### **CSCI 4176 & CSCI 5708**

### **Mobile Computing**

# **Project Update Form**

# Due: In Lab on Wednesday November 21st, 2018 **NUMBER 7**

1. **Participation (Grey Columns Filled in by TA):**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Name (Print)  **Adjust** | | Banner Number | OK? | Git  Data | Comments |
| 1. |  | Jessica Castelino | B00804805 |  |  |  |
| 2. |  | Jismy Johnson | B00813344 |  |  |  |
| 3. |  | Navneet Singh |  |  |  |  |
| 4. |  | Sharon Alva | B00813350 |  |  |  |
| 5. |  | Shubham Narang |  |  |  |  |
| 6. |  | Utsav Soni | B00812689 |  |  |  |

|  |
| --- |
| **Mark / 5** |

1. **Progress Report**

**Overview:**

To Do List is an application which helps users to maintain and monitor their day-to-day tasks. Our target audience for this application are the students currently studying in Halifax. For this application we are using Android DB persistence. DB implementation and mapping of tables is done as per the requirements identified in the analysis. The main functionalities of To Do application include: add a new task, update an existing task, delete a task, tag a location to a task, and set reminder to each task. Other functionalities include use of camera feature for quickly adding a task, getting task notifications, and a visual display of current status of tasks.

**Status:**

Application has the create, update and delete functionalities implemented. A new task can be added to the lists of tasks. When the application is opened, it displays the existing lists of tasks. Implementation of the list view of tasks is completed. The application will display the existing tasks when opened. This is displayed by getting the data from DB at runtime. New task creation is implemented, where on click of a floating button, a pop-up appears. The pop-up prompts for title of the task. Also, sub tasks can be added from this popup. A new pop-up appears to add sub tasks. At a time, only one sub task can be added to the task. Delete option is implemented to remove individual sub tasks from the main task.

A navigation drawer is implemented which opens from the left-most end of the screen. The menu is created to provide quick access of all the pending and completed tasks. Currently, a static menu drawer is implemented which has options; pending tasks, completed tasks, and Feedback. However, implementation of these options is pending.

Every task created can be tagged to a specific location so that user gets notifications when near that location. For this feature, the create pop-up has an option implemented to tag a location. On click of this option, it redirects to a new screen where you can search for a location. This was implemented using Google API. For this, the application requests for Internet permission from the user. When searching for a location, a set of places are displayed in drop down list. This is integrated and saved in database against the respective task.

Another feature implemented within the task creation is adding a reminder date. This option is paced right next to the tag location option. A calendar opens where you can select and save the date for the task.

Camera feature is implemented and integrated into the application for saving an image instead of text in the task. Two options are implemented; one is that user can upload a new image, and second is take from the device’s gallery. The image is displayed just below the sub tasks, if any.

Current location of the user is taken by using the GPS feature of the device. This is implemented using FusedLocationProviderClient API of Google play services. For every task that is tagged to a location will display a notification when the user is near the tagged location. A broadcast notification is implemented where using the current coordinates and the location coordinates are matched to display the notification. This feature is implemented and integrated in the application. It fetches the coordinate values from database to compare with the user’s current location.

For this application, as the User Interface (UI) is important from user’s point of view, a consistent and user-friendly UI is implemented. The font type, size, and color are all maintained consistent throughout the application screens. Also, the size of cards, displaying the lists, change size dynamically to adjust the text in the view. The elements of the screen are adaptive, meaning whatever the screen size be, it will adjust accordingly. Focus on the UI of the application is more to provide a better-quality application.

|  |
| --- |
| **Mark / 5** |

1. **Identification of Problems**
2. Internal project updates

During the initial development phase, there was a lot of confusion on the amount of work completed. Some team members sent updates by e-mail, others on WhatsApp and there was no single place where we could keep a record of what has been done and what is coming next. To resolve this issue, we made use of Slack app. All the communication regarding project updates were restricted to Slack to ensure that everyone was on the same page.

1. Adhering to coding conventions

After the initial commits on GitHub, it was noticed that the code developed violated the Java coding conventions on various levels. As a team, we investigated the problem, invested our time to read the Java coding conventions defined on the Oracle website, and ensured that all rectify it without affecting the timelines.

1. Integration efforts

Integration is crucial to ensure that all parts of the project fit together. Our milestones incorporated development of individual tasks, however, the integration efforts required were not considered. This affected our timelines.

1. Bugs

Minor bugs in the code were detected during the implementation. This was then resolved by discussing within the group. Using this method helped in accomplishing the deliverables faster, as a different approach for a problem, solved the issue.

1. Differences in Gradle version

Some team members found a difficulty to run the code pulled from GitHub after the initial commits were made. On investigating the problem, we realized that there was a mismatch in the Gradle versions used by every team member. As a result, the code updated to GitHub could not be executed and problems related to Gradle sync were encountered. We updated the Gradle to the same version on our machines to resolve this issue.

1. Change in design
2. Database restructuring

We had kept separate tables to store the individual tasks and the reminder details. The location-based reminder functionality, which runs as a background service, compares the user location with the Database needed to be restructured. The database was normalized to BCNF form. and the as the designed DB was not efficient for implementation. Since we realized that redundant data was getting loaded into DB which caused unnecessary work. This is avoided by combining two tables and restructured accordingly.

1. Enhancing architecture
2. Performance
3. Reducing memory footprint

|  |
| --- |
| **Mark / 5** |

1. **Expected Functionality**
2. Add new task
3. Add a sub task:
4. Update existing task:
5. Delete task:
6. Camera feature :
7. Location tagging:
8. Notification:
9. Drawable Menu:
10. Setting reminder for tasks:

|  |
| --- |
| **Total: Out of 15** |

TA Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Instructor Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_