SMUCT Bus Tracking Flutter App: Project Overview & FAQ

Project Overview

Project Name

- **flutterbus** (SMUCT Bus Tracking)

Purpose

- A Flutter application for student authentication and bus tracking, using Firebase for authentication and Firestore for user data, and Google Maps for bus tracking.

Main Features

- **User Authentication:** Email/password and Google Sign-In using Firebase.
- **User Registration:** Collects name, email, password, department, student ID, semester, batch, and contact number.
- **Profile Management:** Displays user info from Firestore.
- **Bus Tracking:** Shows a bus route and status on a Google Map.
- **Navigation:** Auth check, login, signup, profile, and bus tracking screens.

Main Files and Their Roles

`lib/main.dart`

- Initializes Firebase (with web config if running on web).
- Sets up the root widget (`MyApp`), which uses `AuthCheckScreen` as the home.

`lib/screens/auth_check_screen.dart`

- Listens to Firebase authentication state.
- Shows `ProfileScreen` if logged in, otherwise shows `LoginScreen`.

`lib/screens/login_screen.dart`

- UI for user login (email/password and Google).
- Handles login logic and error display.
- Navigates to signup via a button.

`lib/screens/signup_screen.dart`

- UI for user registration.
- Validates and collects user info.
- On successful signup, creates a user document in Firestore and returns to login.
- Has a “Already Registered? Log in here.” button at the top.

`lib/screens/profile_screen.dart`

- Displays user info from Firestore.
- Allows sign out.
- Has a button to go to the bus tracking screen.

`lib/screens/bus_tracking_screen.dart`

- Shows a Google Map with a sample route and bus status.
- Displays bus details and a timeline of stops.

API Usage & Backend Explanation

Where APIs Are Used

1. Firebase Authentication API

- **Used in:** `login_screen.dart`, `signup_screen.dart`, `auth_check_screen.dart`, `profile_screen.dart`
- **Purpose:** Handles user sign up, login, logout, and authentication state.
- **How it works:**
  - The app uses `FirebaseAuth.instance` to create users, sign in, and listen for authentication state changes.
  - Google Sign-In is integrated using the `google_sign_in` package, which provides OAuth tokens to Firebase for authentication.

2. Cloud Firestore API

- **Used in:** `signup_screen.dart`, `login_screen.dart`, `profile_screen.dart`
- **Purpose:** Stores and retrieves user profile data.
- **How it works:**
  - On signup, user details are saved in Firestore under the `users` collection.
  - On login (especially with Google), the app checks/updates the user document.
  - The profile screen listens to changes in the user’s Firestore document and displays the data.

3. Google Maps API

- **Used in:** `bus_tracking_screen.dart`
- **Purpose:** Displays a map and bus route.
- **How it works:**
  - The app uses the `google_maps_flutter` package to embed a Google Map.
  - The map is initialized with a camera position and can be controlled via the `GoogleMapController`.

How the Backend Works

- **Backend:** The app uses Firebase as a backend-as-a-service (BaaS).
- **Authentication:**
  - Handles user registration, login, and session management.
  - Supports both email/password and Google OAuth.
- **Database:**
  - Firestore is used to store user profiles.
  - Each user has a document in the `users` collection, keyed by their UID.
- **No custom server:**
  - All backend logic is handled by Firebase services.
  - The app communicates directly with Firebase via the provided SDKs.

How the Connection is Made

- **Firebase Initialization:**
  - In `main.dart`, `Firebase.initializeApp()` is called with web options if running on web.
- **Authentication:**
  - The app calls methods like `signInWithEmailAndPassword`, `createUserWithEmailAndPassword`, and `signInWithCredential` (for Google).
  - The SDK handles secure communication with Firebase servers.
- **Firestore:**
  - The app uses `FirebaseFirestore.instance.collection('users').doc(uid)` to read/write user data.
  - Real-time updates are received via `snapshots()` streams.
- **Google Maps:**
  - The map widget connects to Google Maps using the API key (must be set up in the project).

How Each Part Works

- **main.dart:** Initializes Firebase and launches the app.
- **auth_check_screen.dart:** Listens for authentication state and routes to the correct screen.

- **login__screen.dart:** Handles login logic, error display, and navigation to signup.
- **signup__screen.dart:** Handles registration, validation, Firestore user creation, and navigation back to login.
- **profile__screen.dart:** Displays user info from Firestore, allows logout, and navigation to bus tracking.
- **bus__tracking__screen.dart:** Shows a map and bus route, with a timeline of stops.

Questions & Answers (FAQ)

UI/Navigation Modification

Q: On the signup page, there's a button labeled 'Already Registered'. When clicked, it takes the user to the login page. Right now, it's positioned at the top — I want to move it to the bottom. How can I do that?
A: In `signup_screen.dart`, the "Already Registered?" row is near the top of the form. To move it to the bottom, cut the `Row` widget containing the button and paste it just before the last `SizedBox(height: 20)` at the end of the `Column` in the form.

Q: I want the 'Sign up' button on the signup page to be at the top, above the form fields. How can I do that?
A: Move the `ElevatedButton` widget for 'Sign up' from the bottom of the `Column` to the top, just after the title.

Q: On the login page, how do I add a 'Forgot Password?' link below the password field?
A: In `login_screen.dart`, add a `TextButton` widget below the password `TextField` in the `Column`.

Q: How do I make the profile screen show the user's email in bold?
A: In `profile_screen.dart`, update the `Text` widget for email to use `style: TextStyle(fontWeight: FontWeight.bold)`.

Code and Logic

Q: How does the app know if a user is logged in or not?
A: The `AuthCheckScreen` uses a `StreamBuilder` on `FirebaseAuth.instance.authStateChanges()`. If the user is logged in, it shows the `ProfileScreen`; otherwise, it shows the `LoginScreen`.

Q: Where is the user data (like name, department, etc.) stored?
A: User data is stored in Firestore under the `users` collection, with each user's UID as the document ID.

Q: How can I add a new field to the user profile (e.g., address)?
A: Add a new `TextEditingController` and input field in `signup_screen.dart`, update the Firestore `set` call to include the new field, and update `profile_screen.dart` to display it.

Q: How do I change the initial location shown on the bus tracking map?
A: In `bus_tracking_screen.dart`, change the `_center` variable to the desired latitude and longitude.

Q: How do I add a password confirmation field to the signup form?
A: Add a new `TextEditingController` for password confirmation, add a new input field, and update the validator to check if both passwords match.

Q: How do I change the app's primary color?
A: In `main.dart`, update the `primarySwatch` in the `ThemeData` of `MaterialApp`.

Q: How do I show a loading spinner during login or signup?
A: The app already shows a `CircularProgressIndicator` when `_isLoading` is true in both login and signup screens.

Q: How do I navigate from the profile screen to the bus tracking screen?
A: The floating action button in `profile_screen.dart` uses `Navigator.of(context).push` to open `BusTrackingScreen`.

Q: How do I sign out?
A: Tap the logout icon in the app bar on the profile screen. This calls `FirebaseAuth.instance.signOut()` and `GoogleSignIn().signOut()`.

Q: How do I handle login errors?
A: Login errors are caught and displayed using a `SnackBar` in `login_screen.dart`.

Q: How do I add Google Maps API key for web?
A: Add your API key to the `web/index.html` file as per the Google Maps Flutter documentation.

Q: How do I validate user input in the signup form?
A: Each `TextFormField` in `signup_screen.dart` has a `validator` function that checks for empty fields, valid email, and password length.

Q: How do I listen for real-time updates to user data?
A: The profile screen uses a `StreamBuilder` on the user's Firestore document, so any changes are reflected instantly.

Q: How do I handle errors from Firebase?
A: Errors are caught in `try-catch` blocks and displayed using `SnackBar` or error messages in the UI.

Q: How do I add a new screen to the app?
A: Create a new Dart file in `lib/screens/`, define a `StatelessWidget` or `StatefulWidget`, and use `Navigator.of(context).push` to navigate to it.

Q: How do I restrict access to the bus tracking screen to only logged-in users?
A: Only authenticated users can reach the profile screen, and the bus tracking screen is only accessible from there.

Troubleshooting

Q: I get a 'FirebaseAuthException: invalid-credential' error. What does it mean?
A: This means the credentials provided are incorrect, malformed, or expired. Double-check your email/password or Google sign-in setup.

Q: Google Maps shows a 'BillingNotEnabledMapError'. What should I do?
A: Enable billing for your Google Cloud project and ensure your Maps API key is set up correctly.

Q: The app says 'User data not found' on the profile screen. Why?
A: This means there is no Firestore document for the current user. Ensure the signup process completes successfully and creates the user document.

Q: How do I update the app to use a new Firebase project?
A: Update the `firebaseOptions` in `main.dart` with your new project's credentials, and update the `google-services.json` and `GoogleService-Info.plist` files for Android and iOS.

UI/UX

Q: How do I change the gradient background on the login or signup screens?
A: Update the `BoxDecoration`'s `LinearGradient` colors in the respective screen's `Container`.

Q: How do I make the 'Sign up' button disabled until all fields are valid?
A: Check the form's validity before enabling the button, or use the `Form`'s `onChanged` callback to update the button state.

Q: How do I add a profile picture to the user profile?
A: Add an image picker to the signup/profile screen, upload the image to Firebase Storage, and save the URL in Firestore.

Q: How do I add more bus routes to the tracking screen?
A: Update the data in `bus_tracking_screen.dart` to include more routes, or fetch them from Firestore if you want dynamic data.
