C868 – Software Capstone Project Summary

Task 2 – Section C

Capstone Proposal Project Name: WeSchedule – A Customer / Appointment Scheduler

Student Name: James Blankenship

Table of Contents

Application Design and Testing

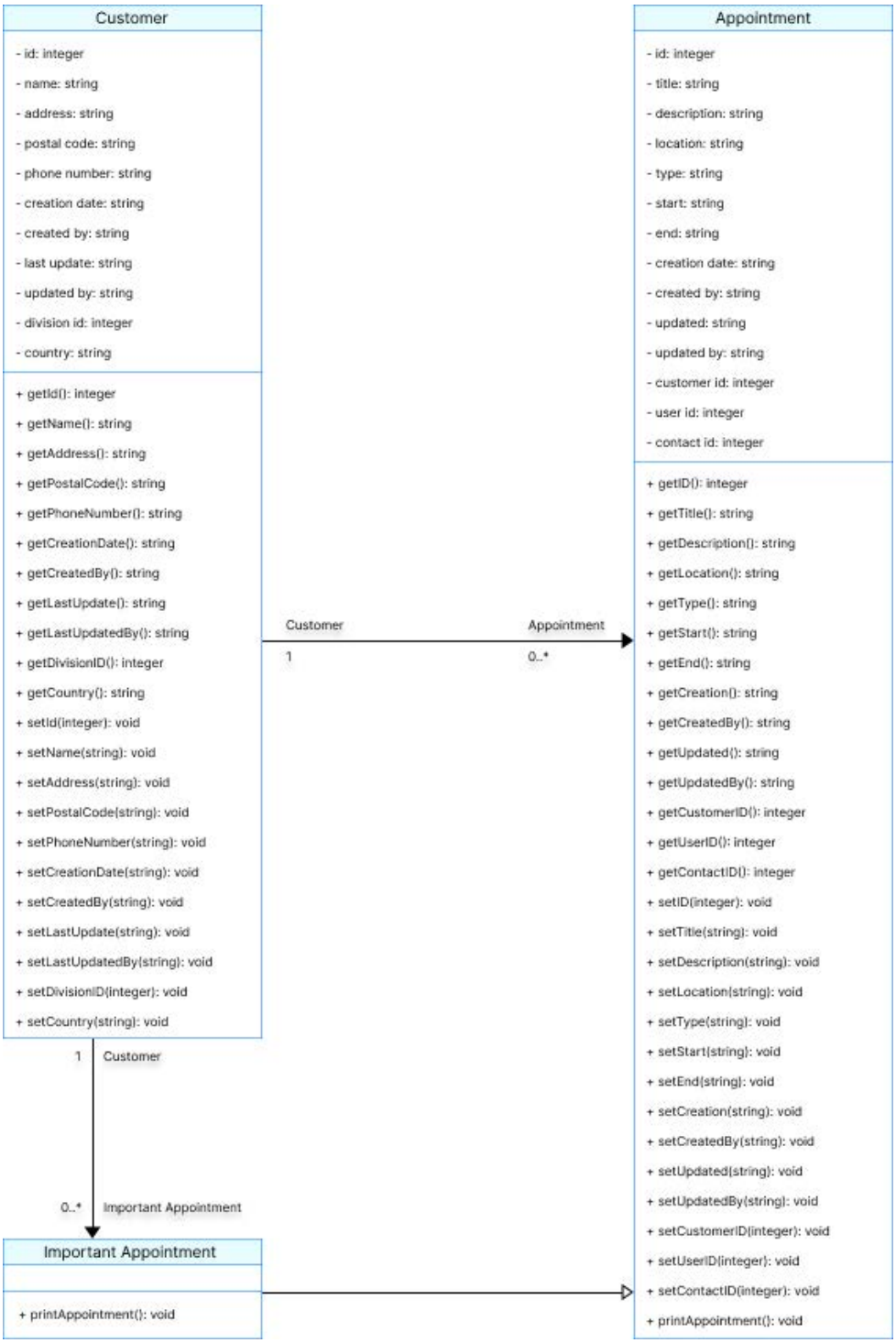
Design Document

Class Design

The illustration below represents the class design for the WeSchedule Java application. The classes shown directly map to the database tables that form the interior of the application's data.

The three main classes are Customer, Appointment, and Important Appointment. There's a one-to-many relationship between Customer and Appointment as well as Customer and Important Appointment. Conversely, there's a necessary many-to-one relationship between Appointment and Customer (an appointment may only belong to one customer) and Important Appointment and Customer (an important appointment may only belong to one customer). Not depicted is the Controller class used to navigate around the activities or the database connector class used to connect and query the database.

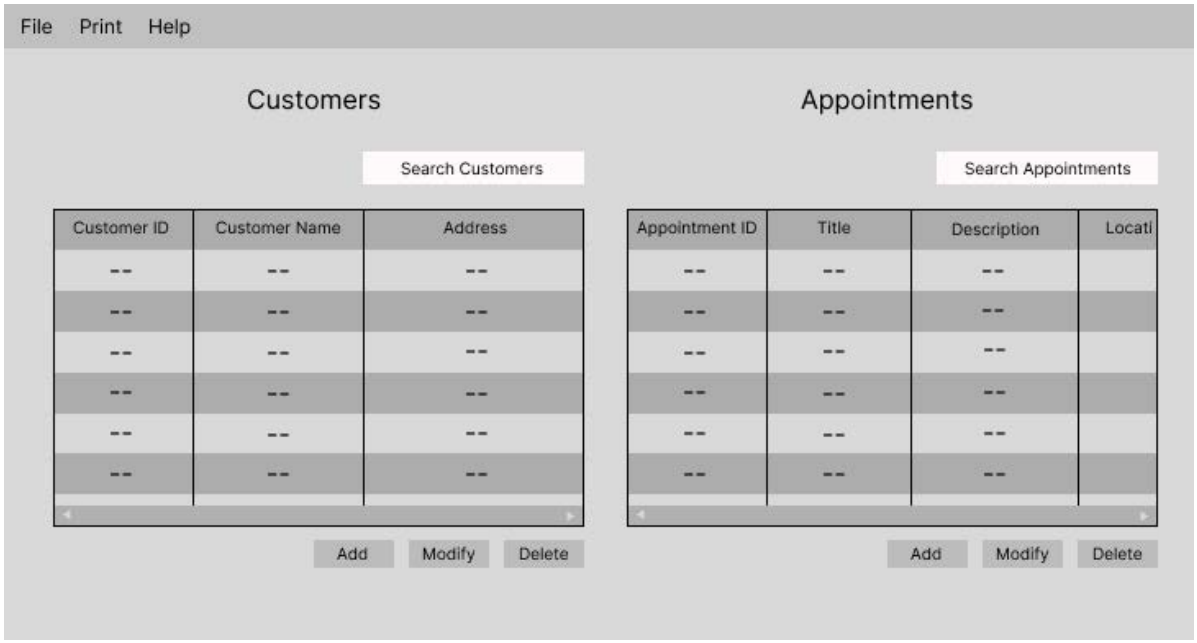
Private class variables are denoted with a minus sign (-). For instance, the Customer's private class variable `id` is only visible within the Customer class, and can not be referred to from other classes. Public class methods are denoted with a plus sign (+). For instance, the Appointment's public class method `printAppointment()` cannot be called without first creating an appointment object and calling it's method.



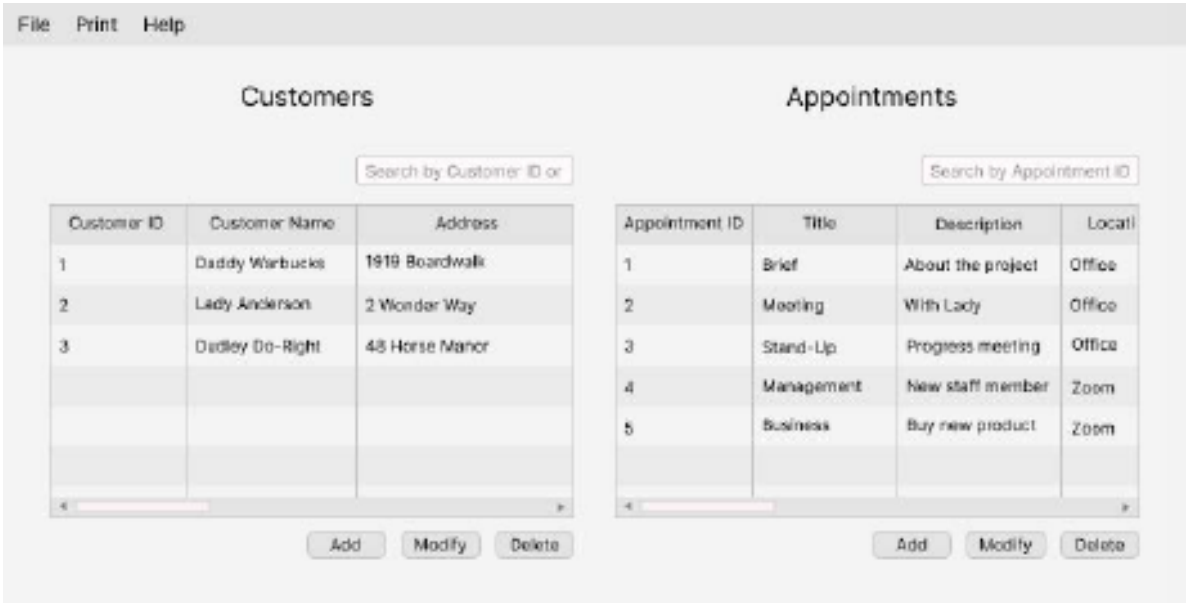
UI Design

Below are the low and high-fidelity versions of the dashboard screen. On the left side is a table view displaying all customer records in the database. Users are able to add, modify, and delete customer records via the buttons below the table view. There is a search bar above the table view to allow users to search for

particular customers. On the right side is a table view displaying all appointment records in the database. Users are able to add, modify, and delete appointment records via the buttons below the table view. There is a search bar above the table view to allow users to search for particular appointments. There is also a toolbar at the top to allow users to view reports and close the application.



Low fidelity version of the dashboard screen



High fidelity version of the dashboard screen

Test Plan

Introduction

Purpose

In order to test the methods of the Appointment class function correctly, I wrote several unit tests using JUnit, a Java testing framework. Since the appointment class is essential to the application, I wanted to confirm that an invalid appointment could not be saved to the database. There are several ways an appointment can be invalid, but the main focus of the tests is to ensure that a customer cannot be double booked. Validations in Java are built to run automatically whenever an appointment is created or updated. The three unit tests verify that the appropriate validation methods are in place to prevent an appointment record from being added to the database if it overlaps with another one under the same customer.

Overview

Since the application's core is a customer appointment tracker, the logic involved in creating and saving appointments needs to be carefully considered. Validation methods ensure that an appointment is not overlapping another by checking other appointments with the same customer. They are called just before an appointment is saved, and if the appointment overlaps any ones in the database, the appointment will not be added to the database. This test confirms that the validation rules responsible for preventing a customer from being double-booked do indeed stop double-booked customer appointments from getting into the database. Other validation tests are in place for business hours but will not be covered here.

The unit test is written using the JUnit framework and executed in the IntelliJ IDE by right-clicking the "UnitTest" directory and selecting "Run 'All Tests.'" This will run the three test cases covering adding overlapping appointments, adding non-overlapping appointments, and adding edge case appointments.

Test Plan

Items

To ensure that an overlapping appointment cannot be added, we need three model instances: one customer and two appointments. Both appointments will be connected to the same customer. The first one will be given a specific start and end time and be saved to the database. A second appointment will also be created with a given a start or end time that overlaps with the first appointment. We can then demonstrate that the second appointment is overlapping and will not be added to the database.

A second test is required to prove that two appointments can be created for the same customer when they do not overlap. For this purpose, we set the start time of the second appointment to have no overlap with the first one and make sure that it does get saved successfully.

We can create adjacent appointments that do not overlap to test the edge cases. It is anticipated that the appointments are valid and get saved to the database.

Functions/Features

To establish the validation for overlapping appointments in the Controller class, I called the `handleAddAppointmentButtonAction` class method that contains the logic with the name of the controller object that contains the values to add: `controller.handleAddAppointmentButtonAction()`.

Deliverables/Outcomes

When the test is run, it produces structured console output. Each test is given a name that indicates its exact behavior, e.g., "testing overlapping appointments." The full line in the test output will be "Adding first appointment, printing appointment type: IMPORTANT APPOINTMENT Adding second appointment, should overlap and not be added to the database: Appointment is overlapping another, did not get added to database."

Tasks

To execute the unit test the following steps are required:

1. Right-click the project folder in IntelliJ and click New -> Directory.
2. Right-click the newly created directory and click New -> Package.
3. Right-click the newly created package and click New -> Java Class.
4. Write the code to test, i.e. set up the validation using JUnit framework.
5. Run the test by right-clicking on the new directory and selecting `Run 'All Tests'`
6. Examine the output to determine pass or fail.

Needs

The described configuration has a few dependencies. The Java packaged libraries used here are:

- `junit-4.10-extended:1.0.4` (JUnit test framework)
- `mysql-connector-java-8.0.25` (MySQL database connector for Java)

Pass/Fail Criteria

The criteria for successful validation of an overlapping appointment is that the test's `testOverlap()` method assertion returns true. A failing test would return an AssertionError and would not be able to confirm the validation criteria.

Specifications

The screenshot below is of the test code described above. It is a section from the UnitTest class and uses the JUnit testing framework. The test file is located at `UnitTest/test/UnitTest.java` in the root directory of the application.

```
public UnitTest() {
    // Initialize and declare panel to set texts.
    JFXPanel fxPanel = new JFXPanel();

    // Initialize controllers.
    controller = new Controller();
    controller2 = new Controller();
    controller3 = new Controller();
}

@Test
public void testEdgeCase() throws Exception {...}

@Test
public void testOverlap() throws Exception {...}

@Test
public void testNoOverlap() throws Exception {...}
```

Procedures

As seen in the code section above, three tests verify that the behavior of the overlapping appointments validation is correct. Since the three tests are in the same class, they are capable of sharing variables. The tests share a typical connection instance to the database in this case.

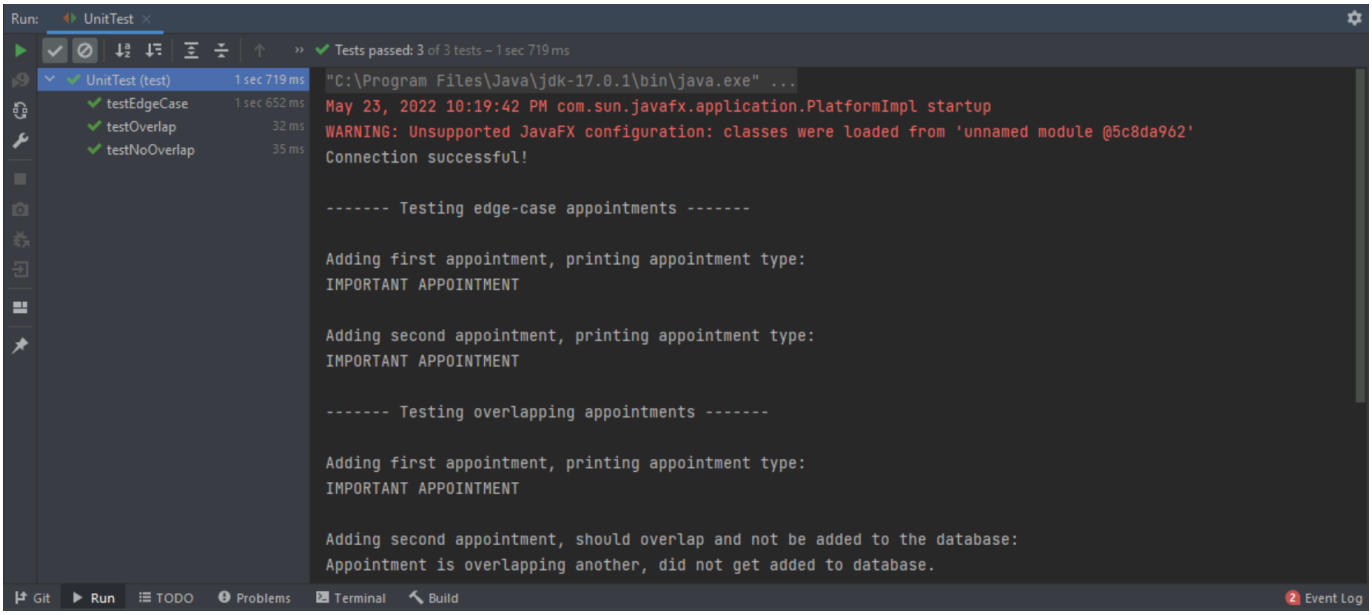
The first test, "`testEdgeCase`," creates a new appointment at 12:00 pm with a duration of 120 minutes. The test then creates another appointment on the same day as the first, at 2:00 pm, 120 minutes after the first one. Using JUnit's assert method, the test sets up the expectation that the appointment would be valid and output: "Adding first appointment, printing appointment type: IMPORTANT APPOINTMENT Adding second appointment, printing appointment type: IMPORTANT APPOINTMENT."

The second test, "`testOverlap`," strives to confirm that another appointment can, in fact, be created for the customer on the same day as long as the appointment time does not overlap. The test creates an appointment at 12:00 pm for a duration of 120 minutes, and after that, it creates another appointment at 1:00 pm, overlapping the first appointment. The functions assert method is expected to return true, meaning there is overlapping.

The third test, "`testNoOverlap`," creates an appointment that starts at 12:00 pm with a duration of 120 minutes. Another appointment is then created that starts at 10:00 pm for a duration of 90 minutes. The assert method is expected to return true, meaning there is no overlapping.

Results

The console output below is printed when running all three tests in isolation. The green checkmark indicates that the tests were executed successfully, and the line above the output that reads "`Tests passed: 3 of 3 tests`" means that 3 out of 3 tests ran successfully. As such, the assumptions expressed in the test can be established.



Source Code

A compressed version of the source code can be found in the file WeSchedule.zip

Link to GitHub Code

https://github.com/jbla484/Capstone

User Guide

Set Up and Run Application for Maintenance Purposes

Several steps are needed to configure and run the Java native application in a local environment for the reason of changing the current code base or fixing bugs. This is a summary of the steps required to configure the application on Windows-based systems. For Unix-based systems, please follow the instructions here: https://gorails.com/setup/windows/10.

Prerequisites

The following system programs need to be installed:

- Java SE version 17.0.1
- JavaFX version 11.0.2
- MySQL Connector for Java version 8.0.25
- The MySQL database management system

Installation

Visit the link above to download the project and:

1. `save the jar to your desktop` (this will make the application easy to find)
2. `the database is already created` (ease of use for users)

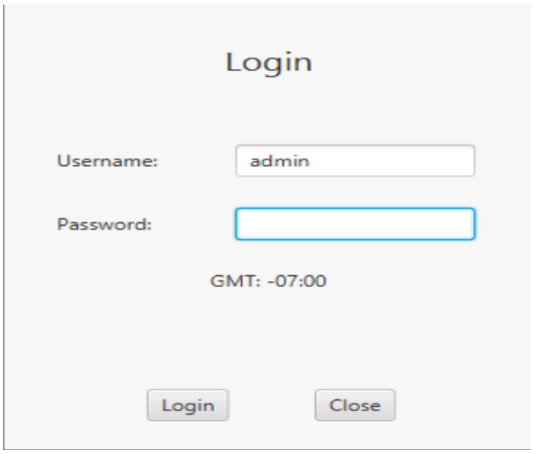
Guide For Admin User

The rest of this User Guide will supply a walk-through of the WeSchedule customer appointment tracking application to display its functionality in detail.

Login with an Admin Account

Since the data tracked by WeSchedule is susceptible, new contact creation has been disabled by default. A senior developer, usually part of the development team, will distribute the admin credentials to select admin users who can then create contacts.

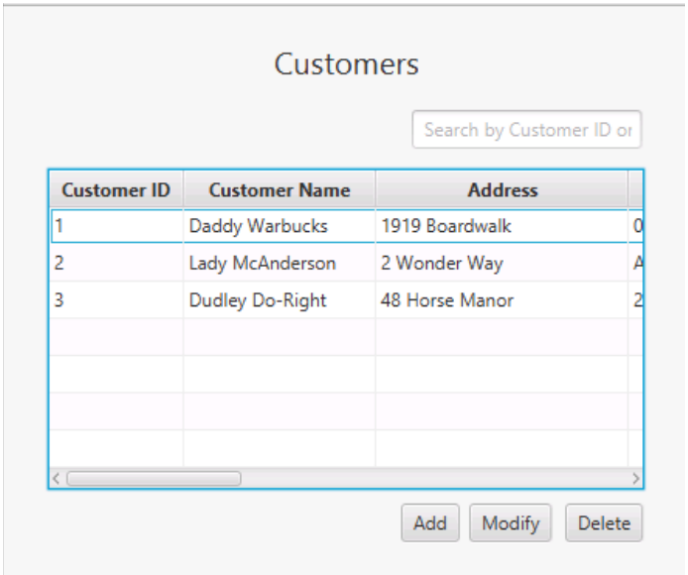
To log in with an admin account, you must enter "admin" as the username and "admin" as the password on the login screen. Once entered, click the login button. You will then be redirected to the dashboard screen if the login credentials are correct. All features in the application should be available to logged-in users. (The application may take a moment to load.)



Viewing and Managing Customers

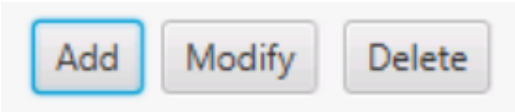
Viewing Customers

Log in to an account and navigate to the dashboard to view all customers. The customer data is in the customer table on the left of the screen and is sorted by customer id. Users are capable of sorting the customer data other ways by clicking the column names on the top of the table.



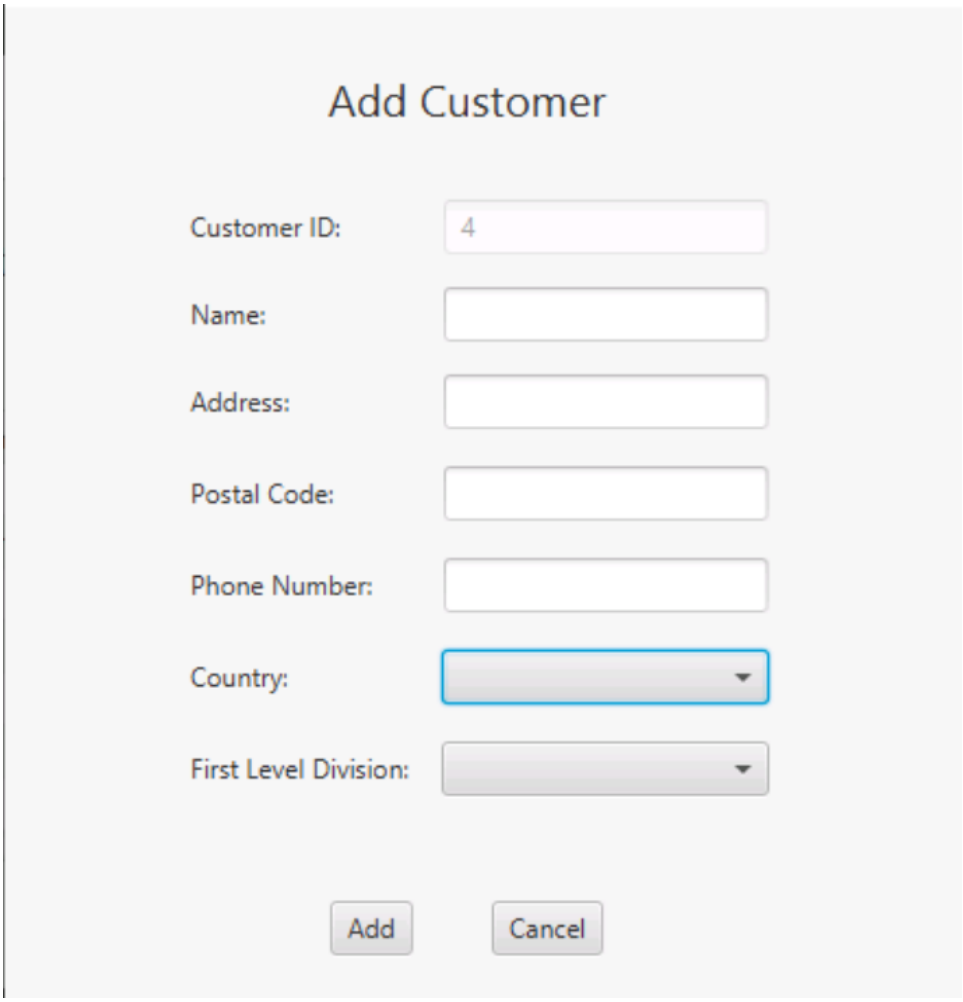
A search filter above the table searches the customers by id or name. For example, if the search filter is filled with "Ander," all customers with that given string of characters in their name would be shown in the table, e.g., "Lady McAnderson." Note that removing any text in the search filter will reset the filter on the table. The buttons below the table perform actions on the selected customer, like adding a new customer, modifying a current one, or deleting a customer.

Creating a New Customer



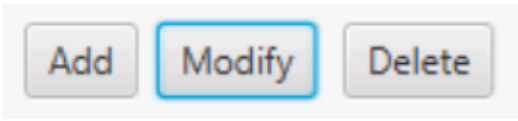
To create a new customer, click on the “Add” button under the customer table on the dashboard screen.

The form to create a new customer is relatively self-explanatory. All fields in the form are required to have input. The country and division can be chosen from a pre-defined drop-down list. The address has validations for the correct format in the chosen country.



Updating Customer Information

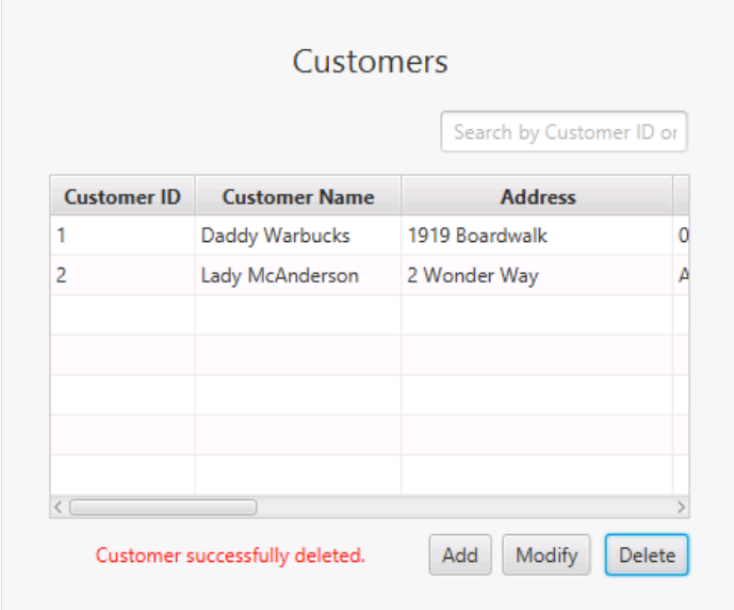
The form to update a customer can be reached by navigating to the dashboard screen, clicking on the customer needing to be updated, and clicking on the modify button under the customer table.

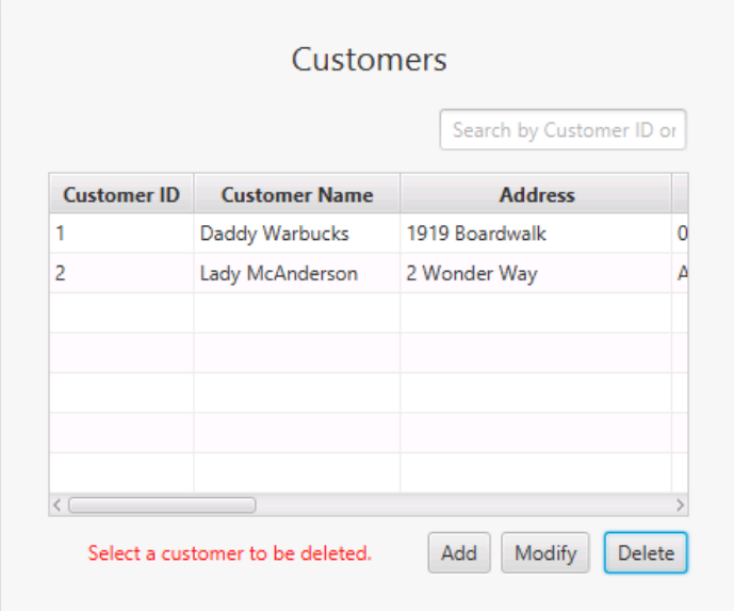


The modify customer form is very similar to the create customer form, except that it has the existing customer properties filled in. The form also displays the customer's upcoming appointments in a table with radio buttons allowing users to sort the appointments by week and month. The validation rules apply in the same way.

Removing a Customer

Deleting a customer record is only possible if the customer has no associated appointments. To delete a customer, first select a customer from the customer table and click the delete button. If a user attempts to delete a customer with associated appointments, the error message "Select a customer to be deleted." is displayed. If a customer is successfully deleted, the message "Customer successfully deleted." will be displayed.

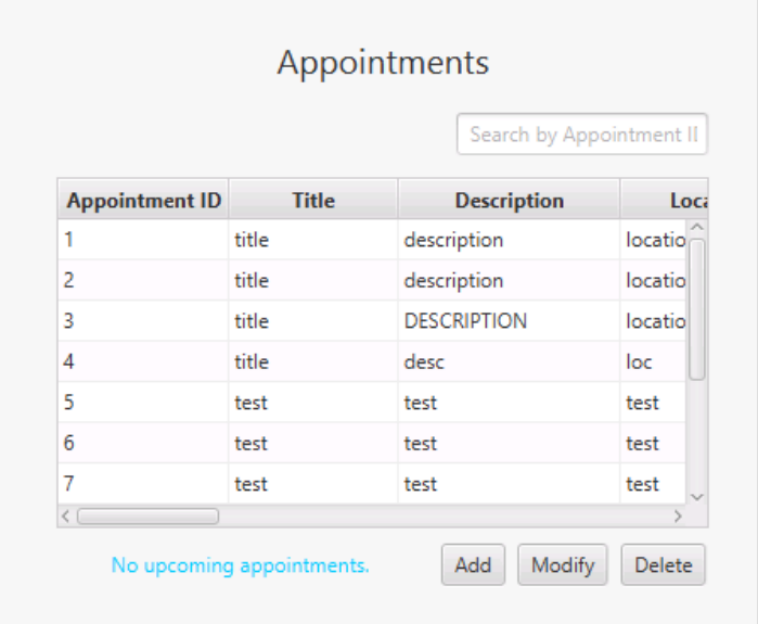




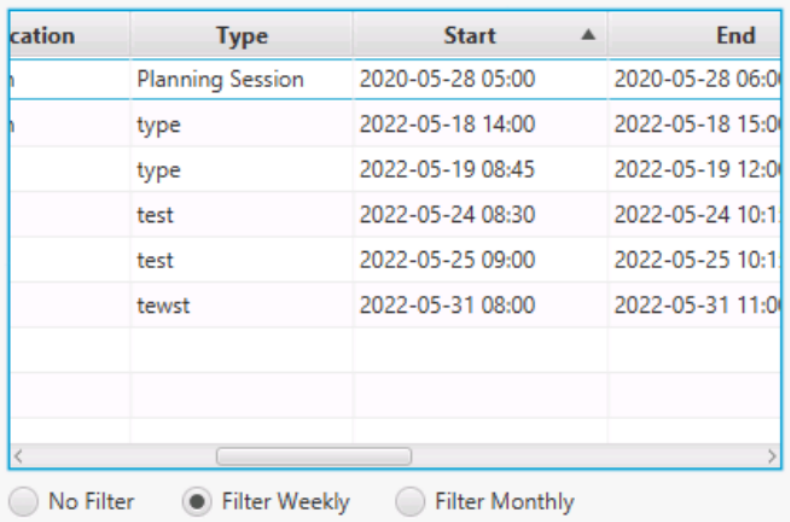
Viewing and Managing Appointments

Viewing Appointments by Month and Week

To view the appointment table, log in to an account and navigate to the dashboard screen. This will reveal a table view that displays all appointments by appointment id. A search filter above the table searches the appointments by id or name. For example, if the search filter is filled with "loc," all appointments with that given string of characters in their name would be shown in the table, e.g., "appointment 4." Note that removing any text in the search filter will reset the filter on the table. The buttons below the table perform actions on the selected appointment, like adding a new appointment, modifying a current one, or deleting an appointment.



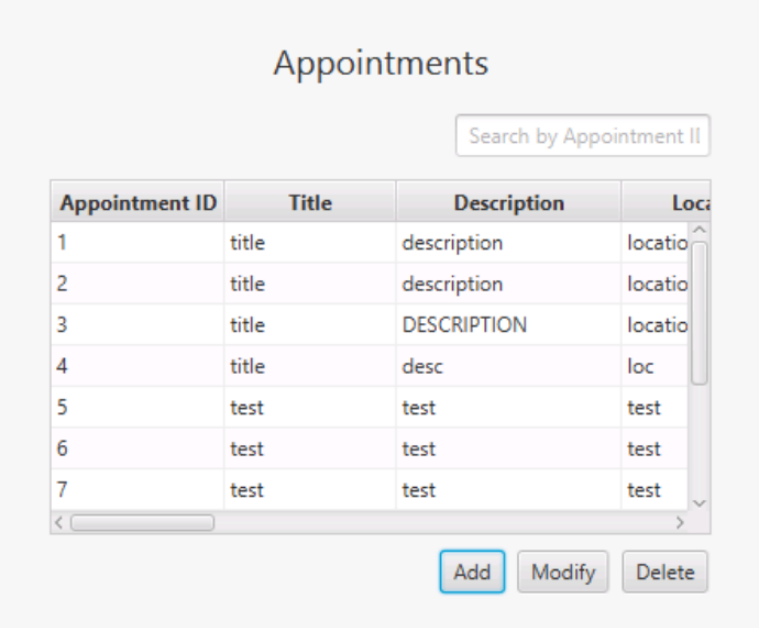
To change the filters on the appointment data, click the table's columns at the top to sort the data. The modify customer screen has more detailed associated appointment table that allows users to sort customer appointments by week and by month.

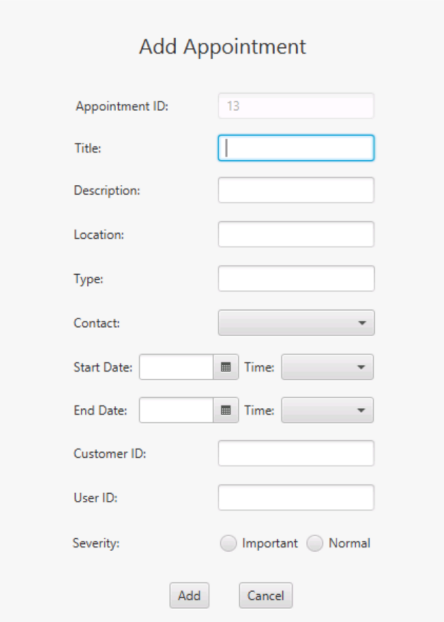


To view the appointment details, search for the appointment name above the table and scroll through the table to see the displayed information.

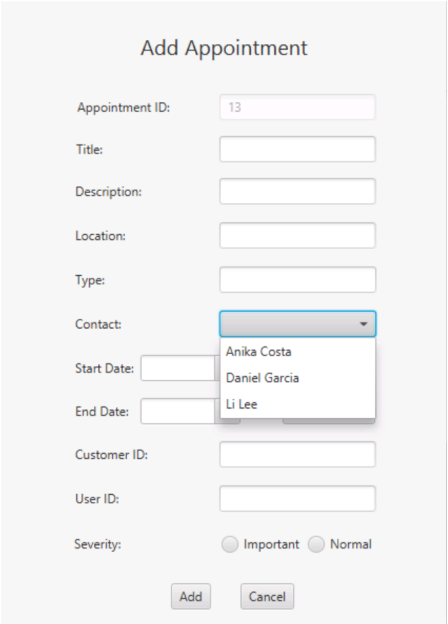
Creating New Appointments

To create a new appointment, login to an account and navigate to the dashboard screen. Right under the appointment table there is an "Add" button, which when clicked, redirects the user to the add appointment screen.

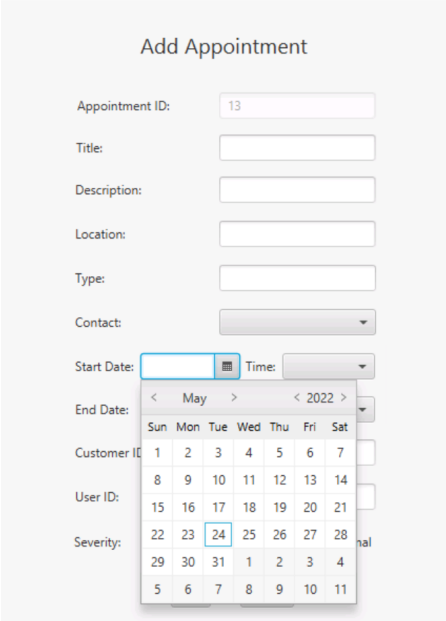




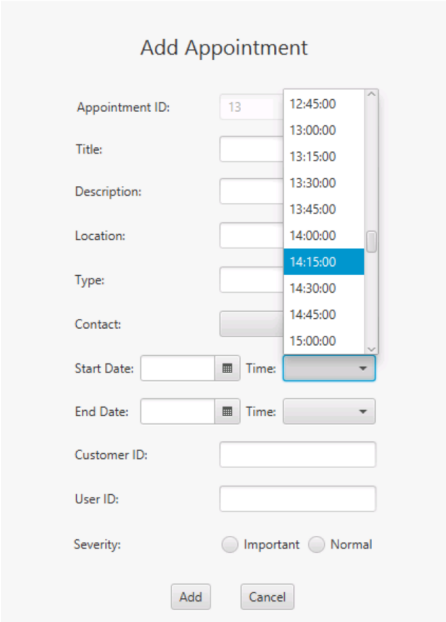
Clicking on the"Add" button will trigger a new screen with a form to add an appointment.



To set the contact for the appointment, click on the contact drop-down box and select the preferred contact.



To set a date for the appointment, click the "Start Date" field. A date picker will emerge to allow setting the date. Click on a date to select it. The left and right arrows at the top allow for navigation between months and years.

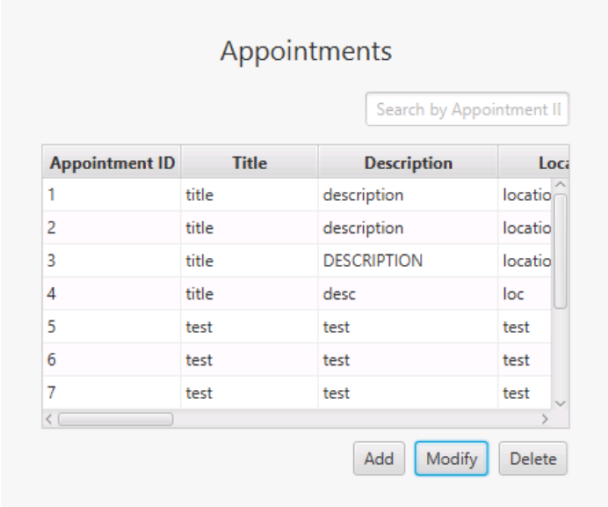


Once a date has been selected, click the time field to display a drop-down list of times. By default, only times during Northwest Schedule's business hours (8 a.m. to 10 pm. EST) are valid. Set the hour of the appointment by scrolling up and down with a mouse or the built-in scrollbar.

Finally, input the appointment's customer id, user id, and severity, then click on "Add." The appointment will be created, the screen will close, and the appointment will appear in the appointment table on the dashboard.

Updating Appointments

The modify form for an appointment can be arrived at from the dashboard by clicking on an appointment in the appointment table, then clicking on the "Modify" button. For instance:



Dashboard screen with modify and delete buttons.

The modify appointment form works the same as the add form. There are three drop-down lists – one for the contact and two for the time. The form also contains two date pickers for choosing start and end dates. The dates can be changed via the date picker and the time via the time drop-down list.

Deleting an Appointment

The dashboard screen allows users to delete appointments. First click on the appointment to be deleted, then click on the "Delete" button under the appointment table. Doing so will delete the appointment and show a confirmation message on the screen.

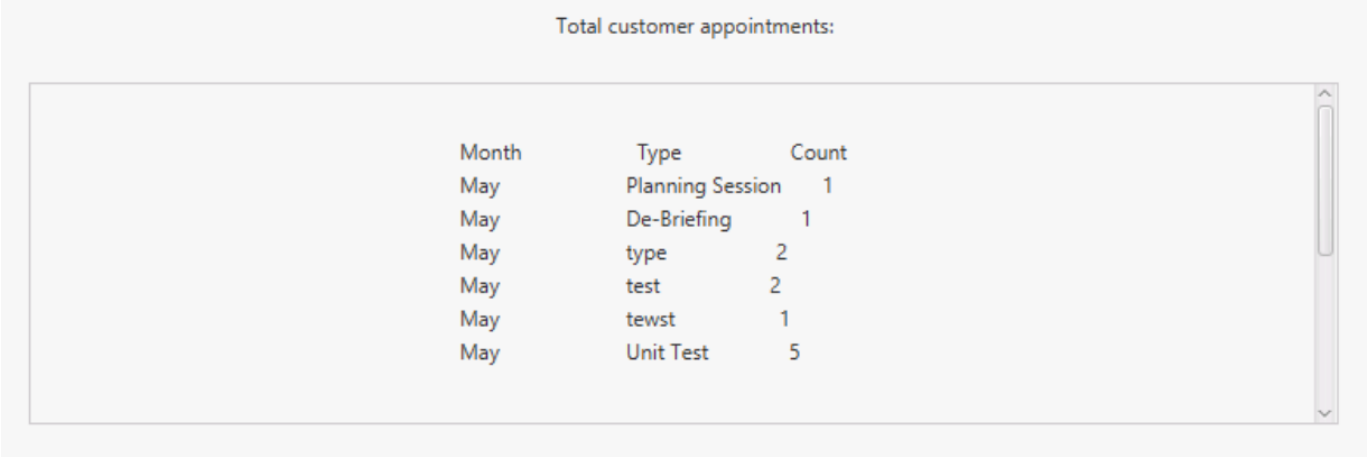
Viewing Appointment Reports

Three types of reports are accessible to users of the application:

1. A tabular report to display the total number of customer appointments by type and month
2. A tabular report to display a schedule for each contact in your organization.
3. A tabular report to display the total number of scheduled hours per contact.

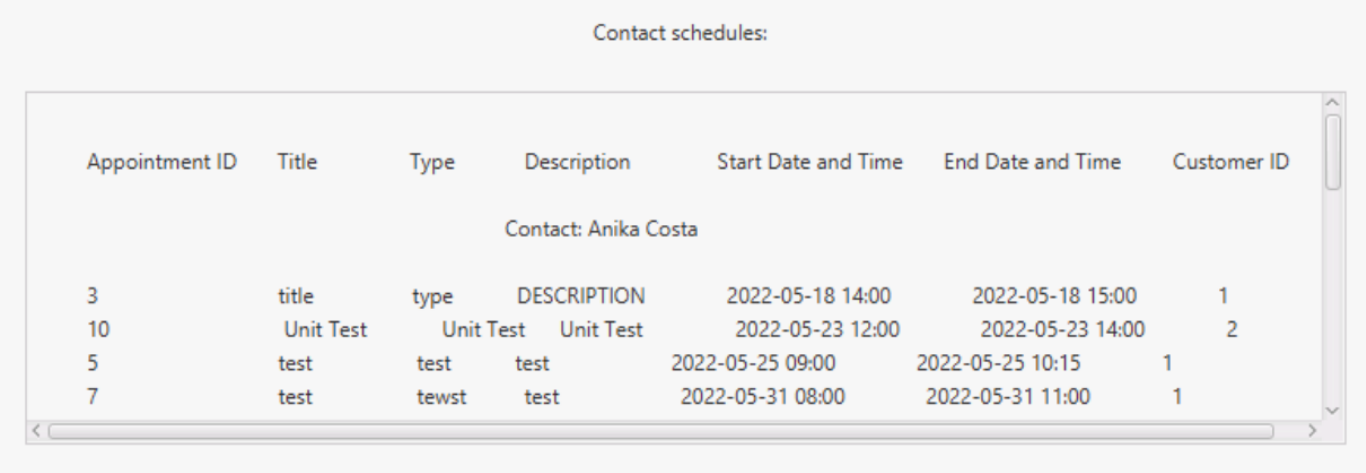
Appointments by Type and Month

This report can be viewed by clicking on the “Print” menu option from the menu bar at the top of the dashboard screen and scrolling down. The page displays a tabular view of all appointments grouped by month and type, and a count of the grouped appointments.



Appointment Schedule For Each Contact

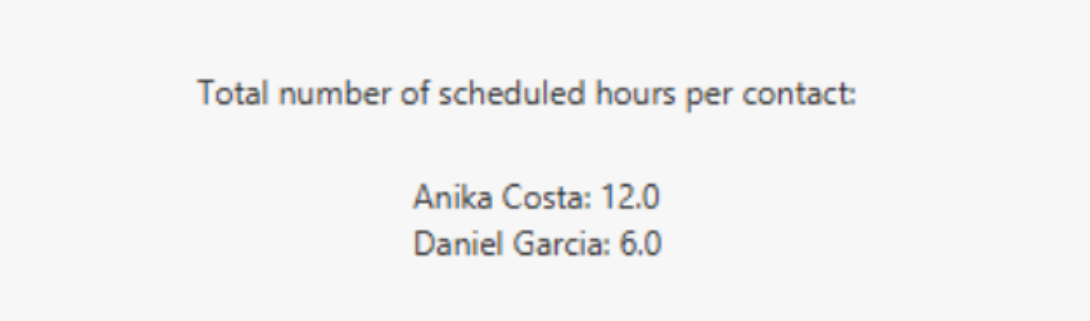
The report that displays a schedule for each contact can be accessed by clicking on the “Print” menu option from the menu bar at the top of the dashboard screen and scrolling down. The report shows a schedule in tabular text format. This information can be used to make decisions with scheduling and help contacts better manage their time.


Contact schedules:

Appointment ID	Title	Type	Description	Start Date and Time	End Date and Time	Customer ID
Contact: Anika Costa						
3	title	type	DESCRIPTION	2022-05-18 14:00	2022-05-18 15:00	1
10	Unit Test	Unit Test	Unit Test	2022-05-23 12:00	2022-05-23 14:00	2
5	test	test	test	2022-05-25 09:00	2022-05-25 10:15	1
7	test	tewst	test	2022-05-31 08:00	2022-05-31 11:00	1

Total Number of Scheduled Appointment Hours

The report with the total number of scheduled hour per contact can be accessed by clicking on the “Print” menu option from the menu bar at the top of the dashboard screen.


Total number of scheduled hours per contact:

Anika Costa: 12.0
Daniel Garcia: 6.0