

**Department of Software Engineering**

**MEHRAN UNIVERSITY OF ENGINEERING AND**

**TECHNOLOGY, JAMSHORO**

Complex Engineering Problem Report

ON

Blogging Application  
Cross Platform Microblogging Application

Submitted by

Payal Kumari (22SW049)

Hira Mukhtiar(22SW004)

**Section 1**

### 1. Real-World Problem Identification

In small communities, course groups, and campus clubs, communication often relies on fragmented platforms like WhatsApp, Facebook, or email chains. These platforms tend to generate excessive notifications, lack topic organization, and offer limited control over post visibility.  
There is a need for a lightweight, focused, and noise-free communication platform where authenticated members can share concise updates, announcements, or thoughts similar to microblogging but tailored to closed groups.

Thus, the Blogging App addresses this gap by providing a simple, cross-platform space for quick, text-based posts without unnecessary distractions or ads.

### 2. Proposed Solution

The solution is a cross-platform mobile blogging application built using Flutter and integrated with Firebase Authentication and Cloud Firestore.  
The app allows registered users to:

* Log in or sign up securely.
* Post short text updates (“blogs”) to a shared wall.
* View all posts in real time using Firestore’s snapshot streams.
* Automatically sync content across devices and platforms.

The system follows a Model-View-Controller (MVC) inspired architecture to maintain modularity, ensuring scalability and maintainability.

### 3. Key Features

The Blogging App includes the following core functionalities:

1. **User Authentication:**   
   Users can sign up and log in using their email and password. All credentials are securely managed by Firebase Authentication, with form validation to prevent invalid input.
2. **Profile Page:**   
   Users can edit their personal information such as username and bio. This information is stored in the Firestore “users” collection.
3. **Create Post:**   
   Authenticated users can create new posts that appear instantly on the main feed using Firestore real-time updates.
4. **Like and Comment System:**   
   Each post can be liked or commented on by other users. Comments are stored as subdocuments under the related post.
5. **Delete Post (User-Owned Only):**   
   Users can delete only their own posts, ensuring privacy and ownership control.
6. **Responsive Feed:**   
   Posts are displayed in a scrollable, real-time feed with author details and timestamps.

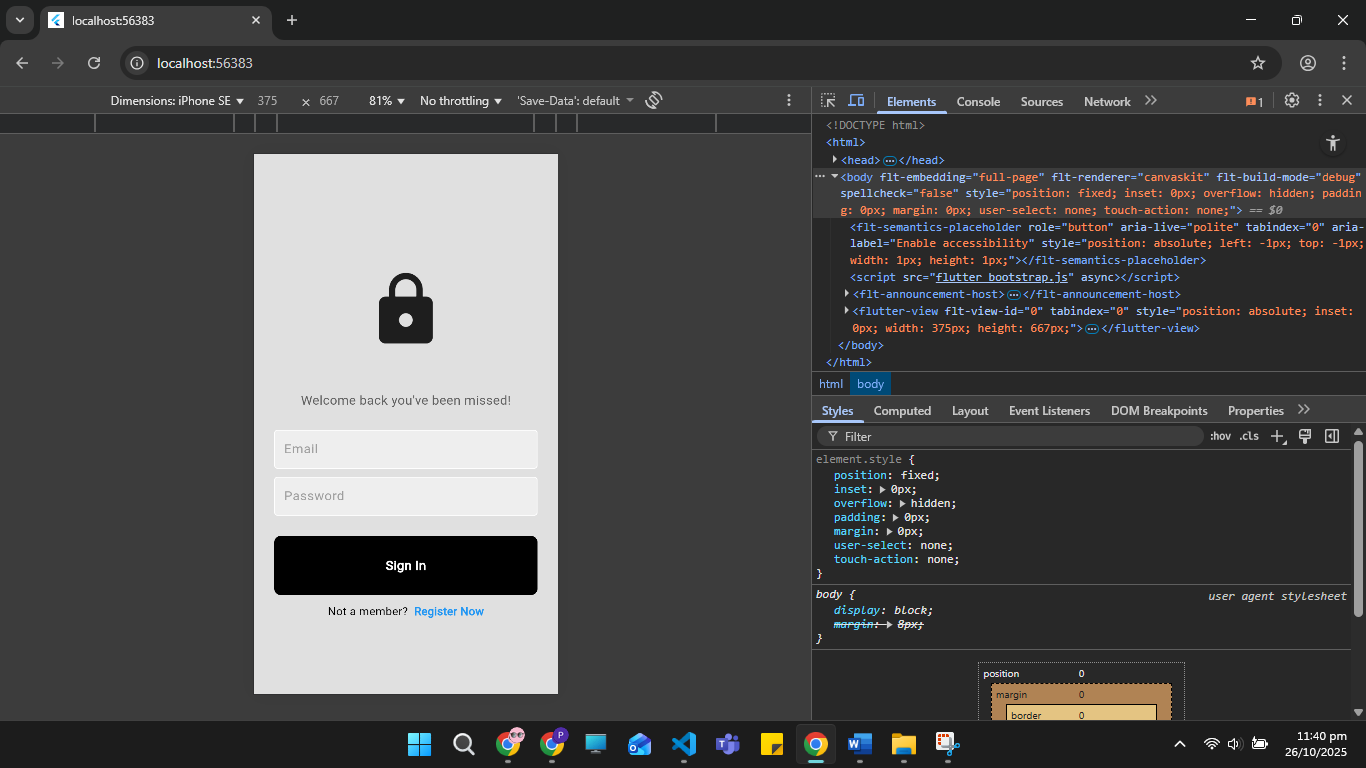
These features collectively provide a complete microblogging experience suitable for small communities and academic groups.

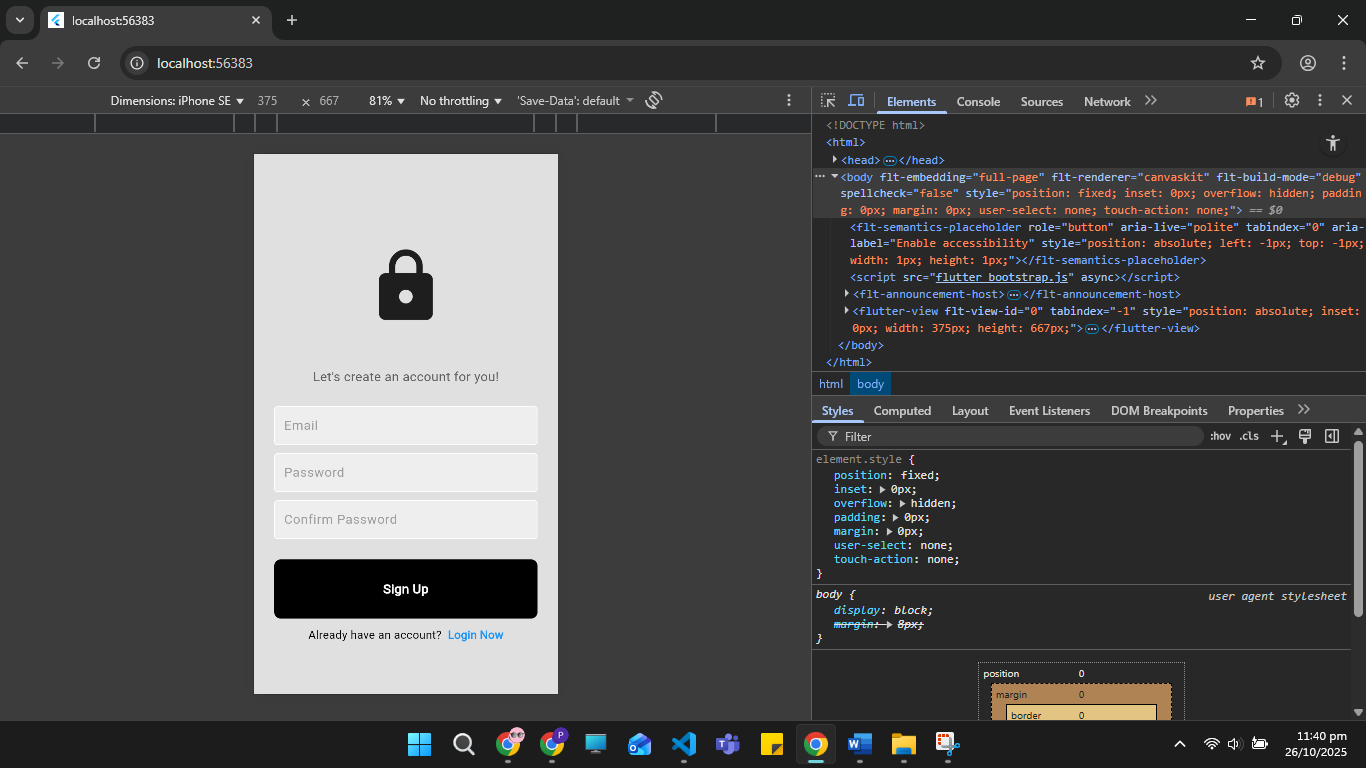
### 4. Responsive User Interfaces

The UI was designed with Flutter’s responsive layout principles to ensure consistency across screen sizes and orientations.  
Material Design widgets are used throughout the interface to maintain uniformity and a native feel.

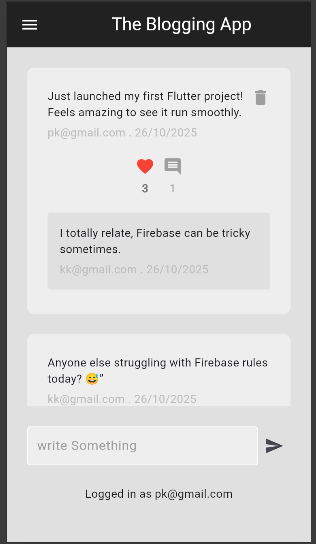
**Key Screens:**

**Login / Sign-Up Screen: User authentication with Firebase.**

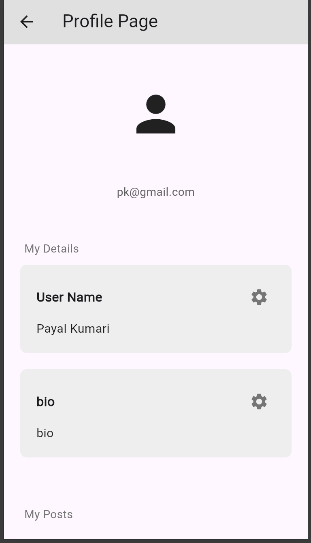
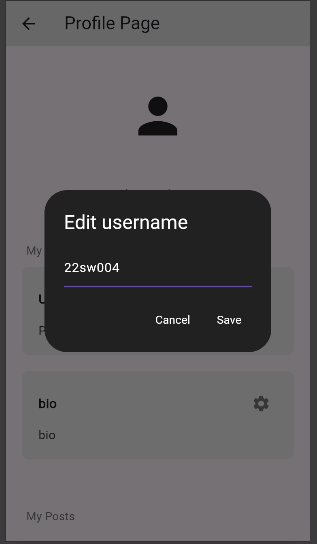




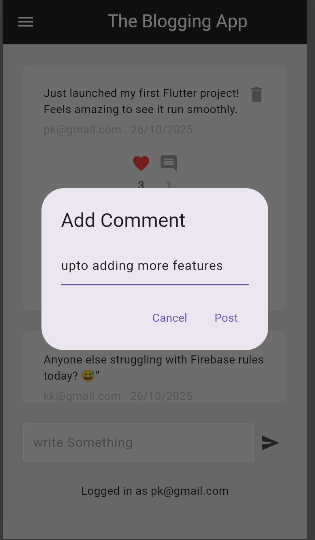
**1. Home Feed: Displays all recent posts fetched via Firestore snapshots.**



1. Add Post Screen: Text input field and “Post” button for publishing updates.
2. Profile Screen: Displays user info and logout or edit options.

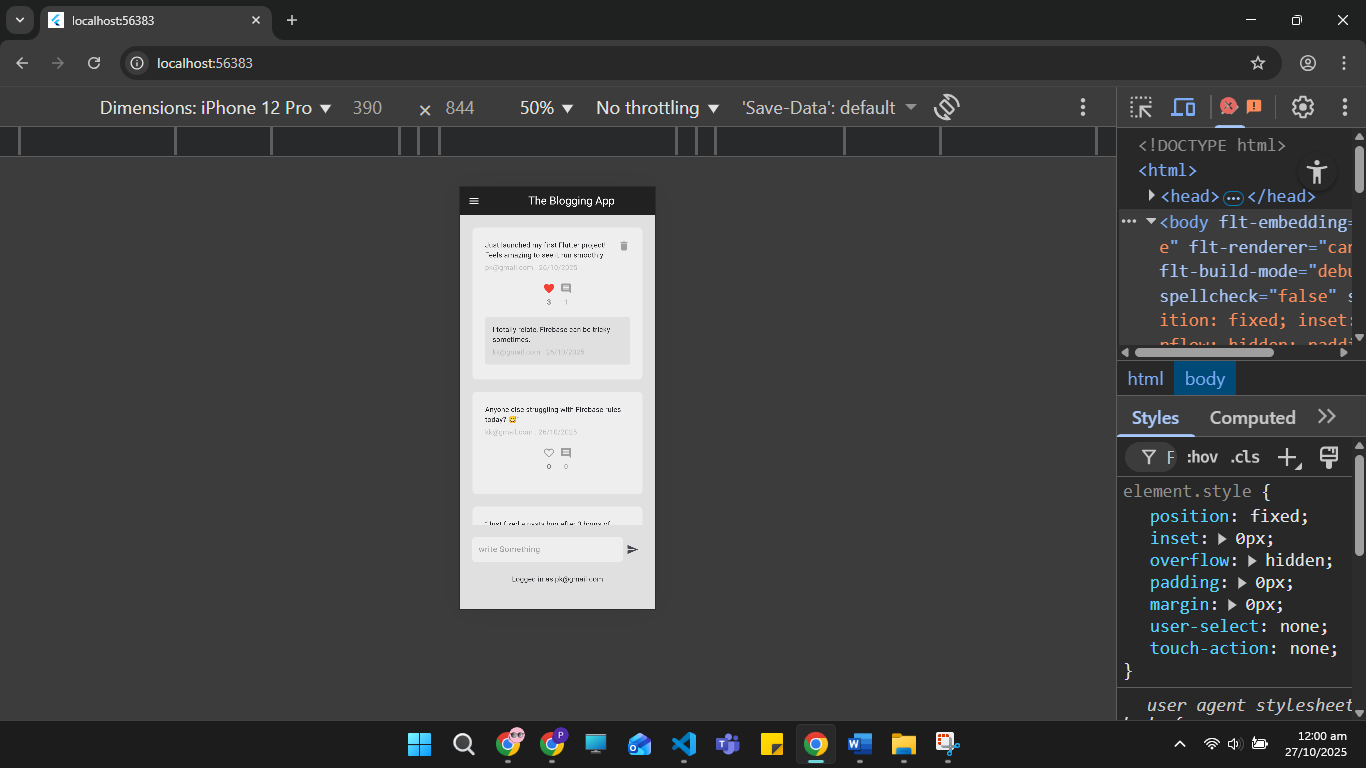
1. **Comment Screen: Allows viewing and adding comments on posts.**



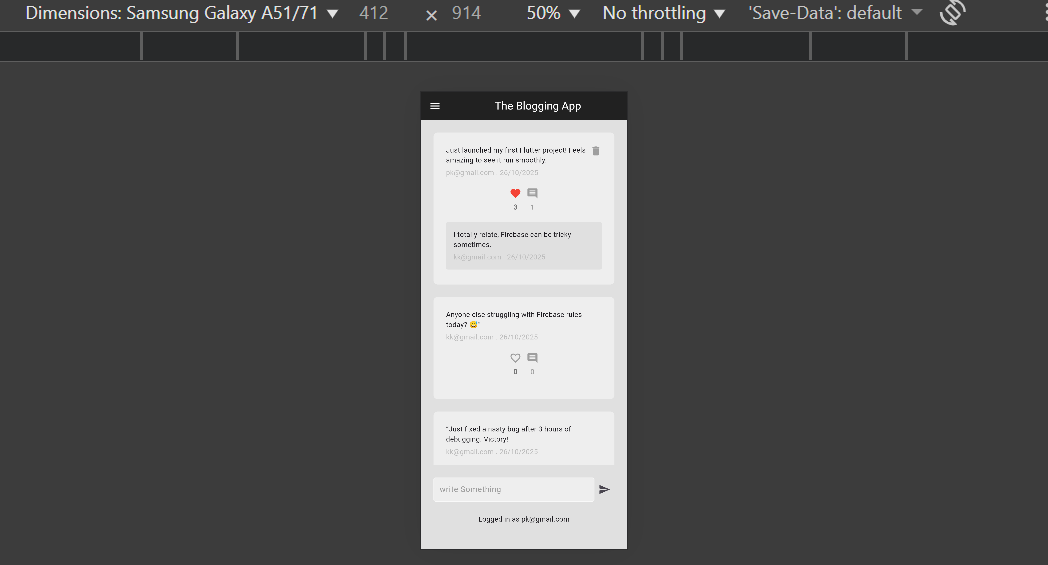
**Responsive Testing:**

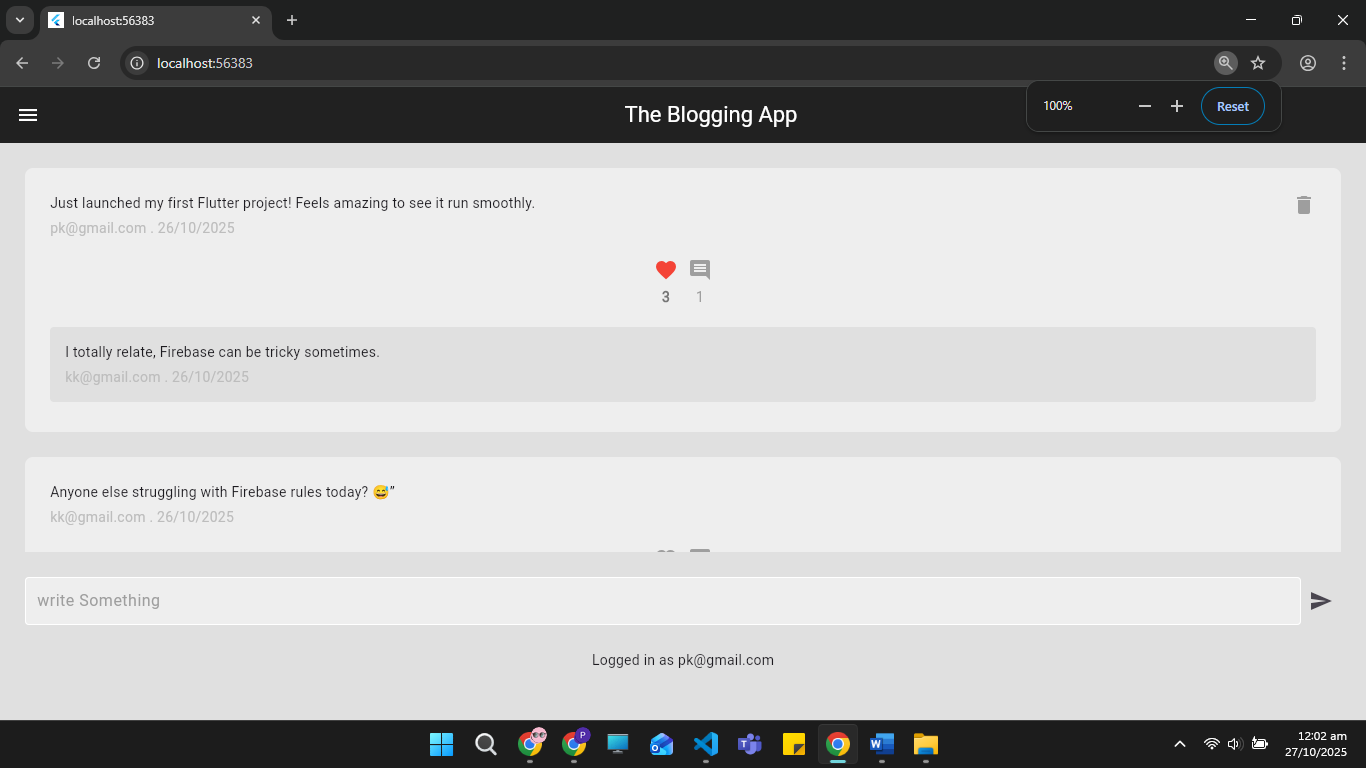
* Verified on Android, iOS, and Web (via Flutter’s web support).

**IOS**



**Samsung**



**Web**  


### 5. Data Storage and Management

**Technology Used:** Firebase Cloud Firestore

**Database Structure:**

* **Collection 1: userposts**  
  Stores all posts and their associated data, including:
  + Author details (name, UID)
  + Post content (text, timestamp)
  + Likes (count and list of user IDs who liked)
  + Subcollection comments for storing individual comments with commenter ID, content, and time
* **Collection 2: users**  
  Stores user profile details such as:
  + Username
  + Bio
  + Email

**Authentication:**

* Firebase Authentication is used for secure user management.
* Users register via **email and password**, which are validated for format and strength.
* Authentication tokens ensure that only logged-in users can interact with the database.

### 6. APIs, Packages, and Plugins Used

|  |  |  |
| --- | --- | --- |
| **Plugin / Package** | **Purpose** | **Justification** |
| firebase\_core | Core Firebase initialization | Enables all Firebase services |
| cloud\_firestore | Database | Real-time posts, likes, and comments |
| firebase\_auth | Authentication | Secure user login and signup |
| provider | State management | Simplifies reactive UI updates |
| intl | Timestamp formatting | For readable time display in posts and comments |

These packages were chosen for their reliability, open-source support, and strong integration with Flutter’s ecosystem.

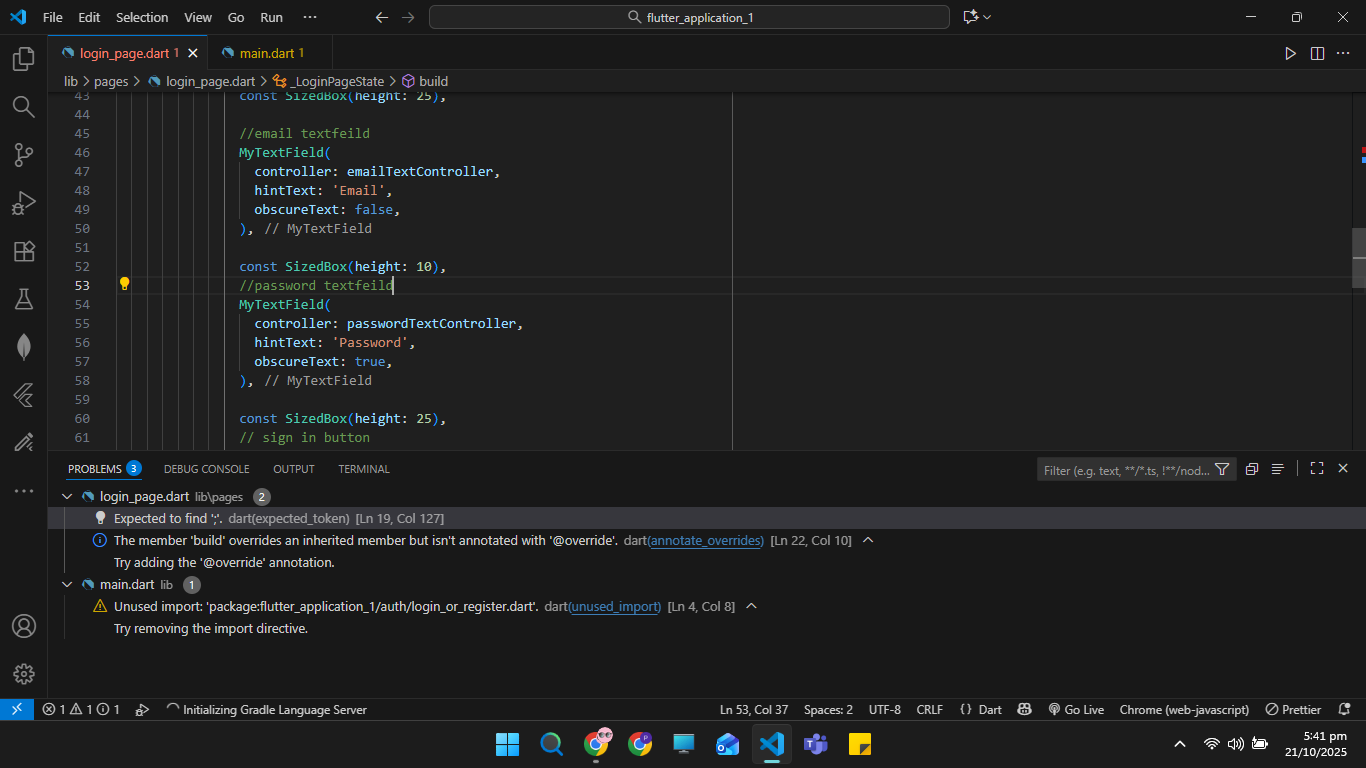
### 7. Issues and Bugs Encountered and Resolved

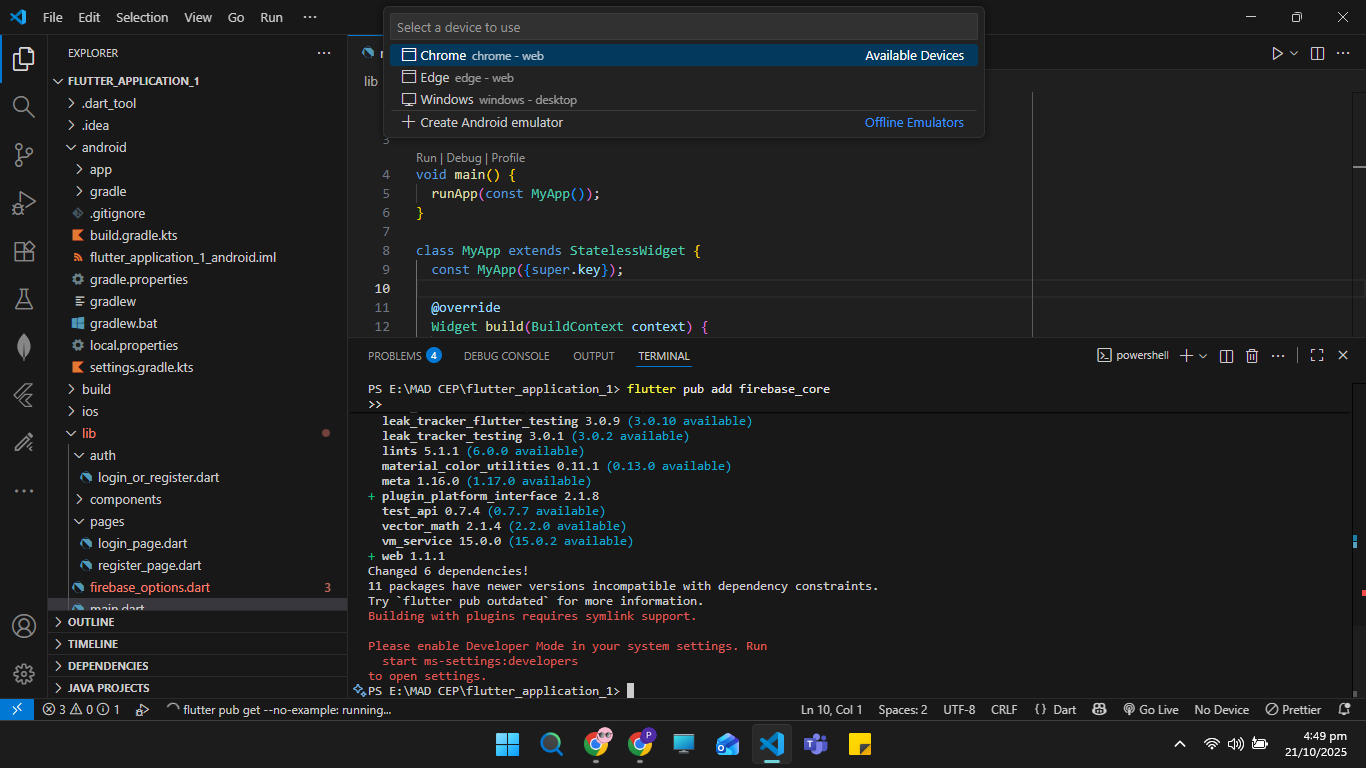
|  |  |  |
| --- | --- | --- |
| **Issue** | **Cause** | **Resolution** |
| Firebase initialization error | Missing configuration in firebase\_options.dart | Regenerated config using FlutterFire CLI |
| Posts not updating in real-time | Improper use of async streams | Switched to Firestore snapshots with StreamBuilder |
| Layout overflow on smaller screens | Fixed pixel sizing in containers | Replaced with Expanded and Flexible widgets |
| Authentication persistence issue | Missing setPersistence() | Added automatic user session management |
| Web build errors | Incomplete dependency setup | Added web compatibility via flutterfire configure |

### Other Issues Encountere and solved as per statements

### 

### 





### 9. Conclusion

The Blogging App successfully fulfills the requirements of a Complex Engineering Problem by integrating multiple technologies Flutter, Firebase, and responsive design to produce a real-time, cross-platform communication tool.  
Through design iteration, debugging, and performance optimization, the project demonstrates professional-level development practice aligned with software engineering principles.