IMD 4008: Mobile User Interfaces – Design & Development – Fall 2018
Assignment 2 *and* 3: Meeting Scheduler – Design & Development

Due Dates: ***Assignment 2***: Friday, November 8; ***Assignment 3***: Friday Nov. 22 (both at 11:59pm)
Late Submissions: Accepted up to 4 days late at a 25% per day penalty.
Grading: Assignment 2 is worth 6% of your final grade; Assignment 3 is worth 7% of your final grade
Group Work: Both assignments can be completed alone or in pairs.

Introduction

Assignments 2 and 3 involve first designing (A2) and then implementing (A3) the UI for a meeting scheduling application for planning meetings. The scheduler should support at least two different views: one for entering meetings of certain types (e.g., senior-project meetings vs. work meetings vs. hanging out with friends) and one to present the user with a list of meetings and the ability to access their relevant information (e.g., time, location, duration, etc.). You have some freedom in how this is presented and the types of information you keep track of, but ultimately, your design should make use of **good layout and navigation principles**, as discussed in class, to change between views, as well as use appropriate timed notifications to inform the user when meeting times are approaching.

Assignment 2 will focus on the *design* of this application. Assignment 3 will involve *developing* the application you designed in A2. Keep this in mind as you design your application; you should keep your design grounded in something you expect you'll be able to implement. Please see the course syllabus for a schedule of upcoming topics relative to plan the features of your application – most of the core functionality of this application has already been covered (e.g., Tutorials 3 and 4) and notifications will be covered in Week 8 (tutorial 5).

Assignment 2: Design

For assignment 2, you will design your meeting scheduler and model competing alternatives with StEM to get predictions of interaction times.

Breadth of Designs [3 marks]

Before starting development of your scheduler, you will first design some "competing" alternatives. Start by drafting some designs (sketches, etc.) that address the Core Functionality (as well as Additional Features you are considering) listed below. As usual, this should demonstrate a *broad range* of possible prototypes, rather than any particularly highly detailed or "deep" prototypes. Submit scans of these sketches to demonstrate the breadth of your designs.

Refined Wireframes [3 marks]

As you converge to preferred designs, you will likely refine these further. Pick your two favourite refinements, and wireframe them following the concepts presented in class. These should be a bit more detailed than sketches. You are encouraged to use the sample tools shown in class (e.g., based on the Mendoza book), or use your choice of other wireframing tool (e.g., Balsamiq, …). This should show refinement over the initial sketches, and more closely resemble complete apps (although without real content; interaction elements should be clear though, e.g., using plausible looking controls, proper keyboards for text input, and so on).

Predictive Modeling of Alternatives [9 marks total]

The wireframes should be designed to support the core user tasks of the scheduler tool, see below. These mainly include adding meeting details using the scheduler view, then reviewing it in the scheduled meeting

list. Both of the designs you wireframe above should support these tasks, as well as any extra features you implement.

Once you have two wireframes that support these tasks (again, following the Core Functionality and Additional Features below) determine which design offers better user performance by modeling and testing each with StEM. Import your wireframes into the StEM tool, and model the following tasks with **both** of your designs:

1) Add a meeting **[*2 marks*]**
2) Delete a meeting **[*2 marks*]**
3) Find a specific meeting via the meeting list **[*2 marks*]**

Pick one of these three tasks and present two variations of the modeling [*3 marks*]. Details are up to you, but this might be adding a meeting in the near future vs. the distance future, updating a meeting to remove 1 or N people from the attendees list, adding a new meeting with one person vs. 10 people, and so on. The point is to test your competing designs under a variety of circumstances and to be critical of your designs. Grading will depend on defining appropriate variations (i.e., different enough to really test the edge cases of your two designs). Consider the examples provided with the StEM tutorials for inspiration here. For example, adding a large number of items vs. a small number of items to a shopping cart yields different results depending on the design (e.g., direct text input of numbers vs. using a slider or button to select).

All of your tasks must be modeled to sufficient detail to provide realistic estimates of usage times – in other words, each button press, swipe, gesture, keystroke, etc. must be modeled for each task for each design alternative. They must also be compared fairly between design alternatives – for example, if adding a meeting involves entering a name, the name compared between both designs should be the same and hence involve the same text input sequence. Using StEM, figure out average times for each design for each task. Provide a short report that includes your StEM results, and declares a "winning" design – which you will go on to build in Assignment 3.

Assignment 3: Development

For assignment 3, implement the "winning" design from Assignment 2. Your project should run on API 26. This is what the TAs will test on when grading, so deviation from this may result in your scheduler not working properly when they get it.

Core Functionality [12 marks]

Your scheduler should support at minimum the following capabilities:

- Entering a meeting with the following details **[*1 mark*]**:
  - Name
  - Other attendees
  - Date and time
  - Type of meeting
  - Location
  - Importance (e.g., based on criteria you deem important)
  - Any other details (e.g., an "other" field, or additional fields you feel are necessary).

- Delete a user-specified meeting from the list [***1 mark***]

- Navigation [***5 marks total***]
  - There should be at least two different screens with some kind of navigation control to move between them. The first screen should be the meeting entry screen, as described above. The second screen should present some kind of summary list of meetings previously scheduled, with the option to update information on them [***2 marks***]
  - These should use fragments as appropriate, with persistent data between them (e.g., ViewModel). **[*3 marks*]**

- Set timed notifications based on the entered time/date so that the app produces a push notification at a user chosen time (e.g., *n* hours/days/weeks, etc.) before the user-set meeting time [***3 marks***].

- Coding Style **[*2 marks*]**
  - Use good coding style, variable names, method decomposition, efficiency, code encapsulation, class structure, commenting, etc. (Note that this item does not apply to the *design* aspect of A2, as coding style is not evident in a design).

Additional Features [8 marks]

For full credit, your scheduler should support any two of the following features. They are worth **[*4 marks*]** each. Note that some are easier than others, but all are equally weighted.

- *Store previously set meetings:* Store the set meeting details in some permanent fashion (e.g., to the device storage), so as to restore the list each time the application is run.
- *Calendar reminders:* Upon setting a meeting, add a reminder to the Android system calendar; this should appear as an event when viewing the calendar on the device.
- *Call to action:* Add the ability that when tapping a notification produced by the app (or button on a notification, or similar, at your discretion) that the application carries out some logical action – for example, launching into the app (e.g., the meeting list, details of the particular meeting, etc.), allowing to add additional notifications at a later time (e.g., "Remind me in 3 days", or similar).
- *Distinct Notifications:* Use distinct properties for each type of meeting (or based on attendees, type of meeting, etc.) See NotificationChannels for details of what is possible. For example, each meeting types notifications might have a distinct background colours, vibration patterns, sounds, notification icons (small and/or large icons), and so on. For full credit, these should be quite distinct.
- *Edit/Delete Meetings*
  - Give the user the ability to remove and/or edit meeting details that have been previously set in the meeting review list. What form this interaction takes is up to you
- *Clear Meetings by Filter*
  - Give the user the ability to wipe out all meetings on a certain day, or of a certain type (or based on some other filter detail). What the UI looks like for this is up to you.
- *Send email invites to other meeting attendees*
  - Upon adding other attendees to a given meeting, the application sends an email to them (i.e., you will need to also provide their email). More points if it accesses the system contact list (i.e., making it easier on the user) and allows the recipient to accept/decline the invite. *Note:* This item will not be covered in class and will involve independent research on your part to get it working.

Other Notes:

- Consider using a ListView for the meeting list (although this is not mandatory, it may make your life easier)
- It should be possible to have multiple meetings on the same day
- How you handle meetings that have passed is up to you (i.e., whether they are removed from the list, or left with some modifier to show they have passed, etc.)
- If desired, you can disable rotation in this application, and decide if you want a landscape or portrait layout.
- Finally, both A2 and A3 will include marks for the following items:
    - Use good UI design principles in deciding which widgets to use in your scheduler UI (e.g., manual entry of a date using a TextInputLayout instead of a DatePicker is a poor design decision).
    - Nice Layout, Visual Design: Make the design look nice. Look into improving the visual style of Buttons (perhaps ImageButtons) and other Views you use. Use a clean layout, with hierarchical ViewGroups as appropriate for organization.

Submission Notes

Please submit a PDF with scans of any sketches, any notes, etc. showing your design process for the initial mockups. This should also include your wireframes, and a screen cap of your StEM results, so as to justify your choice of UI to develop.

Find your project folder on your hard drive. Zip the entire project folder and submit to the CULearn link by the date and time specified above. Include a comment with your submission (e.g., in the CUlearn submission form) that indicates which of the Additional Features you implemented. If you work with a partner, ensure the partner's name also appears in the submission field (only one partner need submit).

A grading rubric will be made available shortly on CULearn.