C868 – Software Capstone Project Summary

Task 2 – Section C

Capstone Proposal Project Name: Client Scheduling Application

Student Name: Henry Ventura

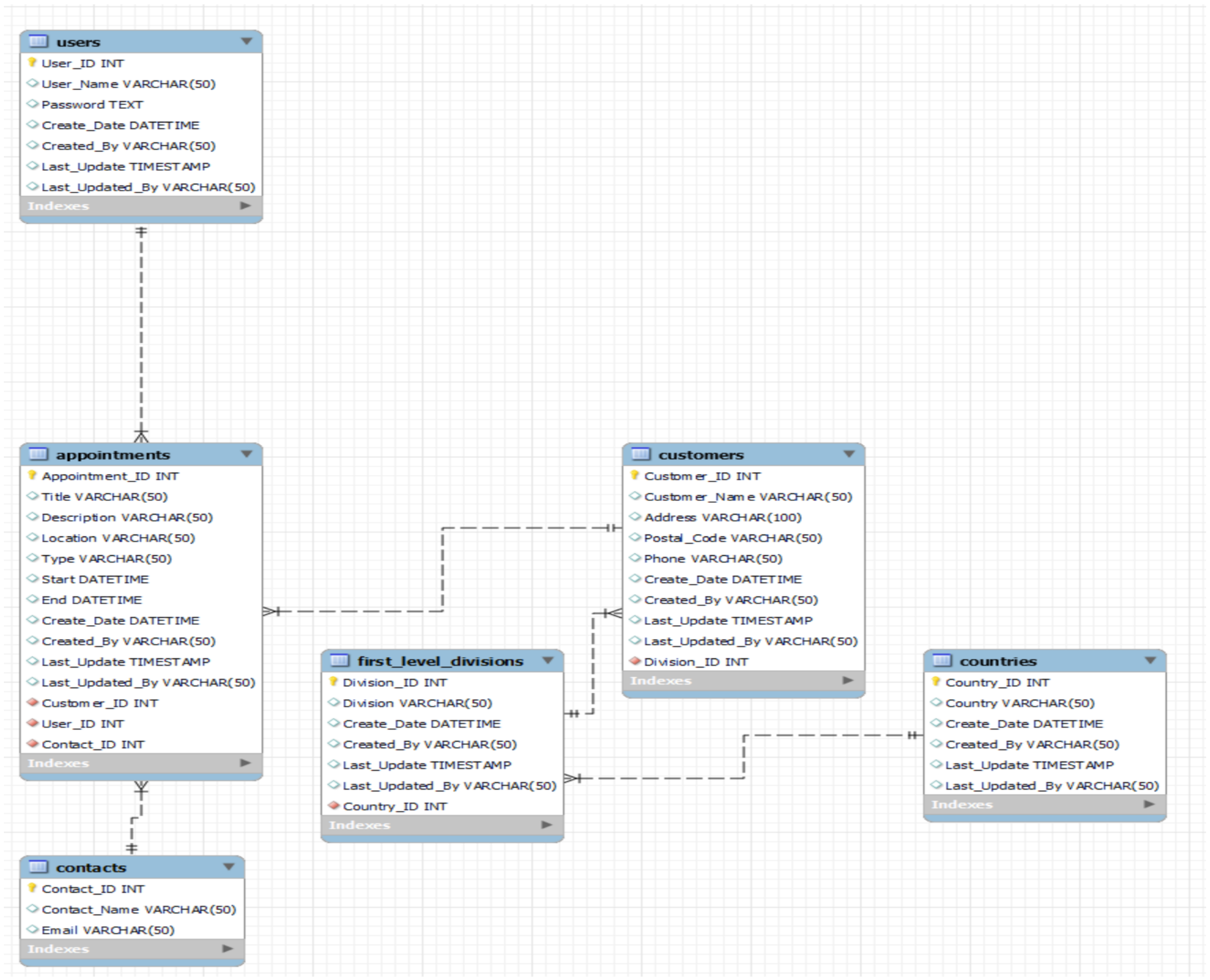
Table of Contents

Application Design and Testing

Design Document

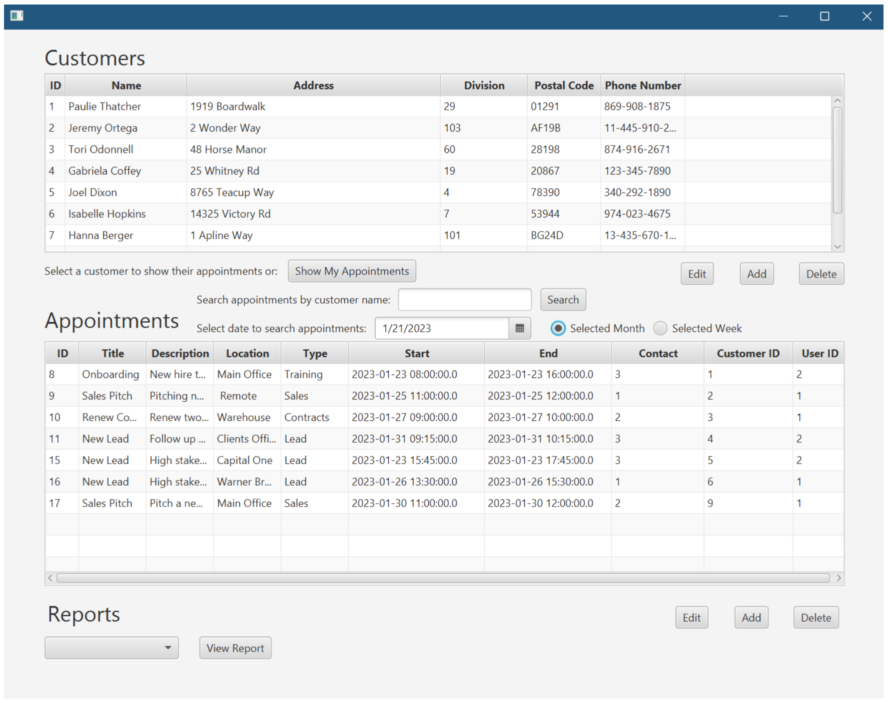
Class Design

The program's database consists of six tables, all connected with one-to-many relationships. The data focuses on the appointment table, which has connections to both the user and customer tables for reporting purposes. The appointment table directly connects with the users, contacts, and customer's table. Hence the foreign keys within the appointments table.



UI Design

The UI offers users a very simple way to add/edit customers and add appointments. Users can view their appointments by clicking the show my appointments button as well as searching customers by first name to see correlated appointments.



Customers

ID	Name	Address	Division	Postal Code	Phone Number
1	Paulie Thatcher	1919 Boardwalk	29	01291	869-908-1875
2	Jeremy Ortega	2 Wonder Way	103	AF19B	11-445-910-2...
3	Tori Odonnell	48 Horse Manor	60	28198	874-916-2671
4	Gabriela Coffey	25 Whitney Rd	19	20867	123-345-7890
5	Joel Dixon	8765 Teacup Way	4	78390	340-292-1890
6	Isabelle Hopkins	14325 Victory Rd	7	53944	974-023-4675
7	Hanna Berger	1 Apline Way	101	BG24D	13-435-670-1...

Select a customer to show their appointments or:

Search appointments by customer name:

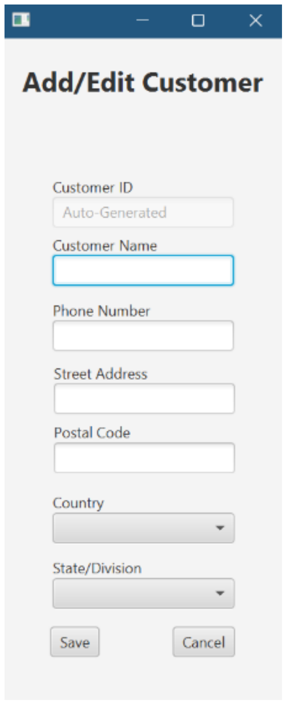
Appointments

Select date to search appointments: 1/21/2023

[✓ Selected Month] Selected Week

ID	Title	Description	Location	Type	Start	End	Contact	Customer ID	User ID
8	Onboarding	New hire t...	Main Office	Training	2023-01-23 08:00:00.0	2023-01-23 16:00:00.0	3	1	2
9	Sales Pitch	Pitching n...	Remote	Sales	2023-01-25 11:00:00.0	2023-01-25 12:00:00.0	1	2	1
10	Renew Co...	Renew two...	Warehouse	Contracts	2023-01-27 09:00:00.0	2023-01-27 10:00:00.0	2	3	1
11	New Lead	Follow up ...	Clients Offi...	Lead	2023-01-31 09:15:00.0	2023-01-31 10:15:00.0	3	4	2
15	New Lead	High stake...	Capital One	Lead	2023-01-23 15:45:00.0	2023-01-23 17:45:00.0	3	5	2
16	New Lead	High stake...	Warner Br...	Lead	2023-01-26 13:30:00.0	2023-01-26 15:30:00.0	1	6	1
17	Sales Pitch	Pitch a ne...	Main Office	Sales	2023-01-30 11:00:00.0	2023-01-30 12:00:00.0	2	9	1

Reports



Add/Edit Customer

Customer ID: Auto-Generated

Customer Name

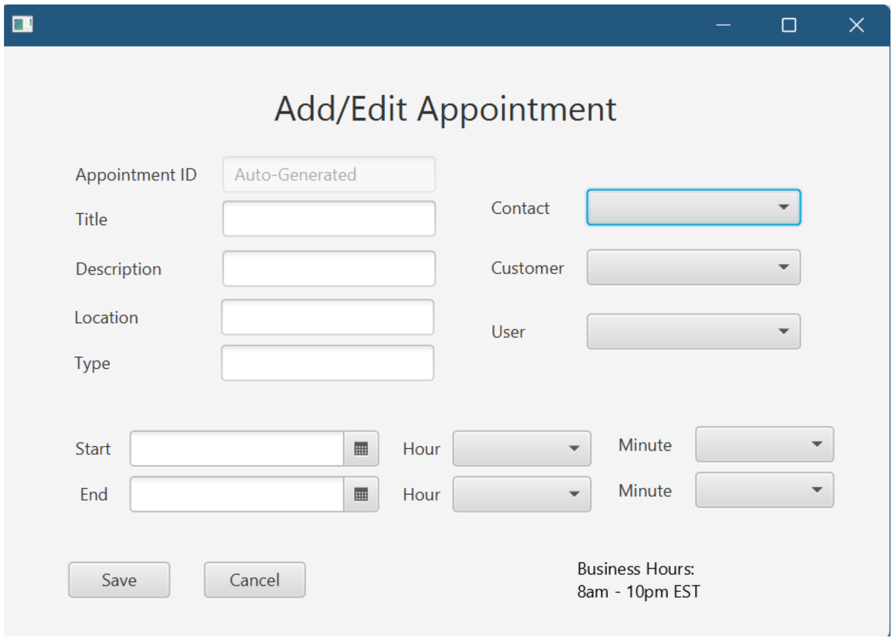
Phone Number

Street Address

Postal Code

Country

State/Division



Add/Edit Appointment

Appointment ID: Auto-Generated

Title

Description

Location

Type

Contact

Customer

User

Start Hour Minute

End Hour Minute

Business Hours:
8am - 10pm EST

Unit Test Plan

Introduction

Purpose

I developed and carried out unit tests for all public database access methods using Junit5. The purpose of the test was to ensure the functionality of database commands such as save, delete, edit, and search.

Overview

The program’s main purpose is to manage data stored in a SQL database by performing operations such as saving, editing, searching, viewing, manipulating, and deleting through a GUI interface. I had already tested the program's functionality in practice, so the next step was to demonstrate that it works correctly through unit testing. This process of testing helped me make corrections to methods that were not working correctly. For example, a major issue I had was the application was not retrieving database items after clicking the sort-by-month radio box. I was able to implement a fix all the tests passed.

Test Plan

Items

A test class is established for a particular real class, where JUnit testing code is written for all the public methods. After writing the test methods, I ran them by using the 'Run Test' command in the IDE, which executed each method in the test class and displayed the results in a separate screen.

Features

In order to verify its functionality, each examination must establish a direct link to the data repository, establish some initial parameters, perform the primary function utilizing these parameters and then match the outcome with a predefined outcome. When the outcome matches the predefined result, the examination is considered successful. On the other hand, if the outcome doesn't match, it could be an indication of a problem with either the primary function or the examination script.

Deliverables

Each evaluation that passes, returns a statement indicating its success, including the duration of the evaluation. A collection of evaluations that have all passed will provide a success rate to the developer.

Tasks

A significant portion of the effort in creating a comprehensive set of evaluations is dedicated to the composition of the evaluation scripts. Once the scripts are error-free, the process of conducting evaluations can be done by executing a single 'Run Test' command on each of the examination groups.

Needs

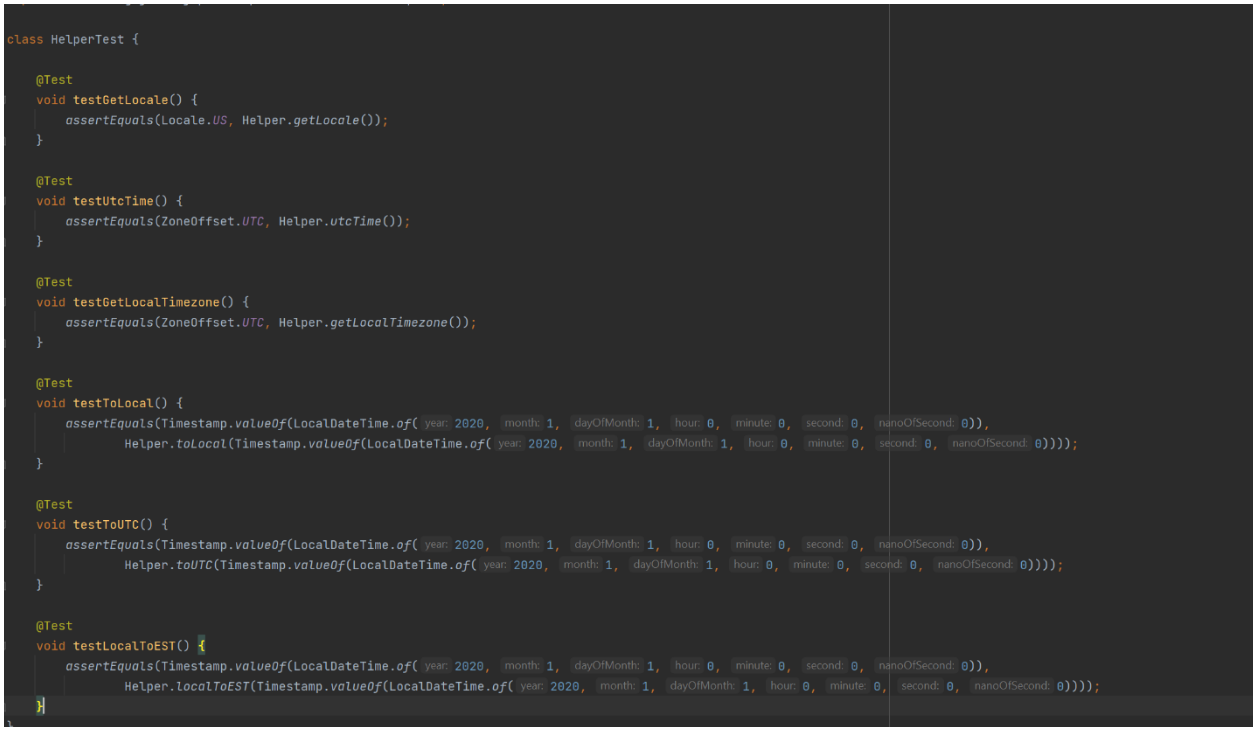
To ensure accurate results, each examination group must establish a direct connection to the same SQL database as the class being examined. I employed Junit5, a popular Java-based evaluation framework commonly supported by most integrated development environments, to compose the evaluations. The functionalities provided by this framework in my preferred IDE, IntelliJ, were utilized to execute the evaluations.

Pass/Fail Criteria

At the conclusion of each test, the outcome of the primary function and the predefined outcome are compared. If they match, the test is considered successful. If they don't match, it would indicate an unsuccessful test. It's important to ensure that the test doesn't return a false positive result.

Specifications

A few tests I ran were to test the time conversions from EST to UTC and from UTC to the systems local timezone



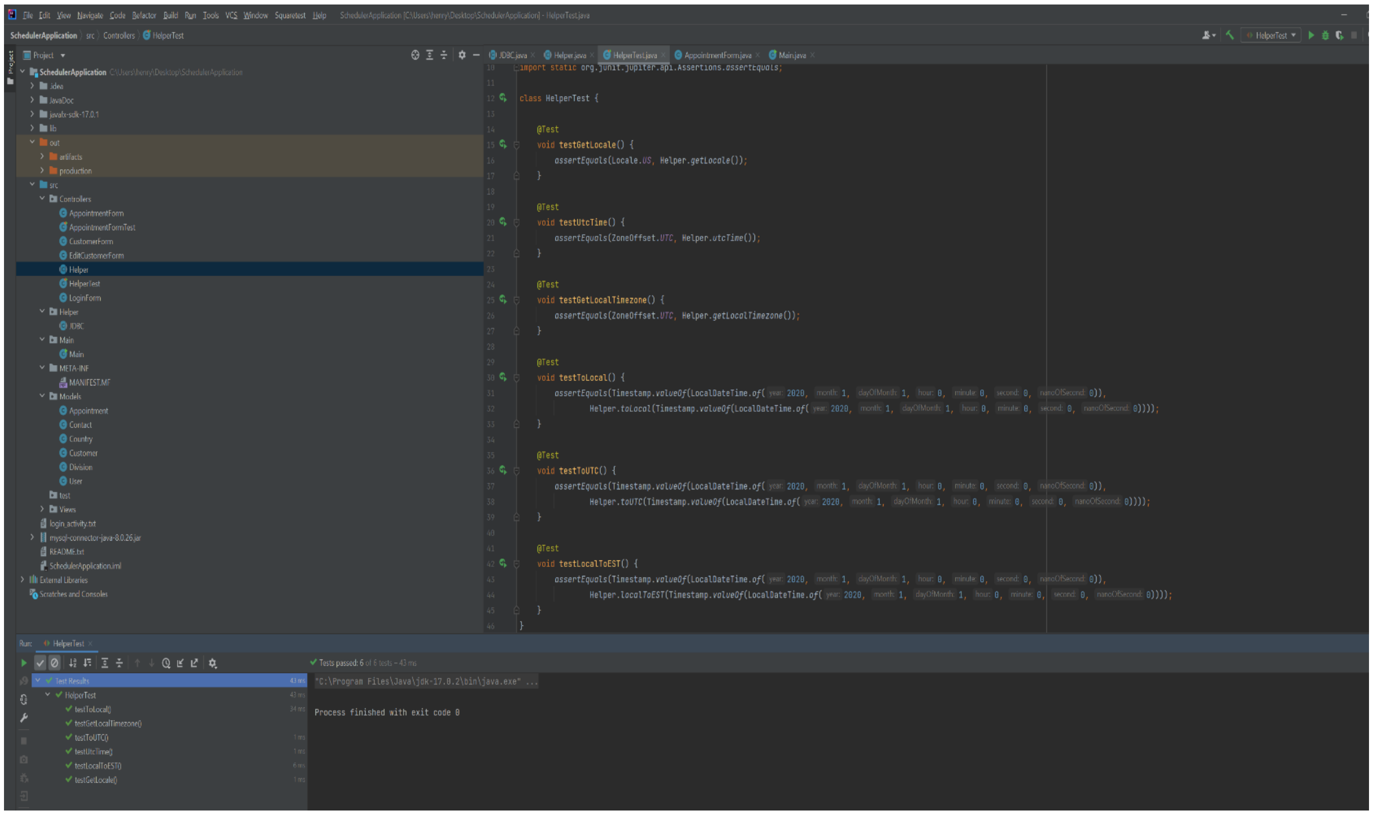
```
class HelperTest {

    @Test
    void testGetLocale() {
        assertEquals(Locale.US, Helper.getLocale());
    }

    @Test
    void testUtcTime() {
        assertEquals(ZoneOffset.UTC, Helper.utcTime());
    }

    @Test
    void testGetLocalTimezone() {
        assertEquals(ZoneOffset.UTC, Helper.getLocalTimezone());
    }

    @Test
    void testToLocal() {
        assertEquals(Timestamp.valueOf(LocalDateTime.of( year: 2020, month: 1, dayOfMonth: 1, hour: 0, minute: 0, second: 0, nanoOfSecond: 0)),
                Helper.toLocal(Timestamp.valueOf(LocalDateTime.of( year: 2020, month: 1, dayOfMonth: 1, hour: 0, minute: 0, second: 0, nanoOfSecond: 0))));
    }

    @Test
    void testToUTC() {
        assertEquals(Timestamp.valueOf(LocalDateTime.of( year: 2020, month: 1, dayOfMonth: 1, hour: 0, minute: 0, second: 0, nanoOfSecond: 0)),
                Helper.toUTC(Timestamp.valueOf(LocalDateTime.of( year: 2020, month: 1, dayOfMonth: 1, hour: 0, minute: 0, second: 0, nanoOfSecond: 0))));
    }

    @Test
    void testLocalToEST() {
        assertEquals(Timestamp.valueOf(LocalDateTime.of( year: 2020, month: 1, dayOfMonth: 1, hour: 0, minute: 0, second: 0, nanoOfSecond: 0)),
                Helper.localToEST(Timestamp.valueOf(LocalDateTime.of( year: 2020, month: 1, dayOfMonth: 1, hour: 0, minute: 0, second: 0, nanoOfSecond: 0))));
    }
}
```

Procedures

1. Determine which classes and methods require unit testing.
2. Create a corresponding test class for the chosen class to be evaluated.
3. Develop test cases for each public method within the original class within the test class.
4. Execute the tests and evaluate any failed test cases. Identify and correct the necessary changes.
5. Verify that all test cases have passed. Review and document the results.
6. Repeat the process for all classes that require testing.
7. Repeat the testing process on a regular basis to ensure that the code remains functional and to catch any new bugs or issues that may arise.
8. Make sure that the test cases cover all possible scenarios and edge cases to ensure that the code is thoroughly tested.
9. Utilize code coverage tools to measure the percentage of code that is covered by the test cases, and aim for a high coverage percentage.
10. Consider using test-driven development (TDD) where tests are written before the actual code, to ensure that the code is written in a testable way and all necessary functionality is covered by tests.

Results

The first time running these tests there was a problem with the conversion between EST to UTC resulting in a bad conversion to the systems local time. I was able to fix the conversion calculation of the method and the tests passed.



```java
import static org.junit.jupiter.api.Assertions.assertEquals;

class HelperTest {

    @Test
    void testGetLocale() {
        assertEquals(Locale.US, Helper.getLocale());
    }

    @Test
    void testUtcTime() {
        assertEquals(ZoneOffset.UTC, Helper.utcTime());
    }

    @Test
    void testGetLocalTimezone() {
        assertEquals(ZoneOffset.UTC, Helper.getLocalTimezone());
    }

    @Test
    void testToLocal() {
        assertEquals(Timestamp.valueOf(LocalDateTime.of( year: 2020, month: 1, dayOfMonth: 1, hour: 0, minute: 0, second: 0, nanoOfSecond: 0)),
                Helper.toLocal(Timestamp.valueOf(LocalDateTime.of( year: 2020, month: 1, dayOfMonth: 1, hour: 0, minute: 0, second: 0, nanoOfSecond: 0))));
    }

    @Test
    void testToUTC() {
        assertEquals(Timestamp.valueOf(LocalDateTime.of( year: 2020, month: 1, dayOfMonth: 1, hour: 0, minute: 0, second: 0, nanoOfSecond: 0)),
                Helper.toUTC(Timestamp.valueOf(LocalDateTime.of( year: 2020, month: 1, dayOfMonth: 1, hour: 0, minute: 0, second: 0, nanoOfSecond: 0))));
    }

    @Test
    void testLocalToEST() {
        assertEquals(Timestamp.valueOf(LocalDateTime.of( year: 2020, month: 1, dayOfMonth: 1, hour: 0, minute: 0, second: 0, nanoOfSecond: 0)),
                Helper.localToEST(Timestamp.valueOf(LocalDateTime.of( year: 2020, month: 1, dayOfMonth: 1, hour: 0, minute: 0, second: 0, nanoOfSecond: 0))));
    }
}
```

```
Tests passed: 6 of 6 tests – 43 ms

Test Results              43 ms
  HelperTest              43 ms
    testToLocal()         34 ms
    testGetLocalTimezone()
    testToUTC()           1 ms
    testUtcTime()         1 ms
    testLocalToEST()      6 ms
    testGetLocale()       1 ms

"C:\Program Files\Java\jdk-17.0.2\bin\java.exe" ...

Process finished with exit code 0
```

C4. Source Code

All source code and files are included in the SchedulerApplication folder within the zip file. The database includes two user's admin and test. The user "admin" has a password of "admin", and the "test" user has a password of "test"

User Guide

Introduction

Discover the ultimate guide for Marvel Sales Co's Sales Management System. This guide will take you through the process of setting up and logging into the system, as well as providing detailed instructions on how to utilize the various tools and functions. Use it as an initial reference or as a go-to resource as you become more familiar with the program. Going through this guide when you first start using the system will make the learning process smoother.

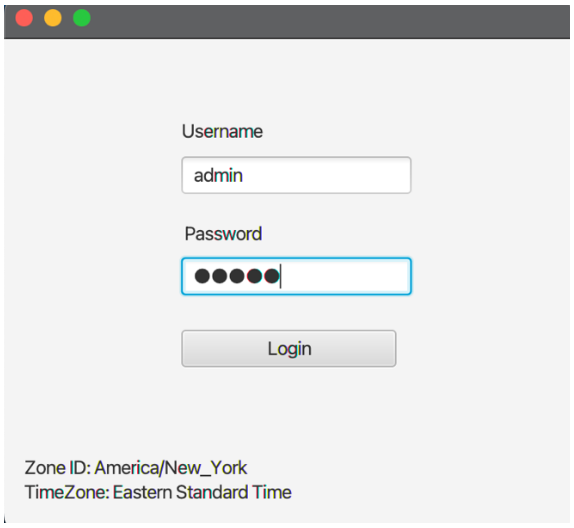
Installation and Using the Application

The application needs to be run through IntelliJ. Download the Zip file named SchedulerApplication and unzip it.

1. The JavaFX SDK and SQL connector libraries are included in the zip. Open the project in IntelliJ and add the libraries.
    a. Do this by going into File → Project structure → Libraries →
    b. Now you can add the provided libraries to the project
2. For the application to run properly, the run configurations need to be updated.
    a. Run → Edit Configurations → Modify Options → Select VM Options
    b. In the VM options copy --module-path /Users/henryventura/Desktop/SchedulerApplication/javafx-sdk-17.0.1/lib --add-modules=javafx.controls,javafx.fxml
    c. Replace the path of the command above with the path of the JavaFX SDK lib folder provided in the project folder.
3. You should now be able to run the application.

Login Screen

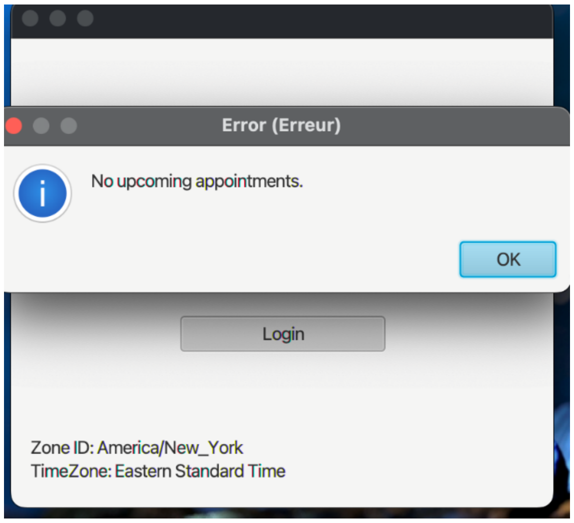
1. The database has two user accounts. “test” and “admin”.
2. To log in use either “test” or “admin” as the username and “test” or “admin” as the password.
3. The login screen shows you your time zone information and that corresponds to the appointment times you’ll see in the application.



admin

●●●●●

Zone ID: America/New_York
TimeZone: Eastern Standard Time



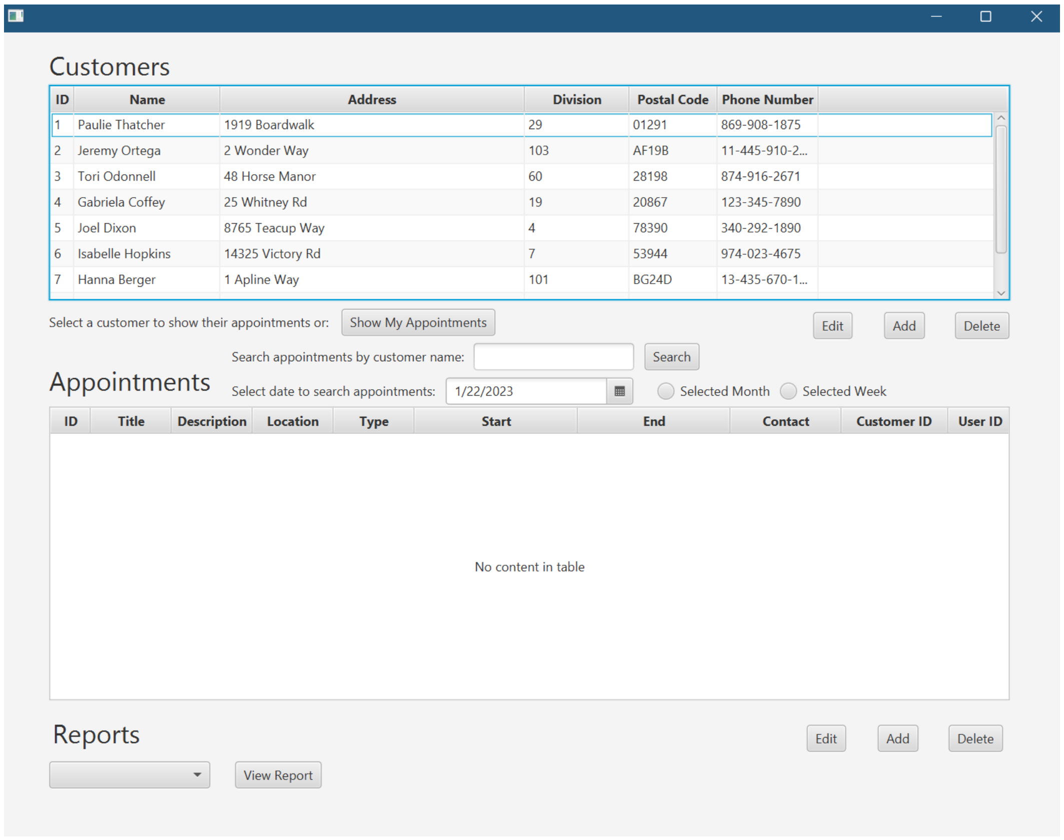
No upcoming appointments.

Zone ID: America/New_York
TimeZone: Eastern Standard Time

4. Once logged in, you’ll be notified of any upcoming appointments.

Appointment and Customer Screen

1. In the main screen you can see which customers have upcoming appointments by clicking on their name in the table. You can also search their name in the search box.
2. The reports dropdown menu can generate three different reports.
3. The “Show My Appointments button” will populate the appointments table with appointments correlated with the logged in user.



Customers

ID	Name	Address	Division	Postal Code	Phone Number
1	Paulie Thatcher	1919 Boardwalk	29	01291	869-908-1875
2	Jeremy Ortega	2 Wonder Way	103	AF19B	11-445-910-2...
3	Tori Odonnell	48 Horse Manor	60	28198	874-916-2671
4	Gabriela Coffey	25 Whitney Rd	19	20867	123-345-7890
5	Joel Dixon	8765 Teacup Way	4	78390	340-292-1890
6	Isabelle Hopkins	14325 Victory Rd	7	53944	974-023-4675
7	Hanna Berger	1 Apline Way	101	BG24D	13-435-670-1...

Select a customer to show their appointments or: Show My Appointments

Edit Add Delete

Search appointments by customer name: Search

Appointments

Select date to search appointments: 1/22/2023

Selected Month Selected Week

ID	Title	Description	Location	Type	Start	End	Contact	Customer ID	User ID

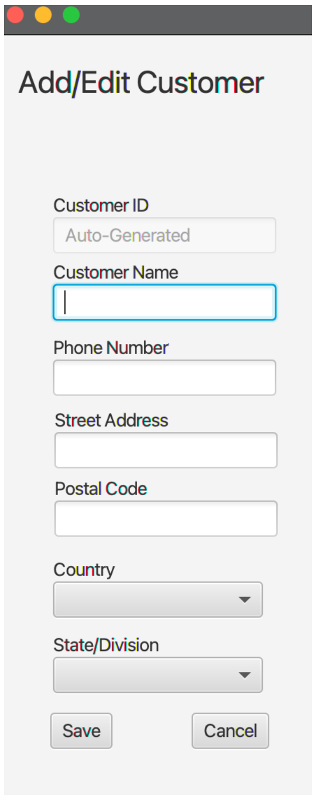
No content in table

Reports

Edit Add Delete

View Report

4. The "Selected Month" and "Selected Week" radio buttons sort appointments by the month or week that was selected in the date picker.
5. You can edit/delete customers or appointments by selecting the item from the table and clicking the edit or delete button.
6. The add button in the customer table will take you to the add customer form. Here you can fill in the customer's name, phone number, street address, postal code, country, and state. All entries need to be filled in for the customer to be saved.



Add/Edit Customer

Customer ID: Auto-Generated

Customer Name

Phone Number

Street Address

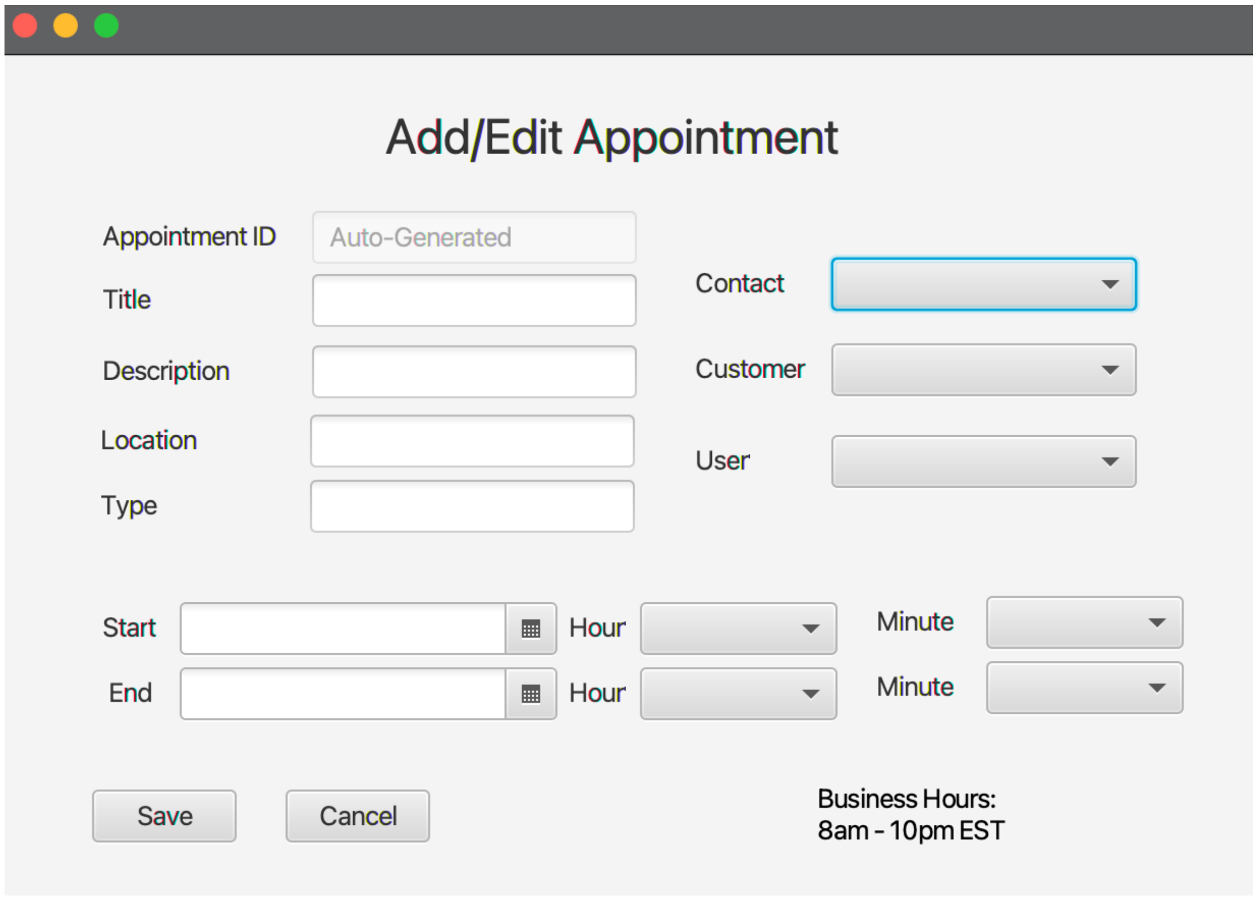
Postal Code

Country

State/Division

Save Cancel

7. The add/edit appointment form will allow you to create an appointment and correlate it with a customer, user, and contact.
8. Appointments can only be set during business hours and cannot overlap with existing appointments.
9. All fields need to be filled out in order to save the appointment.



Add/Edit Appointment

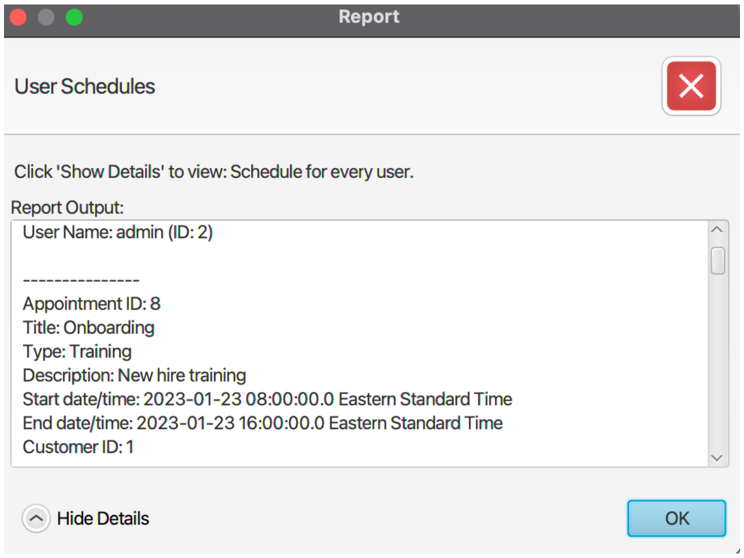
Appointment ID	Auto-Generated		
Title		Contact	
Description		Customer	
Location		User	
Type			

Start		Hour		Minute	
End		Hour		Minute	

Business Hours:
8am - 10pm EST

Reports

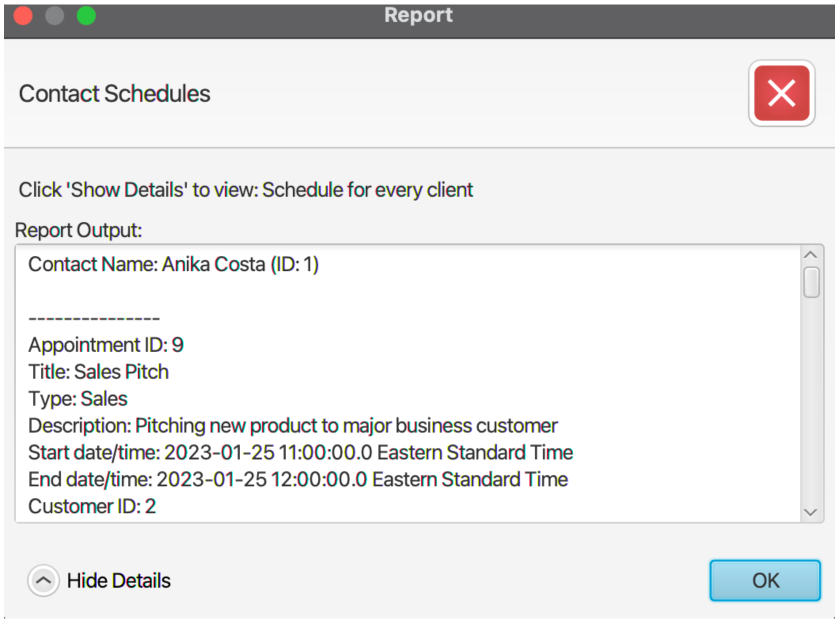
1. To access the reporting feature, from the Main screen, click on the drop-down men on the bottom left and select a report. Once selected press the view report button
2. The user schedule report shows the upcoming appointments filtered by account users.
    a. The report shows the appointment ID as well as the title, type, description, start date and time, and end date and time.



Click 'Show Details' to view: Schedule for every user.
Report Output:

```
User Name: admin (ID: 2)

--------------
Appointment ID: 8
Title: Onboarding
Type: Training
Description: New hire training
Start date/time: 2023-01-23 08:00:00.0 Eastern Standard Time
End date/time: 2023-01-23 16:00:00.0 Eastern Standard Time
Customer ID: 1
```

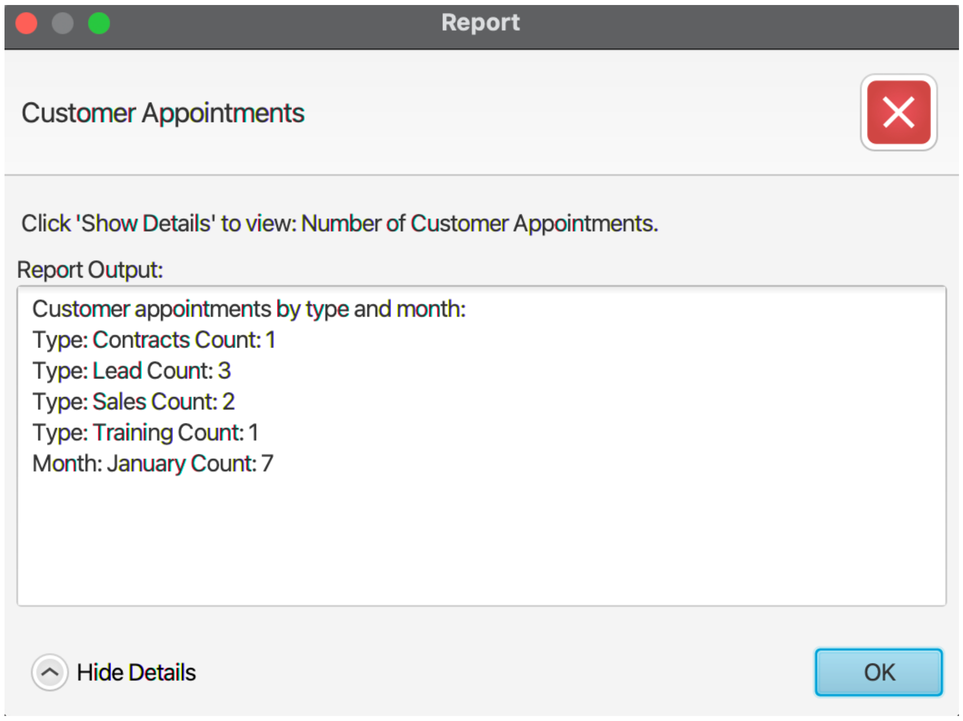
3. The contact schedules report filters a report based on the appointments associated with the business contacts.



Click 'Show Details' to view: Schedule for every client
Report Output:

```
Contact Name: Anika Costa (ID: 1)

---------------
Appointment ID: 9
Title: Sales Pitch
Type: Sales
Description: Pitching new product to major business customer
Start date/time: 2023-01-25 11:00:00.0 Eastern Standard Time
End date/time: 2023-01-25 12:00:00.0 Eastern Standard Time
Customer ID: 2
```

4. The customer appointments report shows all scheduled appointments by type and month. You can specify the type of appointment in the add appointment form.



Click 'Show Details' to view: Number of Customer Appointments.

Report Output:

```
Customer appointments by type and month:
Type: Contracts Count: 1
Type: Lead Count: 3
Type: Sales Count: 2
Type: Training Count: 1
Month: January Count: 7
```