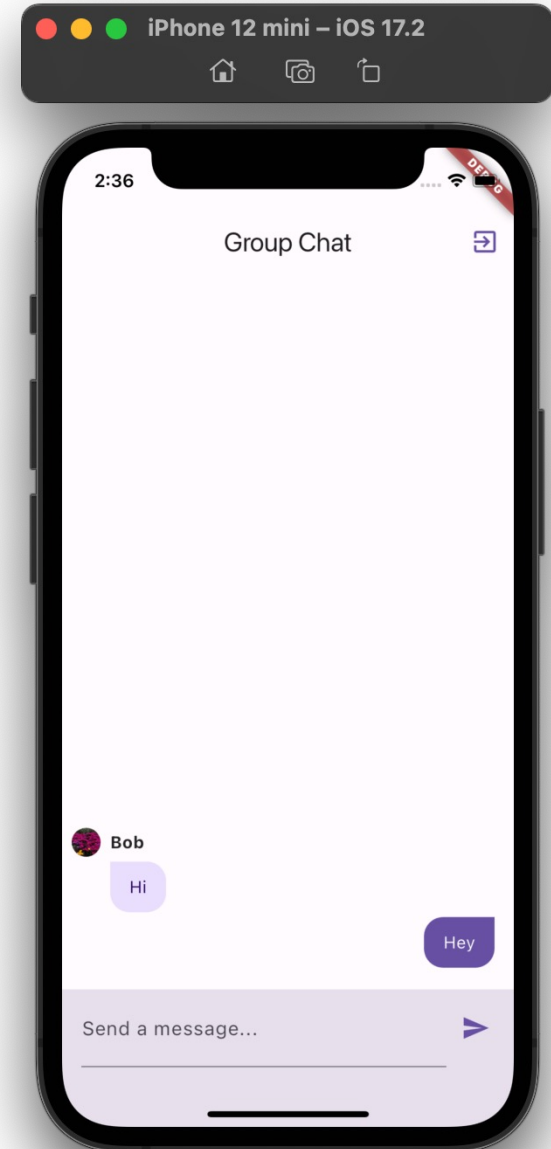


Authentication & Image Upload

Shan-Hung Wu
CS, NTHU

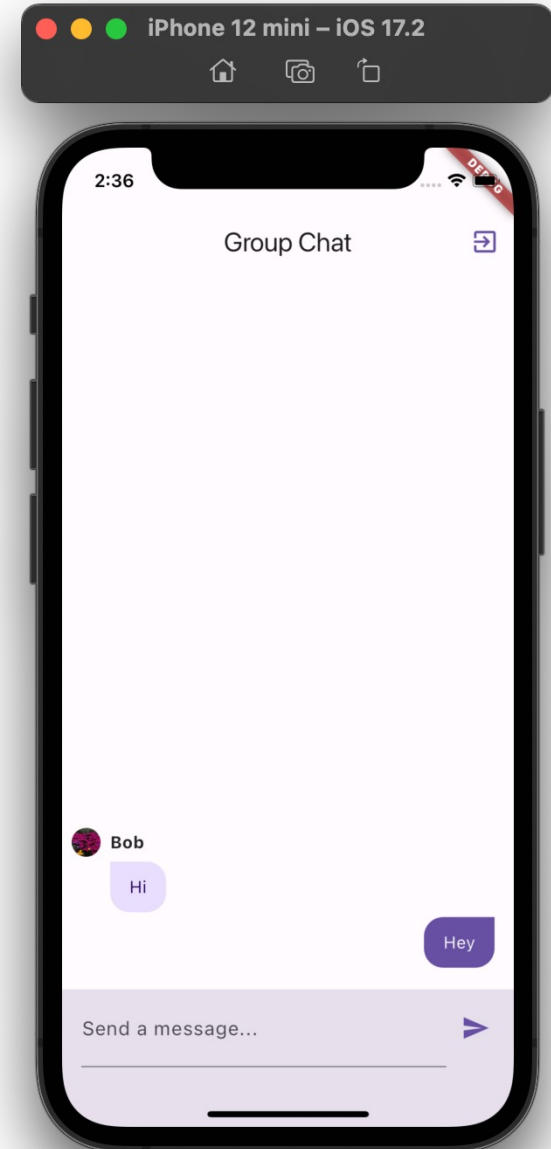
Let's Chat!

- Real-time messaging
- Authentication
 - Sign up & log in
 - Single sign on
- Splash screen
- Image upload
- Security rules
- Custom claims in JWT
- Push notifications



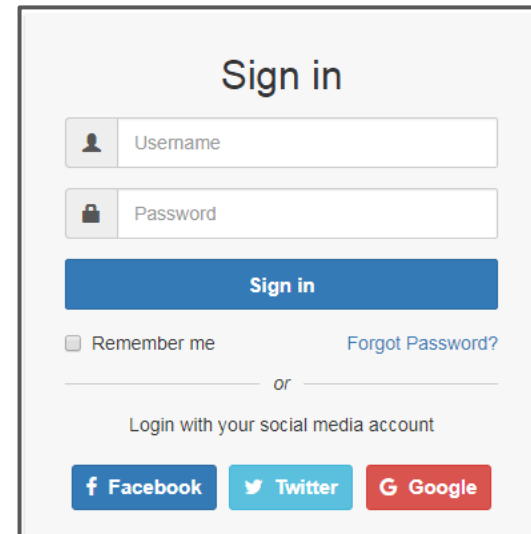
Let's Chat!

- Real-time messaging
- **Authentication**
 - Sign up & log in
 - Single sign on
- Splash screen
- Image upload
- Security rules
- Custom claims in JWT
- Push notifications

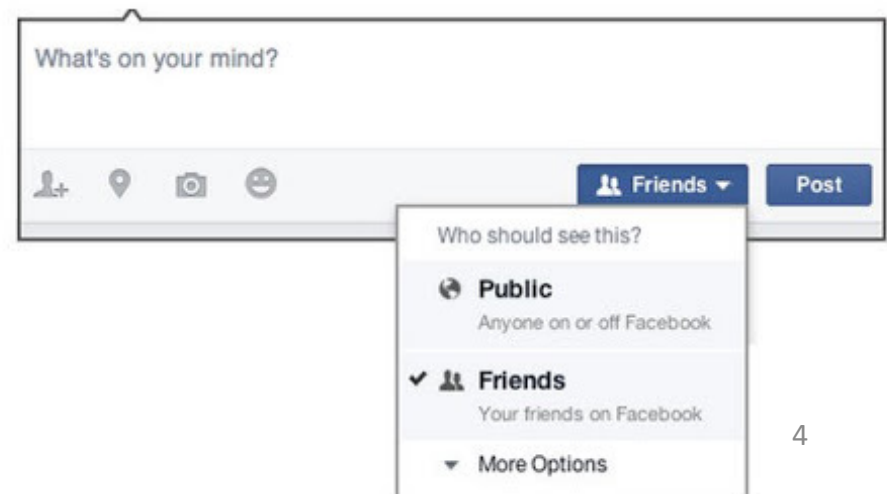


Authentication vs. Authorization

- **Authentication**: the process to verify you are who you said
 - Firebase Auth
- **Authorization**: the process to decide if you have permission to access a resource
 - Firestore security rules



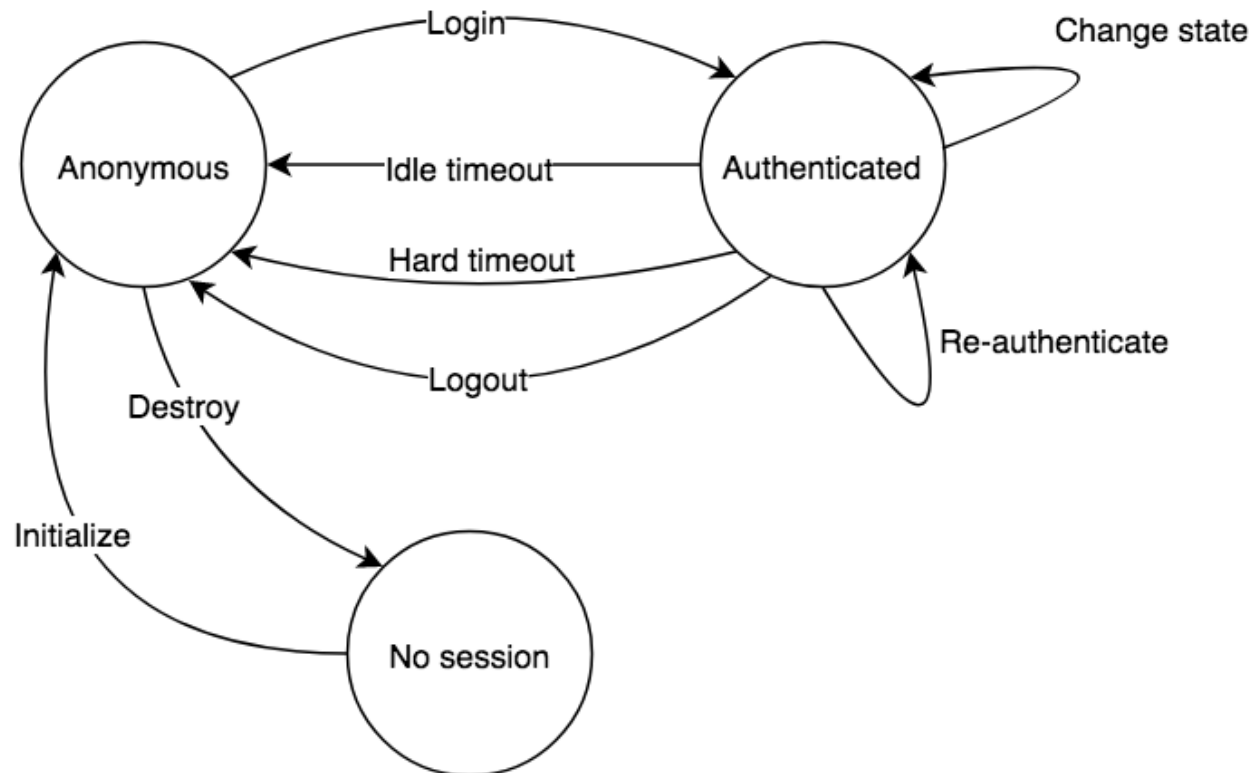
A screenshot of a Facebook 'Sign in' form. It features a title 'Sign in' at the top. Below it are two input fields: 'Username' with a person icon and 'Password' with a lock icon. A blue 'Sign in' button is positioned below the password field. Underneath the button are checkboxes for 'Remember me' and a link for 'Forgot Password?'. A horizontal line with the word 'or' in the center separates this from the social media login section. This section is titled 'Login with your social media account' and includes three buttons: 'Facebook' (with the 'f' logo), 'Twitter' (with the bird logo), and 'Google' (with the 'G' logo).



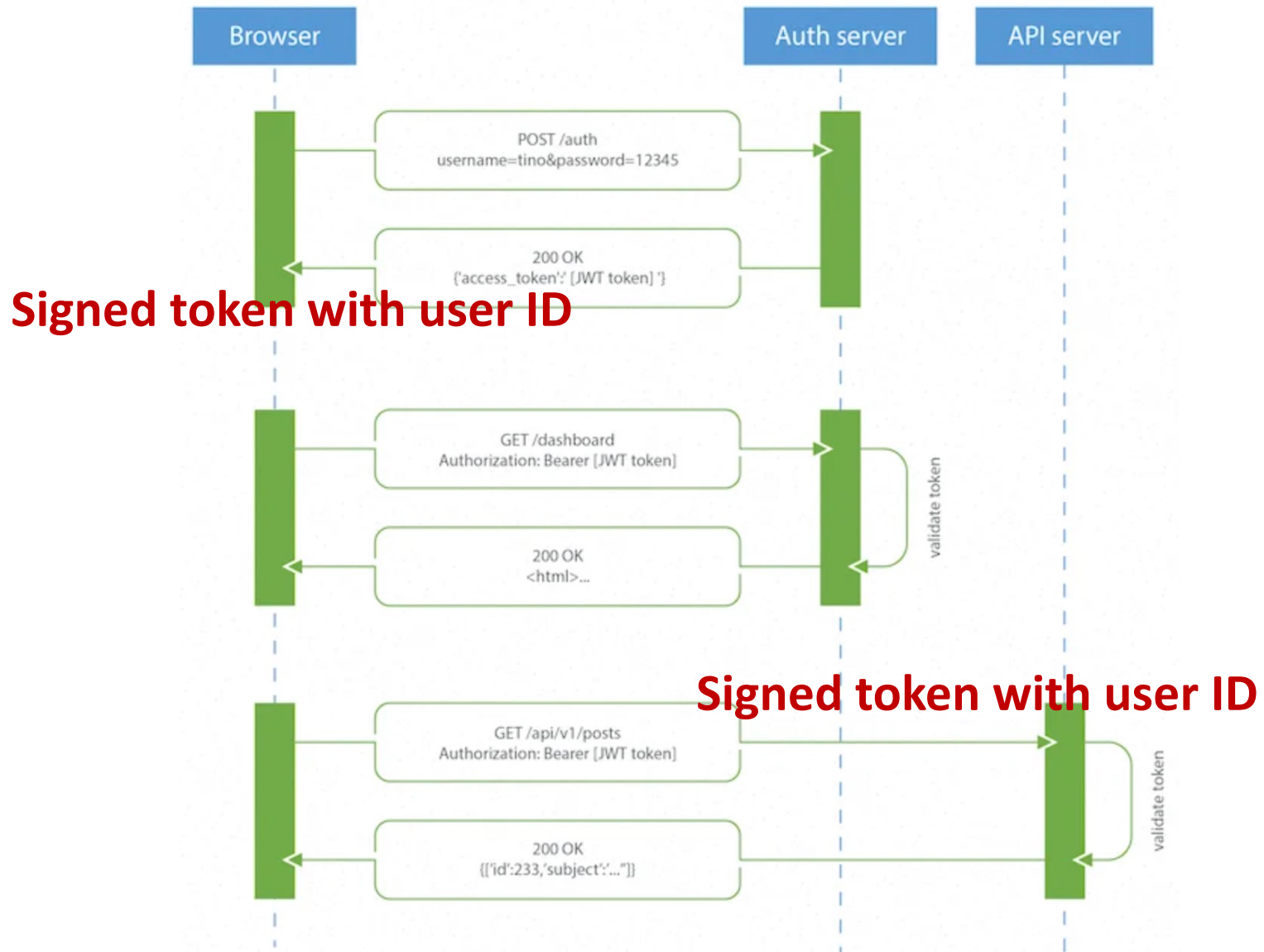
A screenshot of a Facebook post creation interface. At the top is the text 'What's on your mind?'. Below this is a row of icons for adding content: a person (add friend), a location pin, a camera, and a smiley face. To the right of these icons are two buttons: 'Friends' with a dropdown arrow and a 'Post' button. A dropdown menu is open from the 'Friends' button, titled 'Who should see this?'. It lists three options: 'Public' (with a globe icon and the description 'Anyone on or off Facebook'), 'Friends' (with a checkmark, a person icon, and the description 'Your friends on Facebook'), and 'More Options' (with a downward arrow icon).

Session Management

- The process of securely handling multiple requests to a server from a single client (user)



Sessions based on Signed Tokens



JavaScript Web Tokens (JWT)

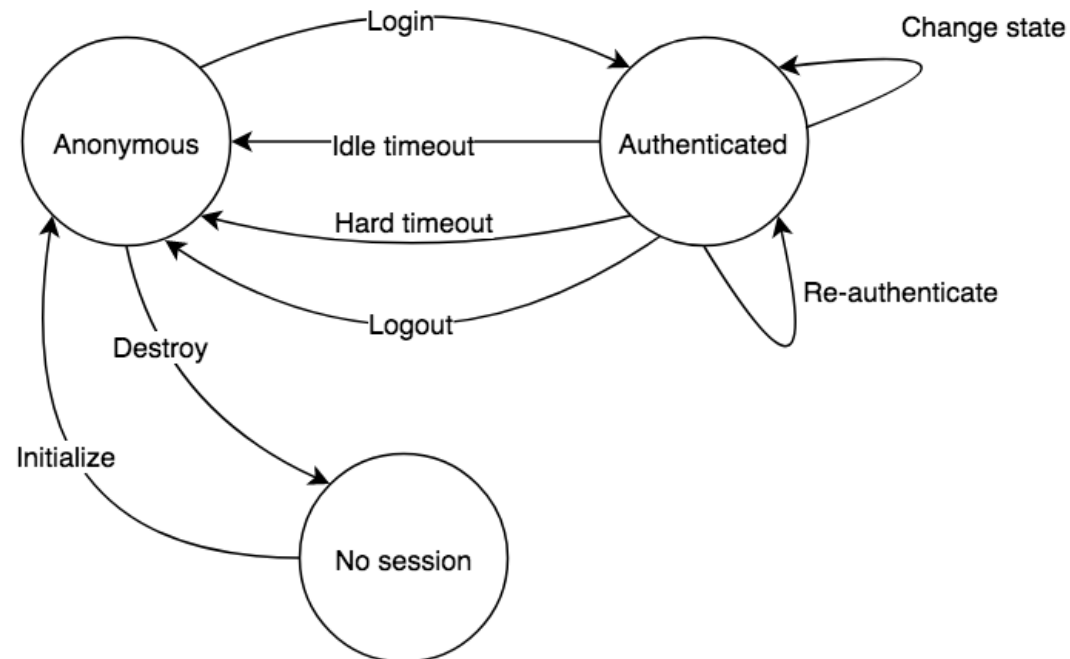
```
// Login response from server
{
  token: e2ZahC5b // JWT token
}
// Subsequent request from client
Authorization: Bearer e2ZahC5b // added by JS
```

- Signed tokens with *self-describing claims*
 - E.g., user ID, expiration date, etc.
- ***Cannot be forged*** due to signatures

(uid, expdate, sha256(uid, expdate, secret))

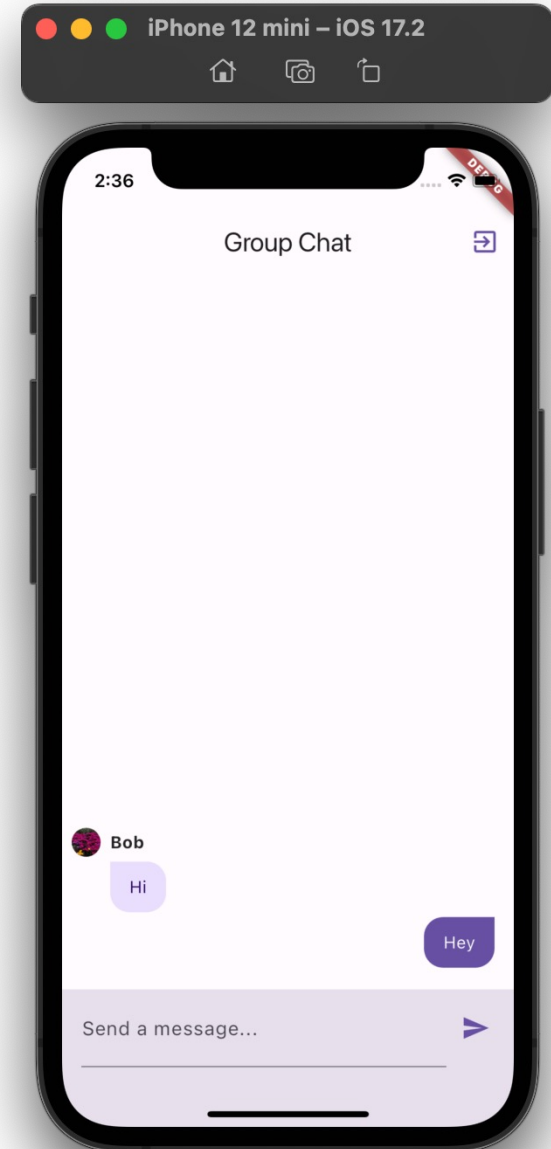
Session Management with Tokens

- Short-lived: **ID tokens**
- Long-lived: **refresh tokens**
 - Saved to secure storage at clients

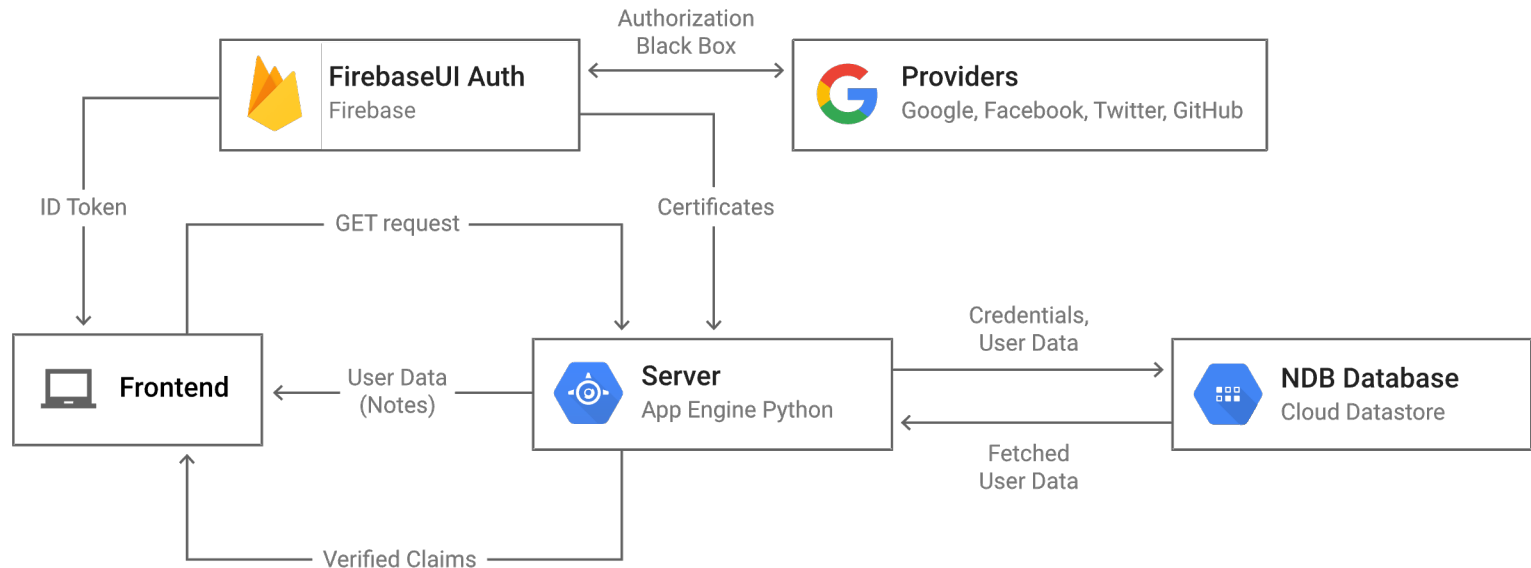


Let's Chat!

- Real-time messaging
- Authentication
 - Sign up & log in
 - Single sign on
- Splash screen
- Image upload
- Security rules
- Custom claims in JWT
- Push notifications



Firestore Email/Password Auth



- Enable it in Firebase Console
- Token-based session management implemented for you in client-side Auth SDK

Sign Up

- Firebase stores the email and a securely *hashed version* of the password in its own database
- . Firebase handles the storage and security of this data, ensuring that passwords are never stored in plain text.

Log In

- Firebase checks the submitted credentials against its database
- If the credentials match, Firebase issues **both** ID token and refresh tokens to the client
- Client-side Auth SDK stores these tokens in **secure** local storage
- When the ID token expires, Auth SDK automatically uses the refresh token to fetch a new ID token

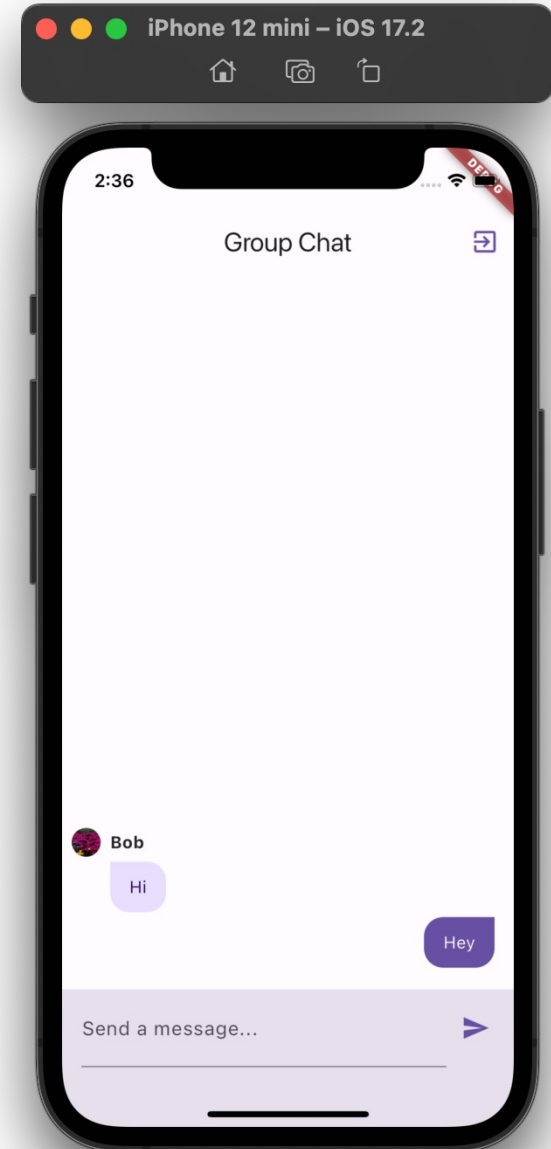
Routing

- In NavigationService:

```
final routerConfig = GoRouter(  
  routes: [...],  
  redirect: (context, state) {  
    // Get the current user  
    final User? currentUser = FirebaseAuth.instance.currentUser;  
    final bool goingToLoginPage = state.location == '/login';  
  
    if (currentUser == null && !goingToLoginPage) {  
      // User is not logged in and trying to access a route  
      return '/login';  
    }  
    // no redirection otherwise  
    return null;  
  }  
);
```

Let's Chat!

- Real-time messaging
- Authentication
 - Sign up & log in
 - Single sign on
- Splash screen
- Image upload
- Security rules
- Custom claims in JWT
- Push notifications



Single Sign-On (SSO)

The image shows a sign-in form with the following elements:

- Sign in** (Title)
- Username** (Input field with a user icon)
- Password** (Input field with a lock icon)
- Sign in** (Blue button)
- ☐ **Remember me**
- [Forgot Password?](#)
- or** (Text separator)
- Login with your social media account** (Text separator)
- f Facebook** (Blue button)
- Twitter** (Light blue button)
- G Google** (Red button)

The social media login section, including the text "Login with your social media account" and the three buttons, is highlighted with a red rectangular border.

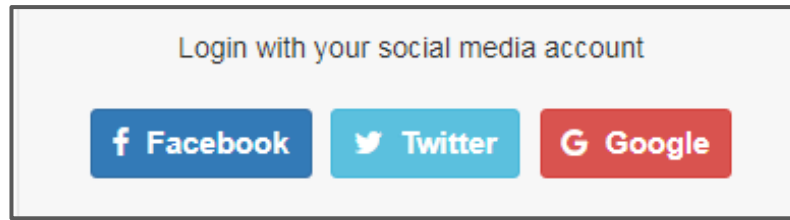
Open ID Connect (OIDC) vs. OAuth



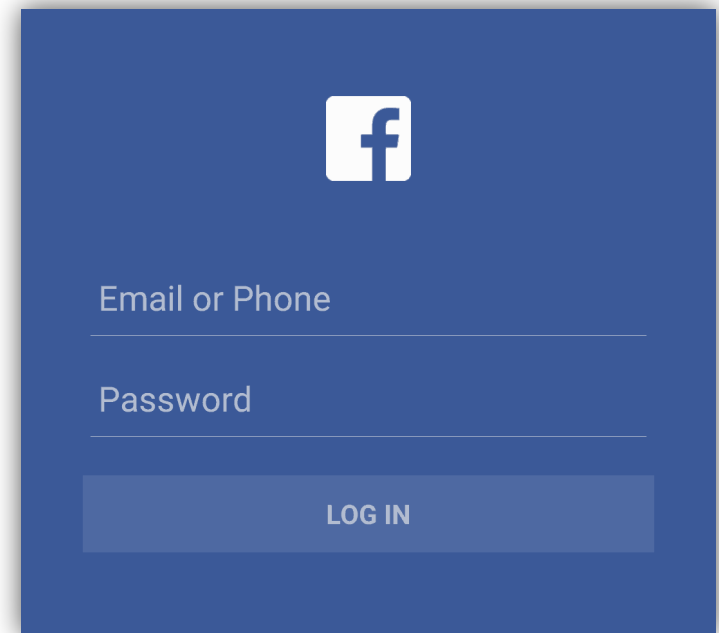
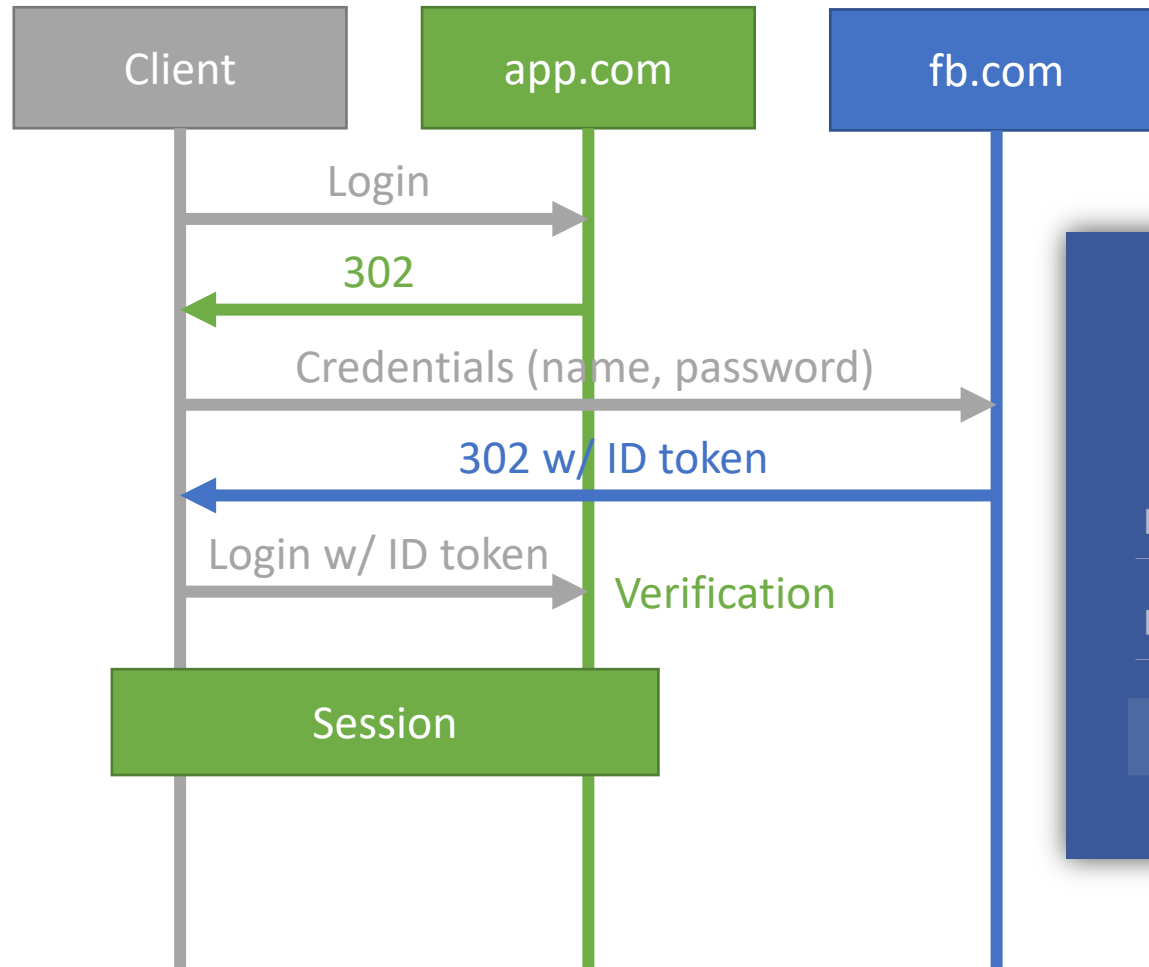
- Authentication

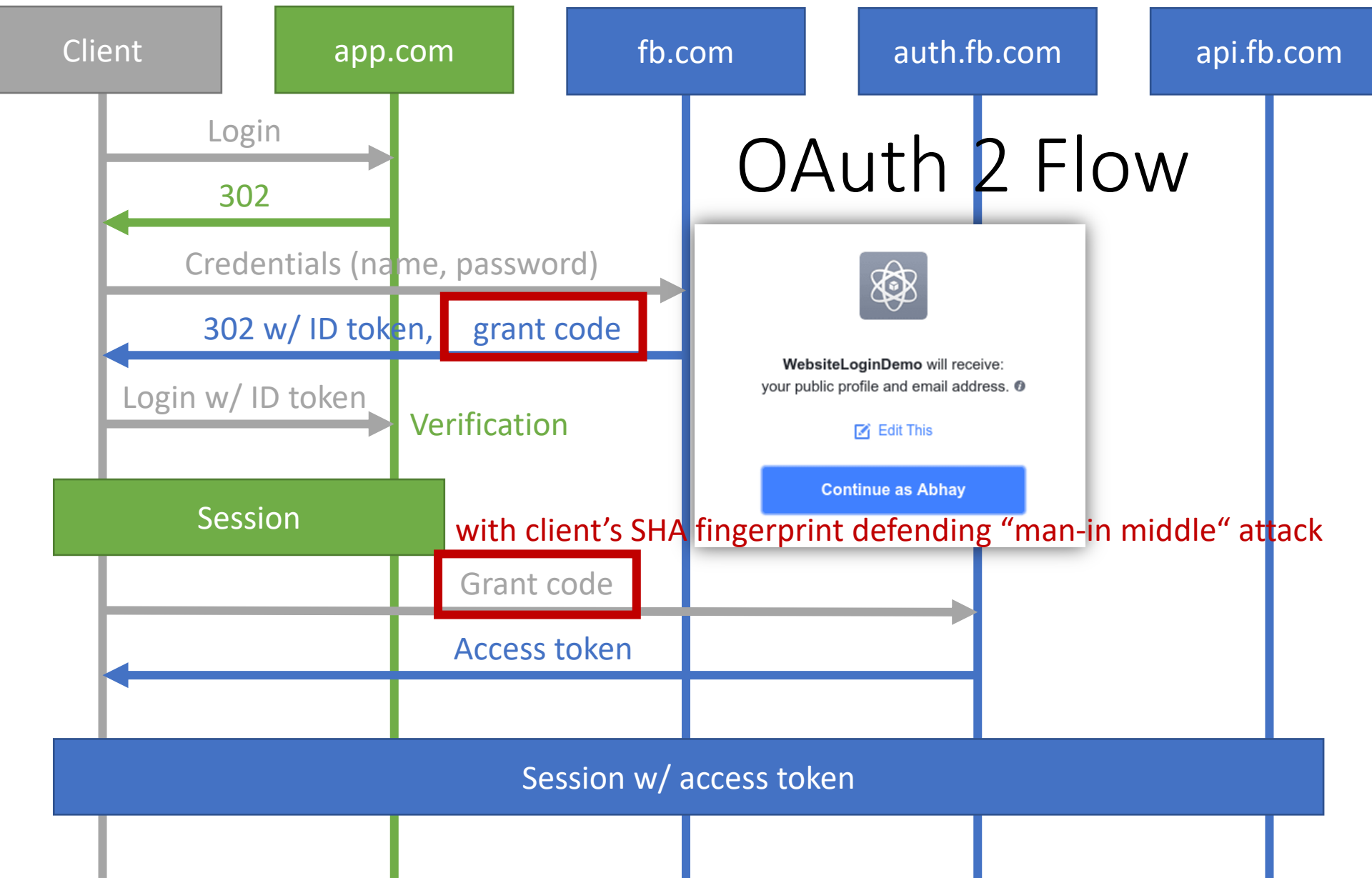


- Authorization



OIDC Flow



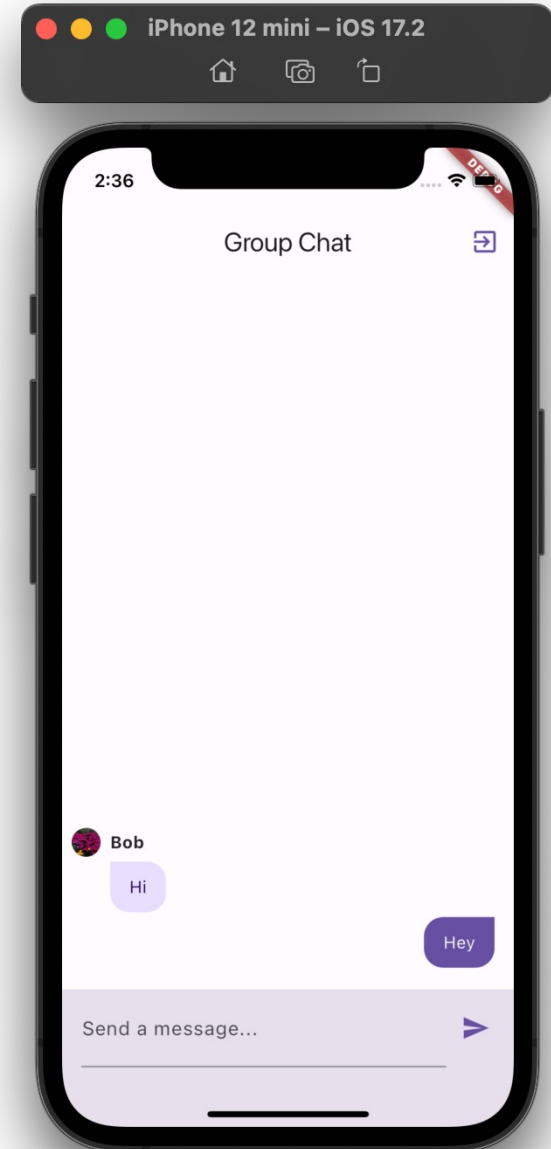


Firebase Sign-in with Google

- Upload your client's SHA fingerprint via Firebase Console
- Firebase creates an account regardless of sign-in methods
- ***Account linking*** for simplifying user management

Let's Chat!

- Real-time messaging
- Authentication
 - Sign up & log in
 - Single sign on
- **Splash screen**
- Image upload
- Security rules
- Custom claims in JWT
- Push notifications



Blinking Home Page

- `FirebaseAuth.instance.currentUser` returns `null` when
 - Firebase Auth is initializing (e.g., loading ID token)
 - User is not logged in
- On slower devices, there's a “blink” before home page shows
 - Due to route switch
- Add a [splash page](#) to avoid this problem
 - Native; need separated generation command:
`dart run flutter_native_splash:create`

Splash Page

- In main():

```
WidgetsBinding widgetsBinding = WidgetsFlutterBinding.ensureInitialized();
FlutterNativeSplash.preserve(widgetsBinding: widgetsBinding);

runApp(StreamBuilder<User?>(
  stream: FirebaseAuth.instance.authStateChanges(),
  builder: (context, snapshot) {
    if (snapshot.connectionState == ConnectionState.waiting) {
      // Keep splash screen until auth state is ready
      return const SizedBox.shrink();
    }

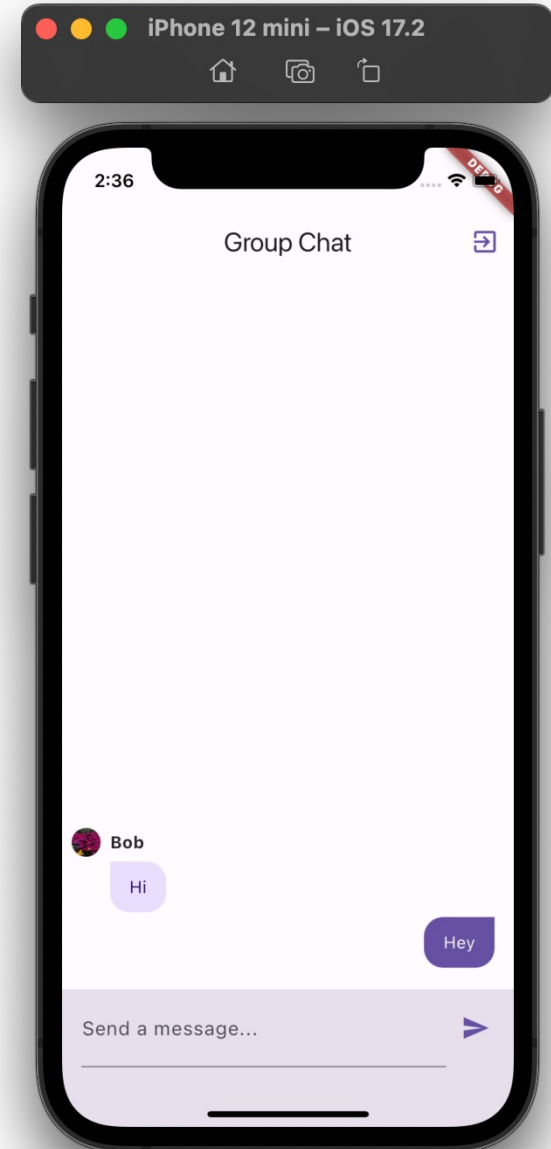
    FlutterNativeSplash.remove();

    // Rebuild MyApp to update the route based on the auth state
    return MyApp();
  },
));
```

- In “/android/app/build.gradle”, change minSdkVersion to 26 or higher

Let's Chat!

- Real-time messaging
- Authentication
 - Sign up & log in
 - Single sign on
- Splash screen
- **Image upload**
- Security rules
- Custom claims in JWT
- Push notifications



Cloud Storage



Cloud Storage
for Firebase

- Stores large files (>1MB)
- Optimized for uploading/downloading large files
- Charges based on data size and network bandwidth
- Limited query capabilities
 - List files in bucket, download by path, get metadata
- No real-time listening

Image Picker

- The cross-platform [image_picker](#) package
 - Configuration needed
- 1. Returns a file
- 2. Upload the image file to Cloud Storage and get image URL
- 3. Save the URL in Firestore
- 4. Use `NetworkImage` to display the image in widgets