Designing an Employee Management System Using tools provided by MongoDB and Java

SUNY POLYTECHNIC INSTITUTE

Author: Delanovic, Adis
Instructor: Spetka, Scott
Project Report for CS498 Capstone Project
Date: 05/28/2021

Table of Contents

List of Figures

List of Figures cont.

List of Abbreviations

Acknowledgements

I would like to acknowledge my CS370 team this semester. While working on a 3D Model Viewer project I learned a lot that assisted me in the completion of this project. One of my group members, Maverick Coryell, assisted me greatly in how FXML and FXML Loaders are used properly with JavaFX. FX Markup Language is an extension of the "Extensible Markup language" (XML) created by Oracle. It is a method of defining rules for encoding documents or formatting pages. This helped me efficiently design a UI that is seamless in switching back and forth between different views. I would also like to thank Professor Spetka for allowing me the opportunity to create something that I would not even know where to start a couple years ago.

Preface

In 2018, I began a new position working for the Air Force Research Laboratory as a Civilian Police Officer in Rome, NY. At this time, I was not yet enrolled as a Computer Science Student at SUNY Poly. One of the first things I noticed that the Police Department there relies heavily on a lot of their things being done on hard paper. This includes scheduling, requesting time off, and training. To make matters worse most of this stuff was also decentralized. One would need to go to several different places just to find the information for scheduling or sign training documents. This is inefficient and can lead to confusion as to where to find necessary information.

This is not just the department, but the entirety of the U.S Air Force Law Enforcement that probably does not have an adequate system set up specifically to cater to Police Officer and Security Forces. There are approximately 40 thousand law enforcement officers at the Federal level amongst different agencies. There exists a large market to improve the efficiency of all these agencies. Creating a centralized system where each agency can select and use the many features as necessary would be the ideal end scenario.

Around this time, I created a presentation as to how the department can be improved and not only presented it to my Chief and Deputy Chief but also another Computer Scientist that had the connections and skills necessary to develop the program. I called the solution an "Air Force Civilian Police Portal". Figure 1 is from the slides taken from that presentation. It shows my initial assessment and ideas for the program.

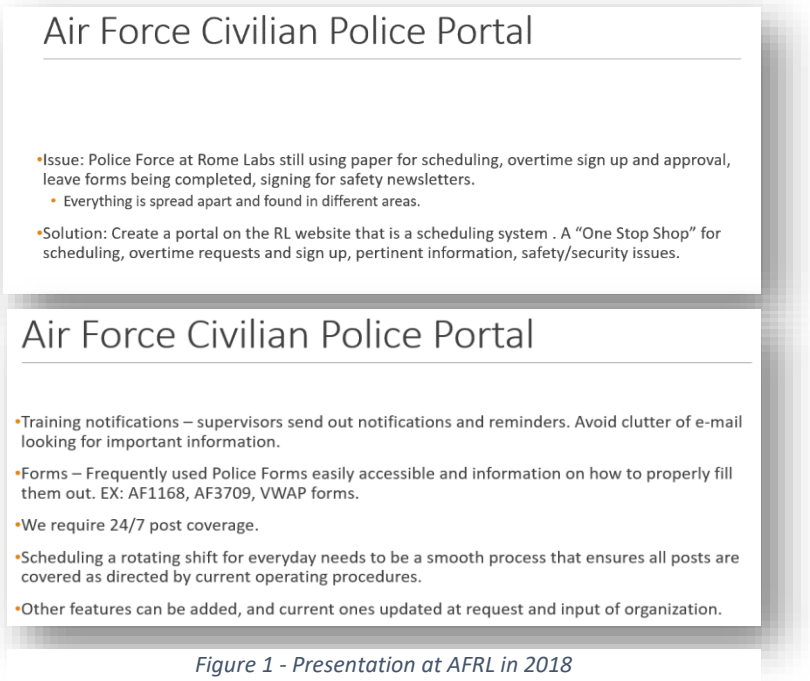

Figure 1 - Presentation at AFRL in 2018

While my presentation was well received and agreed upon—there was still no concrete plan as to how to move forward regarding making this system. At the time of my presentation, I was in no position to create even a single line of code that added two values let alone an employee management system. Soon after this I enrolled into SUNY Polytechnic Institute and now a few years after not only can I write code that can add two values—but I believe I have gained technical skills to create an Employee Management System. This system would specifically cater to Air Force Security Forces and Civilian Police.

The idea I present here stemmed from trying to solve a real-world problem and I am happy to have the opportunity to work on something I could not even come close to creating a few years ago. I also did not revisit this project until I started thinking about what to do for my Capstone during Fall 2020 Semester. Although I do think there is not adequate time in one semester to complete this in its entirety I do hope to build a good base where I will be able to continue working on this system long after the class ends. I also hope to be able to present this to my peers once again and perhaps move forward to a fully deployable and working system.

Project Proposal and Purpose

In January 2021 I spoke to a potential Employer for a job as a Software Engineer upon graduation. The main language used by this employer is Java, specifically Java for Android Development. This is one of the biggest reasons I decided to use Java for my Capstone Project. One of the goals of this project is to be even more proficient in the Java language. This is primarily that if the employer does keep going with the tentative job offer I would be even more prepared.

Another objective is to go through the entire planning process of developing a new program. This means creating flow charts, class diagrams, methods/functions, and anything else just to make the coding part of the project much easier and fluid. This part also includes doing research by reading the documentation directly and not refer to tutorials. By improving my documentation reading abilities as well as my pre-planning abilities I think it can help me greatly in the future.

The crux of the Capstone will include using MongoDB and Java. I intend to develop a user interface that allows for all CRUD (create, read, update and delete) operations on objects stored in the database. MongoDB is a no-SQL database where all objects are primarily stored as Json. Json stands for JavaScript Object Notation and allows for storing of data. The objects are organized and represented in a hierarchy allowing for much greater readability. My editor of choice for Java is IntelliJ Idea. Initially I was going to use PostgreSQL. After some research, it was apparent that for what I was trying to accomplish it was much better using MongoDB than PostgreSQL. Unline MongoDB—PostgreSQL uses the Structured Query Language (SQL) as a means of storing or accessing data. The MongoDB Java driver also appears superior to the SQL Java driver. At its most basic the project an employee Management System. However, there are several tasks that I wish to accomplish during this project.

- Learn MongoDB/Java and how they can be used together to create and manage a database.
- Proper planning and execution.

- The reading of documentation instead of tutorials.
- User level Permissions. A regular employee would not have the same access as a supervisor.
- Authentication to login to the application
- A proper GUI that allows for the ease of use of the application.
- Properly update values with UI (User Interface) using CRUD (Create, Read, Update and Delete)
- Ensure all updates are atomic.
- Possible concurrency issues with multiple programs running.

As mentioned in the preface my inspiration for this project is that my workplace still uses paper for scheduling, requesting time off, etc. By completing this project, I hope to improve my skills as a Java programmer, learn MongoDB and proper use of non-SQL databases, and create something that a business could possibly even utilize.

Part I: Development Environment

The system was designed on the following Computer Specifications

Processor: Intel® Core™ i7-7700 CPU @ 3.60Ghz

RAM: 16GB

Operating System: Windows 10 Home Edition

Graphics Card: MSI Radeon RX 580

The three main tools required for me to be able to get through the project with some success is being able to understand the Java MongoDB API/Driver, MongoDB Atlas and MongoDB Realm. There are many drawbacks with the MongoDB Java Driver and the free version of MongoDB places various limitations on what methods can be used. The editor used for this project will be Idea IntelliJ and all code is commented using Javadocs.

The Java MongoDB Driver provides both synchronous and asynchronous interaction with MongoDB. The full driver can be found at https://mongodb.github.io/mongo-java-driver/4.2/driver/ for version 4.2. It is also the latest version as of this report. The best part about this driver is the extensive documentation that can be easily followed. The main use of this driver will be to connect to the database and query results as well as append additional items to the collection.

MongoDB Atlas is a hosting service that provides a developer the ability to host the database on the cloud. It uses a multi-cloud data distribution system that distributes data across over 75 cloud regions using Amazon Web Services, Microsoft Azure and Google cloud. This means that I will not be using MongoDB on my local device during the creation of this project. The database will provide permanent storage on the cloud and persistence with the added security features provided by MongoDB.

MongoDB Realm has many features. It is primarily used for Android and iOS development. It also has an extensive software development kit (SDK) that caters to Node.js and Web developers. There are also several

limitations with using MongoDB Realm for developing software using Java as the front end. Earlier on in the project Professor Spetka recommended the MERN development stack. The MERN stack consists of MongoDB, Express, React and Node.js as the development tools used. I have some experience using this stack and after research, I do agree that it would have been the ideal method of creating this application. One of my main objectives of this Capstone was to further my knowledge of Java. This was my main reason for not going with the MERN stack.

MongoDB Realm will be used to configure the collections that are hosted within MongoDB Atlas. One of the main objectives of this Capstone was to learn about user permissions and how to successfully utilize them to create an application. MongoDB Realm makes it possible to set these data permissions within the database as well as add authenticated users. These three will create the foundation of the entire project.

Part 2: MongoDB Java Drivers

The MongoDB Java Drivers provide "both synchronous and asynchronous interaction with MongoDB" [1]. The drivers boast a large scale of features, which include the Binary JSON library. The MongoDB Java Drivers also use Reactive streams to provide asynchronous processing to the application. Reactive stream's primary purpose is to "govern the exchange of stream data across an asynchronous boundary…while ensuring the receiving side is not forced to buffer arbitrary amounts of data [2]." The primary purpose of the Java Drivers within my project is to 1) create the connection between the application and the database, and 2) utilize all create, read, update, and delete operations to achieve the desired outcome.

Creating the connection to MongoDB is not too difficult. If everything was set up and configured on the database—the connection will succeed. The code in Figure 2 creates the connection and is available in the mongodbStream.java class. The mongodbStream class contains methods to connect to the database, store username / passwords to be sent over ConnectionString, and error handle any problems with connecting. The class also retrieves the main collection that has all the user basic information and is stored in "Members."

```
public void connectDatabase()
{
    //Sets a level to the JULLogger, lots of visible text in red on console.
    java.util.logging.Logger.getLogger("org.mongodb.driver").setLevel(Level.SEVERE);

    try
    {
       // Create a connection string
       ConnectionString connectString = new ConnectionString(
        "mongodb://" + encodeValue(getUsername()) + ":" + getPassword() +  "@realm.mongodb.com:27020/?authMechanism=PLAIN&authSource=%24external&ssl=t

        //Set the settings for the connection
        MongoClientSettings settings = MongoClientSettings.builder()
                .applyConnectionString(connectString)
                .retryWrites(true)
                .build();

        //Create the connection
        mongo_client = MongoClients.create(settings);
        System.out.println("Connected");
        database = mongo_client.getDatabase("SecurityForcesCollection");
        MongoCollection<Document> collection = database.getCollection("Members");

    }
    catch(Exception e)
    {
        System.out.println("Did not connect to Database properly");
        System.out.println(e);
    }

}
```

Figure 2 - Connecting to the Database Using Reactive Java Drivers

The other reasons for the Java Driver focus on CRUD operations. The extensive BSON library helps accomplish this. BSON stands for "Binary JSON" and is "a binary-encoded serialization of JSON-like documents [3]." Several classes represent BSON Documents. They include BsonDocument, Document, DBObject, and Bson. These classes are used

interchangeably, as needed, to query for JSON data from MongoDB. I separated all CRUD operation functions into their own directly under “databaseOps”. The operations are then separated into multiple classes depending on the request. For example, there are employee, supervisor, request, and schedule options.

```java
package com.application.databaseOps;

import com.application.connection.mongodbStream;
import com.mongodb.client.MongoCollection;
import org.bson.Document;
import org.bson.conversions.Bson;
import org.bson.types.ObjectId;
import java.util.Objects;
import static com.mongodb.client.model.Filters.eq;

public class requestsIO {
    public static MongoCollection<Document> requestsCollection = mongodbStream.database.getCollection("TimeOffRequests");

    public static void sendNewRequest(String requestStart, String requestEnd, String typeRequest, String startTime, String endTime, String reason) {
        try {
            Document request = new Document("_id", new ObjectId());
            request.append("user_id", employeeIO.getUserId())
                    .append("request_start", requestStart)
                    .append("request_end", requestEnd)
                    .append("type", typeRequest)
                    .append("start_time", startTime)
                    .append("end_time", endTime)
                    .append("reason", reason)
                    .append("supervisor_id", employeeIO.getSupervisorId())
                    .append("supervisor_name", employeeIO.getSupervisor())
                    .append("status", "pending");

            requestsCollection.insertOne(request);
        } catch (Exception e) {
            e.printStackTrace();
        }

    }
```

Figure 3 - Using .insertOne() and the Document/ObjectId classes to create a new Object in the Database

The example shown in Figure 3 is in my requestsIO.java class. The purpose of this class is to allow an employee to submit time off requests and have the data inserted into the database. The supervisor_id is what must match for the employees’ supervisor to get the request. The function takes several parameters from the user interface and inserts it into the database. The creation of a new “_id” using the new ObjectId() class is also demonstrated. The _id is a randomly generated value within the database.

These examples provided above only crack the surface of the Java Drivers capabilities. Other features include text searching, encryption, TLS/SSL, geospatial searches, compression of messages, logging, and monitoring of services. TLS stands for “Transport Layer Security” and SSL stands for “Secure Socket Layer”. TLS and SSL are used to encrypt data being sent from one access point to another. MongoDB limits the size of Documents to 16MB. If a document exceeds that limit—the GridFS library becomes even more useful as it allows for the partitioning of and storage of filers larger than 16MB all identified by the same unique object id.

Part 3: MongoDB Atlas

The primary use for MongoDB Atlas is to manage my database with an easy to access user interface. MongoDB Atlas allows for insertion of data using the interface, admin access control settings, and other primarily only admin related settings. In Figure 4 its can be seen how data is represented for the application. The Security Forces Collection is the primary database where all the sub-collections are stored. The database is broken down into several different collections for better organization. These separations also assist in producing cleaner code when querying or inserting data. In one of the collections all the personal information for the employee is stored, in another the work schedule and finally any requests from employees to employer. Figure 5 shows further of how data is represented in the database for the scheduling.

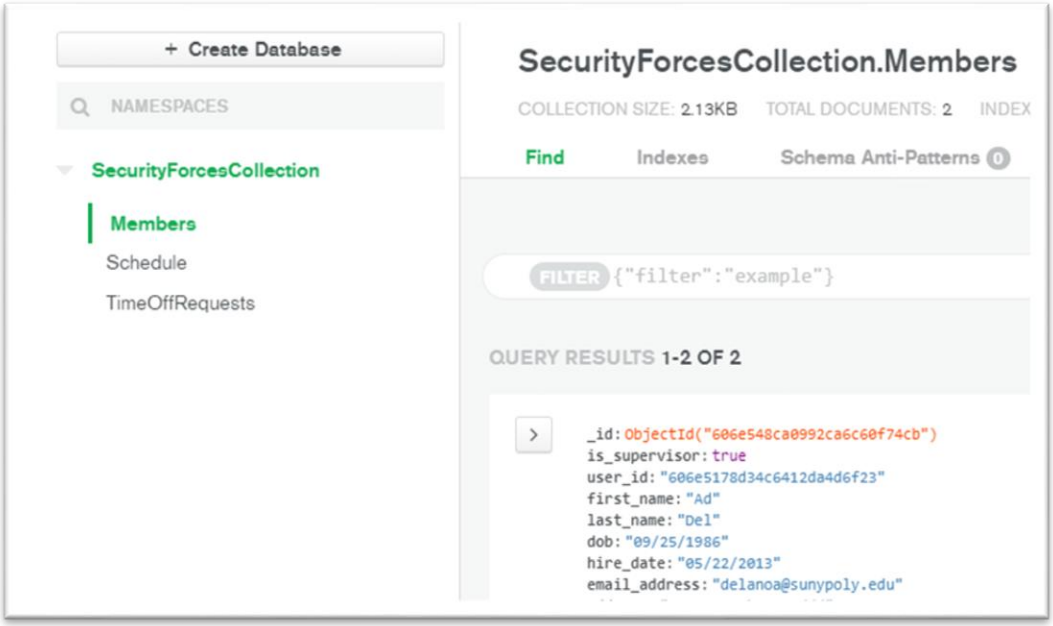

Figure 4 – Database with collections in it.

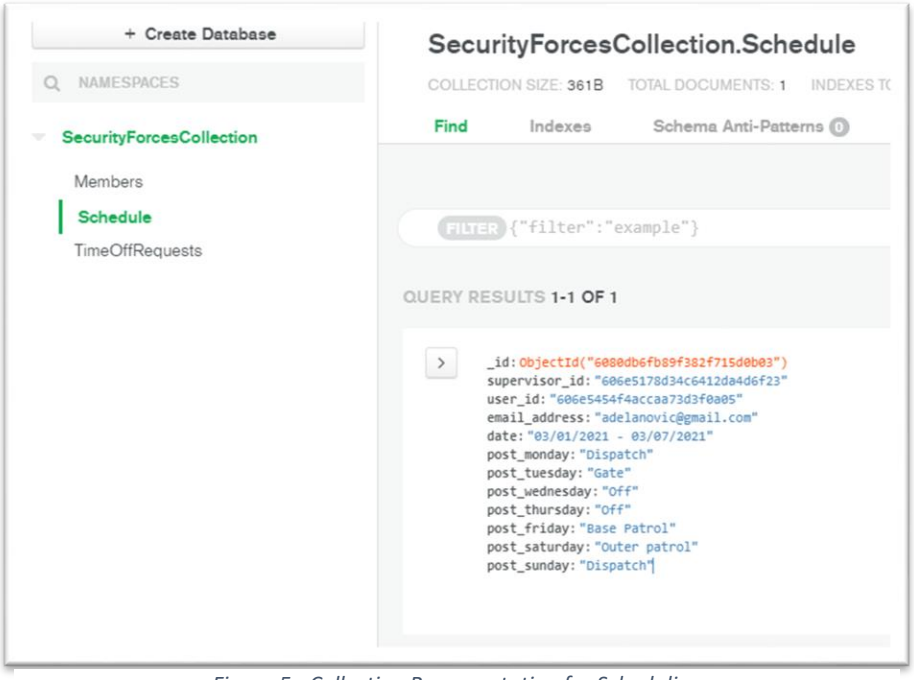

Figure 5 - Collection Representation for Scheduling

Part 4: MongoDB Realm

MongoDB Realm was the backbone of creating access control for data stored in the collections. Ensuring that supervisors had greater access to data than regular employees is essential to any employee management software. In my application the shifts are broken down into three separate elements. Not only did I have to ensure that supervisors had greater access than regular employees—I also had to ensure that supervisors only had access to their elements. The creation of access rules and any role-based permissions, authentication and users are all handled by MongoDB Realm. There are no passwords stored in plaintext and they are encrypted using SHA-256. SHA-256 "generates an almost unique 256-bit (32-byte) signature for a text [5]."

Figure 6 shows how users of the application are stored. Upon the registration of a new user—each user is given a login e-mail/password combination as well as a uniquely generated Id. Using the Id combined with a Boolean of "is_supervisor" is how the application can distinguish between regular employees and supervisors.

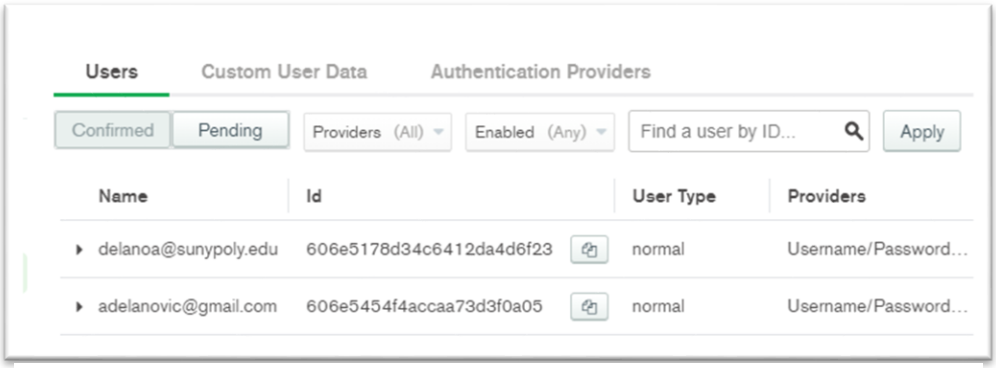

Figure 6 - Users and their unique IDs

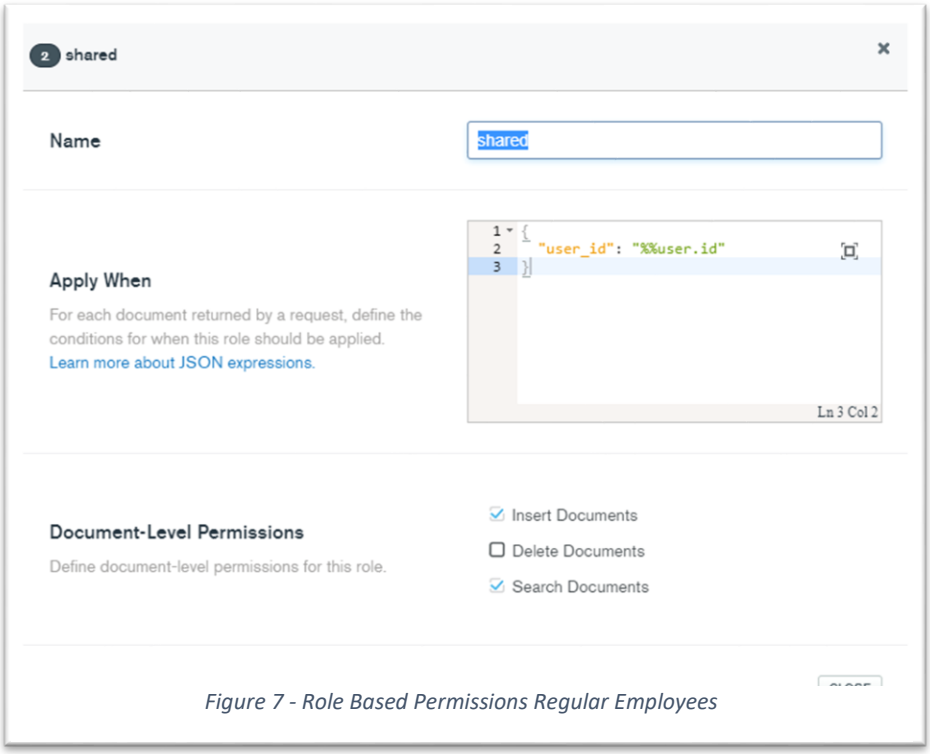

Figure 7 - Role Based Permissions Regular Employees

In Figure 7, we can also see how role based permissions for regular employees are determined. Inserting and searching of documents is the only thing permitted.

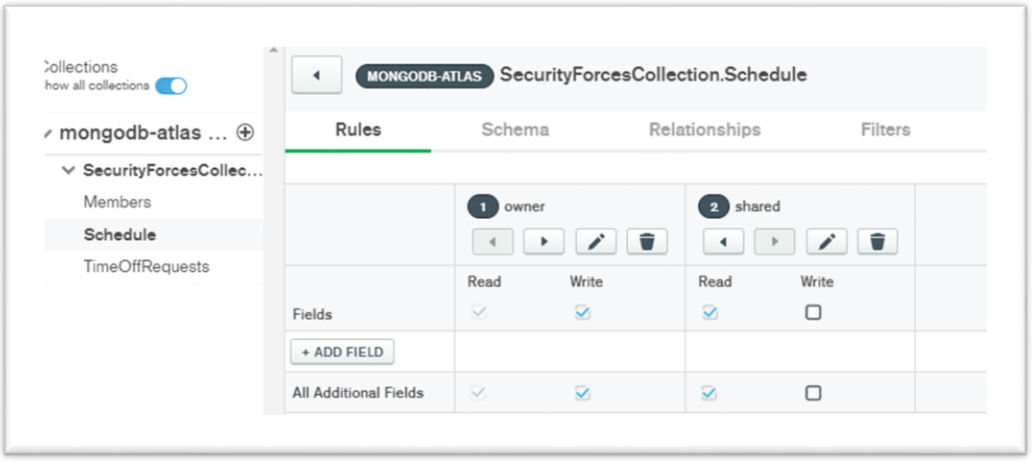

Figure 9 - Role based permissions.

Figures 9, displays how user permissions are set for creating of schedules. As shown in Figure 8, field "supervisor_id" represents the unique user Id for each application supervisor.

Supervisors can insert, delete, and search any documents in the scheduling collection for their employees. In the same collection regular employees are represented by "user_id" and their only permission is to search and insert new requests. Every request is tied to a user_id making the request, and the supervisor_id responsible for any approvals.

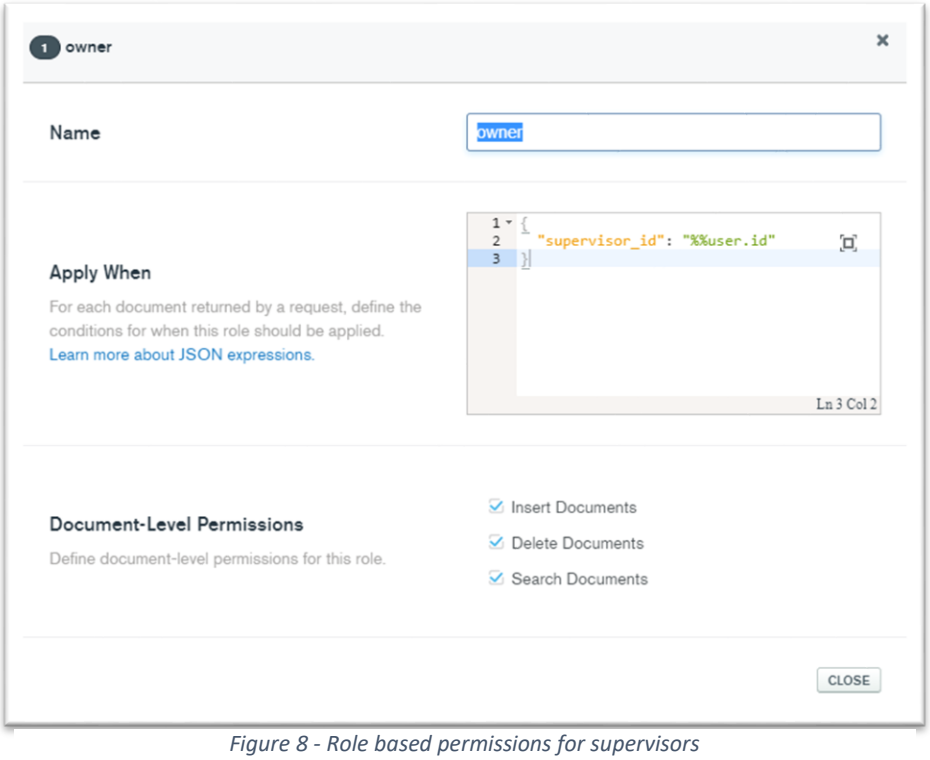

Figure 8 - Role based permissions for supervisors

Part 5: Putting It All Together

The first goal was to create a successful connection to a MongoDB Atlas Cluster and MongoDB Atlas Application. Users are added to the application directly using the MongoDB Admin management tools. The code provided on Figure 10 successfully connects to the MongoDB Realm Application with a correct username and password provided. It then gets the "Security Forces Collection" database and the default collection of "Members".

```
public void connectDatabase() {
    //Sets a level to the JULLogger, lots of visible text in red on console.
    java.util.logging.Logger.getLogger("org.mongodb.driver").setLevel(Level.SEVERE);

    try {
        // Create a connection string
        ConnectionString connectString = new ConnectionString(
         "mongodb://" + encodeValue(getUsername()) + ":" + getPassword() +  "@realm.mongodb.com:27020/?au

         //Set the settings for the connection
         MongoClientSettings settings = MongoClientSettings.builder()
                 .applyConnectionString(connectString)
                 .retryWrites(true)
                 .build();

         //Create the connection
         mongo_client = MongoClients.create(settings);
         System.out.println("Connected");
         database = mongo_client.getDatabase( databaseName: "SecurityForcesCollection");
         MongoCollection<Document> collection = database.getCollection( collectionName: "Members");
```

Figure 10 - Code for Connecting to Application

With a successful connection to the database, it was time to insert some data that I could utilize as well as implement the query methods I needed in Java. This required a lot of reading through the Java Driver documentation and testing. At times, my permissions were not correct and allowed users that should not be able to, access certain data.

```
/**
 * Gets the sick time used of the current logged in user.
 * @return String that contains the sick time used of the user.
 */
public static String getSickTimeUsed() {
    Bson filter = eq( fieldName: "email_address", loginStage.activeUser.getUsername());
    return (String) (Objects.requireNonNull(employeesCollection.find(filter).first())).get("sick_time_used");
}

//Setter Methods

/**
 * Sets the first name of the user
 * @param firstName, a String that contains the first name of the user
 */
public static void setFirstName(String firstName){
    Document find = employeesCollection.find(new Document("first_name", getFirstName())).first();

    try {
        if (find != null) {
            Bson newValue = new Document("first_name", firstName);
            Bson operation = new Document("$set", newValue);
            employeesCollection.updateOne(find, operation);
        }
    }catch(Exception e){
        e.printStackTrace();
```

Figure 11 - A query example of find and update one

Figure 11 demonstrates how finding/querying a result is achieved and how a value in the database can be updated. The getSickTimeUsed function query a collection based on the user login and returns a value of "sick_time_used". The Bson filter

utilizes an “eq” (equal to) way to find the first value. The method that follows it as a method used to update a first name in the database. The Document class instead of the Bson filter. If the value is found, the “$set” and “updateOne” operations are used to update the value in the database.

Having successfully connected to the database and utilized basic update operations I then set out to find out how to conduct insert operations. Figure 12 shows how a document is inserted into a collection. Utilizing the Document class with the ObjectId class several JSON fields can be appended, and their values determined. After the appends are all completed the document is inserted successfully into the database. The above code is what handled the time off requests by employees.

```
/**
 * Returns the user ID of the current user that made the request
 * @param requestStart date request end date, type of request, start time, end time and reason. All update certain thing
 */
    public static void sendNewRequest(String requestStart, String requestEnd, String typeRequest, String startTime, Stri
        try {
            Document request = new Document("_id", new ObjectId());
            request.append("user_id", employeeIO.getUserId())
                    .append("first_name", employeeIO.getFirstName())
                    .append("last_name", employeeIO.getLastName())
                    .append("request_start", requestStart)
                    .append("request_end", requestEnd)
                    .append("type", typeRequest)
                    .append("start_time", startTime)
                    .append("end_time", endTime)
                    .append("reason", reason)
                    .append("supervisor_id", employeeIO.getSupervisorId())
                    .append("supervisor_name", employeeIO.getSupervisor())
                    .append("status", "pending");

            requestsCollection.insertOne(request);
```

Figure 12 - An example of inserting a document into a collection

The last operation is the delete operation. Figure 14 shows how the delete operation can be used to remove an object from the database. An employee can cancel a requested a day off by removing the request from the database.

```
cancelBtn.setOnMouseClicked(f -> {
    Node source = (Node) f.getSource();
    int row = GridPane.getRowIndex(source);
    MongoCollection<Document> requestsCollection = mongodbStream.database.getCollection( collectionName: "TimeOffRequests");
    Bson filter = and(eq( fieldName: "user_id", employeeIO.getUserId()), eq( fieldName: "request_start",requestsIO.requests.get(row).getRequest_start()));
    requestsCollection.deleteOne(filter);
    cancelBtn.setText("Removed");
    cancelBtn.setStyle("-fx-background-color:   #f6a91c; -fx-text-fill: black");
});
```

Figure 13 - An example of a delete operation

Having gotten good understanding of how the CRUD operations work with the Java MongoDB Driver I started planning my User Interface and how I expect the finished product to react to user input. On the left side of Figure 14, shows a flowchart I created very early on that helped me out a lot when I got lost as to what my vision for the program was. On the right side of Figure 14, you can see a UML diagram for my connection class.

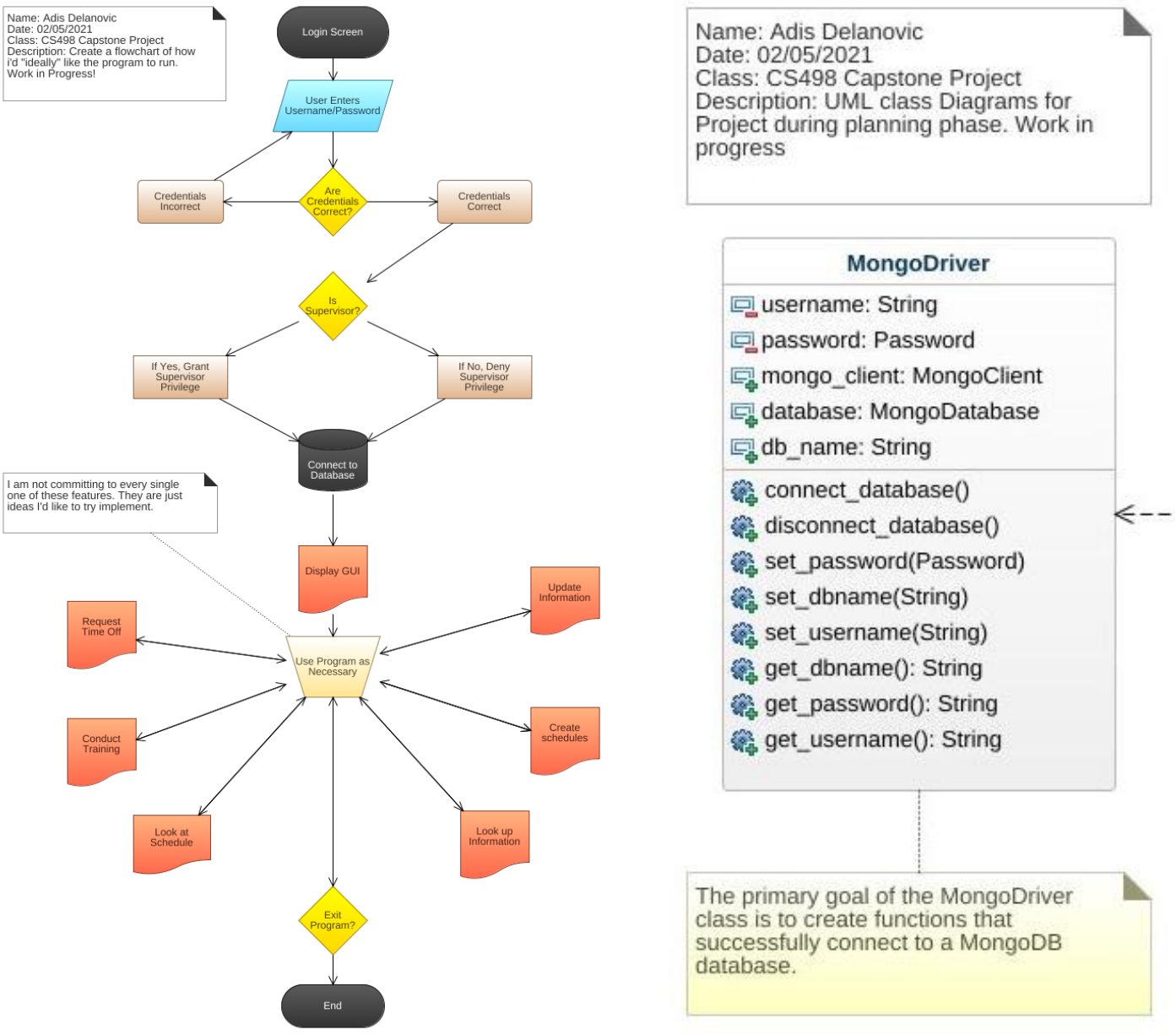

Figure 14 – Design Phase, Initial Requirements on the left and a UML diagram for a class on the right.

One of the tools that became useful was Googles GSON to JSON library. This library allowed me to create classes that mimic the JSON request and store the request into arrays whenever necessary. The Gson library "provides a powerful framework for converting between JSON strings and Java objects. This library helps to avoid needing to write boilerplate code to parse JSON responses yourself [4]". The way they are created can be seen in the requestResponse.java, passdownsResponse.java and employeeResponse.java classes. Figure 15 shows an example of a Json to Gson class.

```
package com.application.databaseOps;
import com.google.gson.annotations.Expose;
import com.google.gson.annotations.SerializedName;

/**
 * Defines the Time Off request JSON to GSON Array.
 * Class: CS498 Capstone Project
 * Instructor: Professor Spetka, Scott
 * Date: 02/15/2021
 * Last Revision: 02/15/2021
 * @author Adis Delanovic
 *
 */

public class requestResponse {
    @SerializedName("user_id")
    @Expose
    private String userId;

    @SerializedName("request_start")
    @Expose
    private String requestStart;

    @SerializedName("request_end")
    @Expose
    private String request_end;
```

Figure 15 - Json to GSON class

Making the JSON to GSON class is the first step. The second step involves a function that converts the two between each other. In Figure 16, a function is show that converts a JSON response to a GSON response. For every item found in the collection the Json is parsed, converted into a Java class, and stored into an ArrayList made up of such class. In this example, the ArrayList is created of items requestResponse.class. This function is getting time off requests from employees to a supervisor.

```
/**
 * Returns all requests of a user. Utilizes the google GSON library to convert from JSON to
 * an ArrayList. The requestResponse ArrayList is populated.
 */
public static void getRequestsForSupervisor() {
    Consumer<Document> printConsumer = new Consumer<Document>() {
        @Override
        public void accept(final Document document) {
            String response = document.toJson();
            JsonElement je = JsonParser.parseString(response);
            Gson gson = new GsonBuilder().create();
            requestResponse responses  = gson.fromJson(je, requestResponse.class);
            requests.add(responses);
        }
    };

    requestsCollection.find(eq( fieldName: "supervisor_id", employeeIO.getUserId()))
            .forEach(printConsumer);
}
```

Figure 16 - Json to Gson converter

Part 6: Demonstration

When a user runs the program the first thing they see is a login page as shown in Figure 17. The username is associated with an email for the employee and the password is associated with their account as well. MongoDB Realm is used for the authentication and all passwords are stored securely and encrypted using SHA-256 sent over SSH and TLS. SSH is "Secure Shell Proctocol" and its used for "secure remote login and other secure network services over an insecure network [5]".

Figure 17 – Login Page

The user interface allows for the logging in of regular employees and those with extra privileges (i.e a Supervisor). The database is and objects in the database are set up where if a regular employee did try to login with supervisor privileges they would be denied. Inside the database is a unique user_id that represents each user and a Boolean "is_supervisor" that defines if a user is allowed extra privileges as shown in Figure 18.

```
_id: ObjectId("6090c4d79abd58314a640d59")
is_supervisor: false
user_id: "606e5454f4accaa73d3f0a05"
first_name: "Bruce"
last_name: "Springsteen"
dob: "10/03/1983"
hire_date: "10/22/2017"
email_address: "adelanovic@gmail.com"
address: "3353 Lansing Street"
```

Figure 18 – Database Object Representation

The login view also provides several other features. It error checks for proper username and passwords as well as catches any general error that a user might enter. There is also a method attached that provides a user to "Show Password" instead of viewing it as "***" characters as shown in Figure 19 and 20.

In Figure 19, the error handling for attempting to login as a supervisor without the proper permissions. The access is denied and the user is unable to login to the supervisor view.

Figure 19 – Permission Denied for Supervisor Access

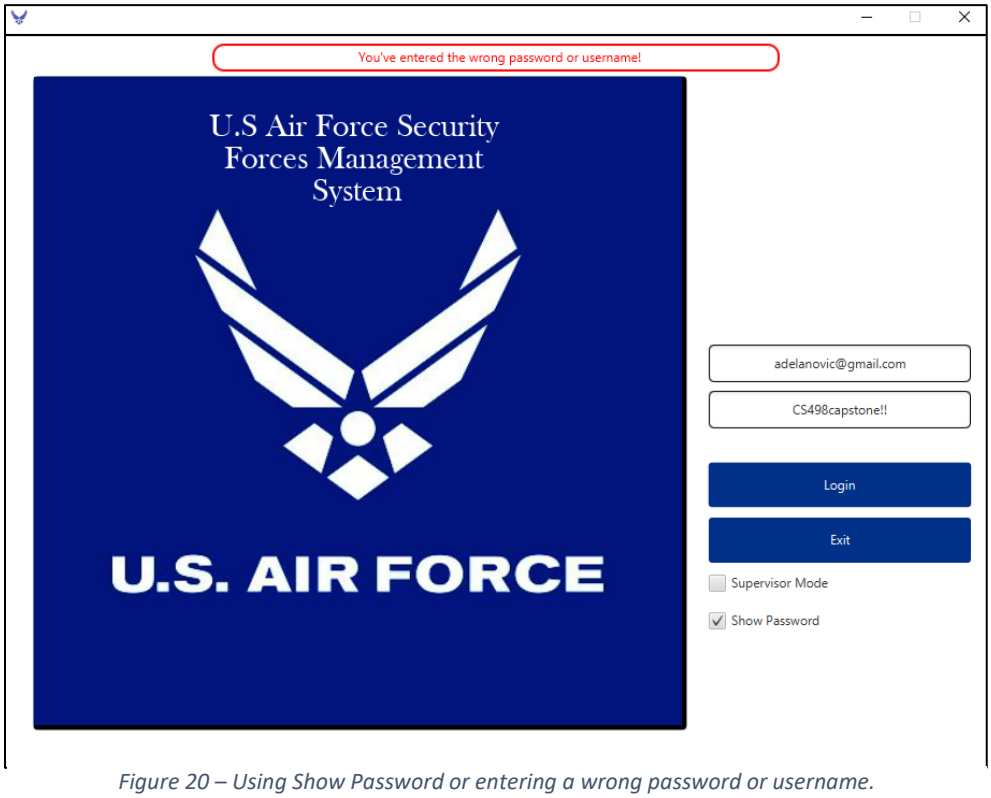

Figure 20 – Using Show Password or entering a wrong password or username.

In Figure 20 it is show how a user can show their password as plaintext string. The view rotates back and forth depending on if "Show Password" is selected. If a user enters a wrong password or username they are notified.

When a regular employee successfully logs in, they are taken to a view that does not allow for any privileged methods as shown in Figure 21. They can do what most regular employees can do. View their information, request time off, access basic resources etc.

Figure 21 - An employee viewing their general Information

An employee can view the general information about them. Their contact information, job information and specialties are all displayed. Employees are also allowed to edit certain things about them such as their contact information by clicking the "Edit Contact Information" button. Once anything is edited—the proper function is triggered, and the data is permanently changed in the database. Clicking the "Save" button as shown in Figure 22 and 23 will alter the data. An employee can also edit their emergency contact information and save it permanently. While the regular employee will never have the same access that a supervisor would have—they still must be given a small amount of customization on their data. All items are easily accessible using a Tab Pane.

Figure 22 - An employee changes their address, and permanently saves it

In Figure 22, We can see that this employee decided to edit their personal information in under the "Contact Information". Once the fields are edited—the employee can click "Save" for permanent storage in the database.

Just as with the Contact Information in Figure 22, we can see that in Figure 23, an Employee is also able to edit their primary and secondary contacts. Once again they are able to click "Save" and store the data permanently.

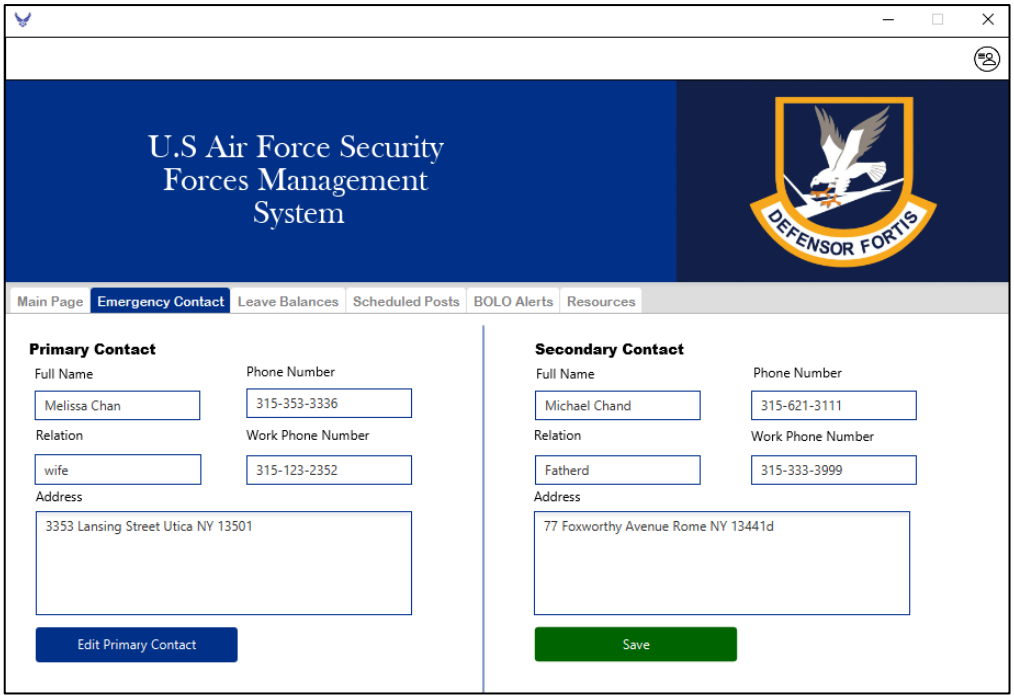

Figure 23 – An employee editing all of their emergency contact information

Another thing that required implementation was allowing employees the ability to schedule time off, as well as see previous time off requests. As shown in Figure 24, an employee can see all their prior time off requests on the right. They can see pertinent information such as the status of their request (“approved”, “denied”, and “ending”) as well as cancel any requests or archive denied requests. All the required information is present for requesting time off such as start, end dates, start and end times, and reasons. The leave balances of an employee are on the left and they can keep track of it as they make their requests. The design of the interface makes it easy for employees to request time off.

Figure 24 - Showing the employee requesting time off page.

In Figure 25, we see an example of an employee requesting a date off. They are able to use a “Date Picker” instead of having to enter dates manually. This feature is present in many applications such as this one.

Figure 25 – An employee selecting a date to request a day off

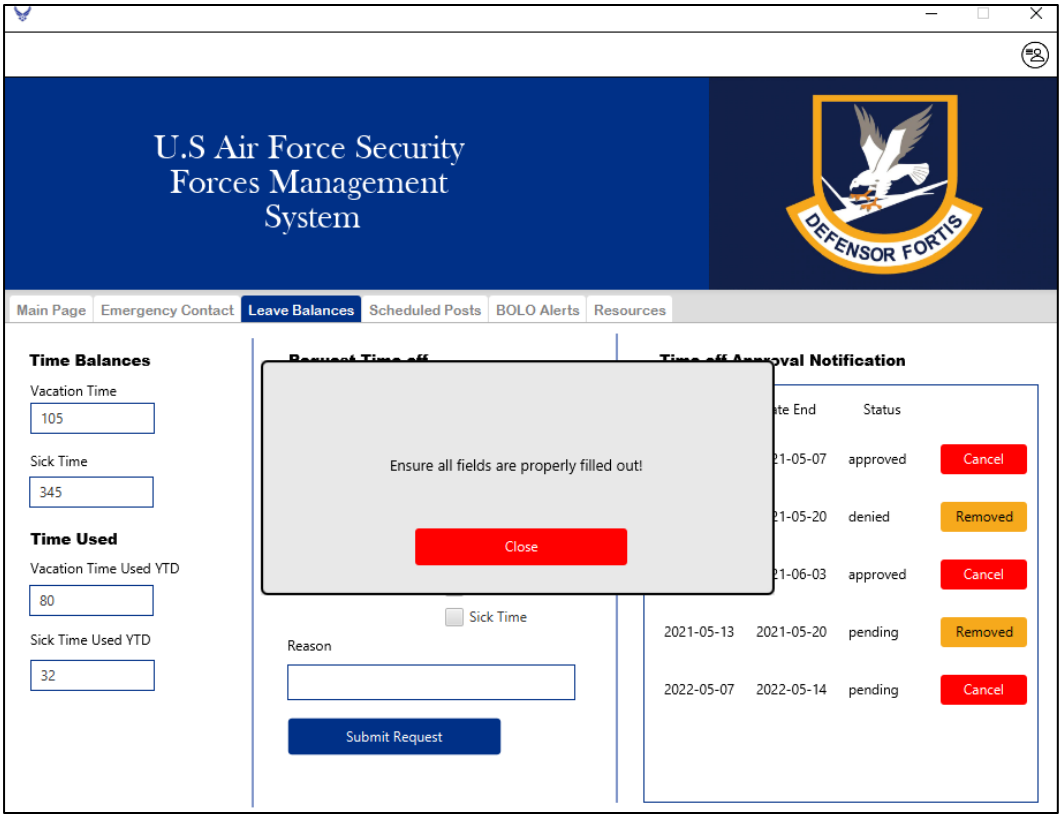

Figure 26 – Error checking an employees request

All fields are also error checked as shown in Figures 26 and Figure 27. In Figure 26 we see that this employee did not properly fill out all required fields. Error handling includes, if the start date exceeds the end date or if the start time exceeds the end time. All times are required to be entered in 0000:2400 military format. Other error checking includes empty fields, or double selections of both sick and vacation time. When an employee submits a request successfully it is also pertinent they are given

feedback that the request has gone through. The new request will also load onto the time approval notifications pane.

Figure 27- Successful Submission of a request

In Figure 27, there is a new submission for a time off request and it is successfully submitted through the system. The request is stored permanently on the database.

In Figure 28, there is a new submission request for time off approval while the ones the employee archived are successfully removed

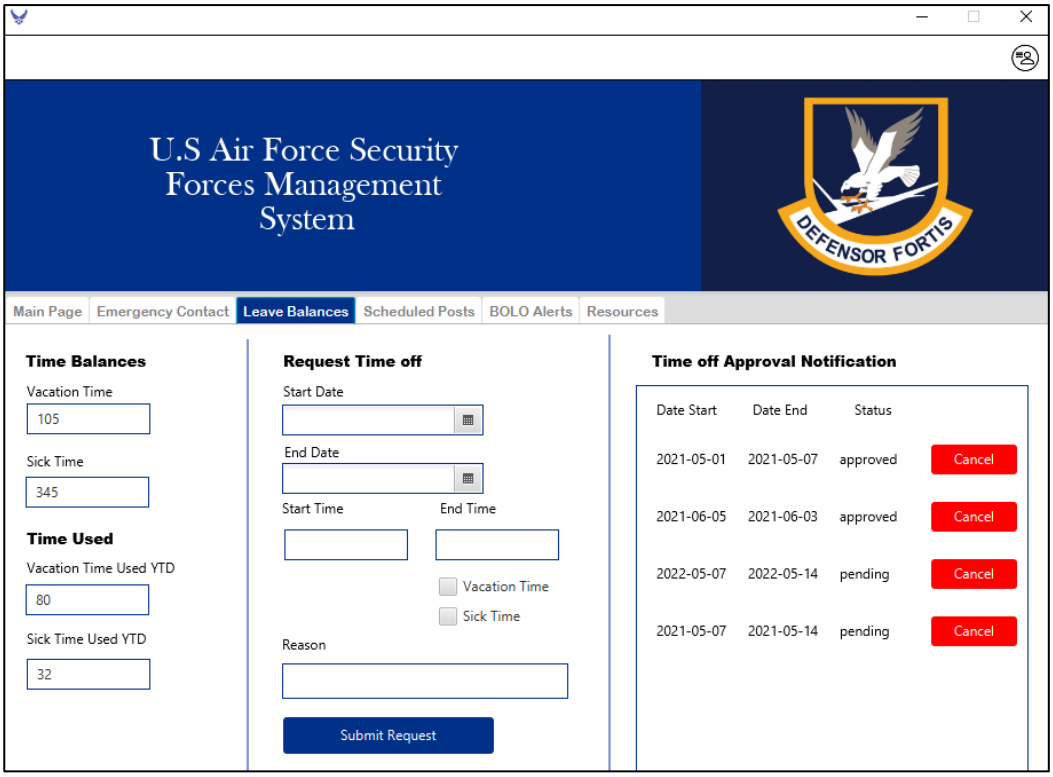

Figure 28 – Shows a new submission for a Time off request

One of the final resources left to implement that is pertinent to their job would be the ability to view a schedule for the week. An employee can do just that! They can view their weekly schedule and find out where they are posted as shown in Figure 30. For Police or Security Forces it is important to always know where you are the following day so that you can come prepared.

Figure 30 – An employee views their schedule

Figure 29 - An employee browsing BOLO Alerts (Be on a Lookout)

Some miscellaneous items I implemented was using the Java Web Engine to load resources, that are pertinent to the job, inside the employee management system. In Figure 29, the top ten most wanted fugitives are shown and are directly loaded from the FBI website into the application.

Another example of using the WebEngine to load outside resources is shown in Figure 31. The Air Force has many websites that provide Security Forces and Airmen the resources they need. These resources are directly loaded into the application.

Figure 31 - An employee browsing additional resources.

Figure 32 - Once finished, clicking in the top right corner, brings up an additional menu

Figure 32 demonstrates exiting an application. A user clicks on the profile icon in the corner and brings up a new menu where they can either "Exit" or "Logout". That concludes the demonstration on what a regular employee can do. There are many improvements that can be made to these features as well as some bug fixes that I had issues with. The implementation of giving regular employees only certain privileges, while compartmentalizing supervisor privileges, was a

success. Other features I would like to implement searching schedules of certain dates, as well as the archiving of previous work dates. Adding or subtracting from an employee leave balance based on the leave request would also be an essential addition.

After the successful implementation of all regular employee methods, it was time to make the supervisor user interface. Upon successful authentication and a check of the user is a supervisor—the supervisor user interface is loaded. The immediate difference is clear as the Tabs are gone, and the menu is located at the top. Since this employee, management system specifically targets a certain demographic (Security Forces/Air Force Police) the supervisor is brought to a default page in Figure 33.

Figure 33 – Default View for a supervisor login.

One of a supervisor's main duties is to create schedules. The program allows the ability for the supervisor to do just that. Clicking on the "Scheduling" menu brings a supervisor to a new interface. The immediate display of the employees is dynamically generated and displayed on the left as shown in Figure 34. All these employees only match the "supervisor_id" of the current supervisor logged in. Therefore, seeing other supervisors' employees would not be possible.

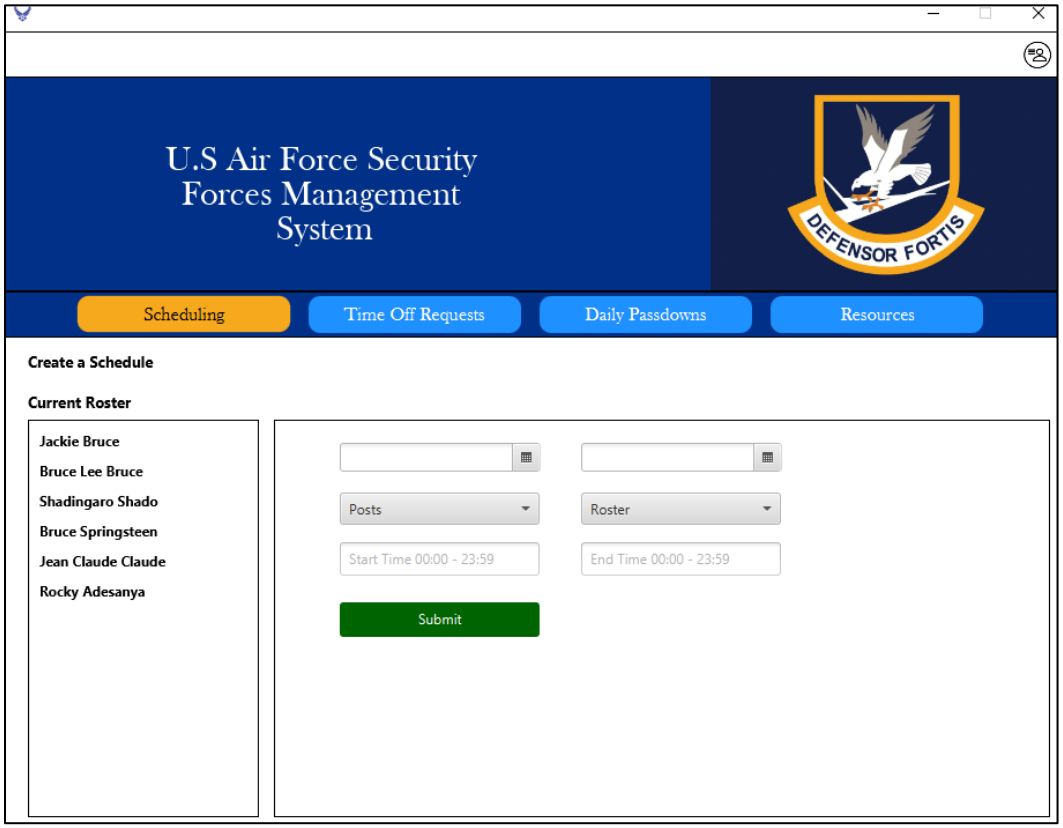

Figure 34 – A supervisor making a schedule

Figure 35 - A supervisor making a schedule, selecting posts.

Using drop downs a supervisor can easily see what posts need to have someone there as shown in Figure 35 and Figure 36. Dates can also be selected, as well as start and end times. The roster combo box also displays the current roster for easy selection instead of having to type employees' names in. There are several improvements that can be made to the scheduling. There is some

error checking involved but ideally checks with the database should be made if there is a data/post filled already for a certain day.

Figure 37 continues to demonstrate the ease of use and drop down menus associated with creating a schedule. In this instance—the names are selected from a roster instead of typed in.

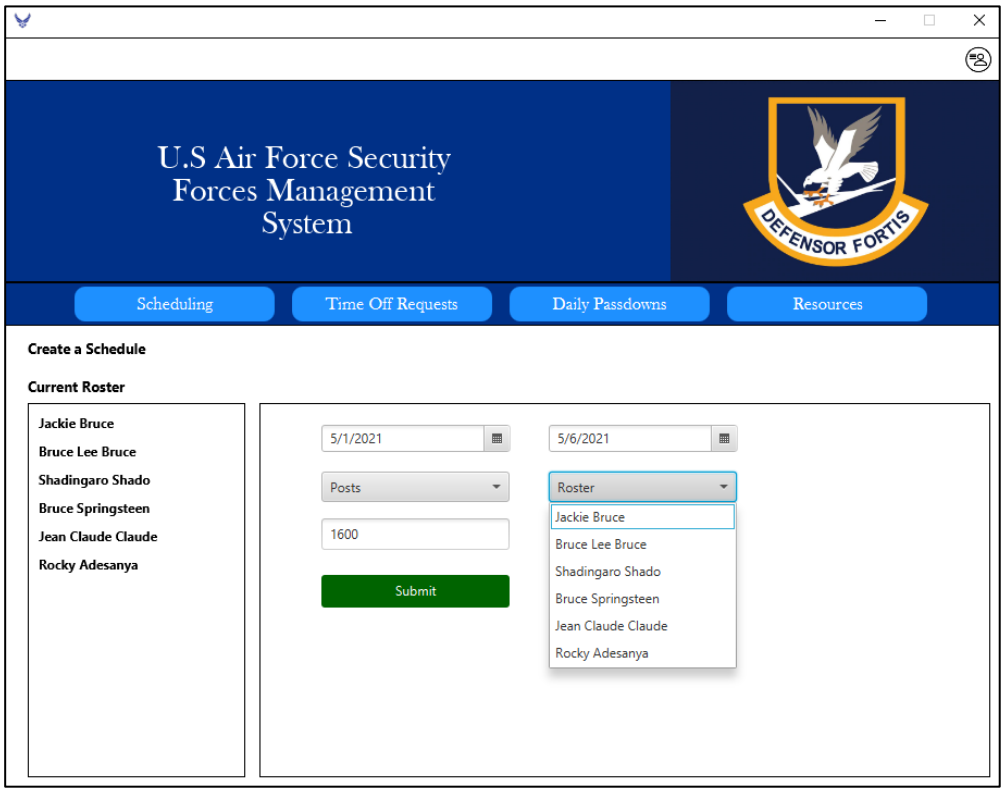

Figure 37 - A supervisor making a schedule, selecting an employee from their roster.

Figure 36 – Successful Submission of a new schedule

Figure 36 demonstrates the feedback a user of the application will receive upon a successful submission of a new shift schedule. The schedule is immediately also placed into the database for permanent storage and only accessible by a supervisor.

A supervisor is also capable of approving or denying any request time off as shown in Figure 38. As its updated by the supervisor as either approved or denied, the employee is notified appropriately. Here we have employee “Jackie Bruce” just wanted all the time off in the world. The user interface is changed appropriately as the supervisor approves or denies time off.

Figure 38 - A supervisor approving or denying time off.

The final menu I implemented was allowing supervisors to communicate between each other. It is pertinent, especially in law enforcement settings, that supervisors can share information between each other and keep the shifts updated on anything that happens or will happen. The "Daily Passdowns" menu provides just that as shown in Figure 40. A supervisor can either enter a new pass down and click "Submit" to store it into the database or they can view previous pass downs that they or other supervisors

Figure 40 - A supervisor can either submit a new pass down or view other ones

The pass downs include whom they were submitted by the date and time of the submission as well as the information itself as shown in Figure 39. A supervisor should be able to print and save these—however I experienced some issues with the print methods. I left them in as I intend to finish this development after the class ends.

Figure 39 - A supervisor viewing previous pass downs

Figure 41 - A supervisor passing information down

As with anything in the program—the user is always notified of any successes or failures. Figures 41 and 42 show a successful insert into the pass downs and update in the user interface. Once a supervisor is done, they can click the right-hand corner and exit or logout just as a regular employee.

In Figure 42 we can see that there are now additional items on the passdowns and all the proper information is stored.

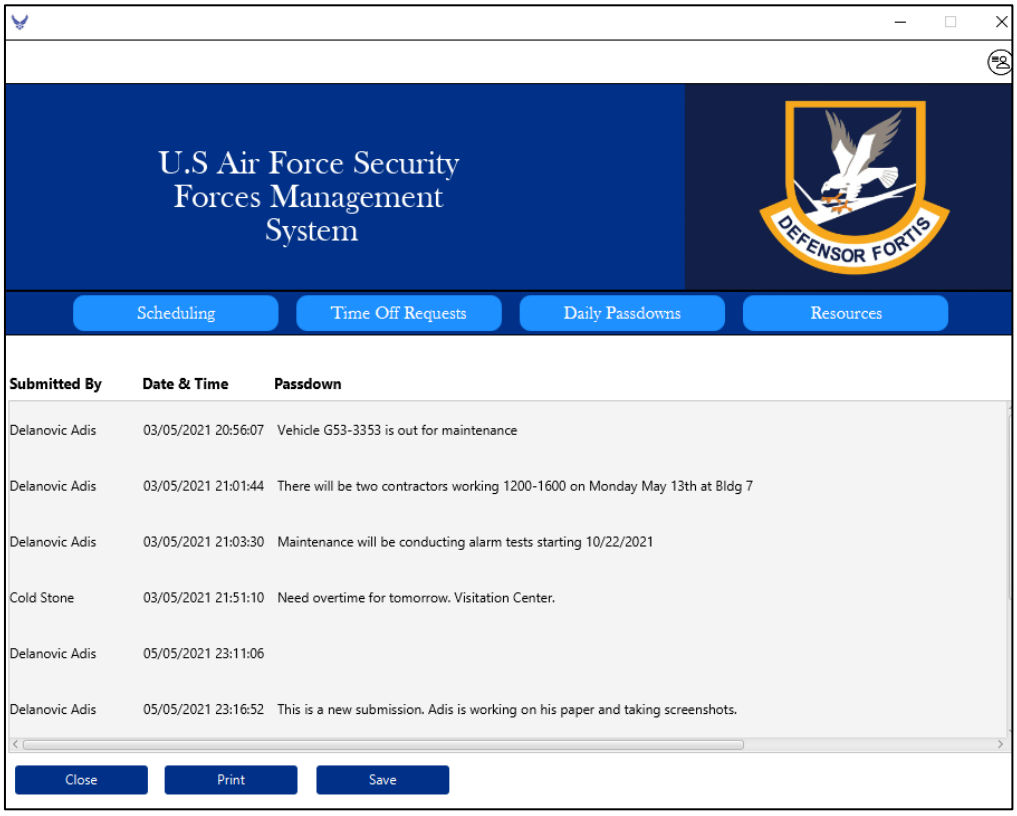

Figure 42 - A supervisor viewing an updated pass down list

Figure 43 - Pass downs storage as seen in the database.

In Figure 43, we show the inner workings of the database and display how the pass downs data is stored into the database. We have the unique _id, the user_id of the user that submitted it, the name of the supervisor and the date/time as well as the pass down itself. Although I covered many of the tasks required by a supervisor—there are many things that can be improved. Specifically, all archiving of data should be done so that all past and present requests can be archived. This includes the time off requests and pass downs. There could also be a different user interface design that allows for easier scheduling for employees. These are all issues I plan to continue to work on as this class ends.

Part 7: Results and Conclusions

I had several objectives when I first started this project. One of the first ones was to be able to plan, develop and deliver a piece of software that can be utilized in the real world. My inspiration for it came from my current job that uses hard paper for most employee management such as requesting time off. Another goal was to become a better Java Developer. I had some potential employers talk to me about being hired on as an Android Java Developer upon graduation. With that said, I was able to achieve all my starting goals and deliver a product I enjoyed working on.

Prior to this project I was not very knowledgeable in the extensive libraries out there for creating a user interface. As can be seen from my earlier logs, I would write tons of code in Java and have everything generated dynamically. While working on this project I learned about FXML and FXML loaders. This made the creation of the UI more akin to designing a webpage than anything else. Learning how to manage these stages using controllers also taught me a lot about Java dot notation chaining. It required a lot of time and testing to get everything working correctly. This also assisted me in finding such methods for other programming languages such as Python and C++. The proper handling of dependencies using Maven and Gradle was something I learned as well. Normally I would download the entire *.jar files and save them into my program directly. Using Gradle I was able to define my dependencies without having the need for local storage.

The decision to use MongoDB instead of PostgreSQL was the right one for this project. I ended up learning a new technology that I had a very limited amount of knowledge in. There were many hours spent on reading through MongoDB documentation and database security. I know that I could not store my passwords as plaintext into my document and needed a proper authentication handler. MongoDB Realm proved to be more capable for the task and I was able to achieve SHA-256/TLS/SSL encrypted authentication. Combing the data clusters from MongoDB Atlas to MongoDB Realm helped me achieve authentication, user privileges, as well as persistent data storage. MongoDB is something I look forward to using again in the future! With the Java MongoDB Driver I was able to successfully implement all create, read, update, and delete methods for my

program. One of the best parts of all these technologies was the extensive documentation available on their websites. There was no need to find guides outside of the website. Prior to this I only had limited knowledge with Java from a previous course.

Using all these technologies together, I was able to gain experience in a multitude of new technologies. All the requirements of my program were properly implemented and function. Although I do believe that there are better methods out there in creating an employee management system. Such as using the MERN stack and creating a website would have been easier. The MongoDB Realm SDK provides great support for JavaScript development and only a tiny amount for Java development. All authentication flows and commands would have been much cleaner had I implemented this project in JavaScript. For example, there does not exist insert a new user for the Java Driver while if I had used JavaScript—there does exist one. All users must be inserted manually using the MongoDB dashboard. There are workarounds for it—but they are not nearly as effective or efficient as using the MongoDB Realm SDK would have been. My approach to an Employee Management System would be a little different than a Desktop application in the future. I did, however, gain invaluable experience that I will carry forward with me.

//Source Code: mongodbStream.java

```
package com.application.connection;


import com.mongodb.ConnectionString;
import com.mongodb.client.*;
import com.mongodb.MongoClientSettings;
import org.bson.Document;
import com.mongodb.client.MongoCollection;
import java.io.UnsupportedEncodingException;
import java.net.URLEncoder;
import java.nio.charset.StandardCharsets;
import java.util.logging.Level;


/**
 * Creates a connection between the application and the MongoDB Cluster.
 * Class: CS498 Capstone Project
 * Instructor: Professor Spetka, Scott
 * Date: 02/06/2021
 * Last Revision: 02/06/2021
 * @author Adis Delanovic
 *
 */


public class mongodbStream {
    private String username;
    private String password;
    private String db_name;
    public static MongoClient mongo_client; //In documents referred to as
'cursor' for methods.
    public static MongoDatabase database; //In documentation referred to as 'db'
for methods.
```

//Source Code: mongodbStream.java

```
/**
 * Creates the connection to the Database using provided password and username
 * @return 1 on success, -1 on failure
 */

    public void connectDatabase() {
        //Sets a level to the JULLogger, lots of visible text in red on console.
java.util.logging.Logger.getLogger("org.mongodb.driver").setLevel(Level.SEVERE);

        try {
           // Create a connection string
           ConnectionString connectString = new ConnectionString(
            "mongodb://" + encodeValue(getUsername()) + ":" + getPassword() +
"@realm.mongodb.com:27020/?authMechanism=PLAIN&authSource=%24external&ssl=true&ap
pName=securityforces-otexj:mongodb-atlas:local-userpass");

            //Set the settings for the connection
            MongoClientSettings settings = MongoClientSettings.builder()
                    .applyConnectionString(connectString)
                    .retryWrites(true)
                    .build();

            //Create the connection
            mongo_client = MongoClients.create(settings);
            System.out.println("Connected");
            database = mongo_client.getDatabase("SecurityForcesCollection");
            MongoCollection<Document> collection =
database.getCollection("Members");
//Source Code: mongodbStream.java

        }
        catch(Exception e) {
            System.out.println("Did not connect to Database properly");
            System.out.println(e);
```

```
        }

    }
/**
 * Disconnects the connection from the database.
 */

    public static void disconnect_database() {
        try {
            mongo_client.close();
        }
        catch(Exception e) {
            System.out.println("Unable to disconnect from Database!");
        }
    }

/**
 * Sets the user password.
 * @param password, String that contains a user provided password
 */

    public void setPassword(String password) {
        this.password = password;
    }
```

//Source Code: mongodbStream.java

```
/**
 * Sets the database name.
 * @param db_name, String that contains a user provided database
*/
    public void setDbname(String db_name) {
        this.db_name = db_name;
    }
```

```
/**
 * Sets the user email.
 * @param username, String that contains a user provided email for logging in.
 */

    public void setEmailAddress(String username) {
        this.username = username;
    }

/**
 * Gets the current Database.
 * @return db_name, String
 */

    public String get_dbname() {
        return this.db_name;
    }
/**
 * Gets the current user provided password.
 * @return password, String
*/
```

//Source Code: mongodbStream.java

```
    public String getPassword() {
        return this.password;
    }

/**
 * Gets the current Database.
 * @return db_name, String
 */
```

```
    public String getUsername() {
        return this.username;
    }

/**
 * Encode an email address so that it can be used as the login method.
 * @param value, user provided email address
 * @return value, encoded email address
 */

    private String encodeValue(String value) {
        try {
            return URLEncoder.encode(value, StandardCharsets.UTF_8.toString());
        } catch (UnsupportedEncodingException ex) {
            throw new RuntimeException(ex.getCause());
        }
    }
}
```

//Source Code: employeeIO.java

```
package com.application.databaseOps;

import com.application.connection.mongodbStream;
import com.application.gui.loginStage;
import com.google.gson.Gson;
import com.google.gson.GsonBuilder;
import com.google.gson.JsonElement;
import com.google.gson.JsonParser;
import com.mongodb.client.MongoCollection;
import org.bson.BsonArray;
import org.bson.Document;
import org.bson.conversions.Bson;
```

```
import java.util.ArrayList;
import java.util.Objects;
import java.util.function.Consumer;

import static com.mongodb.client.model.Filters.eq;

/**
 * Defines the Employee Operations to the database.
 * Class: CS498 Capstone Project
 * Instructor: Professor Spetka, Scott
 * Date: 02/15/2021
 * Last Revision: 02/15/2021
 * @author Adis Delanovic
 *
 */
```

//Source Code: employeeIO.java

```
public class employeeIO {
    public static MongoCollection<Document> employeesCollection =
mongodbStream.database.getCollection("Members");
    public static ArrayList<employeeResponse> dataRequested = new ArrayList<>();

    /**
     * Returns the user ID that is currently logged in
     * @return String that contains the User Id from the database.
     */
    public static String getUserId() {
        Bson filter = eq("email_address", loginStage.activeUser.getUsername());
        return (String)
(Objects.requireNonNull(employeesCollection.find(filter).first())).get("user_id")
;
    }
```

```
    /**
     * Returns the supervisor ID of the user that is currently logged in
     * @return String that contains the supervisor ID from the database.
     */
    public static String getSupervisorId() {
        Bson filter = eq("email_address", loginStage.activeUser.getUsername());
        return (String)
(Objects.requireNonNull(employeesCollection.find(filter).first())).get("superviso
r_id");
    }
```

//Source Code: employeeIO.java

```
/**
     * Checks if the current user is a supervisor
     * @return boolean, returns true if the user is a supervisor, false
otherwise.
     */
    public static Boolean getisSupervisor() {
        Bson filter = eq("email_address", loginStage.activeUser.getUsername());
        return (Boolean)
(Objects.requireNonNull(employeesCollection.find(filter).first())).get("is_superv
isor");
    }

    /**
     * Gets the first name of the current logged in user.
     * @return String that contains the first name.
     */
    public static String getFirstName() {
        Bson filter = eq("email_address", loginStage.activeUser.getUsername());
        return (String)
(Objects.requireNonNull(employeesCollection.find(filter).first())).get("first_nam
e");
```

```
    }

  //Source Code: employeeIO.java

  /**
     * Gets the last name of the current logged in user.
     * @return String that contains the last name.
     */
    public static String getLastName() {
        Bson filter = eq("email_address", loginStage.activeUser.getUsername());
        return (String)
(Objects.requireNonNull(employeesCollection.find(filter).first())).get("last_name
");
    }

    /**
     * Gets the date of birth of the current logged in user.
     * @return String that contains the date of birth.
     */
    public static String getDob() {
        Bson filter = eq("email_address", loginStage.activeUser.getUsername());
        return (String)
(Objects.requireNonNull(employeesCollection.find(filter).first())).get("dob");
    }

//Source Code: employeeIO.java

   /**
     * Gets the hire date of the current logged in user.
     * @return String that contains the hire date.
     */
    public static String getHireDate() {
        Bson filter = eq("email_address", loginStage.activeUser.getUsername());
```

```
        return (String)
(Objects.requireNonNull(employeesCollection.find(filter).first())).get("hire_date
");
    }


    /**
     * Gets the email address of the current logged in user.
     * @return String that contains the email address.
     */
    public static String getEmailAddress() {
        Bson filter = eq("email_address", loginStage.activeUser.getUsername());
        return (String)
(Objects.requireNonNull(employeesCollection.find(filter).first())).get("email_add
ress");
    }
```

//Source Code: employeeIO.java

```
   /**
     * Gets the address of the current logged in user.
     * @return String that contains the address.
     */
    public static String getAddress() {
        Bson filter = eq("email_address", loginStage.activeUser.getUsername());
        return (String)
(Objects.requireNonNull(employeesCollection.find(filter).first())).get("address")
;
    }


    /**
     * Gets the city of the current logged in user.
     * @return String that contains the city.
     */
    public static String getCity() {
```

```
        Bson filter = eq("email_address", loginStage.activeUser.getUsername());
        return (String)
(Objects.requireNonNull(employeesCollection.find(filter).first())).get("city");
    }
```

//Source Code: employeeIO.java

```
 /**
     * Gets the zipcode of the current logged in user.
     * @return String that contains the zipcode.
     */
    public static String getZipcode() {
        Bson filter = eq("email_address", loginStage.activeUser.getUsername());
        return (String)
(Objects.requireNonNull(employeesCollection.find(filter).first())).get("zipcode")
;
    }

    /**
     * Gets the state of the current logged in user.
     * @return String that contains the state.
     */
    public static String getState() {
        Bson filter = eq("email_address", loginStage.activeUser.getUsername());
        return (String)
(Objects.requireNonNull(employeesCollection.find(filter).first())).get("state");
    }
```

//Source Code: employeeIO.java

```
   /**
     * Gets the next evaluation date of the current logged in user.
     * @return String that contains the evaluation date.
     */
    public static String getNextEval() {
        Bson filter = eq("email_address", loginStage.activeUser.getUsername());
        return (String)
(Objects.requireNonNull(employeesCollection.find(filter).first())).get("next_eval
");
    }

    /**
     * Gets the salary of the current logged in user.
     * @return Double that contains the salary.
     */
    public static Double getSalary() {
        Bson filter = eq("email_address", loginStage.activeUser.getUsername());
        return (Double)
(Objects.requireNonNull(employeesCollection.find(filter).first())).get("salary");
    }
```

//Source Code: employeeIO.java

```
   /**
     * Gets the shift assigned of the current logged in user.
     * @return String that contains the shift assigned.
     */
    public static String getShiftAssigned() {
        Bson filter = eq("email_address", loginStage.activeUser.getUsername());
```

```
        return (String)
(Objects.requireNonNull(employeesCollection.find(filter).first())).get("shift_ass
igned");
    }

    /**
     * Gets the supervisor name of the current logged in user.
     * @return String that contains the supervisor name.
     */
    public static String getSupervisor() {
        Bson filter = eq("email_address", loginStage.activeUser.getUsername());
        return (String)
(Objects.requireNonNull(employeesCollection.find(filter).first())).get("superviso
r");
    }
```

//Source Code: employeeIO.java

```
  /**
     * Gets the job title of the current logged in user.
     * @return String that contains the first name.
     */
    public static String getJobTitle() {
        Bson filter = eq("email_address", loginStage.activeUser.getUsername());
        return (String)
(Objects.requireNonNull(employeesCollection.find(filter).first())).get("job_title
");
    }

    /**
     * Gets the certifications of the current logged in user.
     * @return BsonArray that contains the certifications of the user.
     */
```

```
    public static BsonArray getCertifications() {
        Bson filter = eq("email_address", loginStage.activeUser.getUsername());
        return (BsonArray)
(Objects.requireNonNull(employeesCollection.find(filter).first())).get("certifica
tions");
    }
```

//Source Code: employeeIO.java

```
/**
     * Gets the primary contact of the current logged in user.
     * @return String that contains the primary contact name of the user.
     */
    public static String getPrimaryContact() {
        Bson filter = eq("email_address", loginStage.activeUser.getUsername());
     return (String)
(Objects.requireNonNull(employeesCollection.find(filter).first())).get("primary_c
ontact");
    }

    /**
     * Gets the primary contact phone of the current logged in user.
     * @return String that contains the primary contact phone of the user.
     */
    public static String getPrimaryPhone() {
        Bson filter = eq("email_address", loginStage.activeUser.getUsername());
        return (String)
(Objects.requireNonNull(employeesCollection.find(filter).first())).get("primary_p
hone");
    }
```

//Source Code: employeeIO.java

```
/**
     * Gets the primary relation of the current logged in user.
     * @return String that contains the primary relation of the user.
     */
    public static String getPrimaryRelation() {
        Bson filter = eq("email_address", loginStage.activeUser.getUsername());



  return (String)
(Objects.requireNonNull(employeesCollection.find(filter).first())).get("primary_r
elation");
    }


    /**
     * Gets the primary contact work phone of the current logged in user.
     * @return String that contains the primary contact work phone of the user.
     */
    public static String getPrimaryWorkPhone() {
        Bson filter = eq("email_address", loginStage.activeUser.getUsername());
        return (String)
(Objects.requireNonNull(employeesCollection.find(filter).first())).get("primary_w
ork_phone");
    }
```

//Source Code: employeeIO.java

```
/**
     * Gets the primary contact address of the current logged in user.
     * @return String that contains the primary contact address of the user.
     */


 public static String getPrimaryAddress() {
```

```
        Bson filter = eq("email_address", loginStage.activeUser.getUsername());
        return (String)
(Objects.requireNonNull(employeesCollection.find(filter).first())).get("primary_a
ddress");
    }

    /**
     * Gets the secondary contact name of the current logged in user.
     * @return String that contains the secondary contact name of the user.
     */
    public static String getSecondaryContact() {
        Bson filter = eq("email_address", loginStage.activeUser.getUsername());
        return (String)
(Objects.requireNonNull(employeesCollection.find(filter).first())).get("secondary
_name");
    }
```

//Source Code: employeeIO.java

```
/**
     * Gets the secondary contact phone of the current logged in user.
     * @return String that contains the secondary contact phone of the user.
     */
    public static String getSecondaryPhone() {
        Bson filter = eq("email_address", loginStage.activeUser.getUsername());
        return (String)
(Objects.requireNonNull(employeesCollection.find(filter).first())).get("secondary
_phone_number");
    }

    /**
     * Gets the secondary contact relation of the current logged in user.
     * @return String that contains the secondary contact relation of the user.
     */
```

```
    public static String getSecondaryRelation() {
        Bson filter = eq("email_address", loginStage.activeUser.getUsername());
        return (String)
(Objects.requireNonNull(employeesCollection.find(filter).first())).get("secondary
_relation");
    }
```

//Source Code: employeeIO.java

```
/**
     * Gets the secondary contact work phone of the current logged in user.
     * @return String that contains the secondary contact work phone of the user.
     */
    public static String getSecondaryWorkPhone() {
        Bson filter = eq("email_address", loginStage.activeUser.getUsername());
        return (String)
(Objects.requireNonNull(employeesCollection.find(filter).first())).get("secondary
_work_phone");
    }

    /**
     * Gets the secondary contact address of the current logged in user.
     * @return String that contains the secondary contact address of the user.
     */
    public static String getSecondaryAddress() {
        Bson filter = eq("email_address", loginStage.activeUser.getUsername());
        return (String)
(Objects.requireNonNull(employeesCollection.find(filter).first())).get("secondary
_address");
    }
```

//Source Code: employeeIO.java

```
/**
     * Gets the vacation time of the current logged in user.
     * @return String that contains the vacation time of the user.
     */
    public static String getVacationTime() {
        Bson filter = eq("email_address", loginStage.activeUser.getUsername());
        return (String)
(Objects.requireNonNull(employeesCollection.find(filter).first())).get("vacation_
time");
    }

    /**
     * Gets the sick time of the current logged in user.
     * @return String that contains the sick time of the user.
     */
    public static String getSickTime() {
        Bson filter = eq("email_address", loginStage.activeUser.getUsername());
        return (String)
(Objects.requireNonNull(employeesCollection.find(filter).first())).get("sick_time
");
    }
```

//Source Code: employeeIO.java

```
/**
     * Gets the vacation time used of the current logged in user.
     * @return String that contains the vacation time used of the user.
     */
    public static String getVacationTimeUsed() {
        Bson filter = eq("email_address", loginStage.activeUser.getUsername());
```

```
        return (String)
(Objects.requireNonNull(employeesCollection.find(filter).first())).get("vacation_
time_used");
    }


    /**
     * Gets the sick time used of the current logged in user.
     * @return String that contains the sick time used of the user.
     */
    public static String getSickTimeUsed() {
        Bson filter = eq("email_address", loginStage.activeUser.getUsername());
        return (String)
(Objects.requireNonNull(employeesCollection.find(filter).first())).get("sick_time
_used");
    }


    //Setter Methods
    //Source Code: employeeIO.java


/**
     * Sets the first name of the user
     * @param firstName, a String that contains the first name of the user
     */
    public static void setFirstName(String firstName){
        Document find = employeesCollection.find(new Document("first_name",
getFirstName())).first();


        try {
            if (find != null) {
                Bson newValue = new Document("first_name", firstName);
                Bson operation = new Document("$set", newValue);
                employeesCollection.updateOne(find, operation);
            }
        }catch(Exception e){
```

```
            e.printStackTrace();
        }
    }
```

//Source Code: employeeIO.java

```
/**
     * Sets the last name of the user
     * @param lastName, String,  contains the lastName of the user
     */
    public static void setLastName(String lastName){
        Document find = employeesCollection.find(new Document("last_name",
getLastName())).first();

        try {
            if (find != null) {
                Bson newValue = new Document("last_name", lastName);
                Bson operation = new Document("$set", newValue);
                employeesCollection.updateOne(find, operation);
            }
        }catch(Exception e){
            e.printStackTrace();
        }
    }

    /**
     * Sets the address of the user
     * @param address, String,  contains the address of the user
     */
    public static void setAddress(String address){
        Document find = employeesCollection.find(new Document("address",
getAddress())).first();
```

```
        try {
            if (find != null) {
 //Source Code: employeeIO.java

                Bson newValue = new Document("address", address);
                Bson operation = new Document("$set", newValue);
                employeesCollection.updateOne(find, operation);
            }
        }catch(Exception e){
            e.printStackTrace();
        }
    }

    /**
     * Sets the first name of the user
     * @param zipcode, String,  contains the zipcode of the user
     */
    public static void setZipcode(String zipcode){
        Document find = employeesCollection.find(new Document("zipcode",
getZipcode())).first();

        try {
            if (find != null) {
                Bson newValue = new Document("zipcode", zipcode);
                Bson operation = new Document("$set", newValue);
                employeesCollection.updateOne(find, operation);
            }
        }catch(Exception e){
            e.printStackTrace();
        }
    }
```

//Source Code: employeeIO.java

```
/**
     * Sets the city of the user
     * @param city, String, contains the city of the user
     */
    public static void setCity(String city){
        Document find = employeesCollection.find(new Document("city",
getCity())).first();

        try {
            if (find != null) {
                Bson newValue = new Document("city", city);
                Bson operation = new Document("$set", newValue);
                employeesCollection.updateOne(find, operation);
            }
        }catch(Exception e){
            e.printStackTrace();
        }
    }

    /**
     * Sets the State of the user
     * @param state, String, contains the State of the user
     */
    public static void setState(String state){
        Document find = employeesCollection.find(new Document("state",
getState())).first();

        try {
            if (find != null) {
```

//Source Code: employeeIO.java

```
      Bson newValue = new Document("state", state);
                Bson operation = new Document("$set", newValue);
                employeesCollection.updateOne(find, operation);
            }
        }catch(Exception e){
            e.printStackTrace();
        }
    }
    //Primary Contact
    /**
     * Sets the primary contact name of the current user
     * @param newEmergencyName, String,  contains the full primary contact name
of the user
     */
    public static void setEmergencyName(String newEmergencyName) {
        Document find = employeesCollection.find(new Document("primary_contact",
getPrimaryContact())).first();

        try {
            if (find != null) {
                Bson newValue = new Document("primary_contact",
newEmergencyName);
                Bson operation = new Document("$set", newValue);
                employeesCollection.updateOne(find, operation);
            }
        }catch(Exception e){
            e.printStackTrace();
        }
    }
    //Source Code: employeeIO.java

    /**
     * Sets the primary contact phone of the current user
```

```
     * @param newEmergencyPhone, String,  contains the full primary contact phone
of the user
     */
    public static void setEmergencyPhone(String newEmergencyPhone) {
        Document find = employeesCollection.find(new Document("primary_phone",
getPrimaryPhone())).first();

        try {
            if (find != null) {
                Bson newValue = new Document("primary_phone", newEmergencyPhone);
                Bson operation = new Document("$set", newValue);
                employeesCollection.updateOne(find, operation);
            }
        }catch(Exception e){
            e.printStackTrace();
        }
    }

    /**
     * Sets the primary contact name of the current user
     * @param newRelation, String,  contains the full primary contact relation of
the user
     */
    public static void setEmergencyRelation(String newRelation) {
        Document find = employeesCollection.find(new Document("primary_relation",
getPrimaryRelation())).first();
```

//Source Code: employeeIO.java

```
 try {
            if (find != null) {
                Bson newValue = new Document("primary_relation", newRelation);
                Bson operation = new Document("$set", newValue);
```

```
                employeesCollection.updateOne(find, operation);
            }
        }catch(Exception e){
            e.printStackTrace();
        }
    }

    /**
     * Sets the primary contact address of the current user
     * @param newAddress, String, contains the full primary contact address of
the user
     */
    public static void setEmergencyAddress(String newAddress) {
        Document find = employeesCollection.find(new Document("primary_address",
getPrimaryAddress())).first();

        try {
            if (find != null) {
                Bson newValue = new Document("primary_address", newAddress);
                Bson operation = new Document("$set", newValue);
                employeesCollection.updateOne(find, operation);
            }
        }catch(Exception e){
```

//Source Code: employeeIO.java

```
 e.printStackTrace();
        }
    }

    /**
     * Sets the primary contact work phone of the current user
     * @param newWorkPhone, String, contains the full primary contact work phone
of the user
```

```
     */
    public static void setEmergencyWorkPhone(String newWorkPhone) {
        Document find = employeesCollection.find(new
Document("primary_work_phone", getPrimaryWorkPhone())).first();

        try {
            if (find != null) {
                Bson newValue = new Document("primary_work_phone", newWorkPhone);
                Bson operation = new Document("$set", newValue);
                employeesCollection.updateOne(find, operation);
            }
        }catch(Exception e){
            e.printStackTrace();
        }
    }

    //Secondary Contact
```

//Source Code: employeeIO.java

```
  /**
     * Sets the secondary contact name of the current user
     * @param newEmergencyName, String, contains the full secondary contact name
of the user
     */
    public static void setSecEmergencyName(String newEmergencyName) {
        Document find = employeesCollection.find(new Document("secondary_name",
getSecondaryContact())).first();

        try {
            if (find != null) {
                Bson newValue = new Document("secondary_name", newEmergencyName);
                Bson operation = new Document("$set", newValue);
```

```
                employeesCollection.updateOne(find, operation);
            }
        }catch(Exception e){
            e.printStackTrace();
        }
    }

    /**
     * Sets the secondary contact phone of the current user
     * @param newEmergencyPhone, String,  contains the full secondary phone number of the user
     */
    public static void setSecEmergencyPhone(String newEmergencyPhone) {
        Document find = employeesCollection.find(new Document("secondary_phone_number", getSecondaryPhone())).first();
```

//Source Code: employeeIO.java

```
try {
            if (find != null) {
                Bson newValue = new Document("secondary_phone_number", newEmergencyPhone);
                Bson operation = new Document("$set", newValue);
                employeesCollection.updateOne(find, operation);
            }
        }catch(Exception e){
            e.printStackTrace();
        }
    }

    /**
     * Sets the secondary contact name of the current user
     * @param newRelation, String, contains the secondary contact relation of the user
```

```
     */
    public static void setSecEmergencyRelation(String newRelation) {
        Document find = employeesCollection.find(new
Document("secondary_relation", getSecondaryRelation())).first();

        try {
            if (find != null) {
                Bson newValue = new Document("secondary_relation", newRelation);
                Bson operation = new Document("$set", newValue);
                employeesCollection.updateOne(find, operation);
            }
        }catch(Exception e){
            e.printStackTrace();
```

`//Source Code: employeeIO.java`

```
 }
    }

    /**
     * Sets the secondary contact address of the current user
     * @param newAddress, String,  contains the secondary contact address of the
user
     */
    public static void setSecEmergencyAddress(String newAddress) {
        Document find = employeesCollection.find(new
Document("secondary_address", getSecondaryAddress())).first();

        try {
            if (find != null) {
                Bson newValue = new Document("secondary_address", newAddress);
                Bson operation = new Document("$set", newValue);
                employeesCollection.updateOne(find, operation);
            }
```

```
        }catch(Exception e){
            e.printStackTrace();
        }
    }

    /**
     * Sets the secondary contact work phone of the current user
     * @param newWorkPhone, String,  contains the secondary contact relation of
the user
     */
    public static void setSecEmergencyWorkPhone(String newWorkPhone) {
        //Source Code: employeeIO.java

Document find = employeesCollection.find(new Document("secondary_work_phone",
getSecondaryWorkPhone())).first();

        try {
            if (find != null) {
                Bson newValue = new Document("secondary_work_phone",
newWorkPhone);
                Bson operation = new Document("$set", newValue);
                employeesCollection.updateOne(find, operation);
            }
        }catch(Exception e){
            e.printStackTrace();
        }
    }

    /**
     * Handles getting all the current users certifications.
     *
     */
    public static void getCertificationList() {
```

```
        Consumer<Document> printConsumer = new Consumer<Document>() {
            @Override
            public void accept(final Document document) {
                String response = document.toJson();
                JsonElement je = JsonParser.parseString(response);
                Gson gson = new GsonBuilder().create();
                employeeResponse responses  = gson.fromJson(je,
employeeResponse.class);
                dataRequested.add(responses);
       //Source Code: employeeIO.java

     }
        };

        employeesCollection.find(eq("user_id", employeeIO.getUserId()))
                .forEach(printConsumer);
    }
}
```

//Source Code: employeeResponse.java

```
package com.application.databaseOps;

import com.google.gson.annotations.Expose;
import com.google.gson.annotations.SerializedName;
import java.util.List;



/**
 * Handles certifications of users
 * Class: CS498 Capstone Project
 * Instructor: Professor Spetka, Scott
 * Date: 02/15/2021
```

```
 * Last Revision: 02/15/2021
 * @author Adis Delanovic
 *
 */
public class employeeResponse {

    @SerializedName("is_supervisor")
    @Expose
    private boolean isSupervisor;

    @SerializedName("user_id")
    @Expose
    private String userId;

    @SerializedName("first_name")
    @Expose
```

//Source Code: employeeResponse.java

```
private String firstName;

    @SerializedName("last_name")
    @Expose
    private String lastName;

    @SerializedName("certifications")
    @Expose
    private List<String> certifications;
    /**
     * Returns if the user is a supervisor
     * @return String, containing the user ID
     */
    public boolean getIsSupervisor(){
```

```
        return isSupervisor;
    }

    /**
     * Returns the user ID of the current user that made the request
     * @return String, containing the user ID
     */
    public String getUserId(){
        return userId;
    }

    /**
     * Returns the first name of the user
     * @return String, containing the user ID
//Source Code: employeeResponse.java

 */
    public String getFirstName(){
        return firstName;
    }

    /**
     * Returns the last name of a user
     * @return String, containing the user ID
     */
    public String getLastName(){
        return lastName;
    }

    /**
     * Returns a list of certificates of the user
     * @return List<String>, containing a list of the users current
certifications
```

```
     */
    public List<String> getCerts() {

        return certifications;
```

//Source Code: passdownsResponse.java

```
package com.application.databaseOps;
import com.google.gson.annotations.Expose;
import com.google.gson.annotations.SerializedName;

public class passdownsResponse {
    @SerializedName("user_id")
    @Expose
    private String userId;

    @SerializedName("supervisor_name")
    @Expose
    private String supervisorName;

    @SerializedName("date")
    @Expose
    private String date;

    @SerializedName("passdown")
    @Expose
    private String passdown;

    /**
     * Returns the first name of the user
     * @return String, containing the user ID
     */
```

```
    public String getUserId(){
        return userId;
    }
```

//Source Code: passdownsResponse.java

```
    /**
     * Returns the first name of the user
     * @return String, containing supervisor that created the passdown
     */
    public String getSupervisorName(){
        return supervisorName;
    }
    /**
     * Returns the first name of the user
     * @return String, containing the date of the passdown
     */
    public String getDate(){
        return date;
    }
    /**
     * Returns the first name of the user
     * @return String, containing the passdown
     */
    public String getPassdown(){
        return passdown;
    }
```

//Source Code: requestResponse.java

```
     package com.application.databaseOps;
     import com.google.gson.annotations.Expose;
     import com.google.gson.annotations.SerializedName;
```

```
/**
 * Defines the Time Off request JSON to GSON Array.
 * Class: CS498 Capstone Project
 * Instructor: Professor Spetka, Scott
 * Date: 02/15/2021
 * Last Revision: 02/15/2021
 * @author Adis Delanovic
 *
 */

public class requestResponse {
    @SerializedName("user_id")
    @Expose
    private String userId;

    @SerializedName("request_start")
    @Expose
    private String requestStart;

    @SerializedName("request_end")
    @Expose
    private String request_end;

    @SerializedName("type")
```

//Source Code: requestResponse.java

```
@Expose
    private String type;

    @SerializedName("start_time")
    @Expose
    private String startTime;

    @SerializedName("end_time")
```

```
        @Expose

    private String endTime;

        @SerializedName("reason")
        @Expose
        private String reason;

        @SerializedName("supervisor_id")
        @Expose
        private String supervisorId;

        @SerializedName("supervisor_name")
        @Expose
        private String supervisorName;

        @SerializedName("status")
        @Expose
        private String status;
```

//Source Code: requestResponse.java

```
       @SerializedName("first_name")
```

//Source Code: requestResponse.java

```
      @Expose
        private String firstName;

        @SerializedName("last_name")
        @Expose
        private String lastName;


    /**
```

```
     * Returns the user ID of the current user that made the request
     * @return String, containing the user ID
     */
        public String getUserId() {
            return userId;
        }

    /**

     * Returns the request end date of the current user that made the request
     * @return String, containing the request end date
     */
        public String getRequest_end() {
            return request_end;
        }

    /**
     * Returns the request start date of the current user that made the request
     * @return String, containing the request start date
     */
        public String getRequest_start(){
            return requestStart;
        }

    /**
```

//Source Code: requestResponse.java

```
     * Returns the request status of the current user that made the request
     * @return String, containing the status of the request (pending/approved)
     */
        public String getRequest_Status(){
            return status;
        }
```

```
/**
 * Returns the reason of the current user that made the request
 * @return String, containing the reason for the request
 */
    public String getReason(){
        return reason;
    }
    /**
     * Returns the reason of the current user that made the request

 * @return String, containing the first name of employee making request
     */
    public String getFirstName(){
        return firstName;
    }
    /**
```

//Source Code: requestResponse.java

```
 * Returns the reason of the current user that made the request
     * @return String, containing the last name of employee making request
     */
    public String getLastName(){
        return lastName;
    }

    /**
     * Returns the reason of the current user that made the request
     * @return String, containing the start time of a request

  */
    public String getStartTime(){
        return startTime;
    }
    /**
```

```
         * Returns the reason of the current user that made the request
         * @return String, containing the end time of a request
         */
        public String getEndTime(){
            return endTime;
        }
        /**
         * Returns the reason of the current user that made the request
         * @return String, containing the status of a request
         */
        public String getType(){
            return type;
```

//Source Code: requestsIO.java

```
package com.application.databaseOps;

import com.application.connection.mongodbStream;
import com.mongodb.client.MongoCollection;
import org.bson.Document;
import org.bson.conversions.Bson;
import org.bson.types.ObjectId;
import com.google.gson.Gson;
import com.google.gson.GsonBuilder;
import com.google.gson.JsonElement;
import com.google.gson.JsonParser;
import java.util.*;
import java.util.function.Consumer;

import static com.mongodb.client.model.Filters.eq;

/**
 * Defines the Employee request operations for requesting time off.
 * Class: CS498 Capstone Project
 * Instructor: Professor Spetka, Scott
 * Date: 02/15/2021
```

```
       * Last Revision: 02/15/2021
       * @author Adis Delanovic
       *
       */


      public class requestsIO {
          public static MongoCollection<Document> requestsCollection =
mongodbStream.database.getCollection("TimeOffRequests");
          public static ArrayList<requestResponse> requests = new ArrayList<>();
```

//Source Code: requestsIO.java

```
      /**
       * Returns the user ID of the current user that made the request
       * @param requestStart date request end date, type of request, start time,
end time and reason. All update certain things in the DB for the request.
       */
          public static void sendNewRequest(String requestStart, String
requestEnd, String typeRequest, String startTime, String endTime, String reason)
{
              try {
                  Document request = new Document("_id", new ObjectId());
                  request.append("user_id", employeeIO.getUserId())
                          .append("first_name", employeeIO.getFirstName())
                          .append("last_name", employeeIO.getLastName())
                          .append("request_start", requestStart)
                          .append("request_end", requestEnd)
                          .append("type", typeRequest)
                          .append("start_time", startTime)
                          .append("end_time", endTime)
                          .append("reason", reason)
                          .append("supervisor_id", employeeIO.getSupervisorId())
                          .append("supervisor_name", employeeIO.getSupervisor())
                          .append("status", "pending");

                  requestsCollection.insertOne(request);
```

```
                //Fixing Bug on Dyanmic Loading
                //mainStage.regEmployees.setValues();

            } catch (Exception e) {
//Source Code: requestsIO.java

               e.printStackTrace();
            }
        }

    /**
     * Returns all requests of a user. Utilizes the google GSON library to
convert from JSON to
     * an ArrayList. The requestResponse ArrayList is populated.
     */
        public static void getAllRequests() {
            Consumer<Document> printConsumer = new Consumer<Document>() {
                @Override
                public void accept(final Document document) {
                    String response = document.toJson();
                    JsonElement je = JsonParser.parseString(response);
                    Gson gson = new GsonBuilder().create();
                    requestResponse responses  = gson.fromJson(je,
requestResponse.class);
                    requests.add(responses);
                }
            };

            requestsCollection.find(eq("user_id", employeeIO.getUserId()))
                    .forEach(printConsumer);
        }

    /**
```

```
       * Returns the request start date of the current user that made the request
       * @return String, containing the request start date.
       */
//Source Code: requestsIO.java


          public static String getRequestStart() {
              Bson filter = eq("user_id", employeeIO.getUserId());

return(String)(Objects.requireNonNull(requestsCollection.find(filter).first())).g
et("request_start");
          }


      /**
       * Returns the request end of the current user that made the request
       * @return String, containing request end.
       */
          public static String getRequestEnd() {
              Bson filter = eq("user_id", employeeIO.getUserId());

return(String)(Objects.requireNonNull(requestsCollection.find(filter).first())).g
et("request_end");
          }


      /**
       * Returns the request status of the current user that made the request
       * @return String, containing the request status (pending/approved)
       */
          public static String getRequestStatus() {
              Bson filter = eq("user_id", employeeIO.getUserId());

return(String)(Objects.requireNonNull(requestsCollection.find(filter).first())).g
et("status");
          }



          /**
```

//Source Code: requestsIO.java

```
          * Returns all requests of a user. Utilizes the google GSON library to
convert from JSON to
          * an ArrayList. The requestResponse ArrayList is populated.
          */
         public static void getRequestsForSupervisor() {
             Consumer<Document> printConsumer = new Consumer<Document>() {
                 @Override
                 public void accept(final Document document) {
                     String response = document.toJson();
                     JsonElement je = JsonParser.parseString(response);
                     Gson gson = new GsonBuilder().create();
                     requestResponse responses  = gson.fromJson(je,
requestResponse.class);
                     requests.add(responses);
                 }
             };

             requestsCollection.find(eq("supervisor_id", employeeIO.getUserId()))
                     .forEach(printConsumer);
         }
```

//Source Code: scheduleIO.java

```
     package com.application.databaseOps;

     import com.application.connection.mongodbStream;
     import com.application.gui.loginStage;
     import com.mongodb.client.MongoCollection;
     import javafx.scene.paint.Color;
     import javafx.scene.text.Text;
     import org.bson.Document;
     import org.bson.conversions.Bson;
     import org.bson.types.ObjectId;
```

```
import java.util.Objects;

import static com.mongodb.client.model.Filters.and;
import static com.mongodb.client.model.Filters.eq;



/**
 * Defines the operations required to get the current schedule of the
employee.
 * Class: CS498 Capstone Project
 * Instructor: Professor Spetka, Scott
 * Date: 02/15/2021
 * Last Revision: 02/15/2021
 * @author Adis Delanovic
 *
 */

public class scheduleIO {
```

//Source Code: scheduleIO.java

```
 public static MongoCollection<Document> scheduleCollection =
mongodbStream.database.getCollection("Schedule");
    public static MongoCollection<Document> employeesCollection =
mongodbStream.database.getCollection("Members");



/**
 * Display the weekly schedule for the employee
 */
    public static void getSchedule(){
        String textStyle = "-fx-font-size: 16px;";
        Text date = new Text();
        Text postMonday = new Text();
```

```
            Text postTuesday = new Text();
            Text postWednesday = new Text();
            Text postThursday = new Text();
            Text postFriday = new Text();
            Text postSaturday = new Text();
            Text postSunday = new Text();

            postMonday.setStyle(textStyle);
            postMonday.setFill(Color.WHITE);
            postFriday.setFill(Color.WHITE);
            postFriday.setStyle(textStyle);
            postSaturday.setStyle(textStyle);
            postSaturday.setFill(Color.WHITE);
            postTuesday.setStyle(textStyle);
            postTuesday.setFill(Color.WHITE);
            postWednesday.setStyle(textStyle);
            postWednesday.setFill(Color.WHITE);
```

//Source Code: scheduleIO.java

```
      postThursday.setStyle(textStyle);
            postThursday.setFill(Color.WHITE);
            postSunday.setStyle(textStyle);
            postSunday.setFill(Color.WHITE);
            date.setFill(Color.WHITE);
            date.setStyle(textStyle);

            postMonday.setText(getPostMonday());
            postTuesday.setText(getPostFriday());
            postWednesday.setText(getPostWednesday());
            postThursday.setText(getPostThursday());
            postFriday.setText(getPostFriday());
            postSaturday.setText(getPostSaturday());
            postSunday.setText(getPostSunday());
            date.setText(getDate());
```

```
            loginStage.mainStage.regEmployees.scheduleGridPane.add(date, 1,0);
            loginStage.mainStage.regEmployees.scheduleGridPane.add(postMonday,
1,2);
            loginStage.mainStage.regEmployees.scheduleGridPane.add(postTuesday,
2,2);

loginStage.mainStage.regEmployees.scheduleGridPane.add(postWednesday, 3,2);
            loginStage.mainStage.regEmployees.scheduleGridPane.add(postThursday,
4,2);
            loginStage.mainStage.regEmployees.scheduleGridPane.add(postFriday,
5,2);
            loginStage.mainStage.regEmployees.scheduleGridPane.add(postSaturday,
6,2);

loginStage.mainStage.regEmployees.scheduleGridPane.add(postSunday,7,2);
         }
```

//Source Code: scheduleIO.java

```
     /**
      * Display the current date range of the schedule.
      * @return String, contains the current Date range of the schedule
      */
         public static String getDate() {
            Bson filter = eq("email_address",
loginStage.activeUser.getUsername());
            return (String)
(Objects.requireNonNull(scheduleCollection.find(filter).first())).get("date");
         }


     /**
      * Display the post for Monday.
      * @return String, contains the current post for Monday.
      */
         public static String getPostMonday() {
            Bson filter = eq("email_address",
loginStage.activeUser.getUsername());
```

```
            return (String)
(Objects.requireNonNull(scheduleCollection.find(filter).first())).get("post_monda
y");
         }


     /**
      * Display the current post for Tuesday.
      * @return String, contains the current post for Tuesday.
      */
         public static String getPostTuesday() {
            Bson filter = eq("email_address",
loginStage.activeUser.getUsername());
```

//Source Code: scheduleIO.java

//Source Code: scheduleIO.java

```
            return (String)
(Objects.requireNonNull(scheduleCollection.find(filter).first())).get("post_tuesd
ay");
         }
     /**
      * Display the current post for Wednesday.
      * @return String, contains the current post for Wednesday.
      */
         public static String getPostWednesday() {
            Bson filter = eq("email_address",
loginStage.activeUser.getUsername());
            return (String)
(Objects.requireNonNull(scheduleCollection.find(filter).first())).get("post_wedne
sday");
         }


     /**
      * Display the current post for Thursday.
      * @return String, contains the current post for Thursday.
      */
         public static String getPostThursday() {
```

```
            Bson filter = eq("email_address",
loginStage.activeUser.getUsername());

            return (String)
(Objects.requireNonNull(scheduleCollection.find(filter).first())).get("post_thurs
day");

         }


     /**

      * Display the current post for Friday.

      * @return String, contains the current post for Friday.

      */

         public static String getPostFriday() {
```

//Source Code: scheduleIO.java

```
      Bson filter = eq("email_address", loginStage.activeUser.getUsername());

            return (String)
(Objects.requireNonNull(scheduleCollection.find(filter).first())).get("post_frida
y");

         }


     /**

      * Display the current post Saturday.

      * @return String, contains the current post for Saturday.

      */

         public static String getPostSaturday() {

            Bson filter = eq("email_address",
loginStage.activeUser.getUsername());

            return (String)
(Objects.requireNonNull(scheduleCollection.find(filter).first())).get("post_satur
day");

         }


     /**

      * Display the current post Sunday

      * @return String, contains the current post for Sunday.

      */

         public static String getPostSunday() {
```

```
            Bson filter = eq("email_address",
loginStage.activeUser.getUsername());

            return (String)
(Objects.requireNonNull(scheduleCollection.find(filter).first())).get("post_sunda
y");

        }


        /**

         * Creates a new schedule for an employee
```

//Source Code: scheduleIO.java

```
      * @param  name, start, end, starTime, endTime, post

          */

         public static void createNewSchedule(String name, String start, String
end, String startTime, String endTime, String post) {

            Document request = new Document("_id", new ObjectId());

            request.append("name", name)

                    .append("post", post)

                    .append("start_date", start)

                    .append("end_date", end)

                    .append("start_time", startTime)

                    .append("end_time", endTime)

                    .append("supervisor_id", employeeIO.getUserId())

                    .append("supervisor_name", employeeIO.getFirstName() + " " +
employeeIO.getLastName());


            scheduleCollection.insertOne(request);


        }

    }
```

//Source Code: launchApp.java

```
package com.application;

import javafx.application.Application;



/**
 * Launches the application and loads the required first stage set up.
 * Class: CS498 Capstone Project
 * Instructor: Professor Spetka, Scott
 * Date: 02/15/2021
 * Last Revision: 02/15/2021
 * @author Adis Delanovic
 *
 */

public class launchApp
{

    public static void main(String[] args)
    {
        Application.launch(initialstageLogin.class, "String arg");
    }
}
```

//Source Code: initialstageLogin.java

```
package com.application;

import com.application.gui.loginStage;
import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
```

```
import javafx.scene.Scene;
import javafx.stage.Stage;
import javafx.scene.image.Image;
import java.io.IOException;

/**
 * Defines how the initial stage is set up and what stages are loaded first. Currently takes you to the login panes.
 * Class: CS498 Capstone Project
 * Instructor: Professor Spetka, Scott
 * Date: 02/15/2021
 * Last Revision: 02/15/2021
 * @author Adis Delanovic
 *
 */

public class initialstageLogin extends Application
{
    @Override
    public void start(Stage primaryStage) throws IOException {
        loginStage logIn = new loginStage();
        try {

            FXMLLoader loadLogin = new FXMLLoader();
```

//Source Code: initialstageLogin.java

```
        loadLogin.setController(logIn);
            Parent root = FXMLLoader.load(getClass().getResource("/stages/login.fxml"));
            primaryStage.getIcons().add(new Image(initialstageLogin.class.getResourceAsStream("/images/af_icon.png")));
            primaryStage.setScene(new Scene(root, 919, 690));
            primaryStage.show();
            primaryStage.setMaxWidth(1000);
            primaryStage.setMaxHeight(1000);
```

```
                primaryStage.setResizable(false);

            }catch(Exception e){
                e.printStackTrace();
            }
        }
    }
```

//Source Code: loginStage.java

```
    package com.application.gui;

    import com.application.connection.mongodbStream;
    import com.application.databaseOps.employeeIO;
    import com.mongodb.MongoCommandException;
    import com.mongodb.MongoSecurityException;
    import com.mongodb.client.MongoCollection;
    import javafx.application.Platform;
    import javafx.event.ActionEvent;
    import javafx.fxml.FXMLLoader;
    import javafx.scene.Parent;
    import javafx.scene.control.*;
    import javafx.scene.layout.AnchorPane;
    import javafx.fxml.FXML;
    import javafx.scene.layout.Pane;
    import javafx.scene.text.Text;
    import org.bson.Document;
    import org.bson.conversions.Bson;

    import java.io.IOException;
    import java.util.Objects;

    import static com.mongodb.client.model.Filters.eq;
```

```
     /**
      * Controller class for the login.fxml page. Controls all login procedures.
      * Class: CS498 Capstone Project
      * Instructor: Professor Spetka, Scott
      * Date: 02/15/2021
```

//Source Code: loginStage.java

```
     * Last Revision: 02/15/2021
      * @author Adis Delanovic
      *
      */
     public class loginStage {
         @FXML public AnchorPane loginPane;
         @FXML public PasswordField passwordField;
         @FXML public Pane loginErrorField;
         @FXML public Text errorReason;
         @FXML public CheckBox showPasswordChecked;
         @FXML public TextField passwordPlainText;
         @FXML private Button loginBtn;
         @FXML private CheckBox supervisorMode;
         public TextField usernameField;
         public static mongodbStream activeUser = new mongodbStream();
         public static parentmainStage mainStage;


     /**
      * Handles all login requests after the login button was clicked. Loads the
appropriate errors or *.fxml stages.
      */
         @FXML
         private void loginClicked() throws IOException {
             try {
                 activeUser.setPassword(passwordField.getText());
                 activeUser.setEmailAddress(usernameField.getText());
                 activeUser.connectDatabase();
```

```
                if(supervisorMode.isSelected() && employeeIO.getisSupervisor())
{
```

//Source Code: loginStage.java

```
                    FXMLLoader mainStageLoader = new
FXMLLoader(getClass().getResource("/stages/parentStage.fxml"));
                    Parent root = mainStageLoader.load();
                    mainStage = mainStageLoader.getController();
                    loginBtn.getScene().setRoot(root);

                    mainStage.viewsLoader = new
FXMLLoader(getClass().getResource("/stages/supervisorMode.fxml"));
                    Parent supervisorRoot = mainStage.viewsLoader.load();
                    mainStage.supervisor =
mainStage.viewsLoader.getController();
                    mainStage.mainView.getChildren().setAll(supervisorRoot);
                    mainStage.supervisor.loadDefaultPane();

                }else if(supervisorMode.isSelected() &&
!employeeIO.getisSupervisor()){
                    FXMLLoader mainStageLoader = new
FXMLLoader(getClass().getResource("/stages/login.fxml"));
                    loginErrorField.setVisible(true);
                    errorReason.setText("Permission Denied! You are not a
supervisor");
                    Parent root = mainStageLoader.load();
                    mainStage.mainView.getChildren().setAll(root);

                }else if(passwordField.getText().equals("")){
                    FXMLLoader mainStageLoader = new
FXMLLoader(getClass().getResource("/stages/login.fxml"));
                    loginErrorField.setVisible(true);
                    errorReason.setText("Password must be entered!");
                    Parent root = mainStageLoader.load();
                    mainStage.mainView.getChildren().setAll(root);
```

//Source Code: loginStage.java

```
            }else if(usernameField.getText().equals("")){
                        FXMLLoader mainStageLoader = new
FXMLLoader(getClass().getResource("/stages/login.fxml"));
                        loginErrorField.setVisible(true);
                        errorReason.setText("Username must be entered!");
                        Parent root = mainStageLoader.load();
                        mainStage.mainView.getChildren().setAll(root);
                    }
                    else {
                        Bson filter = eq("email_address",
loginStage.activeUser.getUsername());
                        String emailAdrs = (String)
(Objects.requireNonNull(employeeIO.employeesCollection.find(filter).first())).get
("email_address");

                        FXMLLoader mainStageLoader = new
FXMLLoader(getClass().getResource("/stages/parentStage.fxml"));
                        Parent root = mainStageLoader.load();
                        mainStage = mainStageLoader.getController();
                        loginBtn.getScene().setRoot(root);
                        mainStage.viewsLoader = new
FXMLLoader(getClass().getResource("/stages/regularEmployee.fxml"));

                        Parent employeesRoot = mainStage.viewsLoader.load();
                        mainStage.regEmployees =
mainStage.viewsLoader.getController();
                        mainStage.mainView.getChildren().setAll(employeesRoot);
                        mainStage.regEmployees.setValues();
                    }

                } catch (MongoCommandException | MongoSecurityException e ) {
                    loginErrorField.setVisible(true);
```

//Source Code: loginStage.java

```
        errorReason.setText("You've entered the wrong password or username!");
                mongodbStream.disconnect_database();


            } catch (Exception loadException){
                Alert alert = new Alert(Alert.AlertType.CONFIRMATION);
                alert.setTitle("An Error Occurred!");
                alert.setHeaderText("Warning");
                alert.setContentText("A restart of the application is required.
An error unexpected error occurred!");
            }
         }


     /**
      * Exits the application and disconnects from the database cleaning up any
active threads..
      */
         public void exitClicked(ActionEvent actionEvent) {
             Platform.exit();
         }


     /**
      * Toggles the show password checkbox. The user is shown the password as
either masked with '*' or plain text.
      */
         public void toggleShowPassword(ActionEvent actionEvent) {
             if (showPasswordChecked.isSelected()) {
                 passwordPlainText.setText(passwordField.getText());
                 passwordPlainText.setVisible(true);
                 passwordField.setVisible(false);
             } else {
```

//Source Code: loginStage.java

```
           passwordField.setText(passwordPlainText.getText());
                 passwordPlainText.setVisible(false);
                 passwordField.setVisible(true);
```

```
            }
        }
    }
```

//Source Code: parentmainStage.java

```
    package com.application.gui;

    import com.application.connection.mongodbStream;
    import javafx.application.Platform;
    import javafx.event.ActionEvent;
    import javafx.fxml.FXML;
    import javafx.fxml.FXMLLoader;
    import javafx.scene.Parent;
    import javafx.scene.control.Button;
    import javafx.scene.input.MouseEvent;
    import javafx.scene.layout.AnchorPane;
    import javafx.scene.layout.Pane;

    import static com.application.gui.loginStage.mainStage;

    /**
     * Controller for the main and primary stage of the application after a
successful login.
     * All other stages are attached to it. This is the parent stage.
     * Class: CS498 Capstone Project
     * Instructor: Professor Spetka, Scott
     * Date: 02/15/2021
     * Last Revision: 02/15/2021
     * @author Adis Delanovic
     *
     */

    public class parentmainStage {
        public FXMLLoader viewsLoader;
```

```
        public regemployeeStage regEmployees;
```

//Source Code: parentmainStage.java

```
        public supervisorModeStage supervisor;
        public loginStage loginView;
        @FXML
        public AnchorPane mainstageAnchorPane;
        @FXML
        public AnchorPane mainView;
        @FXML
        public AnchorPane menuBarAnchorPane;
        @FXML
        public Pane applicationPane;
        @FXML
        public Button exitApplication;
        @FXML
        public Button logOutBtn;
        @FXML
        public Button cancelBtn;


    /**
     * Opens the main menu located in the top right hand corner. Closes if its
already open.
     */
        public void openExitMenu(MouseEvent mouseEvent) {
            if (!applicationPane.isVisible()) {
                applicationPane.setVisible(true);
            } else {
                applicationPane.setVisible(false);
            }
        }


    /**
```

//Source Code: parentmainStage.java

```
     * Exits the application, disconnects from the database and cleans up all
residual threads.
      */
         public void exitApplicationBtnClicked(ActionEvent actionEvent) {
             Platform.exit();
         }

     /**
      * Logs out of the application and displays the login stage again.
      * Calls on the mongodbstream disconnect_database() method to close all
threads.
      */
         public void logOutBtnClicked(ActionEvent actionEvent) {
             try {
                 mongodbStream.disconnect_database();
                 viewsLoader = new
FXMLLoader(getClass().getResource("/stages/login.fxml"));
                 Parent root = viewsLoader.load();
                 loginView = viewsLoader.getController();
                 logOutBtn.getScene().setRoot(root);
             } catch (Exception exception) {
                 exception.printStackTrace();
             }
         }

     }
```

//Source Code: regemployeeStage.java

```
     package com.application.gui;

     import com.application.connection.mongodbStream;
     import com.application.databaseOps.employeeIO;
     import com.application.databaseOps.requestsIO;
```

```
import com.application.databaseOps.scheduleIO;
import com.mongodb.client.MongoCollection;
import javafx.beans.value.ChangeListener;
import javafx.beans.value.ObservableValue;
import javafx.event.ActionEvent;
import javafx.event.Event;
import javafx.scene.Node;
import javafx.scene.control.*;
import javafx.scene.layout.*;
import javafx.fxml.FXML;
import javafx.scene.text.Text;
import javafx.scene.web.WebEngine;
import javafx.scene.web.WebView;
import org.bson.Document;
import org.bson.conversions.Bson;

import java.time.format.DateTimeFormatter;
import java.util.Date;

import static com.mongodb.client.model.Filters.and;
import static com.mongodb.client.model.Filters.eq;



/**
```

//Source Code: regemployeeStage.java

```
 * Controller class for the regular employee login stage. Controls are GUI information contained/displayed in the panes.
 * Also updates, deletes, information as required as well as error handling operations.
 * Class: CS498 Capstone Project
 * Instructor: Professor Spetka, Scott
 * Date: 02/15/2021
```

```
 * Last Revision: 02/15/2021
 * @author Adis Delanovic
 *
 */
public class regemployeeStage {
    @FXML public AnchorPane mainDashboard;

    @FXML public Button editPersonalInfo;
    @FXML public Button savePersonalInfo;
    @FXML public TextField firstName;
    @FXML public TextField lastName;
    @FXML public TextField hireDate;
    @FXML public TextField salary;
    @FXML public TextField nextEvaluation;
    @FXML public TextField shiftAssigned;
    @FXML public TextField supervisor;
    @FXML public TextField streetAddress;
    @FXML public TextField city;
    @FXML public TextField state;
    @FXML public TextField zipcode;
    @FXML public TextField jobTitle;
    @FXML public TextField emergencyName;
    @FXML public TextField relation;
```

//Source Code: regemployeeStage.java

```
    @FXML public TextField emergencyPhone;
    @FXML public TextArea emergencyAddress;
    @FXML public TextField emergencyWorkPhone;
    @FXML public Button editPrimaryContact;
    @FXML public Button savePrimaryContact;
    @FXML public TextField emergencyNameSec;
    @FXML public TextField emergencyPhoneSec;
    @FXML public TextField emergencyRelationSec;
    @FXML public TextField emergencyWorkPhoneSec;
```

```
        @FXML public TextArea emergencyAddressSec;
        @FXML public Button editSecondaryContact;
        @FXML public Button saveSecondaryContact;
        @FXML public TextField vacationTime;
        @FXML public TextField sickTime;
        @FXML public TextField vacationTimeYTD;
        @FXML public TextField sickTimeYTD;
        @FXML public DatePicker requestStartDate;
        @FXML public DatePicker requestEndDate;
        @FXML public Pane approvalNotifications;
        @FXML public ScrollPane resourceScrollPane;
        @FXML public WebView resourceWebView;
        @FXML public WebView bolowebView;
        public WebEngine webEngine;
        @FXML public Button submitRequest;
        @FXML public TextField requestReason;
        @FXML public TextField requestStartTime;
        @FXML public CheckBox sickTimeChecked;
        @FXML public CheckBox vacationTimeChecked;
        @FXML public TextField requestEndTime;
        @FXML public GridPane scheduleGridPane;
```

//Source Code: regemployeeStage.java

```
        @FXML public Text timeapprovalNotification;
        @FXML public Tab schedulePane;
        @FXML public Button closeErrorField;
        @FXML public AnchorPane requestErrorPane;
        @FXML public Text errorTexTField;
        @FXML public AnchorPane requestSuccessPane;
        @FXML public Text requestSuccessField;
        @FXML public Button requestSuccessBtn;
        @FXML public AnchorPane leaveAnchorPane;
        @FXML public Pane certsPane;
```

```
        /**
     * Sets up all the initial values when a user logs in.
     */
        @FXML
        public void setValues(){
            //Personal Information
            firstName.setText(employeeIO.getFirstName());
            firstName.setEditable(false);
            lastName.setText(employeeIO.getLastName());
            lastName.setEditable(false);
            streetAddress.setText(employeeIO.getAddress());
            streetAddress.setEditable(false);
            city.setText(employeeIO.getCity());
            city.setEditable(false);
            state.setText(employeeIO.getState());
            state.setEditable(false);
            zipcode.setText(employeeIO.getZipcode());
            zipcode.setEditable(false);
```

//Source Code: regemployeeStage.java

```
    //Job Related Information
            salary.setText(employeeIO.getSalary().toString());
            salary.setEditable(false);
            hireDate.setText(employeeIO.getHireDate());
            hireDate.setEditable(false);
            jobTitle.setText(employeeIO.getJobTitle());
            jobTitle.setEditable(false);

            //certifications.setText(employeeIO.getCertifications().toString());
            nextEvaluation.setText(employeeIO.getNextEval());
            nextEvaluation.setEditable(false);
            shiftAssigned.setText(employeeIO.getShiftAssigned());
            shiftAssigned.setEditable(false);
```

```
            supervisor.setText(employeeIO.getSupervisor());
            supervisor.setEditable(false);

            //Emergency Contact Primary
            emergencyName.setText(employeeIO.getPrimaryContact());
            emergencyPhone.setEditable(false);
            emergencyPhone.setText(employeeIO.getPrimaryPhone());
            emergencyName.setEditable(false);
            emergencyAddress.setText(employeeIO.getPrimaryAddress());
            emergencyAddress.setEditable(false);
            relation.setText(employeeIO.getPrimaryRelation());
            relation.setEditable(false);
            emergencyWorkPhone.setText(employeeIO.getPrimaryWorkPhone());
            emergencyWorkPhone.setEditable(false);

            //Emergency Contact Secondary
```

//Source Code: regemployeeStage.java

```
            emergencyNameSec.setText(employeeIO.getSecondaryContact());
            emergencyNameSec.setEditable(false);
            emergencyAddressSec.setText(employeeIO.getSecondaryAddress());
            emergencyAddressSec.setEditable(false);
            emergencyRelationSec.setText(employeeIO.getSecondaryRelation());
            emergencyRelationSec.setEditable(false);
            emergencyWorkPhoneSec.setText(employeeIO.getSecondaryWorkPhone());
            emergencyWorkPhoneSec.setEditable(false);
            emergencyPhoneSec.setText(employeeIO.getSecondaryPhone());
            emergencyPhoneSec.setEditable(false);

            //Leave Times
            vacationTime.setText(employeeIO.getVacationTime());
            vacationTime.setEditable(false);
            vacationTimeYTD.setText(employeeIO.getVacationTimeUsed());
            vacationTimeYTD.setEditable(false);
```

```
            sickTimeYTD.setText(employeeIO.getSickTimeUsed());
            sickTimeYTD.setEditable(false);
            sickTime.setText(employeeIO.getSickTime());
            sickTime.setEditable(false);

            //Disables the Schedule pane if its a supervisor
            if(employeeIO.getisSupervisor()){
                schedulePane.setDisable(true);
            }
            setUpCertifications();
            getTimeApproval();
        }
```

//Source Code: regemployeeStage.java

```
     /**
      * Sets up the resources page webView.
      */
         public void displayResourcesRegular(Event event) {
            webEngine = resourceWebView.getEngine();
            webEngine.load("https://www.afsfc.af.mil/");

            //Error Check loading
            webEngine.getLoadWorker().exceptionProperty().addListener(new
ChangeListener<Throwable>() {
                @Override
                public void changed(ObservableValue<? extends Throwable>
observable, Throwable oldValue, Throwable newValue) {
                    System.out.println("Error Occured!");
                }
            });
        }

     /**
      * Sets up the Bolo Page webView.
```

```
       */
          public void displayBoloPage(Event event) {
               webEngine = bolowebView.getEngine();
               webEngine.load("https://www.fbi.gov/wanted/topten");

               //Error Check loading
               webEngine.getLoadWorker().exceptionProperty().addListener(new
ChangeListener<Throwable>() {
                   @Override
                   public void changed(ObservableValue<? extends Throwable>
observable, Throwable oldValue, Throwable newValue) {
```

//Source Code: regemployeeStage.java

```
                       System.out.println("Error Occured!");
                   }
               });
          }

      /**
       * Handles the stage if the edit personal information button is clicked.
       */
          public void editPersonalInfoClicked(ActionEvent actionEvent) {
               savePersonalInfo.setVisible(true);
               editPersonalInfo.setVisible(false);
               firstName.setEditable(true);
               lastName.setEditable(true);
               streetAddress.setEditable(true);
               city.setEditable(true);
               zipcode.setEditable(true);
               state.setEditable(true);

          }

      /**
```

```
       * Handles the stage if the save personal info. button is clicked.
       */
          public void savePersonalInfoClicked(ActionEvent actionEvent) {
               savePersonalInfo.setVisible(false);
               editPersonalInfo.setVisible(true);
               firstName.setEditable(false);
               lastName.setEditable(false);
               streetAddress.setEditable(false);
```

//Source Code: regemployeeStage.java

```
            city.setEditable(false);
               zipcode.setEditable(false);
               state.setEditable(false);

               String newFname = firstName.getText();
               String newLname = lastName.getText();
               String newAddress = streetAddress.getText();
               String newCity = city.getText();
               String newZipcode = zipcode.getText();
               String newState = state.getText();

               employeeIO.setFirstName(newFname);
               employeeIO.setLastName(newLname);
               employeeIO.setAddress(newAddress);
               employeeIO.setCity(newCity);
               employeeIO.setState(newState);
               employeeIO.setZipcode(newZipcode);

          }
      /**
       * Handles the stage if the edit primary contact button is clicked.
       */
          public void editPrimaryContactClicked(ActionEvent actionEvent) {
               emergencyName.setEditable(true);
```

```
            emergencyPhone.setEditable(true);
            relation.setEditable(true);
            emergencyWorkPhone.setEditable(true);
            emergencyAddress.setEditable(true);
            editPrimaryContact.setVisible(false);
            savePrimaryContact.setVisible(true);
```

//Source Code: regemployeeStage.java

```
    }

    /**
     * Handles the stage if the save primary contact button is clicked.
     */
        public void savePrimaryContactClicked(ActionEvent actionEvent) {
            emergencyName.setEditable(false);
            emergencyPhone.setEditable(false);
            relation.setEditable(false);
            emergencyWorkPhone.setEditable(false);
            emergencyAddress.setEditable(false);
            editPrimaryContact.setVisible(true);
            savePrimaryContact.setVisible(false);

            String newEmergencyName = emergencyName.getText();
            String newEmergencyPhone = emergencyPhone.getText();
            String newRelation = relation.getText();
            String newAddress = emergencyAddress.getText();
            String newWorkPhone = emergencyWorkPhone.getText();

            employeeIO.setEmergencyName(newEmergencyName);
            employeeIO.setEmergencyPhone(newEmergencyPhone);
            employeeIO.setEmergencyRelation(newRelation);
            employeeIO.setEmergencyAddress(newAddress);
            employeeIO.setEmergencyWorkPhone(newWorkPhone);
        }
```

```
    /**
     * Handles the stage if the edit secondary contact button is clicked.
     */
```

//Source Code: regemployeeStage.java

```
        public void editSecondaryContactClicked(ActionEvent actionEvent) {
            emergencyNameSec.setEditable(true);
            emergencyPhoneSec.setEditable(true);
            emergencyRelationSec.setEditable(true);
            emergencyWorkPhoneSec.setEditable(true);
            emergencyAddressSec.setEditable(true);
            editSecondaryContact.setVisible(false);
            saveSecondaryContact.setVisible(true);
        }

    /**
     * Handles the stage if the save secondary contact button is clicked.
     */
        public void saveSecondaryContactClicked(ActionEvent actionEvent) {
            emergencyNameSec.setEditable(false);
            emergencyPhoneSec.setEditable(false);
            emergencyRelationSec.setEditable(false);
            emergencyWorkPhoneSec.setEditable(false);
            emergencyAddressSec.setEditable(false);
            editSecondaryContact.setVisible(true);
            saveSecondaryContact.setVisible(false);

            String newEmergencyName = emergencyNameSec.getText();
            String newEmergencyPhone = emergencyPhoneSec.getText();
            String newRelation = emergencyRelationSec.getText();
            String newAddress = emergencyAddressSec.getText();
            String newWorkPhone = emergencyWorkPhoneSec.getText();
```

```
            employeeIO.setSecEmergencyName(newEmergencyName);
            employeeIO.setSecEmergencyPhone(newEmergencyPhone);
```

//Source Code: regemployeeStage.java

```
            employeeIO.setSecEmergencyRelation(newRelation);
            employeeIO.setSecEmergencyAddress(newAddress);
            employeeIO.setSecEmergencyWorkPhone(newWorkPhone);

            String monday = requestsIO.getRequestStart();
            System.out.println(monday);

        }

    /**
     * Handles the stage if the submit request for time off button is clicked.
     */
        public void submitRequestClicked(ActionEvent actionEvent) {

            try {
                String requestStart =
requestStartDate.getValue().format(DateTimeFormatter.ofPattern("yyyy-MM-dd"));
                String requestEnd =
requestEndDate.getValue().format(DateTimeFormatter.ofPattern("yyyy-MM-dd"));
                String typeRequest;
                String startTime = requestStartTime.getText();
                String endTime = requestEndTime.getText();
                String reason = requestReason.getText();
                int startInt = Integer.parseInt(String.valueOf(startTime));
                int endInt = Integer.parseInt(String.valueOf(endTime));

                if (vacationTimeChecked.isSelected() &&
!sickTimeChecked.isSelected()) {
                    typeRequest = "Vacation";
```

//Source Code: regemployeeStage.java

```
                requestsIO.sendNewRequest(requestStart, requestEnd, typeRequest,
startTime, endTime, reason);

                    requestSuccessPane.setVisible(true);

                    requestSuccessField.setText("Your request was successfully
submitted!");


                } else if (sickTimeChecked.isSelected() &&
!vacationTimeChecked.isSelected()) {

                    typeRequest = "Sick";

                    requestsIO.sendNewRequest(requestStart, requestEnd,
typeRequest, startTime, endTime, reason);

                    requestSuccessPane.setVisible(true);

                    requestSuccessField.setText("Your request was successfully
submitted!");


                } else if (sickTimeChecked.isSelected() &&
vacationTimeChecked.isSelected()) {

                    requestErrorPane.setVisible(true);

                    errorTexTField.setText("You can't select both sick and
vacation time!");


                } else
if(requestStartDate.getValue().isAfter(requestEndDate.getValue())) {

                    requestErrorPane.setVisible(true);

                    errorTexTField.setText("The start date must be before the
end date!");


                } else if(startInt > endInt) {

                    requestErrorPane.setVisible(true);

                    errorTexTField.setText("The start time must be before the
end time! Use 0000 2359 Format");


                }else{
```

//Source Code: regemployeeStage.java

```
            typeRequest = "Unknown";

                    requestsIO.sendNewRequest(requestStart, requestEnd,
typeRequest, startTime, endTime, reason);
```

```
                    requestSuccessPane.setVisible(true);
                    requestSuccessField.setText("Your request was successfully
submitted!");
                }

            }catch(Exception e){
                requestErrorPane.setVisible(true);
                errorTexTField.setText("Ensure all fields are properly filled
out!");
            }

        }

    /**
     * Gets the schedule for the current user logged in.
     */
        public void getScheduleForEmployee(Event event) {
            scheduleIO.getSchedule();
        }

    /**
     * Displays all the current user time off requests in a GridPane. Allows for
the ability to cancel requests.
     */
        public void getTimeApproval(){

            ColumnConstraints[] columnConstraintses = new ColumnConstraints[10];
            RowConstraints[] rowConstraintses = new RowConstraints[10];
```

//Source Code: regemployeeStage.java

```
            for (int i = 0 ; i < 10 ; i++) {
                columnConstraintses[i] = new ColumnConstraints(80);
                rowConstraintses[i] = new RowConstraints(50);
            }
```

```
            int i,j = 0,k = 0;

            requestsIO.getAllRequests();
            GridPane gridPane = new GridPane();

            gridPane.getColumnConstraints().addAll(columnConstraintses);
            gridPane.getRowConstraints().addAll(rowConstraintses);

            for (i = 0; i < requestsIO.requests.size(); i++) {
                Button cancelBtn = new Button();
                cancelBtn.setText("Cancel");
                cancelBtn.setMinHeight(23);
                cancelBtn.setMinWidth(75);
                cancelBtn.setStyle("-fx-background-color:  red; -fx-text-fill:
white");
                Text requestStart = new Text();
                Text requestEnd = new Text();
                Text status = new Text();


requestStart.setText(requestsIO.requests.get(i).getRequest_start());
                requestEnd.setText(requestsIO.requests.get(i).getRequest_end());
                status.setText(requestsIO.requests.get(i).getRequest_Status());
```

//Source Code: regemployeeStage.java

```
                gridPane.add(requestStart, k,j);
                gridPane.add(requestEnd, k+1, j);
                gridPane.add(status, k+2, j);
                gridPane.add(cancelBtn, k+3,j);

                //Handles the Cancel Button
                cancelBtn.setOnMouseClicked(f -> {
                    Node source = (Node) f.getSource();
```

```
                    int row = GridPane.getRowIndex(source);
                    MongoCollection<Document> requestsCollection =
mongodbStream.database.getCollection("TimeOffRequests");
                    Bson filter = and(eq("user_id", employeeIO.getUserId()),
eq("request_start",requestsIO.requests.get(row).getRequest_start()));
                    requestsCollection.deleteOne(filter);
                    cancelBtn.setText("Removed");
                    cancelBtn.setStyle("-fx-background-color:   #f6a91c; -fx-
text-fill: black");
                });

                j++;
            }
            approvalNotifications.getChildren().addAll(gridPane);

        }

        /**
         * Displays all the current user Certifications
         */
        public void setUpCertifications(){
            VBox certs = new VBox();
```

//Source Code: regemployeeStage.java

```
        employeeIO.getCertificationList();

            for(int i = 0; i < employeeIO.dataRequested.size(); i++){
                for(int j = 0; j <
employeeIO.dataRequested.get(i).getCerts().size(); j++) {
                    Text cert = new Text();
                    cert.setStyle("-fx-font-size: 12px; -fx-font-weight: bold");
                    cert.setText("[" + (j+1) + "] " +
employeeIO.dataRequested.get(i).getCerts().get(j) + "\n");
                    certs.getChildren().add(cert);
                }
            }
```

```
          certsPane.getChildren().addAll(certs);
          }



      /**
       * Handles the stage if there is an error with the request for time off.
       */
          public void closeErrorFieldClicked(ActionEvent actionEvent) {
              requestErrorPane.setVisible(false);


          }


      /**
       * Handles the stage if the request time off was successful.
       */
          public void requestBtnConfirmationClicked(ActionEvent actionEvent) {
              requestSuccessPane.setVisible(false);
```

//Source Code: regemployeeStage.java

```
       requestReason.setText("");
              requestStartDate.getEditor().clear();
              requestEndDate.getEditor().clear();
              requestStartTime.setText("");
              requestEndTime.setText("");


              if(sickTimeChecked.isSelected() ||
vacationTimeChecked.isSelected()){
                  sickTimeChecked.setSelected(false);
                  vacationTimeChecked.setSelected(false);
              }
          }
      }
```

//Source Code: supervisorModestage.java

```
package com.application.gui;

import com.application.connection.mongodbStream;
import com.application.databaseOps.*;
import com.google.gson.Gson;
import com.google.gson.GsonBuilder;
import com.google.gson.JsonElement;
import com.google.gson.JsonParser;
import com.mongodb.client.MongoCollection;
import javafx.beans.value.ChangeListener;
import javafx.beans.value.ObservableValue;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.geometry.Insets;
import javafx.scene.Node;
import javafx.scene.control.*;
import javafx.scene.layout.*;
import javafx.scene.text.Text;
import javafx.scene.web.WebEngine;
import javafx.scene.web.WebView;
import org.bson.Document;
import org.bson.conversions.Bson;
import org.bson.types.ObjectId;
import java.io.IOException;
import java.text.SimpleDateFormat;
import java.time.format.DateTimeFormatter;
import java.util.ArrayList;
import java.util.Date;
import java.util.function.Consumer;
```

//Source Code: supervisorModestage.java

```
import static com.application.gui.loginStage.mainStage;
import static com.mongodb.client.model.Filters.*;



/**
 * Controller for all operations on the supervisor stage. All updates, deletes and error handling
 * for a supervisor that is logged in are controlled by this class.
 * Class: CS498 Capstone Project
 * Instructor: Professor Spetka, Scott
 * Date: 02/15/2021
 * Last Revision: 02/15/2021
 * @author Adis Delanovic
 *
 */
public class supervisorModeStage {
    @FXML
    public Button resourcesBtn;
    @FXML
    public Button passdownsBtn;
    @FXML
    public Button requestsOffBtn;
    @FXML
    public Button schedulingBtn;
    @FXML
    public AnchorPane supervisorMainAnchor;
    @FXML
    public AnchorPane resourcesMainPane;
    @FXML
    public WebView supervisorWebView;
```

//Source Code: supervisorModestage.java

```
    @FXML
        public WebEngine webEngine;
        @FXML
        public AnchorPane passdownsAnchor;
        @FXML
        public AnchorPane requestsAnchor;
        @FXML
        public AnchorPane schedulingAnchor;
        @FXML
        public Pane timeOffRequests;
        @FXML
        public Button submitPassdowns;
        @FXML
        public Pane viewPassDownsPane;
        @FXML
        public Button viewPassdowns;
        @FXML
        public ScrollPane viewPassDownsScrollPane;
        @FXML
        public Button closePassdownsBtn;
        @FXML
        public TextArea passdownsTextArea;
        @FXML
        public AnchorPane requestSuccessPane;
        @FXML
        public Text requestSuccessField;
        @FXML
        public Button requestSuccessBtn;
        @FXML
        public Button printPassdowns;
```

//Source Code: supervisorModestage.java

```
       @FXML
          public Button savePassdowns;
          @FXML
          public Pane currentRoster;
          @FXML
          public Pane schedulePane;
          @FXML
          public Pane createaSchedulePane;
          @FXML
          public VBox rosterVbox;
          @FXML public ComboBox comboBox;
          @FXML public Button submitSchedule;
          @FXML public DatePicker dateScheduleFrom;
          @FXML public TextField employeeName;
          @FXML public Pane schedulelSuccess;
          @FXML public Text schedulelSuccessField;
          @FXML public Button schedulelSuccessBtn;
          @FXML  public DatePicker dateScheduleTo;
          @FXML public ComboBox employeeComboBox;
          @FXML public TextField scheduleStartTime;
          @FXML public TextField scheduleEndTime;
          @FXML public Pane schedulError;
          @FXML public Text scheduleErrorField;
          @FXML public Button schedulelErrorBtn;


          public GridPane gridPane = new GridPane();
          public VBox vbox = new VBox();
          public static ArrayList<passdownsResponse> requestPassdowns = new
ArrayList<passdownsResponse>();
```

//Source Code: supervisorModestage.java

```
      public static ArrayList<employeeResponse> employeeRoster = new
ArrayList<employeeResponse>();



        /**
         * Handles the stage if the display schedule button is clicked.
         */
        public void displaySchedulePane(ActionEvent actionEvent) {
            resourcesMainPane.setVisible(false);
            passdownsAnchor.setVisible(false);
            schedulingAnchor.setVisible(true);
            requestsAnchor.setVisible(false);

            this.vbox.getChildren().clear();
            getRoster();
        }

        /**
         * Handles the stage if request time off button is clicked.
         */
        public void requestTimeOffPane(ActionEvent actionEvent) {
            resourcesMainPane.setVisible(false);
            passdownsAnchor.setVisible(false);
            schedulingAnchor.setVisible(false);
            requestsAnchor.setVisible(true);
            getTimeApproval();
        }

        /**
         * Handles the stage if the daily passdowns button is clicked.
```

//Source Code: supervisorModestage.java

```
         */
         public void displayDailyPassdownsPane(ActionEvent actionEvent) {
             passdownsAnchor.setVisible(true);
             resourcesMainPane.setVisible(false);
             schedulingAnchor.setVisible(false);
             requestsAnchor.setVisible(false);
         }


         /**
          * Handles the stage for the Resources Pane. Currently takes a
supervisor to resources for forms and information.
          */
         public void displayResourcesPane(ActionEvent actionEvent) {
             resourcesMainPane.setVisible(true);
             passdownsAnchor.setVisible(false);
             schedulingAnchor.setVisible(false);
             requestsAnchor.setVisible(false);
             webEngine = supervisorWebView.getEngine();
             webEngine.load("https://www.e-publishing.af.mil/Product-Index/");



             webEngine.getLoadWorker().exceptionProperty().addListener(new
ChangeListener<Throwable>() {
                 @Override
                 public void changed(ObservableValue<? extends Throwable>
observable, Throwable oldValue, Throwable newValue) {
                     System.out.println("Error Occured!");
                 }
             });
         }
```

//Source Code: supervisorModestage.java

```
    /**
        * Handles getting all the day off requests created by an employee.
        */

       public void getTimeApproval() {
           ColumnConstraints[] columnConstraintses = new ColumnConstraints[10];
           RowConstraints[] rowConstraintses = new RowConstraints[10];

           for (int i = 0; i < 10; i++) {
               columnConstraintses[i] = new ColumnConstraints(90);
               rowConstraintses[i] = new RowConstraints(50);
           }

           int i, j = 0, k = 0;

           requestsIO.getRequestsForSupervisor();

           gridPane.getColumnConstraints().addAll(columnConstraintses);
           gridPane.getRowConstraints().addAll(rowConstraintses);

           for (i = 0; i < requestsIO.requests.size(); i++) {
               Button approveBtn = new Button();
               Button denyBtn = new Button();

               Text requestName = new Text();
               Text requestStart = new Text();
               Text requestEnd = new Text();
```

//Source Code: supervisorModestage.java

```
    Text status = new Text();
               Text requestReason = new Text();
               Text starTime = new Text();
```

```
                Text endTime = new Text();
                Text typeRequest = new Text();


                String nameofEmployee =
requestsIO.requests.get(i).getFirstName() + " " +
requestsIO.requests.get(i).getLastName();
                requestName.setText(nameofEmployee);

requestStart.setText(requestsIO.requests.get(i).getRequest_start());
                requestEnd.setText(requestsIO.requests.get(i).getRequest_end());
                status.setText(requestsIO.requests.get(i).getRequest_Status());
                requestReason.setText(requestsIO.requests.get(i).getReason());
                starTime.setText(requestsIO.requests.get(i).getStartTime());
                endTime.setText(requestsIO.requests.get(i).getEndTime());
                typeRequest.setText(requestsIO.requests.get(i).getType());


                if
(requestsIO.requests.get(i).getRequest_Status().equals("approved")) {
                    approveBtn.setMinHeight(23);
                    approveBtn.setMinWidth(75);
                    approveBtn.setText("Approved");
                    approveBtn.setStyle("-fx-background-color:   #f6a91c; -fx-
text-fill: black");


                    denyBtn.setText("Deny");
                    denyBtn.setMinHeight(23);
```

//Source Code: supervisorModestage.java

```
            denyBtn.setMinWidth(75);
                    denyBtn.setStyle("-fx-background-color:  red; -fx-text-fill:
white");
                } else if
(requestsIO.requests.get(i).getRequest_Status().equals("denied")) {
                    approveBtn.setText("Approve");
                    approveBtn.setMinHeight(23);
                    approveBtn.setMinWidth(75);
```

```
                    approveBtn.setStyle("-fx-background-color:  blue; -fx-text-
fill: white");


                    denyBtn.setText("Denied");
                    denyBtn.setMinHeight(23);
                    denyBtn.setMinWidth(75);
                    denyBtn.setStyle("-fx-background-color:  #f6a91c; -fx-text-
fill: black");
                } else {
                    approveBtn.setText("Approve");
                    approveBtn.setMinHeight(23);
                    approveBtn.setMinWidth(75);
                    approveBtn.setStyle("-fx-background-color:  blue; -fx-text-
fill: white");



                    denyBtn.setText("Deny");
                    denyBtn.setMinHeight(23);
                    denyBtn.setMinWidth(75);
                    denyBtn.setStyle("-fx-background-color:  red; -fx-text-fill:
white");
                }


                gridPane.add(requestName, k, j);
```

//Source Code: supervisorModestage.java

```
     gridPane.add(requestStart, k + 1, j);
                gridPane.add(requestEnd, k + 2, j);
                gridPane.add(starTime, k + 3, j);
                gridPane.add(endTime, k + 4, j);
                gridPane.add(requestReason, k + 5, j);
                gridPane.add(typeRequest, k + 6, j);
                gridPane.add(status, k + 7, j);
                gridPane.add(approveBtn, k + 8, j);
                gridPane.add(denyBtn, k + 9, j);
```

```
                //Handles the approve button
                approveBtn.setOnMouseClicked(f -> {
                    Node source = (Node) f.getSource();
                    int row = GridPane.getRowIndex(source);
                    MongoCollection<Document> requestsCollection =
mongodbStream.database.getCollection("TimeOffRequests");
                    Bson filter = and(eq("supervisor_id",
employeeIO.getUserId()), eq("request_start",
requestsIO.requests.get(row).getRequest_start()));

                    Bson newValue = new Document("status", "approved");
                    Bson operation = new Document("$set", newValue);

                    requestsCollection.updateOne(filter, operation);
                    approveBtn.setText("Approved");
                    approveBtn.setStyle("-fx-background-color:   #f6a91c; -fx-
text-fill: black");
                });

                //Handles the cancel button
                denyBtn.setOnMouseClicked(f -> {
```

//Source Code: supervisorModestage.java

```
      Node source = (Node) f.getSource();
                    int row = GridPane.getRowIndex(source);
                    MongoCollection<Document> requestsCollection =
mongodbStream.database.getCollection("TimeOffRequests");
                    Bson filter = and(eq("supervisor_id",
employeeIO.getUserId()), eq("request_start",
requestsIO.requests.get(row).getRequest_start()));

                    Bson newValue = new Document("status", "denied");
                    Bson operation = new Document("$set", newValue);

                    requestsCollection.updateOne(filter, operation);
```

```
                    denyBtn.setText("Denied");
                    denyBtn.setStyle("-fx-background-color:   #f6a91c; -fx-text-
fill: black");
                });

                j++;
            }
            mainStage.supervisor.timeOffRequests.getChildren().clear();

mainStage.supervisor.timeOffRequests.getChildren().addAll(mainStage.supervisor.gr
idPane);
        }

        /**
         * Handles displaying the default pane for supervisor mode
         */

        public void loadDefaultPane() {
            resourcesMainPane.setVisible(true);
            passdownsAnchor.setVisible(false);
```

//Source Code: supervisorModestage.java

```
           schedulingAnchor.setVisible(false);
            requestsAnchor.setVisible(false);
            webEngine = supervisorWebView.getEngine();
            webEngine.load("https://www.e-publishing.af.mil/Product-Index/");



            webEngine.getLoadWorker().exceptionProperty().addListener(new
ChangeListener<Throwable>() {
                @Override
                public void changed(ObservableValue<? extends Throwable>
observable, Throwable oldValue, Throwable newValue) {
                    System.out.println("Error Occured!");
                }
            });
```

```
        }

        /**
         * Handles submitting a new passdown to the database for storage.
         */
        public void submitPassdownsClicked(ActionEvent actionEvent) {
            MongoCollection<Document> passdowns =
mongodbStream.database.getCollection("Passdowns");
            try {
                SimpleDateFormat formatter = new SimpleDateFormat("dd/MM/yyyy
HH:mm:ss");
                Date date = new Date();
                String currentDate = formatter.format(date);
                Document request = new Document("_id", new ObjectId());
                request.append("user_id", employeeIO.getUserId())
                        .append("supervisor_name", employeeIO.getLastName() + "
" + employeeIO.getFirstName())
```

//Source Code: supervisorModestage.java

```
            .append("date", currentDate)
                        .append("passdown", passdownsTextArea.getText());

                passdowns.insertOne(request);

                requestSuccessPane.setVisible(true);

            } catch (Exception e) {
                e.printStackTrace();
            }
        }

        /**
         * Handles displaying all the passdowns in the database
         */
        public void viewPassdownsClicked(ActionEvent actionEvent) {
```

```
            viewPassDownsPane.setVisible(true);
            MongoCollection<Document> passdowns =
mongodbStream.database.getCollection("Passdowns");

            ColumnConstraints[] columnConstraintses = new ColumnConstraints[10];
            RowConstraints[] rowConstraintses = new RowConstraints[10];

            for (int i = 0; i < 10; i++) {
                columnConstraintses[i] = new ColumnConstraints(120);
                rowConstraintses[i] = new RowConstraints(50);
            }

            int i, j = 0, k = 0;
```

//Source Code: supervisorModestage.java

```
            requestsIO.getRequestsForSupervisor();
            GridPane gridPane = new GridPane();

            gridPane.getColumnConstraints().addAll(columnConstraintses);
            gridPane.getRowConstraints().addAll(rowConstraintses);

            Consumer<Document> printConsumer = new Consumer<Document>() {
                @Override
                public void accept(final Document document) {
                    String response = document.toJson();
                    JsonElement je = JsonParser.parseString(response);
                    Gson gson = new GsonBuilder().create();
                    passdownsResponse responses = gson.fromJson(je,
passdownsResponse.class);
                    requestPassdowns.add(responses);
                }
            };

            passdowns.find()
```

```
                    .forEach(printConsumer);

            for (i = 0; i < requestPassdowns.size(); i++) {
                Text supervisorName = new Text();
                Text dateOfPassdown = new Text();
                Text passDown = new Text();


supervisorName.setText(requestPassdowns.get(i).getSupervisorName());
                dateOfPassdown.setText(requestPassdowns.get(i).getDate());
                passDown.setText(requestPassdowns.get(i).getPassdown());
```

//Source Code: supervisorModestage.java

```
        gridPane.add(supervisorName, k, j);
                gridPane.add(dateOfPassdown, k + 1, j);
                gridPane.add(passDown, k + 2, j);

                j++;
            }

            viewPassDownsScrollPane.setContent(gridPane);
        }

        /**
         * Handles closing the passdowns and viewing a new scene
         */
        public void closePassdownsClicked(ActionEvent actionEvent) {
            viewPassDownsPane.setVisible(false);
        }

        /**
         * Handles confirming that a new passdown was submitted
         */
        public void requestBtnConfirmationClicked(ActionEvent actionEvent) {
```

```
            requestSuccessPane.setVisible(false);
            passdownsTextArea.setText("");
        }

        /**
         * Handles printing the passdowns to a local printer
         */
        public void printPassdownsClicked(ActionEvent actionEvent) {
```

//Source Code: supervisorModestage.java

```
        }

        /**
         * Handles saving the passdowns for local storage
         */
        public void savePassdownsBtnClicked(ActionEvent actionEvent) throws
IOException {

        }

        /**
         * Handles getting the current employees roster for a supervisor
         */
        public void getRoster() {

            MongoCollection<Document> employeesCollection =
mongodbStream.database.getCollection("Members");
            Consumer<Document> printConsumer = new Consumer<Document>() {
                @Override
                public void accept(final Document document) {
                    String response = document.toJson();
                    JsonElement je = JsonParser.parseString(response);
                    Gson gson = new GsonBuilder().create();
                    employeeResponse responses = gson.fromJson(je,
employeeResponse.class);
```

```
                    employeeRoster.add(responses);
                }
            };

            employeesCollection.find(eq("supervisor_id",
employeeIO.getUserId()))
```

//Source Code: supervisorModestage.java

```
       .forEach(printConsumer);

            setUpRoster();

            comboBox.getItems().removeAll(comboBox.getItems());
            employeeComboBox.getItems().removeAll(employeeComboBox.getItems());
            comboBox.getItems().addAll("Dispatch Operations", "Logistics",
"Patrol 1", "Patrol 2", "Patrol 3", "Perimeter");
            for (com.application.databaseOps.employeeResponse employeeResponse :
employeeRoster) {
                employeeComboBox.getItems().add(employeeResponse.getFirstName()
+ " " + employeeResponse.getLastName());
            }
      }

          /**
           * Handles setting up the roster for the current supervisor logged in
           */
          public void setUpRoster() {
              mainStage.supervisor.currentRoster.getChildren().clear();
              vbox.setPadding(new Insets(10, 10, 10, 10));
              vbox.setSpacing(10);
              comboBox.getItems().removeAll(comboBox.getItems());
              for (com.application.databaseOps.employeeResponse employeeResponse :
employeeRoster) {
                  Text employee = new Text();
                  employee.setStyle("-fx-font-size: 12px; -fx-font-weight: bold");
```

```
                employee.setText(employeeResponse.getFirstName() + " " +
employeeResponse.getLastName());
```

//Source Code: supervisorModestage.java

```
       vbox.getChildren().add(employee);
              }
                  mainStage.supervisor.currentRoster.getChildren().addAll(vbox);
          }

          /**
           * Handles Creating a new schedule for an employee
           */
          public void submitScheduleUpdate() {

              String startDate =
dateScheduleFrom.getValue().format(DateTimeFormatter.ofPattern("yyyy-MM-dd"));
              String endDate =
dateScheduleTo.getValue().format(DateTimeFormatter.ofPattern("yyyy-MM-dd"));
              String post = comboBox.getValue().toString();
              String startTime = scheduleStartTime.getText();
              String endTime = scheduleEndTime.getText();
              String employeeName = employeeComboBox.getValue().toString();
              int startInt = Integer.parseInt(String.valueOf(startTime));
              int endInt = Integer.parseInt(String.valueOf(endTime));

              if(startInt > endInt) {
                  schedulError.setVisible(true);
                  scheduleErrorField.setText("The start time must be before the
end time! Use 0000 2359 Format");
              }else{
                  scheduleIO.createNewSchedule(employeeName, startDate, endDate,
startTime, endTime, post);
                  schedulelSuccess.setVisible(true);
              }
```

//Source Code: supervisorModestage.java

```
    }

    /**
     * Handles creating a new schedule for an employee success
     */
    public void submitRequestSuccessBtn(ActionEvent actionEvent) {
        schedulelSuccess.setVisible(false);
    }

    /**
     * Handles creating a new schedule for an employee fail
     */
    public void scheduleErrorBtn(ActionEvent actionEvent) {
        schedulError.setVisible(false);
    }
}
```

//Source Code: login.fxml

```
<?xml version="1.0" encoding="UTF-8"?>


<?import javafx.geometry.Insets?>
<?import javafx.scene.Cursor?>
<?import javafx.scene.control.Button?>
<?import javafx.scene.control.CheckBox?>
<?import javafx.scene.control.PasswordField?>
<?import javafx.scene.control.TextField?>
<?import javafx.scene.image.Image?>
<?import javafx.scene.image.ImageView?>
<?import javafx.scene.layout.AnchorPane?>
<?import javafx.scene.layout.Pane?>
<?import javafx.scene.text.Text?>


<AnchorPane id="loginPane" fx:id="loginPane" cache="true" prefHeight="690.0"
prefWidth="919.0" style="-fx-background-color: ffffff; -fx-border-color: black; -
fx-border-width: 2px;" stylesheets="@../styles/loginStyle.css"
xmlns="http://javafx.com/javafx/15.0.1" xmlns:fx="http://javafx.com/fxml/1"
fx:controller="com.application.gui.loginStage">
   <children>
      <Button fx:id="loginBtn" layoutX="658.0" layoutY="399.0"
mnemonicParsing="false" onAction="#loginClicked" prefHeight="41.0"
prefWidth="244.0" style="-fx-background-color: #003087;" text="Login"
textFill="WHITE" />
      <ImageView fitHeight="603.0" fitWidth="603.0" layoutX="30.0"
layoutY="40.0" pickOnBounds="true" preserveRatio="true" style="-fx-effect:
dropshadow(three-pass-box, rgba(2,2,2,2), 2,2,2,2);">
         <image>
            <Image url="@../images/login.jpg" />
         </image>
      </ImageView>
      <Text id="mainHeader" fill="WHITE" layoutX="169.0" layoutY="95.0"
strokeType="OUTSIDE" strokeWidth="0.0" text="U.S Air Force
```

//Source Code: login.fxml

```
Security Forces Management System" textAlignment="CENTER"
wrappingWidth="324.08984375" />

           <Button fx:id="exitBtn" layoutX="658.0" layoutY="450.0"
mnemonicParsing="false" onAction="#exitClicked" prefHeight="41.0"
prefWidth="244.0" style="-fx-background-color: #003087;" text="Exit"
textFill="WHITE" />

             <TextField id="usernameField" fx:id="usernameField"
alignment="CENTER" blendMode="DARKEN" layoutX="658.0" layoutY="289.0"
prefHeight="35.0" prefWidth="244.0" promptText="Username" snapToPixel="false"
style="-fx-border-color: black; -fx-border-radius: 5px;">

              <cursor>

                 <Cursor fx:constant="TEXT" />

              </cursor>

              <opaqueInsets>

                 <Insets bottom="5.0" left="5.0" right="5.0" top="5.0" />

              </opaqueInsets></TextField>

             <PasswordField id="passwordField" fx:id="passwordField"
alignment="CENTER" cache="true" cacheShape="false" centerShape="false"
focusTraversable="false" layoutX="658.0" layoutY="332.0" prefHeight="35.0"
prefWidth="244.0" promptText="Password" scaleShape="false" style="-fx-border-
color: black; -fx-border-radius: 5px;">

              <cursor>

                 <Cursor fx:constant="TEXT" />

              </cursor></PasswordField>

           <TextField id="usernameField" fx:id="passwordPlainText"
alignment="CENTER" blendMode="DARKEN" layoutX="658.0" layoutY="332.0"
prefHeight="35.0" prefWidth="244.0" promptText="Password" snapToPixel="false"
style="-fx-border-color: black; -fx-border-radius: 5px;" visible="false">

              <cursor>

                 <Cursor fx:constant="TEXT" />

              </cursor>

              <opaqueInsets>

                 <Insets bottom="5.0" left="5.0" right="5.0" top="5.0" />

              </opaqueInsets>
```

//Source Code: login.fxml

```
        </TextField>
```

```
            <CheckBox fx:id="supervisorMode" layoutX="658.0" layoutY="502.0"
mnemonicParsing="false" text="Supervisor Mode" textAlignment="CENTER" />

            <CheckBox fx:id="showPasswordChecked" layoutX="658.0" layoutY="537.0"
mnemonicParsing="false" onAction="#toggleShowPassword" text="Show Password" />

            <Pane fx:id="loginErrorField" layoutX="196.0" layoutY="9.0"
prefHeight="26.0" prefWidth="528.0" style="-fx-background-color: transparent; -
fx-border-color: black; -fx-border-radius: 10px; -fx-border-width: 2; -fx-border-
color: red;" visible="false">

               <children>

                  <Text fx:id="errorReason" fill="RED" layoutX="87.0"
layoutY="17.0" strokeType="OUTSIDE" strokeWidth="0.0" text="Error Field"
textAlignment="CENTER" wrappingWidth="362.13671875" />

               </children>

            </Pane>

         </children>

      </AnchorPane>
```

//Source Code: parentStage.fxml

```
      <?import javafx.scene.control.Button?>

      <?import javafx.scene.image.Image?>

      <?import javafx.scene.image.ImageView?>

      <?import javafx.scene.layout.AnchorPane?>

      <?import javafx.scene.layout.Pane?>


      <AnchorPane id="dashboardAnchorPane" fx:id="mainstageAnchorPane"
maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-
Infinity" prefHeight="650.0" prefWidth="919.0" style="-fx-background-color:
#ffffff;" xmlns="http://javafx.com/javafx/15.0.1"
xmlns:fx="http://javafx.com/fxml/1"
fx:controller="com.application.gui.parentmainStage">

         <children>

            <AnchorPane id="menuBarAnchorPane" fx:id="menuBarAnchorPane"
prefHeight="40.0" prefWidth="318.0" style="-fx-background-color: ffffff; -fx-
border-color: black;" AnchorPane.leftAnchor="0.0" AnchorPane.rightAnchor="0.0"
AnchorPane.topAnchor="0.0">

               <children>

                  <ImageView fx:id="cornerIcon" fitHeight="27.0" fitWidth="31.0"
layoutX="880.0" layoutY="7.0" onMouseClicked="#openExitMenu">

                     <image>
```

```
                        <Image url="@../images/user_menu.png" />
                     </image>
                  </ImageView>
                  <Pane fx:id="applicationPane" layoutX="631.0" layoutY="3.0"
prefHeight="35.0" prefWidth="240.0" style="-fx-background-color: E0E0E0E0; -fx-
border-color: black; -fx-border-radius: 5 5 5 5; -fx-border-width: 2px;"
visible="false">
                     <children>
                        <Button fx:id="exitApplication" layoutX="14.0"
layoutY="2.0" mnemonicParsing="false" onAction="#exitApplicationBtnClicked"
prefHeight="30.0" prefWidth="100.0" style="-fx-background-color: #f6a91c;"
text="EXIT" textFill="WHITE" />
                        <Button fx:id="logOutBtn" layoutX="124.0" layoutY="2.0"
mnemonicParsing="false" onAction="#logOutBtnClicked"
```

//Source Code: parentStage.fxml

```
     prefHeight="30.0" prefWidth="100.0" style="-fx-background-color: #f6a91c;"
text="LOGOUT" textFill="WHITE" />
                     </children>
                  </Pane>
               </children></AnchorPane>
            <AnchorPane id="dashViewsAnchorPane" fx:id="mainView" layoutY="40.0"
prefHeight="608.0" prefWidth="919.0" style="-fx-background-color: #ffffff;"
AnchorPane.bottomAnchor="2.0" AnchorPane.leftAnchor="0.0"
AnchorPane.rightAnchor="0.0" AnchorPane.topAnchor="40.0" />
         </children>
      </AnchorPane>
```

//Source Code: regEmployee.fxml

```
      <?xml version="1.0" encoding="UTF-8"?>

      <?import javafx.scene.Cursor?>
      <?import javafx.scene.control.Button?>
      <?import javafx.scene.control.CheckBox?>
      <?import javafx.scene.control.DatePicker?>
      <?import javafx.scene.control.ScrollPane?>
      <?import javafx.scene.control.Tab?>
```

```
      <?import javafx.scene.control.TabPane?>
      <?import javafx.scene.control.TextArea?>
      <?import javafx.scene.control.TextField?>
      <?import javafx.scene.image.Image?>
      <?import javafx.scene.image.ImageView?>
      <?import javafx.scene.layout.AnchorPane?>
      <?import javafx.scene.layout.ColumnConstraints?>
      <?import javafx.scene.layout.GridPane?>
      <?import javafx.scene.layout.Pane?>
      <?import javafx.scene.layout.RowConstraints?>
      <?import javafx.scene.shape.Line?>
      <?import javafx.scene.text.Font?>
      <?import javafx.scene.text.Text?>
      <?import javafx.scene.web.WebView?>


      <AnchorPane fx:id="mainDashboard" maxHeight="-Infinity" maxWidth="-Infinity"
minHeight="-Infinity" minWidth="-Infinity" prefHeight="650.0" prefWidth="919.0"
style="-fx-background-color: #003087; -fx-border-radius: 3;"
stylesheets="@../styles/regularEmployeeStyle.css"
xmlns="http://javafx.com/javafx/15.0.1" xmlns:fx="http://javafx.com/fxml/1"
fx:controller="com.application.gui.regemployeeStage">
         <children>
            <TabPane layoutY="184.0" prefHeight="459.0" prefWidth="919.0"
tabClosingPolicy="UNAVAILABLE">
```

//Source Code: regEmployee.fxml

```
        <tabs>
                <Tab closable="false" text="Main Page">
                  <content>
                    <AnchorPane id="personalInfo" minHeight="0.0" minWidth="0.0"
prefHeight="650.0" prefWidth="919.0">
                           <children>
                              <TextField fx:id="firstName" layoutX="27.0"
layoutY="75.0" style="-fx-border-color: #00308f; -fx-background-color: #ffffff;"
/>
                              <Button fx:id="editPersonalInfo" layoutX="26.0"
layoutY="307.0" mnemonicParsing="false" onAction="#editPersonalInfoClicked"
```

```
prefHeight="31.0" prefWidth="184.0" style="-fx-background-color: #003087;"
text="Edit Contact Information" textFill="WHITE" />

                              <Text layoutX="28.0" layoutY="60.0"
strokeType="OUTSIDE" strokeWidth="0.0" text="First Name" />

                              <Text layoutX="202.0" layoutY="60.0"
strokeType="OUTSIDE" strokeWidth="0.0" text="Last Name" />

                              <TextField fx:id="lastName" layoutX="201.0"
layoutY="77.0" style="-fx-background-color: #ffffff; -fx-border-color: #00308f;"
/>

                              <Text layoutX="629.0" layoutY="123.0"
strokeType="OUTSIDE" strokeWidth="0.0" text="Hire Date" />

                              <TextField fx:id="hireDate" layoutX="629.0"
layoutY="132.0" style="-fx-border-color: #00308f; -fx-background-color: #ffffff;"
/>

                              <Text layoutX="28.0" layoutY="158.0"
strokeType="OUTSIDE" strokeWidth="0.0" />

                              <Text layoutX="629.0" layoutY="60.0"
strokeType="OUTSIDE" strokeWidth="0.0" text="Salary" />

                              <TextField fx:id="salary" layoutX="629.0"
layoutY="69.0" style="-fx-background-color: #ffffff; -fx-border-color: #00308f;"
/>

                              <Text layoutX="461.0" layoutY="239.0"
strokeType="OUTSIDE" strokeWidth="0.0" text="Certifications" />

                              <Line endX="-100.0" endY="409.0" layoutX="498.0"
layoutY="10.0" startX="-100.0" stroke="#00308f" />
```

//Source Code: regEmployee.fxml

```
      <Text layoutX="456.0" layoutY="60.0" strokeType="OUTSIDE" strokeWidth="0.0"
text="Next Evaluation" />

                              <TextField fx:id="nextEvaluation" layoutX="456.0"
layoutY="69.0" style="-fx-background-color: #ffffff; -fx-border-color: #00308f;"
/>

                              <Text layoutX="458.0" layoutY="123.0"
strokeType="OUTSIDE" strokeWidth="0.0" text="Shift Assigned" />

                              <TextField fx:id="shiftAssigned" layoutX="458.0"
layoutY="132.0" style="-fx-background-color: ffffff; -fx-border-color: #00308f;"
/>

                              <Text layoutX="460.0" layoutY="180.0"
strokeType="OUTSIDE" strokeWidth="0.0" text="Supervisor" />

                              <TextField fx:id="supervisor" layoutX="458.0"
layoutY="190.0" style="-fx-background-color: #ffffff; -fx-border-color: #00308f;"
/>
```

```
                              <Text layoutX="28.0" layoutY="123.0"
strokeType="OUTSIDE" strokeWidth="0.0" text="Street Address" />

                              <TextField fx:id="streetAddress" layoutX="27.0"
layoutY="132.0" prefHeight="25.0" prefWidth="327.0" style="-fx-background-color:
ffffff; -fx-border-color: #00308f;" />

                              <Text layoutX="29.0" layoutY="180.0"
strokeType="OUTSIDE" strokeWidth="0.0" text="City" />

                              <Text layoutX="206.0" layoutY="177.0"
strokeType="OUTSIDE" strokeWidth="0.0" text="Zipcode" />

                              <Text layoutX="28.0" layoutY="240.0"
strokeType="OUTSIDE" strokeWidth="0.0" text="State" />

                              <TextField fx:id="city" layoutX="27.0"
layoutY="190.0" style="-fx-background-color: ffffff; -fx-border-color: #00308f;"
/>

                              <TextField fx:id="state" layoutX="27.0"
layoutY="250.0" prefHeight="27.0" prefWidth="148.0" style="-fx-background-color:
ffffff; -fx-border-color: #00308f;" />

                              <TextField fx:id="zipcode" layoutX="201.0"
layoutY="186.0" style="-fx-background-color: ffffff; -fx-border-color: #00308f;"
/>

                              <Button fx:id="savePersonalInfo" layoutX="26.0"
layoutY="307.0" mnemonicParsing="false" onAction="#savePersonalInfoClicked"
prefHeight="31.0" prefWidth="184.0"
```

//Source Code: regEmployee.fxml

```
     style="-fx-background-color: #006400;" text="Save" textFill="WHITE"
visible="false" />

                              <Text layoutX="22.0" layoutY="37.0"
strokeType="OUTSIDE" strokeWidth="0.0" text="Contact Information">

                                 <font>

                                    <Font name="Arial Black" size="13.0" />

                                 </font>

                              </Text>

                              <Text layoutX="441.0" layoutY="37.0"
strokeType="OUTSIDE" strokeWidth="0.0" text="Job Information">

                                 <font>

                                    <Font name="Arial Black" size="13.0" />

                                 </font>

                              </Text>
```

```
                              <TextField fx:id="jobTitle" layoutX="629.0"
layoutY="190.0" style="-fx-border-color: #00308f; -fx-background-color: #ffffff;"
/>

                              <Text layoutX="632.0" layoutY="180.0"
strokeType="OUTSIDE" strokeWidth="0.0" text="Job Title" />

                              <Pane layoutX="458.0" layoutY="248.0"
prefHeight="174.0" prefWidth="324.0" style="-fx-border-color: #00308f;">

                                 <children>

                                    <Pane fx:id="certsPane" layoutX="29.0"
layoutY="14.0" prefHeight="147.0" prefWidth="264.0" />

                                 </children>

                              </Pane>

                           </children></AnchorPane>

                  </content>

                </Tab>

                <Tab text="Emergency Contact">

                  <content>

                    <AnchorPane id="personalInfo" minHeight="0.0" minWidth="0.0"
prefHeight="180.0" prefWidth="200.0">
```

//Source Code: regEmployee.fxml

```
               <children>

                              <Text layoutX="27.0" layoutY="59.0"
strokeType="OUTSIDE" strokeWidth="0.0" text="Full Name" />

                              <TextField fx:id="emergencyName" layoutX="27.0"
layoutY="70.0" style="-fx-background-color: ffffff; -fx-border-color: #00308f;"
/>

                              <Text layoutX="28.0" layoutY="115.0"
strokeType="OUTSIDE" strokeWidth="0.0" text="Relation" />

                              <TextField fx:id="relation" layoutX="27.0"
layoutY="128.0" prefHeight="28.0" prefWidth="152.0" style="-fx-background-color:
ffffff; -fx-border-color: #00308f;" />

                              <Text layoutX="220.0" layoutY="57.0"
strokeType="OUTSIDE" strokeWidth="0.0" text="Phone Number" />

                              <TextField fx:id="emergencyPhone" layoutX="220.0"
layoutY="68.0" style="-fx-background-color: ffffff; -fx-border-color: #00308f;"
/>

                              <Text layoutX="28.0" layoutY="171.0"
strokeType="OUTSIDE" strokeWidth="0.0" text="Address" />
```

```
                              <TextArea fx:id="emergencyAddress" layoutX="28.0"
layoutY="180.0" prefHeight="95.0" prefWidth="343.0" style="-fx-background-color:
ffffff; -fx-border-color: #00308f;" />

                              <Text layoutX="220.0" layoutY="115.0"
strokeType="OUTSIDE" strokeWidth="0.0" text="Work Phone Number" />

                              <TextField fx:id="emergencyWorkPhone"
layoutX="220.0" layoutY="129.0" style="-fx-background-color: ffffff; -fx-border-
color: #00308f;" />

                              <Button fx:id="editPrimaryContact" layoutX="28.0"
layoutY="286.0" mnemonicParsing="false" onAction="#editPrimaryContactClicked"
prefHeight="31.0" prefWidth="184.0" style="-fx-background-color: #003087;"
text="Edit Primary Contact" textFill="WHITE" />

                              <Button fx:id="savePrimaryContact" layoutX="28.0"
layoutY="286.0" mnemonicParsing="false" onAction="#savePrimaryContactClicked"
prefHeight="31.0" prefWidth="184.0" style="-fx-background-color: #006400;"
text="Save" textFill="WHITE" visible="false" />

                              <Line endX="-100.0" endY="409.0" layoutX="536.0"
layoutY="11.0" startX="-100.0" stroke="#00308f" />
```

//Source Code: regEmployee.fxml

```
                              <Text layoutX="22.0" layoutY="37.0"
strokeType="OUTSIDE" strokeWidth="0.0" text="Primary Contact">
                                 <font>
                                    <Font name="Arial Black" size="13.0" />
                                 </font>
                              </Text>
                              <Text layoutX="483.0" layoutY="37.0"
strokeType="OUTSIDE" strokeWidth="0.0" text="Secondary Contact">
                                 <font>
                                    <Font name="Arial Black" size="13.0" />
                                 </font>
                              </Text>
                              <Text layoutX="484.0" layoutY="59.0"
strokeType="OUTSIDE" strokeWidth="0.0" text="Full Name" />
                              <TextField fx:id="emergencyNameSec" layoutX="483.0"
layoutY="70.0" style="-fx-background-color: ffffff; -fx-border-color: #00308f;"
/>
                              <Text layoutX="682.0" layoutY="58.0"
strokeType="OUTSIDE" strokeWidth="0.0" text="Phone Number" />
```

```
                              <Text layoutX="482.0" layoutY="115.0"
strokeType="OUTSIDE" strokeWidth="0.0" text="Relation" />

                              <Text layoutX="680.0" layoutY="116.0"
strokeType="OUTSIDE" strokeWidth="0.0" text="Work Phone Number" />

                              <Text layoutX="482.0" layoutY="171.0"
strokeType="OUTSIDE" strokeWidth="0.0" text="Address" />

                              <TextField fx:id="emergencyPhoneSec" layoutX="680.0"
layoutY="70.0" style="-fx-background-color: ffffff; -fx-border-color: #00308f;"
/>

                              <TextField fx:id="emergencyRelationSec"
layoutX="483.0" layoutY="129.0" style="-fx-background-color: ffffff; -fx-border-
color: #00308f;" />

                              <TextField fx:id="emergencyWorkPhoneSec"
layoutX="680.0" layoutY="129.0" style="-fx-background-color: ffffff; -fx-border-
color: #00308f;" />
```

//Source Code: regEmployee.fxml

```
     <TextArea fx:id="emergencyAddressSec" layoutX="482.0" layoutY="180.0"
prefHeight="95.0" prefWidth="343.0" style="-fx-background-color: ffffff; -fx-
border-color: #00308f;" />

                              <Button fx:id="editSecondaryContact" layoutX="484.0"
layoutY="286.0" mnemonicParsing="false" onAction="#editSecondaryContactClicked"
prefHeight="31.0" prefWidth="184.0" style="-fx-background-color: #003087;"
text="Edit Secondary Contact" textFill="WHITE" />

                              <Button fx:id="saveSecondaryContact" layoutX="483.0"
layoutY="286.0" mnemonicParsing="false" onAction="#saveSecondaryContactClicked"
prefHeight="31.0" prefWidth="184.0" style="-fx-background-color: #006400;"
text="Save" textFill="WHITE" visible="false" />

                           </children></AnchorPane>

                  </content>

                </Tab>

                  <Tab fx:id="leaveBalances" text="Leave Balances">

                    <content>

                      <AnchorPane id="personalInfo" fx:id="leaveAnchorPane"
minHeight="0.0" minWidth="0.0" prefHeight="180.0" prefWidth="200.0">

                           <children>

                              <Text layoutX="23.0" layoutY="63.0"
strokeType="OUTSIDE" strokeWidth="0.0" text="Vacation Time" />

                              <Text layoutX="23.0" layoutY="124.0"
strokeType="OUTSIDE" strokeWidth="0.0" text="Sick Time" />
```

```
                              <Text layoutX="23.0" layoutY="279.0"
strokeType="OUTSIDE" strokeWidth="0.0" text="Sick Time Used YTD" />

                              <Text layoutX="23.0" layoutY="217.0"
strokeType="OUTSIDE" strokeWidth="0.0" text="Vacation Time Used YTD" />

                              <TextField fx:id="vacationTime" layoutX="23.0"
layoutY="69.0" prefHeight="25.0" prefWidth="108.0" style="-fx-background-color:
ffffff; -fx-border-color: #003087;" />

                              <TextField fx:id="sickTime" layoutX="22.0"
layoutY="133.0" prefHeight="25.0" prefWidth="108.0" style="-fx-background-color:
ffffff; -fx-border-color: #003087;" />

                              <TextField fx:id="sickTimeYTD" layoutX="23.0"
layoutY="292.0" prefHeight="25.0" prefWidth="108.0" style="-fx-background-color:
ffffff; -fx-border-color: #003087;" />
```

//Source Code: regEmployee.fxml

```
               <TextField fx:id="vacationTimeYTD" layoutX="22.0" layoutY="227.0"
prefHeight="25.0" prefWidth="108.0" style="-fx-background-color: ffffff; -fx-
border-color: #003087;" />

                              <Line endX="15.0" endY="392.5" layoutX="201.0"
layoutY="27.0" startX="15.0" startY="-13.5" stroke="#002b94" />

                              <Text layoutX="247.0" layoutY="37.0"
strokeType="OUTSIDE" strokeWidth="0.0" text="Request Time off">

                                 <font>

                                    <Font name="Arial Black" size="13.0" />

                                 </font>

                              </Text>

                              <TextField fx:id="requestStartTime" layoutX="248.0"
layoutY="179.0" prefHeight="25.0" prefWidth="108.0" style="-fx-background-color:
ffffff; -fx-border-color: #003087;" />

                              <CheckBox fx:id="sickTimeChecked" layoutX="383.0"
layoutY="246.0" mnemonicParsing="false" text="Sick Time" />

                              <CheckBox fx:id="vacationTimeChecked"
layoutX="383.0" layoutY="220.0" mnemonicParsing="false" text="Vacation Time" />

                              <Text layoutX="247.0" layoutY="63.0"
strokeType="OUTSIDE" strokeWidth="0.0" text="Start Date" />

                              <Text layoutX="248.0" layoutY="116.0"
strokeType="OUTSIDE" strokeWidth="0.0" text="End Date" />

                              <Text layoutX="246.0" layoutY="165.0"
strokeType="OUTSIDE" strokeWidth="0.0" text="Start Time" />

                              <TextField fx:id="requestEndTime" layoutX="380.0"
layoutY="179.0" prefHeight="25.0" prefWidth="108.0" style="-fx-background-color:
ffffff; -fx-border-color: #003087;" />
```

```
                              <Text layoutX="384.0" layoutY="165.0"
strokeType="OUTSIDE" strokeWidth="0.0" text="End Time" />

                              <Text layoutX="22.0" layoutY="37.0"
strokeType="OUTSIDE" strokeWidth="0.0" text="Time Balances">

                                 <font>

                                    <Font name="Arial Black" size="13.0" />

                                 </font>

                              </Text>
```

//Source Code: regEmployee.fxml

```
       <Text layoutX="23.0" layoutY="192.0" strokeType="OUTSIDE"
strokeWidth="0.0" text="Time Used ">

                                 <font>

                                    <Font name="Arial Black" size="13.0" />

                                 </font>

                              </Text>

                              <TextField fx:id="requestReason" layoutX="246.0"
layoutY="296.0" prefHeight="31.0" prefWidth="250.0" style="-fx-background-color:
ffffff; -fx-border-color: #003087;" />

                              <Text layoutX="246.0" layoutY="284.0"
strokeType="OUTSIDE" strokeWidth="0.0" text="Reason" />

                              <Line endX="15.0" endY="392.5" layoutX="517.0"
layoutY="23.0" startX="15.0" startY="-13.5" stroke="#002b94" />

                              <Text layoutX="569.0" layoutY="37.0"
strokeType="OUTSIDE" strokeWidth="0.0" text="Time off Approval Notification">

                                 <font>

                                    <Font name="Arial Black" size="13.0" />

                                 </font>

                              </Text>

                              <DatePicker fx:id="requestStartDate" layoutX="246.0"
layoutY="70.0" style="-fx-background-color: ffffff; -fx-border-color: #003087;"
/>

                              <DatePicker fx:id="requestEndDate" layoutX="246.0"
layoutY="121.0" style="-fx-background-color: ffffff; -fx-border-color: #003087;"
/>

                              <Button fx:id="submitRequest" layoutX="247.0"
layoutY="343.0" mnemonicParsing="false" onAction="#submitRequestClicked"
prefHeight="31.0" prefWidth="184.0" style="-fx-background-color: #003087;"
text="Submit Request" textFill="WHITE" />
```

```
                              <Pane fx:id="approvalNotificationsTwo"
layoutX="555.0" layoutY="53.0" prefHeight="363.0" prefWidth="343.0" style="-fx-
background-color: ffffff; -fx-border-color: #003087;">

                                 <children>

                                    <Text layoutX="18.0" layoutY="26.0"
strokeType="OUTSIDE" strokeWidth="0.0" text="Date Start" />
```

//Source Code: regEmployee.fxml

```
      <Text fx:id="timeapprovalNotification" layoutX="102.0" layoutY="26.0"
strokeType="OUTSIDE" strokeWidth="0.0" text="Date End" />

                                    <Text layoutX="191.0" layoutY="26.0"
strokeType="OUTSIDE" strokeWidth="0.0" text="Status" />

                                    <Pane fx:id="approvalNotifications"
layoutX="18.0" layoutY="39.0" prefHeight="312.0" prefWidth="307.0" />

                                 </children></Pane>

                              <AnchorPane fx:id="requestErrorPane" layoutX="223.0"
layoutY="32.0" prefHeight="149.0" prefWidth="441.0" style="-fx-background-color:
#E8E8E8; -fx-border-color: black; -fx-border-radius: 5px; -fx-border-width: 2px;"
visible="false">

                                 <children>

                                    <Pane layoutX="1.0" prefHeight="200.0"
prefWidth="441.0">

                                       <children>

                                          <Text fx:id="errorTexTField"
layoutY="96.0" strokeType="OUTSIDE" strokeWidth="0.0" text="Error Field"
textAlignment="CENTER" wrappingWidth="441.13671875">

                                             <font>

                                                <Font size="13.0" />

                                             </font>

                                          </Text>

                                          <Button fx:id="closeErrorField"
layoutX="133.0" layoutY="146.0" mnemonicParsing="false"
onAction="#closeErrorFieldClicked" prefHeight="31.0" prefWidth="184.0" style="-
fx-background-color: red;" text="Close" textFill="WHITE" />

                                       </children>

                                    </Pane>

                                 </children>

                              </AnchorPane>

                              <AnchorPane fx:id="requestSuccessPane"
layoutX="233.0" layoutY="42.0" prefHeight="149.0" prefWidth="441.0" style="-fx-
```

```
background-color: #E8E8E8; -fx-border-color: black; -fx-border-radius: 5px; -fx-
border-width: 2px;" visible="false">
                                    <children>
```

//Source Code: regEmployee.fxml

```
      <Pane layoutX="1.0" prefHeight="200.0" prefWidth="441.0">
                                          <children>
                                             <Text fx:id="requestSuccessField"
layoutY="96.0" strokeType="OUTSIDE" strokeWidth="0.0" text="Success Field"
textAlignment="CENTER" wrappingWidth="441.13671875">
                                                <font>
                                                   <Font size="13.0" />
                                                </font>
                                             </Text>
                                             <Button fx:id="requestSuccessBtn"
layoutX="133.0" layoutY="146.0" mnemonicParsing="false"
onAction="#requestBtnConfirmationClicked" prefHeight="31.0" prefWidth="184.0"
style="-fx-background-color: #006400;" text="Close" textFill="WHITE" />
                                          </children>
                                       </Pane>
                                    </children>
                                 </AnchorPane>
                              </children></AnchorPane>
                        </content>
                     </Tab>
                     <Tab fx:id="schedulePane"
onSelectionChanged="#getScheduleForEmployee" text="Scheduled Posts">
                        <content>
                          <AnchorPane id="personalInfo" minHeight="0.0" minWidth="0.0"
prefHeight="180.0" prefWidth="200.0">
                              <cursor>
                                 <Cursor fx:constant="DEFAULT" />
                              </cursor>
                              <children>
                                 <Text layoutX="22.0" layoutY="37.0"
strokeType="OUTSIDE" strokeWidth="0.0" text="Current Schedule">
                                    <font>
```

//Source Code: regEmployee.fxml

```
      <Font name="Arial Black" size="13.0" />
                                </font>
                              </Text>
                              <Pane fx:id="currentSchedulePane" layoutY="41.0"
prefHeight="390.0" prefWidth="919.0">
                                 <children>
                                    <GridPane fx:id="scheduleGridPane"
layoutX="24.0" layoutY="37.0" prefHeight="90.0" prefWidth="885.0" style="-fx-
background-color: #003087; -fx-background-radius: 5 5 5 5; -fx-border-color:
black; -fx-border-radius: 5;">
                                       <columnConstraints>
                                         <ColumnConstraints hgrow="SOMETIMES"
minWidth="10.0" prefWidth="100.0" />
                                         <ColumnConstraints hgrow="SOMETIMES"
minWidth="10.0" prefWidth="100.0" />
                                           <ColumnConstraints hgrow="SOMETIMES"
minWidth="10.0" prefWidth="100.0" />
                                           <ColumnConstraints hgrow="SOMETIMES"
minWidth="10.0" prefWidth="100.0" />
                                           <ColumnConstraints hgrow="SOMETIMES"
minWidth="10.0" prefWidth="100.0" />
                                           <ColumnConstraints hgrow="SOMETIMES"
minWidth="10.0" prefWidth="100.0" />
                                           <ColumnConstraints hgrow="SOMETIMES"
minWidth="10.0" prefWidth="100.0" />
                                           <ColumnConstraints hgrow="SOMETIMES"
minWidth="10.0" prefWidth="100.0" />
                                       </columnConstraints>
                                       <rowConstraints>
                                           <RowConstraints maxHeight="47.0"
minHeight="10.0" prefHeight="33.0" vgrow="SOMETIMES" />
                                         <RowConstraints maxHeight="47.0"
minHeight="10.0" prefHeight="33.0" vgrow="SOMETIMES" />
                                         <RowConstraints maxHeight="68.0"
minHeight="10.0" prefHeight="57.0" vgrow="SOMETIMES" />
```

//Source Code: regEmployee.fxml

```
             </rowConstraints>
```

```
                                    <children>
                                        <Text fill="WHITE" strokeType="OUTSIDE"
strokeWidth="0.0" text="Monday" GridPane.columnIndex="1" GridPane.rowIndex="1">
                                            <font>
                                                <Font name="System Bold"
size="14.0" />
                                            </font>
                                        </Text>
                                        <Text fill="WHITE" strokeType="OUTSIDE"
strokeWidth="0.0" text="Tuesday" GridPane.columnIndex="2" GridPane.rowIndex="1">
                                            <font>
                                                <Font name="System Bold"
size="14.0" />
                                            </font>
                                        </Text>
                                        <Text fill="WHITE" strokeType="OUTSIDE"
strokeWidth="0.0" text="Wednesday" GridPane.columnIndex="3"
GridPane.rowIndex="1">
                                            <font>
                                                <Font name="System Bold"
size="14.0" />
                                            </font>
                                        </Text>
                                        <Text fill="WHITE" strokeType="OUTSIDE"
strokeWidth="0.0" text="Thursday" GridPane.columnIndex="4" GridPane.rowIndex="1">
                                            <font>
                                                <Font name="System Bold"
size="14.0" />
                                            </font>
                                        </Text>
```

//Source Code: regEmployee.fxml

```
                                    <Text fill="WHITE" strokeType="OUTSIDE"
strokeWidth="0.0" text="Friday" GridPane.columnIndex="5" GridPane.rowIndex="1">
                                            <font>
                                                <Font name="System Bold"
size="14.0" />
                                            </font>
```

```
                                        </Text>
                                        <Text fill="WHITE" strokeType="OUTSIDE"
strokeWidth="0.0" text="Saturday" GridPane.columnIndex="6" GridPane.rowIndex="1">
                                           <font>
                                              <Font name="System Bold"
size="14.0" />
                                           </font>
                                        </Text>
                                        <Text fill="WHITE" strokeType="OUTSIDE"
strokeWidth="0.0" text="Sunday" GridPane.columnIndex="7" GridPane.rowIndex="1">
                                           <font>
                                              <Font name="System Bold"
size="14.0" />
                                           </font>
                                        </Text>
                                        <Text fill="WHITE" strokeType="OUTSIDE"
strokeWidth="0.0" text="Day:" textAlignment="CENTER" GridPane.rowIndex="1">
                                           <font>
                                              <Font name="System Bold"
size="14.0" />
                                           </font>
                                        </Text>
                                        <Text fill="WHITE" strokeType="OUTSIDE"
strokeWidth="0.0" text="Post:" GridPane.rowIndex="2">
                                           <font>
```

//Source Code: regEmployee.fxml

```
                           <Font name="System Bold" size="14.0" />
                                           </font>
                                        </Text>
                                        <Text fill="WHITE" strokeType="OUTSIDE"
strokeWidth="0.0" text="Date Range:">
                                           <font>
                                              <Font name="System Bold"
size="14.0" />
                                           </font>
                                        </Text>
```

```
                                    </children>
                                 </GridPane>
                              </children>
                           </Pane>
                        </children>
                     </AnchorPane>
                  </content>
               </Tab>
               <Tab fx:id="boloSelected" onSelectionChanged="#displayBoloPage"
text="BOLO Alerts">
                  <content>
                     <AnchorPane id="personalInfo" minHeight="0.0"
minWidth="0.0" prefHeight="180.0" prefWidth="200.0">
                        <cursor>
                           <Cursor fx:constant="DEFAULT" />
                        </cursor>
                        <children>
                           <Text layoutX="22.0" layoutY="37.0"
strokeType="OUTSIDE" strokeWidth="0.0" text="Be On a Look Out ">
                              <font>
                                 <Font name="Arial Black" size="13.0" />
                              </font>
```

//Source Code: regEmployee.fxml

```
        </Text>
                           <ScrollPane fx:id="boloscrollPane"
prefHeight="427.0" prefWidth="919.0">
                              <content>
                                 <WebView fx:id="bolowebView"
prefHeight="417.0" prefWidth="919.0" />
                              </content>
                           </ScrollPane>
                        </children>
                     </AnchorPane>
                  </content>
               </Tab>
```

```
                <Tab fx:id="resourcesPane"
onSelectionChanged="#displayResourcesRegular" text="Resources">
                    <content>
                        <AnchorPane id="personalInfo" minHeight="0.0"
minWidth="0.0" prefHeight="180.0" prefWidth="200.0">
                            <cursor>
                                <Cursor fx:constant="DEFAULT" />
                            </cursor>
                            <children>
                                <Text layoutX="22.0" layoutY="37.0"
strokeType="OUTSIDE" strokeWidth="0.0">
                                    <font>
                                        <Font name="Arial Black" size="13.0" />
                                    </font>
                                </Text>
                                <ScrollPane fx:id="resourceScrollPane" layoutY="2.0"
prefHeight="427.0" prefWidth="919.0">
                                    <content>
                                        <WebView fx:id="resourceWebView"
prefHeight="417.0" prefWidth="919.0" />
```

//Source Code: regEmployee.fxml

```
                                    </content>
                                </ScrollPane>
                            </children>
                        </AnchorPane>
                    </content>
                </Tab>
            </tabs>
        </TabPane>
        <Text id="mainHeader" fill="WHITE" layoutX="106.0" layoutY="70.0"
strokeType="OUTSIDE" strokeWidth="0.0" text="U.S Air Force Security Forces
Management System" textAlignment="CENTER" wrappingWidth="324.08984375" />
        <ImageView id="defensorFortis" fitHeight="184.0" fitWidth="307.0"
layoutX="612.0" pickOnBounds="true" preserveRatio="true">
            <image>
                <Image url="@../images/defensor-fortis.png" />
```

```
            </image>
         </ImageView>
      </children>
   </AnchorPane>
```

//Source Code: supervisorMode.fxml

```
   <?xml version="1.0" encoding="UTF-8"?>

   <?import javafx.scene.control.Button?>
   <?import javafx.scene.control.ComboBox?>
   <?import javafx.scene.control.DatePicker?>
   <?import javafx.scene.control.ScrollPane?>
   <?import javafx.scene.control.TextArea?>
   <?import javafx.scene.control.TextField?>
   <?import javafx.scene.image.Image?>
   <?import javafx.scene.image.ImageView?>
   <?import javafx.scene.layout.AnchorPane?>
   <?import javafx.scene.layout.Pane?>
   <?import javafx.scene.layout.VBox?>
   <?import javafx.scene.text.Font?>
   <?import javafx.scene.text.Text?>
   <?import javafx.scene.web.WebView?>

   <AnchorPane fx:id="supervisorMainAnchor" maxHeight="-Infinity" maxWidth="-
Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="650.0"
prefWidth="919.0" style="-fx-background-color: #003087;"
stylesheets="@../styles/supervisorModeStyle.css"
xmlns="http://javafx.com/javafx/15.0.1" xmlns:fx="http://javafx.com/fxml/1"
fx:controller="com.application.gui.supervisorModeStage">
      <children>
         <Pane id="menuBar" layoutX="-11.0" layoutY="184.0" prefHeight="42.0"
prefWidth="930.0" style="-fx-background-color: #003087;">
            <children>
               <Button id="menuButtons" fx:id="schedulingBtn" layoutX="75.0"
layoutY="5.0" mnemonicParsing="false" onAction="#displaySchedulePane"
prefHeight="31.0" prefWidth="184.0" text="Scheduling" textFill="WHITE">
```

```
                    <font>
                        <Font name="Lucida Console" size="12.0" />
```

//Source Code: supervisorMode.fxml

```
            </font>
                  </Button>
                  <Button id="menuButtons" fx:id="requestsOffBtn" layoutX="275.0"
layoutY="5.0" mnemonicParsing="false" onAction="#requestTimeOffPane"
prefHeight="31.0" prefWidth="184.0" text="Time Off Requests" textFill="WHITE">
                     <font>
                        <Font name="Lucida Console" size="12.0" />
                     </font>
                  </Button>
                  <Button id="menuButtons" fx:id="passdownsBtn" layoutX="475.0"
layoutY="5.0" mnemonicParsing="false" onAction="#displayDailyPassdownsPane"
prefHeight="31.0" prefWidth="184.0" text="Daily Passdowns" textFill="WHITE">
                     <font>
                        <Font name="Lucida Console" size="12.0" />
                     </font>
                  </Button>
                  <Button id="menuButtons" fx:id="resourcesBtn" layoutX="675.0"
layoutY="5.0" mnemonicParsing="false" onAction="#displayResourcesPane"
prefHeight="31.0" prefWidth="184.0" text="Resources" textFill="WHITE">
                     <font>
                        <Font name="Lucida Console" size="12.0" />
                     </font>
                  </Button>
               </children>
            </Pane>
            <Text id="mainHeader" fill="WHITE" layoutX="116.0" layoutY="80.0"
strokeType="OUTSIDE" strokeWidth="0.0" text="U.S Air Force Security Forces
Management System" textAlignment="CENTER" wrappingWidth="324.08984375" />
            <ImageView id="defensorFortis" fitHeight="184.0" fitWidth="307.0"
layoutX="612.0" pickOnBounds="true" preserveRatio="true">
               <image>
```

//Source Code: supervisorMode.fxml

```
                <Image url="@../images/defensor-fortis.png" />
               </image>
            </ImageView>
            <Pane id="schedulePane" fx:id="mainSupervisorPane" layoutY="225.0"
prefHeight="423.0" prefWidth="919.0">
               <children>
                  <AnchorPane fx:id="resourcesMainPane" prefHeight="425.0"
prefWidth="919.0" visible="false">
                     <children>
                        <ScrollPane layoutX="14.0" layoutY="14.0"
prefHeight="405.0" prefWidth="902.0">
                           <content>
                              <WebView fx:id="supervisorWebView"
prefHeight="398.0" prefWidth="907.0" />
                           </content>
                        </ScrollPane>
                     </children>
                  </AnchorPane>
                  <AnchorPane fx:id="passdownsAnchor" prefHeight="425.0"
prefWidth="919.0" visible="false">
                     <children>
                        <Pane layoutY="29.0" prefHeight="395.0" prefWidth="919.0">
                           <children>
                              <Text layoutX="21.0" layoutY="27.0"
strokeType="OUTSIDE" strokeWidth="0.0" text="Create new Passdown">
                                 <font>
                                    <Font name="System Bold" size="15.0" />
                                 </font>
                              </Text>
                              <Button fx:id="submitPassdowns" layoutX="21.0"
layoutY="182.0" mnemonicParsing="false" onAction="#submitPassdownsClicked"
```

//Source Code: supervisorMode.fxml

```
      prefHeight="31.0" prefWidth="184.0" style="-fx-background-color: #003087;"
text="Submit" textFill="WHITE" />
```

```
                              <Button fx:id="viewPassdowns" layoutX="220.0"
layoutY="182.0" mnemonicParsing="false" onAction="#viewPassdownsClicked"
prefHeight="31.0" prefWidth="184.0" style="-fx-background-color: #003087;"
text="View Passdowns" textFill="WHITE" />

                              <TextArea fx:id="passdownsTextArea" layoutX="21.0"
layoutY="37.0" prefHeight="141.0" prefWidth="876.0" />

                              <Pane fx:id="viewPassDownsPane" prefHeight="395.0"
prefWidth="919.0" style="-fx-background-color: white;" visible="false">

                                 <children>

                                    <ScrollPane fx:id="viewPassDownsScrollPane"
layoutX="6.0" layoutY="29.0" prefHeight="316.0" prefWidth="908.0" />

                                    <Button fx:id="closePassdownsBtn"
layoutX="12.0" layoutY="356.0" mnemonicParsing="false"
onAction="#closePassdownsClicked" prefHeight="20.0" prefWidth="119.0" style="-fx-
background-color: #003087;" text="Close" textFill="WHITE" />

                                    <Text layoutX="7.0" layoutY="19.0"
strokeType="OUTSIDE" strokeWidth="0.0" text="Submitted By"
wrappingWidth="91.13671875">

                                       <font>

                                          <Font name="System Bold" size="13.0" />

                                       </font>

                                    </Text>

                                    <Text layoutX="126.0" layoutY="19.0"
strokeType="OUTSIDE" strokeWidth="0.0" text="Date &amp;amp; Time"
wrappingWidth="91.13671875">

                                       <font>

                                          <Font name="System Bold" size="13.0" />

                                       </font>

                                    </Text>
```

//Source Code: supervisorMode.fxml

```
            <Text layoutX="244.0" layoutY="19.0" strokeType="OUTSIDE"
strokeWidth="0.0" text="Passdown" wrappingWidth="91.13671875">

                                       <font>

                                          <Font name="System Bold" size="13.0" />

                                       </font>

                                    </Text>
```

```
                                    <Button fx:id="printPassdowns" layoutX="146.0"
layoutY="356.0" mnemonicParsing="false" onAction="#printPassdownsClicked"
prefHeight="20.0" prefWidth="119.0" style="-fx-background-color: #003087;"
text="Print" textFill="WHITE" />

                                    <Button fx:id="savePassdowns" layoutX="282.0"
layoutY="356.0" mnemonicParsing="false" onAction="#savePassdownsBtnClicked"
prefHeight="20.0" prefWidth="119.0" style="-fx-background-color: #003087;"
text="Save" textFill="WHITE" />

                                 </children>

                              </Pane>

                           </children>

                        </Pane>

                        <AnchorPane fx:id="requestSuccessPane" layoutX="243.0"
layoutY="52.0" prefHeight="149.0" prefWidth="441.0" style="-fx-background-color:
#E8E8E8; -fx-border-color: black; -fx-border-radius: 5px; -fx-border-width: 2px;"
visible="false">

                           <children>

                              <Pane layoutX="1.0" prefHeight="200.0"
prefWidth="441.0">

                                 <children>

                                    <Text fx:id="requestSuccessField"
layoutY="96.0" strokeType="OUTSIDE" strokeWidth="0.0" text="Passdown Complete!"
textAlignment="CENTER" wrappingWidth="441.13671875">

                                       <font>

                                          <Font size="13.0" />

                                       </font>

                                    </Text>
```

//Source Code: supervisorMode.fxml

//Source Code: supervisorMode.fxml

```
        <Button fx:id="requestSuccessBtn" layoutX="133.0" layoutY="146.0"
mnemonicParsing="false" onAction="#requestBtnConfirmationClicked"
prefHeight="31.0" prefWidth="184.0" style="-fx-background-color: #006400;"
text="Close" textFill="WHITE" />

                                 </children>

                              </Pane>

                           </children>

                        </AnchorPane>

                     </children></AnchorPane>
```

```
                  <AnchorPane fx:id="requestsAnchor" prefHeight="425.0"
prefWidth="919.0" visible="false">
                     <children>
                        <Pane fx:id="timeOffRequests" layoutX="12.0"
layoutY="27.0" prefHeight="395.0" prefWidth="896.0" />
                        <Text layoutX="12.0" layoutY="24.0" strokeType="OUTSIDE"
strokeWidth="0.0" text="Name">
                           <font>
                              <Font name="Verdana Bold" size="12.0" />
                           </font>
                        </Text>
                        <Text layoutX="100.0" layoutY="24.0" strokeType="OUTSIDE"
strokeWidth="0.0" text="Start Date">
                           <font>
                              <Font name="Verdana Bold" size="12.0" />
                           </font>
                        </Text>
                        <Text layoutX="192.0" layoutY="24.0" strokeType="OUTSIDE"
strokeWidth="0.0" text="End Date">
                           <font>
                              <Font name="Verdana Bold" size="12.0" />
                           </font>
                        </Text>
```

//Source Code: supervisorMode.fxml

```
          <Text layoutX="280.0" layoutY="24.0" strokeType="OUTSIDE"
strokeWidth="0.0" text="Start Time">
                           <font>
                              <Font name="Verdana Bold" size="12.0" />
                           </font>
                        </Text>
                        <Text layoutX="375.0" layoutY="23.0" strokeType="OUTSIDE"
strokeWidth="0.0" text="End Time">
                           <font>
                              <Font name="Verdana Bold" size="12.0" />
```

```
                           </font>
                        </Text>
                        <Text layoutX="469.0" layoutY="22.0" strokeType="OUTSIDE"
strokeWidth="0.0" text="Reason">
                           <font>
                              <Font name="Verdana Bold" size="12.0" />
                           </font>
                        </Text>
                        <Text layoutX="554.0" layoutY="23.0" strokeType="OUTSIDE"
strokeWidth="0.0" text="Type">
                           <font>
                              <Font name="Verdana Bold" size="12.0" />
                           </font>
                        </Text>
                        <Text layoutX="642.0" layoutY="23.0" strokeType="OUTSIDE"
strokeWidth="0.0" text="Status">
                           <font>
                              <Font name="Verdana Bold" size="12.0" />
                           </font>
                        </Text>
                     </children></AnchorPane>
```

//Source Code: supervisorMode.fxml

```
      <AnchorPane fx:id="schedulingAnchor" prefHeight="425.0" prefWidth="919.0">
                     <children>
                        <Pane fx:id="schedulePane" prefHeight="425.0"
prefWidth="919.0">
                           <children>
                              <Text layoutX="21.0" layoutY="26.0"
strokeType="OUTSIDE" strokeWidth="0.0" text="Create a Schedule"
wrappingWidth="140.13671875">
                                 <font>
                                    <Font name="System Bold" size="13.0" />
                                 </font>
                              </Text>
```

```
                              <Text layoutX="21.0" layoutY="61.0"
strokeType="OUTSIDE" strokeWidth="0.0" text="Current Roster"
wrappingWidth="140.13671875">
                                 <font>
                                    <Font name="System Bold" size="13.0" />
                                 </font>
                              </Text>
                              <Pane fx:id="currentRoster" layoutX="21.0"
layoutY="71.0" prefHeight="344.0" prefWidth="200.0" style="-fx-border-color:
black;">
                                 <children>
                                    <VBox fx:id="rosterVbox" layoutX="7.0"
layoutY="6.0" prefHeight="332.0" prefWidth="184.0" />
                                 </children></Pane>
                              <Pane fx:id="createaSchedulePane" layoutX="234.0"
layoutY="71.0" prefHeight="344.0" prefWidth="672.0" style="-fx-border-color:
black;">
                                 <children>
                                    <DatePicker fx:id="dateScheduleFrom"
layoutX="57.0" layoutY="20.0" />
```

//Source Code: supervisorMode.fxml

```
         <ComboBox fx:id="comboBox" layoutX="57.0" layoutY="63.0"
prefHeight="28.0" prefWidth="173.0" promptText="Posts" />
                                    <TextField fx:id="scheduleEndTime"
layoutX="266.0" layoutY="106.0" prefHeight="29.0" prefWidth="173.0"
promptText="End Time 00:00 - 23:59" />
                                    <Button fx:id="submitSchedule" layoutX="57.0"
layoutY="158.0" mnemonicParsing="false" onAction="#submitScheduleUpdate"
prefHeight="29.0" prefWidth="173.0" style="-fx-background-color: #006400;"
text="Submit" textFill="WHITE" />
                                    <DatePicker fx:id="dateScheduleTo"
layoutX="266.0" layoutY="20.0" />
                                    <TextField fx:id="scheduleStartTime"
layoutX="57.0" layoutY="106.0" prefHeight="29.0" prefWidth="173.0"
promptText="Start Time 00:00 - 23:59" />
                                    <ComboBox fx:id="employeeComboBox"
layoutX="266.0" layoutY="63.0" prefHeight="28.0" prefWidth="173.0"
promptText="Roster" />
                                    <Pane fx:id="schedulelSuccess" layoutX="126.0"
layoutY="35.0" prefHeight="200.0" prefWidth="441.0" style="-fx-background-color:
```

```
#E8E8E8; -fx-border-color: black; -fx-border-radius: 5px; -fx-border-width: 2px;"
visible="false">

                                    <children>

                                        <Text fx:id="schedulelSuccessField"
layoutY="96.0" strokeType="OUTSIDE" strokeWidth="0.0" text="Schedule Submitted"
textAlignment="CENTER" wrappingWidth="441.13671875">

                                            <font>

                                                <Font size="13.0" />

                                            </font>

                                        </Text>

                                        <Button fx:id="schedulelSuccessBtn"
layoutX="133.0" layoutY="146.0" mnemonicParsing="false"
onAction="#submitRequestSuccessBtn" prefHeight="31.0" prefWidth="184.0" style="-
fx-background-color: #006400;" text="Close" textFill="WHITE" />

                                    </children>

                                </Pane>
```

//Source Code: supervisorMode.fxml

```
            <Pane fx:id="schedulError" layoutX="136.0" layoutY="45.0"
prefHeight="200.0" prefWidth="441.0" style="-fx-background-color: #E8E8E8; -fx-
border-color: black; -fx-border-radius: 5px; -fx-border-width: 2px;"
visible="false">

                                    <children>

                                        <Text fx:id="scheduleErrorField"
layoutY="96.0" strokeType="OUTSIDE" strokeWidth="0.0" text="Schedule Submitted"
textAlignment="CENTER" wrappingWidth="441.13671875">

                                            <font>

                                                <Font size="13.0" />

                                            </font>

                                        </Text>

                                        <Button fx:id="schedulelErrorBtn"
layoutX="133.0" layoutY="146.0" mnemonicParsing="false"
onAction="#scheduleErrorBtn" prefHeight="31.0" prefWidth="184.0" style="-fx-
background-color: red;" text="Close" textFill="WHITE" />

                                    </children>

                                </Pane>

                            </children>

                        </Pane>
```

```
                        </children>
                    </Pane>
                </children></AnchorPane>
            </children></Pane>
        </children>
    </AnchorPane>
```

//Source Code: loginStyle.css

```
    #mainHeader{
        -fx-font-family: "Baskerville Old Face";
        -fx-font-weight: bold;
        -fx-font-size: 30px;
    }
```

//Source Code: regularEmployeeStyle.css

```
    .tab-pane .tab:selected
    {
        -fx-background-color: #003087;
    }

    .tab .tab-label {
        -fx-alignment: CENTER;
        -fx-text-fill: #828282;
        -fx-font-size: 12px;
        -fx-font-weight: bold;
        -fx-font-family: "Arial Rounded MT Bold";
    }

    .tab:selected .tab-label {
        -fx-alignment: CENTER;
        -fx-text-fill: white;
```

```
    }

    .tab {
        -fx-background-color: #ffffff;
    }
```

//Source Code: loginStyle.css

```
    #personalInfo
    {
        -fx-background-color: white;
    }

    #mainHeader{
        -fx-font-family: "Baskerville Old Face";
        -fx-font-weight: bold;
        -fx-font-size: 30px;
    }

    #fortis{
        -fx-border-color: black;
        -fx-border-width: 10px;
    }
```

//Source Code: supervisorModeStyle.css

```
    #mainHeader{
        -fx-font-family: "Baskerville Old Face";
        -fx-font-weight: bold;
        -fx-font-size: 30px;
    }

    #schedulePane{
        -fx-background-color: white;
```

```
    }

    #menuBar{
        -fx-background-color: #003087;
```

//Source Code: supervisorModeStyle.css

```
     -fx-border-color: black;
        -fx-border-width: 2px;
    }

    #menuButtons{
        -fx-background-color:  #1E90FF;
        -fx-background-radius: 10;
        -fx-font-family: "Baskerville Old Face";
        -fx-font-weight: bold;
        -fx-font-size: 16px;
    }

    #menuButtons:hover{
        -fx-background-color: #f6a91c;
        -fx-text-fill: black;
```

//Source Code: build.gradle

```
    plugins {
        id 'java'
        id 'application'
        id 'org.openjfx.javafxplugin' version '0.0.9'
    }

    javafx {
        version = "15.0.1"
        modules = ['javafx.controls', 'javafx.fxml', 'javafx.web']
```

```
    }

//Source Code: build.gradle
    group 'com.application'
    version '1.0'

    sourceCompatibility = 1.8

    repositories {
        mavenCentral()
        flatDir {
            dirs 'libs'
        }
    }
    dependencies {
        testCompile group: 'org.testng', name: 'testng', version: '7.1.0'
        compile group: 'org.mongodb', name: 'mongodb-driver', version: '3.12.7'
        compile group: 'commons-io', name: 'commons-io', version: '2.8.0'
        implementation 'com.google.code.gson:gson:2.8.6'
    }

    test {
        useTestNG()
    }

    mainClassName = "com.application.launch
```

References

[1] MongoDB Java Drivers, *Introduction.* Accessed on : Apr. 18. 2021. [Online]. Available : https://mongodb.github.io/mongo-java-driver/

[2] Reactive Stream, *The Problem.* Accessed on : Apr. 18. 2021. [Online]. Available : http://www.reactive-streams.org/

[3] MongoDB , *Bson.* Accessed on : Apr. 18. 2021. [Online]. Available : https://mongodb.github.io/mongo-java-driver/4.3/bson/

[4] Codepath, *Leveraging the Gson Library.* Accessed on : May 22, 2021. [Online]. Available : https://guides.codepath.com/android/leveraging-the-gson-library

[5] Datatracker, *The Secure Shell (SSH) Protocol Architecure.* Accessed on : May 22, 2021. [Online]. Available : https://datatracker.ietf.org/doc/html/rfc4251