**ANDROID BOOTCAMP PROJECT REPORT**



**TODO APP**

------By Group no. 5

MEMBERS ROLL NO.

Anik Chakraborty 17

Dipan Chakraborty 18

Tirtha Chakraborty 19

Aritra Chatterjee 20

**INTRODUCTION**

Our objective was to create a ToDo application which maintains a record of the user’s tasks that yet to be done. Once registered, the user will be able to store the details(task name, due date, details of what needs to be done) of as many number of tasks as he or she desires.

These tasks can be checked to denote that it has been completed, in which case those tasks will be removed from the task list. The app can handle the task lists of multiple users at a time, but when an user logs in he or she can view only his or her own task list.

**TECHNOLOGIES INVOLVED**

1. **Tools:** Android SDK (Android 9 API 29)
2. **Programming Language:** JAVA
3. **Database:** Realm

**SYSTEM DESIGN**

A number of activities have been incorporated to successfully bring out the functionality of the To-Do application.

**Main activity**:

This is the starting activity of the application. It displays two buttons for the user to either sign up or log in.

**Sign-up activity:**

This activity signifies the entry of a new user. If the user is not registered on this app, he or she has to go through this sign up process(submitting all the necessary credentials) to create an account for managing his or her daily task schedules.

**Log-in activity:**

In case the user is already registered on the app, he or she has to access his or her account by verifying the unique id and his or her password.

**Dashboard activity**:

As the name suggests, this is an user’s private dashboard which displays his or her personal task list. Clicking on any particular task will display the detailed information related to the task. The user may mark the tasks once they are done to remove it from the list. Also the dashboard provides an option to create new tasks, edit user profile and also to logout.

**Create-Task activity:**

This activity is dedicated to make an entry for a new task. It takes the necessary input from the user and culminates all the information together to add a new task to the user’s list.

**Edit Profile activity:**

A registered user may update his or her account information any time through this activity.

**METHODOLOGY**

The app opens with the main\_activity page. It displays two options for the user: sign up and log in. In case the user is not registered on this app and is a new user, his choice must be signup and if he is an existing user he should go for the login option.

If the user chooses to signup, he clicks the signup button on the main activity page, which will open the signup activity. Here, the user is asked to enter his personal information, namely, NAME, EMAIL and MOBILE NUMBER, and also a PASSWORD. With that, when the user clicks the SAVE button, an user id is returned instantly which is an integer and is unique for each user. *[ This is done by an auto-incrementation process. When the user clicks the SAVE button, the maximum id value is fetched from the database and the new id is returned by adding 1 to this maximum value. ]*. Next the user is expected to click on the SUBMIT button, which would store all the user information, including the unique id, into the database. This would lead the user to the login page. Here, he needs to enter his unique id and the PASSWORD. Once he clicks the LOGIN button, his id and password are verified, and if they are found to be correct the dashboard opens.

If the user chooses to login, he or she is directly taken to the login activity(the signup activity in the previous case is skipped).

The dashboard comprises a NAV.Drawer with a hamburger icon which toggles on clicking, to display a menu of: HOME, EDIT PROFILE, LOGOUT. HOME refers to the private dashboard itself. EDIT PROFILE will take the user to the Edit\_Profile activity where the user will be able to update or edit his personal information(but not the user Id ). The changes will be immediately reflected in the database. The LOGOUT option will take the user back to the login activity. .The nav header displays the name of the user currently logged in.

The dashboard lists down all the tasks that the user has created and which has not yet been completed by him. Clicking on any of the tasks will display the detailed account of the task in the dashboard activity itself. Once a task is completed the user can mark it as done by **checking** the task. If the user wishes to remove all the tasks from the list, he is supposed to click on the MARK ALL button(image button) in the dashboard, which will check all the tasks at once. The marked tasks will be removed from the list and will not be displayed from the next login session. However, in case the user has no pending tasks, the dashboard displays a message ‘No Task to Display’.

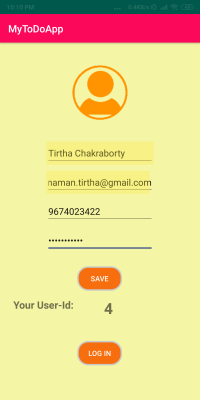
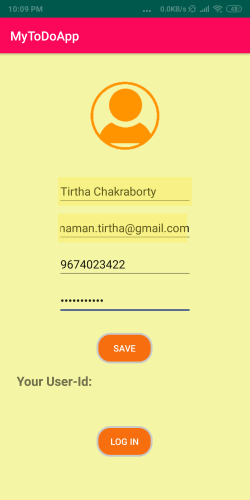
The dashboard also has a provision for the user to go about creating a new task. All he needs to do is click on the CREATE TASK button(image button) which will redirect him to the Create-Task activity.

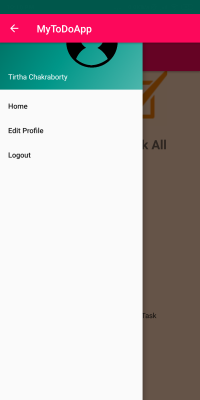
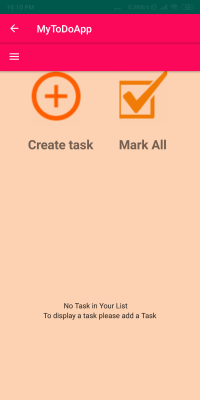
The create-task activity is there to collect information from the user about the new task that he wants to enter, namely, the task name, the date on which the task is due and the details of the task as in how and what all to do. The user may wish to have all his tasks displayed using different colors asper his choice (he might want to maintain a color code, for example: red for official tasks, blue for bill payment tasks, green for some upcoming events etc). This liberty is granted to him in this activity by providing a ColorPicker from which he can choose the one of his choice. The hexcode of the chosen color appears in the adjacent textbox where the font color is the same as that of the chosen color. Even the due date of the task can be selected from a calendar (using a DatePicker) by clicking on the respective button, instead of manually typing it in desired format.

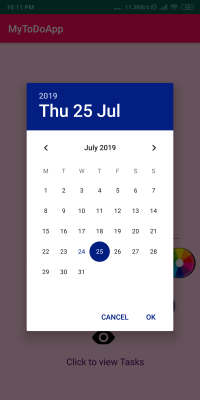
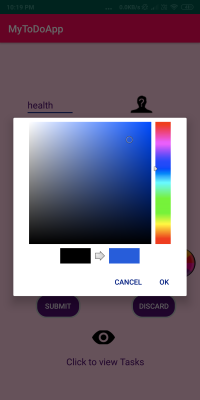
Once done entering all the details and specifications, the user needs to confirm his action by clicking on the SUBMIT button. The click on the SUBMIT button not only ensures the updating of the database with all the details of the new task(including the user’s color choice which is stored in the form of the hexcode), but also redirects to the dashboard where all his tasks her displayed including the one just created. There the user shall see that the new task got is displayed with a background of his chosen color.

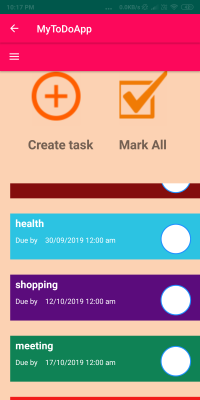
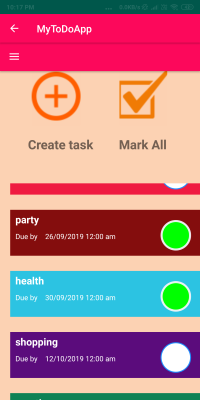
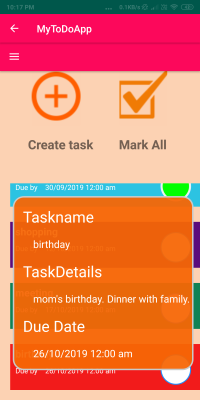
Every time, pressing the back button will take the user to the previous activity. Thus, while in the dashboard activity, pressing the back button will also lead to a logout.

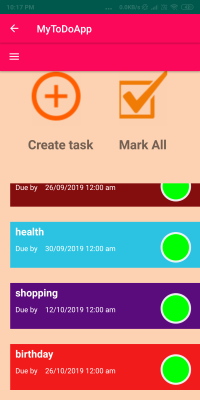
**SNAPSHOTS**







**DATABASE SCHEMA**

A single Realm database was sufficient for our application and it consisted of two Tables as follows:

USER:

|  |  |
| --- | --- |
| **ATTRIBUTE** | **DATA TYPE** |
| userid | Int(primary key) |
| username | String |
| email | String |
| phone | String |
| password | String |

TASK:

|  |  |
| --- | --- |
| **ATTRIBUTE** | **DATA TYPE** |
| taskid | Int(Primary Key) |
| taskname | String |
| taskDetails | String |
| color | String |
| dueDate | Date |
| **ATTRIBUTE** | **DATA TYPE** |
| checked | String(in “true” or “false”) |
| userId | Int(refers to User(userid)) |

**FUTURE SCOPE**

The application can be further modified in the future to make it more efficient for use. For instance, we can generate notifications a few days prior to the due date of the tasks so that the user is kept aware all the while about his pending tasks. It would also be great if some sort of a reminder can be implemented in the form of an alarm, etc, to remind the user that the due date for his task has arrived. Besides this, we can also implement an autodeletion of the tasks once the due date of the task is over, without waiting for the user to mark it as done.

**CONTRIBUTION**

|  |  |  |
| --- | --- | --- |
| **ROLL NO.** | **NAME** | **AREA OF CONTRIBUTION** |
| 17 | Anik Chakraborty | Assisted in Report making |
| 18 | Dipan Chakraborty | Database handling, Connection of front end with backend, Recyclerview List implementation |
| 19 | Tirtha Chakraborty | Front-end/ UI design and implementation, Store user credentials into realm database, generation of autoincremented user id |
| 20 | Aritra Chatterjee | Assisted in Report making |