Primarily because of this issue, lecture time is being wasted.

## **Proposed Solution**

Our application will help the students and professors to easily manage the attendance through a web interface and saving the data to a google spreadsheet. The professor may generate a CVS report file to be uploaded to the canvas to update student attendance.

The advantage of this web application is that it is accessible by different devices. It can also calculate the location of the student to prevent taking attendance outside class. The instructor can generate key to be given to students, and update attendance grades as need with just a push of a button.

## Functional and Non-functional Requirements

For our web application, we identified 5 functional requirements, and 3 non-functional requirements. The most important functional requirements of the project is to make the web application generate unique key to be used by students to take attendance (FR\_3) and for students to be able to take attendance to the web application (FR\_2). The table below shows the list of functional requirements and their descriptions.

Functional Requirement #	Priority	Requirement Description
FR_1	3	Professor shall be able to manually take attendance entry for a student.
FR_2	2	System shall be able to provide a form of submission for the student inside a textbox allowing phrasal response from the student as a form of attendance.
FR_3	1	The web application shall be able to generate a unique four-digit key for the professor and a website URL to share with students to take attendance.
FR_4	4	The professor shall be able to use the web application to calculate attendance grade for each student record.

**Table 1: Functional requirement list**

For non-functional requirements, the most important requirement we have to implement is by using a geolocation calculator to prevent students from taking attendance outside class (NFR\_1). The table below shows our non-functional requirements and their descriptions.

Non-Functional Requirement #	Priority	Requirement Description
NFR_1	1	Only students that are attending the current lecture session shall be able to submit an attendance tracking request to the system. No one externally should be able to submit this request outside of class.
NFR_2	2	Usability: The students shall be able to take attendance by typing five inputs: Student ID, the key and email provided by professor, the class section, their quiz answer to submit their attendance.
NFR_3	3	Documentation: The system's source code shall have javadoc documentation.

**Table 2: Non-functional requirement list**

## Methodology Used

The method behind developing our web application is the agile method. This web application project was developed under a span of 6 weeks. Since most of us don't have prior experience working with APIs and open source coding, we have to learn as we implement our methods.

Java Servlets, HTML5, CSS
Version Control - Open Source - Github
Implemented Google sheets API
Agile Method (Weekly Iterations)
Microsoft Azure Web deployment service
Implemented Google geolocation API

**Table 3: Methodology and Tools used for developing**

### I. Entity Relationship Diagram

Our Entity Relationship Diagram is as follows:

For our ERD, we have 4 entities: Class Session, Student, Instructor, and Course.

- *Class Session*, is a weak entity. Class Session depends on other attributes from Student (SID) and *Instructor* (Class Name, Location).
- *Student* has Professor's Email, their Student ID, geolocation, and their attendance message.
- *Course* has its own SheetId that google classifies each GoogleSheet document.
- *Instructor* has their own email and password for our web application to login, Class Name (sheetName specified in the GoogleSheet).

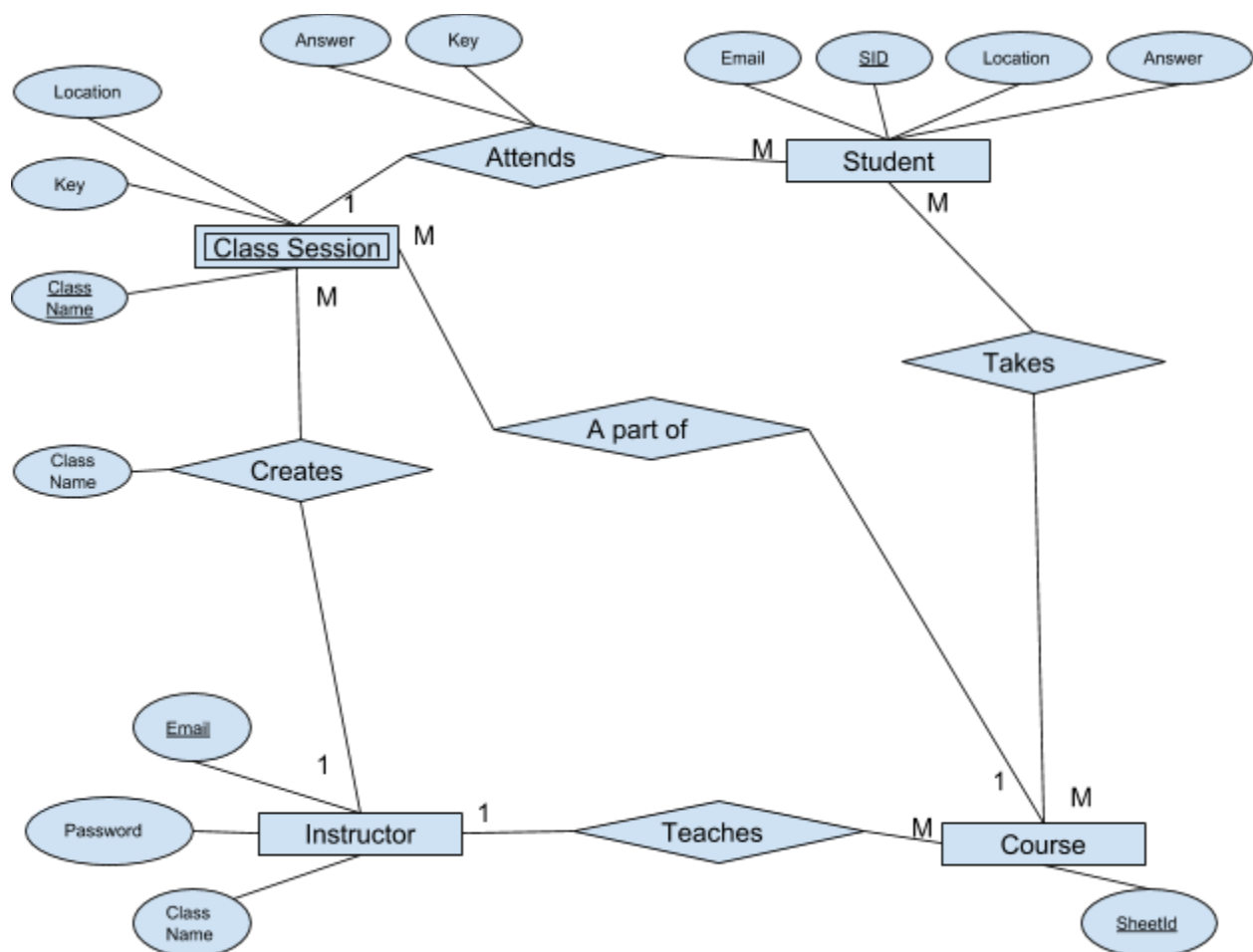
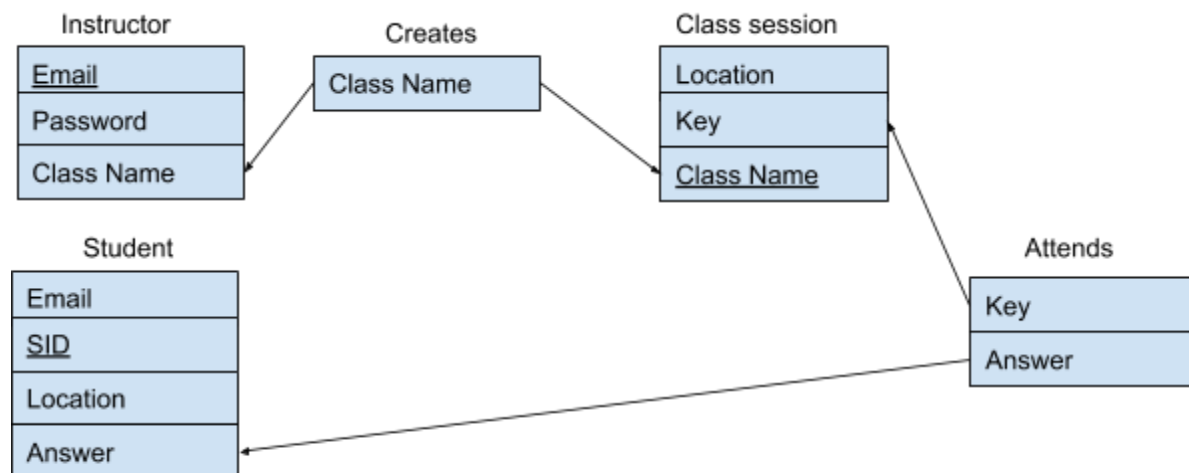


Figure 1: Entity Relationship Diagram



Class Name are foreign keys used to represent the one to many relationship.

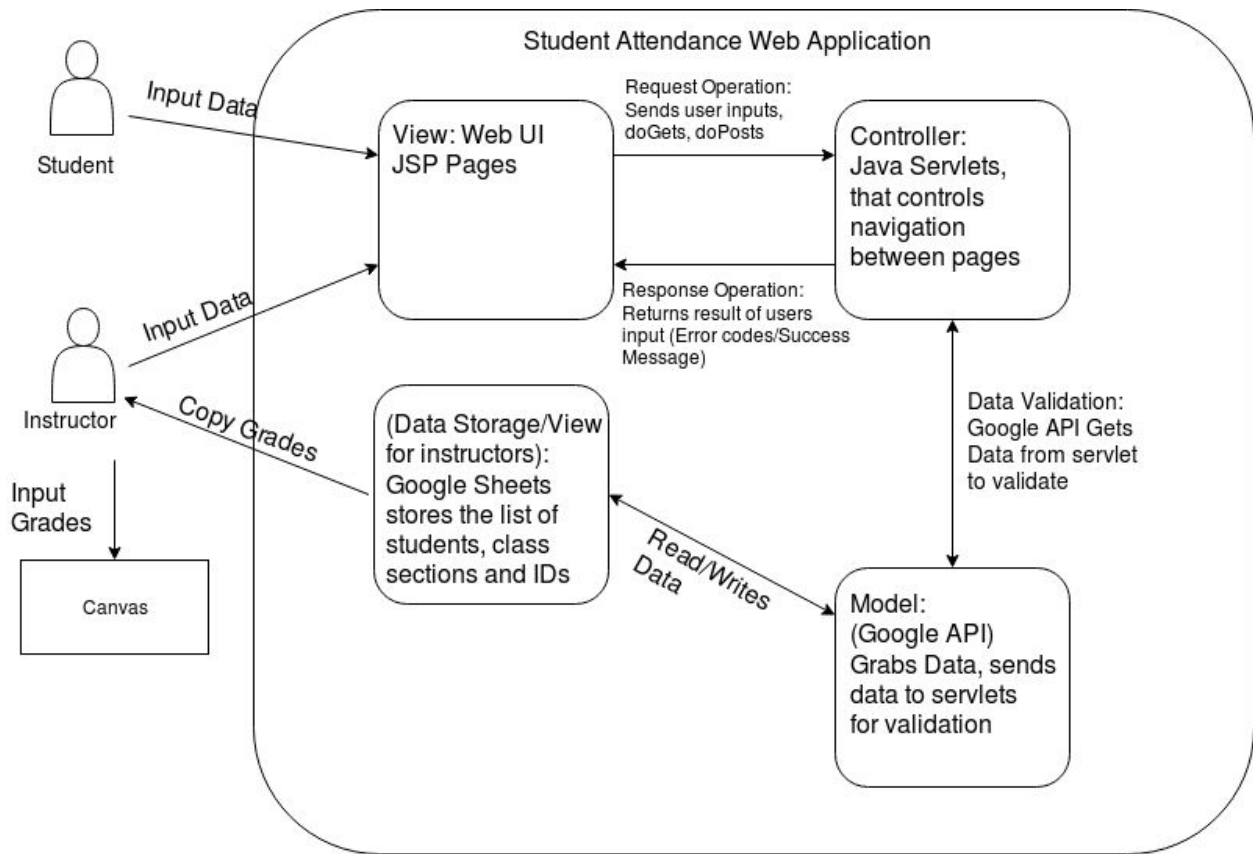
Key is a foreign key used to represent the one to many relationship.

Figure 2: Entity Relationship Table Diagram

## II. System Architecture Design

For our system's architecture, we chose to go with the Model-View-Controller (MVC) design.

- *The Model* representation on our web application will be the Google Sheets and its DataStore. The Google Sheets API has its own data management functionalities to help maintain data that is passed to and from.
- *The View* representation of our application will be the webpages (JSP pages). The webpages will be the main form of interaction for our users to communicate with our web application. After using our product, the Google Sheet should also be a View representation too. This is because the professor would be able to see the confirmation of students taking attendance.
- *The Control* representation of our application will be the Java Servlet. It is in charge of handling data transfer to and from the View and the Model.

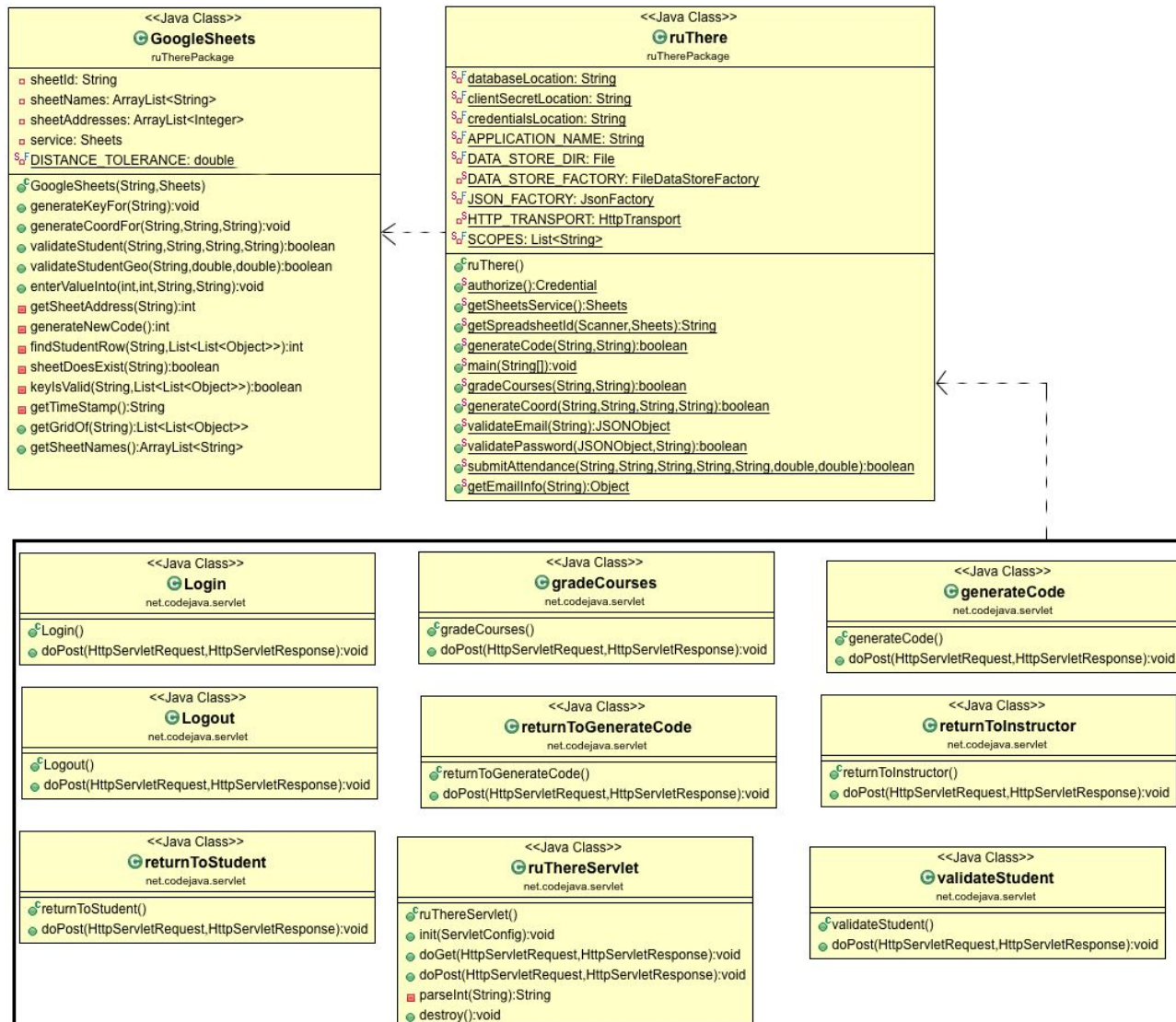


**Figure 3: Model View Controller (MVC) Diagram**

In-Detail description of Figure 3:

When students input the information on the JSP Page UI and it goes to Java Servlet then goes to Google Sheets API. This step will check the information is matching or not and if it is not match then it will send the error message. The error message will send to Java Servlet and JSP Page UI. Once student inputs the correct information then it will pass to Google Sheets. And then professor can see the all information on Google Sheets. Professor is able to grade them manually on Canvas.

### III. UML Class Diagram



**Figure 4: UML Class Diagram**

For the scope of the project, we disclaim that the design is not object-oriented programmed, so the UML class diagram may not give a great idea of how our classes are structured. However, this design was emphasized more on functional programming.

Main UML Class Diagram components:

- The grouped servlet classes located on the bottom of Figure 4 are responsible for processing user input to the main ruThere class and redirecting the user's webpage to another.
- The ruThere main class
- The GoogleSheets class serves as a helper class to refactor the ruThere main class.
  - This includes helper methods such as for entering specific data in certain cells and validating student input

All of these classes are a dependency association to each other. To further explain, the grouped servlet classes are 'using' methods from the ruThere main class. The ruThere main class is 'using' methods from the GoogleSheets class.



## IV. State Diagrams

The state diagrams address the general flow of using our web application for two types of our intended users: the Instructor and Student.

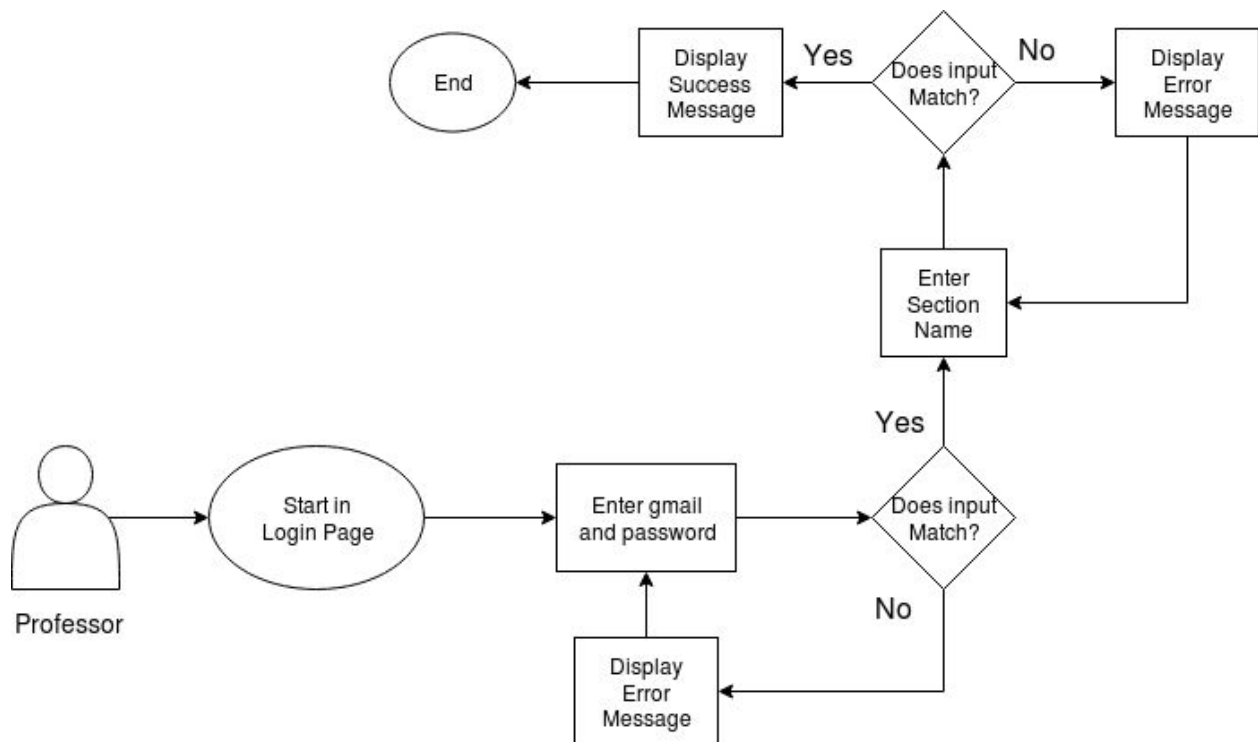


Figure 5: Instructor State Diagram

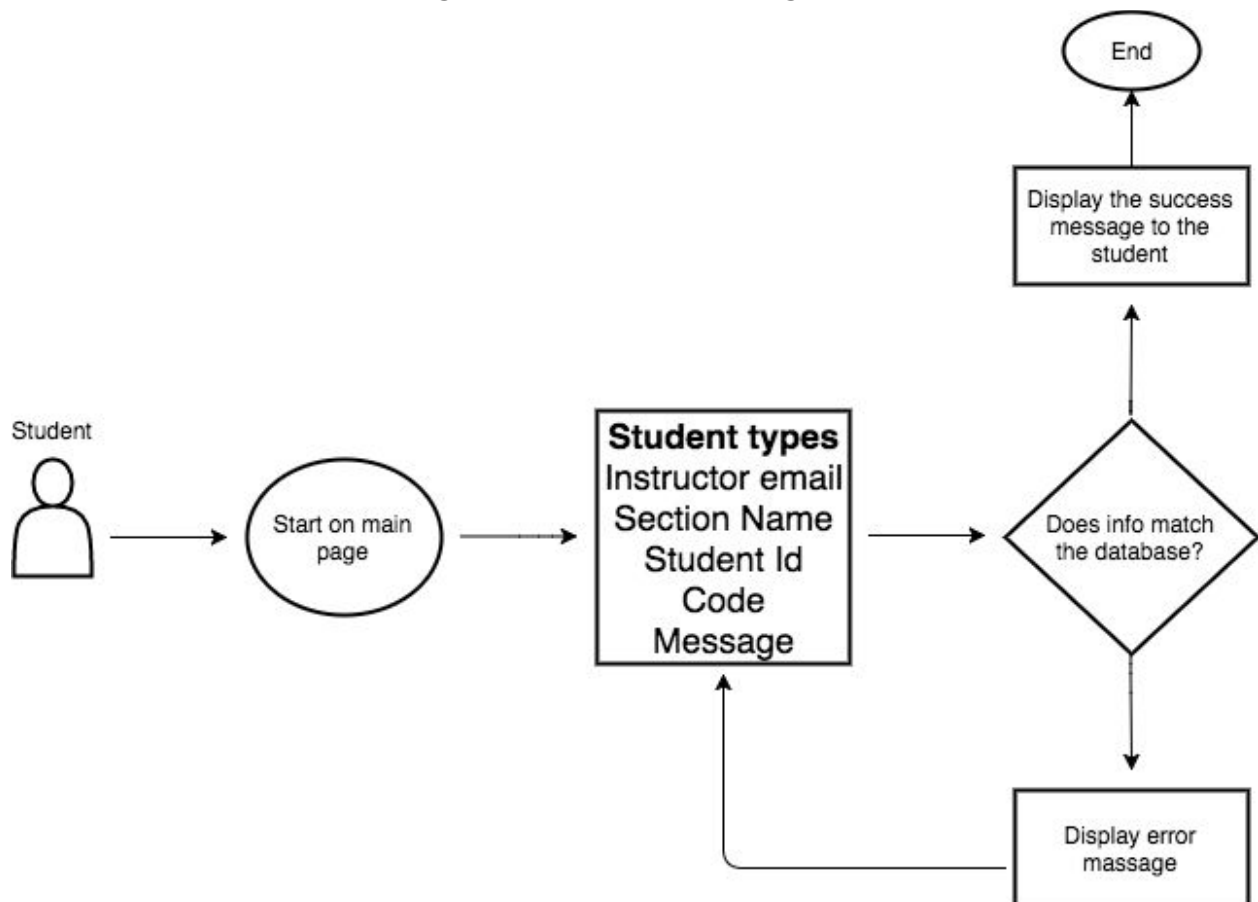


Figure 6: Student State Diagram

## V. Sequence Diagrams

The sequence diagrams address the general flow of using our web application for two types of our intended users: the Instructor and Student. However, this is more of a detailed flow of steps of what is happening internally for our web application.

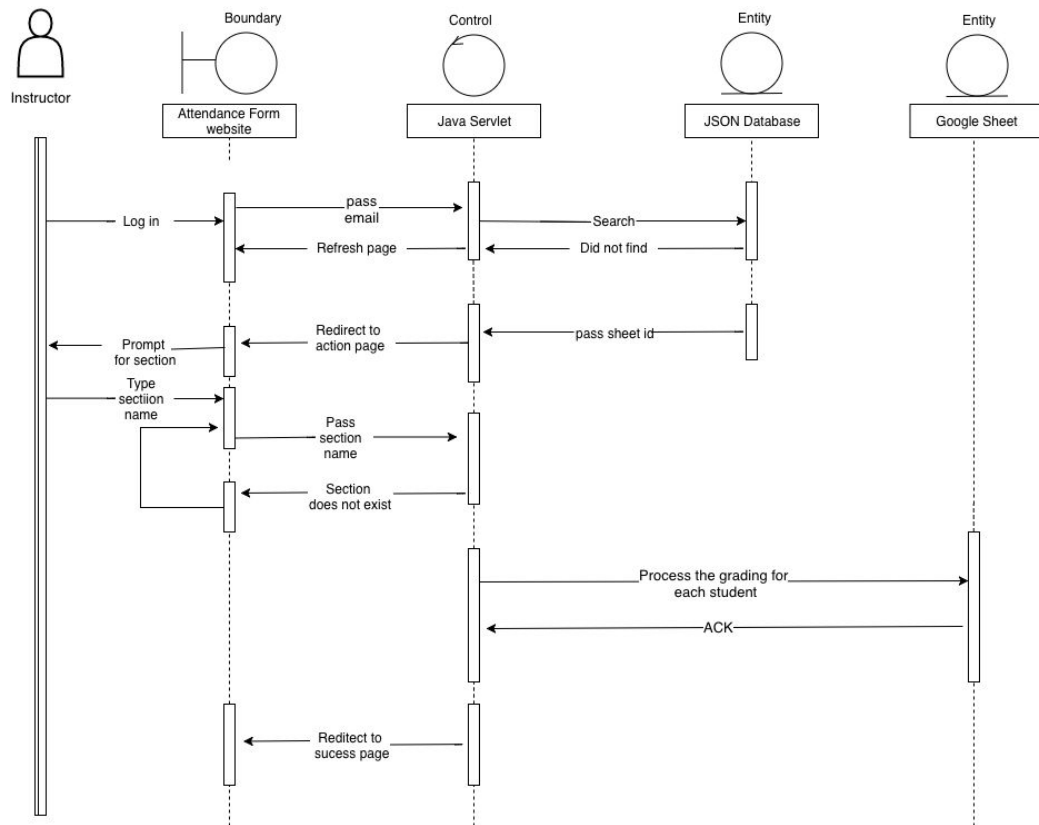


Figure 7: Instructor Sequence Diagram

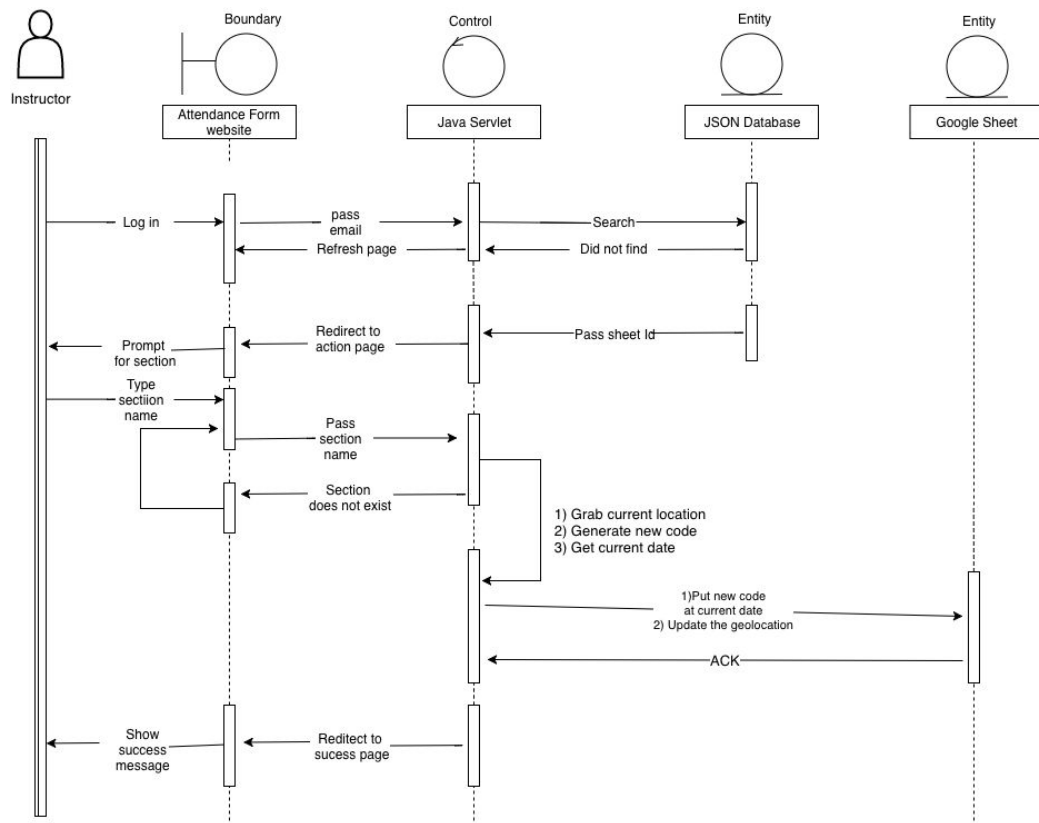
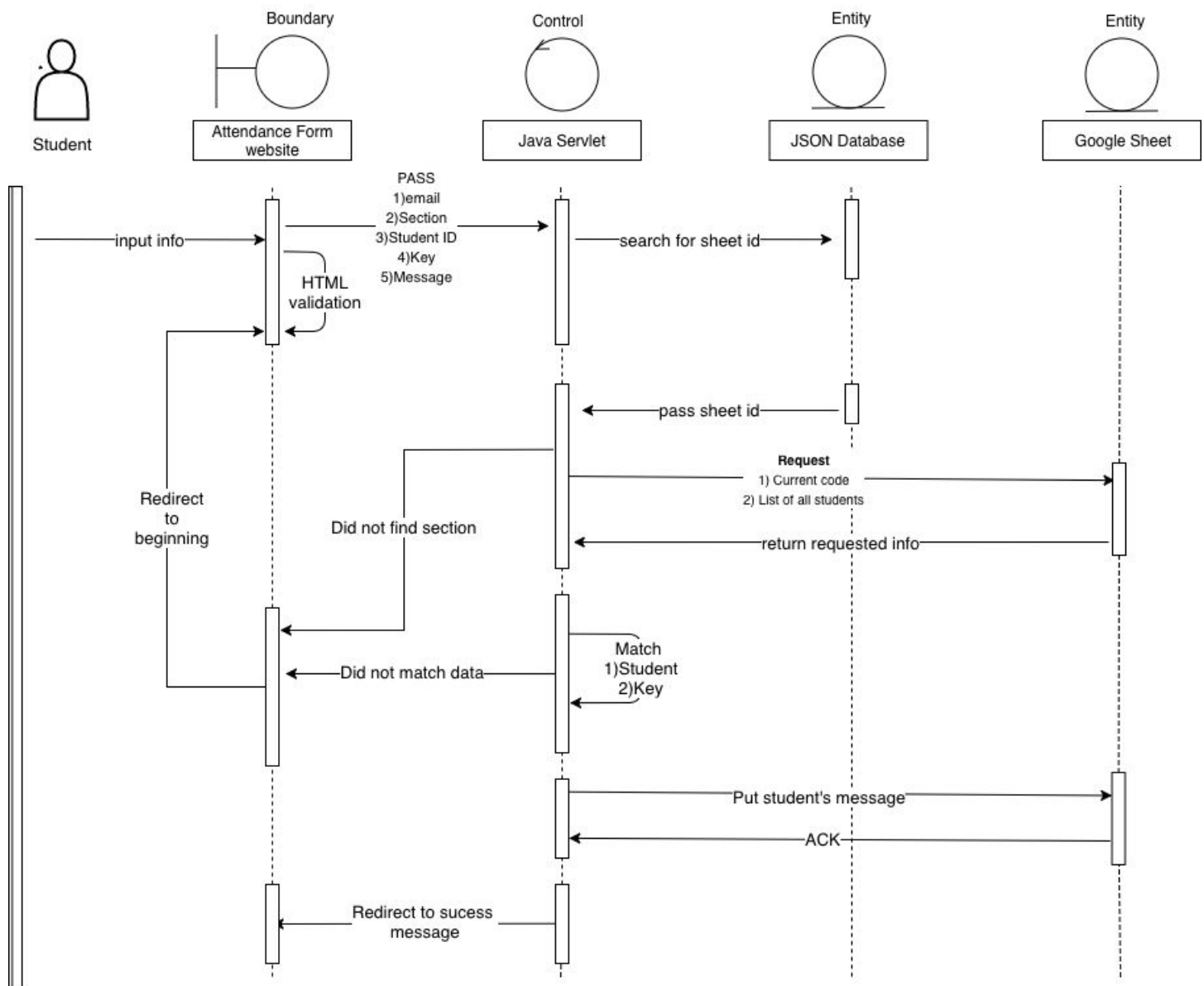


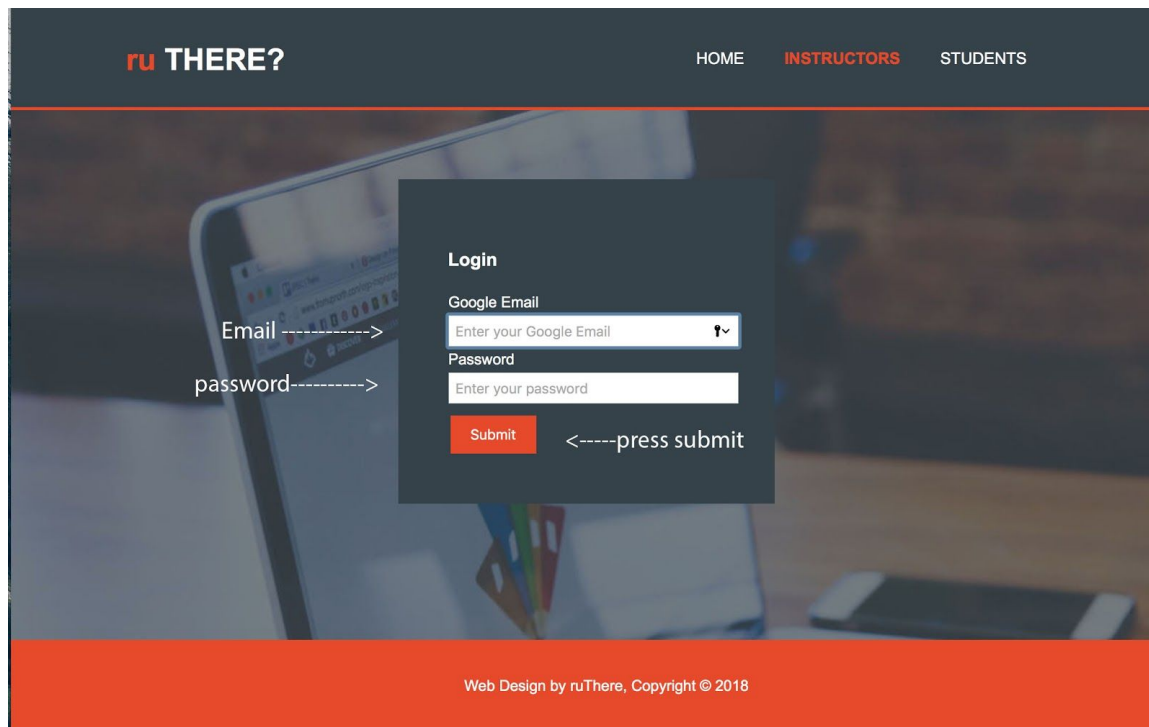
Figure 7.1: Instructor Sequence Diagram (For Grading Use-Case)



**Figure 8: Student Sequence Diagram**

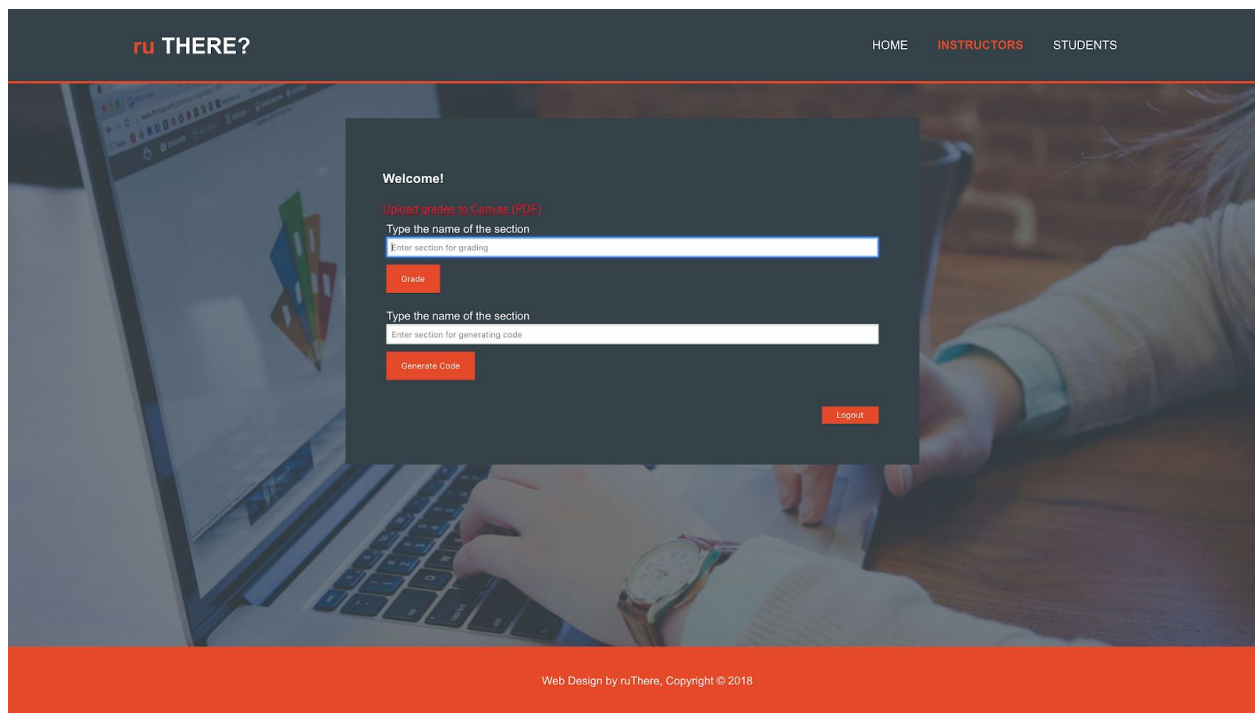
## VI. Instructor Prototype Screens

The User Interface design and the implementation are discussed below. The ruThere web application was implemented with the Google Sheet and Google Geolocation API.



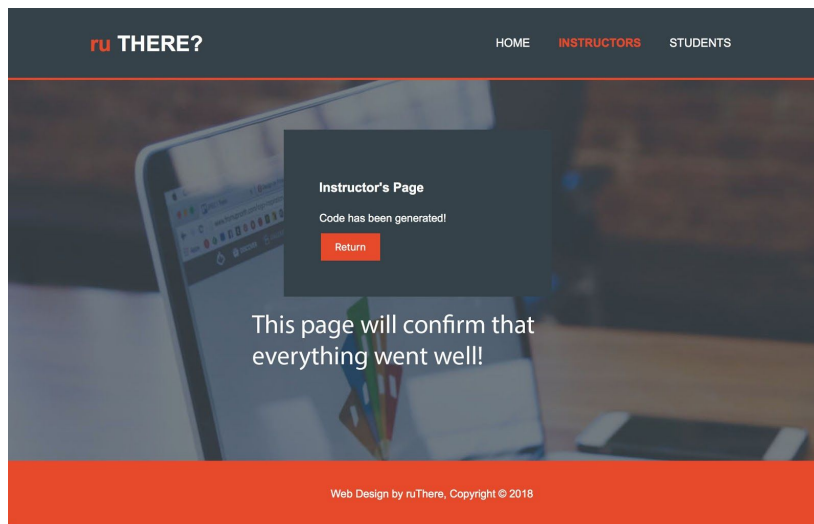
**Figure 9: Instructor Login Page**

This page allows the instructor to login to our web application using their 'ruThere' account.



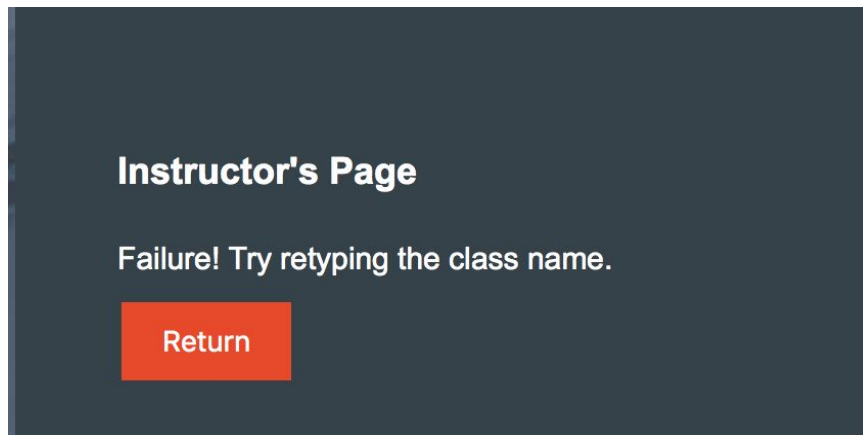
**Figure 10: Instructor Action Page**

This page allows the professor to type the section name and complete three actions. ***Generate*** a new code for a new date or ***grade*** all of the students or ***logout***.



**Figure 11: Instructor Successful Code Generation**

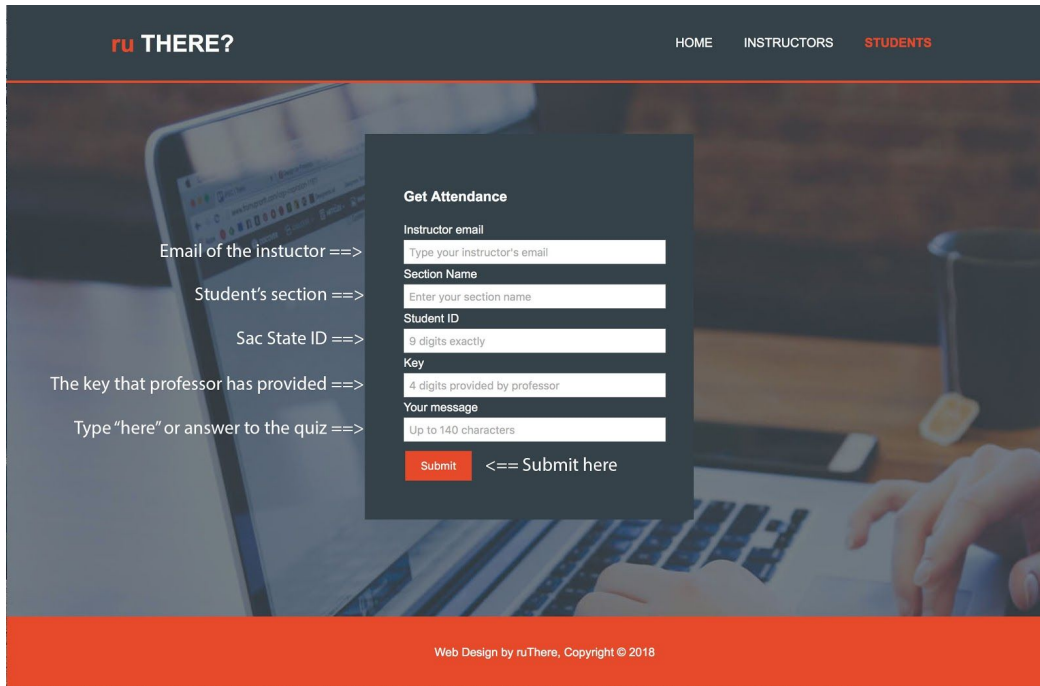
In case the professor is successful and such section exists, he or she will get a message that notifies them about the process completing.



**Figure 12: Instructor Failure Code Generation**

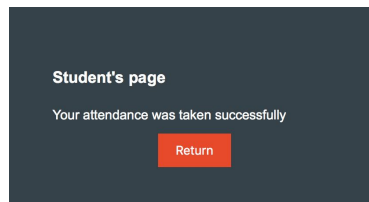
If anything were to go wrong, professor would receive such notification.

## VII. Student Prototype Screens



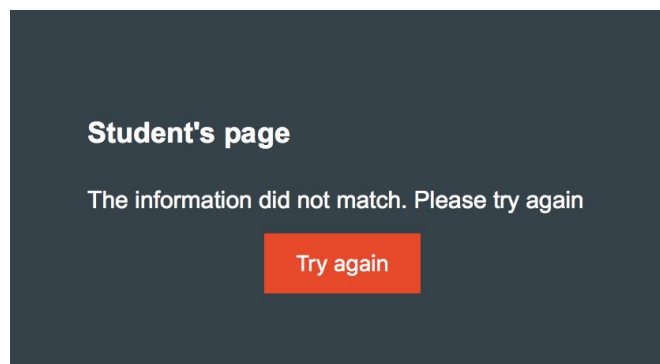
The image shows a web application interface for taking attendance. At the top, there is a dark blue header with the logo "ru THERE?" on the left and navigation links "HOME", "INSTRUCTORS", and "STUDENTS" on the right. The main content area has a background image of a laptop and a person's hands typing. Overlaid on this is a dark grey form titled "Get Attendance". The form contains several input fields: "Instructor email" (with placeholder text "Type your instructor's email"), "Section Name" (with placeholder text "Enter your section name"), "Student ID" (with placeholder text "9 digits exactly"), "Key" (with placeholder text "4 digits provided by professor"), and "Your message" (with placeholder text "Up to 140 characters"). Below these fields are two buttons: a red "Submit" button and a grey "<== Submit here" button. To the left of the form, there are five lines of text, each followed by "==>": "Email of the instructor ==>", "Student's section ==>", "Sac State ID ==>", "The key that professor has provided ==>", and "Type 'here' or answer to the quiz ==>". At the bottom of the page, there is a red footer bar with the text "Web Design by ruThere, Copyright © 2018".

Figure 13: Student Attendance Form



The image shows a dark grey rectangular box representing a success message. It has the title "Student's page" at the top. Below the title, the text "Your attendance was taken successfully" is displayed. At the bottom of the box is a red button with the text "Return".

Figure 14: Student Successful Attendance Submission



The image shows a dark grey rectangular box representing a failure message. It has the title "Student's page" at the top. Below the title, the text "The information did not match. Please try again" is displayed. At the bottom of the box is a red button with the text "Try again".

Figure 15: Student Failure Attendance Submission

## VIII. JSON Database

We used a JSON database to manually store the professor's login details for the web application and the id for their Google Sheet document. The password attributes that you see are not third-party account details from Google, but direct login account details for professor access to our web application.

```
1  {
2      "emails": {
3          "username@gmail.com": {
4              "password": "password",
5              "sheetId": "1VZ63I-Wm-pPDM-MHNODscw9treysG-9JLUyZyAC7rj0"
6          },
7          "username2@gmail.com": {
8              "password": "password",
9              "sheetId": "1m_KFoACldbDieFbcj0VQpqNjfWxunhe-cUsCLB1mCVU"
10         }
11     }
12 }
```

**Figure 16: Sample JSON Database**

## IX. Traceability Matrix

Below is our Traceability Matrix chart. This maps the relationship between the Functional/Non-Functional requirements with various use cases.

	UC_1: Generate_Key	UC_2: Grade_Class	UC_3: Student_Check _In	UC_4: Attendance_ Sheet	UC_5: Review_repo rt
FR_1: Manually take attendance			X		
FR_2 : Form of attendance			X		
FR_3: Provide unique key	X				
FR_4: Grade Calculator		X			
NFR_1: Attendance in class only	X		X		
NFR_2: Usability			X		
NFR_3: Documentatio n	X	X	X	X	X



## **X. Agile Methodology**

Our team decided on using the agile development process to implement the project idea. When the context of the project has a time constraint to under a month, we figured this development process is the best to go to deliver changes on a week by week basis.

Using the agile method, we will be able

### **Insight of Google API**

In this section, we will discuss our experiences and findings as we develop our student attendance web application.

#### **Geolocation API**

Google provides API javascripts to be integrate to our project. Geolocation API is one of the most important Google API we need to implement in order to prevent students to take attendance outside class. The link for this API is: <https://developers.google.com/maps/documentation/javascript/geolocation>. This webpage shows an example of how the google maps calculate the user's current location. However, the downside to this is that it is not very accurate for devices that do not have GPS capabilities.

#### **Spreadsheet API**

In addition to geolocation, spreadsheet is also an important part of our requirement because we use the spreadsheet for saving data of students ID, student names, their attendance grades, and key generated by the instructor. The link for this API is: <https://developers.google.com/sheets/api/>. In this webpage, they provide a quickstart guide for different programming language which is more detailed in by following this link: <https://developers.google.com/sheets/api/quickstart/android>.

## **XI. What we learned**

- Github Version Control
- HTML5, CSS, Javascript development
- Agile Development Model
- Reuse Development Model
  - Google Maps API integration
  - Google Sheets API
- Servlet and Apache Tomcat Servers
- Control flow chart
- White Box and Black Box testing
- UML Class Diagrams, ERD Diagrams, State Diagrams, Sequence Diagram, Use Case Diagrams
- Architectural models
- Cyclomatic Complexity
- Coupling and Cohesion
- Functional and Non-Functional requirements
- Code Coverage
- Association and Generalization
- Hosting Web Applications (With Azure)
- Requirements Process

## **XII. Planned Future Improvements**

- We plan on implementing the timeframe that does not allow the student to check in after the time had passed
- Drop-down table of all the sections given a professor's email
- The way to add a new class using the necessary information
- Get a receipt for a student and a receipt for a professor
- Make the geolocation fool-proof, intruder will be unable to spoof location
- Get a more accurate geolocation from desktop computers.