# C868 – Software Capstone Project Summary

# Task 2 – Section C

**Capstone Proposal Project Name:** Software Agency Scheduling Platform

**Student Name:** Jamey Mathis

## Table of Contents

*Table of Contents* ..... *2*

*Application Design and Testing* ..... *3*

*Design Document* ..... *3*

**Class Design** ..... **3**

**UI Design** ..... **6**

*Unit Test Plan* ..... *17*

**Introduction** ..... **17**

Purpose ..... 17

Overview ..... 17

Test Cases ..... 18

**Test Plan** ..... **19**

Items ..... 19

Features ..... 19

Deliverables ..... 20

Tasks ..... 20

Needs ..... 20

Pass/Fail Criteria ..... 20

**Procedures** ..... **20**

**Results** ..... **21**

*C4. Source Code* ..... *22*

*C5. Link to Live Version* ..... *22*

Task 2 Part C – C868 Software Development Capstone

# Application Design and Testing

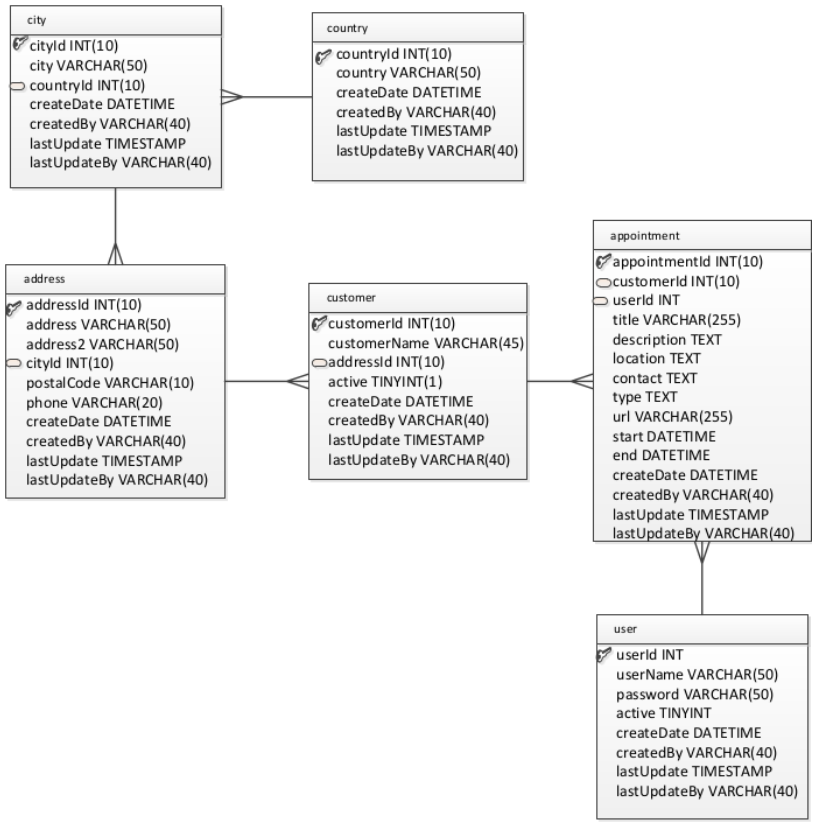
## Design Document

### Class Design

The proposed software consists of six data classes:

- User
  - Represents consultants/staff that will create customers and appointments
- Customer
  - Represents an individual customer record
- Appointment
  - Represents an individual appointment record
- Address
  - Represents the house number and street name of the customer
- City
  - Represents the city in which the customer is located
- Country
  - Represents the country in which the customer is located

This ERD was created based on requirements gathered from all stakeholders, and the data classes relate to each other as follows:



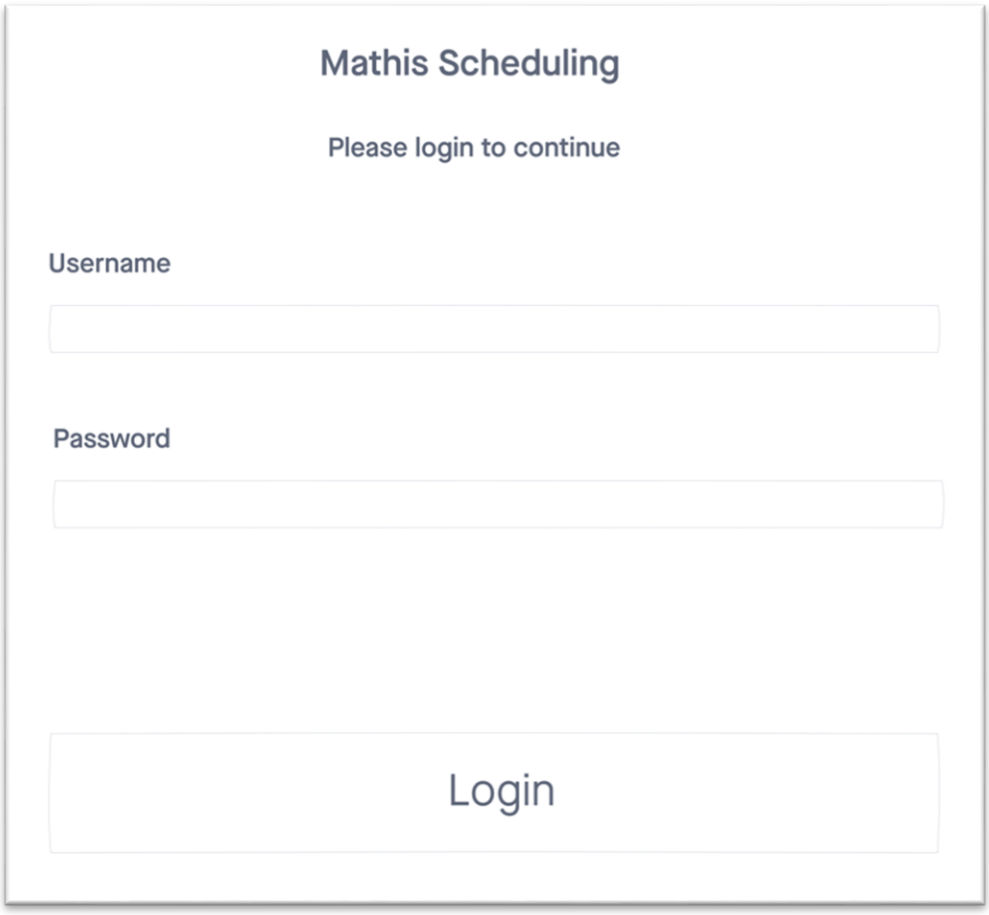

In the application source code, the address, city, and country classes are represented inside of the Customer class for simplicity and enhancing intuitive development.

**UI Design**

The proposed software consists of 11 UI “views” or “screens”. The UI is designed with user-friendliness and data integrity in mind to allow users of various knowledge levels to utilize the full suite of features offered by the application without causing damage to the database.

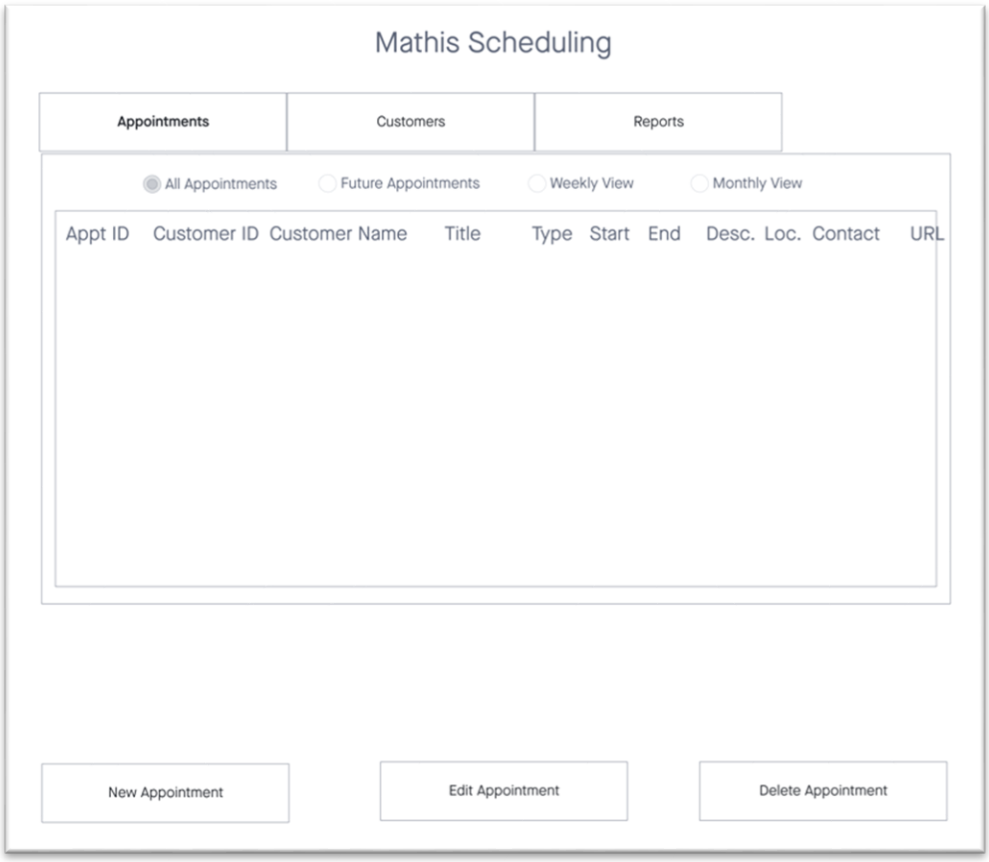
High-fidelity UI screenshots may be seen within the accompanying User Guide. These low-fidelity wireframes were created based on requirements gathered from all stakeholders:

**Login**



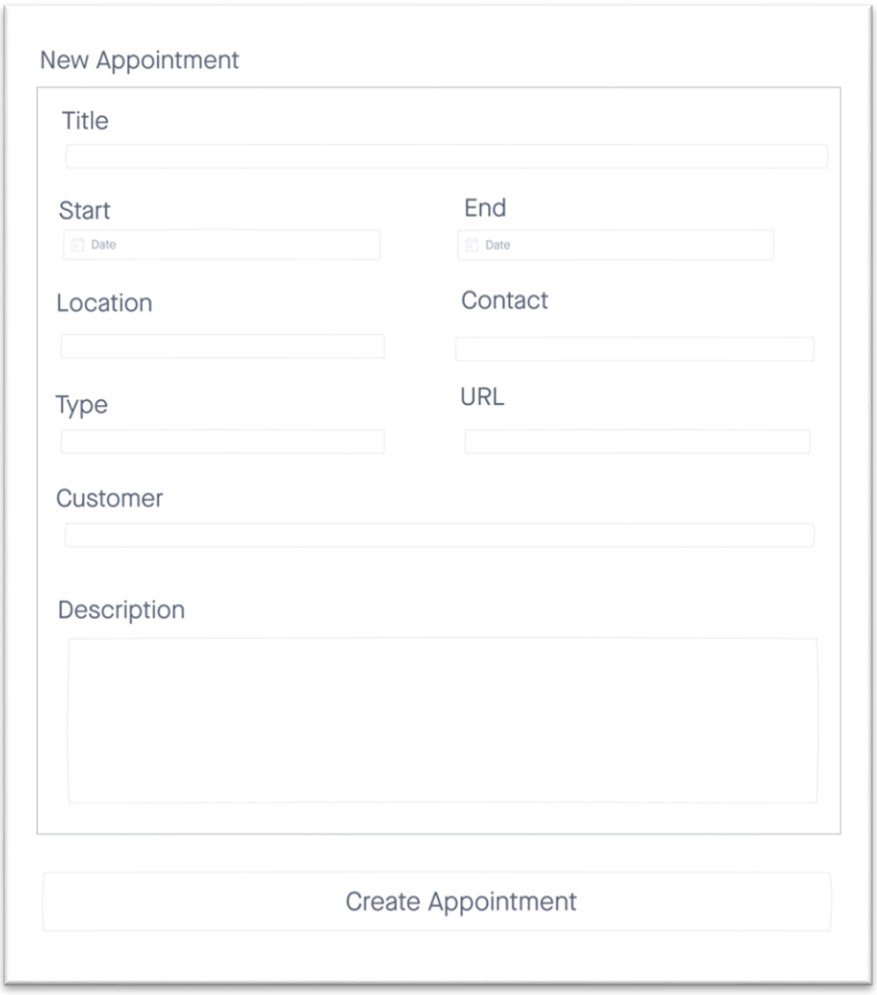

This screen restricts access to the rest of the application to authenticated Users.

**Appointment List**



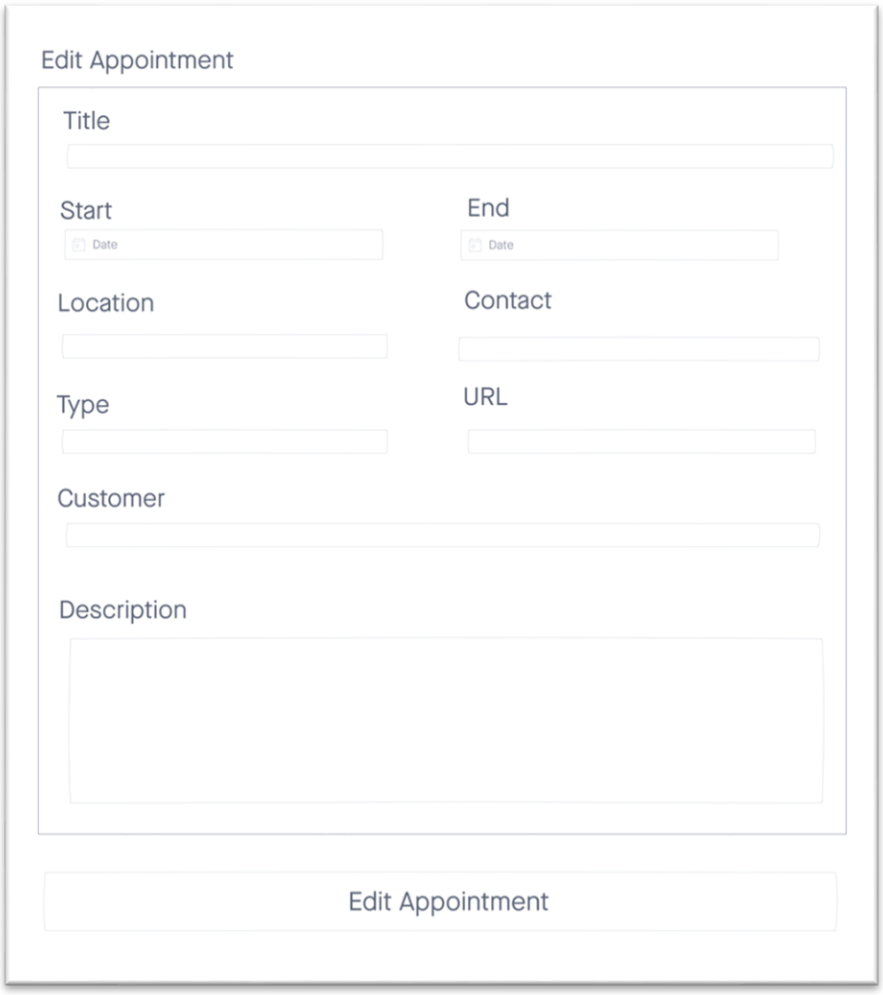

This screen lists all Appointments and may be filtered to show future appointments only, as well as supporting a weekly/monthly view.

**New Appointment**



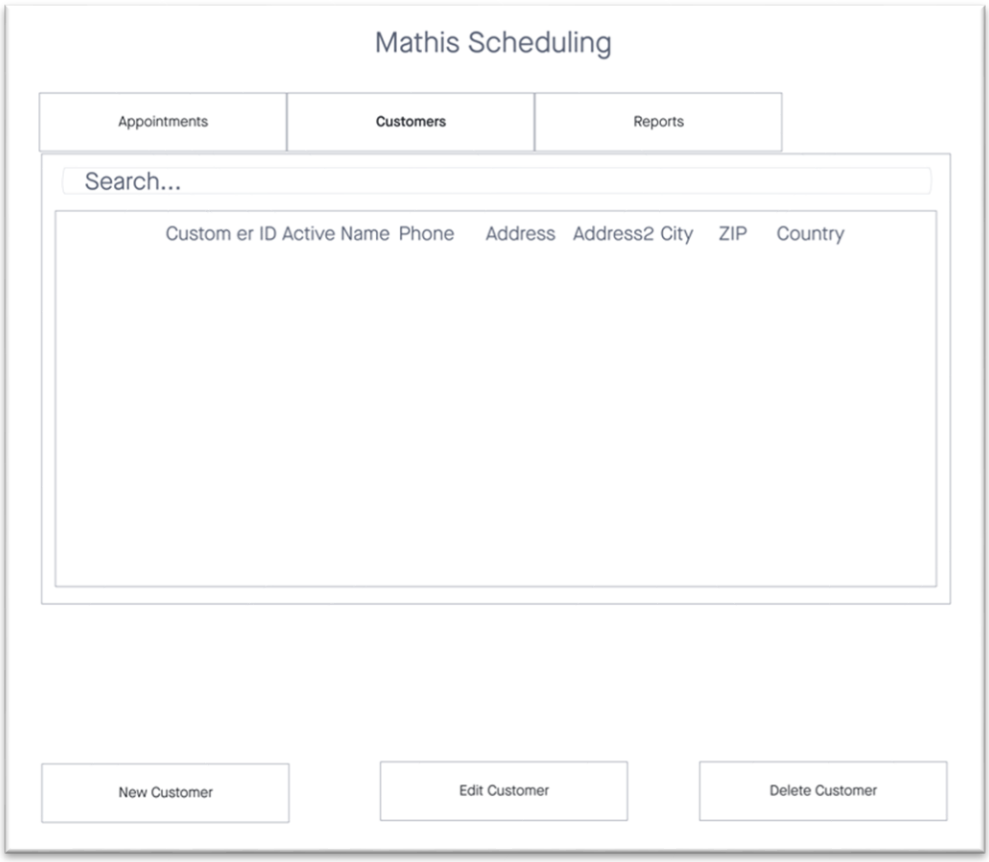

This screen allows the User to insert a new Appointment record.

**Edit Appointment**



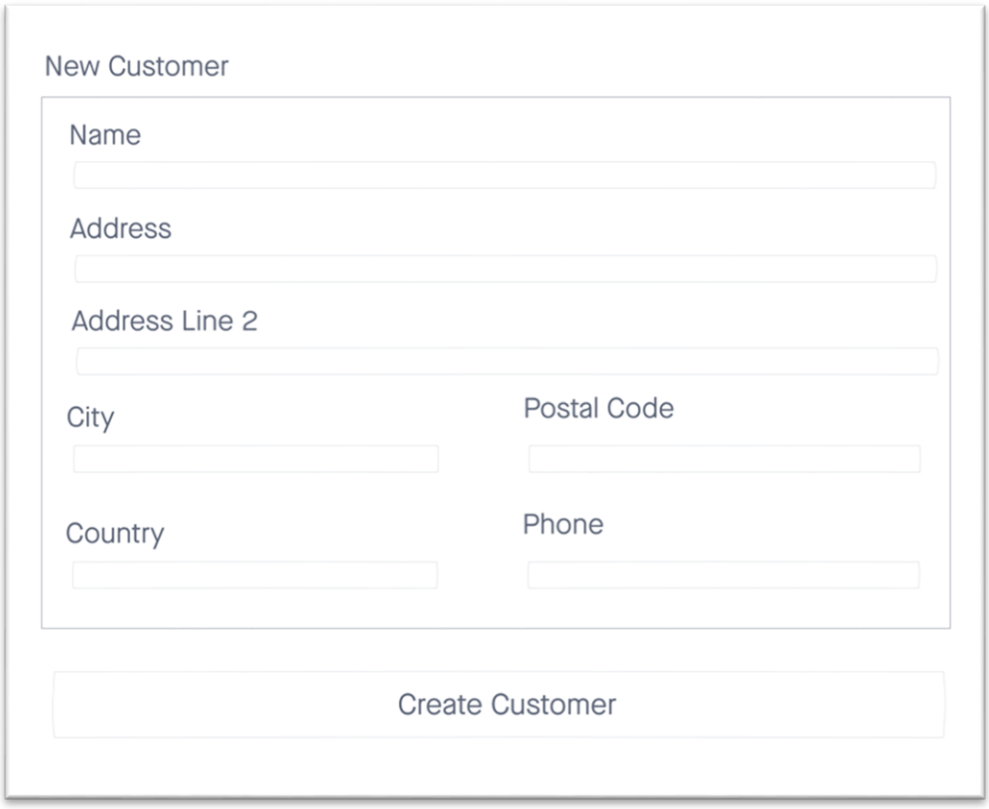

This screen allows the User to edit an existing Appointment record.

**Customer List**



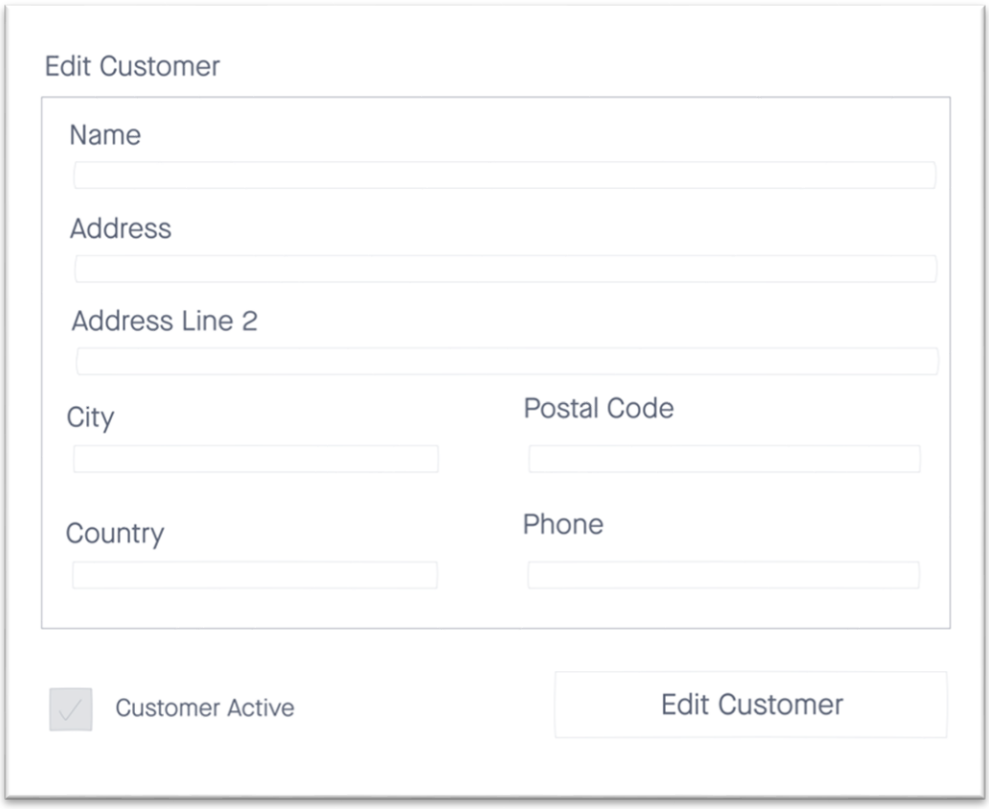

This screen lists all Customers, and may be searched by ID, Name, Phone, and Address.

**New Customer**



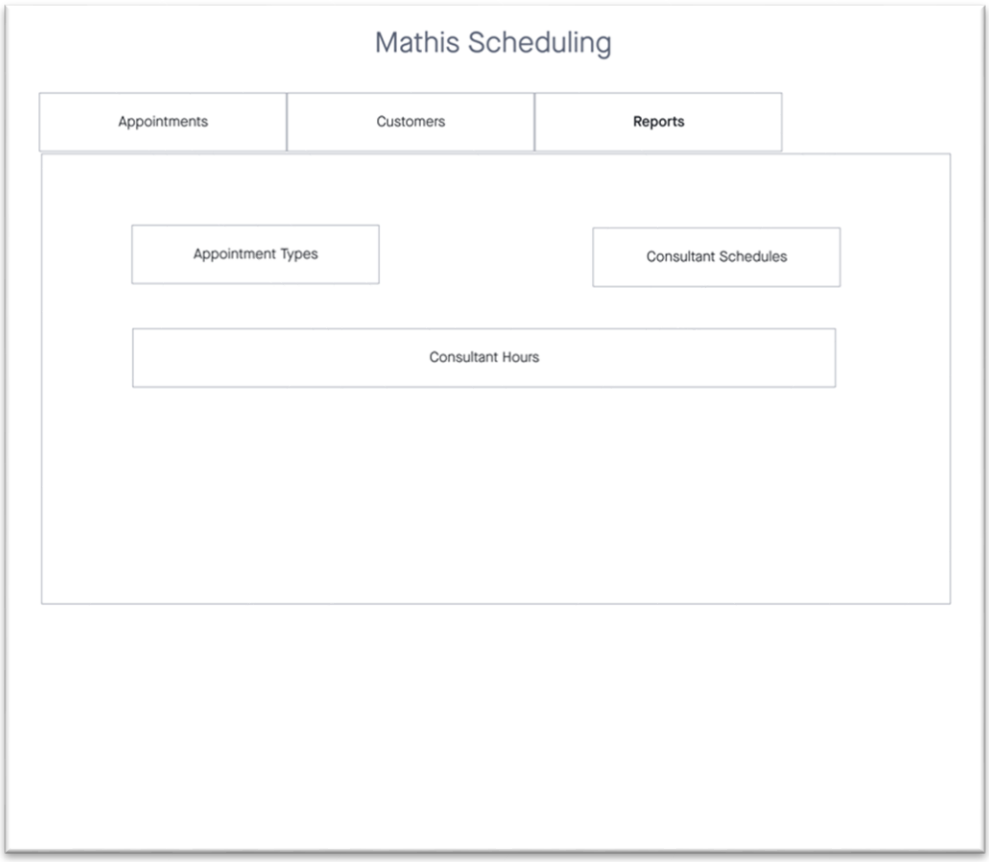

This screen allows the User to insert a new Customer record.

**Edit Customer**



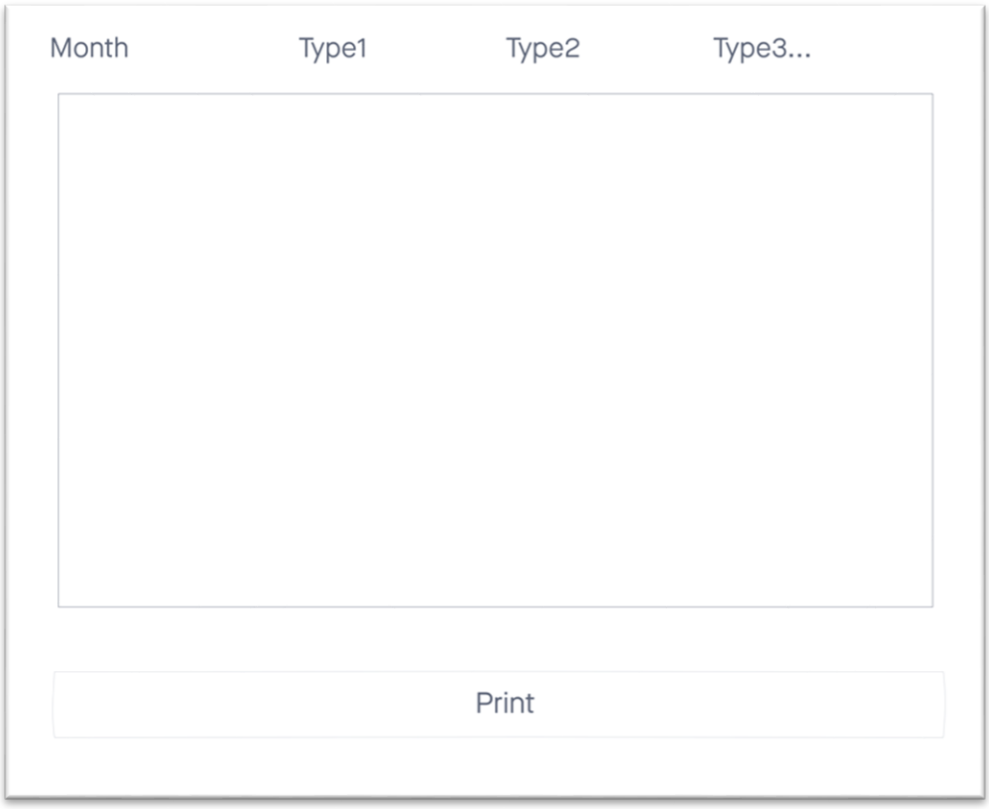

This screen allows the User to edit an existing Customer record.

**Reports**



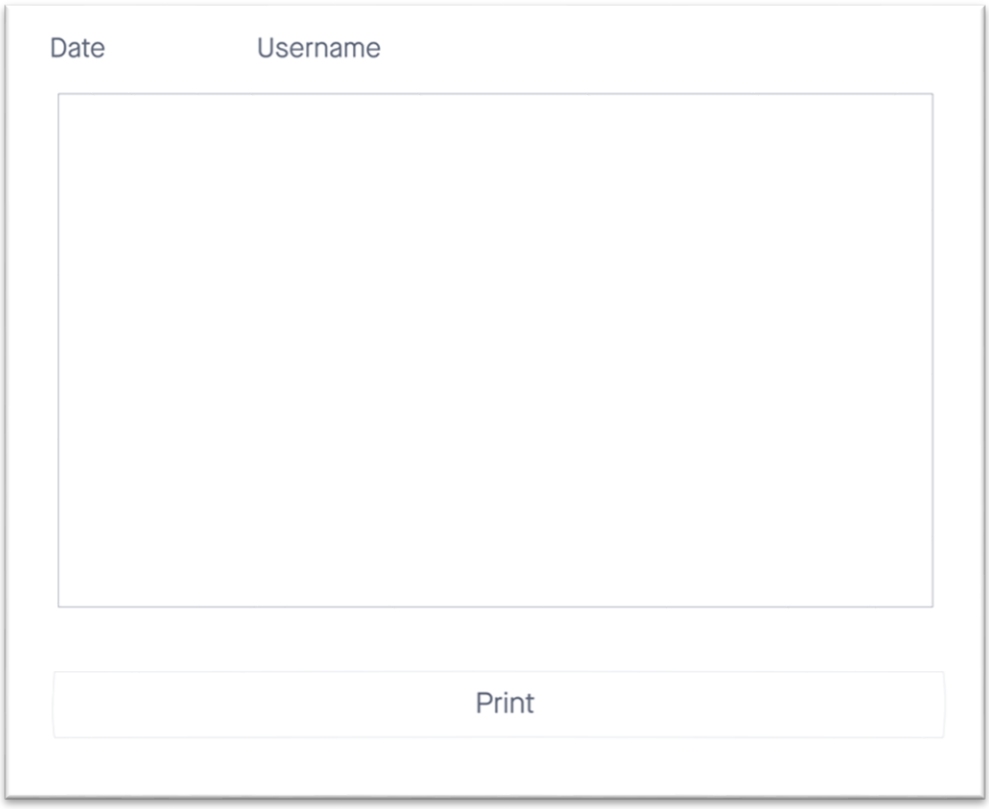

This screen contains buttons to generate the available Report types.

**Report: Appointment Types**



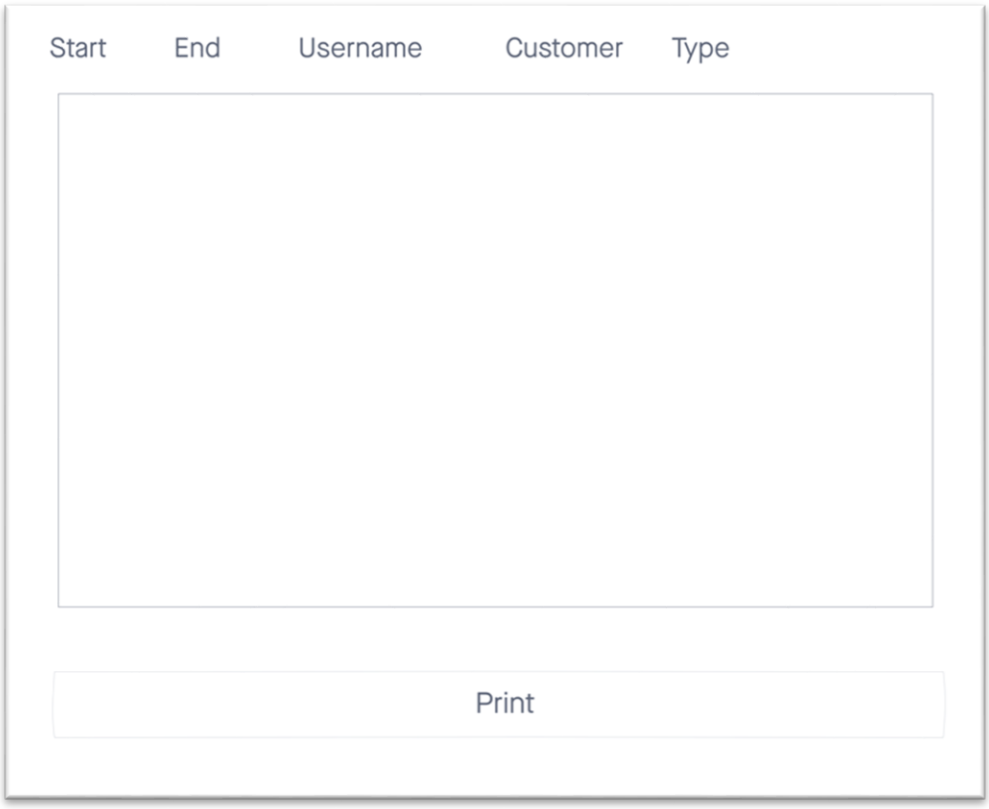

This screen displays a list of Appointment types by month.

**Report: Consultant Hours**




This screen displays each User’s scheduled hours by date.

**Report: Consultant Schedules**




This screen displays each User’s appointments.

# Unit Test Plan

## Introduction

### Purpose

The primary method used for testing was manual testing via the UI to verify the application conformed to specifications and requirements. Individual UI "screens" were considered to be "units" and were integrated into the application after passing the test case. If a bug was discovered, it was documented and assigned to a member of the development team for review.

### Overview

The tests were performed by manipulating the user interface to execute the test case, in which the database and UI were simultaneously tested to perform as expected in the test case. All appointment times were verified to be saved in UTC in the database for easy inter-office scheduling and displayed in the application user's local timezone.

The application's error handling was also tested, to ensure that data integrity was maintained in the event of failure.

**Test Cases**

| | Title | Description | Inputs | Expectation |
|---|---|---|---|---|
| 1 | Business Hours | Action to create appointments outside of business hours (9AM-5PM) should be prevented | Valid appointment data, except appointment time should be out of range | An error should be raised |
| 2 | Appointment Reminders | Users should be notified of appointments upcoming within 15 minutes after logging in | Successful login with appointment within 15 minutes | A notification should be shown regarding the appointment |
| 3 | Overlapping Appointments | Users should not be able to schedule appointments that overlap | Valid appointment data, except the appointment time should overlap an existing appointment | An error should be raised |
| 4 | Valid Customers | Users should be able to create a new customer by providing valid data | Valid customer data | A customer record should be created |
| 5 | Valid Appointments | Users should be able to create a new appointment by providing valid data | Valid appointment data | An appointment record should be created |
| 6 | Invalid Login | The application should block invalid logins | Invalid login credentials | An error should be raised |

## Test Plan

### Items

The application should be running inside the work environment and all tests must pass. In case of failure, the defect is documented and sent to a developer. This process repeats until there are no further failures.

### Features

The following functionality is tested:

- Login
- Search
- Weekly/monthly views
- Generate reports
- Create, view, update, and delete customers
- Create, view, update, and delete appointments

**Deliverables**

This test plan with all test cases and their results.

**Tasks**

- Perform the tasks described in the test cases
- Document any defects and report to the development team
- Repeat all test cases until there are no further failures

**Needs**

The environment needs to have .NET Framework 4.7.2 running on Windows 8 or above with a working Internet connection to successfully perform tests.

**Pass/Fail Criteria**

Each test case describes inputs and expected output; this is used to determine if the test results match what is expected as described by the test case.

**Procedures**

- Launch the application
- Enter the inputs in the test case
- Verify the output matches the test case expectation

**Results**

| Test Case | Result |
|---|---|
| 1. Business Hours | PASS |
| 2. Appointment Reminders | PASS |
| 3. Overlapping Appointments | PASS |
| 4. Valid Customers | PASS |
| 5. Valid Appointments | PASS |
| 6. Invalid Login | PASS |

## C4. Source Code

The source code may be found in the task submission, or on GitHub:

https://github.com/mathiscode/wgu-capstone-c868

## C5. Link to Live Version

A prebuilt release version may be found at:

https://github.com/mathiscode/wgu-capstone-c868/releases