**Chapter1**

**INTRODUCTION**

**1.1 Definition**

The Daily Essentials Reminder App is an innovative and user-friendly application developed using Android Studio that aims to assist users in managing their daily routine by reminding them about essential objects and food items they need on a regular basis. This app is designed to simplify and streamline the process of remembering and organizing these essential items, ensuring a smooth and efficient daily routine for users. It offers a user-friendly interface, enabling users to create, organize, and prioritize their daily essentials and needs effortlessly. The app's key features include task scheduling, reminders, categorize tasks and much more.

**1.2 Problem Statement**

In today's fast-paced and demanding world, individuals often struggle to keep track of their daily tasks, appointments, and essential activities. The lack of an efficient and organized system leads to missed deadlines, forgotten commitments, and increased stress levels. It can be very frustrating for an individual to loose the time sense in their daily tasks as well and forgetting important activities or essentials. Existing manual methods of task management, such as pen-and-paper lists or basic calendar apps, often prove inadequate in effectively addressing these challenges. Consequently, there is a pressing need for a comprehensive solution that simplifies task management, enhances productivity, and promotes better time management skills.

**Chapter 2**

**REQUIREMENTS**

**FUNCTIONAL REQUIREMENTS**

1. **Task Scheduling and Calendar Integration:**

* Enable users to schedule tasks on specific dates and times.
* Provide a calendar view to display tasks and appointments in a visual format.
* Allow synchronization with other calendar applications to ensure a unified schedule.

1. **Reminder and Notification System:**

* Send customizable reminders and notifications to users based on task deadlines or specified intervals.
* Provide options to set reminder preferences, such as notification sound, vibration, or pop-up alerts.
* Allow users to snooze or dismiss reminders based on their preferences.

1. **Task Prioritization and Sorting:**

* Enable users to prioritize tasks based on urgency, importance, or custom criteria.
* Provide sorting options to arrange tasks by due date, priority, or category.
* Allow users to manually rearrange tasks to reflect their preferred order.

1. **Repeat and Recurring Tasks:**

* Allow users to create tasks that repeat on a daily, weekly, monthly, or custom interval basis.
* Provide options to set the duration and end date for recurring tasks.

1. **Notes and Attachments:**

* Allow users to add notes or additional details to tasks for reference or instructions.
* Provide the ability to attach files or links relevant to a task.

**NON FUNCTIONAL REQUIRERMENETS**

1. **Performance:**

* The app should have fast response times and perform efficiently, even with a large number of tasks and reminders.
* It should be able to handle concurrent user interactions without significant delays or performance degradation.

1. **Reliability**:

* The app should be highly reliable, ensuring that tasks, reminders, and user data are accurately stored and maintained.
* It should have backup and recovery mechanisms to prevent data loss in case of system failures or crashes.

1. **Scalability**:

* The app should be designed to handle an increasing number of users and tasks without compromising performance or user experience.
* It should support scalability in terms of data storage, server infrastructure, and user load.

1. **Usability**:

* The app should have a user-friendly and intuitive interface, allowing users to easily navigate and perform tasks without confusion.
* It should provide clear instructions and visual cues to guide users through different features and functionalities.

1. **Accessibility**:

* The app should be accessible to users with disabilities, complying with accessibility guidelines and standards.
* It should support features such as text-to-speech, voice commands, and adjustable font sizes for improved usability.

1. **Compatibility**:

* The app should be compatible with a wide range of devices, operating systems, and screen resolutions.
* It should undergo compatibility testing to ensure smooth functionality across various platforms and device configurations.

1. **Error Handling and Reporting:**

* The app should effectively handle errors and exceptions, providing clear error messages to users when issues arise.

**Chapter 3**

**SYSTEM DESIGN**

**3.1 ARCHITECTURE DIAGRAM:**

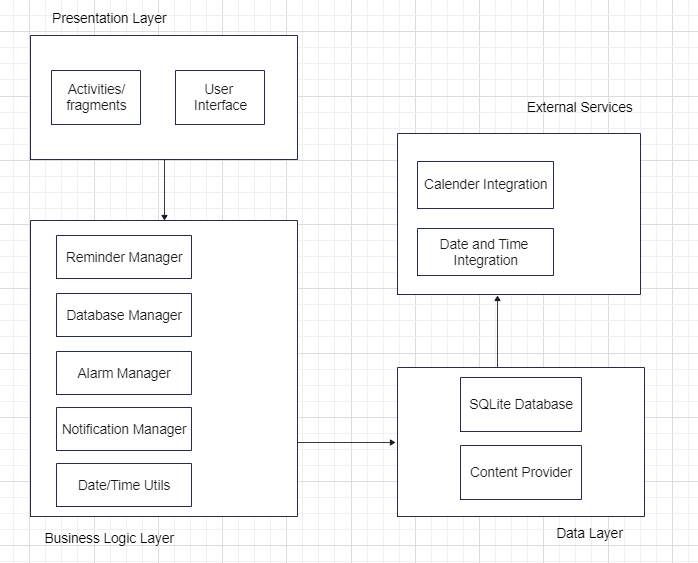
****

Fig 3.1: Architecture Diagram

3.2 USER DIAGRAM:

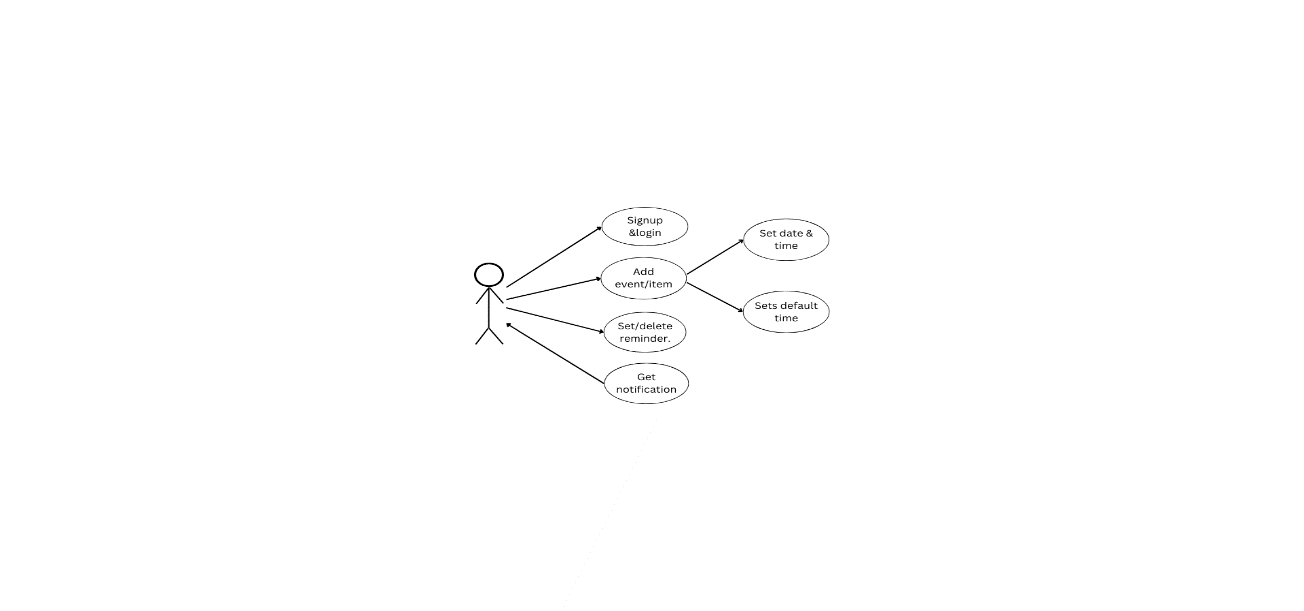
****

Fig 3.2: User Diagram

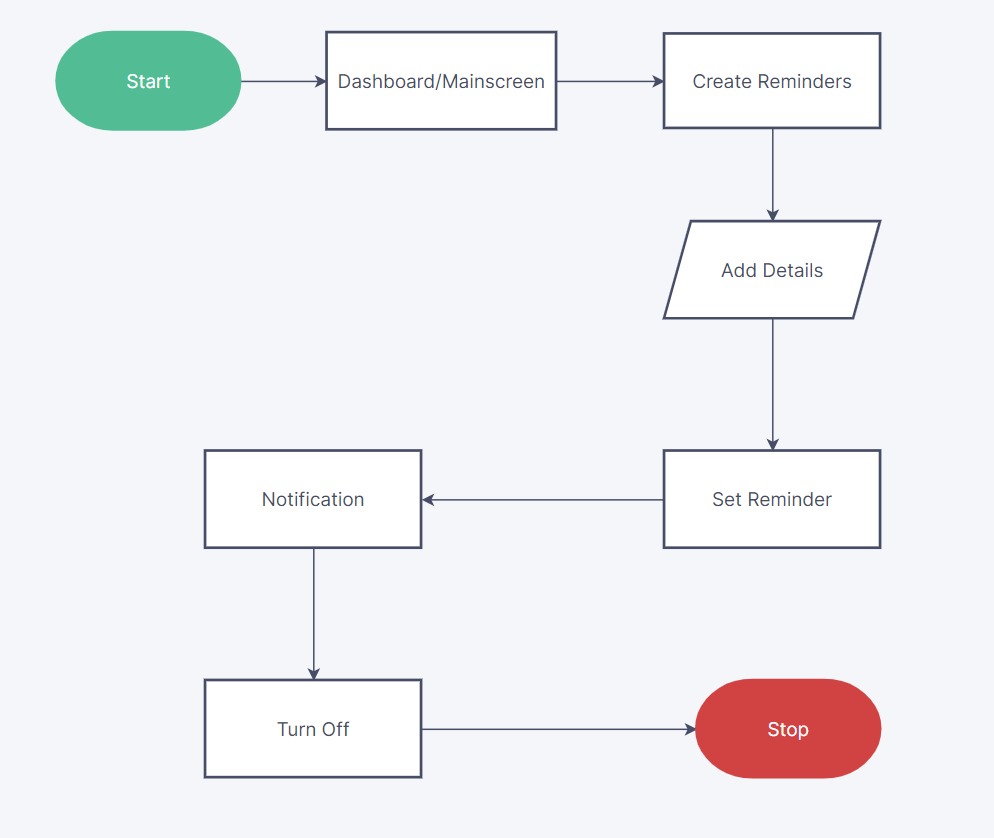
3.3 FLOW CHART:

Fig 3.3: Flow Chart

3.4 SEQUENCE DIAGRAM:

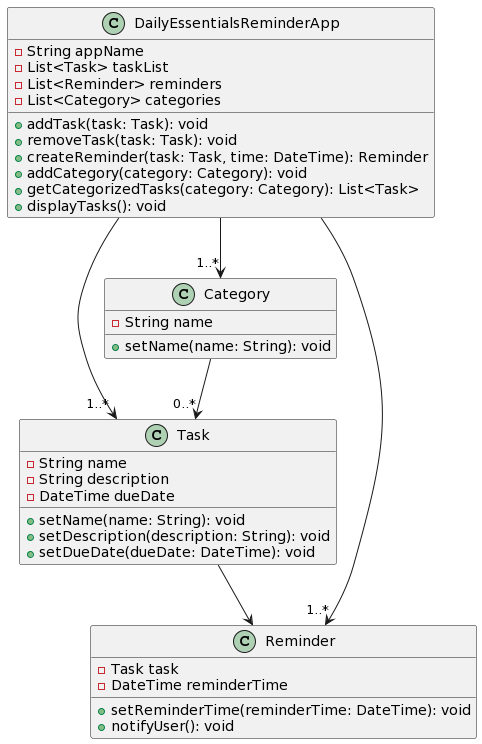


Fig 3.4: Sequence Diagram

3.5 ACTIVITY DIAGRAM:

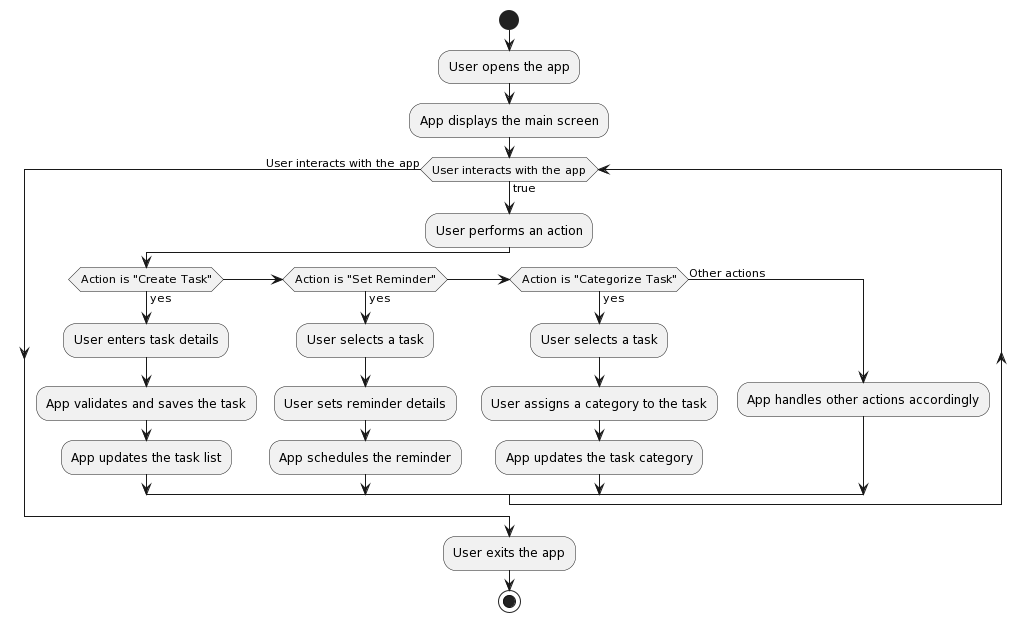


Fig 3.5: Activity Diagram

**Chapter 4**

**IMPLEMENTATIONTop of Form**

**4.1 Pseudocode for Main Activity:**

class MainActivity:

R.id.create reminder)

mRecyclerview ← findViewById(R.id.recyclerView)

dataholder ← ArrayList¡Model¿()

adapter ← myAdapter(dataholder)

// Set click listener for FloatingActionButton

mCreateRem.setOnClickListener(onCreateReminderClick)

// Set up RecyclerView

layoutManager ← LinearLayoutMan-

ager(getApplicationContext())

mRecyclerview.setLayoutManager(layoutManager)

mRecyclerview.setAdapter(adapter)

// Load data from the database

cursor ← dbManager(getApplicationContext()).readallreminders()

while cursor.moveToNext():

// Create a Model object with the retrieved data and add it

to the dataholder list

model ← Model(cursor.getString(1), cursor.getString(2), cur-

sor.getString(3))

dataholder.add(model)

adapter.notifyDataSetChanged() // Update the RecyclerView

end function

function onBackPressed():

finish() // Close the app

super.onBackPressed()

function onCreateReminderClick(view):

intent ← Intent(getApplicationContext(), ReminderActiv-

ity.class)

startActivity(intent) // Start ReminderActivity to add re-

minders

end class

**4.2 Pseudocode for Login Activity:**

**Input**: User details: name, email, password

**Result**: Registration status: success or failure

function RegisterUser(name, email, password);

begin

**if** IsEmailValid(email) and IsPasswordValid(password) then

**if** IsEmailAvailable(email) then

User newUser = CreateUser(name, email, password);

// Save user details in the database

**if** SaveUser(newUser) then

**return** ”Registration successful”;

**end**

**else**

**return** ”Registration failed”;

**end**

**end**

**else**

**return** ”Email already registered”;

**end**

**end**

**else**

**return** ”Invalid email or password”;

**end**

**end**

**function** IsEmailValid(email);

**begin**

// Check email validation criteria

**if** email matches validation rules then

**return** true;

**end**

**else**

**return** false;

**end**

**end**

**function** IsPasswordValid(password);

**begin**

// Check password validation criteria

**if** password matches validation rules **then**

**return** true;

**end**

**else**

**return** false;

**end**

**end**

**function** IsEmailAvailable(email);

**begin**

// Check if email is already registered

**if** email is not in database then

**return** true;

**end**

**else**

**return** false;

**end**

**end**

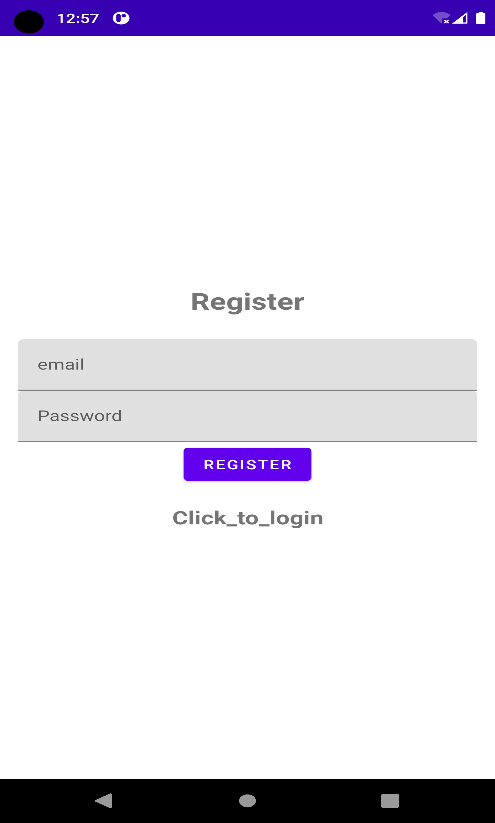
**Chapter 5**

**TESTING**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case** | **Test Description** | **Input** | **Expected Output** | **Actual Output** | **Status** |
| 1 | Valid email and password | email: “hithaumesh@gmail.com”, password: "Hitha@123" | Successful login | Successful login | Pass |
| 2 | Incorrect email format | email: “test”, password: “123” | Error: Invalid email format | Error:Invalid email for format | Pass |
| 3 | Empty email field | email: “ “, password: “123” | Error: Email field cannot be empty | Error: Email field cannot be empty | Pass |
| 4 | Empty password field | email: "[test@example.com](mailto:test@example.com)", password: " " | Error: Password field cannot be empty | Error: Password field cannot be empty | Pass |
| 5 | Incorrect email and password | Email: “hithaumesh@gmail.com”,  Password: "456" | Error: Incorrect email or password | Error: Incorrect email or password | Pass |
| 6 | Email and password in uppercase | Email: “[HITHA@gmail.com](mailto:HITHA@gmail.com)”  Password: “hitha123” | Successful login | Successful login | Pass |
| 7 | Email and password with mixed case | Add the item here : mad lab exam  date : 15-07-2023  time : | Pop up message :  Please enter the date and time | Pop up message :  Please enter the date and time | Pass |
| 8 | Date provided is of past | Add the item here : 1st sem exam  date : 10-03-2021  time : 2:00 | Please enter a valid date | Please enter a valid date | Pass |
| 9 | Date of the item Reminder is not provided | Add the item here : mad lab exam  date :  time : 9:00 | Pop up message :  Please enter the date and time | Pop up message :  Please enter the date and time | Pass |
| 10 | Time for the item Reminder is not provided | Add the item here : mad lab exam  date : 15-07-2023  time : | Pop up message :  Please enter the date and time | Pop up message :  Please enter the date and time | Pass |

**Chapter 6**

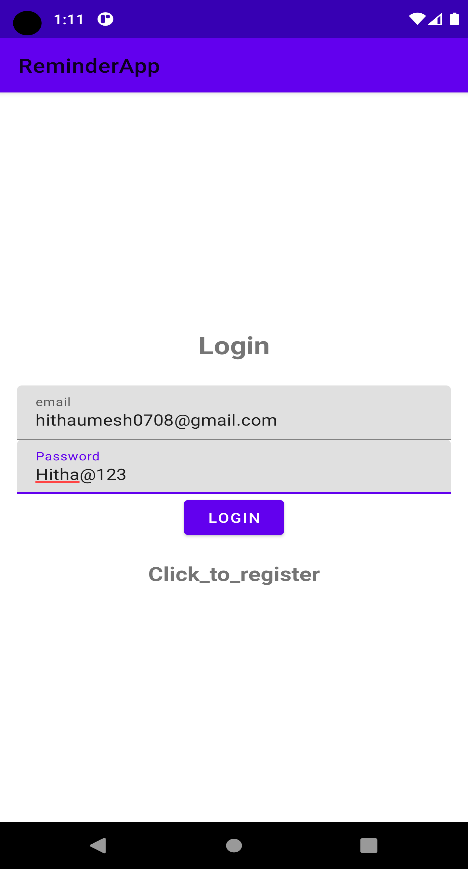
**RESULTS AND DISCUSSION**



**Fig:6.1 Splash Activity Fig:6.2 Register Page**

**Splash Activity Page** is the first screen that appears when the app is launched. It usually contains the app logo and name and is used to display any necessary information while the app loads. It can also be used to show a progress bar or animation while the app loads.

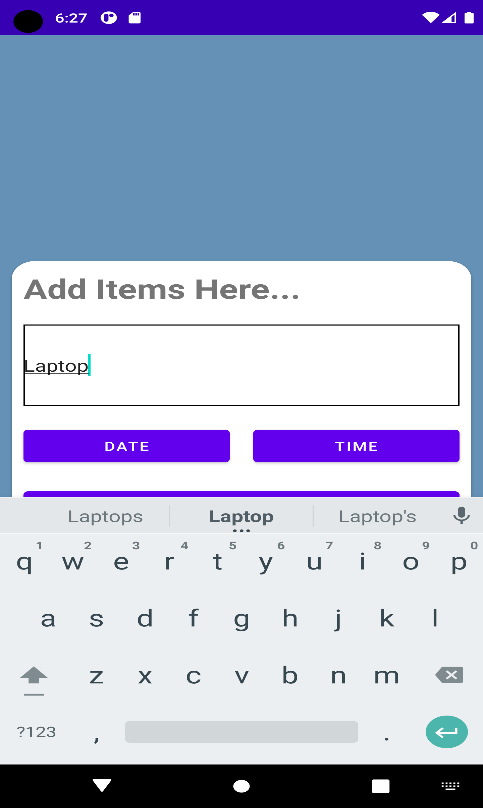
**Register Page** allows users to create an account by entering their details such as name, email address, password, etc. Once registered, users can log in using their email address and password. The register page should have validation checks for the input fields to ensure that the user enters valid data.



**Fig:6.3 Login Page Fig:6.4 Add Reminder**

**Login Page** allows users who have already registered to log in by entering their email address and password. The login page should also have validation checks for the input fields to ensure that the user enters valid data.

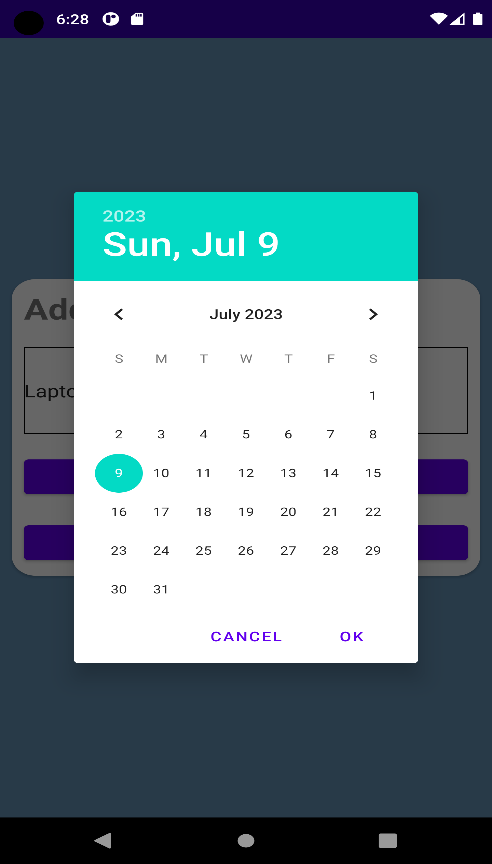
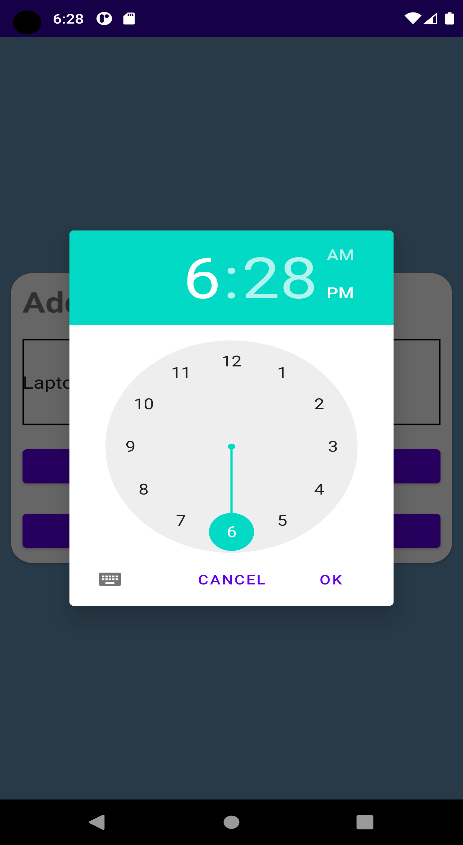
**Add Reminder Page** allows users to add a new reminder by entering the reminder's name, date, time, and other details. Users can also set the frequency of the reminder and choose whether to receive notifications. The add reminder page should have a user-friendly interface with clear labels for each input field.



**Fig:6.5 Add Items Fig:6.6 Entering Items**

**Add Items Page** allows users to add items that they want to be reminded of. They can enter the item's name, quantity, and other details. The add items page should also have a user-friendly interface with clear labels for each input field.

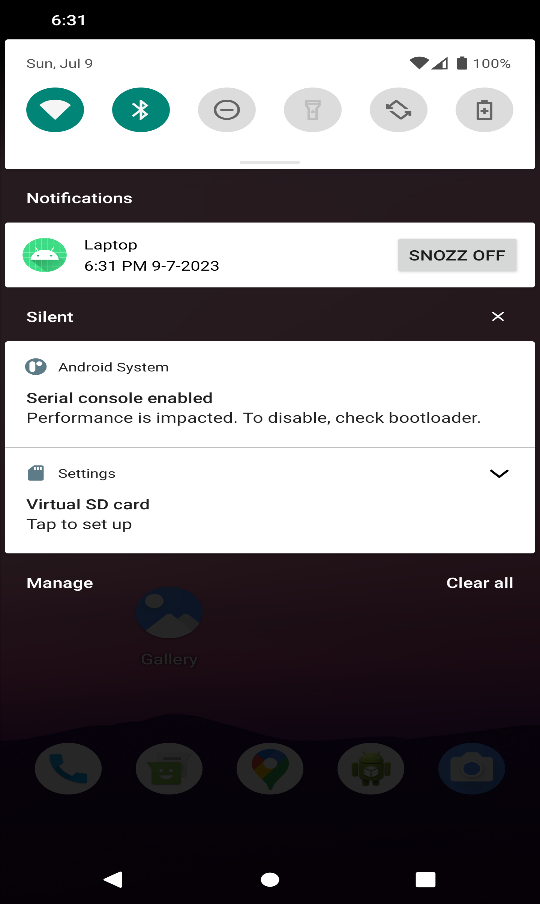
**Entering Items Page** allows users to enter items that they want to be reminded of. They can enter the item's name, quantity, and other details. The entering items page should also have a user-friendly interface with clear labels for each input field.

**Fig: 6.7 Set Date Fig:6.8 Set Time**

**Set Date Page** allows users to set the date for a reminder or an item. They can choose the date from a calendar or enter it manually. The set date page should have a user-friendly interface with clear labels for each input field.

**Set Time Page** allows users to set the time for a reminder or an item. They can choose the time from a clock or enter it manually. The set time page should have a user-friendly interface with clear labels for each input field.

**Fig:6.9 Reminder added Fig:6.10 Alarm Notification**

**Reminder added page** that appears when a user adds a reminder for an essential item. The page displays the details of the reminder such as the name of the item, the date and time of the Reminder Alarm

**Notification Page** appears when an alarm goes off and displays the details of the reminder or item that triggered it. Users can choose to snooze or dismiss the alarm.

**Chapter 7**

**CONCLUSION AND FUTURE ENHANCEMENTS**

In conclusion, the Daily Essentials Reminder App is a user-friendly and innovative solution for managing daily routines and ensuring essential tasks and items are not forgotten. The app simplifies the process of organizing and prioritizing tasks, leading to a more efficient and stress-free daily routine. By offering features such as task scheduling, reminders, and task categorization, the app empowers users to stay on top of their responsibilities and essential needs.

Future Enhancements:

User Accounts and Cloud Sync: Implementing user accounts would allow users to access their reminders and task lists across multiple devices. Cloud synchronization would ensure that changes made on one device are reflected on others, providing a seamless experience.

Location-Based Reminders: Enhancing the app with location-based reminders would enable users to receive reminders when they are in a specific location, such as a grocery store. This feature would assist users in remembering to purchase essential items while they are in the appropriate vicinity.

Integration with Smart Home Devices: Integrating the app with smart home devices, such as voice assistants or smart displays, would provide users with hands-free access to their reminders. Users could simply voice their reminders or have them displayed on a smart screen.

Task Prioritization and Sorting: Adding the ability to prioritize tasks and sort them based on due dates or importance would help users focus on the most critical activities. This feature would improve task management and aid in better time allocation.

Integration with External Calendars: Enabling integration with popular external calendars, such as Google Calendar or Outlook, would allow users to sync their reminders with their existing calendars. This integration would ensure that users have a comprehensive view of their schedules and commitments.

**Chapter 8**

**REFERENCE**

1. developer.android.com
2. developer.android.com/jetpack
3. firebase.google.com/docs
4. https://dev.to/blazebrain/building-a-reminder-app-with-local-notifications-using-workmanager-api-385fhttps://dev.to/blazebrain/building-a-reminder-app-with-local-notifications-using-workmanager-api-385f
5. developer.android.com/reference/androidx/recyclerview/widget/RecyclerView
6. developer.android.com/training/data-storage/sqlitematerial.io/design
7. developer.android.com/guide/topics/ui/notifiers/notifications
8. Object-oriented analysis and design with applications
9. https://electro4u.net/blog/create-and-manage-tasks-easily-with-android-daily-essential-item-reminder-app
10. https://github.com/Hithaukarkera/MAD-Project