WEEK 03

INTERNSHIP REPORT

FA23-BSE-073(MUHAMMAD MUTEE ULLAH)
CODELOUNGE
Wapda Town Lahore

Contents

1. Introduction

The Flutter Todo Application is a cross-platform mobile application developed using the **Flutter** framework and the **Dart** programming language. The primary objective of this project was to design and implement a modern, scalable, and user-friendly task management system that enables users to efficiently organize and track their daily activities.

The project follows the **Model-View-Controller (MVC)** architectural pattern and utilizes the **GetX** package for state management, dependency injection, and routing. This approach ensures a reactive, lightweight, and maintainable codebase while minimizing boilerplate and improving overall performance.

The following demonstrates the GetX implementation learned during the previous week.

2. Duration

The development lifecycle of the application was structured into three well-defined phases over four weeks:

Phase 1 — Planning & Environment Setup (Day 1)

- Project initialization and Flutter environment configuration
- Dependency selection and architecture planning
- UI/UX wireframing for all major screens

Phase 2 — Core Development (Day 2–3-4)

- Implementation of authentication flows
- Development of full CRUD functionality for tasks
- Integration of GetX controllers for state management
- Notification scheduling and calendar functionality

Phase 3 — Testing & Refinement (Day 5)

- UI polishing and consistency improvements
- Debugging and edge-case handling

- Navigation flow refinement
- Final feature validation and performance optimization

3. Overview

The Flutter Todo Application is a feature-rich task management solution targeting both Android and iOS platforms. Users can create and manage tasks enriched with detailed metadata, including:

- Title
- Description
- Priority level (Low, Medium, High)
- Due date
- Tags
- Completion status

The project maintains a clean and modular folder structure, separating responsibilities into dedicated directories such as:

- models/ — Data representations
- controllers/ — Business logic and state management
- screens/ — UI pages
- widgets/ — Reusable UI components

The **GetX** package handles reactive state updates and route management, eliminating dependency on traditional setState() and BuildContext-based navigation. Controllers are registered at application startup using Get.put() to ensure centralized and consistent state handling.

Key screens include:

- Splash Screen
- Login & Registration Screens
- Home Screen with tabbed task lists

- Add Todo Screen
- Calendar Screen
- Notification Screen
- Account Settings Screen with Change Password functionality

Together, these components deliver a complete end-to-end user experience.

4. Tasks Performed

Todo Management

- Designed the Todo model with structured fields:
  id, title, description, createdAt, dueDate, isCompleted, priority, and tags
- Implemented full CRUD operations within TodoController
- Applied reactive state management using Rx variables
- Enabled local persistence through toMap() and fromMap() serialization methods

Notifications & Calendar Integration

- Built a NotificationController for scheduling reminders
- Integrated due-date-based local notifications
- Developed a CalendarScreen for date-oriented task visualization
- Implemented a NotificationScreen for listing scheduled alerts

5. Learning Outcomes

This project contributed significantly to technical growth and practical expertise in Flutter development:

GetX State Management

- Gained hands-on experience with reactive programming using Rx variables and Obx widgets

- Learned dependency injection using Get.put() and Get.find()
- Reduced boilerplate compared to traditional state management solutions such as Provider or Bloc

Application Architecture

- Strengthened understanding of the MVC design pattern
- Improved separation of concerns between models, controllers, and UI layers
- Enhanced code maintainability and scalability

Data Modeling & Serialization

- Designed robust Dart model classes with fromMap() and toMap() methods
- Applied Dart enums (e.g., Priority) for structured categorical data
- Handled null-safety and data validation effectively

Navigation & Lifecycle Management

- Implemented named route navigation using GetX
- Managed authentication state transitions efficiently
- Ensured clean navigation stack handling across login and logout states

6. Challenges and Solutions

Throughout development, several technical challenges were encountered and resolved:

State Synchronization Across Screens

Maintaining a consistent todo list across multiple screens initially caused redundant rebuilds and state inconsistencies.

Solution:

Controllers were registered as permanent singletons using Get.put(..., permanent: true) in main(), ensuring a unified state instance across the entire application lifecycle.

Enum Deserialization Issues

Deserializing the Priority enum from stored string values resulted in runtime errors when encountering invalid or unexpected data.

Solution:

Implemented safe fallback logic using:

Priority.values.firstWhere(..., orElse: () => Priority.low)

This ensured stability even when corrupted or incomplete data was encountered.

Notification Scheduling Accuracy

Handling nullable dueDate fields and timezone differences caused notification timing inconsistencies.

Solution:

- Added strict null checks before scheduling
- Normalized DateTime values to UTC format
- Verified timezone alignment to prevent misfires

7. Conclusion

The Flutter Todo Application demonstrates the successful development of cross-platform mobile application from initial planning to final deployment.

By combining Flutter's flexible UI system with GetX's efficient state management and dependency injection, the project achieves a clean, scalable, and high-performance architecture.

The application delivers a comprehensive feature set including authentication, task prioritization, due-date tracking, local notifications, calendar visualization, and account management — making it a practical tool for real-world productivity.

Beyond technical implementation, this project significantly strengthened expertise in Flutter architecture, Dart modeling, reactive programming principles, and mobile UX design.

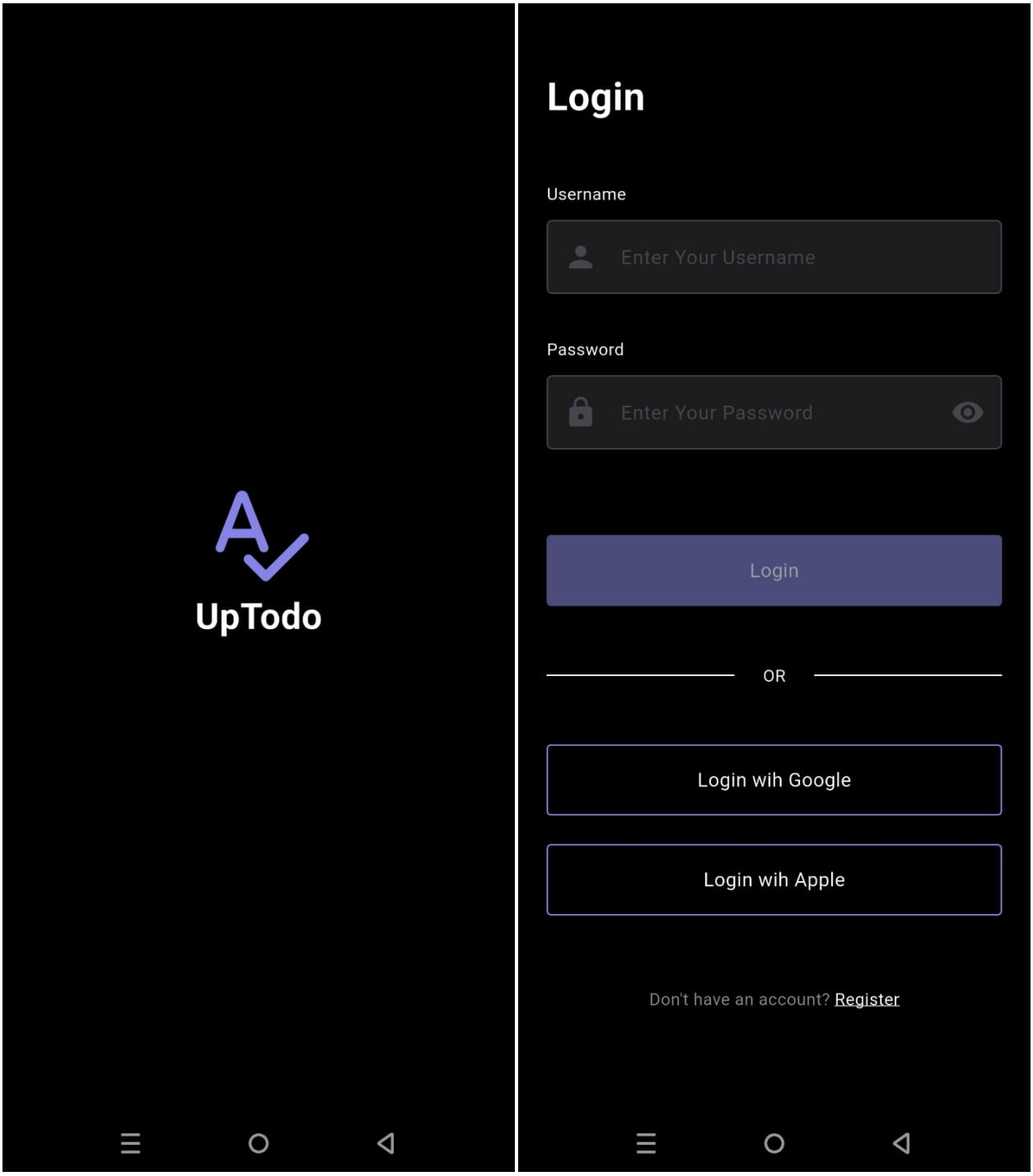
The application can be further extended in future iterations with features such as:

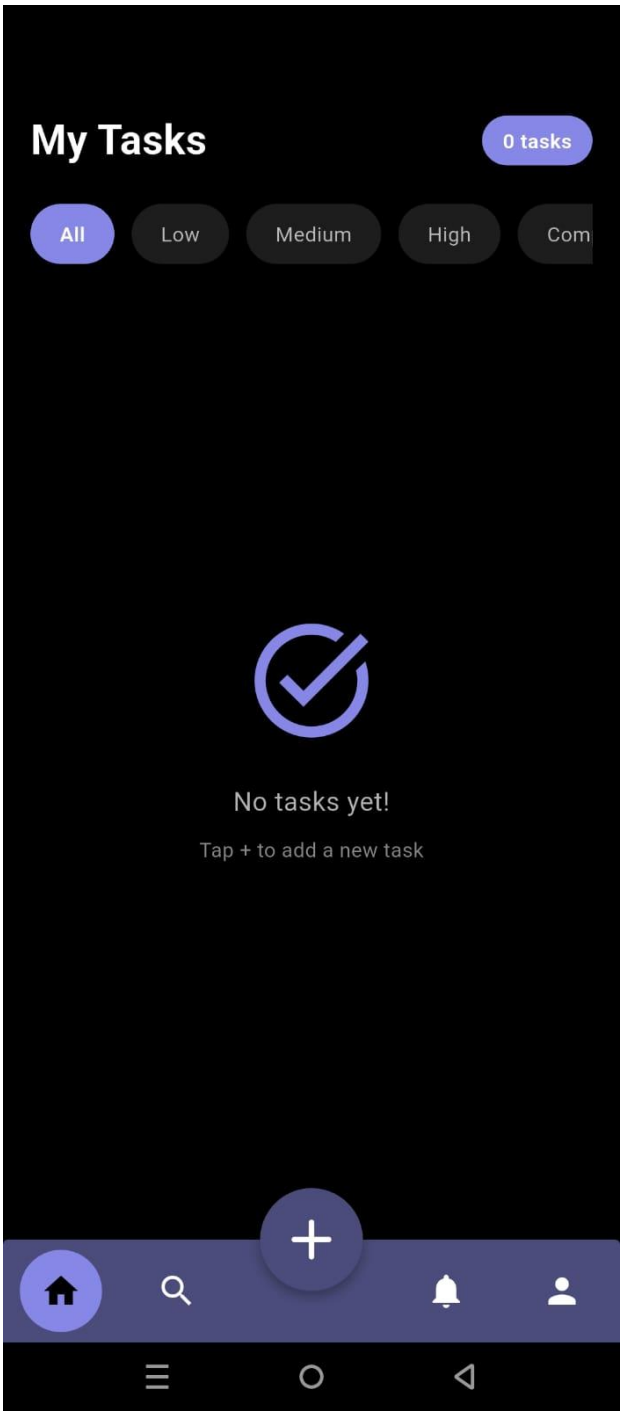
- Cloud synchronization (Firebase integration)
- Team collaboration and shared tasks
- Recurring task scheduling
- Home screen widgets
- Dark/light theme customization

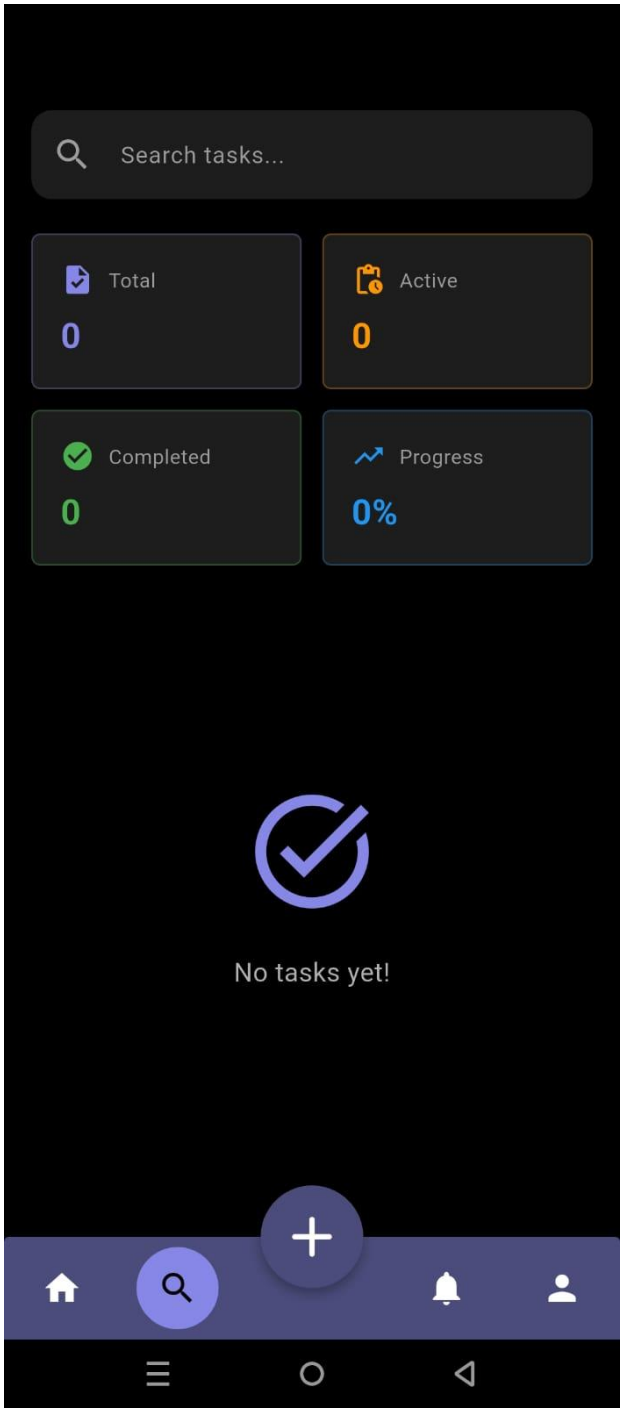
Overall, this project represents a strong foundation for scalable mobile application development and demonstrates readiness for professional Flutter development environments.

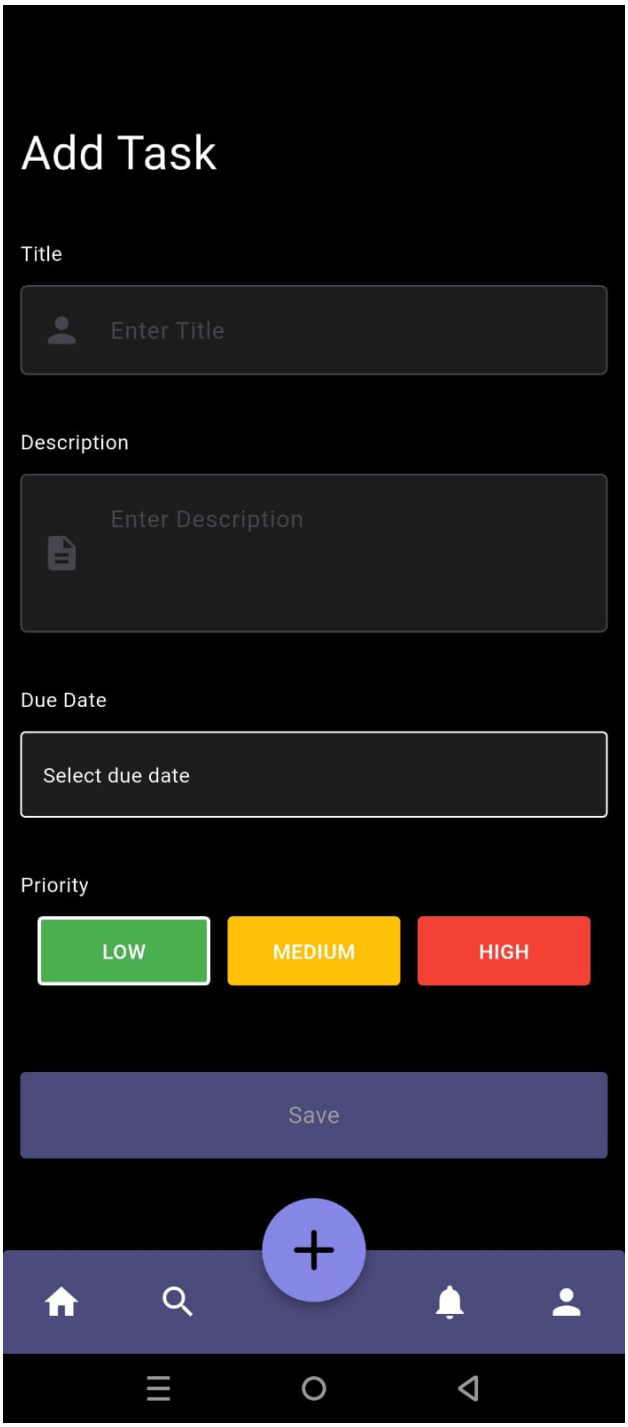
Screenshots:

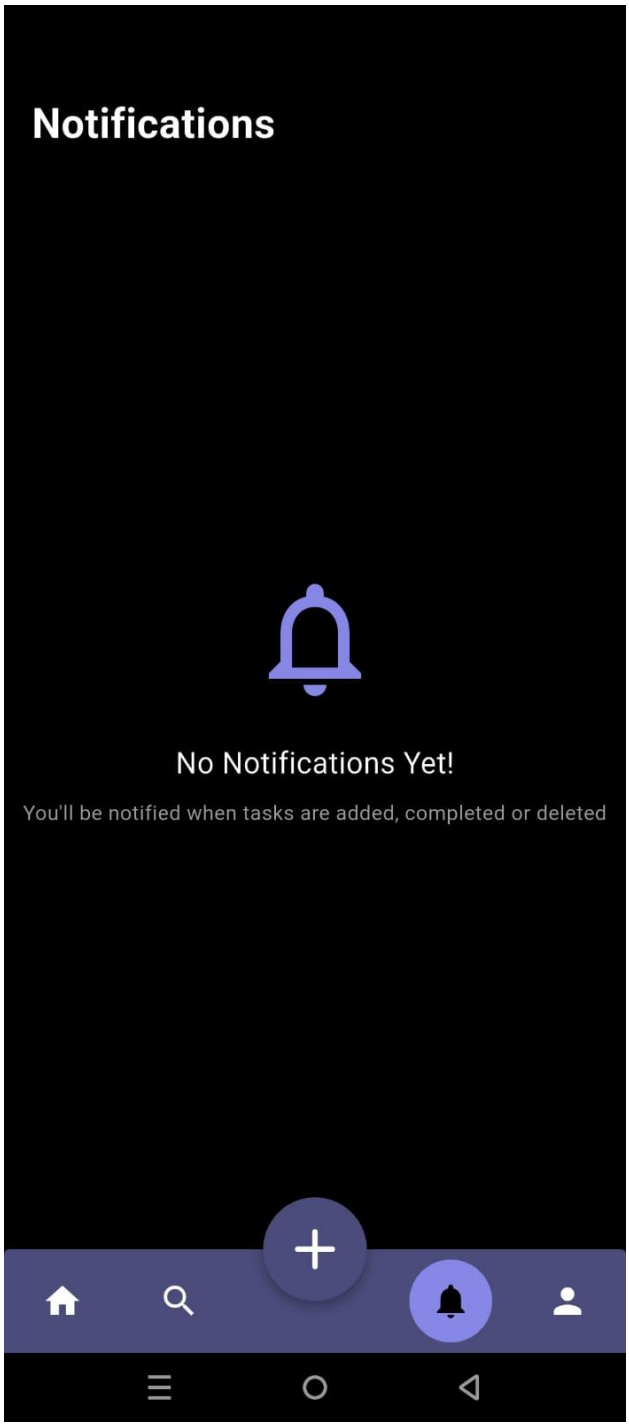
Some of the screenshots are here:

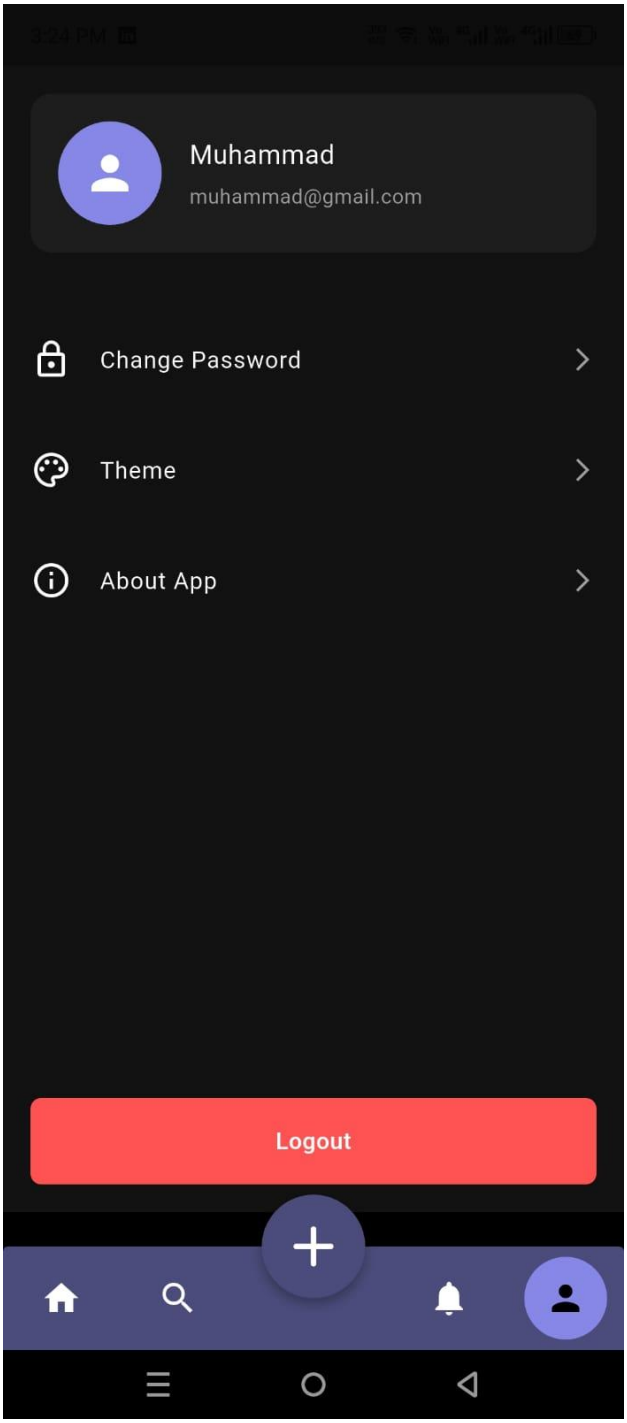


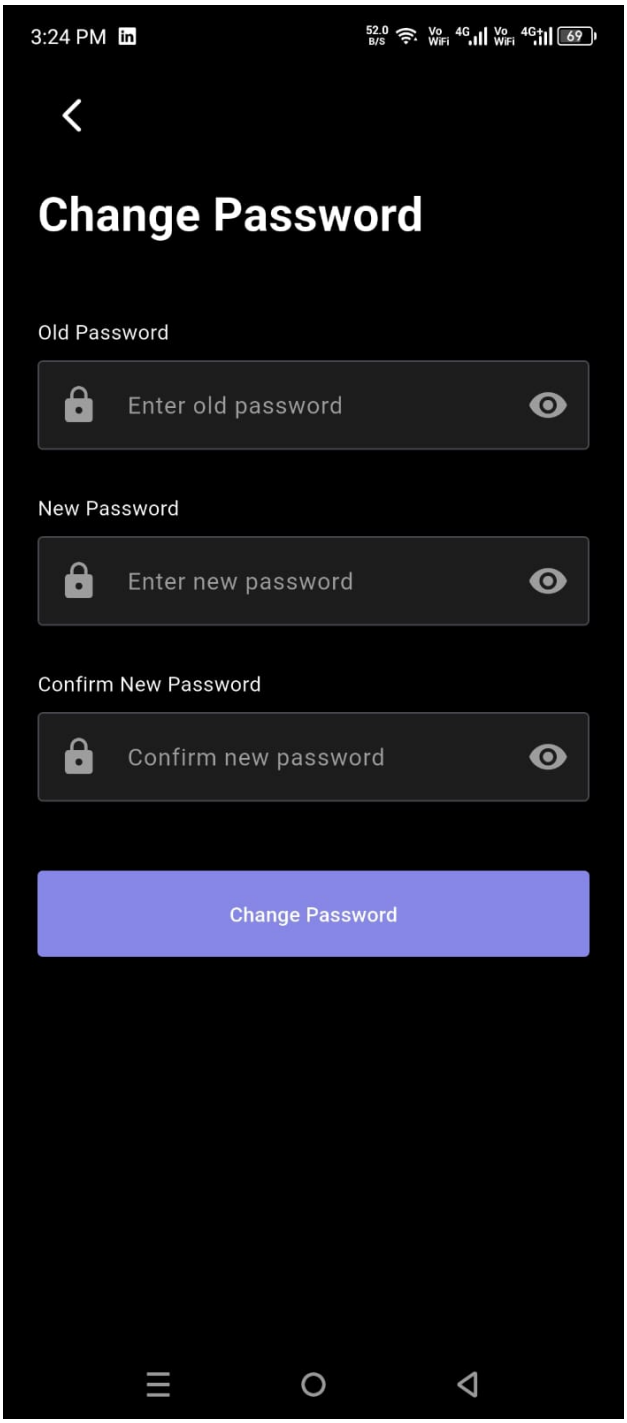

My Tasks
0 tasks
All
Low
Medium
High
Com
No tasks yet!
Tap + to add a new task


Search tasks...
Total
0
Active
0
Completed
0
Progress
0%
No tasks yet!


Add Task
Title
Enter Title
Description
Enter Description
Due Date
Select due date
Priority
LOW
MEDIUM
HIGH
Save


Notifications
No Notifications Yet!
You'll be notified when tasks are added, completed or deleted


Muhammad
muhammad@gmail.com
Change Password
Theme
About App
Logout


3:24 PM
Change Password
Old Password
Enter old password
New Password
Enter new password
Confirm New Password
Confirm new password
Change Password