ToDo

The classic ToDo list

CSD101 FastTrack Project Report

Prof. Harish Chandra Karnick

By: Rishi Balamurugan

2010110883

Table of contents:

1. Problem Statement
2. About
3. Functionality
4. Screenshots
5. Algorithmic ideas
6. Libraries utilized
7. Further work
8. Conclusion

Problem Statement:

We all have the need to remember lots of tasks and notes which may be either quite important or too many in number. Often, we are given a false feeling of productivity when we switch between these tasks and end up doing tasks of lesser priority, while wasting precious time.

Although, there already exist apps that help us to do the same, most contain disruptive ads, may be paid or are just not suited for our needs. Many of them are resource intensive, might require a constant internet connection, are not user friendly or just not suited to our needs.

*ToDo* aims to address these problems as it is lightweight, ad-free and has a very simple and easy-to-use user interface.

About *ToDo*:

*ToDo* is an open-source task management app that allows you to stay organized and keep track of your tasks in a simple way. You can use *ToDo* to track tasks, take notes, create shopping lists, and much more. All while focusing on what matters the most.

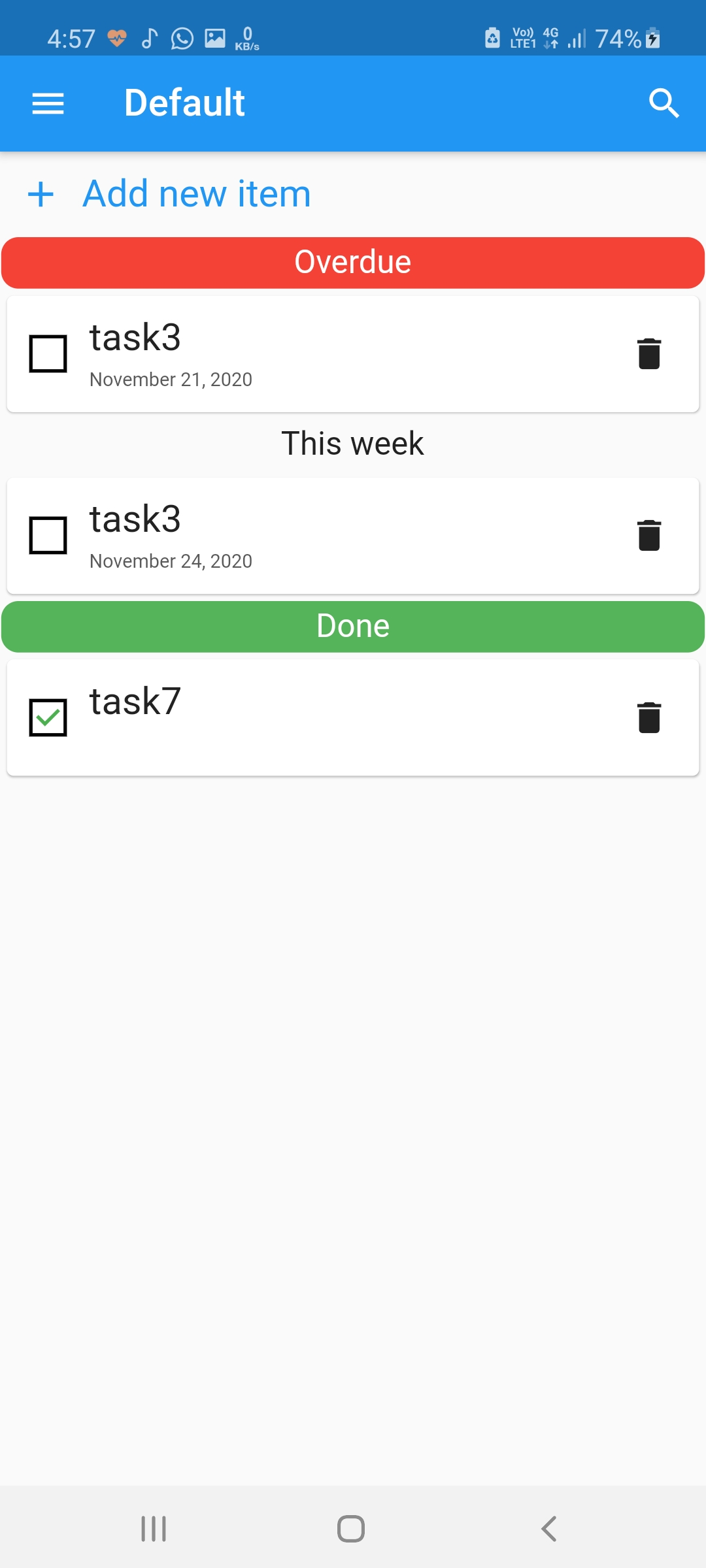
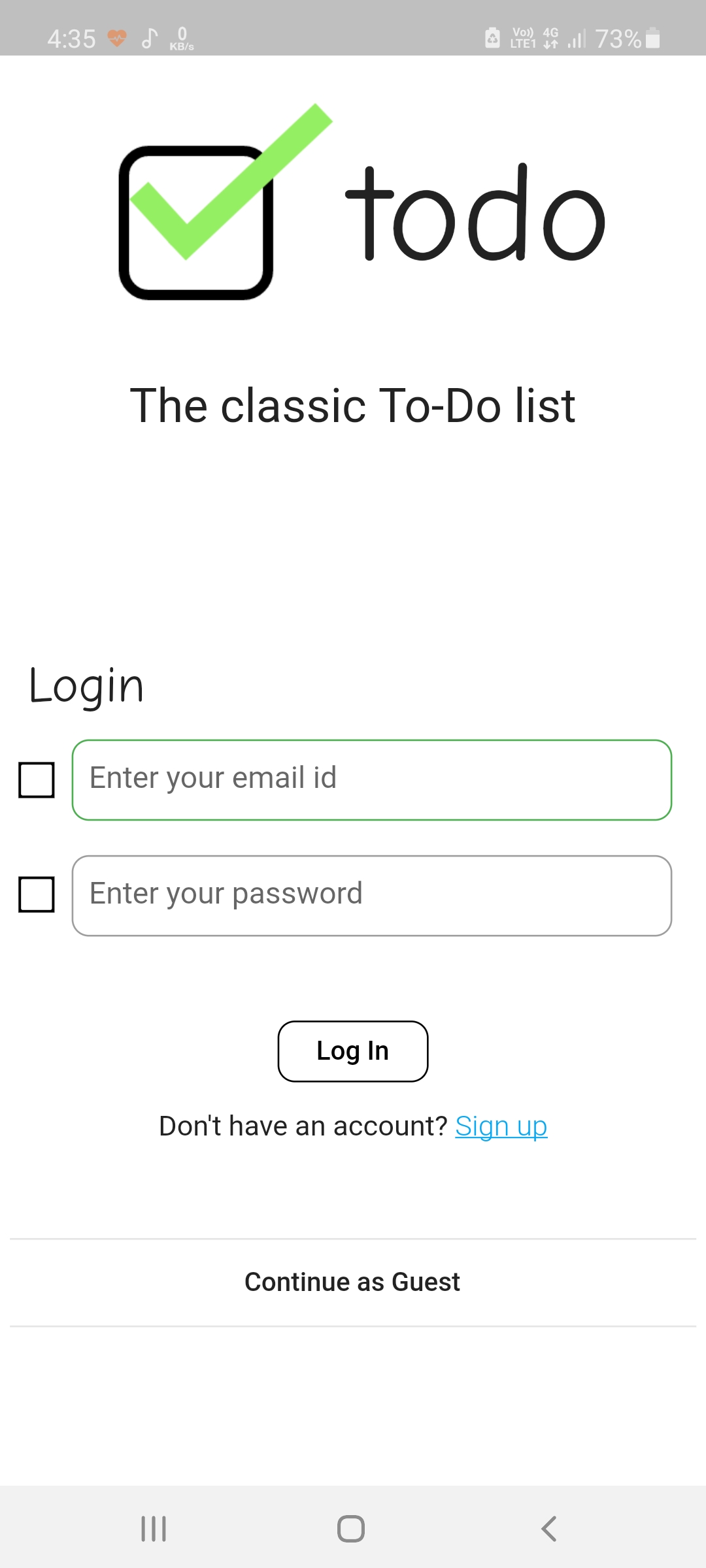
*ToDo* helps you to:

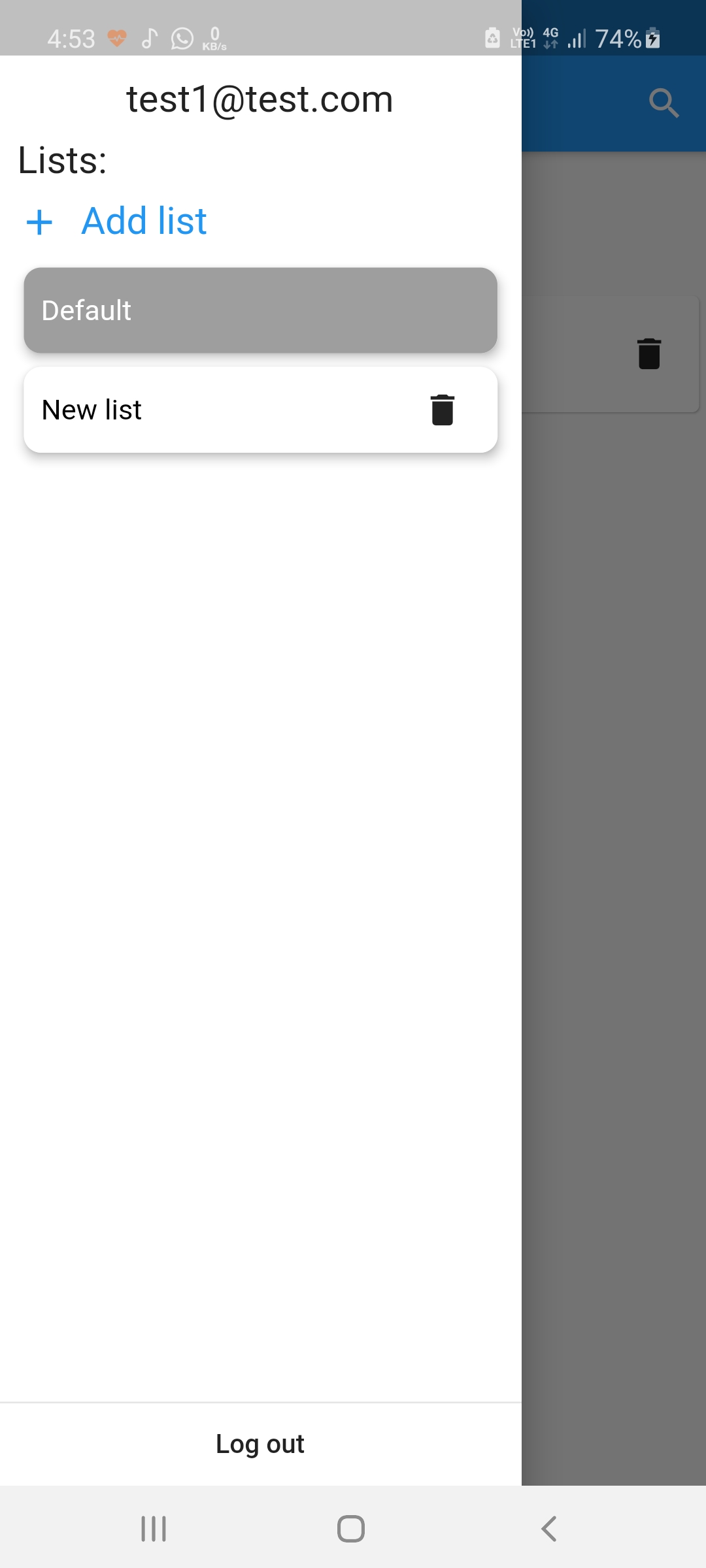
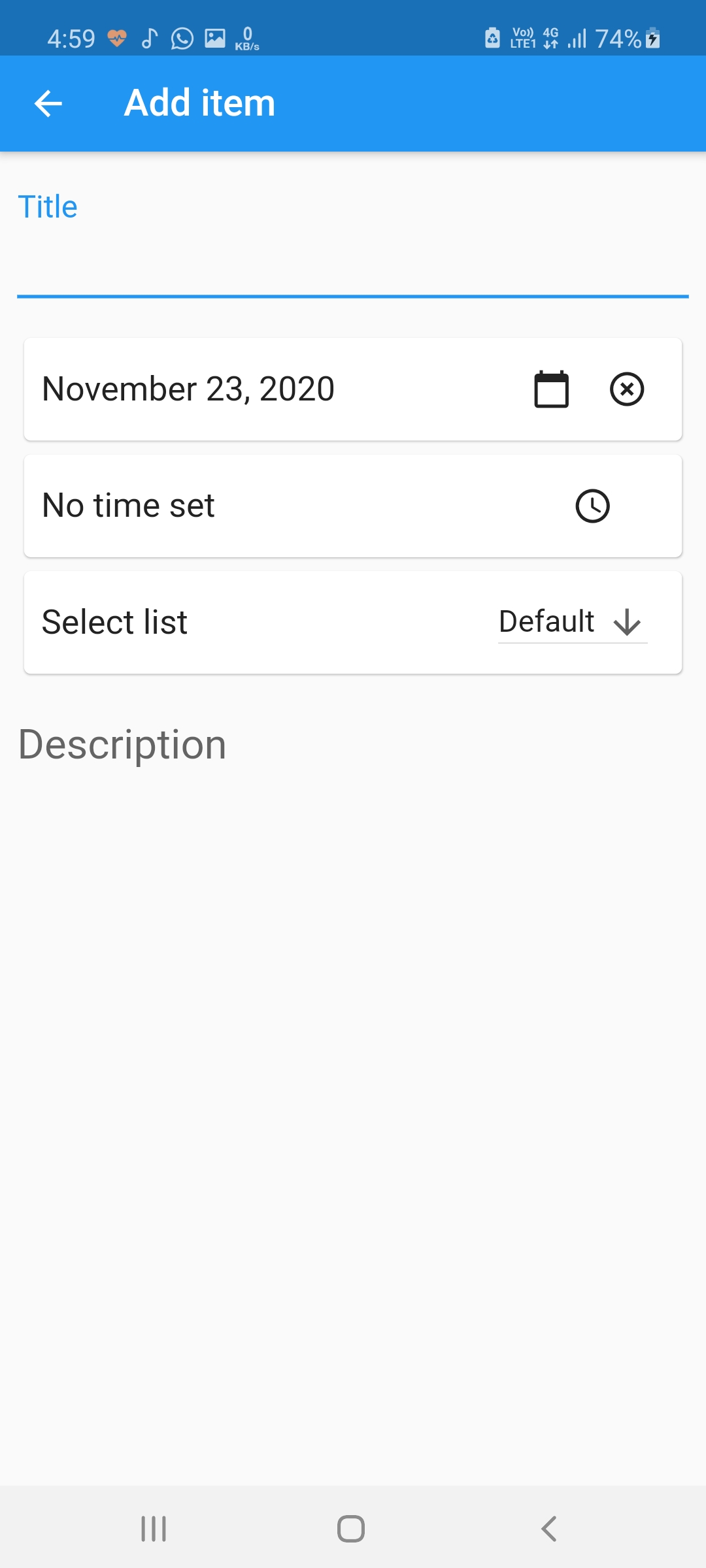
* Organize tasks into lists to increase productivity, save time and group similar tasks.
* Add comments to better inform yourself about the task at hand.
* Easily prioritize tasks by adding a due date and checking the finished tasks.
* *ToDo* shows you tasks that have the closest deadlines and lets you plan your day accordingly.

While you can choose to maintain your notes locally, *ToDo* allows you to sync notes across multiple android devices.

Functionality:

1. Task related:
   1. CRUD operations on tasks
   2. Option to add description to tasks
   3. Tasks are sorted by Due date:
      1. Option to add and remove due date and time
      2. Sorted according to Overdue, This week, Next week, Next month, Later and no due date.
   4. Checkboxes:
      1. Tasks can be checked and unchecked
      2. All checked tasks will be moved to the bottom of the list
2. Multiple Task Lists:
   1. Switch between multiple task lists
   2. Create and delete lists
   3. Add tasks to a specific list
   4. Move tasks between lists
3. Task search:
   1. Search for task names and matching descriptions
4. Sign-in related:
   1. Sign in as guest(local):
      1. Instant sign-in
      2. Stores all data locally
      3. Data is lost upon uninstallation
   2. With email id(Cloud sync):
      1. Create accounts and login using email
      2. Checks for internet connection and syncs tasks in real time
      3. Task stored locally in case of no internet and updates cloud immediately when connection is restored

Screenshots:



Approaches and ideas:

1. Databases:
   1. **SQL storage**: *ToDo* uses the sqflite flutter package to store all the notes on the device. All CRUD operations are handled by the DatabaseHelper class.
   2. Firebase storage: All notes that are stored in Cloud Firestore are used to ensure the local database is properly synced. Every note is stored securely, under firestore rules that only allow access to the user.
2. Sync mechanism:
   1. First trial: Only firebase was used and it worked as planned. No notes were shown when the internet was disconnected.
   2. SQL integration 1.0: Two different widgets were shown according to the internet connection. This had problems with sync when the internet reconnected. As, for every operation the whole sql database was synced.
   3. SQL integration 2.0: In this iteration, sql and firebase were treated as separated entities and both were updated for every operation. The whole sql database was referenced only when initialising and all other instances operated both the databases simultaneously.
   4. Implement sorting according to date: As firebase queries could not directly return processed lists, it was stored in a variable and processed separately. SQL did not pose much of a problem as all data was originally stored in lists.
3. Note data: Firebase and SQL had separate data types to store the due date. Interconversions and formatting took a lot of time and reference to the docs to complete. Many of the other variables had null pointer exceptions that had to be handled in a suitable method.
4. Multiple lists: I decided that creating a separate table and managing the same would be too tedious. Instead, for every new list a new note is added with a unique identification. These notes are used to retrieve the lists and are not displayed. Although this approach has a loophole, the amount of time and effort saved made it worth it.
5. **Internet connectivity**: The internet connection state was not always correct and sometimes gave false positives .To fix this, I used a method from FirebaseAuth that had an error message when the internet was disconnected. This method worked correctly every time and correctly notified the connection when used with the connectivity package.
6. Login mechanism: This was by far the easiest of the bunch, as Firebase provided a simple and secure solution. The only error encountered were the ones that performed operations on the database post sign-in.
7. Checkbox: This problem, although small, took two tries to get right. When the checkbox was put in a bordered container, it gave issues when used in a different device configuration. To solve this, I layered it with the same so that it looked exactly like the logo.

Technologies used:

1. Flutter
2. Firebase
3. Android studio
4. GitHub
5. StackOverflow

Challenges:

1. Designing the note sync system: The logic for this was the hardest to figure out and implement. Coming up with newer and more efficient code took up considerable time. The debugging was very hard as the smallest mistakes in the order of statements caused the databases to go out of sync. Only after a number of iterations and debugging, it started to work without any hitches. It was the most time consuming part of the entire project.
2. Firebase: As this was new to me, everything associated with it was a challenge. From structuring the database, making secure Firebase rules, storing and retrieving data. It was quite a learning experience.
3. Sql database: Flutter required the sqflite plugin for the use of an SQL database. Learning it’s implementation and usage was quite difficult, as it did not support some of the sql functions and also could not store the DateTime variable.
4. Checking for internet connectivity: The connectivity package offered by flutter sometimes did not give the correct output. This turned out to be quite a problem as this is what determined when to completely sync both databases after a loss of internet. I was lucky enough to remember the error messages in the FirebaseAuth docs, which allowed me to solve the problem.
5. Checkbox: Although this was an unexpected problem, the solution was short. Flutter did not allow the checkbox to have the same color combination as the logo. This was solved by layering the two widgets.

Possible extensions:

1. One of the important features for this app, the notifications is still under work.
2. While this app syncs notes across devices, it is yet to support concurrent devices. Which means that, if multiple people are using the same account, the databases may not sync properly.
3. Web support, iOS support. While this app now runs only on android devices, I plan to expand to other platforms.
4. Capture tasks from different applications
5. Option to change the sort by clause of the notes.
6. Alternative sign-in options. Eg: G-mail, facebook, etc.