**C868 – Software Capstone Project Summary**

**Task 2 – Section C**

|  |  |
| --- | --- |
| **Capstone Proposal Project Name:** | http://www.idevnews.com/views/images/uploads/general/wgu_logo.png  RentApp: A Rental Property Appointment Scheduler |
| **Student Name:** | Jeffrey Linn |

**Table of Contents**

Contents

[Table of Contents 2](#_Toc94266427)

[RentApp Scheduler 4](#_Toc94266428)

[Application Design and Testing 4](#_Toc94266429)

[Design Document 4](#_Toc94266430)

[Class Design 4](#_Toc94266431)

[UI Design 6](#_Toc94266432)

[Unit Test Plan 8](#_Toc94266433)

[Introduction 8](#_Toc94266434)

[Purpose and Overview 8](#_Toc94266435)

[Test Plan 8](#_Toc94266436)

[Items 8](#_Toc94266437)

[Features 9](#_Toc94266438)

[Pass/Fail Criteria 10](#_Toc94266439)

[Tasks & Procedures 10](#_Toc94266440)

[Specifications 11](#_Toc94266441)

[Results 13](#_Toc94266442)

[Source Code 15](#_Toc94266443)

[User Guide 16](#_Toc94266444)

[Introduction 16](#_Toc94266445)

[Installation and Using the Application 16](#_Toc94266446)

[*System Requirements* 16](#_Toc94266447)

[*Installation* 16](#_Toc94266448)

[Login (For Admin User) 17](#_Toc94266449)

[Appointments 17](#_Toc94266450)

[*Filter Views of Appointments Table* 17](#_Toc94266451)

[*Create a New Appointment* 18](#_Toc94266452)

[*Modify an existing Appointment* 18](#_Toc94266453)

[*Delete an existing Appointment* 19](#_Toc94266454)

[Customers 19](#_Toc94266455)

[*Create a New Customer* 19](#_Toc94266456)

[*Modify an existing Customer* 19](#_Toc94266457)

[*Delete an existing Customer* 20](#_Toc94266458)

[Navigation 20](#_Toc94266459)

[*Log Out* 20](#_Toc94266460)

[*Exit* 20](#_Toc94266461)

[Reports 21](#_Toc94266462)

RentApp Scheduler

# Application Design and Testing

# Design Document

## Class Design

Above is an image relating to the class design that was used for this application. The program development followed the model, view, controller architecture, and was enhanced by Utility and Helper classes (not pictured) for database connection and method organization.

There are seven basic classes: JDBC (Java Database Connection), Users, Appointments, Customers, Contacts, Division, and Countries. The JDBC class provides the interface to connect with the database to manipulate the data. Private methods are denoted by a minus (-) sign and public methods with a plus (+) sign.

Users will be the client’s staff who will utilize the application and manipulate the data by adding, editing, or deleting both Appointments and Customers. Customer information includes address, phone, name, and the ID, which is autogenerated by the database. Appointment information includes a title, description, start and end times, and likewise its ID is also autogenerated by the database.

Division is the first-level state or province of a particular country where a rental property is located. Available Division selections are filtered based on the Country ID. Most importantly, the start and end times of appointments will be automatically validated to not overlap in the given time-zone of the Appointment.

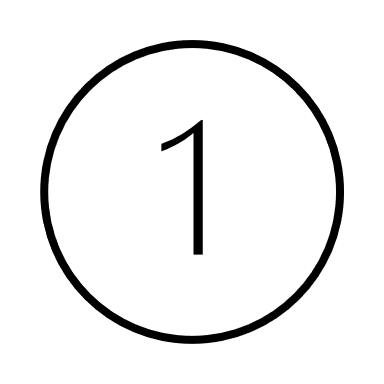
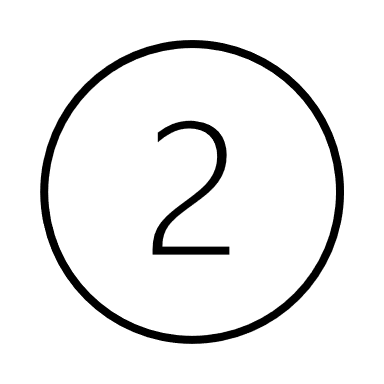
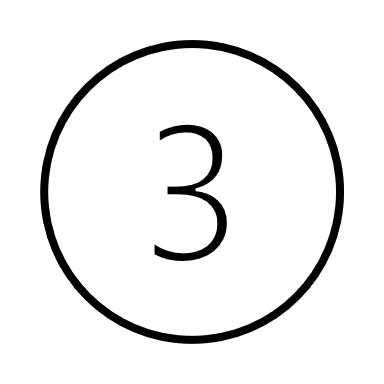
Prospects who are interested in scheduling an appointment to view a rental property are considered Customers. The User is the client office administrator who is in communication with the Customer to establish and set an Appointment in the database. The Customer will have a designated Contact in their Division/Country who will meet them physically to show the rental property. That Contact will be assigned to the Customer by the User for local scheduling purposes. Thus, a User can set an Appointment for a Customer to meet a Contact in a Division/Country to view a rental property at an agreeable time in their time-zone.

## Badge 5 with solid fillBadge 3 with solid fillBadge with solid fillBadge 4 with solid fillBadge 1 with solid fillDiagram Description automatically generated with low confidenceUI Design

Above is an image of the low-fidelity user interface design showing the general flow of the functions, buttons, and pages. There were initially planned to be five or more pages in the application. The final design, however, is only three pages: Login, Welcome, and Reports. Below is a high-fidelity image of the Login, Welcome and Reports designs. The data manipulation functions were made to fit onto the Welcome page to help make the application more streamlined. The Login page allows for language and region recognition, which will facilitate in converting languages and times to the proper geographic time-zone.

Graphical user interface, text, application

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence**Graphical user interface, text, application, email

Description automatically generated**

Login Page to access database of appointments and customers

Welcome Page showing Appointment Table and Customer Table

Reports Page with available reporting features

# Unit Test Plan

## Introduction

### Purpose and Overview

One of the most logic-heavy methods in the application is the “getOverlapList” method, which determines if there is a conflicting appointment time. Due to the high potential for errors in the code logic, it was crucial to fully test and validate the functionality of saving appointments – whether they’re being updated or newly added. To that end, seven conditions were developed and tested to fully validate potential overlaps when a user clicks the Save button to either update or save an appointment.

## Test Plan

### Items

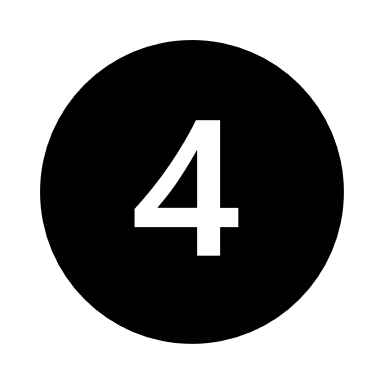
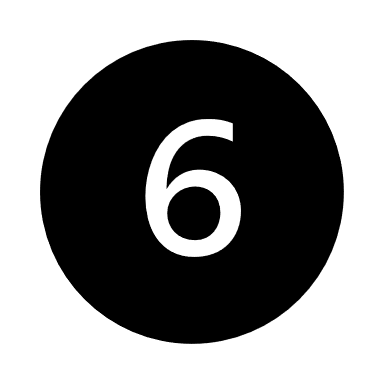
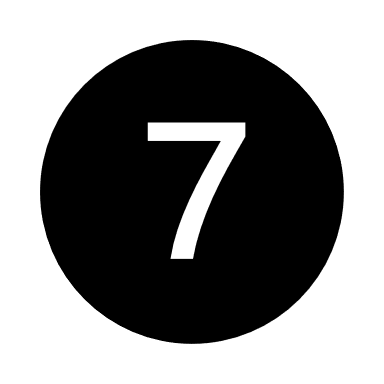
To completely test the overlap method, there had to be two primary categories. First, overlaps when *adding* a new appointment. Second, overlaps when *modifying* an existing appointment. The difference between the two is the presence of an Appointment ID. When creating a new appointment, it must not overlap with any other existing appointments. This particular code was unique because with updating an existing appointment, it must not overlap with others, but also cannot count itself in that logic.

Finally, aside from adding a new appointment (blank Appointment ID), an existing appointment was needed against which to test updating. This means it would already be in the database and have an Appointment ID. Regardless of Customer or the User, the Appointment ID would always be unique. In other words, one Customer could have more than one appointment without error.

### Features

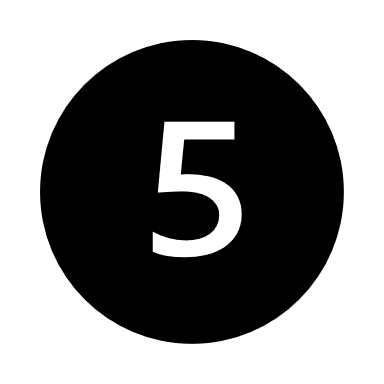
The seven conditions mentioned above were first drawn on paper as a blueprint of the code logic. It was a representation of all the possible scenarios an appointment could be considered an overlap with another. These were used to form the code of each condition. The following images show the seven separate conditions as drawings of “A” appointments overlapping with “B” ones, and the next picture shows them written in the program.

A



B

Text

Description automatically generated

During testing I utilized “System.out.println()” to print out exactly where errors were as they arose. Sometimes, unfortunately, there were false positives. For example, with the logic of condition #4 and #5, I had initially only entered code to validate the edges on the outsides of the appointment times. This did not compare end-time A with *both* the end-time B *and* start-time B. Therefore, unlike the other conditions, #4 and #5 needed a second comparative operator to pass the tests.

### Pass/Fail Criteria

Failing the test would result in the Save function fully processing and inserting the appointment data into the database despite an overlap. Passing the test would result in the Save function denying the process, showing an alert dialog to the user, and printing out a line to the system as to where the error occurred in the conditions.

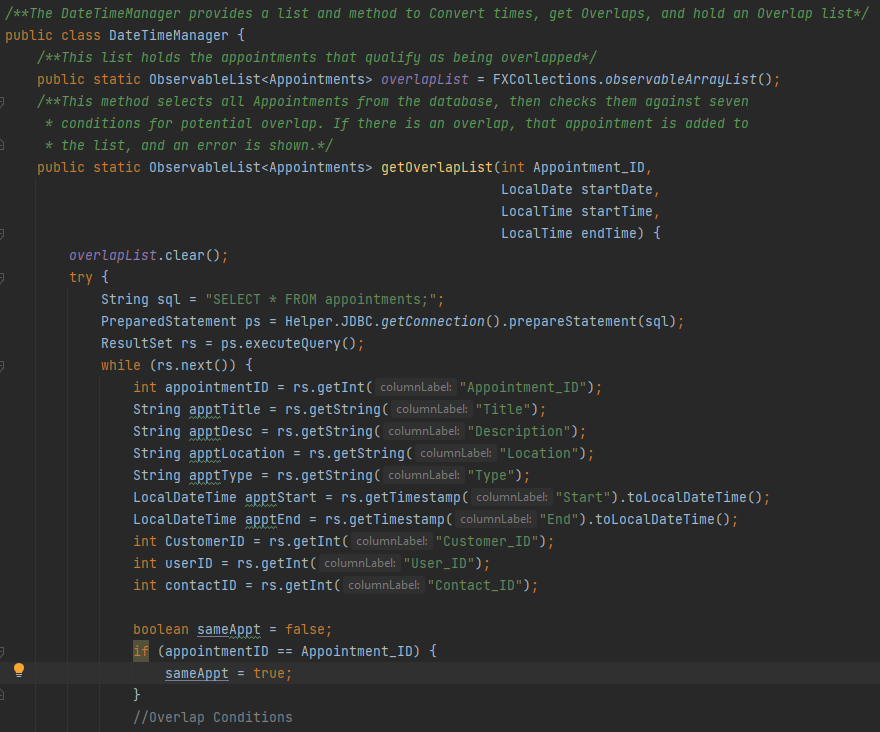
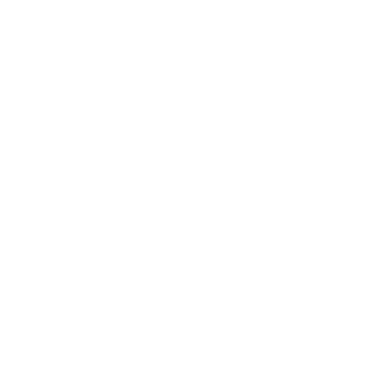
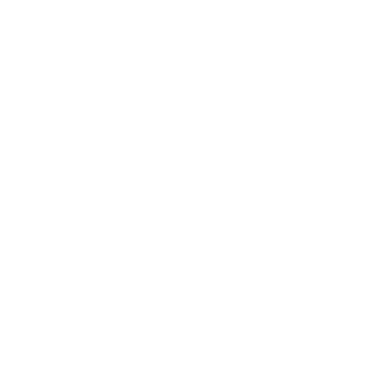
## Tasks & Procedures

Using alert dialogs and print lines in each condition, I was able to test each condition and category. Category #1 was tested by attempting to add a new appointment using a start-time and end-time that would purposely fail condition #1. That condition code was then edited with each test fail. Upon passing the test, the process was repeated with the proceeding conditions. After all seven conditions were passed in category #1, then category #2 was tested in the same manner until all conditions had passed.

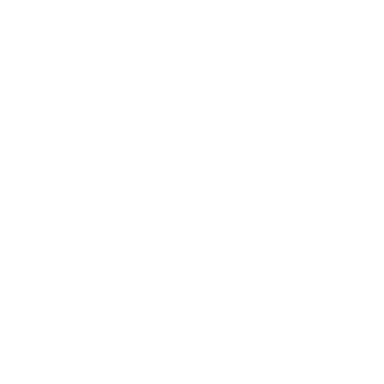
Each condition was tested one by one, and incrementally added on top of the prior conditions. As each new overlap condition was entered into the program, I ran the program and input a new appointment (no appointment ID) and attempted to save an invalid time expecting an error. For example, on overlap condition #3, I attempted to save a time that started at the same time as an existing appointment but ended prior to the other’s end time. Then I attempted to save a valid time. With each successful trial, the next conditional code was added.

The process for testing the logic with modifying appointments was the same. The only difference in category #2 was the built in validation for the Appointment ID being a match, and the appointments being on the same day. If those conditions were passed, then the seven conditions loop would be entered.

## Specifications

These next two images show the full testing code used for this method.

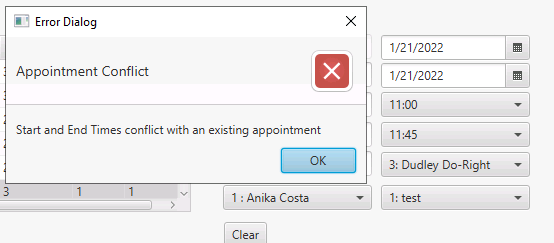
1. Establishes the method “getOverlapList” to populate the list “overlapList.”
2. After clearing the overlapList, the database is accessed for the appointment information as listed with the getInt, getString, and getTimestamp assignments.
3. The Boolean variable is assigned and validated by comparing the appointmentID from the database list versus the Appointment\_ID selected in the application. *Continued…*

Text

Description automatically generated

1. The validation logic is extended. If the selected appointment ID is not the same as any appointment ID’s in the database, *and* the appointments *are going to be* on the same day, then the if-loop is entered for further validation checks.
2. If the appointment meets any of the seven overlap criteria as listed in #4, the appointment is added to the overlapList and considered an “overlap.”
3. An alert dialog lets the User know there is a conflict in the proposed times.
4. The method returns the overlapList.

## Results

When any condition test fails, here is the resulting alert dialog to the User:

Graphical user interface, text, application

Description automatically generated

And the User is prompted to check their entries:

Graphical user interface, text, application

Description automatically generated

When any condition test passes, here is the resulting confirmation dialog:

Graphical user interface, text, application

Description automatically generated

And the System prints the following message:

**Text

Description automatically generated**While developing the code, a NullPointerException error was encountered when attempting to add an appointment. Due to a blank text field for the Appointment ID the method was trying to parse an empty String and convert it to an Integer. This threw the following error message:

Since only new appointments have blank appointment ID’s, this was not a problem for Updating (Category #2 testing). The solution required changing the parameter code to a String of “0” and this resolved the NullPointerException error. See correction here:

Graphical user interface, text

Description automatically generated

# Source Code

For all source files and code please compressed folder titled “RentApp – JLINN.”

# User Guide

## Introduction

The RentApp application will provide the ability for a user to login with a user-name and password, add/edit/delete both appointments and customers, as well as to view reports. This guide will include how to install, log into, and use all the functions of the application. Certain system requirements must exist. Note: the database used is in WGU Virtual Lab Environment.

## Installation and Using the Application

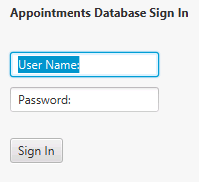
### *System Requirements*

* 64-bit versions of Microsoft Windows 10,8
* 2 GB RAM minimum, 8 GB RAM recommended
* 2.5 GB hard disk space, SSD recommended
* 1024x768 minimum screen resolution
* Java JDK version Java SE 11.0.13
* Java FX version compatible with JDK 11 (JavaFX jdk-11.0.13)
* MySQL Connector Driver (mysql-connector-java-8.1.23)
* MySQL Workbench 8.0.25
* IntelliJ IDEA version 2021.1.3 (Community Edition)

### *Installation*

1. Download the compressed folder titled “RentApp-JLINN” to your desktop and unzip it.
2. Open IntelliJ, and click File, Open. Select the downloaded file.
3. Click OK. The project file will open.
4. Wait for the project to fully load.
5. Ensure the project has the appropriate SDK configuration and Libraries installed in IntelliJ.
6. Press the green arrow in the top right to run the program and log in.

## Login (For Admin User)

1. Enter your User Name and Password
2. **If you already have an account, log in with your account name and password. Otherwise, use “admin” in both text fields, then click the Sign In button.
3. If you need to create a personal user account, contact the database administrator to add your user profile to the database.

## Appointments

### *Filter Views of Appointments Table*

1. Once logged in, click on the radio buttons at the top to filter the appointment table.
   1. View All will show all the appointments in the database.
   2. By Month will show only appointments in the current month.
   3. By Week will show only appointments in the current week.
2. Graphical user interface

   Description automatically generated with medium confidenceThere is a Search bar at the top right of the appointments table as well. Enter the appointment ID number or appointment Title you wish to locate. If there is a match, it will be highlighted automatically.

### *Create a New Appointment*

1. To add a new Appointment to the database, first click the Clear button to the right of the appointment table. This will remove all data from the fields.
2. Graphical user interface

   Description automatically generatedGraphical user interface

   Description automatically generatedEnter the relevant information into all fields, combo boxes, start and end dates for the appointment, and start and end times. Note: the appointment must be on the same date and not different dates. Note: appointment times must not conflict with business hours. An alert will be shown if there are any conflicts.
3. Click the Save button to the left of the table. If the AppointmentID text field is blank, and there are no other errors, then a new Appointment will be created with a new ID.

### *Modify an existing Appointment*

1. Select an appointment from the appointments table.
2. Graphical user interface

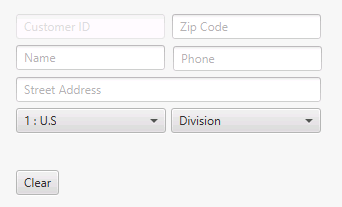
   Description automatically generatedClick the Modify button to the left of the appointments table. The information will be automatically populated into the text fields and boxes on the right. Note: the AppointmentID already exists so the appointment will be updated in the database.
3. When edits are complete, and there are no conflicts, click the Save button. Note: if you click the Clear button, it will erase the Appointment ID and a new appointment will be created upon Saving.

### Graphical user interface Description automatically generated*Delete an existing Appointment*

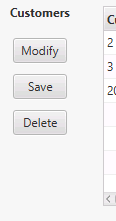
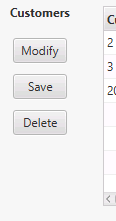
1. Select an appointment from the appointments table.
2. A confirmation dialog will appear, press OK.
3. The appointment will be deleted from the database.

## Customers

### *Create a New Customer*

1. To add a new Customer to the database, first click the Clear button to the right of the Customer table. This will remove all data from the fields.
2. Enter the relevant information into all fields, and combo boxes.
3. Click the Save button to the left of the table. If the CustomerID text field is blank, and there are no other errors, then a new Customer will be created with a new ID.

### *Modify an existing Customer*

1. Select a customer from the Customers table.
2. Click the Modify button to the left of the Customers table. The information will be automatically populated into the text fields and boxes on the right. Note: the Customer ID already exists so the customer will be updated in the database.
3. When edits are complete, and there are no conflicts, click the Save button. Note: if you click the Clear button, it will erase the Customer ID and a new Customer will be created upon Saving.

### *Delete an existing Customer*

1. Select a Customer from the customer table.
2. A confirmation dialog will appear, press OK.
3. The Customer will be deleted from the database.
4. If there are any existing appointments for the Customer, they must first be deleted before the Customer can be deleted.

## Navigation

### *Log Out*

At the bottom left of the Welcome page, there is a Log Out button. This will disconnect the user from the database and go back to the Log In page. The user will have to go through the log-in procedure again to get back to the Welcome page.

### *Exit*

Likewise, the Exit button, next to Log Out, will also disconnect from the database, but will close the program entirely. Clicking this button will ask for confirmation to exit and close.

Graphical user interface, text

Description automatically generated

## Reports

To access the reporting features in the application, simply click the Reports button in the bottom left corner. A new window will appear with three reports to view.

1. *Graphical user interface, text, application

   Description automatically generated****Report #1****:*
   1. Select a Contact person from the drop-down menu.
   2. The appointment table will be filtered to view appointments with only that contact person.
2. *Graphical user interface, website

   Description automatically generated****Report #2****:*
   1. Select a Month from the “Month” drop-down menu.
   2. Select a Type of appointment from the “Type” drop-down menu.
   3. Press the calculate button. The box will show the number of appointments of that selected Type in that selected Month.
3. ***Report #3****:*
   1. Graphical user interface, text, application

      Description automatically generatedSelect a Customer from the drop-down menu on the right.
   2. Graphical user interface, diagram, application

      Description automatically generatedThe box will show the number of appointments from that selected Customer.
4. Back Navigation: To go back to the Welcome Page, click the back button in the lower left corner above the Exit button.
5. Exit: To exit the program and close the window, click the Exit button and confirm.
6. **Login Activity**: to view a report of login activity (users, dates, times, and success):
   1. Graphical user interface, application

      Description automatically generatedSelect the projects file, projects folder, src folder
   2. Select the Files folder, and select “login\_activity.txt”