Software II – Advanced Java Concepts – C195

Performance Assessment Documentation

Western Governors University

Timothy Smith

Student ID: #000865546

11/11/2019

Table of Contents

[PROJECT STRUCTURE 3](#_Toc24398008)

[Folder Structure 3](#_Toc24398009)

[Folder Class Lists 4](#_Toc24398010)

[GETTING STARTED 6](#_Toc24398011)

[Logging In 6](#_Toc24398012)

[Main Screen 6](#_Toc24398013)

[SCHEDULE MANAGER OBJECTIVES 7](#_Toc24398014)

[Objective A: Dual language log-in 7](#_Toc24398015)

[Objective B: Ability to add, update, and delete customer records 7](#_Toc24398016)

[Objective C: Ability to add, update, and delete appointments 8](#_Toc24398017)

[Objective D: Ability to view the calendar by month and by week 9](#_Toc24398018)

[Objective E: Ability to adjust appointment times based on user time zone 9](#_Toc24398019)

[Objective F: Write exceptions to enforce business logic 9](#_Toc24398020)

[Objective G: Write two or more lambda expressions 10](#_Toc24398021)

[Objective H: Alert user to appointment within 15 minutes of log-in 10](#_Toc24398022)

[Objective I: Provide 2 required reports and 1 report of choice 11](#_Toc24398023)

[Objective J: Ability to track user log-ins 11](#_Toc24398024)

[Objective K: Professional Communication 11](#_Toc24398025)

# PROJECT STRUCTURE

## Folder Structure

A close up of a logo

Description automatically generated

(Figure 1. Schedule Manager Folder Structure)

*Items not relevant to evaluation greyed out*

## 

## Folder Class Lists

* schedulemanager
  + ScheduleManager.java
* DAO
  + AddressDaoImpl.java
  + AppointmentDaoImpl.java
  + CityDaoImpl.java
  + CountryDaoImpl.java
  + CrudDao.java
  + CustomerDaoImpl.java
  + SysOp.java
  + SysOpException.java
  + UserDaoImpl.java
* model
  + Address.java
  + Appointment.java
  + ApptTypeByMonthRow.java
  + AvyDailyApptsByUserRow.java
  + City.java
  + Country.java
  + Customer.java
  + User.java
* utility
  + DateTimeHandler.java
* view\_controller
  + AddAppointmentController.java
  + AddAppointmentInputManager.java
  + AddCustomerController.java
  + AppointmentTypesByMonthReportController.java
  + ApptFormException.java
  + AvgDailyApptsByUserReportController.java
  + CalendarView.java
  + ChangePasswordController.java
  + ConsultantScheduleController.java
  + DayPlan.java
  + DayView.java
  + DBServiceManager.java
  + DBTask.java
  + EditAppointmentInputManager.java
  + EditCustomerController.java
  + InputManager.java
  + LoginController.java
  + LoginFormException.java
  + LoginService.java
  + MainScreenController.java
  + MainScreenService.java
  + ManageCitiesCountriesController.java
  + ManageCustomersController.java
  + ManageUsersController.java
  + MonthView.java
  + QuickViewAppointmentController.java
  + SelectAddressController.java
  + Session.java
  + Session.Exception.java
  + ViewModifyAppointmentController.java
  + WeekView.java

# GETTING STARTED

## Logging In

Upon loading the program a log-in screen will be displayed. The log-in credentials are as required:

Username: *test*

Password: *test*

After verifying the credentials, the program will load the main screen. All instructions from this point on will begin from that point unless otherwise noted.

## Main Screen

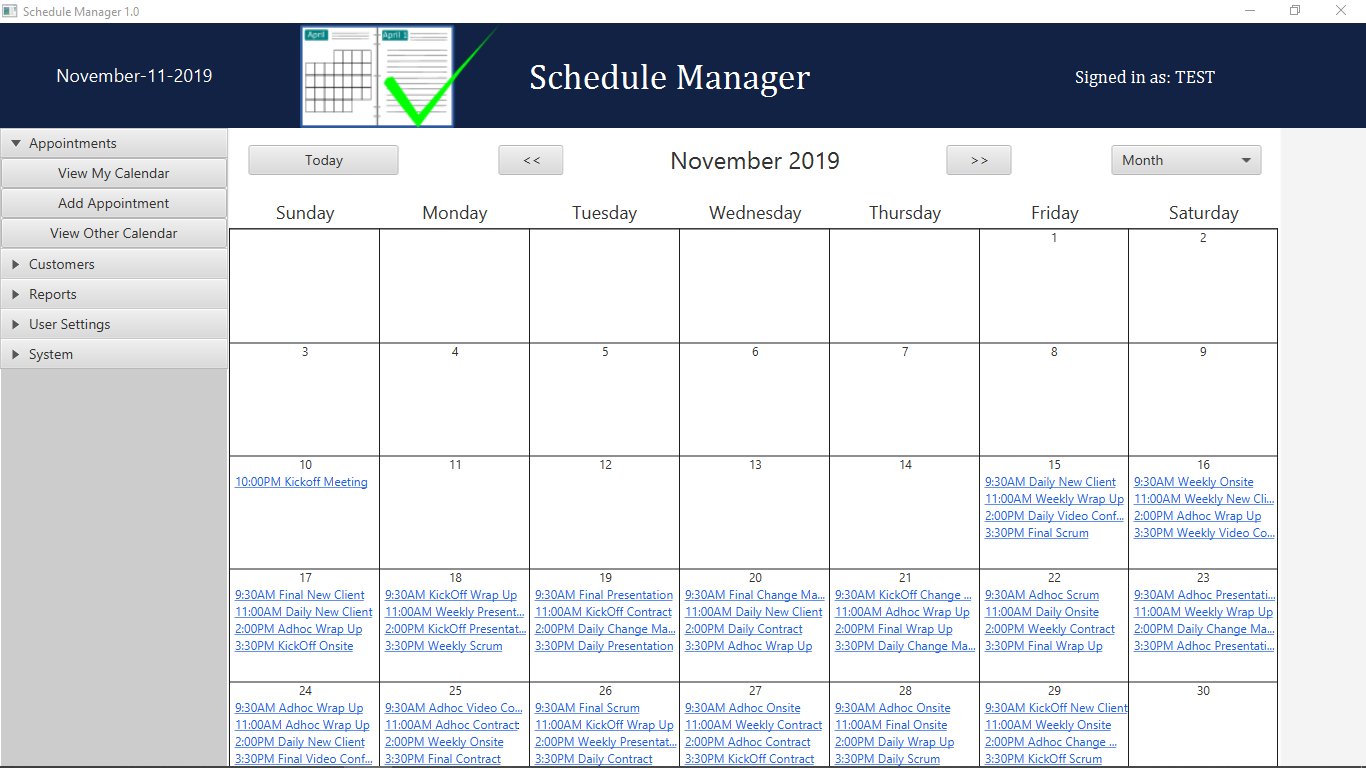


Diagram 1: Main Screen Display

The main view displays appointments in a calendar view. The view can be set to Month, Week, or Day. To access an appointment, click on the link inside the calendar. This will display a quick view of the appointment where you can choose to edit or delete the appointment.

There is a navigation pane on the left and as well as calendar controls above the calendar.

All screens have a progress indicator to alert the user that processes, typically database interactions, are running in the background.

# SCHEDULE MANAGER OBJECTIVES

## Objective A: Dual language log-in

The log-in screen by default displays prompts and messages in English. The requirement is met by assessing if the users default *Local* has French set as the language. If the language is set to French it displays prompts and messages on the log-in screen in French.

The lines involved in the logic are:

* LoginController.java
  + 71-74
  + 116-118
  + 152-158
  + 201-208

The requirement can be tested by changing the Local to have a language of French “fr”. The only changes per the requirement are to the log-in screen and its messages. The labels will now be in French. In addition, if log-in information is missing of incorrect the error messages will display in French. The username *test* and password *test* will still work and proceed to the other screens which will display as normal.

## Objective B: Ability to add, update, and delete customer records

This requirement is met through the Customers section of the left side navigation pane. Interfaces are provided to view, edit, delete and add customers and the details relating to the customers.

These requirements can be tested by selecting Manage Customers under the Customers section of the left navigation pane. This brings up the Manage Customers screen.

ADD - To add a customer, click on the Add Customer button to the upper right of the customer list. An input form will display allowing the user to input the information. There are options for inputting new address information or selecting an address from the system in case of multiple customers from the same location. When adding a new address, the user can select the city from the dropdown. If the city is not populated in the database, the user can click on the add button beside the dropdown list to access the Manage Cities/Countries screen to add the city. If the option is chosen to select the address, the Select Address button is displayed. Clicking this button will bring up a list of addresses that can be selected. The list can be filtered and sorted to find the correct address. Once located, click the checkmark on the row of the address and it will be populated on the new customer. Once all data is entered click Submit.

\* (This form also satisfies Objective F: as exceptions are used to indicate input errors)

Update – To update a customer, click on the pencil icon in the row of the customer to be updated. There are filters on the left side of the screen to help find the desired customer. Clicking the pencil icon will bring up a form to edit the customer. There are options to edit the current address, choose a different address that exists in the system, or add a new address. Note: editing the current address will change it for any customer linked to that address. Once updates are complete, select Submit.

\* (This form also satisfies Objective F: as exceptions are used to indicate input errors and database exceptions)

Delete – To delete a customer, click on the trash icon in the row of the customer to be deleted. After clicking, a dialog will display confirming that you want to delete the selected customer. If confirmed, the user is deleted.

\* (This form also satisfies Objective F: as exceptions are used to indicate input errors and database exceptions. For example, displays database error when trying to delete customer attached to appointments)

## Objective C: Ability to add, update, and delete appointments

This requirement is met through various user input forms most accessed from the main screen.

ADD – To add an appointment select Appointments on the left navigation pane and then choose Add Appointment. An input form is presented for the new appointment information. By default, the start is set to the current date and the next hour of the day (Local time) and the end to ½ hour after that. These can be changed as necessary. Also, the consultant is set to the logged in user but can also be changed as needed. All fields are required. Fields are validated to ensure all our populated and that the date/times conform to business logic. Times cannot:

* Be before the current time
* End time before start time
* Start before 8am or end after 5pm

After input the data, click submit to create the appointment.

\* (This form also satisfies Objective F: as exceptions are used to indicate input errors)

Update – Appointments can be edited from any of the calendar views (Month, Week, Day) by selecting the link in the calendar view. Clicking the link brings up the appointment quick view. This displays some of the key appointment info and has buttons to edit or delete the appointment. Click the pencil icon to edit the appointment. An input form is presented to allow the user to edit the appointment. All fields are required. Fields are validated to ensure all our populated and that the date/times conform to business logic. Times cannot:

* Be before the current time
* End time before start time
* Start before 8am or end after 5pm

After input the data, click submit to create the appointment.

\* (This form also satisfies Objective F: as exceptions are used to indicate input errors)

Delete - Appointments can be deleted from any of the calendar views (Month, Week, Day) by selecting the link in the calendar view. Clicking the link brings up the appointment quick view. This displays some of the key appointment info and has buttons to edit or delete the appointment. Select the trash icon to delete the appointment. After clicking, a dialog will display confirming that you want to delete the selected appointment. If confirmed, the appointment is deleted.

## Objective D: Ability to view the calendar by month and by week

This requirement is demonstrated on the main screen where the user can select Month, Week or Day view. The default is Month. Selecting another option will change the view to the selected timeframe.

The requirement can be verified by selecting an option from the view dropdown on the upper right-hand side of the main screen.

## Objective E: Ability to adjust appointment times based on user time zone

This requirement is met using the java.time API, specifically LocalDateTime/LocalDate/LocalTime.

The requirement can be verified by:

* Noting appointment times
* Logging out
* Change system time to a different time zone
* Logging in
* Noting that times should now be displayed according to new time zone

## Objective F: Write exceptions to enforce business logic

All of the following are required, with at least 2 different mechanisms being used.

* Scheduling an appointment outside of business hours
  + Prevented by form validators that throw custom exceptions of class *ApptFormException* the input form receives the exception, displays a message and highlights the field.
  + Appointment add form validators are in class *AddAppointmentInputManager* lines:
    - 267 – 547
  + Appointment edit form validators are in class *EditAppointmentInputManager* lines:
    - 366 - 610
* Scheduling overlapping appointments
  + On submit of a new or updated appointment the database is queried for overlapping appointments. If conflicts exist, the query handler throws an *ApptFormException*. The input form accepts the exception and displays an alert dialog to the user.
  + The mechanism can be found in class *DBTask* lines 158 - 166
* Entering nonexistent or invalid customer data
  + Customer data is validated on multiple levels
    - At the appointment level customers are selected via dropdown so invalid data cannot be entered. The field is required and validated by the validators in *AddAppointmentInputManager* and *EditAppointmentInputManager* (lines listed above)
    - Additionally, customer data is validated when adding or editing a customer using change listeners on the JavaFX controls. The value is checked and if it is not valid a message is displayed beside the field and the field is highlighted.
      * + This logic is found in:

*AddCustomerController* lines 258 – 344

* + - * + This logic is found in:

*EditCustomerController* lines 334 - 419

* Entering incorrect username and password
  + Login credentials are validated against the database and the calling service throws a custom exception *SessionException*. After the 3rd failed attempt the program closes. The login form displays an error message to the user when the exception is thrown.
    - The logic can be found in:
      * *LoginService* lines 40-57
      * *Session* lines 122 – 163

All four cases required are controlled as required. The mechanisms in place are:

* Validation handlers throwing custom *ApptFormException*
  + Appointment add and update
* Query return values triggering custom *ApptFormException*
  + Appointment add and update
* Using choice boxes to limit customer choices to valid customers
  + Appointment add and update
* Change listeners validating input and displaying messages directly without exceptions
  + Customer add and update
  + Also implemented in user add/update, address add/update and city/country add/update
* Matching user input to query results to verify login credentials and throwing custom *SessionException*
  + Log-in screen

## Objective G: Write two or more lambda expressions

This requirement is implemented thought out the program in several places and situations. Below is a list of some of the lambda expressions

* *AddAppointmentController*
  + 170, 176, 286, 376, 612
* *AppointmentTypeByMonthController*
  + 74, 123
* *DayView*
  + 65
* *MainScreenController*
  + 167, 535
* ManageCitiesCountriesController
  + 198

## Objective H: Alert user to appointment within 15 minutes of log-in

This requirement is met by checking for appointments within 15minutes of log-in time and displaying a popup if there is a conflict. The logic disregards meetings that have/should have already started.

This requirement can be tested by creating an appointment to happen within the next 15 minutes then logging out and logging in. The logic is located in:

* *MainScreenController* at lines 158 – 185

## Objective I: Provide 2 required reports and 1 report of choice

This requirement is met by the following reports accessed in the Reports section of main screen left navigation pane:

* Number of Appointment Types by Month
  + Displays a pie chart of the selected month showing the percentage of each appointment type. If the selected month has no data it displays a message in place of the chart
* Schedule by Consultant
  + Prompts to select a consultant, then displays the consultants schedule
* Average Daily Appointments by Consultant
  + Displays a bar chart ranking the consultants from high to low by their average appointments per day

## Objective J: Ability to track user log-ins

This requirement has been implementing writing to the file *userlog.text* in the root folder of the project. A log entry is made for every successful log-in as well as any failed log-in attempts. Each log entry includes a status followed by user(if known) and a timestamp.

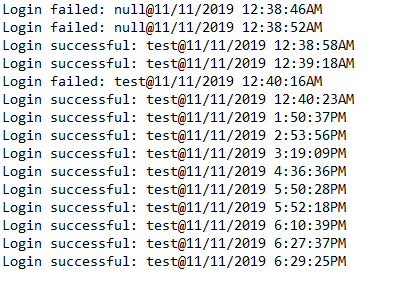


Diagram 2: User Log File

The requirement can be verified by keeping track of login attempts and verifying against file contents.

## Objective K: Professional Communication

This requirement is met through the content of the program and the descriptions contained in this document.