SDD: 401 Senior Project - USchedule

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.

The application will take the residents' preferences and then generate a call schedule by the constraints inputted by the administrator. The application will also be generalized to be usable by other industries such as restaurants, retail, etc. The purpose is to make scheduling efficient, simple and scalable.

6.1.1 System Objectives

The objective of this application is to provide administrators with a new webapp that can take in the residents' preferences and then generate a call schedule based off the constraints inputted by the user admin. The current scheduling systems that exist are expensive and performing this task by hand is tedious. This application will reduce time and cost for users 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 on AWS Lambda

6.1.2.3 Algorithm - We are using Google OR Tools for help with our algorithm

6.1.2.4 Database - MySQL on AWS Relational Database Services
We are using MySQL Workbench for our database.

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

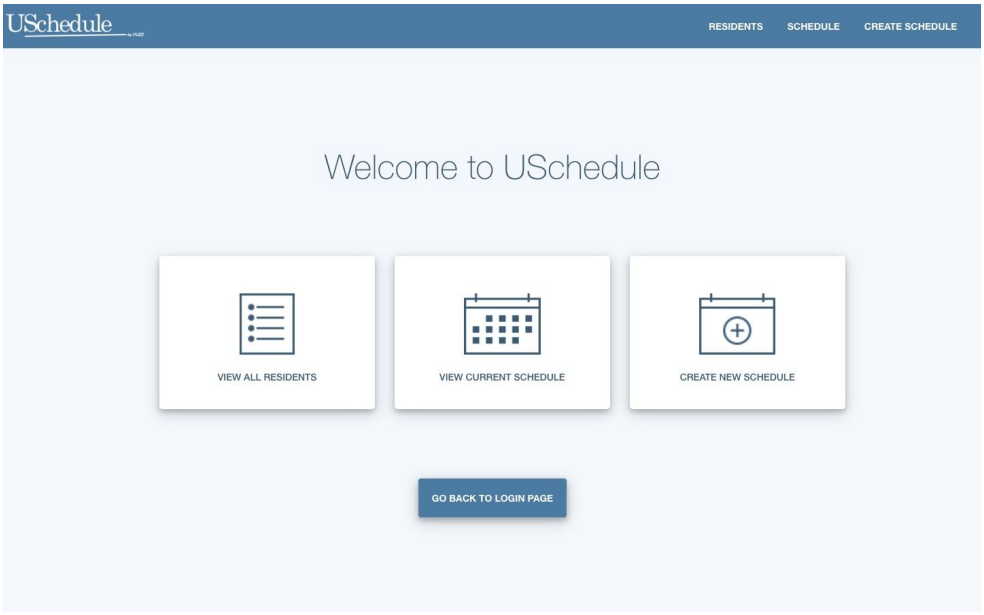
6.1.2.6 Design Mockups - Sketch
We are using the application Sketch to make mock ups of our front end design.

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 as our platform for organizing and working on written documents.

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.



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

Employee Table Values

First Last Name	year	office	shift	•••

Edit
Delete

6.2.1 Major Software Components

6.2.1.1 The GUI displays a Login Page

- The login page will communicate with the user login data in the database to verify correct username and password information has been entered. If forgot password button is selected, it will

communicate with the database to grab email information for that user. It will also work to store new user data in the database when creating a new account. The login button will direct the user to the home page.

6.2.1.2 The GUI displays a Home Page

- The home page will allow the user to visit the scheduling and resident page. It will also have a button allowing the user to view the current schedule. When selected, this will communicate with the database to retrieve the schedule which will be stored as a JSON object.

6.2.1.3 The GUI displays a Scheduling Page

- The scheduling page will take in the inputted schedule constraints and store them until the schedule has been generated.

6.2.1.4 The GUI displays a Resident Management Page

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

6.2.1.5 The GUI displays an Output Page

- The output page will communicate with the algorithm which communicates with the related classes which communicate with the database. This algorithm will generate the schedule.

6.2.2 Major Software Interactions

6.2.2.1 Login Page

- The login page will send username and password data to the database. The database will communicate whether or not this data is valid.

6.2.2.2 Home Page

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

6.2.2.3 Scheduling Page

- The scheduling page will grab the resident data from the database to display all current residents to be scheduled. The scheduling page will take in constraints for each resident which will be used as constraints in the algorithm which generates the schedule.

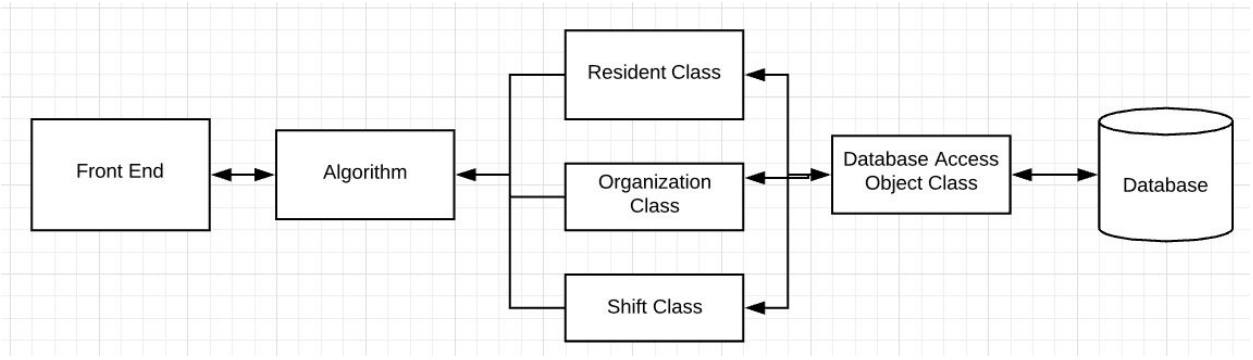
6.2.2.4 Resident Management Page

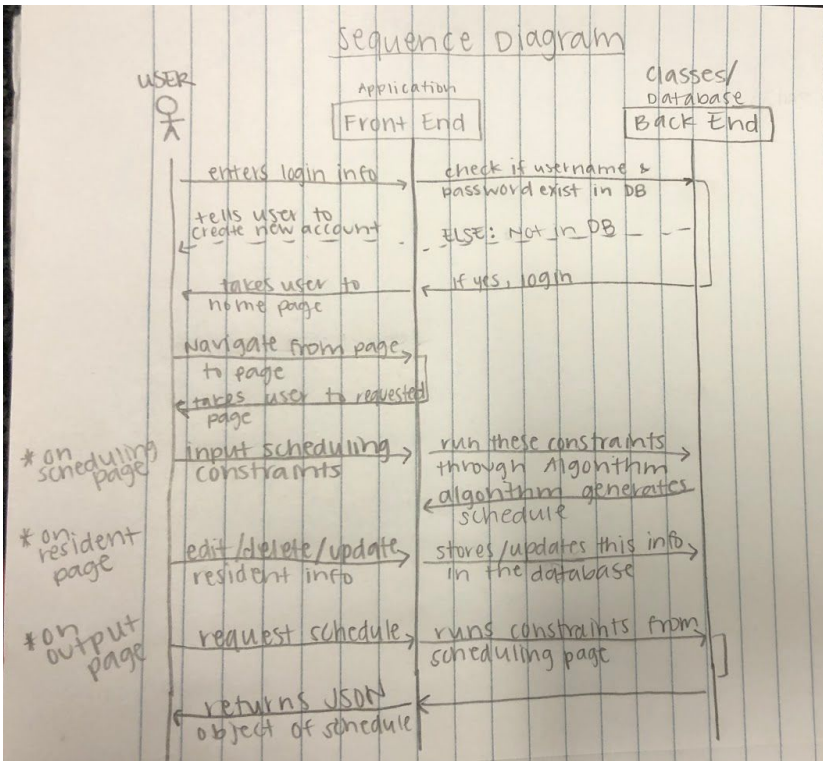
- The resident management page will send resident data to be stored in the database. It will also edit or delete current resident data from the database.

6.2.2.5 Output Page

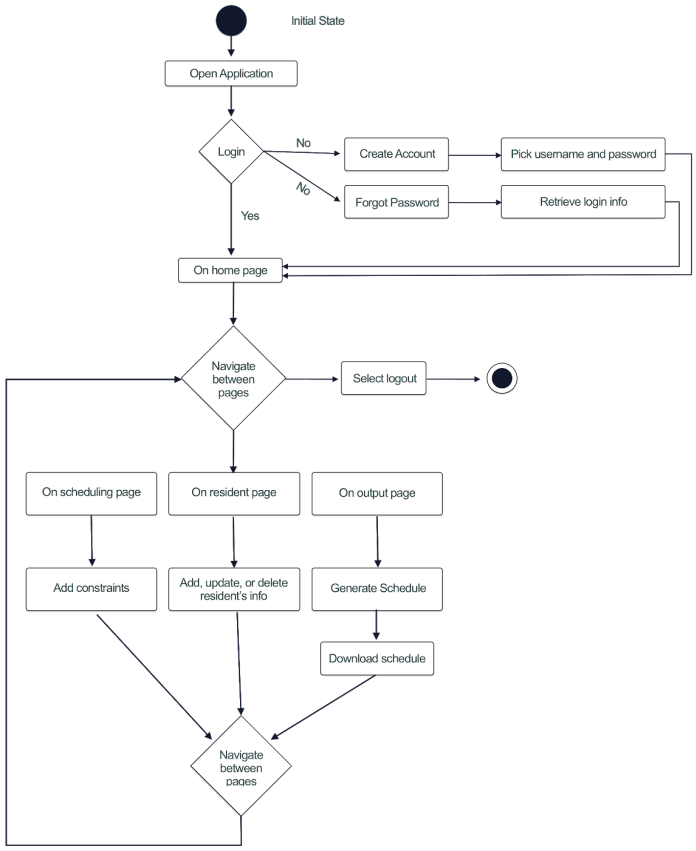
- The output page will grab the created JSON object for the schedule which was generated from the algorithm.

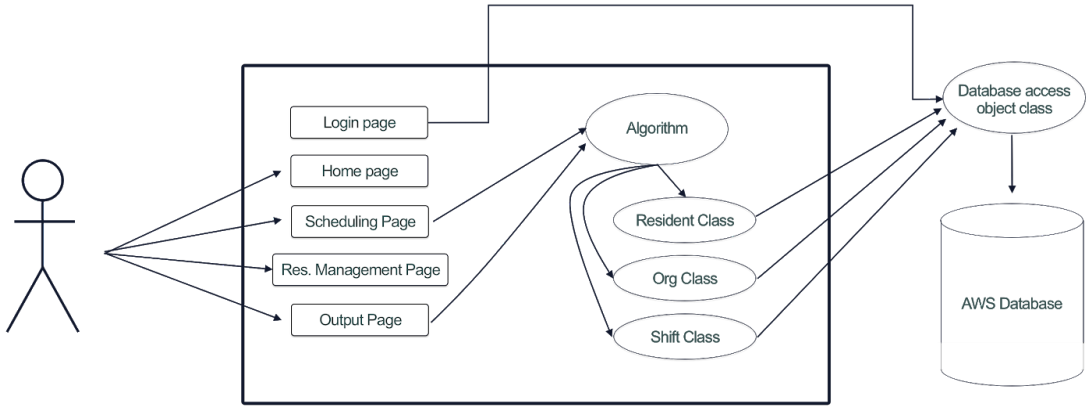
6.2.3 Architectural Design Diagrams





Activity Diagram





6.3 CSC and CSU Descriptions

Our application consists of our Graphical User Interface (GUI), our Server, and our 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. The server will be using Python on AWS Lambda. 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, Offices table, and Shifts table. The Employee table will include columns: Employee ID, Employee first name, Employee last name, Employee Year, Office ID, and Shift ID. The Offices table will consist of columns: Office ID, Office Name, Office Address, Office City, and Office State. The Shifts table will include columns: Shift ID, Shift Type, Shift Length, and Shift Duration.

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, and year

6.3.1.3 Shift Class

This class has an initializer method and fields: ID, type, length, duration, and location.

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 username and password data, as well as the inputted constraints, 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 in OR tools

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

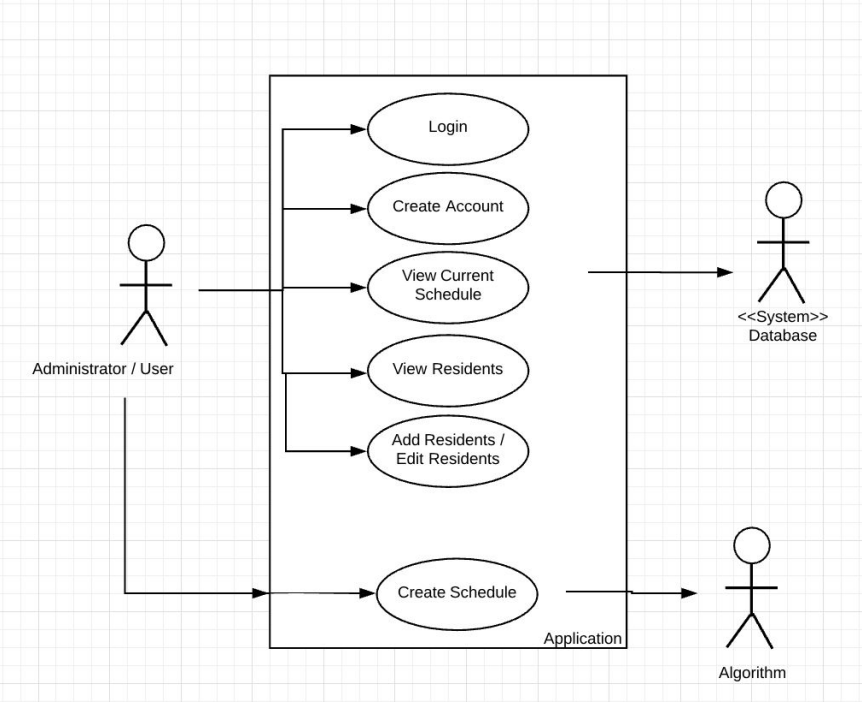
In the DAO we are using a list and dictionaries. In this, we are calling queries. When it is a select all, each result will be placed in a list. When it's selecting a single, the query result will be placed in a dictionary.

6.3.4 Detailed Design Diagrams

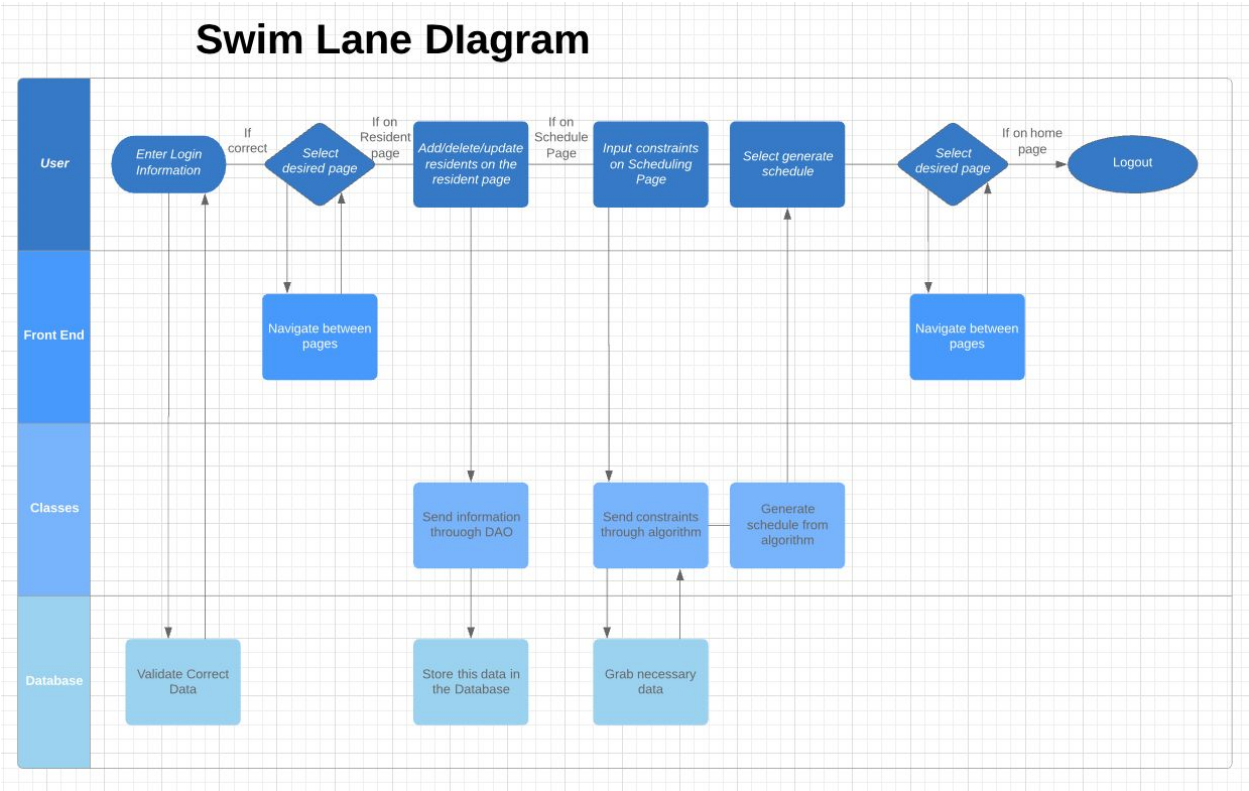
*See Activity Diagram from section 6.2.3 Architectural Design Diagrams

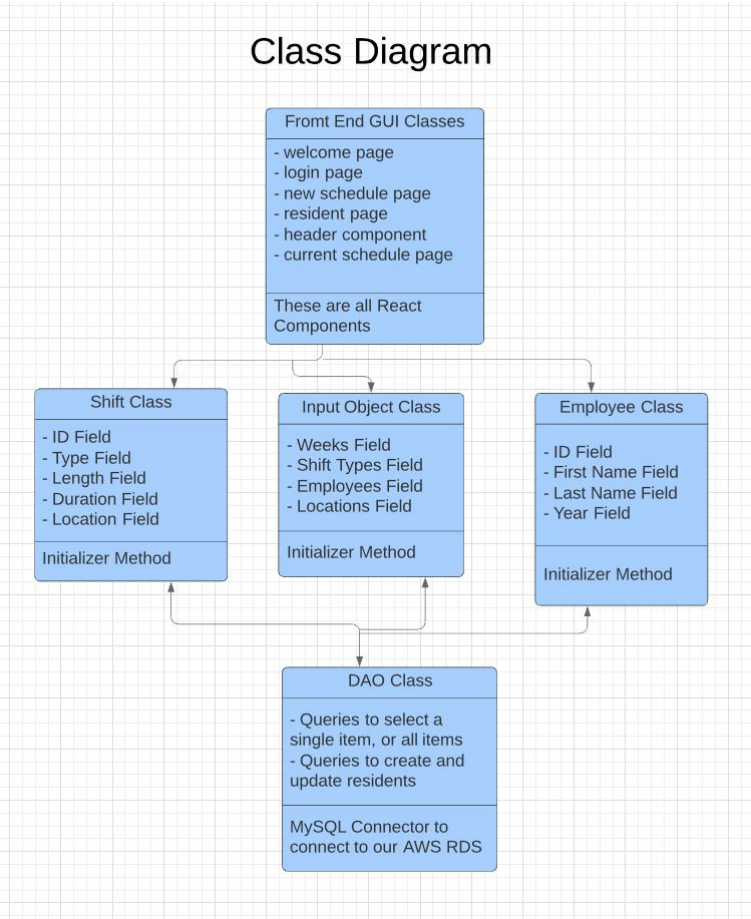
*See Sequence Diagram from section 6.2.3 Architectural Design Diagrams

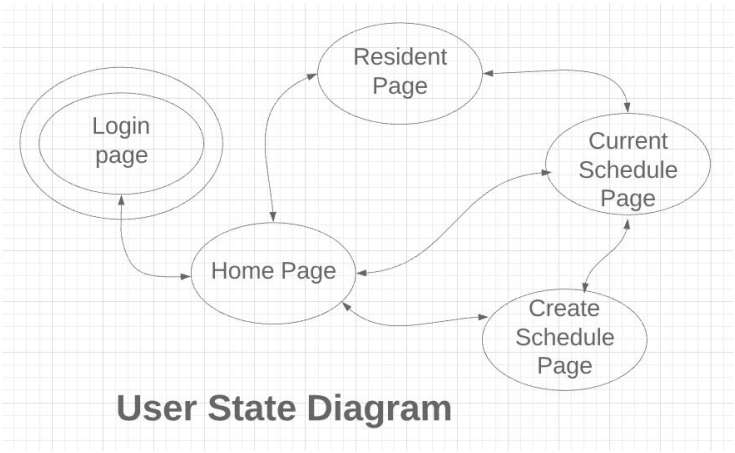
Use Case Diagram



Swim Lane Diagram





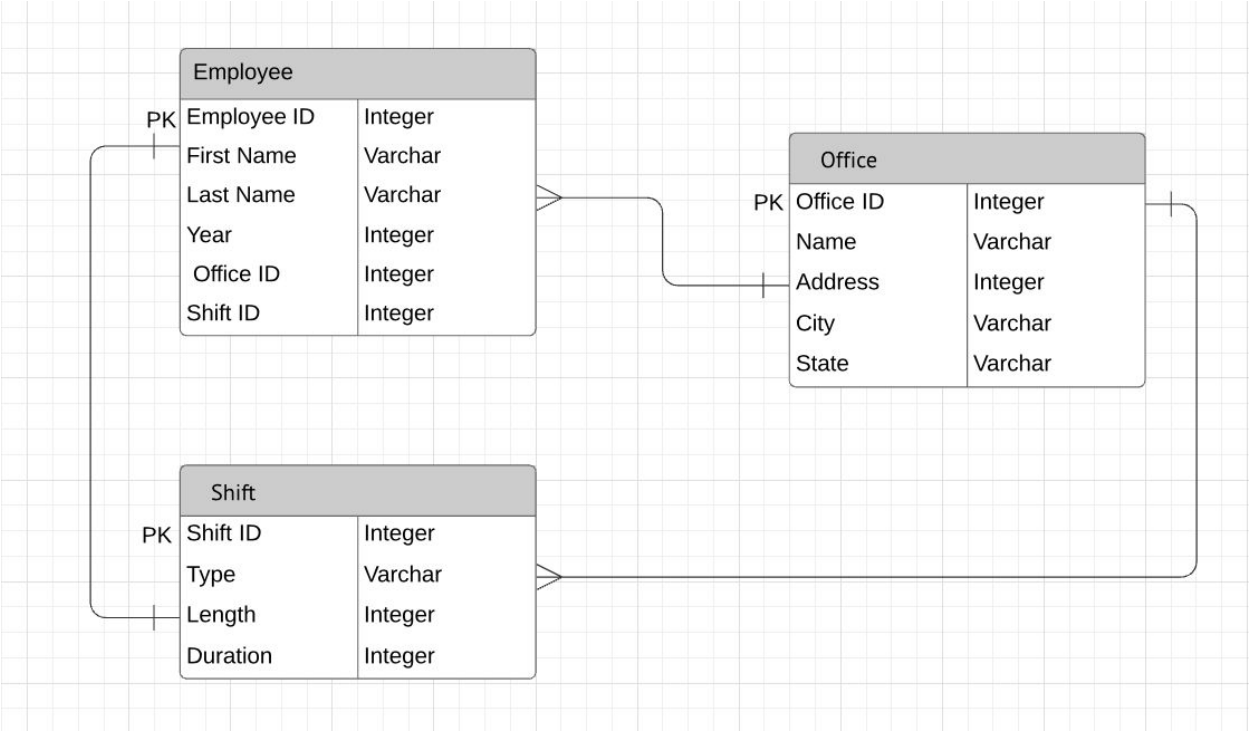


6.4 Database Design and Description

The database for our application will be storing employee data as well as organization or office data, and data about a given shift. We will be using MySQL and AWS RDS to work with this data. This database will include three tables: the employee table, the office table, and the shift 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, Office ID, and Shift ID. The primary key in this table will be the employee ID since no two employees will have the same ID. The Offices table will be storing data about a given office and consists of the columns: Office ID, Office Name, Office Address, Office City, and Office State. In this table, Office ID will be the primary key. The Shifts table will include information about a given work shift and will have columns: Shift ID, Shift Type, Shift Length, and Shift Duration. 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.