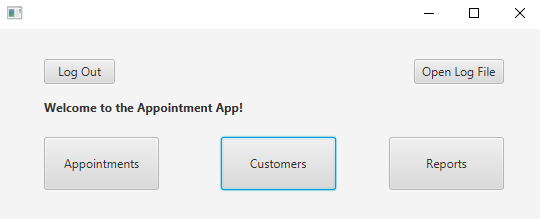
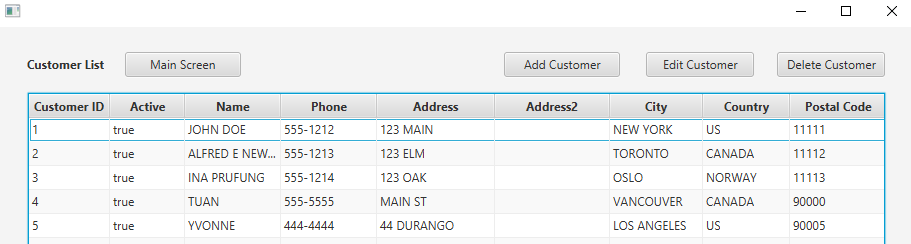
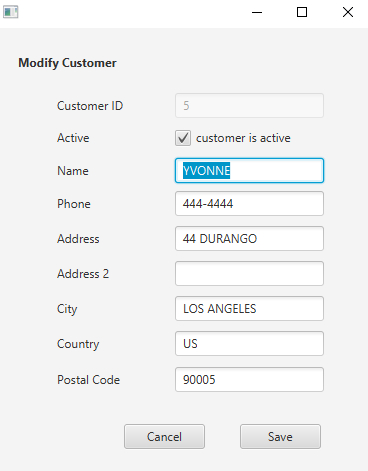
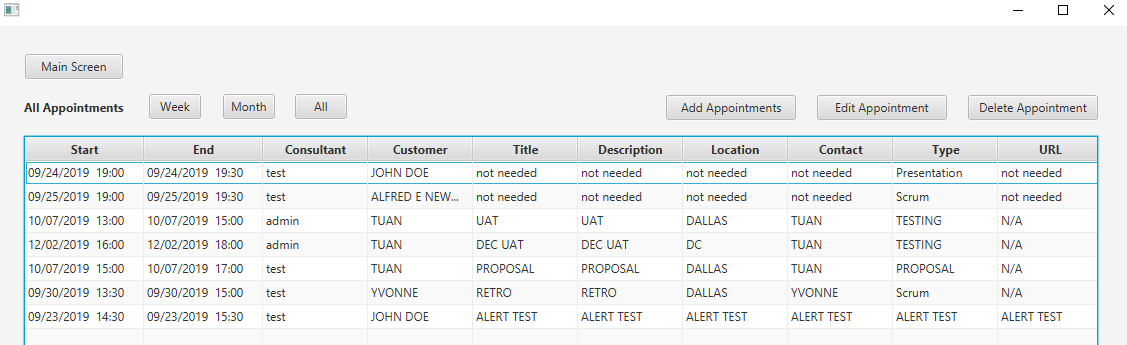
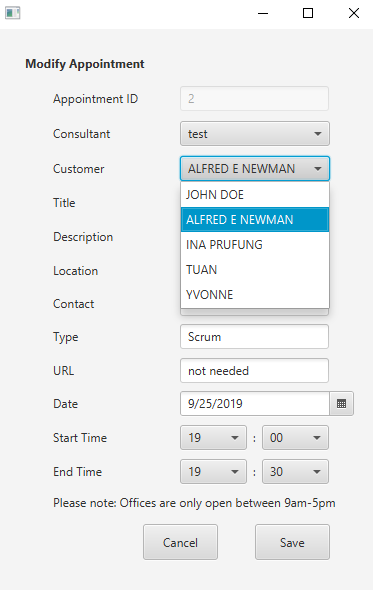
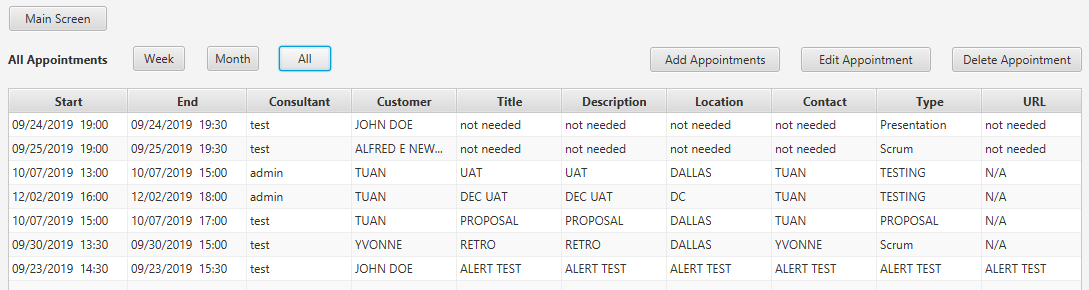
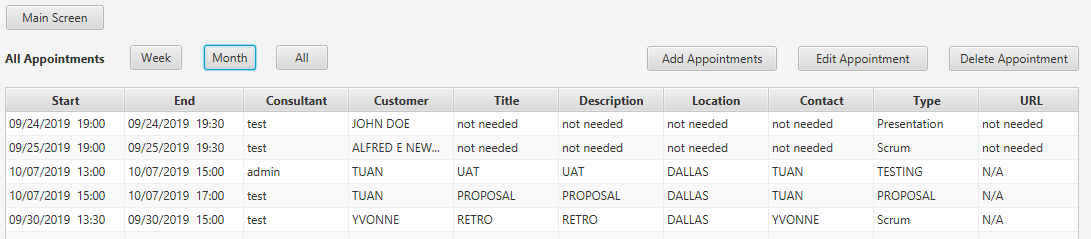
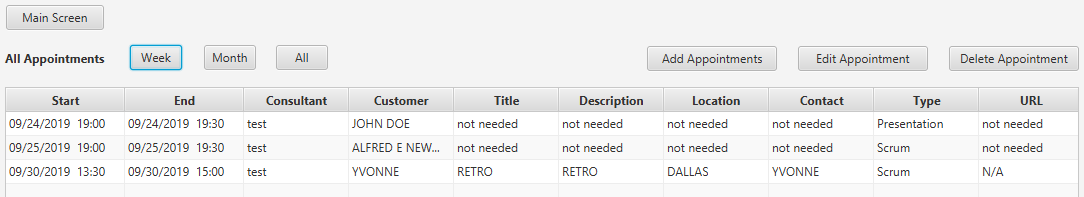
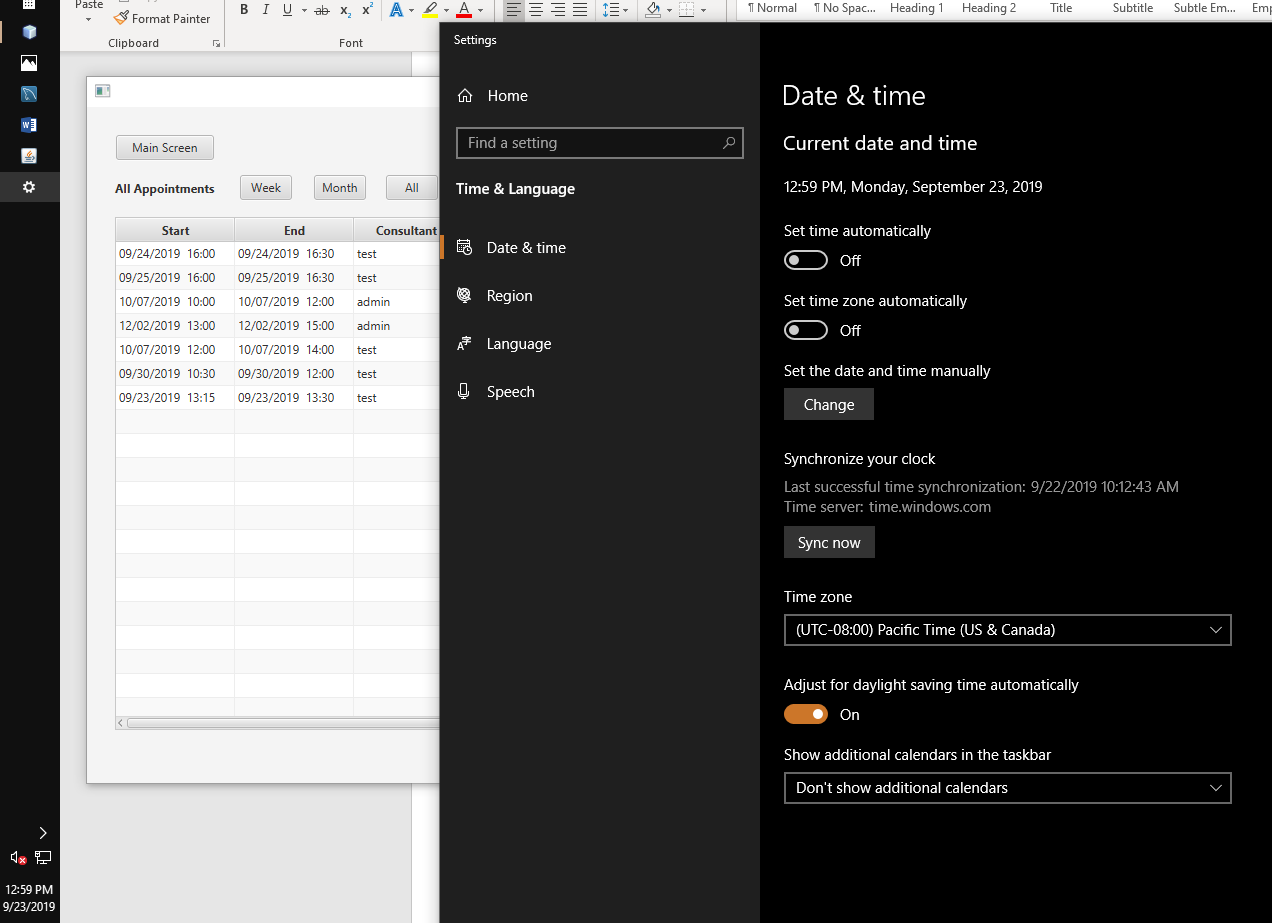
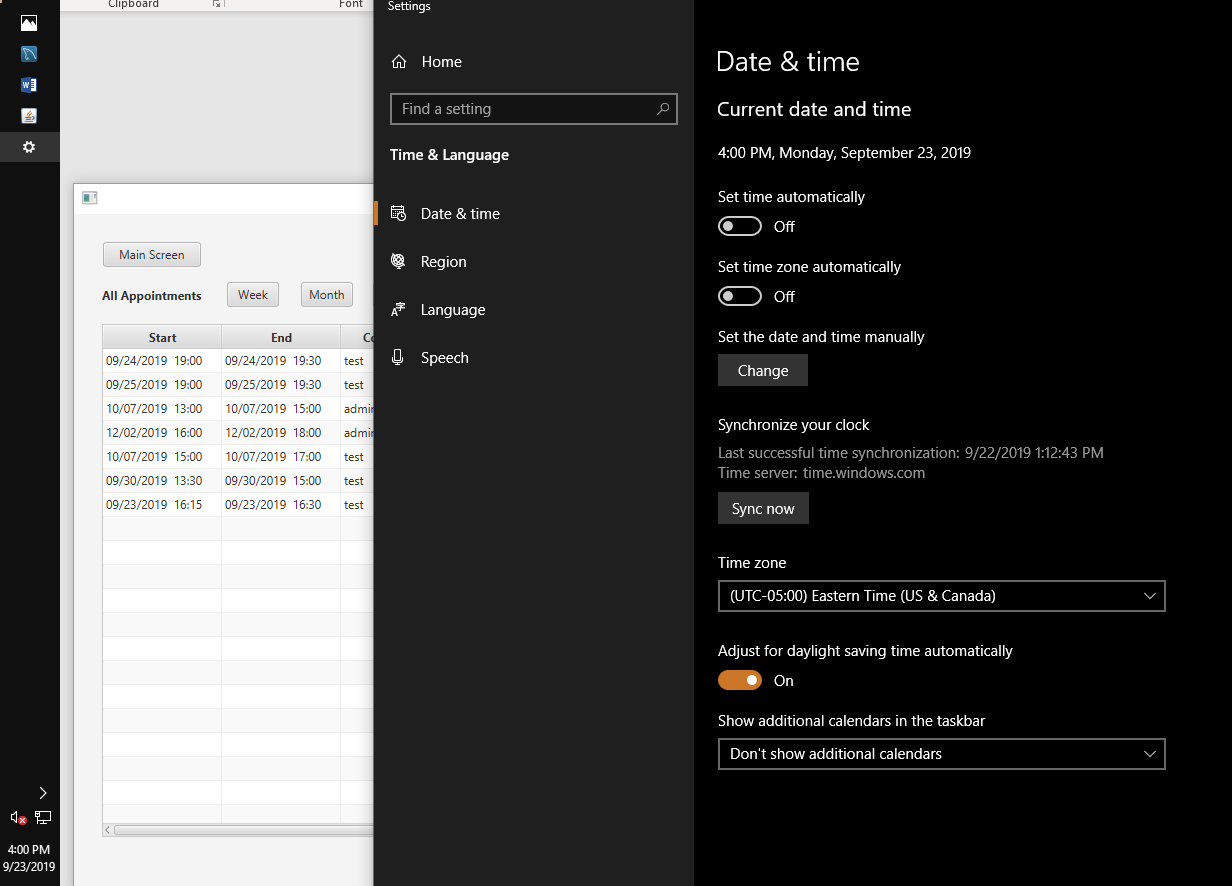
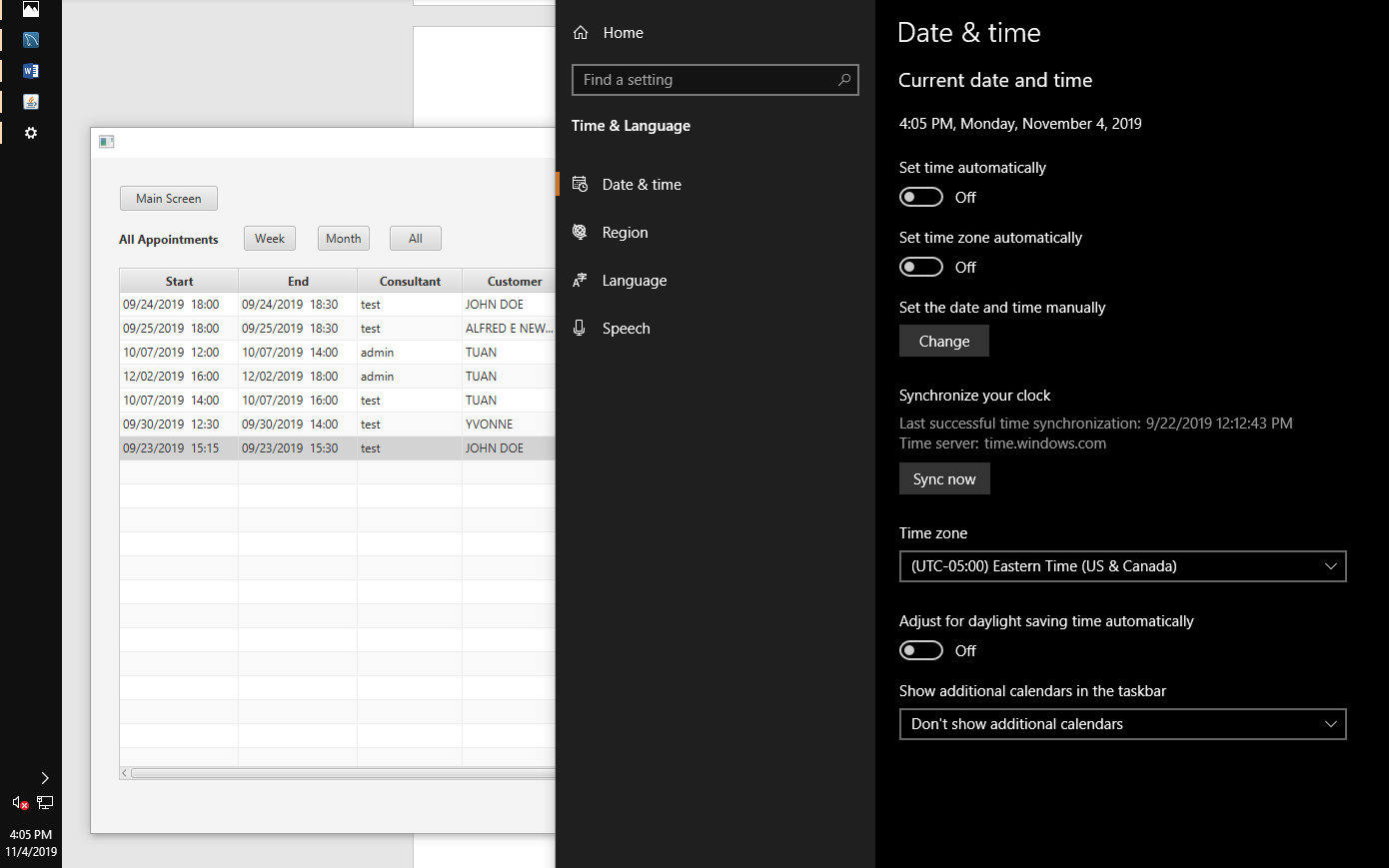
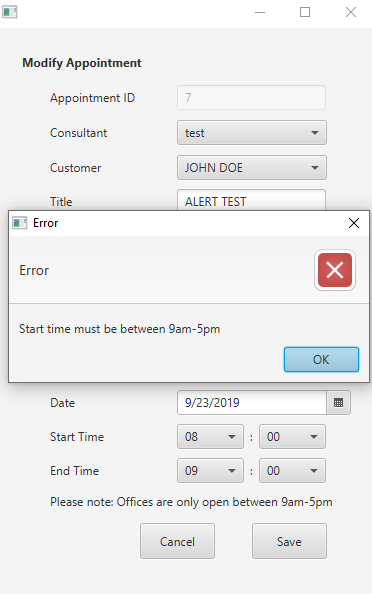
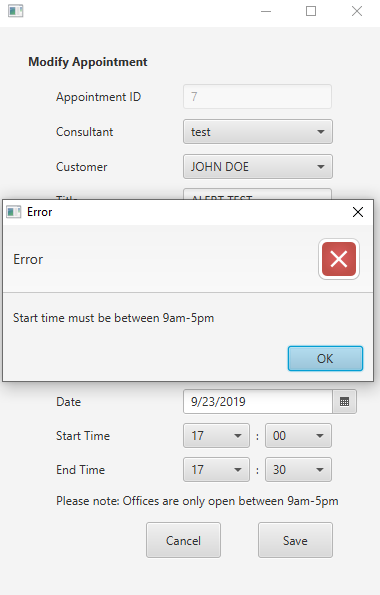
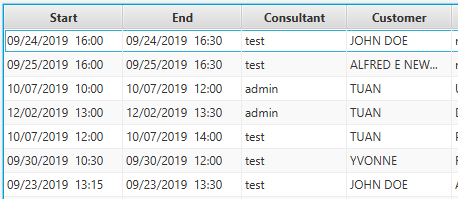
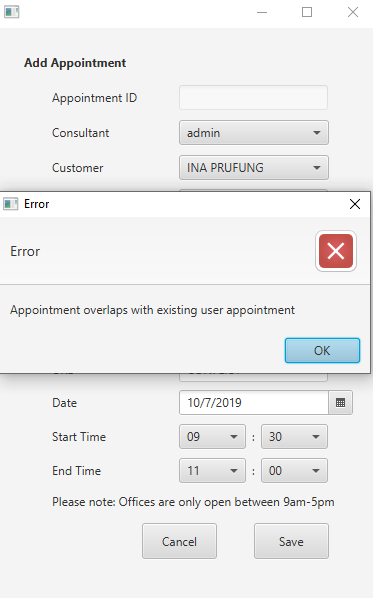
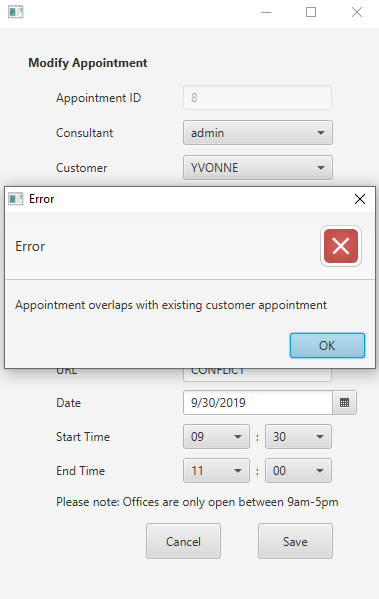
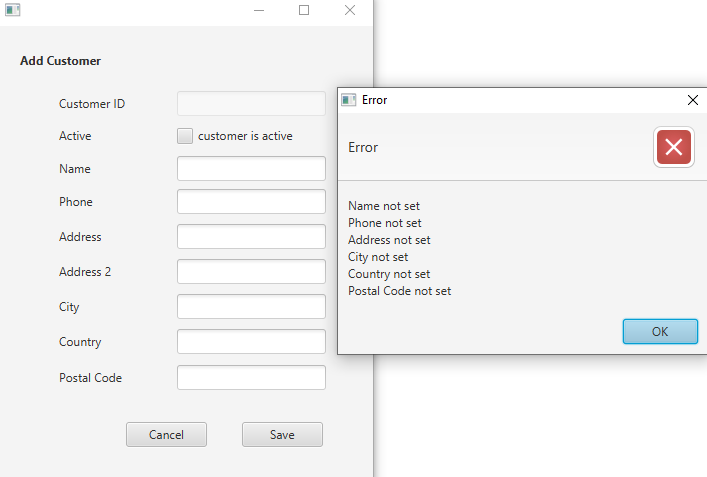
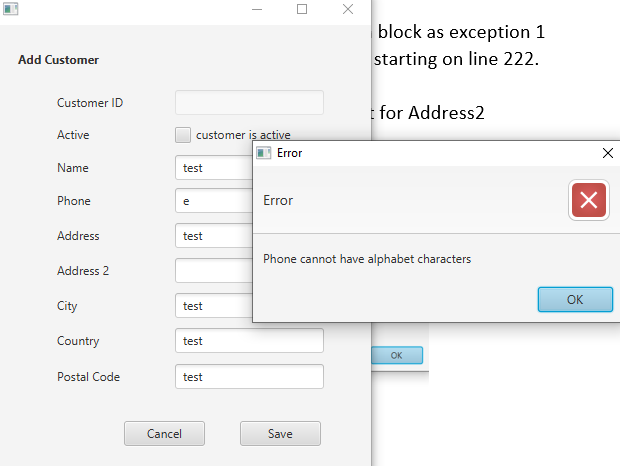
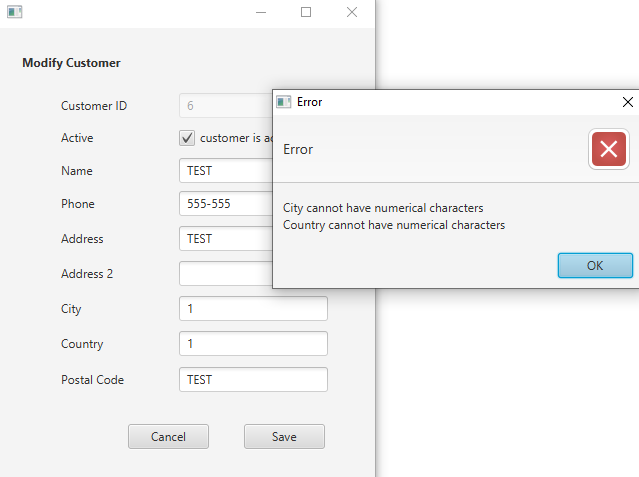
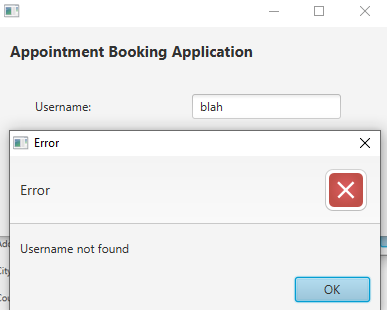
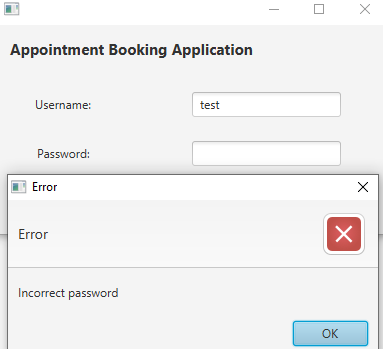
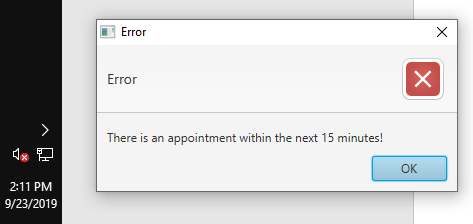
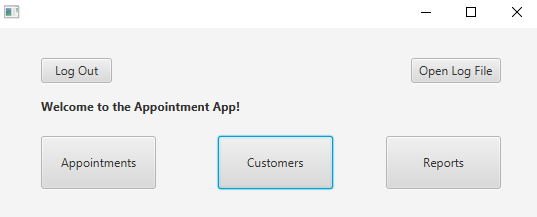
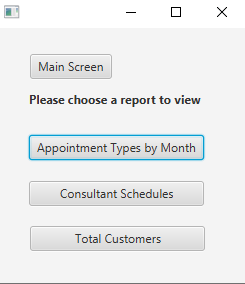
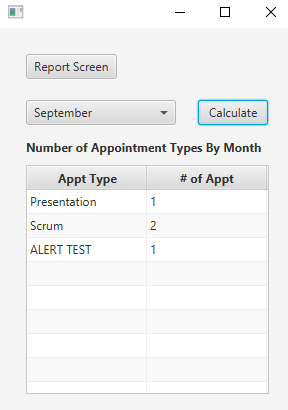
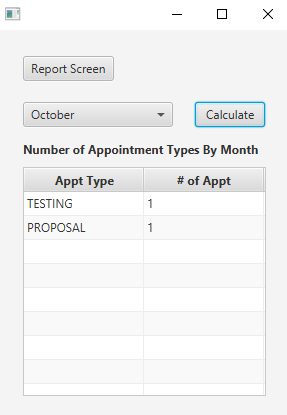
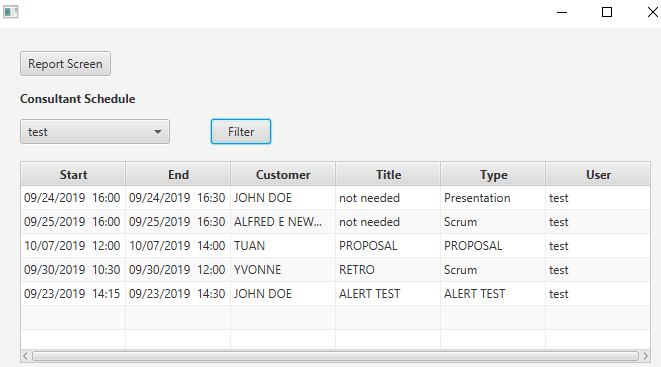
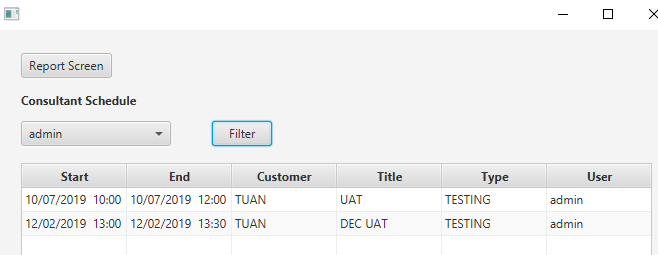
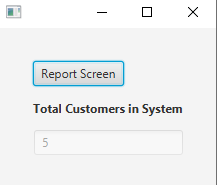
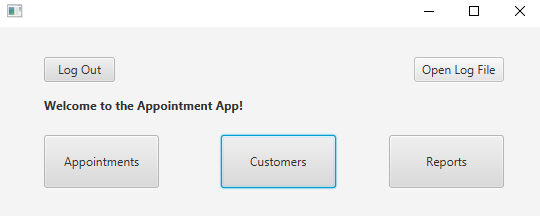
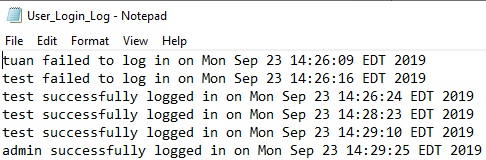
**Requirements Walkthrough for C195**

1. **Create a log-in form that can determine the user’s location and translate log-in and error control messages into 2 languages**
   1. By accessing appointmentapp\_tuannguyen/AppointmentApp\_TuanNguyen.java you can uncomment line 49 to have the login menu display in Spanish. Leaving it commented out will display the menu in English.
2. **Provide the ability to add, update, and delete customer records in the database, including name, address, and phone number**
   1. Once you log in, you can access the Customers menu where there will be buttons for adding customers, editing existing customers, and deleting customers. Please note, you will not be able to delete customers that have existing appointments.  
      ****  
      ****
   2. The Add/Edit customer information screen has all fields relevant to a customer  
      ****
3. **Provide the ability to add, update, and delete appointments, capturing the type of appointment and a link to the specific customer record in the database**
   1. Once you log in, you can access the Appointments menu where there will be buttons for adding appointments, editing existing appointments, and deleting appointments.  
      ****
   2. Appointment fields (including type) are captured and appointment customers are set with a dropdown of the customers in the database. The records are set with references in the Customer table.  
      ****
4. **Provide the ability to view the calendar by month and by week**
   1. On the Appointments menu, you can filter for “All” appointments, appointments for the upcoming “Month” (next 30 days), or appointments for the upcoming “Week” (next 7 days).  
      All Filter  
      ****  
      Month Filter  
      ****  
      Week Filter  
      ****
5. **Provide the ability to automatically adjust appointment times based on user time zones and daylight saving time**
   1. Appointment times within the application are shown in local time based off the time zone of the user’s computer. Times are saved in UTC within the database and converted to local time once retrieved.
   2. The last appointment here was made in PDT for 1:15-1:30pm  
      ****
   3. Changing to EDT shows all the appointments moving 3 hours ahead  
      ****
   4. Turning the clock to 11/4 (Daylight Savings Time ends on 11/3), we see that the time adjusts automatically back 1 hour.  
      ****
6. **Write exception controls to prevent the following and incorporate 2 different mechanisms of exception control:**
   1. **Scheduling appointments outside business hours**
      1. User is limited to scheduling between the hours of 09:00 and 17:00 locally  
         **** 
      2. Try catches were implemented in View\_Controller/AppointmentAddEditController.java and begins on line 125. The exception is caught on line 192 and line 206.
   2. **Scheduling overlapping appointments**
      1. An error will appear if the user tries to schedule an appointment that conflicts one of the user’s other appointments or if it conflicts with another appointment the customer already has.  
         ****
      2. Here are try to add a new appointment for the “admin” user
         1. Trying to add a new appointment that conflicts with the 10/7 10am-12pm user appointment  
            ****
         2. Trying to add a new appointment that conflicts with the 9/30 10:30am-12pm customer appointment  
            ****
         3. This error is caught using the same try catch block as exception 1 handling. The exception is caught with logic starting on line 222.
   3. **Entering nonexistent or invalid customer data**
      1. The customer does not allow any blank fields except for Address2  
         ****
      2. The phone field does not allow alphabetical characters  
         ****
      3. The city and country fields cannot have numerical characters  
         ****
      4. Customer field error handling are done similarly to the Appointment fields with a Try Catch block. In file View\_Controller/CustomerAddEditController.java lines 107 through 152 handles the Customer fields.
   4. **Entering incorrect username or password**
      1. Nonexistent username error message  
         ****
      2. Incorrect password username error  
         ****
      3. Username and password error throws an exception of the respective error message. The exception handling can be found in View\_Controller/LoginController.java on line 88 through 113. Exceptions are thrown on lines 100 and 113.
7. **Write 2 or more lambda expressions to make your program more efficient, justifying each lambda expression with an in-line comment**
   1. Although there are plenty of lambda expressions throughout the program, here are 2 examples:
      1. In the View\_Controller/AppointmentScreenController.java file, lines 136 through 163 show how Lambda Expressions were implemented for the Week and Month filter buttons to create a FilteredList. In-line comments are on lines 136-138.
      2. In the View\_Controller/LoginController.java file, lines 149 through 165 show a Lambda Expression used to display the alert if there is an upcoming appointment within 15 minutes of logging in.
8. **Write code to provide an alert if there is an appointment within 15 minutes of the user’s log-in**
   1. In the View\_Controller/LoginController.java file, lines 149 through 165 show the alert that displays if there is an upcoming appointment within 15 minutes of logging in. Here I have changed one of my appointments to be 2:15-2:30pm on 9/23.  
      
   2. Here is the alert that appears after I change my clock to 9/23 at 2:11pm.  
      ****
9. **Provide the ability to generate the following reports:**
   1. The following reports can be found in the Reports menu  
      ****  
      ****
   2. **Number of appointment types by month**
      1. Here the user selects the desired month and clicks “Calculate” the number of meetings for each appointment type for that month will be populated in the table.  
         **** 
   3. **Schedule for each consultant**
      1. Here the user selects the consultant (user) and clicks “Filter” to have the table of appointments filter only for that consultant.  
         ****  
           
         ****
   4. **One additional report**
      1. I chose to create a third report that calculates the number of customers in the database  
         ****
10. **Provide the ability to track user activity by recording timestamps for user logins in a .txt file. Each new record should be appended to the log file if the file already exists**
    1. The code to log successful and unsuccessful login attempts is in View\_Controller/LoginController.java on lines 91, 104, and 129.
    2. The log file is generated in the root folder of the project (AppointmentApp\_TuanNguyen) as “User\_Login\_Log”.
    3. It will be created if it does not exist and the file will be appended to with each attempt
    4. The file can also be accessed through the Main Screen by clicking “Open Log File”  
       ****  
         
       ****