Name:Bhushan Mukund Kor **Academic Year:2024-2025**

Division: D15C **Roll No: 28**

Experiment 6

Aim: To Connect Flutter UI with fireBase database.

Theory:

Firebase Integration in Flutter

Firebase serves as a robust backend service for Flutter apps, offering two primary database solutions: Cloud Firestore and Realtime Database. These databases allow smooth synchronization of data between the app and the cloud, with support for real-time updates and offline access.

Firebase Database Options

- Cloud Firestore: A scalable NoSQL document-oriented database that organizes data into collections and documents. It is ideal for apps with complex and nested data structures and offers advanced querying features.
- Realtime Database: A JSON-based database optimized for real-time performance. It's best suited for applications that demand immediate data updates, such as chat apps or live dashboards.

Key Features

- Real-Time Sync: Both databases offer instant data syncing, ensuring the app UI updates immediately when the data changes.
- Offline Support: Firebase caches data locally on the device, allowing apps to remain functional without internet and automatically sync when reconnected.
- Scalability: Firestore is designed to scale seamlessly with increasing data volumes, while Realtime Database is built for fast and lightweight operations.

Integrating Firebase with Flutter

Setup Steps:

- Add the required packages like firebase_core, cloud_firestore, or firebase_database.
- Configure the Firebase project by including platform-specific files (e.g., google-services.json for Android, GoogleService-Info.plist for iOS).
- Initialize Firebase in the app's entry point (main.dart).

Authentication (Optional): For secure data access, integrate Firebase Authentication to allow only authorized users to read or write data.

CRUD Operations

- Create: Insert new documents in Firestore or nodes in the Realtime Database.
- Read: Retrieve data through direct queries, snapshots, or real-time listeners.
- Update: Modify existing data with support for atomic operations and transactions.

- Delete: Remove specific documents or data nodes, ensuring consistency across related records.

Handling Real-Time Data

- StreamBuilder: A Flutter widget that listens to live data streams from Firestore or Realtime Database and rebuilds the UI accordingly.
- State Management: Tools like Provider, Riverpod, or Bloc can be used alongside Firebase for maintaining and updating application state efficiently.

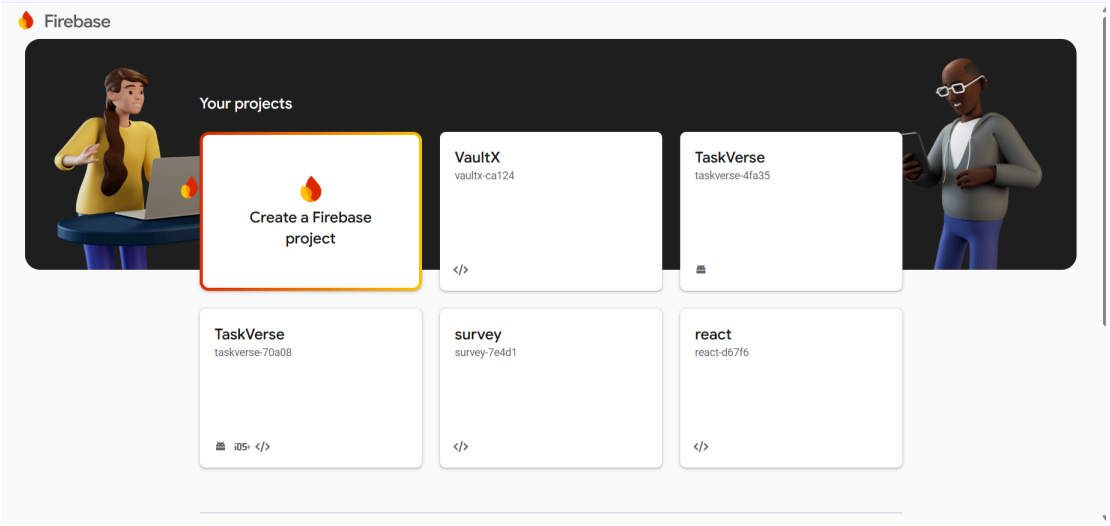
Security and Access Control

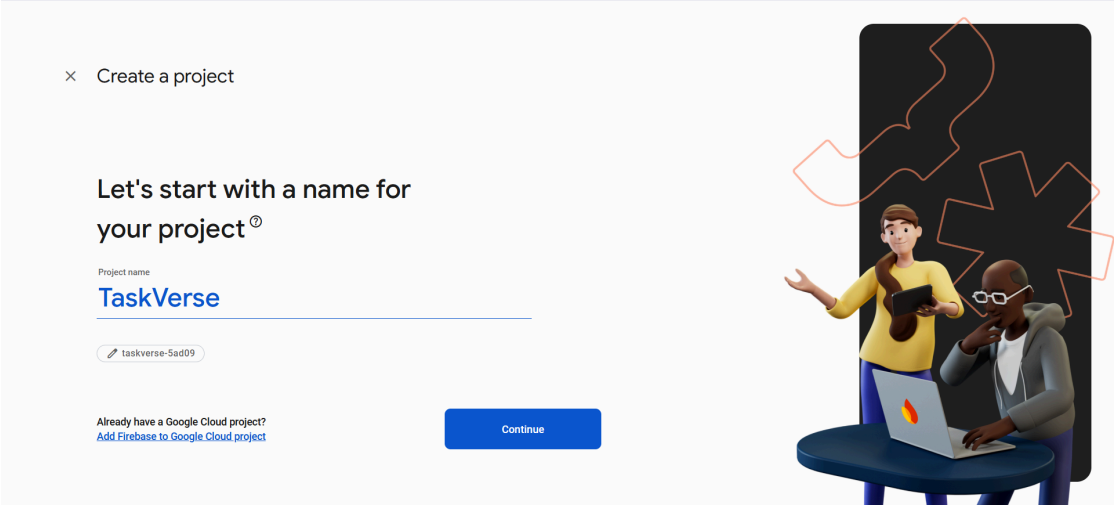
- Firestore Rules: Use Firebase's rule language to define fine-grained access control, such as restricting read/write operations to authenticated users or specific user roles.
- Realtime Database Rules: Similar to Firestore but written in a JSON structure, allowing for flexible role-based permission systems.

Steps:

1. **Create a firebase project.**

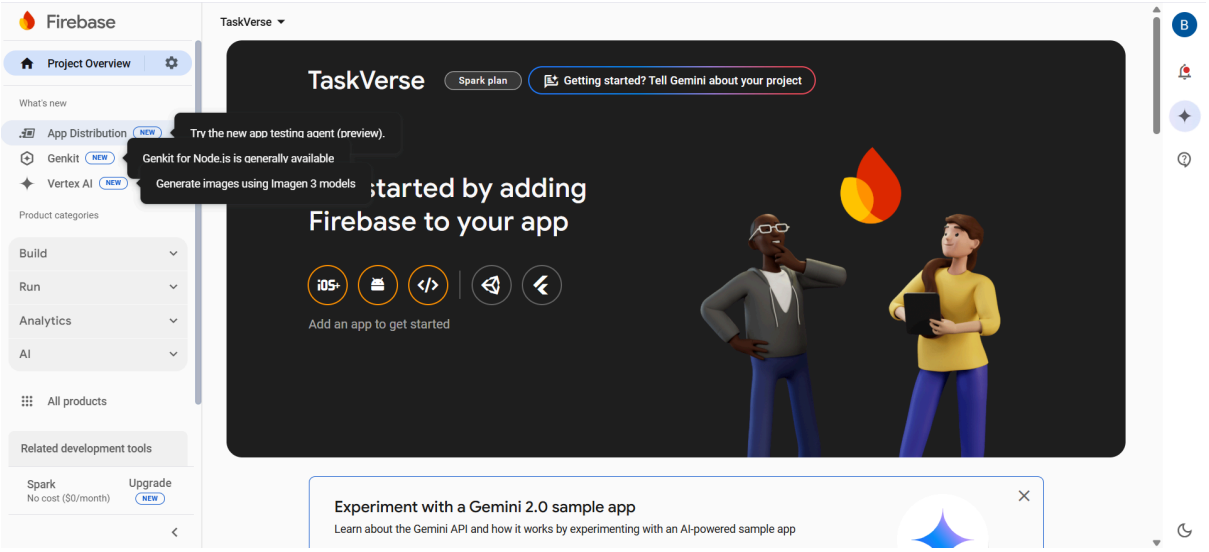
First, log in with your Google account to manage your Firebase projects. From within the Firebase dashboard, select the Create new project button and give it a name:





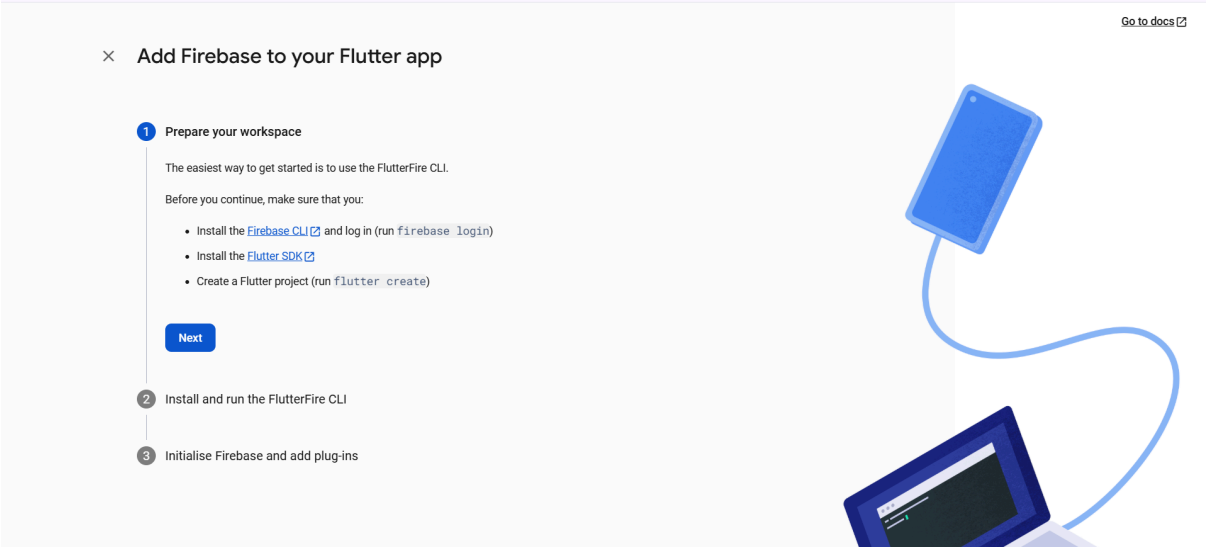
Click on continue, next it asks to enable Gemini and then asks for enable google analytics. We do not require these 2 things for this project, but if you do, simply check the box.

This is the console of our newly created project.



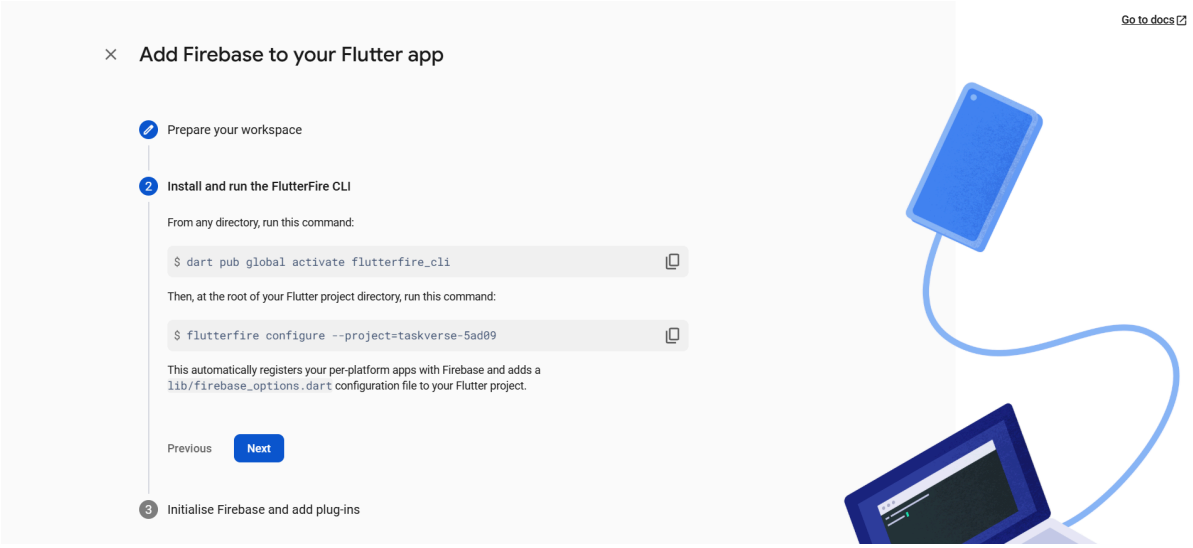
2. Connecting Firebase to Flutter App

On the dashboard, you will see the Flutter logo, click on that to set up a firebase for a Flutter app. Install all the Flutter CLI and other things we have already done so no need to do it agian

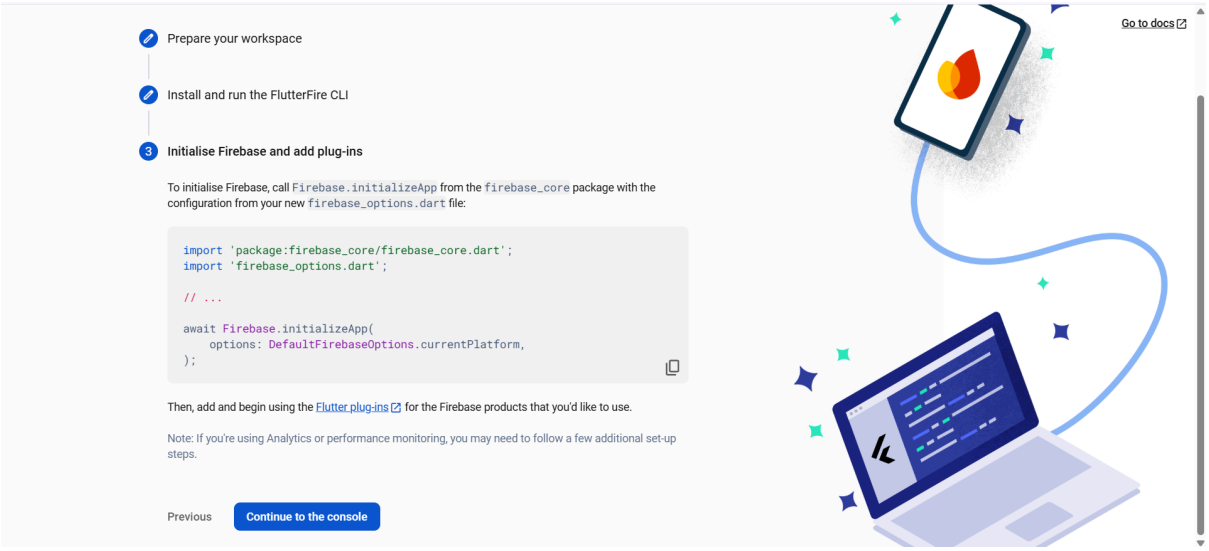


Name:Bhushan Mukund Kor **Academic Year:2024-2025**

Division: D15C **Roll No: 28**



Now Run this commands.It will automatically add all the necessary files and do configurations.

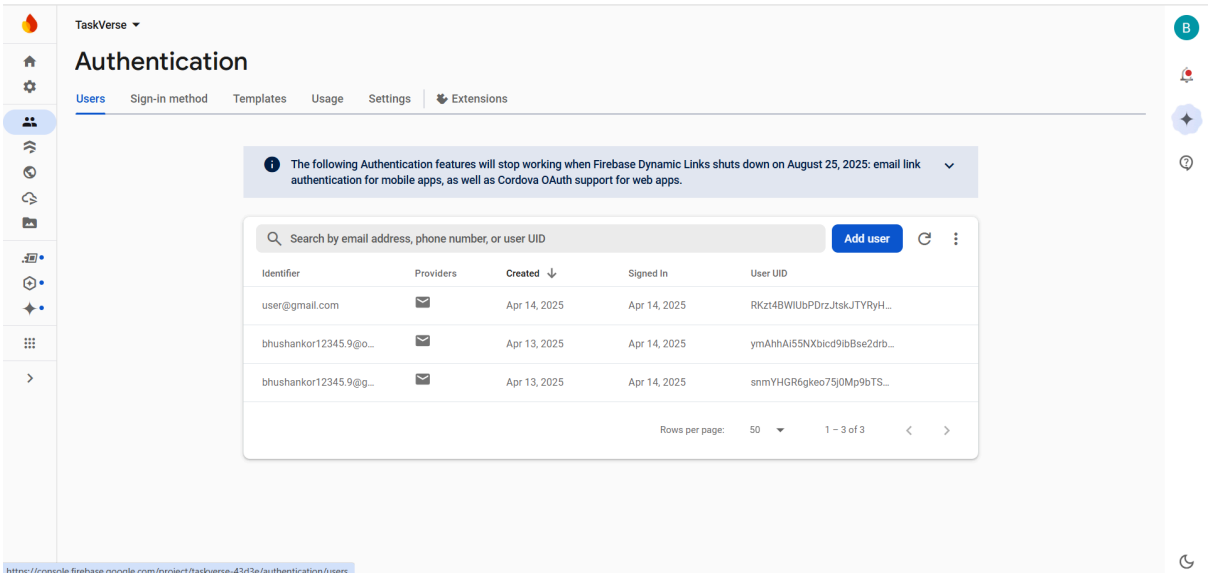


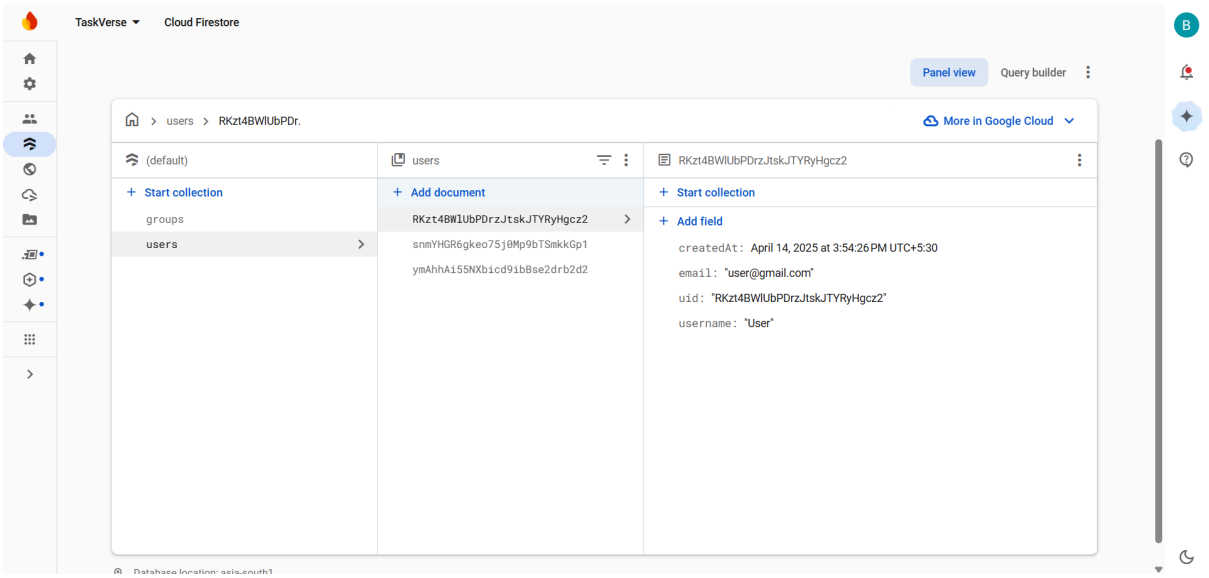
Now Add this Code in main.dart to initialise the firebase and add plugins.

Name:Bhushan Mukund Kor **Academic Year:2024-2025**

Division: D15C **Roll No: 28**

Authorization and firestore:





GitHub Link: https://github.com/BKCODE2003/ToDo

Conclusion:

By successfully connecting the Flutter UI with the Firebase database, you have learned how to create dynamic and data-driven applications. This integration allows real-time data synchronization between the app and the backend, enabling functionalities such as data storage, retrieval, and updates. Understanding how to connect Flutter with Firebase is essential for developing modern applications that require persistent data, user management, and seamless cloud connectivity.