SDD: 402 Senior Project - USchedule

Owners: *Annie Flora, Amelia Jay, Liam Namba, Cristalina Nguyen, Sophia Prochnow, Christian Santander*

NOTE: Parts of this document have not changed, and are taken from 401 (SDD) Software Design Document

6.1. Introduction

This Software Design Description Document presents the architecture and detailed design for the software for the USchecule project. This Project is a Web Application which provides an application to generate work schedules for employees in an organization. We are specifically focusing on the LAC+USC Psychiatry Residency program as our organization and are working with Chief resident Dr. Kelly Jones.

The application will take the residents' preferences and then generate a call schedule by the constraints given to us by the administrator, Dr. Jones. The purpose is to make scheduling efficient and simple.

As part of design, we will continue to work on making an optimal algorithm that satisfies all of Dr. Jones' constraints. We will design more mockups to make a visually pleasing graphical user interface, and then implement these on the front end. We will continue to add to the database and work on connecting it with the front end. We will determine how we would like to launch our final product and deliver it to Dr. Jones.

6.1.1 System Objectives

The objective of this application is to provide Dr. Jones with a new webapp that can take in the residents' preferences and then generate a call schedule based on the constraints inputted by the user admin (her). The current scheduling systems that exist are expensive and performing this task by hand is tedious. This application will reduce time and cost for her as the schedule will be generated within seconds.

6.1.2 Hardware, Software, and Human Interfaces

6.1.2.1 Graphical User Interface - Node.js and React on AWS
- We are using Visual Studio Code version 1.38 and up as our code editor.

6.1.2.2 Server - Python/Flask

6.1.2.3 Algorithm - Python

6.1.2.4 Database - MySQL on AWS Relational Database Services
- We are using MySQL Workbench for our database.
- * Working with LMU Build as we ran out of AWS credits

6.1.2.5 Operating Systems - MacBook Pro/Air
- MacOS Mojave 10.14 and up

6.1.2.6 Design Mockups - Sketch

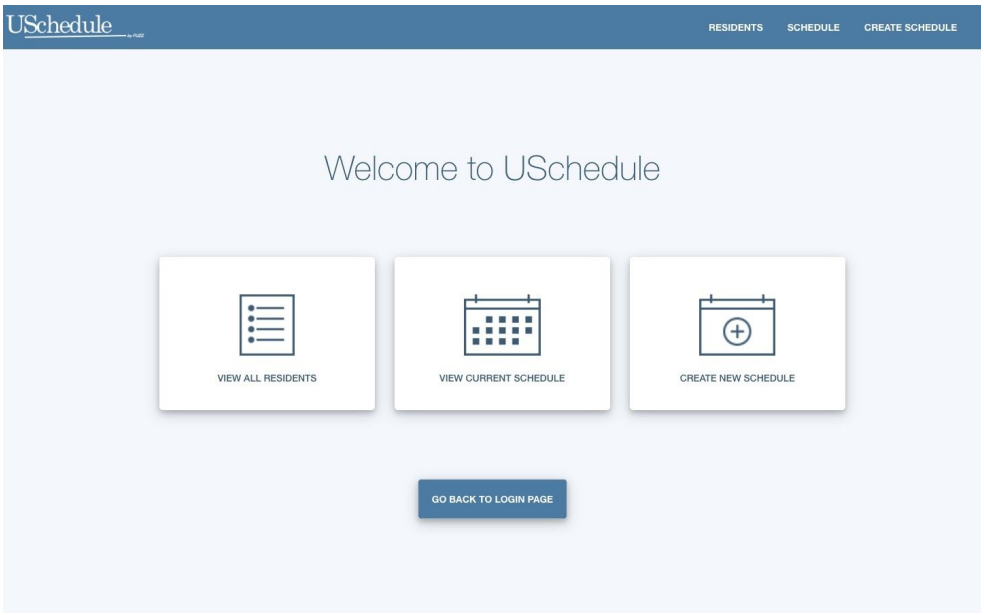
- We are using the application Sketch to make mock ups of our front end design.

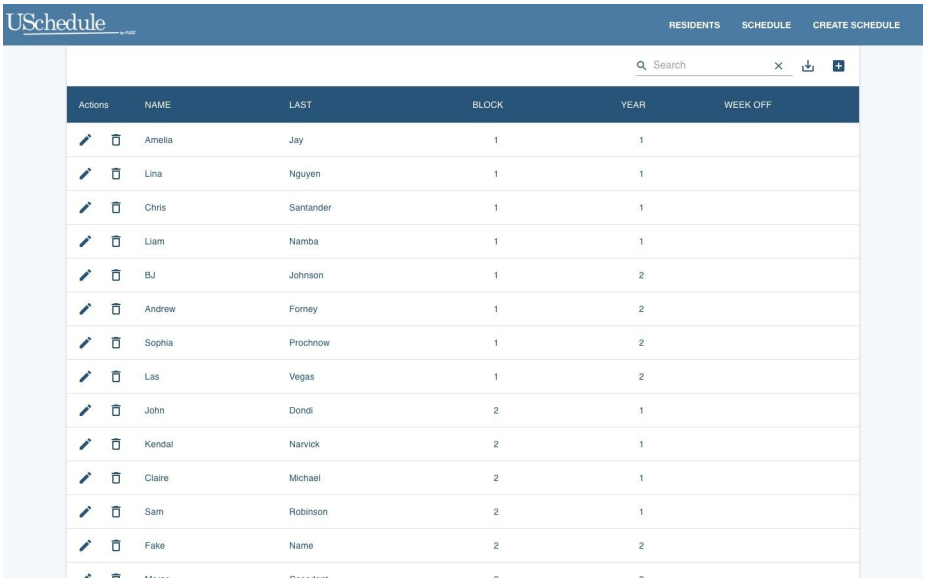
6.1.2.7 Other Resources

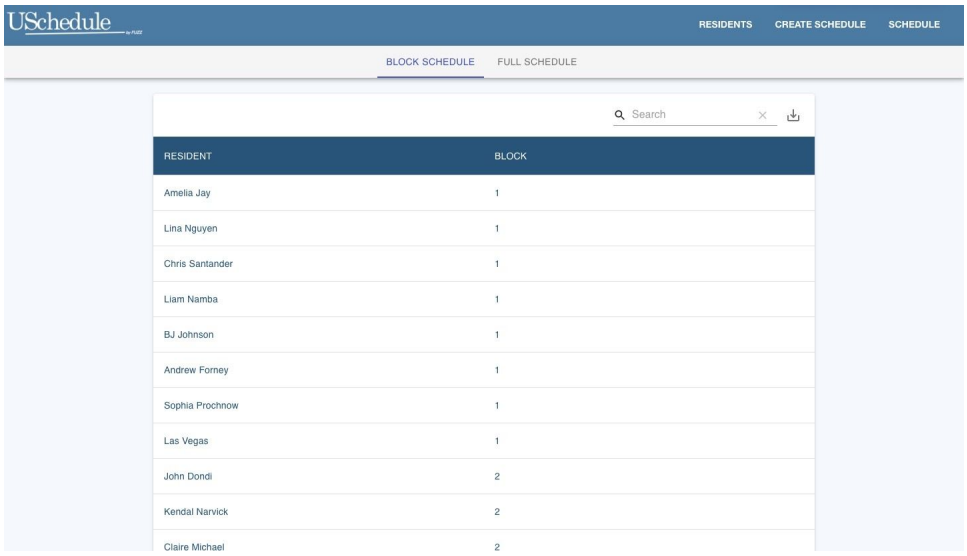
- We are using Trello to organize and prioritize tasks for the project.
- We have been using Google Drive as our platform for organizing and working on written documents.
- GitHub

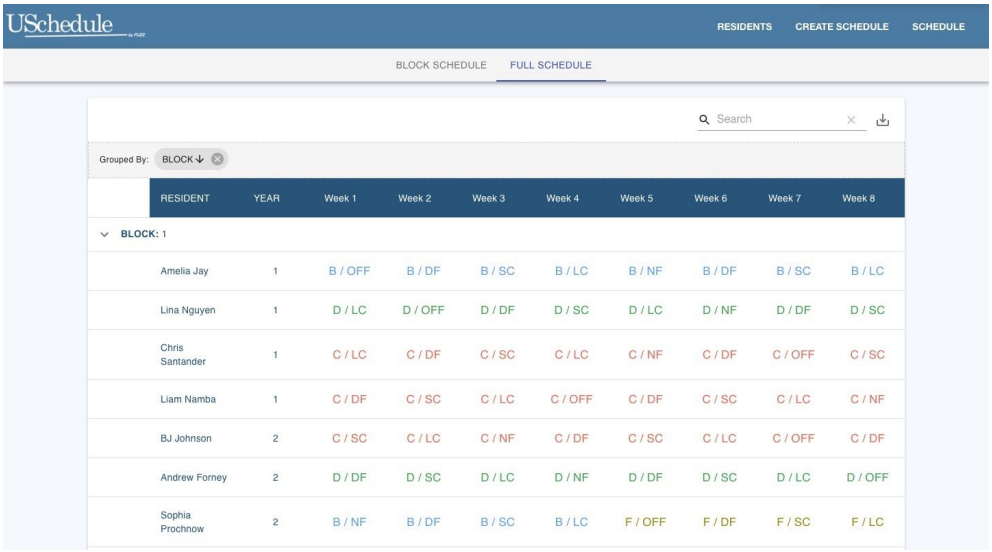
6.2 Architectural Design

With the design of our system, our users will be interacting with our front end web application. Our front end will be interacting directly with the Algorithm. The algorithm will be communicating with each of the needed resident, organization, and shift classes. Those classes will be communicating with the database to grab all data relating to that specific class.




UScheduleby FUZZ
RESIDENTS
SCHEDULE
CREATE SCHEDULE
Search
Actions
NAME
LAST
BLOCK
YEAR
WEEK OFF
Amelia Jay 1 1
Lina Nguyen 1 1
Chris Santander 1 1
Liam Namba 1 1
BJ Johnson 1 2
Andrew Forney 1 2
Sophia Prochnow 1 2
Las Vegas 1 2
John Dondi 2 1
Kendal Narvick 2 1
Claire Michael 2 1
Sam Robinson 2 1
Fake Name 2 2

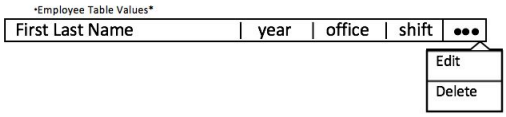

UScheduleby FUZZ
RESIDENTS
CREATE SCHEDULE
SCHEDULE
BLOCK SCHEDULE
FULL SCHEDULE
Search
RESIDENT
BLOCK
Amelia Jay 1
Lina Nguyen 1
Chris Santander 1
Liam Namba 1
BJ Johnson 1
Andrew Forney 1
Sophia Prochnow 1
Las Vegas 1
John Dondi 2
Kendal Narvick 2
Claire Michael 2


UScheduleby FUZZ
RESIDENTS
CREATE SCHEDULE
SCHEDULE
BLOCK SCHEDULE
FULL SCHEDULE
Search
Grouped By:
BLOCK
RESIDENT
YEAR
Week 1
Week 2
Week 3
Week 4
Week 5
Week 6
Week 7
Week 8
BLOCK: 1
Amelia Jay 1 B / OFF B / DF B / SC B / LC B / NF B / DF B / SC B / LC
Lina Nguyen 1 D / LC D / OFF D / DF D / SC D / LC D / NF D / DF D / SC
Chris Santander 1 C / LC C / DF C / SC C / LC C / NF C / DF C / OFF C / SC
Liam Namba 1 C / DF C / SC C / LC C / OFF C / DF C / SC C / LC C / NF
BJ Johnson 2 C / SC C / LC C / NF C / DF C / SC C / LC C / OFF C / DF
Andrew Forney 2 D / DF D / SC D / LC D / NF D / DF D / SC D / LC D / OFF
Sophia Prochnow 2 B / NF B / DF B / SC B / LC F / OFF F / DF F / SC F / LC

UI Layout

Log In			
Home Page	Scheduling Page	Resident Management Page	Output Page
♦ Schedule ⊕ Create Button ⊕ View Residents ⊕ Log Out	♦ List of residents ⊕ Options for each resident ⊕ Float preference drop down ⊕ Vacation preference	⊕ Add new resident □ Employee table values ⊕ Remove resident ♦ Confirm pop up ♦ Edit screen ⊕ Filter ⊕ Generate schedule ♦ Error page	♦ Schedule generates ⊕ Download button

Key
♦ UI Element
⊕ Button
□ Field



6.2.1 Major Software Components

6.2.1.1 The GUI displays a Home Page

- The home page will allow the user to visit the current scheduling, create schedule, and resident page.

6.2.1.2 The GUI displays a Create New Schedule page

- The scheduling page will allow the user to generate a block or full schedule after inputting the start date.

6.2.1.3 The GUI displays a Resident Management Page

- The resident page will communicate and add/edit/pull/delete data from the resident database.

6.2.1.4 The GUI displays an Current Schedule page

- This page will communicate with the database to grab the current stored schedule which will be stored as a JSON object.

6.2.2 Major Software Interactions

6.2.2.1 Home Page

- The home page current schedule button will grab the JSON object for the schedule from the database. The object will include weeks, shifts, employees, block, and ward.

6.2.2.2 Create New Schedule Page

- The scheduling page will grab the resident data from the database and will take in constraints for each resident which will be used as

constraints in the algorithm which then generates the schedule either block or full.

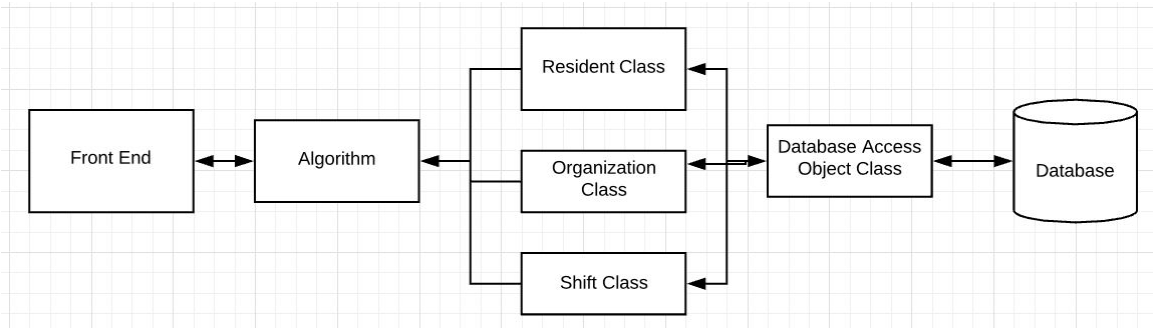
6.2.2.3 Resident Management Page

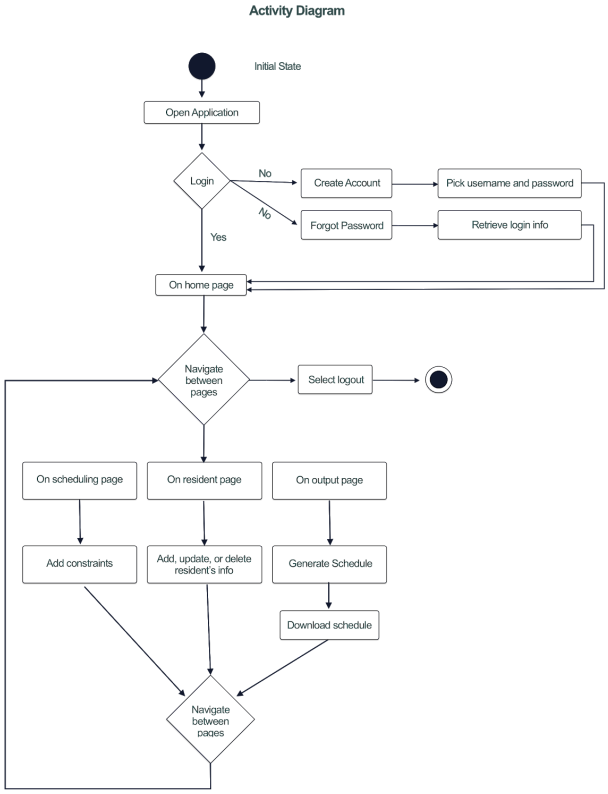
- The resident management page will send resident data to be stored in the database. It will also edit or delete current resident data from the database.

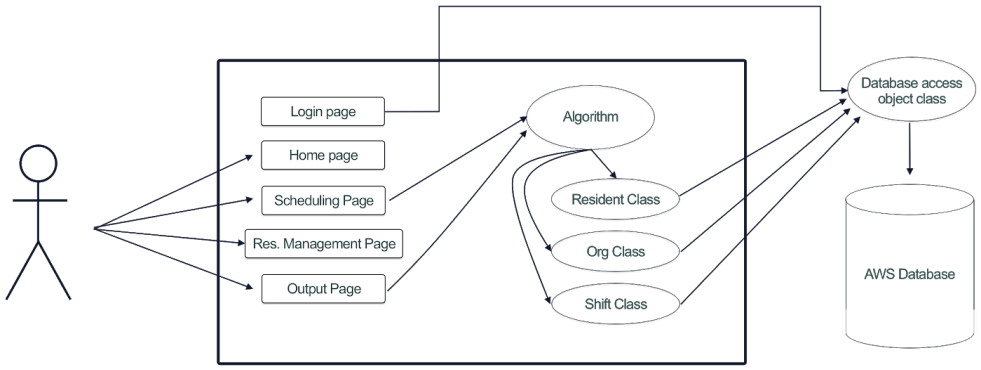
6.2.2.4 Current Schedule Page

- This page will grab the created JSON object for the schedule which was generated from the algorithm.

6.2.3 Architectural Design Diagrams







6.3 CSC and CSU Descriptions

Our application consists of our Graphical User Interface (GUI), Server, and Database. The GUI is being created using React and Node.js. This GUI consists of a Login page, Home page, Create schedule page, Resident page, and a View Current Schedule page. All pages will be connected and easy to navigate between. The Server will get, update, and add data to our database. The server shall also run the algorithm to determine the optimized schedule using the constraints inputted by the user (week off). The server will be using Python and LMU Build. We will have classes for algorithm, employee, shift, and input object class. We will also have a database access object class which will communicate with the database. The Database will be using MySQL on AWS RDS. The database will consist of three tables: Employee table, block schedule table, and Shift schedule table. The Employee table will include columns: Employee ID, Employee first name, Employee last name, Employee Year, week off, block, ward, and Shift . The block schedule table will consist of columns: Employee, year, block. The Shift schedule table will include columns: Shift Type, block, week off, ward.

6.3.1 Class Descriptions

The following sections provide the details of all the classes used in the USchedule application.
The classes are located in Scheduler.py

6.3.1.1 Classes for the Front End GUI

Classes for each page of our front end such as Welcome (home) page, login page, create new schedule page, resident page, header component, and current schedule page. These are all React components.

6.3.1.2 Employee Class

Initializer method, fields: ID, first name, last name, week off, and year.

6.3.1.3 Shift Class

This class has an initializer method and fields: ID, type, ward, block.

6.3.1.4 Input object class

This class has an initializer method and fields: weeks, shift types, employees, and locations.

6.3.1.5 Database Access Object Class

We are using MySQL Connecter to connect to our AWS RDS. Through this we will call queries onto the DB such as: select_all (self, table), selecting a single item from a table, and queries to create and update residents.

6.3.2 Detailed Interface Descriptions

6.3.2.1 GUI

The GUI will be receiving data from the user. This data will be the inputted constraint week off, the date for the schedule, and resident data. This data will be sent to the database to be stored, updated, or removed through the DAO Class. The front end will also be communicating with the Employee Class, and Shift Class.

6.3.2.2 Database

The database will be communicating with the DAO to transfer information. The information in the database will be from the users information inputted in the front end.

6.3.3 Detailed Data Structure Descriptions

6.3.3.1 Algorithm

In the scheduling algorithm, we are using a dictionary with key value pairs to represent whether or not an employee is working that shift. Each constraint will be added using the add method.

6.3.3.2 CP model - schedule

This is what we add constraints to. There is a method called add which we use to add these constraints - ex: model.add(constraint)

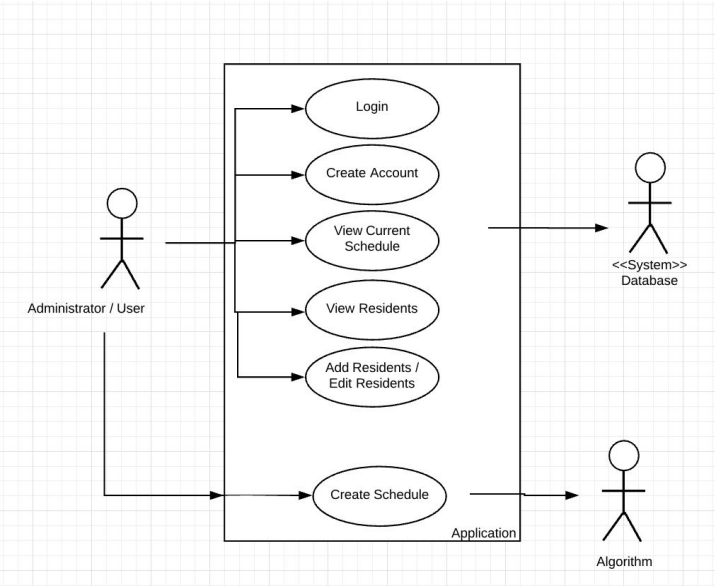
6.3.3.3 The Database Access Object

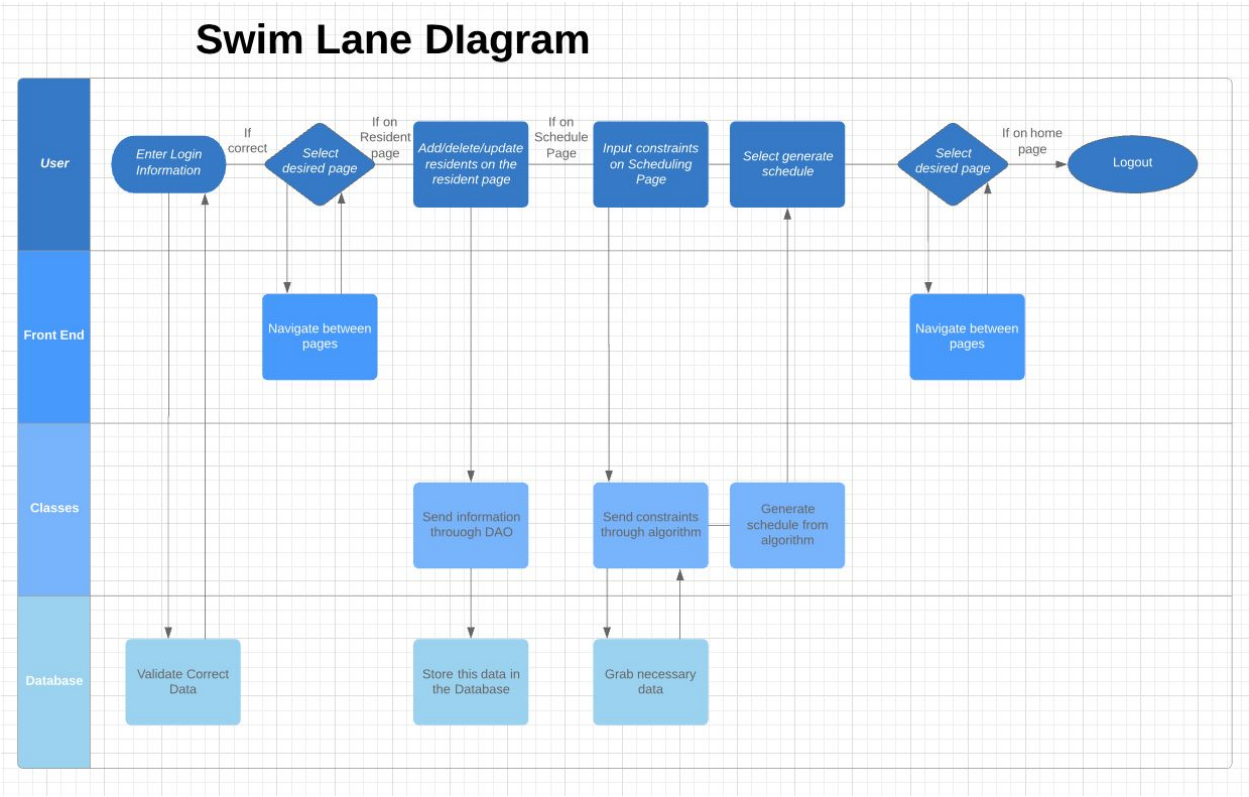
In the DAO we are using a list and dictionaries. In this, we are calling queries. When it is a select all, each result will be placed in a list. When it’s selecting a single, the query result will be placed in a dictionary.

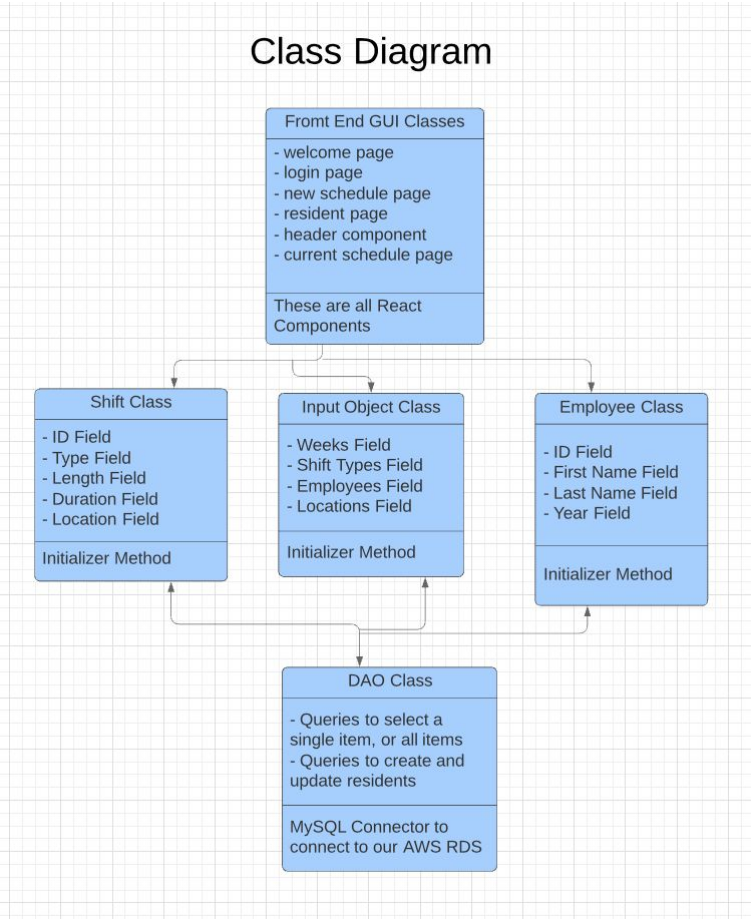
6.3.4 Detailed Design Diagrams

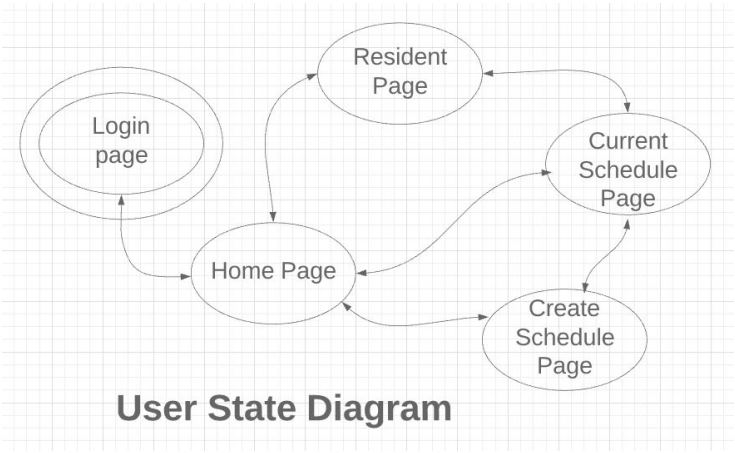
*See Activity Diagram from section 6.2.3 Architectural Design Diagrams

Use Case Diagram







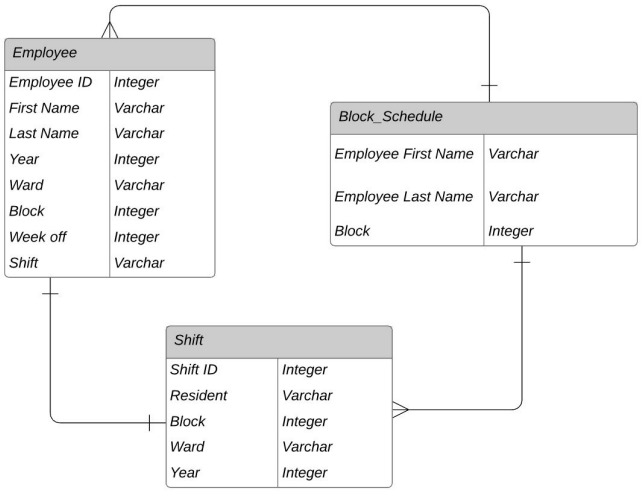


6.4 Database Design and Description

The database for our application will be storing employee data as well as schedule data for block and shift schedules. We will be using MySQL and AWS RDS to work with this data. This database will include three tables: the employee table, block schedule table, and the shift schedule table. The employee table will be storing data about each

employee within the organization. It will include the columns: Employee ID, Employee first name, Employee last name, Employee Year, week off, block, ward, and shift. The primary key in this table will be the employee ID since no two employees will have the same ID. The Block schedule table will be storing data about a given block schedule and consists of the columns: employee, and block. The Shift schedule table will include information about a given work week shift and will have columns: Shift ID, Shift Type, employee, year, ward, and block. The primary key in this table will be the shift ID.

6.4.1 Database Design ER Diagram



6.4.2 Database Access

The database will be accessed using our Database Access Object Class. With this, we are using MySQL Connector to connect our AWS RDS. Through this class, we will be calling queries onto the database. These queries will be selecting data from tables, and creating/updating resident data. This DAO will be communicating with the database as well as the employee, and shift classes to transfer needed data back and forth.

6.4.3 Database Security

We are using AWS RDS for our database. Amazon RDS provides us with a set of features which ensure that our data is securely stored and accessed. We are using the provided security features to control who can log into our database. These AWS security tools that we are using will only allow access from team members' IP address, Dr. Forney’s IP Address, and Dr. Jones’ IP Address.