**Event Management Platform**

**Mobile Application Development Project-Complex Engineering Problem**

By

**Ms. Mariam Memon**

**A logo of a university

AI-generated content may be incorrect.**

Assignment submitted to MUET for the degree of

**Bachelor of Engineering**

In

**Software Engineering**

Assignment submitted by

**22SW009**

**22SW163**

**October 2025**

Real World Problem Identification:

Problem: Event planning is often a chaotic process involving multiple tasks such as managing guest lists, tracking

tasks, setting reminders, and budgeting. The need for a centralized mobile solution to streamline these activities is

evident, especially for personal or small business events.

Proposed Solution:

The Event Planner App centralizes all event planning tasks into one mobile application, allowing users to

organize guest lists, track tasks, and manage budgets efficiently.

The Event Planner App is designed to solve event management problems by providing the following

features:

Animated Splash Screen:

The app launches with a visually engaging animated splash screen displaying the app logo and event

planning elements, enhancing the user experience with a professional and attractive entry point.

Login/Signup:

Allow user registration and login using Firebase Authentication.

Event Dashboard:

A main dashboard screen showcases a list of created events, each presented as a square-shaped card with

an event image, name, and basic details.

Users can add new events using a floating action button (FAB). When clicked, a dialog box appears,

prompting users to input details such as the event name, organizer name, and the option to upload an

image.

Event Details Screen:

Users access the Event Details Screen upon selecting an event from the dashboard. This screen includes:

Interactive buttons for navigating guests, task, and budget management screens.

A calendar icon button next to the event date, allowing users to select and update the event date via a

date picker.

A budget input field where users can specify the total budget for the event.

Guest Management:

The Guest Manager screen offers a search bar for finding and managing guests, with a floating action

button to add new guests.

Guests are displayed with attendance indicators, showing if they are confirmed or pending, for clear and

efficient management.

Task Management:

The To-Do List screen provides an organized view of tasks related to the event. It features a search bar for

task filtering and a FAB to add new tasks.

Tasks can be marked as completed, with the UI updating to display crossed-out text and color changes,

indicating their completion status.

Budget Management:

The Budget Screen displays the event's total budget and allows users to add transactions via a FAB.

A budget input feature enables users to enter the event budget in PKR, with the screen showing the

current budget status and a summary of all transactions.

Responsive User Interface:

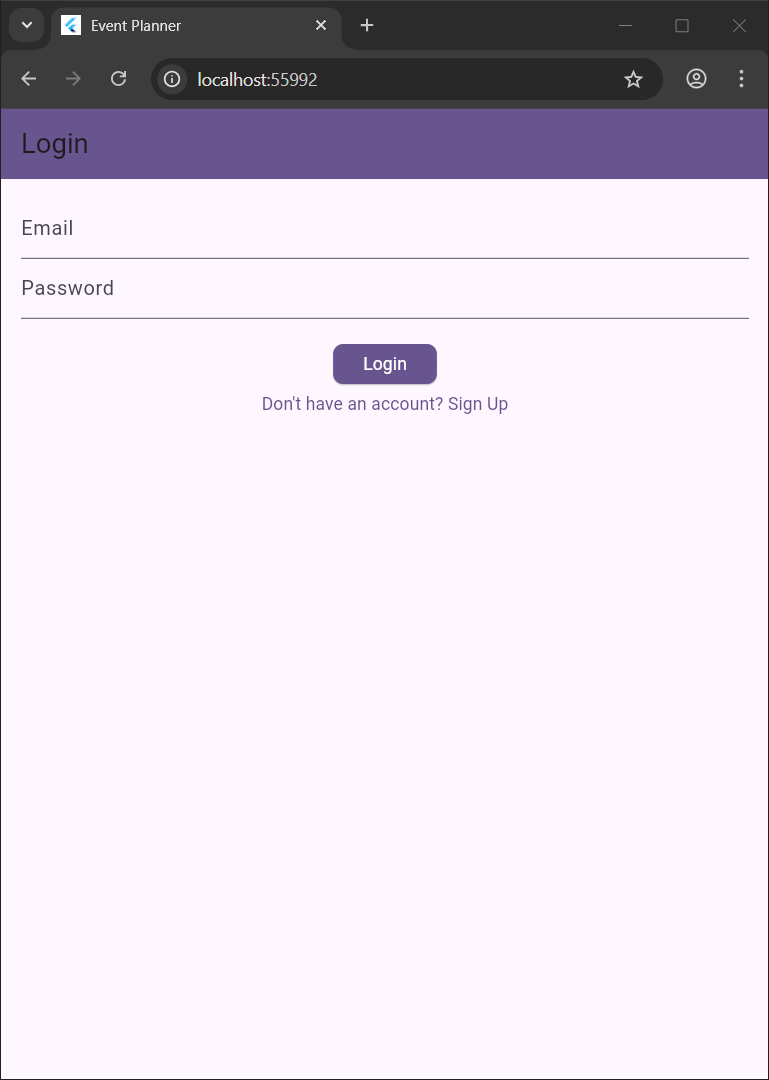
The app maintains a consistent color scheme focused on purple shades, ensuring a modern and cohesive

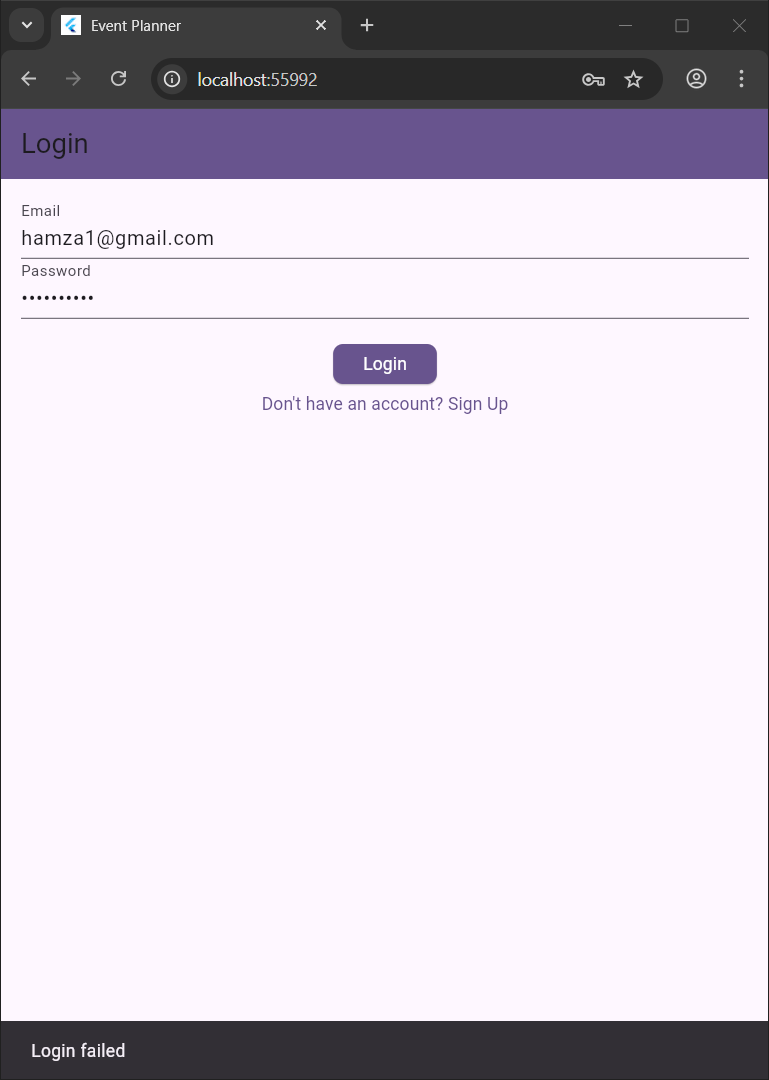
design.

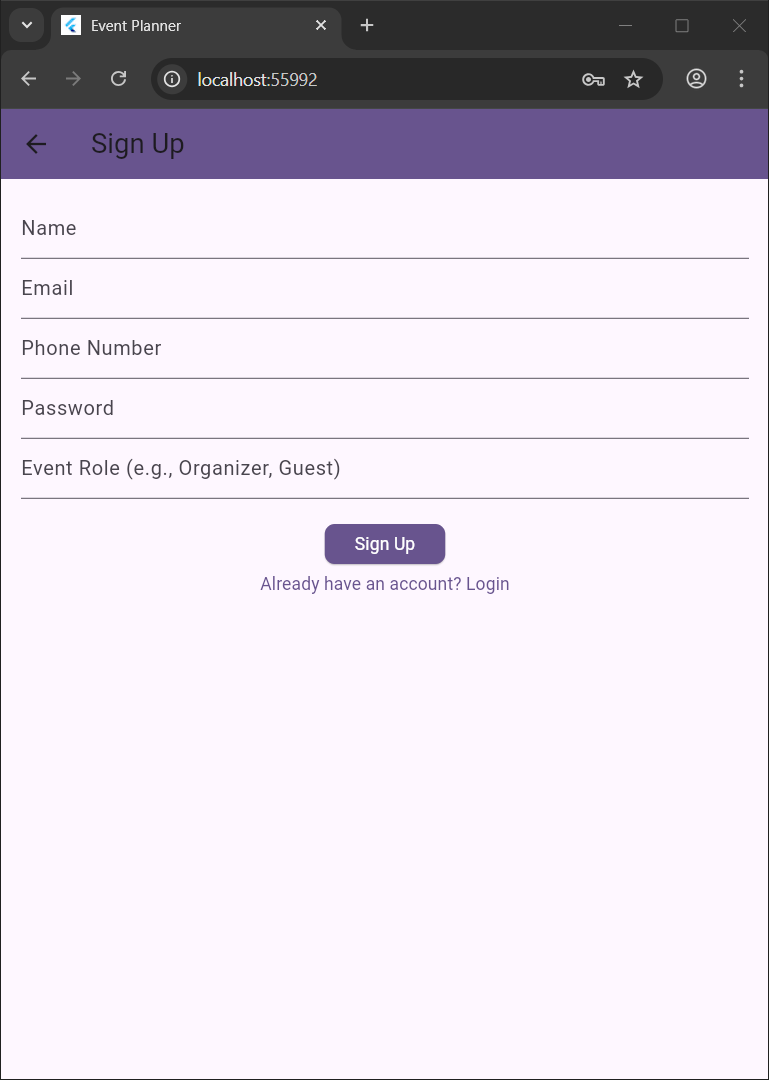
UI components like FABs, input fields, and event cards are responsive, delivering an intuitive user

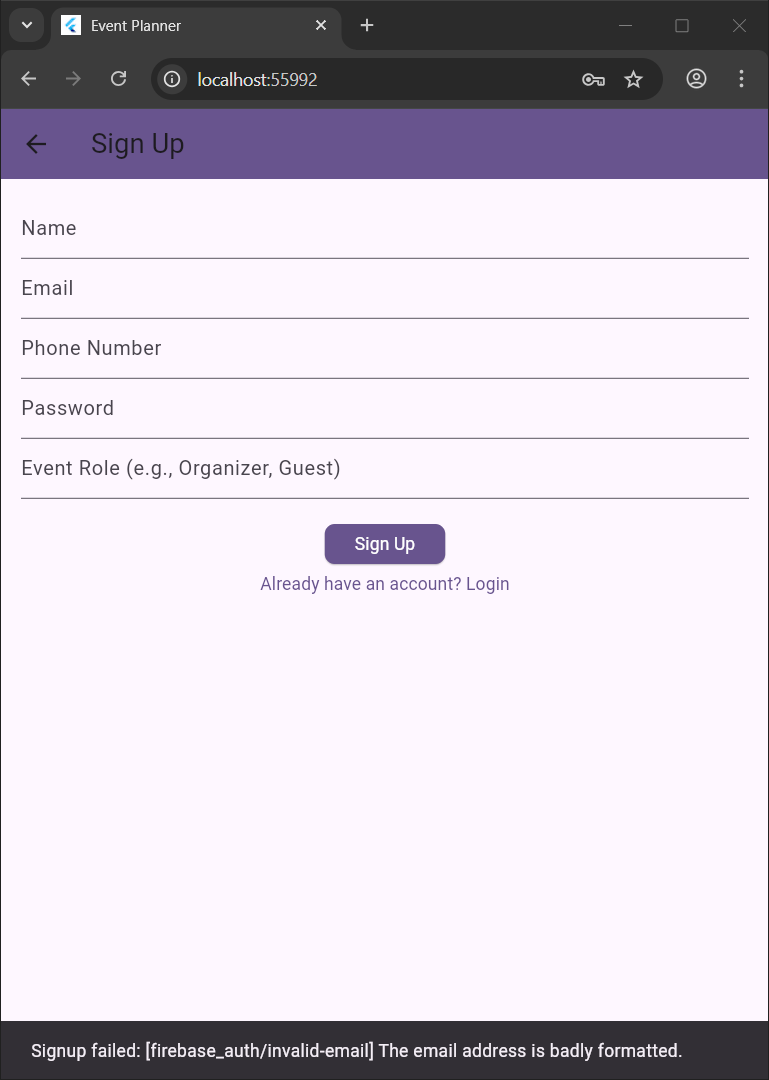
experience on various screen sizes and devices.

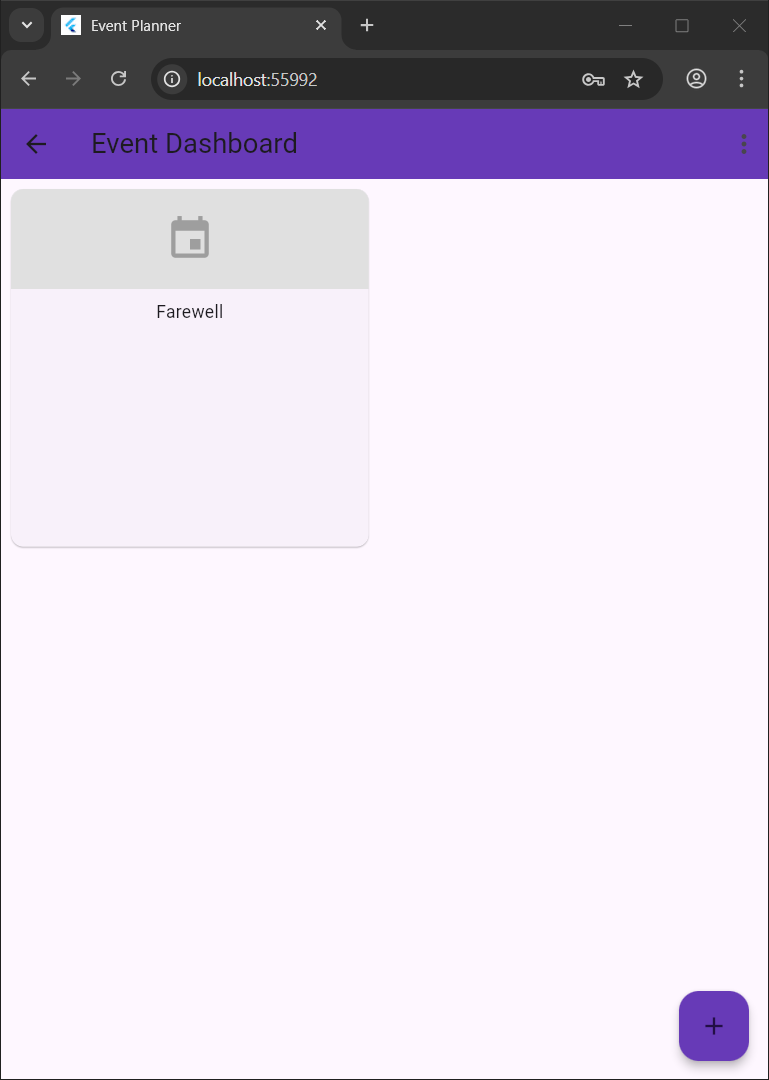
RESPONSIVE USER INTERFACE:

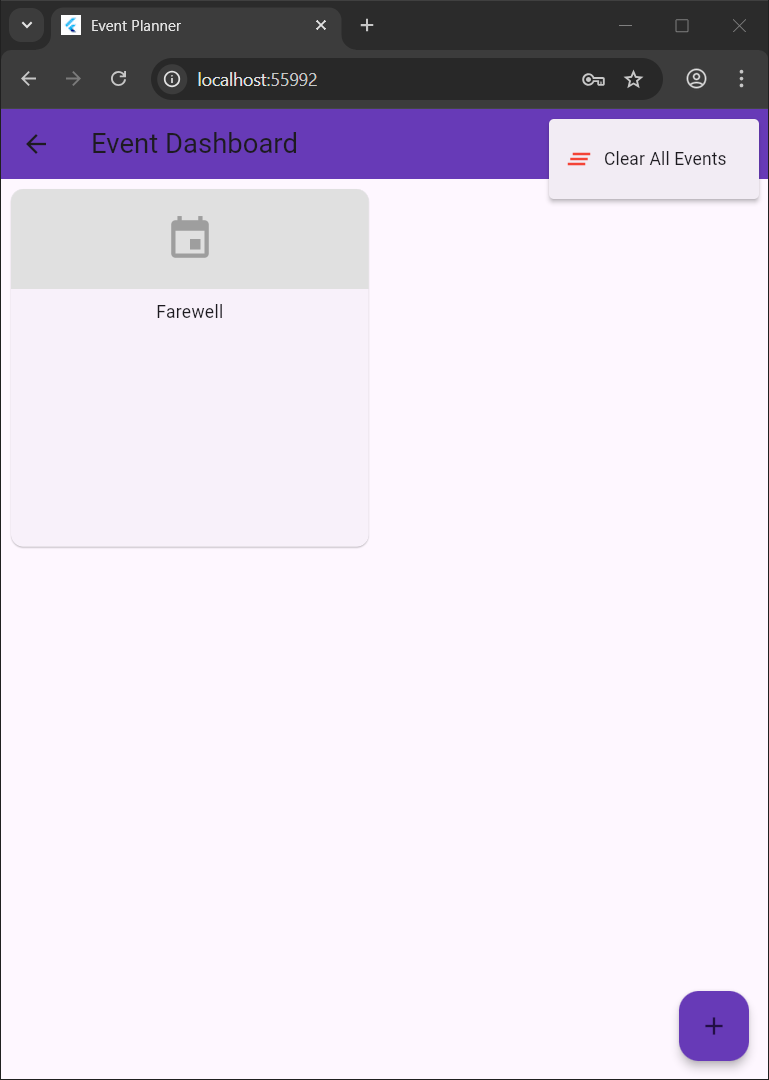


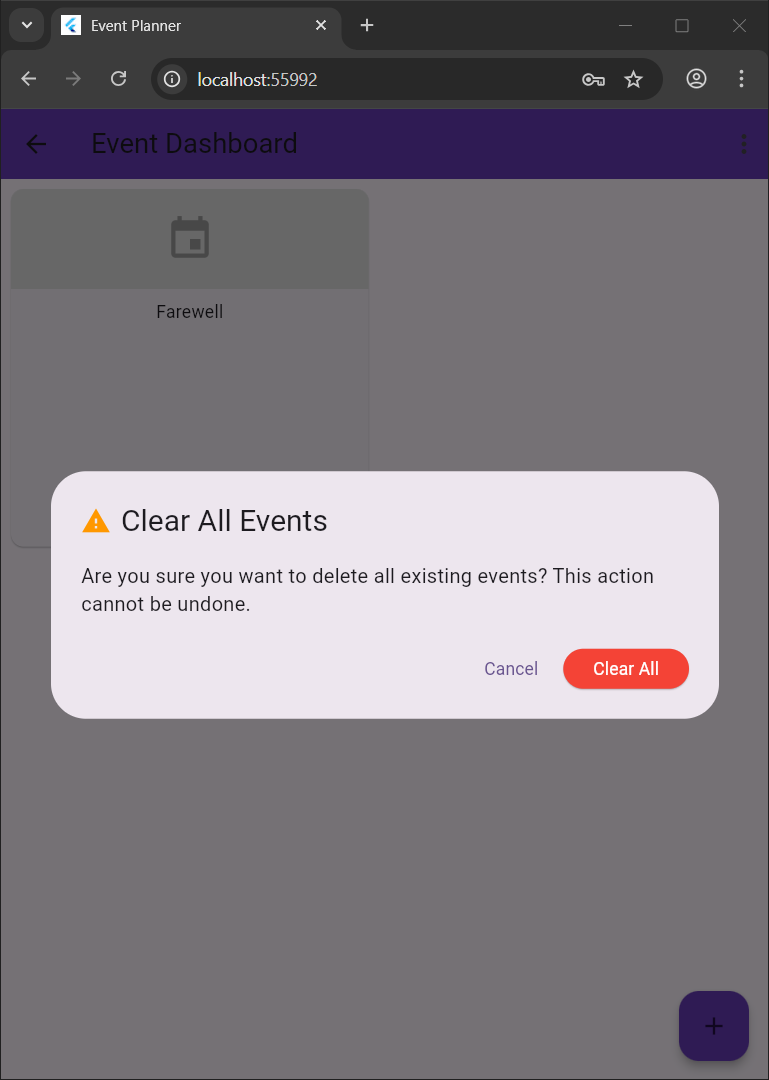


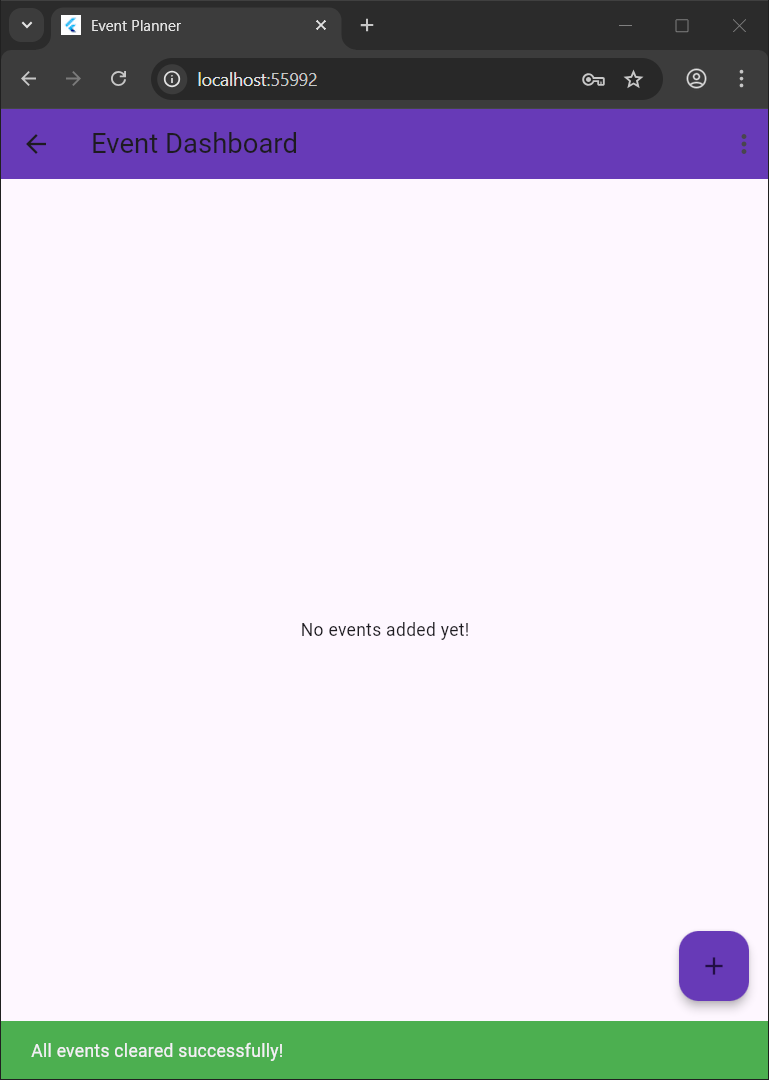


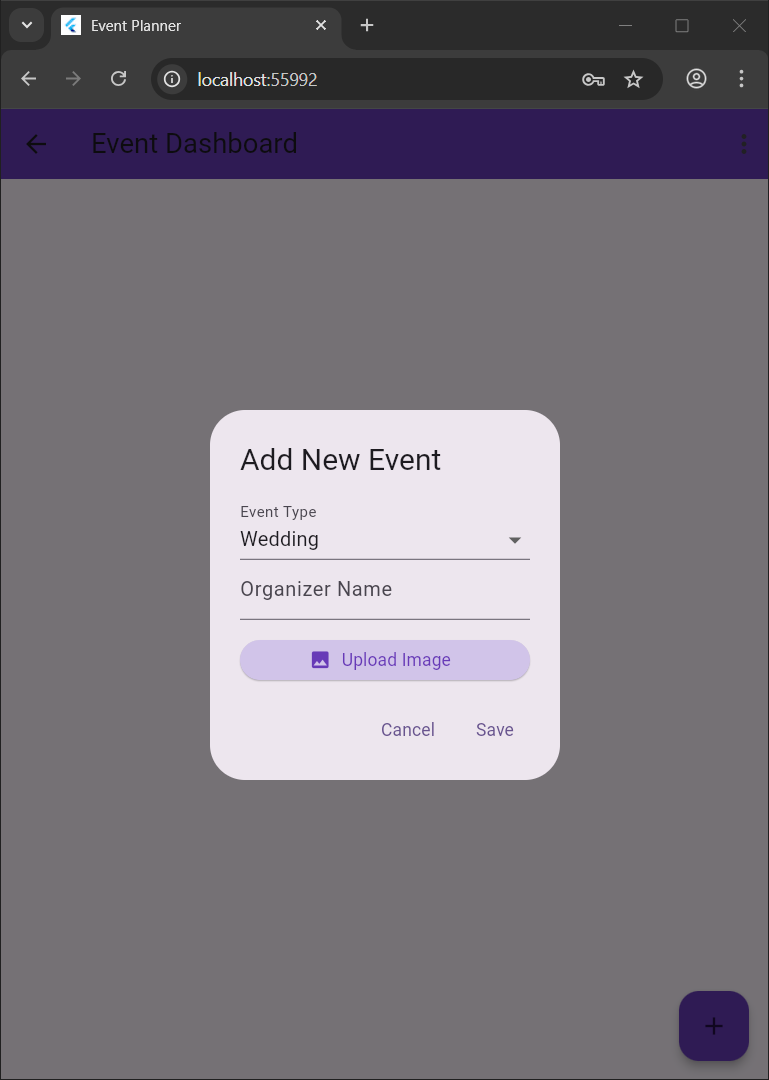


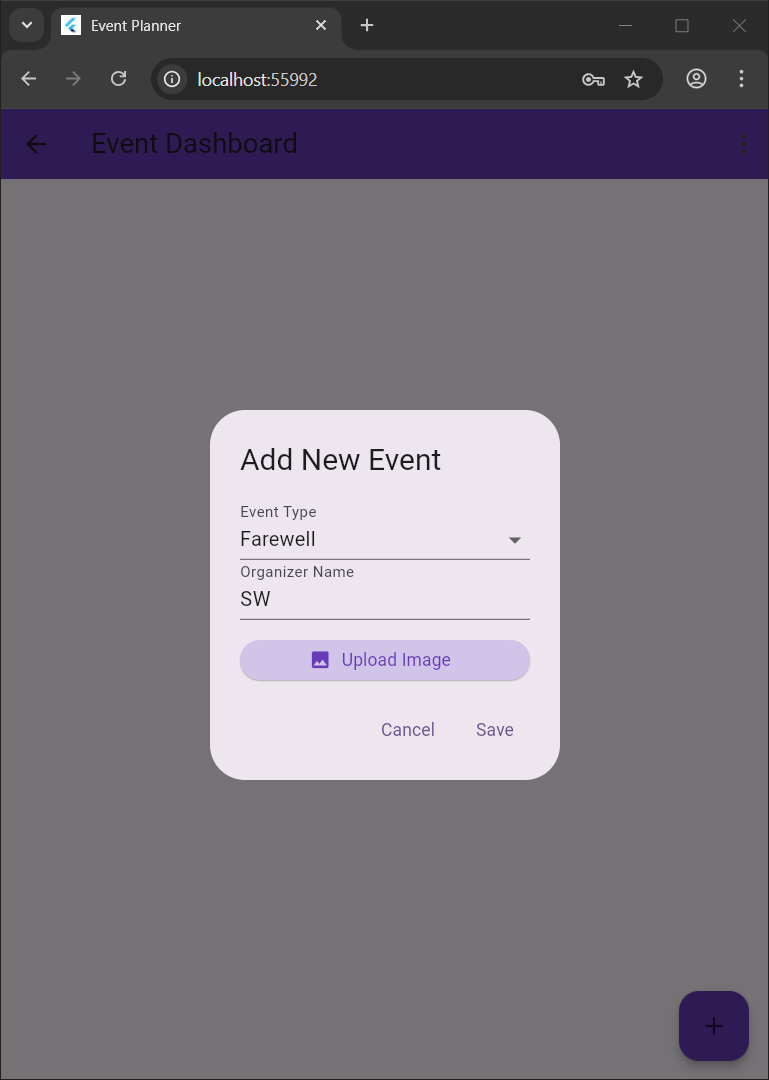


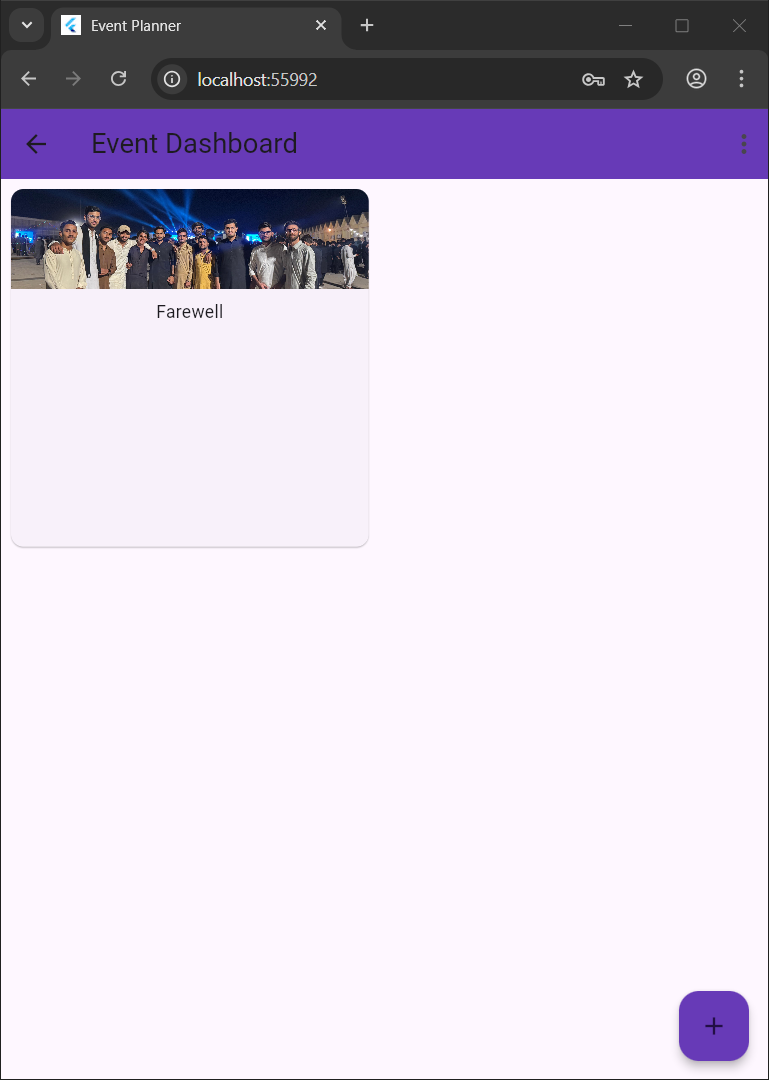


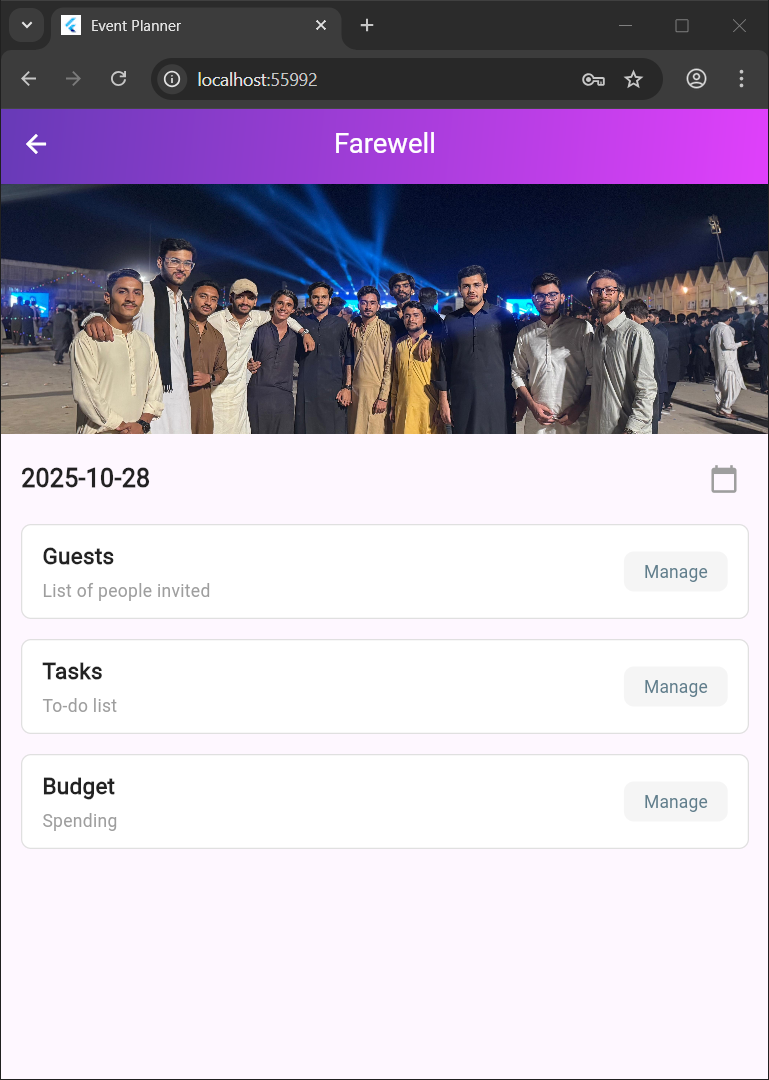


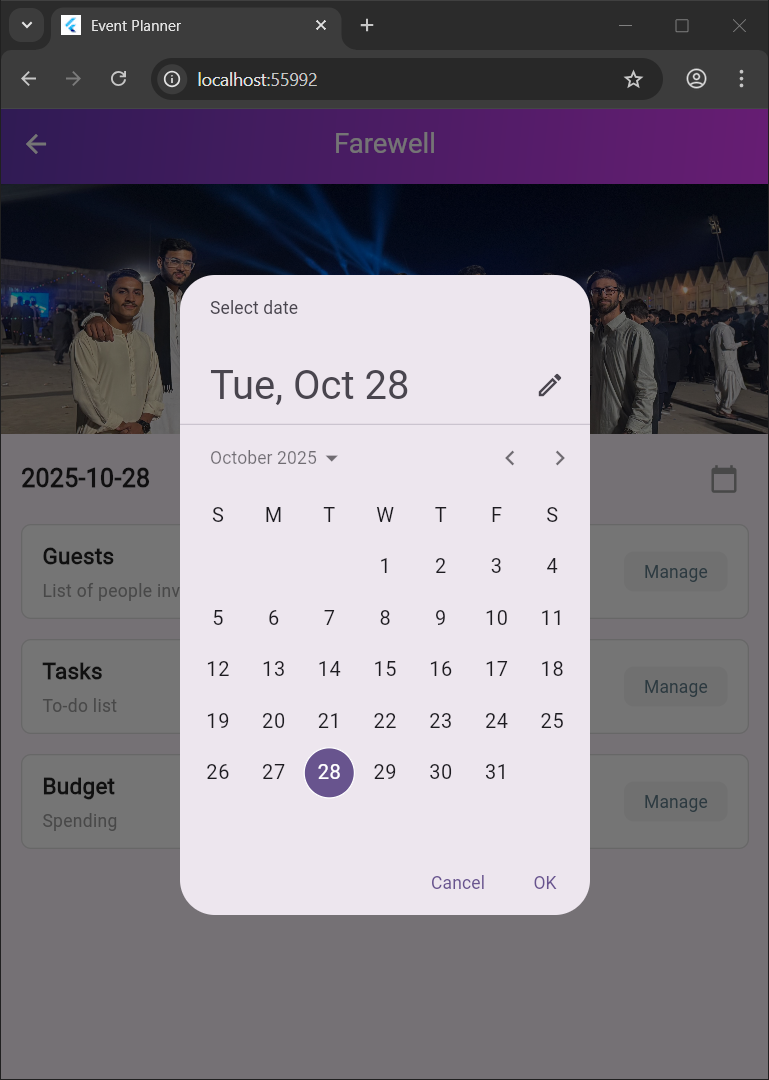


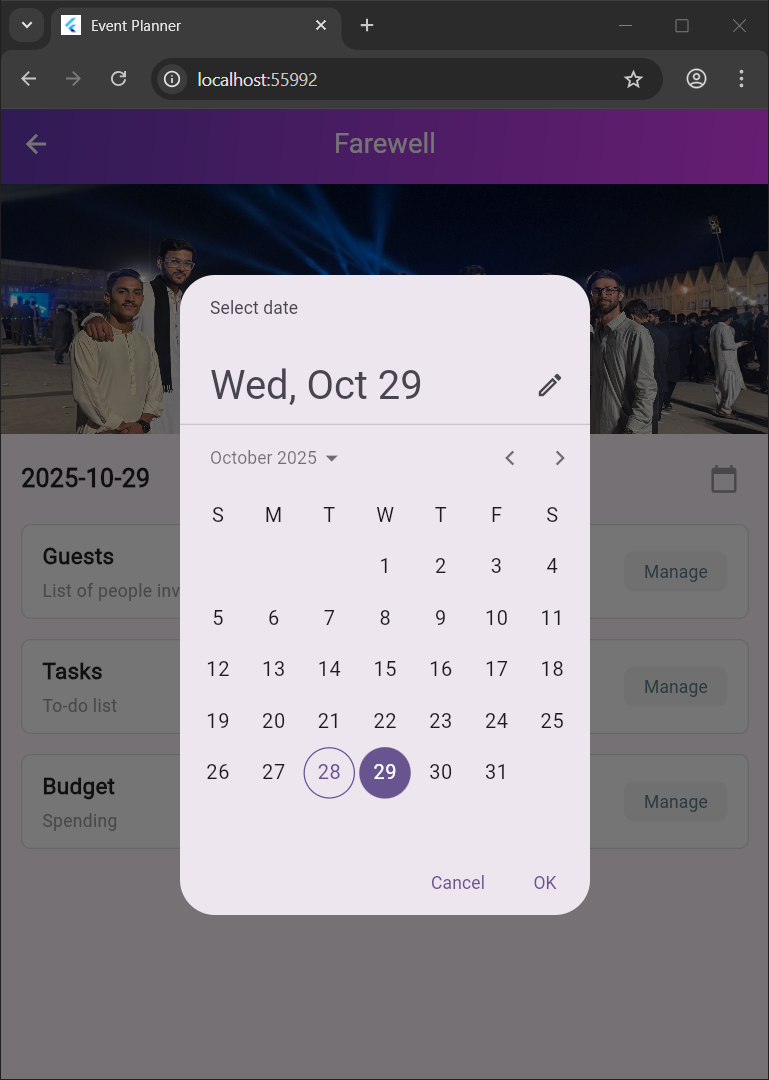


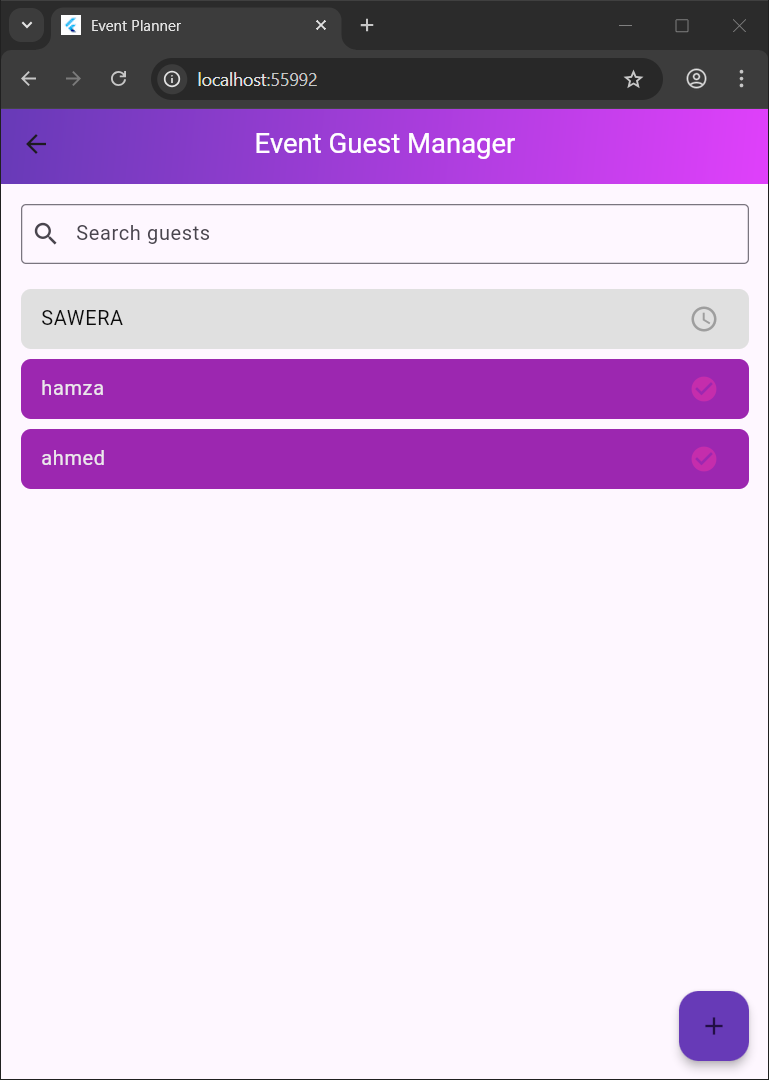


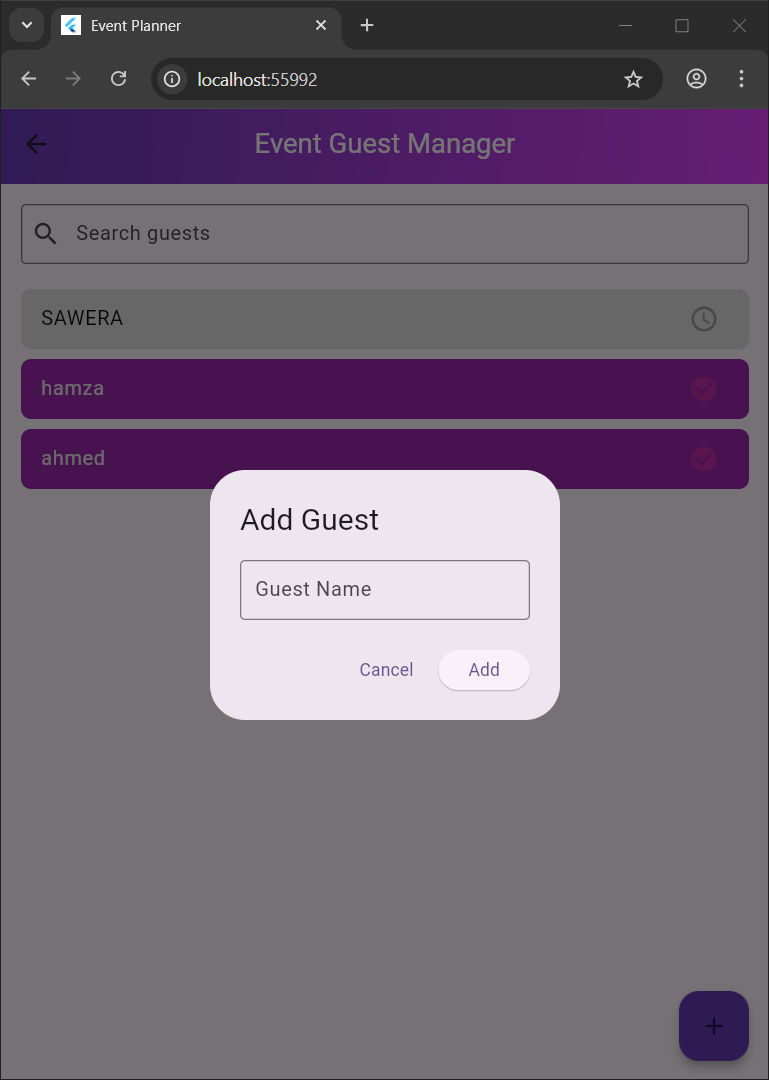


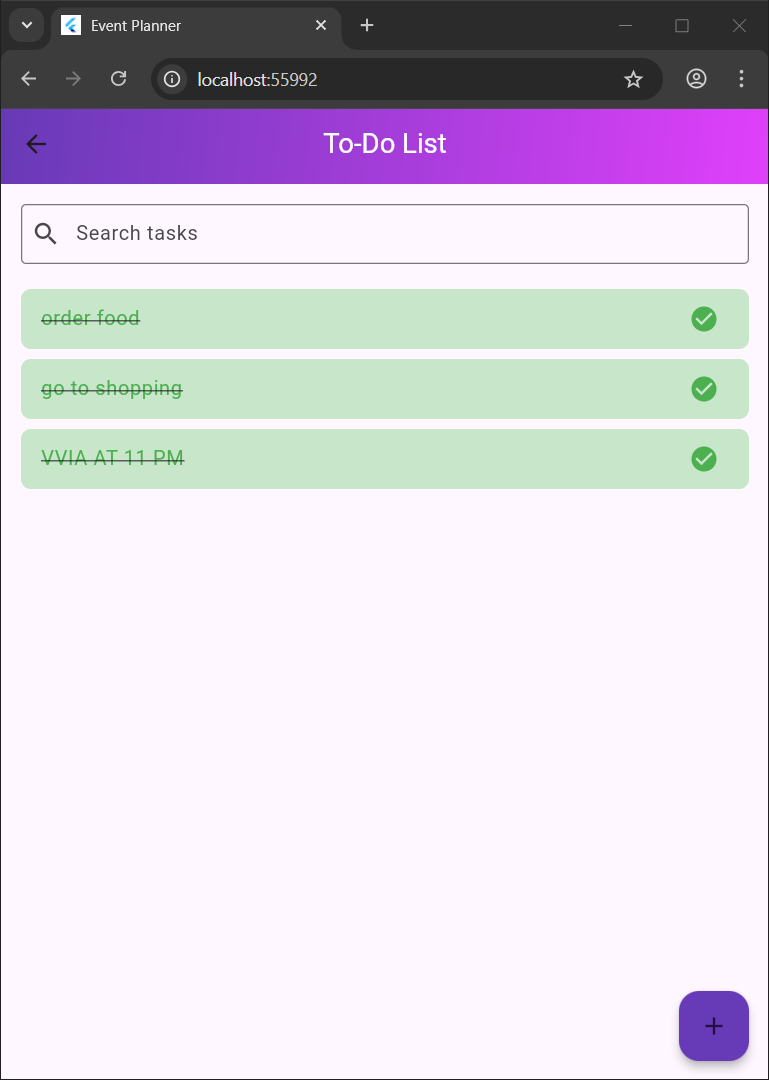


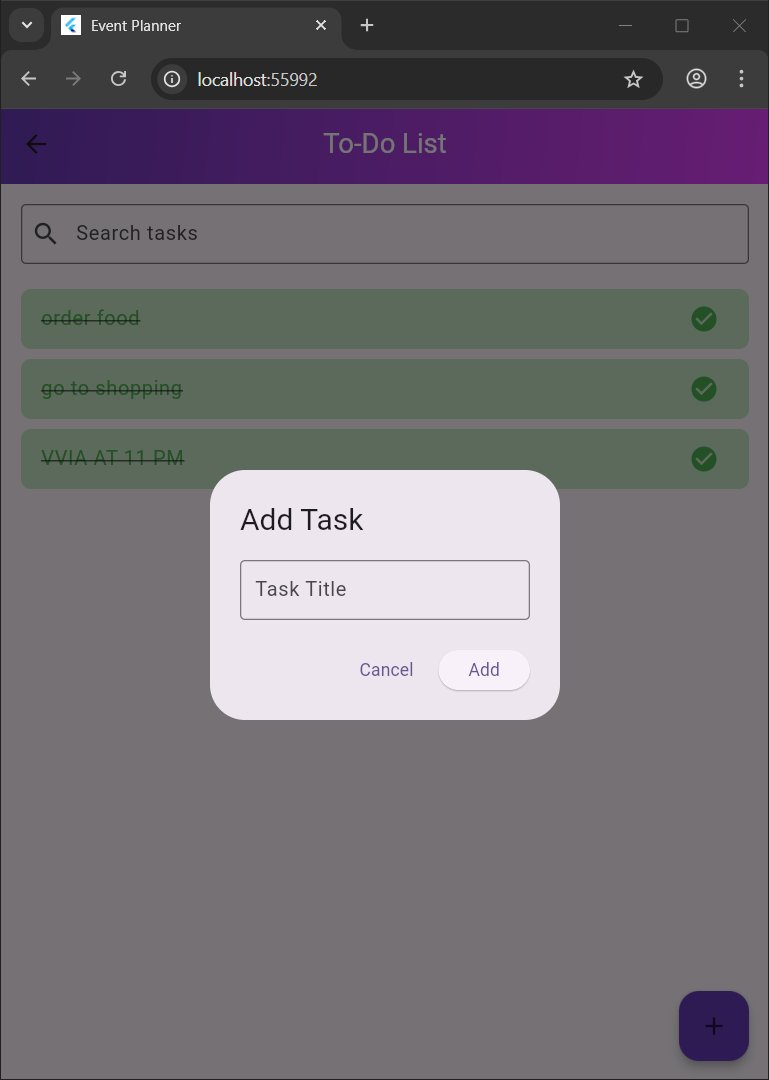


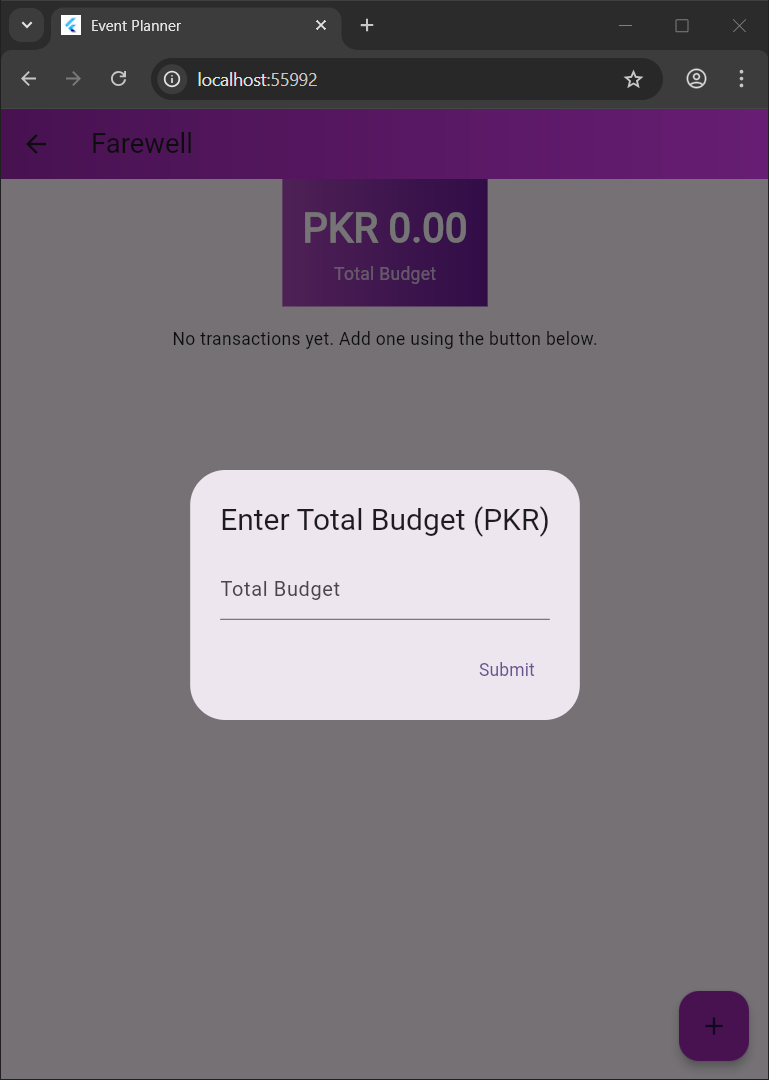


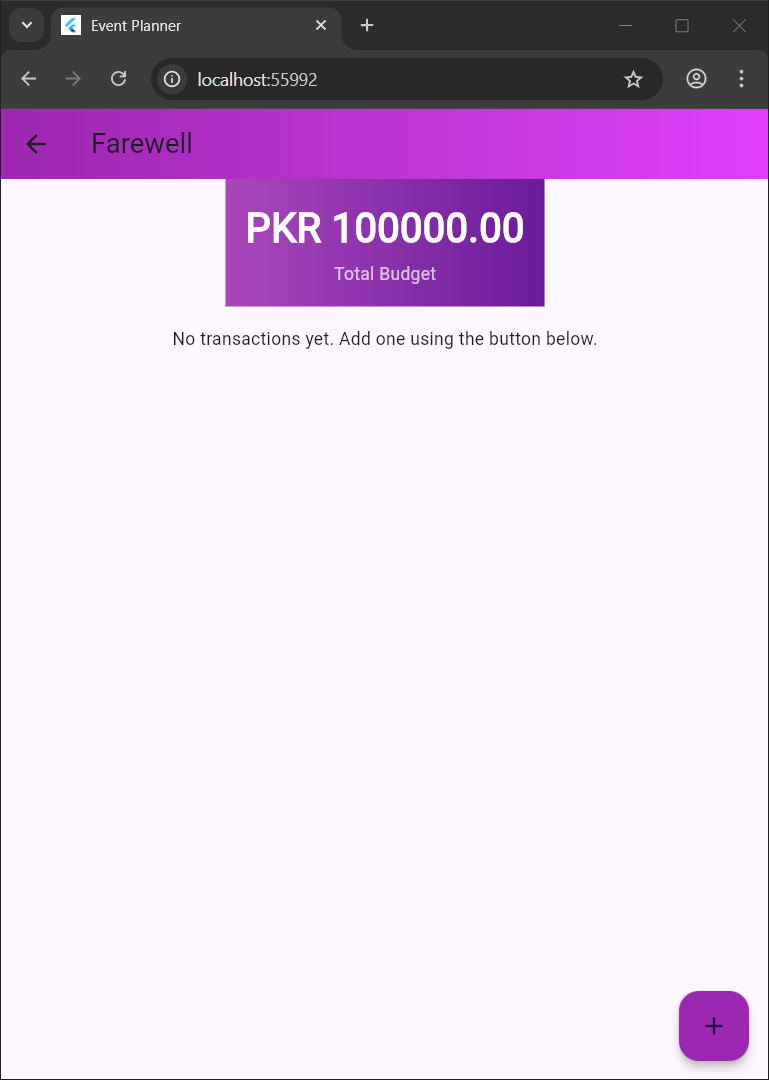


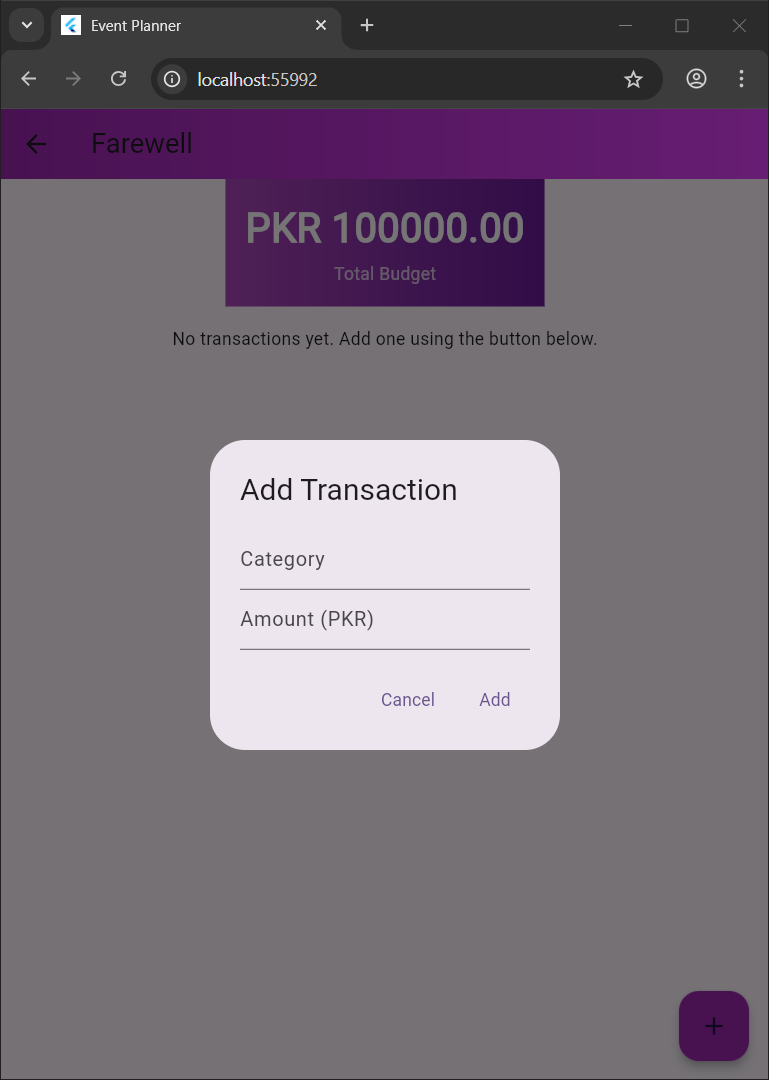


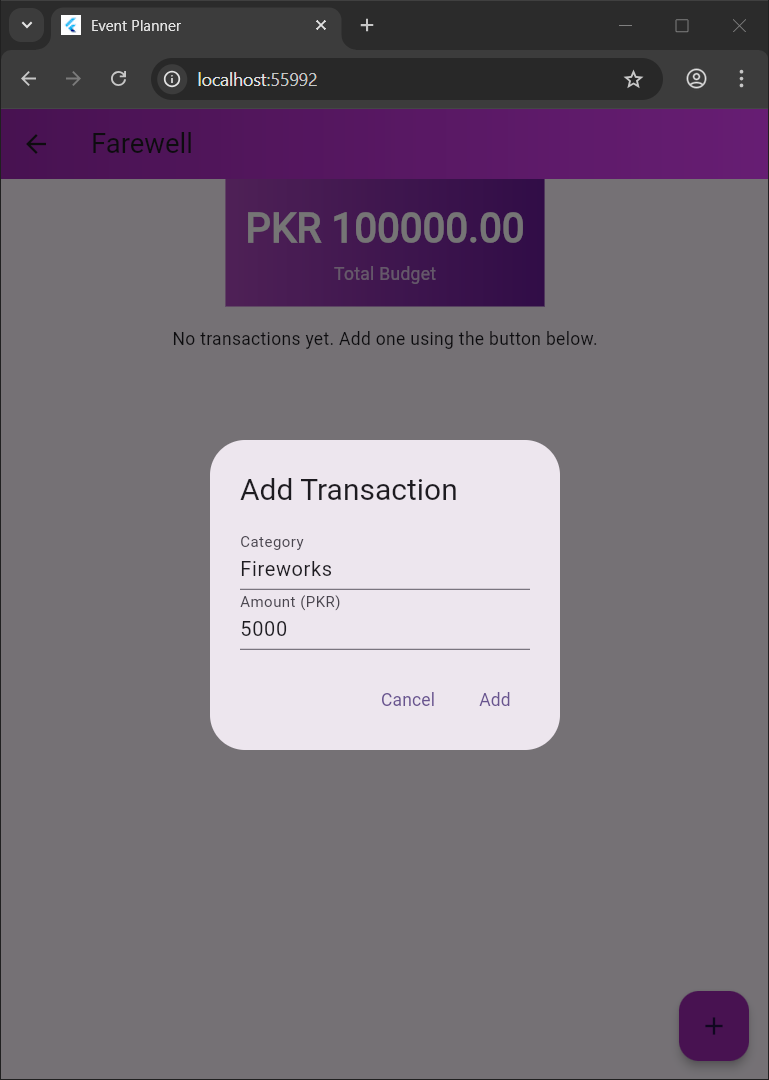


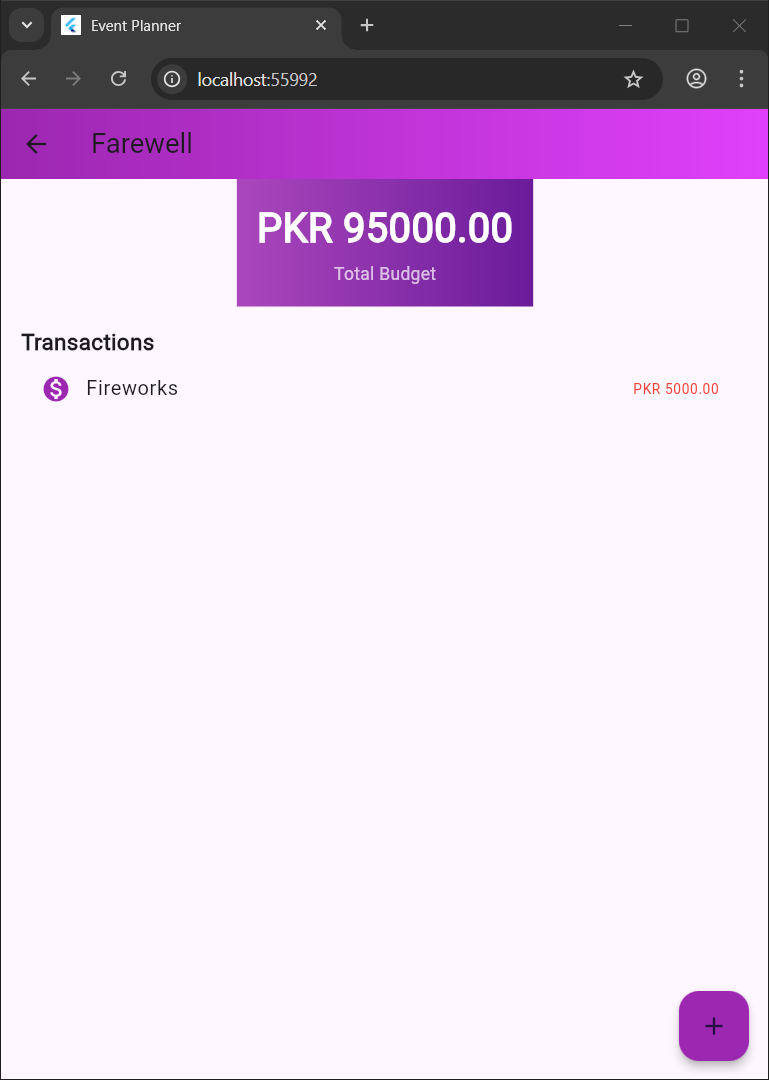












**Data Storage:**

Storage Solution:

Firebase Firestore is used to store data on the cloud, allowing real-time updates and accessibility across

devices.

Justification:

Firebase was chosen due to its seamless integration with Flutter, real-time database capabilities, and

scalability for future app growth

**Optional Section: APIs/Packages/Plug-ins: Firebase**

Packages

• firebase\_core:

o Purpose: Initializes Firebase services within the app.

o Justification: Essential for connecting the app to Firebase’s backend services, enabling

features like authentication and Firestore. Provides a stable foundation for other

Firebase functions.

• firebase\_auth:

o Purpose: Manages user authentication for login, registration, and session management.

o Justification: Simplifies secure user authentication, handling login and signup without

needing to build a custom authentication solution. Ensures data privacy and protection.

• cloud\_firestore:

o Purpose: Provides real-time NoSQL database for storing and retrieving data, like event

details, guest lists, tasks, and budgets.

o Justification: Firestore’s scalability and real-time capabilities make it ideal for dynamic

data updates. Its structure also simplifies data storage and retrieval across multiple users

and devices.

**Issues and Bugs Encountered and Resolved During**

**Development:**

Responsive Layout Problems:

UI elements did not align correctly on different screen sizes, particularly on tablets and larger devices.

State Management Complexity:

Managing state across multiple screens (e.g., Dashboard and DetailScreen) was challenging, causing data

inconsistencies.

Navigation Challenges:

Difficulty in managing navigation between screens, especially when moving back to previous sections.

**Project Deliverables Mapping**

**Working Application**

**GitHub Repository:** <https://github.com/Hanzllasoomro/event-management.git>

* **Branch:** master