

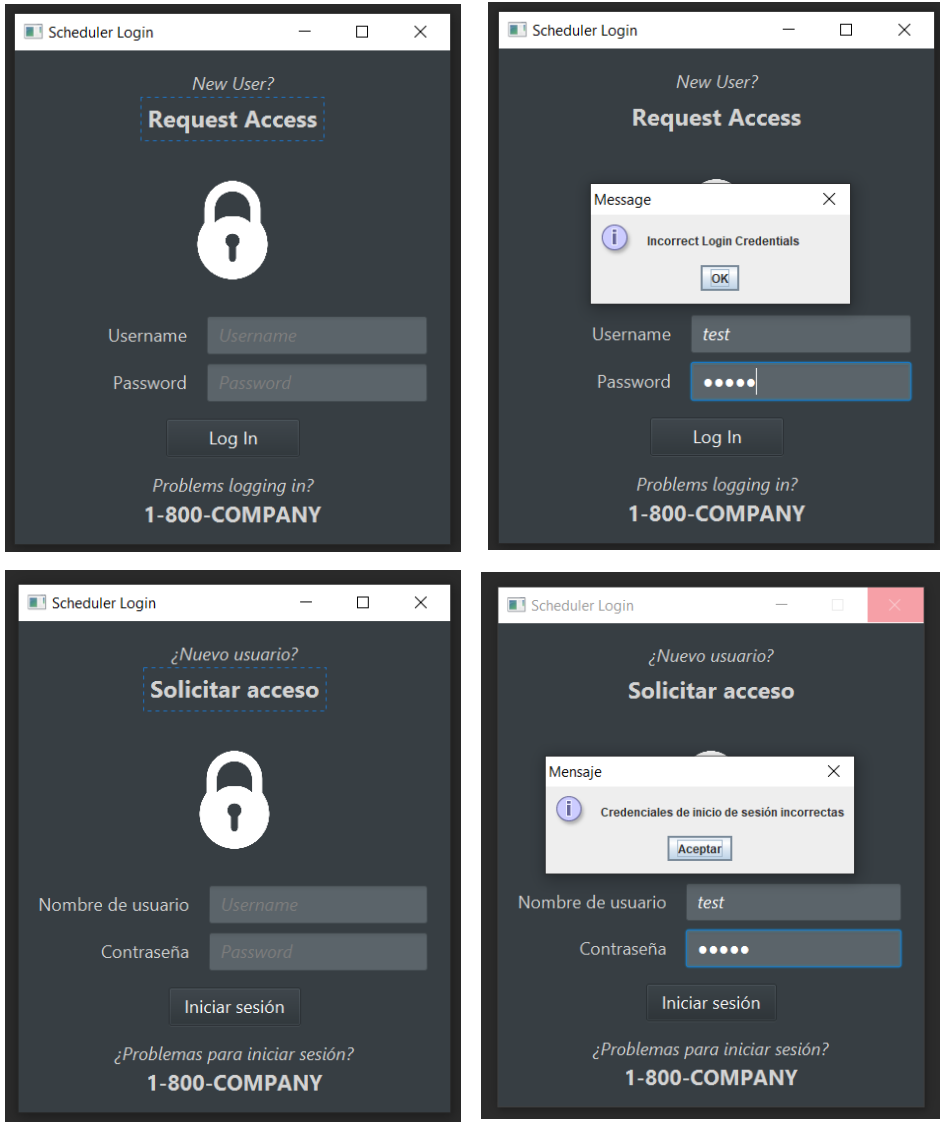
C195 - Software II

Java Application Development

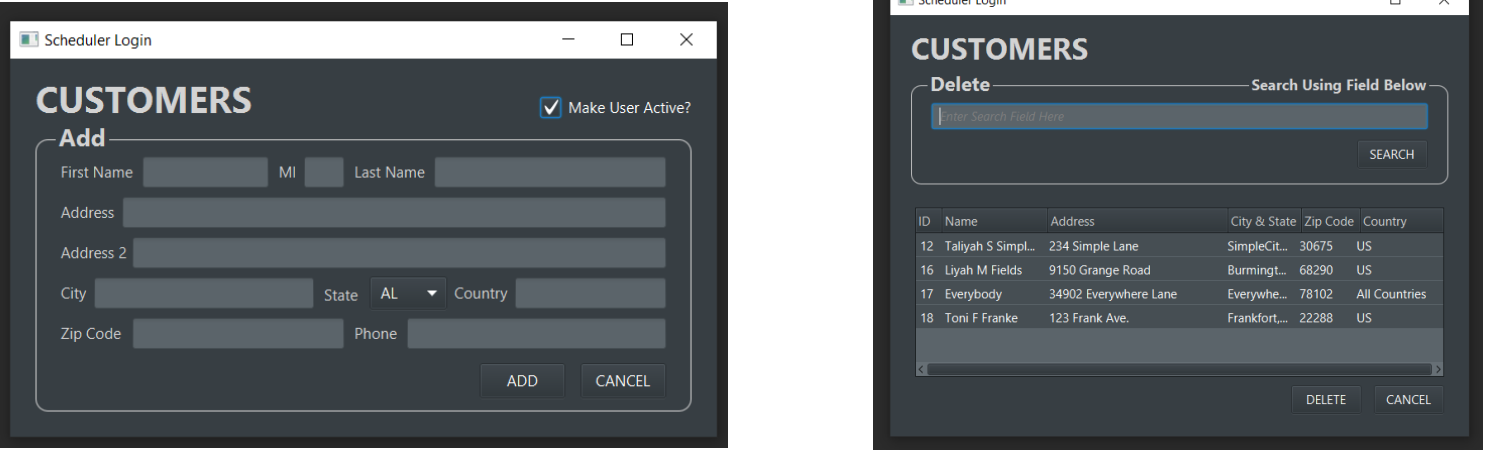
— Scheduling Application —

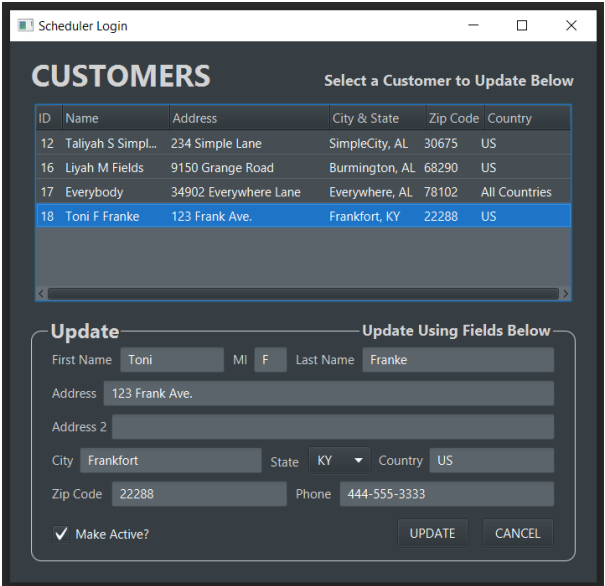
Username test and password test can be used to access dashboard, or you can select ‘Request Access’ link to add a new user. Please note that loading has been slow for the dashboard views. After every operation (ex. adding customer, updating appointment, etc.), the scene will revert back to the dashboard view. Please be patient during the loading process.

- A. Create a log-in form that can determine the user’s location and translate log-in and error control messages (e.g., “The username and password did not match.”) in- to **two** languages.

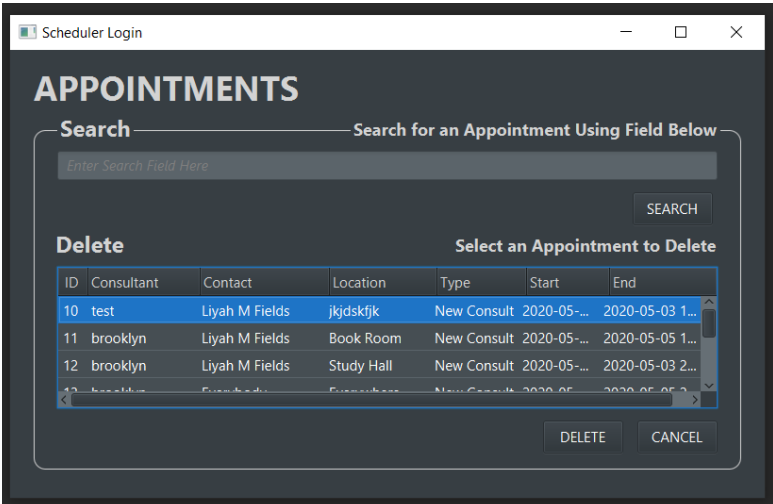
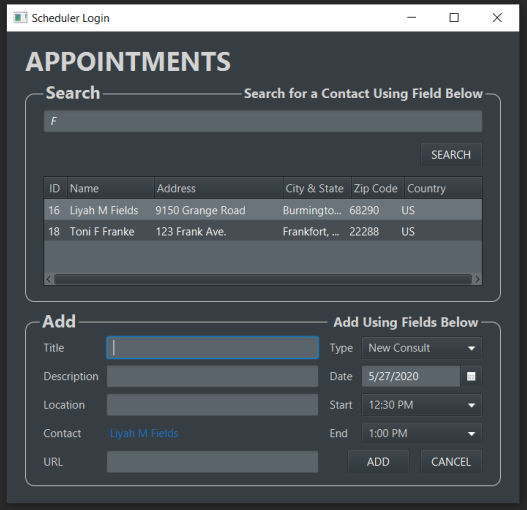
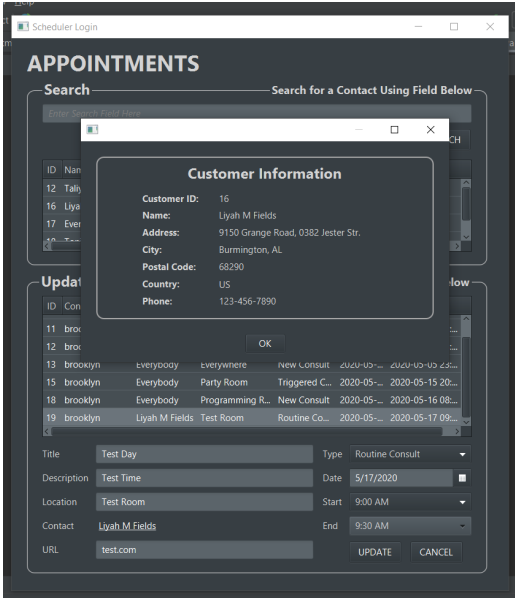
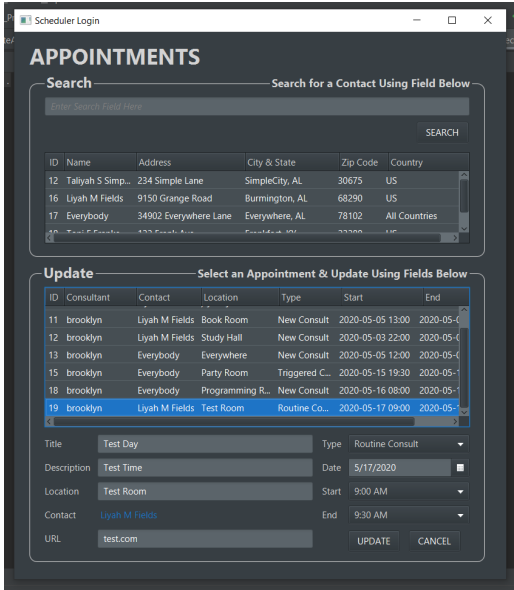


- B. Provide the ability to add, update, and delete customer records in the database, including name, address, and phone number.



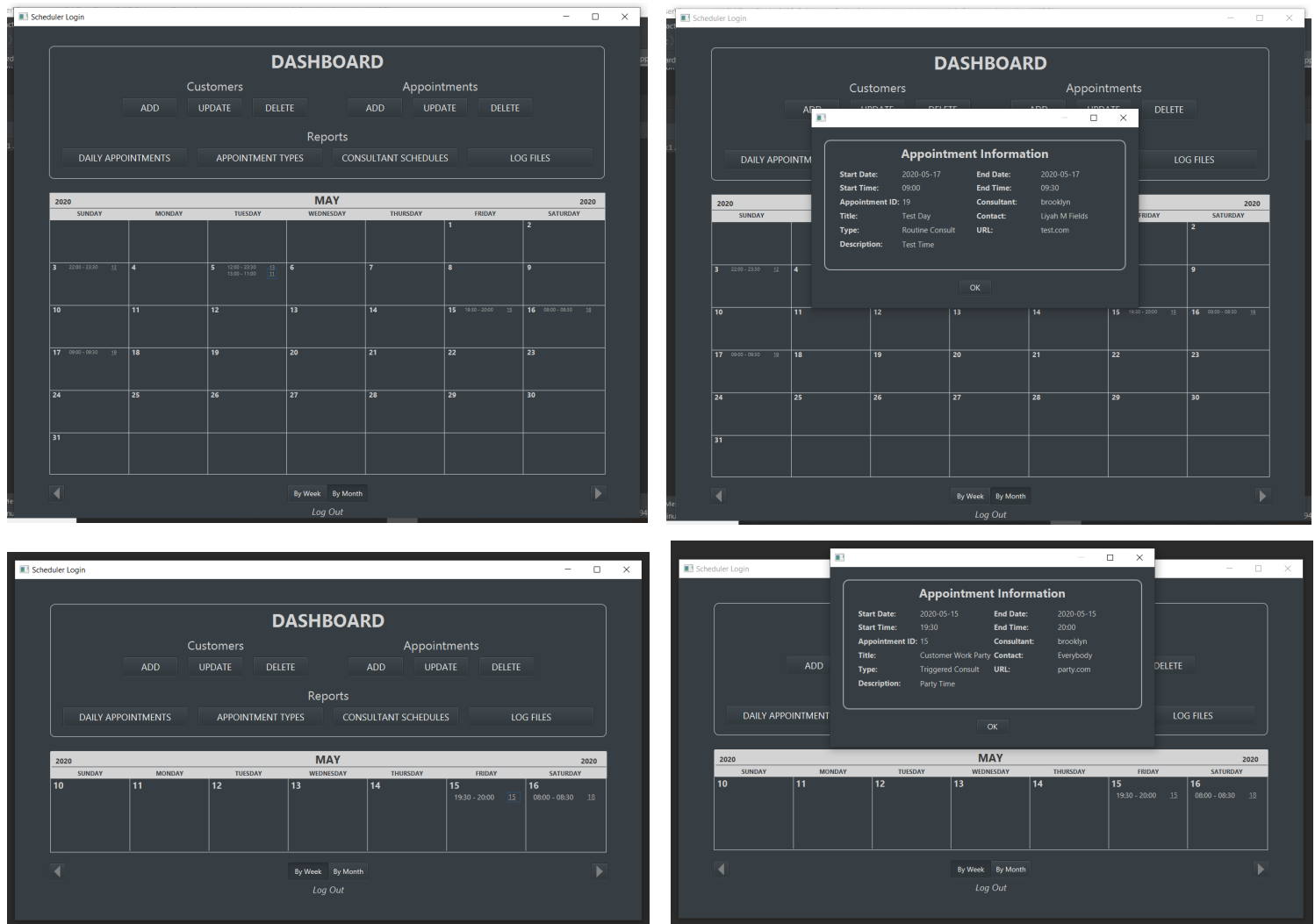


C. Provide the ability to add, update, and delete appointments, capturing the type of appointment and a link to the specific customer record in the database.



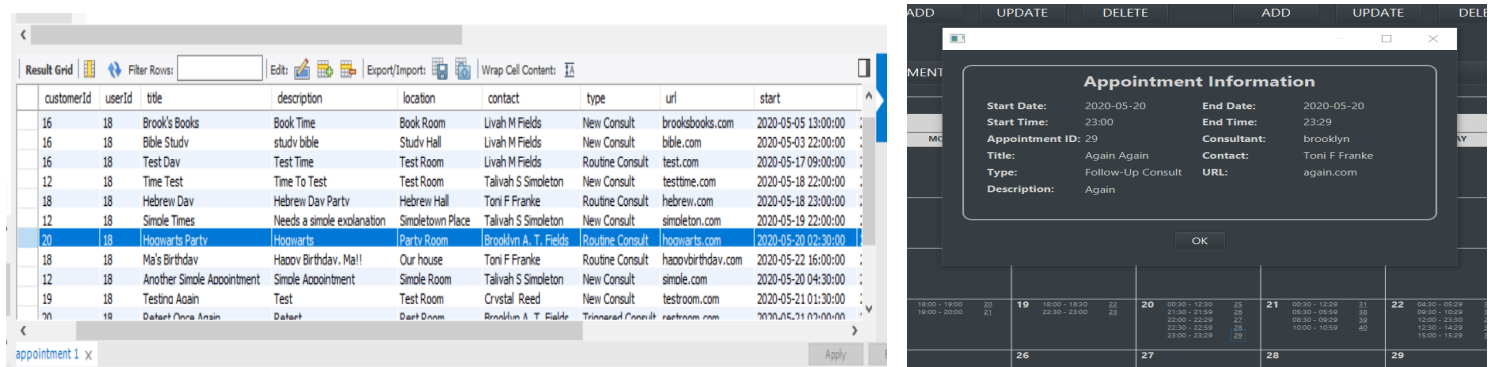
Notes: A search function was added to the add and update windows to search for an available contact. Once a contact is selected, the contact is prepopulated below in a hyperlink. When clicked, this hyperlink will open another window with customer detailed information. Another search function was added to the delete appointments window to search for available appointments.

D. Provide the ability to view the calendar by month and by week.



Notes: A toggle button at the bottom of the scheduler dashboard is used to alternate between a monthly calendar view and weekly calendar view. I designed both views to utilize arrows beneath each corner to alternate between each month in the monthly view or each week with the weekly view. **Please note that loading has been very slow for the dashboard views. Please be patient during the loading process.** Since putting the entire appointment information in small boxes isn't very feasible, I put only the appointment times and a hyperlink of the appointment ID. When clicking the hyperlink, an additional window is opened with the appointment details listed.

E. Provide the ability to automatically adjust appointment times based on user time zones and daylight saving time.



- F. Write exception controls to prevent *each* of the following. You may use the same mechanism of exception control more than once, but you must incorporate *at least two* different mechanisms of exception control.
- scheduling an appointment outside business hours
 - scheduling overlapping appointments
 - entering nonexistent or invalid customer data
 - entering an incorrect username and password

```

// NonAppointmentController.java
...
LocalDateTime endTime = LocalDateTime.of(dateCB.getValue(), timeB);

if (startTime.isAfter(endTime)) {...}

//checking for overlapping appointments
Boolean doesOverlap = true;
int count = 0;

for (CompleteAppointment appointment : SelectDB.selectAllCompleteAppointments()) {
    if (((appointment.getStart().isAfter(startTime)) && (appointment.getStart().isBefore(endTime))) ||
        ((appointment.getEnd().isAfter(startTime)) && (appointment.getEnd().isBefore(endTime))) ||
        (appointment.getStart().equals(startTime)) || (appointment.getEnd().equals(endTime)))
        count += 1;
}
if (count == 0)
    doesOverlap = false;

String startE = convertLDTtoUTC(startTime);
String endE = convertLDTtoUTC(endTime);
String createDate = convertLDTtoUTC(LocalDateTime.now());
String createdBy = User.getUserName();
String lastUpdateBy = User.getUserName();
if (!doesOverlap) {
    InsertDB.addAppointment(customer.getCustomerId(), String.valueOf(User.getUserId()), title, description);
}
else {
    Alert alert = new Alert(Alert.AlertType.ERROR);
    alert.setTitle("Error");
    alert.setHeaderText("Time Error");
    alert.setContentText("Appointment time cannot overlap a pre-existing appointment time!");
    alert.showAndWait();
    return;
}
}

```

```

...
if (usernameTxt.getText().length() == 0) { // Checking for empty field
    JOptionPane.showMessageDialog(parentComponent: null, empty);
    return;
}
else if (passwordTxt.getText().length() == 0) { // Checking for empty field
    JOptionPane.showMessageDialog(parentComponent: null, empty);
    return;
}
else {
    String username = usernameTxt.getText(); // Collecting the input
    String password = passwordTxt.getText(); // Collecting the input
    if (SelectDB.validate_Login(username, password)) {
        user = SelectDB.selectUser(username, password);
        try {
            PrintWriter pw = new PrintWriter(new FileOutputStream(new File(pathname: "s-
pw.append("Username: ").append(usernameTxt.getText()).append("\t Login Time
pw.close());
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        }

        checkForNearApptmt();

        stage = (Stage) ((Button) event.getSource()).getScene().getWindow();
        FXMLLoader loader = new FXMLLoader();
        loader.setLocation(getClass().getResource("name: "/view/Dashboard.fxml"));
        scene = loader.load();
        stage.setScene(new Scene(scene));
        DashboardController controller = loader.getController();
        controller.setCurrentUser(user);
        stage.show();
        return;
    }
    else {
        JOptionPane.showMessageDialog(parentComponent: null, incorrect);
        return;
    }
}
}

```

- G. Write two or more lambda expressions to make your program more efficient, justifying the use of each lambda expression with an in-line comment.

```

private void checkForNearApptmt() throws SQLException {
    availApptmts = SelectDB.selectAllCompleteAppointments();

    LocalDateTime now = LocalDateTime.now().minusMinutes(1);
    LocalDateTime interval = now.plusMinutes(15);

    FilteredList<CompleteAppointment> filteredApptmts = new FilteredList<>(availApptmts);

    filteredApptmts.setPredicate(apptmt -> { //using lambda expression to shorten previous code written that is used to detect an upcoming appointment within the next 15 minutes
        return apptmt.getStart().isAfter(now) && apptmt.getStart().isBefore(interval);
    });

    if (filteredApptmts.isEmpty()) {
        String contact = filteredApptmts.get(0).getContact();
        String startTime = filteredApptmts.get(0).getStart().toLocalTime().toString();
        Alert alert = new Alert(Alert.AlertType.INFORMATION);
        alert.setTitle("Appointment Alert");
        alert.setHeaderText("");
        alert.setContentText("You have an appointment soon with " + contact + " at " + startTime + "! Please check your calendar for more information.");
        alert.showAndWait();
    }
}
}

```

```

public void setDailyApptmts(LocalDateTime time) { //using lambda to create a list of daily appointments from the database-collected monthly appointments list
    if (time == null)
        return;
    if (!allDailyApptmts.isEmpty())
        allDailyApptmts.clear();
    allDailyApptmts = allWeeklyApptmts.stream()
        .filter(apptmt -> (apptmt.getStart().toLocalDate().equals(time.toLocalDate())))
        .collect(Collectors.toCollection(ArrayList::new));
}
}

```

H. Write code to provide an alert if there is an appointment within 15 minutes of the user's log-in.

```
ObservableList<CompleteAppointment> availApptmts = FXCollections.observableArrayList();

private void checkForNearApptmt() throws SQLException {
    availApptmts = SelectDB.selectAllCompleteAppointments();

    LocalDateTime now = LocalDateTime.now().minusMinutes(1);
    LocalDateTime interval = now.plusMinutes(15);

    FilteredList<CompleteAppointment> filteredApptmts = new FilteredList<>(availApptmts);

    filteredApptmts.setPredicate(apptmt -> { //using lambda expression to shorten previous code written that is used to detect an upcoming appointment within the next 15 minutes
        return apptmt.getStart().isAfter(now) && apptmt.getStart().isBefore(interval);
    });

    if (!filteredApptmts.isEmpty()) {
        String contact = filteredApptmts.get(0).getContact();
        String startTime = filteredApptmts.get(0).getStart().toString();
        Alert alert = new Alert(Alert.AlertType.INFORMATION);
        alert.setTitle("Appointment Alert");
        alert.setHeaderText("");
        alert.setContentText("You have an appointment soon with " + contact + " at " + startTime + "! Please check your calendar for more information.");
        alert.showAndWait();
    }
}
```

I. Provide the ability to generate *each* of the following reports:

- number of appointment types by month
- the schedule for each consultant
- one additional report of your choice

Today's Appointments - Notepad

```
File Edit Format View Help
***** MAY 16, 2020 *****
Appointment ID: 18
Date: 2020-05-16
Start Time: 08:00
End Time: 08:30
Contact: Everybody
Title: Programming Time
Description: New Programming Tips
Location: Programming Room
Type: New Consult
URL: program.com
```

Consultant Schedules - Notepad

```
File Edit Format View Help
***** Consultant: test *****
Appointment ID: 10
Date: 2020-05-05
Start Time: 15:30
End Time: 17:30
Contact: Liyah M Fields
Title: fkdjsfkl
Description: jfkdsjfk
Location: jfkdsjfk
Type: New Consult
URL: fkdjsfkl

***** Consultant: brooklyn *****
Appointment ID: 11
Date: 2020-05-05
Start Time: 13:00
End Time: 11:00
Contact: Liyah M Fields
Title: Brook's Books
Description: Book Time
Location: Book Room
Type: New Consult
URL: brooksbooks.com

Appointment ID: 12
Date: 2020-05-03
Start Time: 22:00
End Time: 23:30
Contact: Liyah M Fields
Title: Bible Study
Description: study bible
Location: Study Hall
Type: New Consult
URL: bible.com

Appointment ID: 13
Date: 2020-05-05
```

Appointment Types By Month - Notepad

```
File Edit Format View Help
*****NEW CONSULT*****
JANUARY 2020: 0
FEBRUARY 2020: 0
MARCH 2020: 0
APRIL 2020: 0
MAY 2020: 5
JUNE 2020: 0
JULY 2020: 0
AUGUST 2020: 0
SEPTEMBER 2020: 0
OCTOBER 2020: 0
NOVEMBER 2020: 0
DECEMBER 2020: 0

*****FOLLOW-UP CONSULT*****
JANUARY 2020: 0
FEBRUARY 2020: 0
MARCH 2020: 0
APRIL 2020: 0
MAY 2020: 0
JUNE 2020: 0
JULY 2020: 0
AUGUST 2020: 0
SEPTEMBER 2020: 0
OCTOBER 2020: 0
NOVEMBER 2020: 0
DECEMBER 2020: 0

*****TRIGGERED CONSULT*****
JANUARY 2020: 0
FEBRUARY 2020: 0
MARCH 2020: 0
APRIL 2020: 0
MAY 2020: 1
JUNE 2020: 0
JULY 2020: 0
AUGUST 2020: 0
```

J. Provide the ability to track user activity by recording timestamps for user log-ins in a .txt file. Each new record should be appended to the log file, if the file already exists.

loginTimes - Notepad

```
File Edit Format View Help
Username: test Login Time: 2020-04-25T08:52:41.490
Username: admin Login Time: 2020-04-25T08:54:18.563
Username: test Login Time: 2020-04-25T09:03:58.196
Username: test Login Time: 2020-04-25T09:10:46.351
Username: test Login Time: 2020-04-25T09:48:19.189
Username: test Login Time: 2020-04-25T10:12:00.500
Username: test Login Time: 2020-04-25T10:23:49.831
Username: test Login Time: 2020-04-25T10:30:07.832
Username: test Login Time: 2020-04-25T10:32:04.766
Username: test Login Time: 2020-04-25T12:47:14.473
Username: test Login Time: 2020-04-25T12:51:54.363
Username: test Login Time: 2020-04-25T13:07:27.853
Username: test Login Time: 2020-04-25T13:44:51.049
Username: test Login Time: 2020-04-25T14:03:43.711
Username: test Login Time: 2020-04-25T15:06:43.920
Username: test Login Time: 2020-04-25T15:17:54.380
Username: test Login Time: 2020-04-25T16:07:19.249
Username: test Login Time: 2020-04-25T16:21:07.358
Username: test Login Time: 2020-04-25T16:53:14.353
Username: test Login Time: 2020-04-25T16:55:04.430
Username: test Login Time: 2020-04-25T17:00:33.632
Username: test Login Time: 2020-04-25T17:04:40.160
Username: test Login Time: 2020-04-25T17:21:27.254
Username: test Login Time: 2020-04-25T17:22:25.502
Username: test Login Time: 2020-04-25T17:33:58.524
Username: test Login Time: 2020-04-25T17:36:02.949
Username: test Login Time: 2020-04-25T17:37:19.679
Username: test Login Time: 2020-04-25T17:46:03.695
Username: test Login Time: 2020-04-25T18:00:41.486
Username: test Login Time: 2020-04-25T18:07:11.645
Username: test Login Time: 2020-04-25T18:11:52.769
Username: test Login Time: 2020-04-25T18:14:58.206
Username: test Login Time: 2020-04-25T20:50:17.201
Username: test Login Time: 2020-04-25T21:17:40.071
Username: test Login Time: 2020-04-25T21:21:19.432
Username: test Login Time: 2020-04-25T21:51:19.372
Username: test Login Time: 2020-04-25T22:17:04.211
Username: test Login Time: 2020-04-25T22:19:06.082
```

K. Demonstrate professional communication in the content and presentation of your submission.