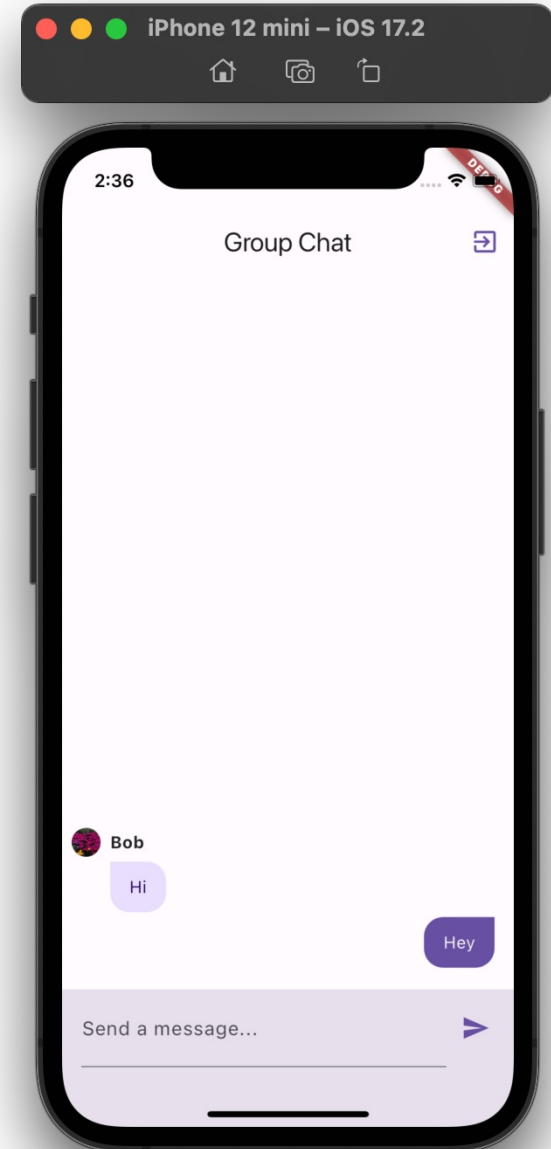


Authentication & Image Upload

Shan-Hung Wu
CS, NTHU

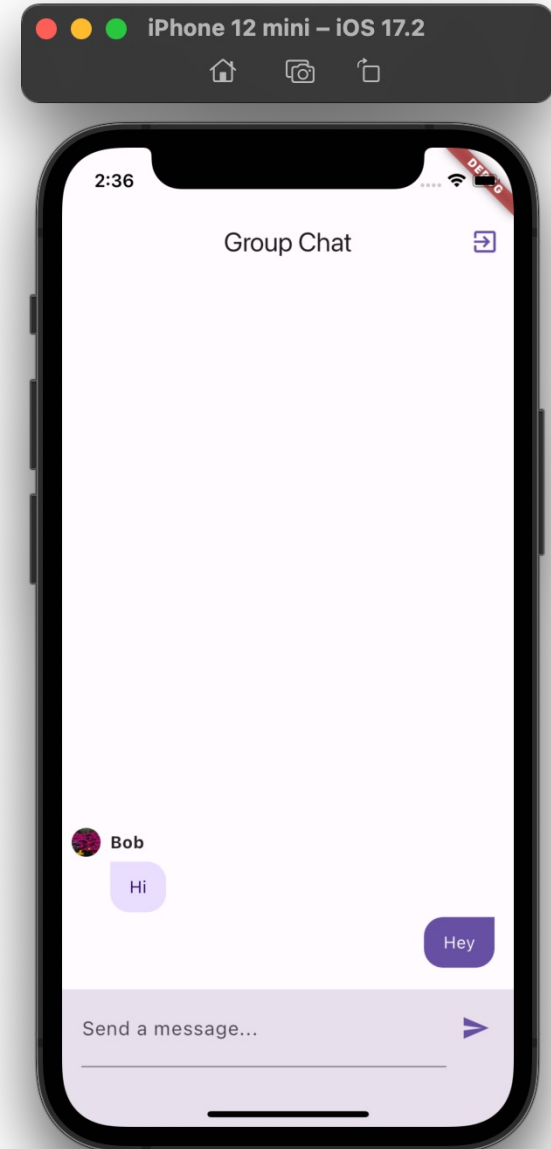
Let's Chat!

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



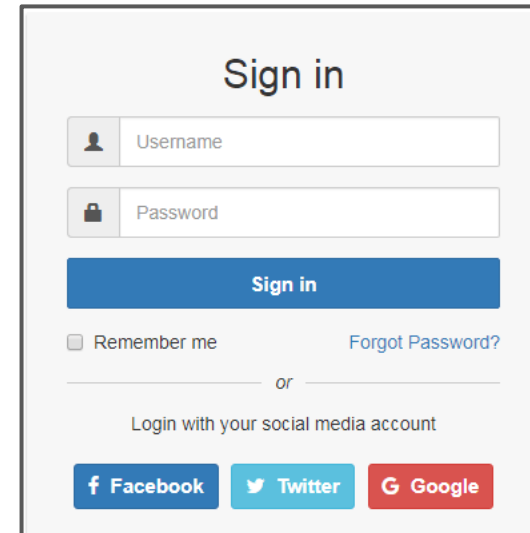
Let's Chat!

- 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



Sign in

Username

Password

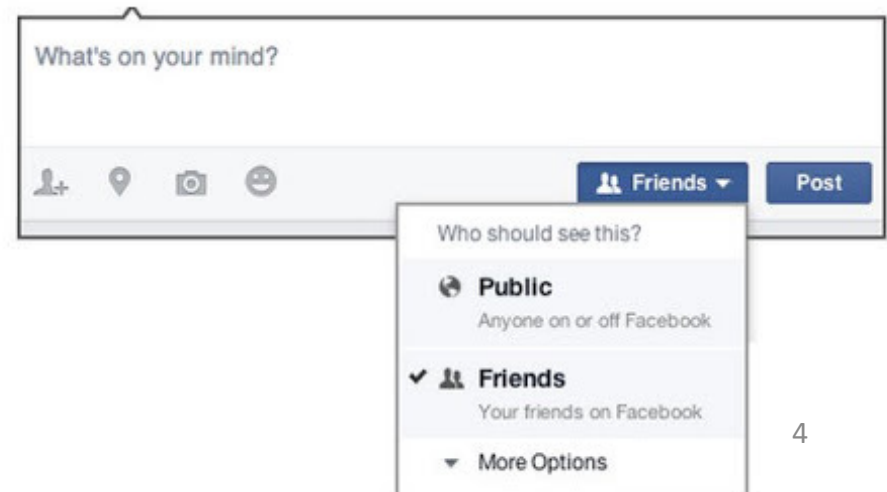
Sign in

Remember me [Forgot Password?](#)

or

Login with your social media account

Facebook Twitter Google



What's on your mind?

Friends Post

Who should see this?

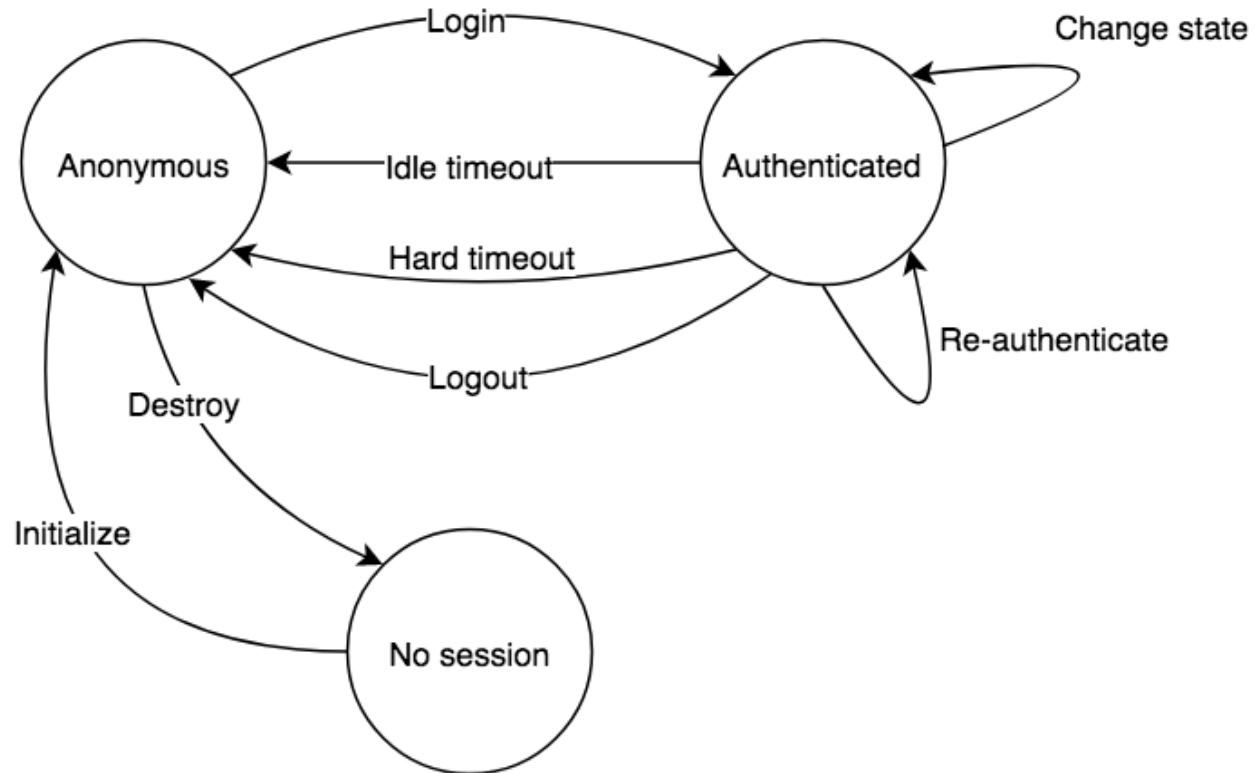
Public
Anyone on or off Facebook

✓ Friends
Your friends on Facebook

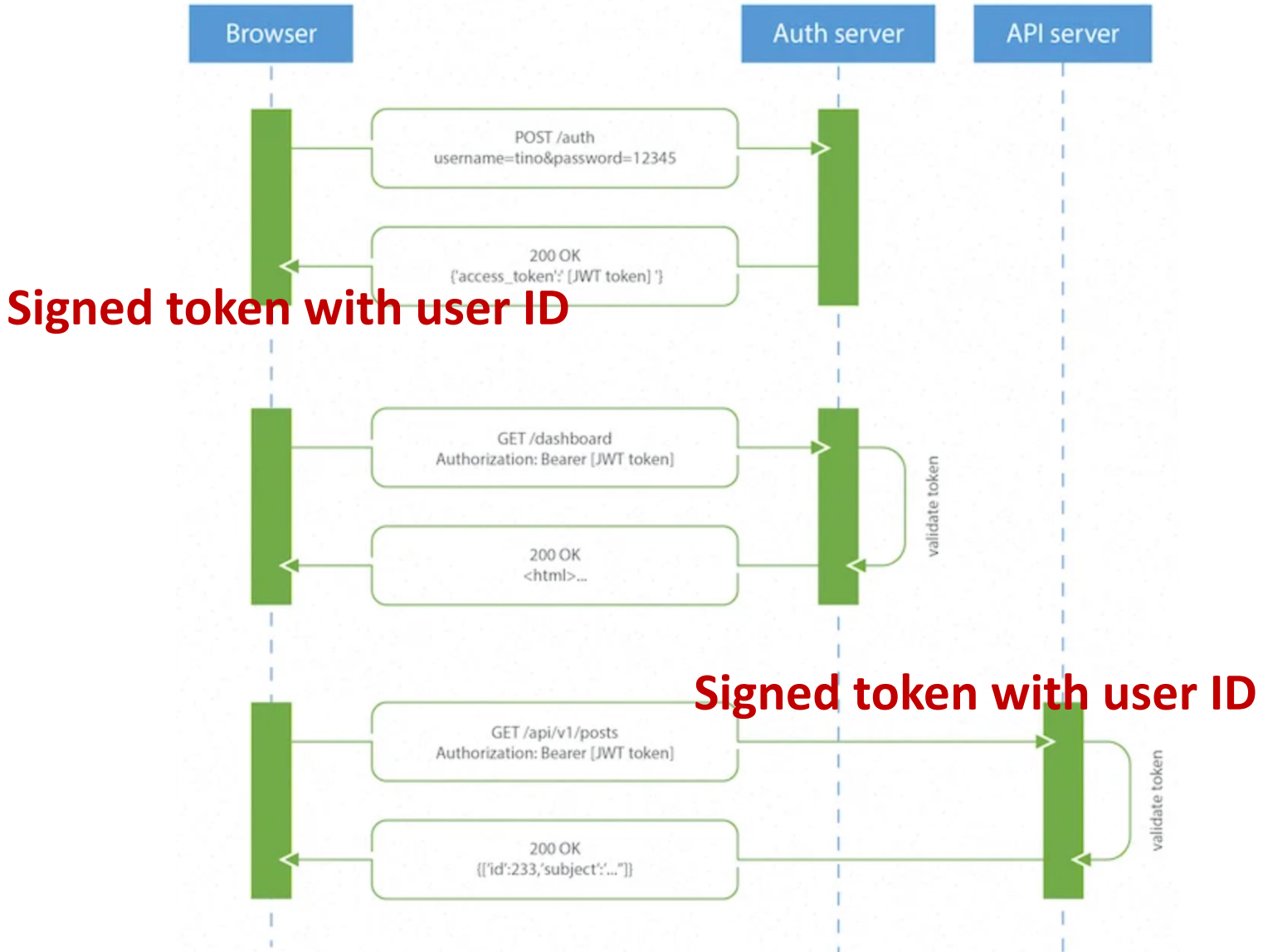
More Options

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

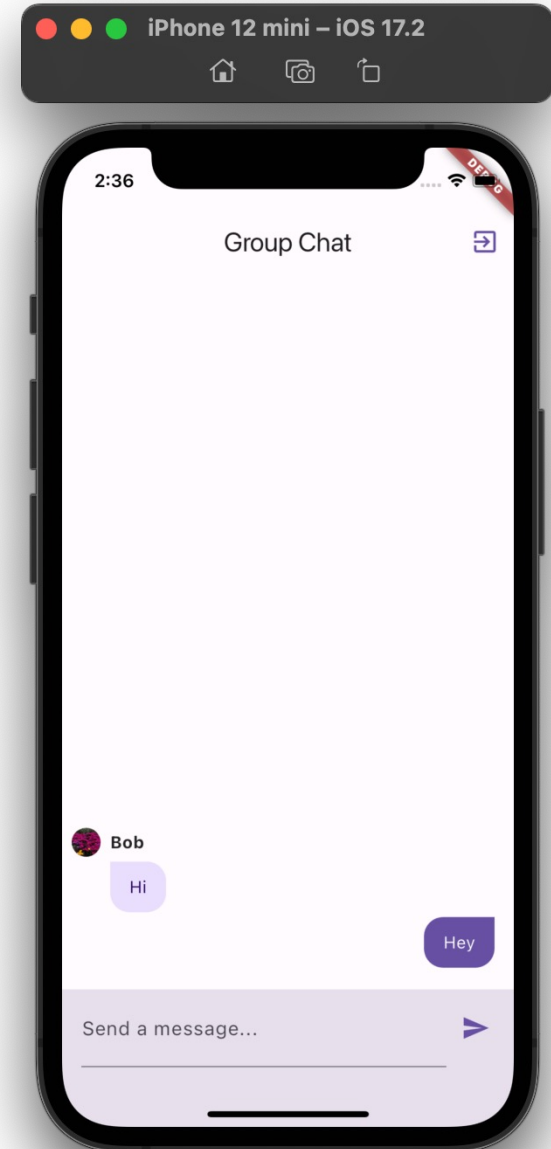
(uid, expdate, sha256(uid, expdate, secret))

Types of Tokens

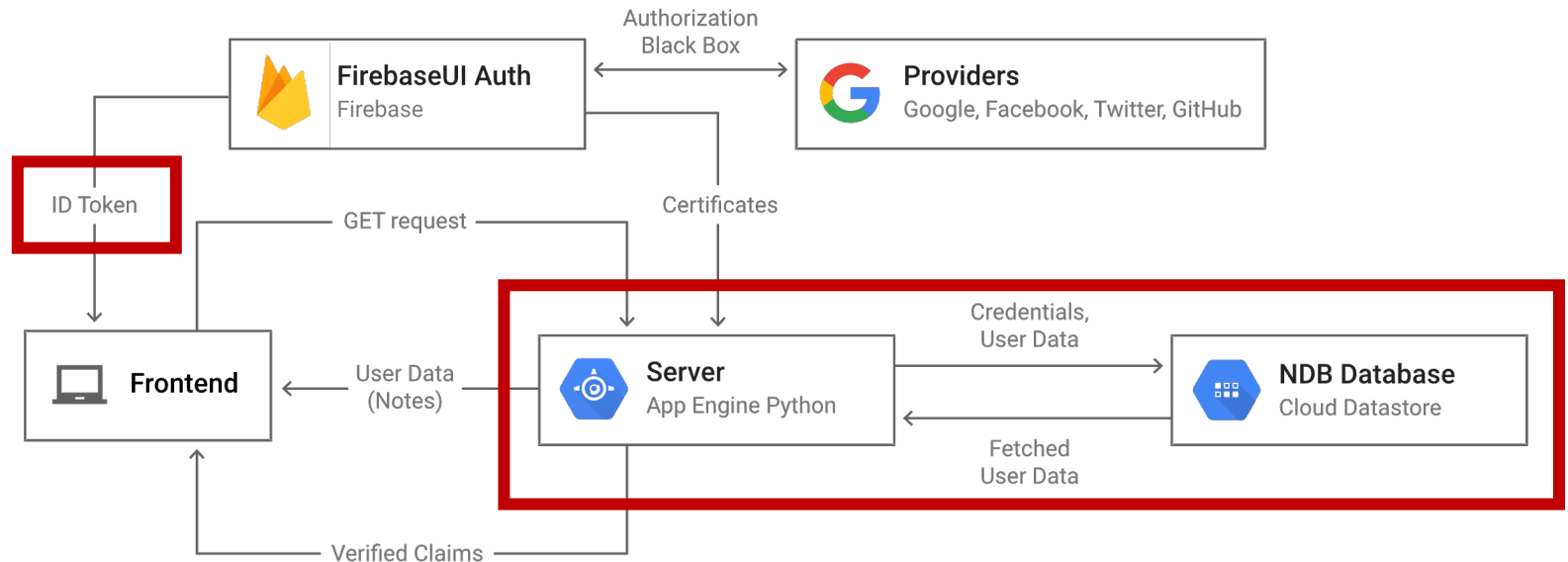
- ***ID token***: identifies a particular user
 - ***Access token***: grants user access to resources
 - Usually short-lived, e.g., a few minutes
 - ***Refresh token***: used to refresh other tokens
 - Usually longer-lived, e.g., tens of days
-
- Need to be saved securely at clients

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



- User DB (credentials only) + token server (ID + refresh)
 1. Enable it in Firebase Console
 2. Use Auth SDK in client

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
-
- See `AuthenticationService.signUp()`

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

- See `AuthenticationService.signIn()`

UI & Routing (1/2)

- In main, a `StreamBuilder` listens to auth state change:

```
runApp(StreamBuilder<User?>(
  stream: FirebaseAuth.instance.authStateChanges(),
  builder: (context, snapshot) {
    if (snapshot.connectionState ==
ConnectionState.waiting) {
      return const SizeBox.shrink();
    }

    // Rebuild MyApp to update the route
    return MyApp(key: ValueKey(snapshot.data == null));
  },
));
```

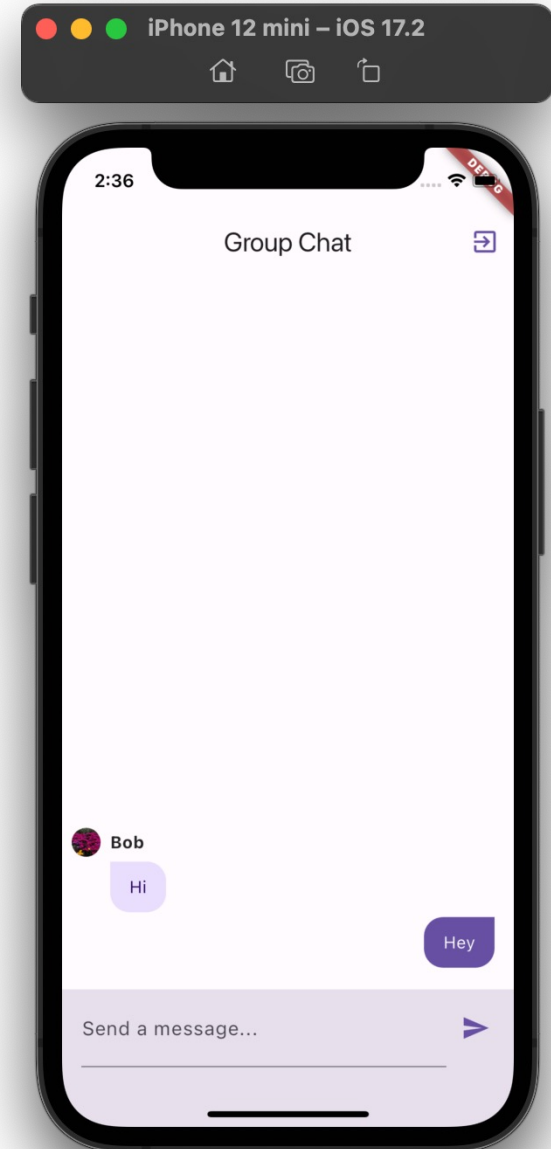
UI & Routing (2/2)

- In `NavigationService`:

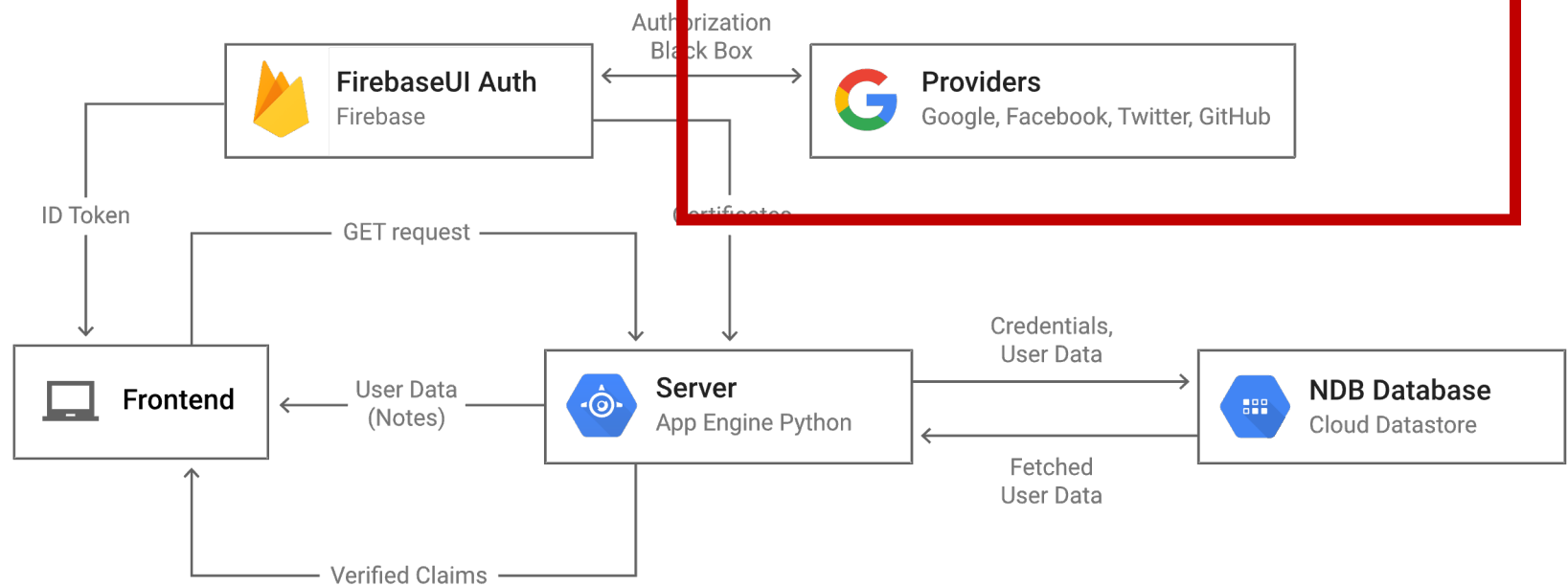
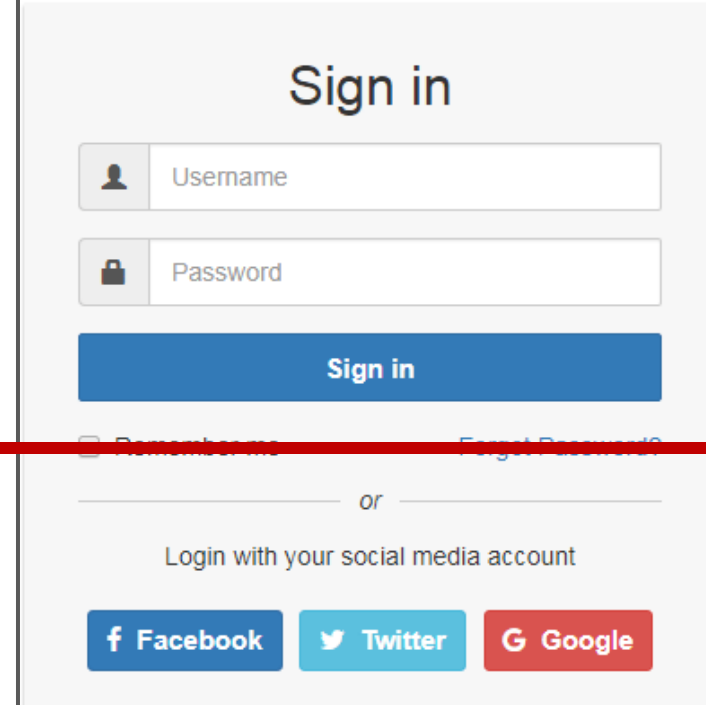
```
final routerConfig = GoRouter(  
  routes: [...],  
  redirect: (context, state) {  
    // Get the current user  
    final User? currentUser = FirebaseAuth.instance.currentUser;  
    final bool goingToLoginPage = state.location == '/login';  
  
    if (currentUser == null && !goingToLoginPage) {  
      // 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)



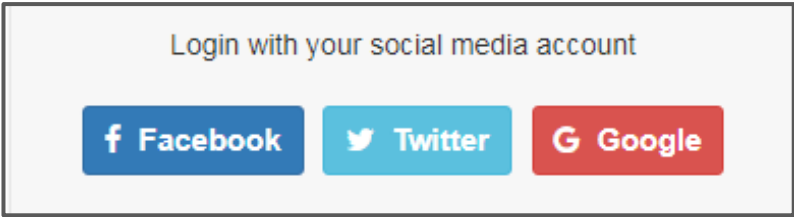
Open ID Connect (OIDC) + OAuth



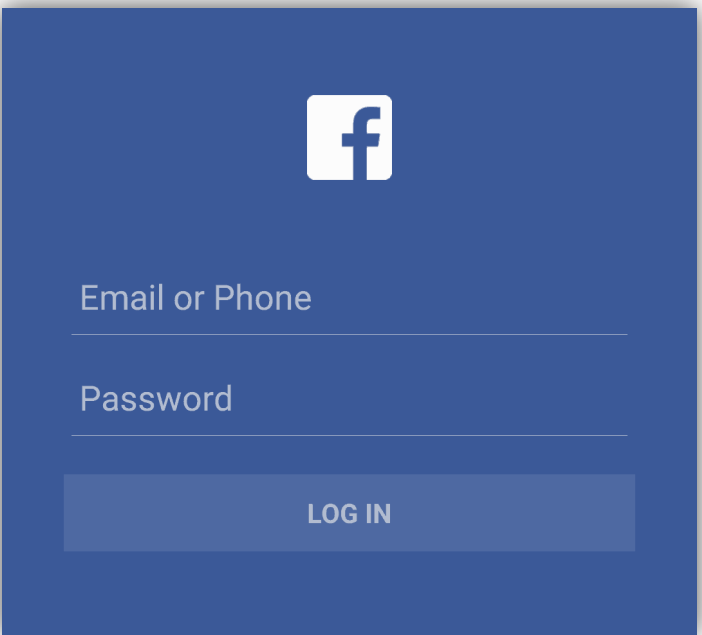
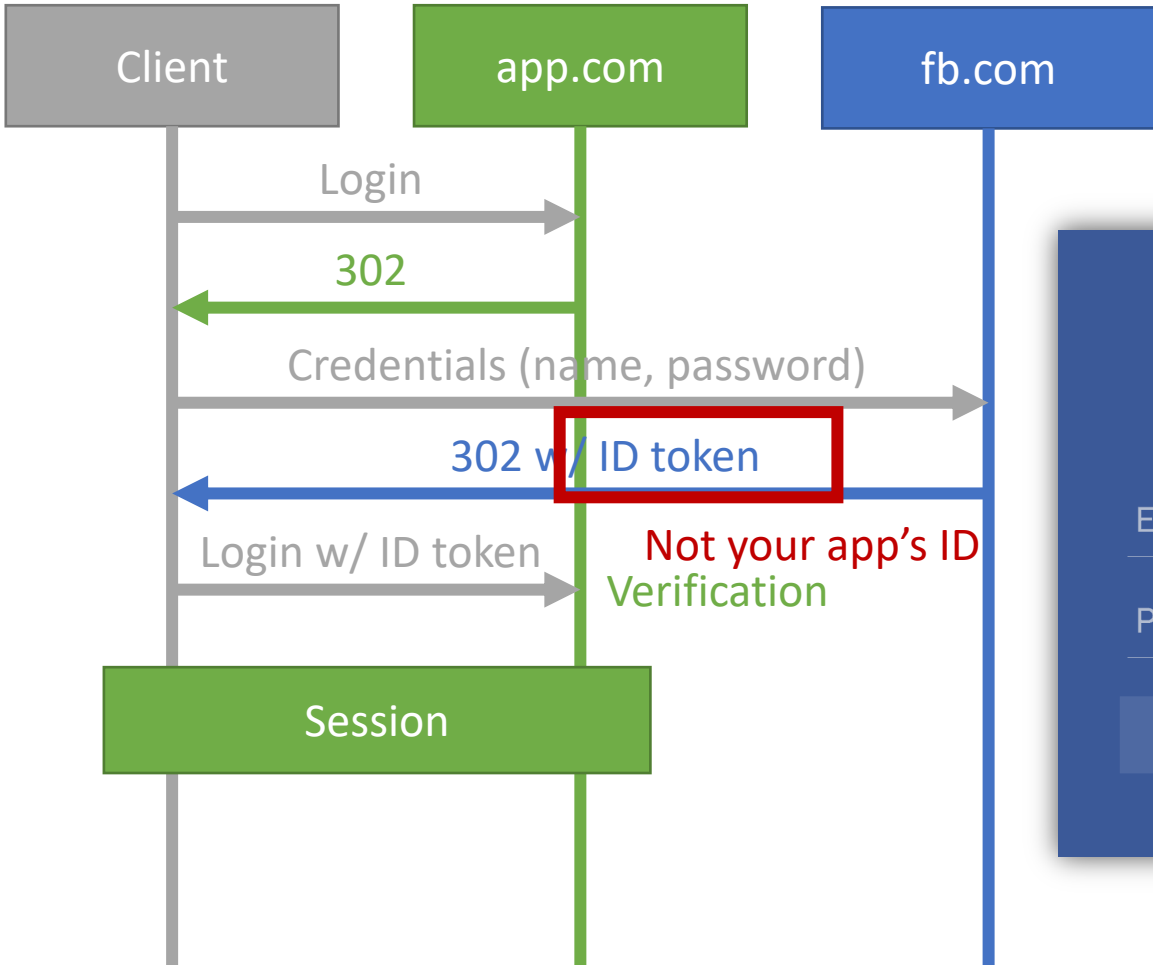
- Authentication

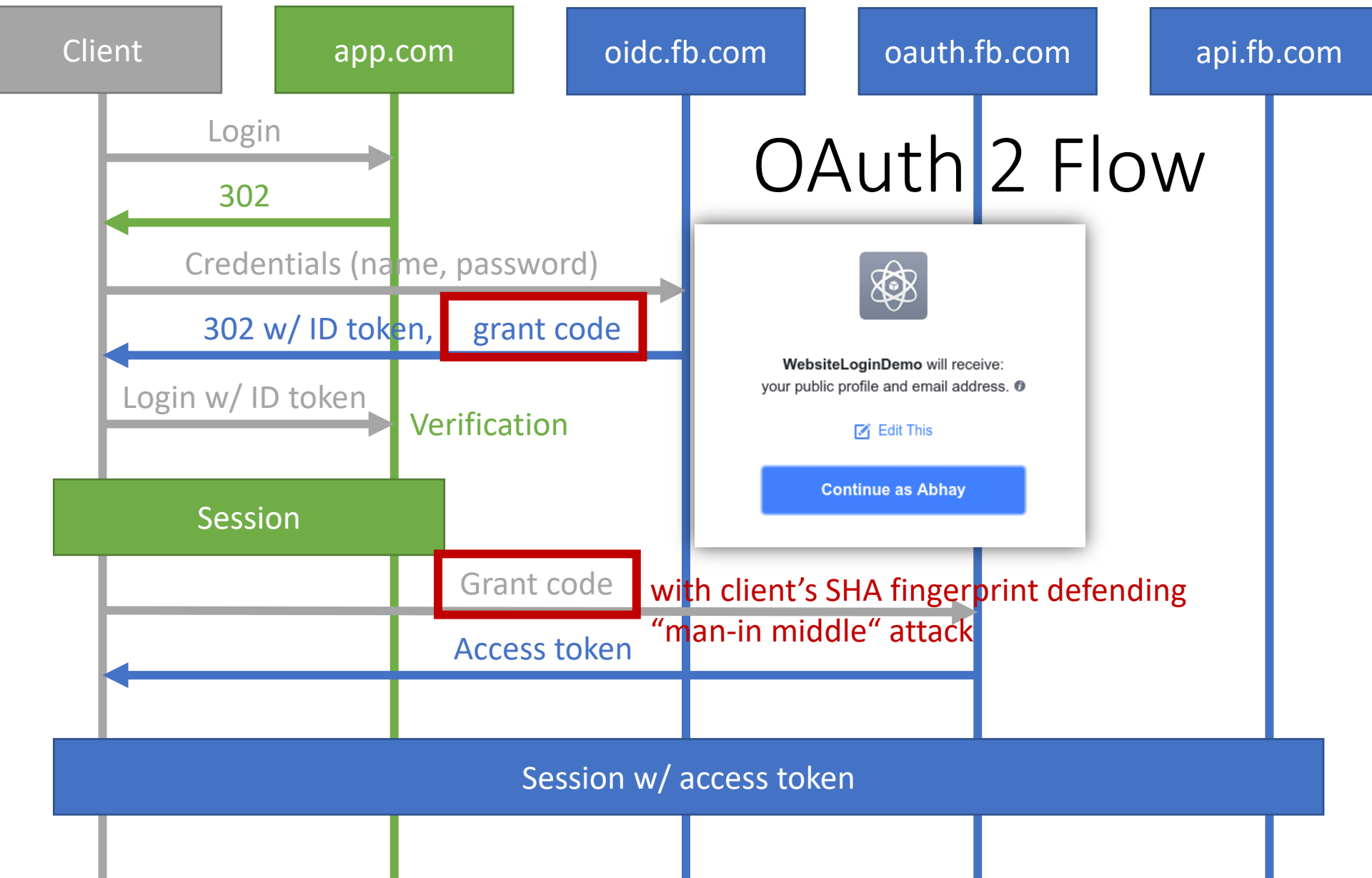


- Authorization



OIDC Flow



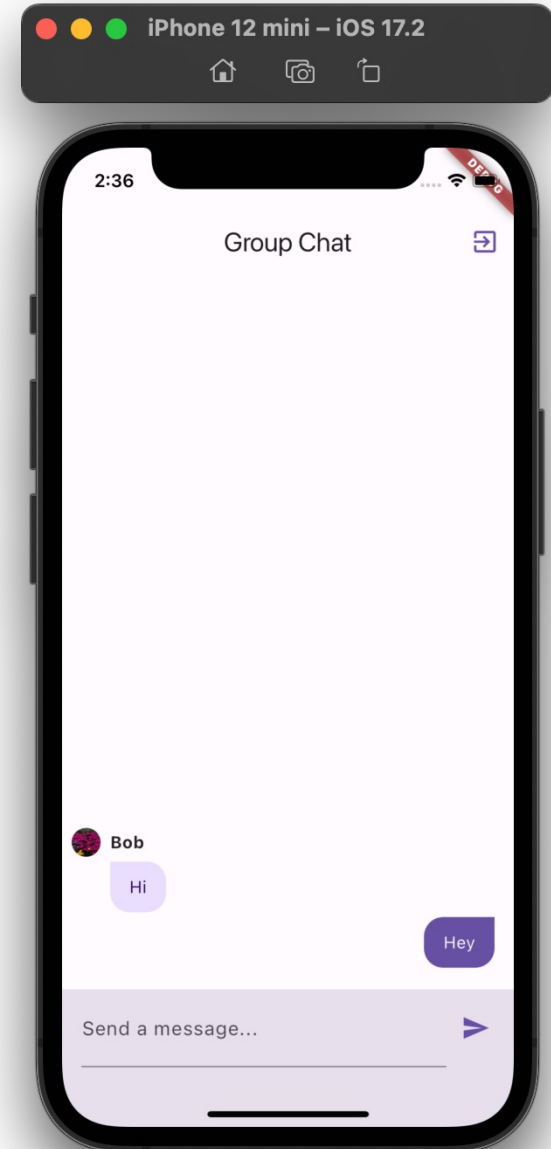


Firebase Sign-in with Google

- Firebase creates an account in its own DB when receiving a new OIDC ID token
- What if you already have an email account using the same email address?
 - Duplicated accounts
 - Complicates account management, e.g., data syncing
- **Account linking** to email account for simplifying user management
- See `AuthenticationService`
`.loginWithGoogle()`

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

- When router redirects, `FirebaseAuth.instance.currentUser` returns `null` when
 - ***Firestore Auth is initializing*** (e.g., loading ID token)
 - User is not logged in
- On slower devices, initialization leads to a “blink” before home page shows
 - Auth page first, then home
- Add a [splash page](#) to avoid this problem
 - Native; need separated generation command:

```
dart run flutter_native_splash:create
```

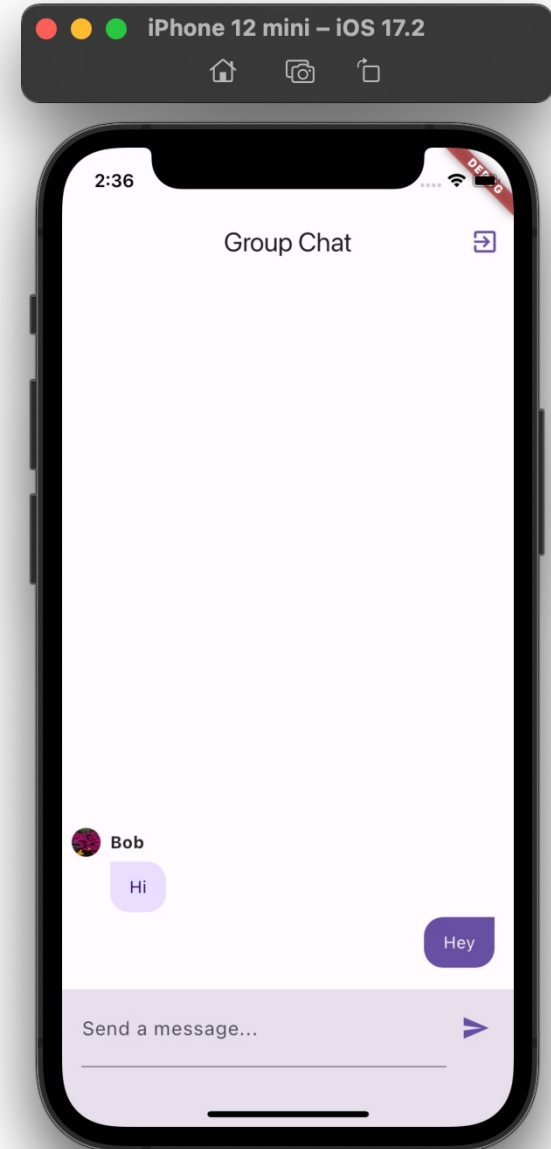
Splash Page

- 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 SizeBox.shrink();  
        }  
  
        FlutterNativeSplash.remove();  
  
        // Rebuild MyApp to update the route based on the auth state  
        return MyApp();  
    },  
));
```

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
- See `AuthPage._submit()` and `authenticationService.signUp()`
 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

References

- [Manage users](#) in Firebase Auth
- [Account linking](#)
- [Anonymous authentication](#)