System Design Document: Chat Application (WhatsApp/Slack Clone)

📌 Project Name

SwiftChat – Real-time iOS Chat Application using SwiftUI and Firebase

🎯 Objective

Create a real-time iOS chat app using SwiftUI, Firebase (Firestore, Authentication, and Storage), and Google Sign-In. The app allows users to authenticate, search users, chat in real time, share media, view typing indicators, and track last seen.

🧱 Architecture Overview

- **Frontend**: SwiftUI (iOS App)
- **Backend**: Firebase (BaaS)
    - Firebase Authentication (Email + Google Sign-In)
    - Firestore (Realtime database)
    - Firebase Storage (Media files)

- **Design Pattern**: MVVM (Model-View-ViewModel)
- **Platform**: iOS 15+

🔐 Authentication

✅ Supported Methods:

- Email/Password Sign In
- Google Sign-In (via Firebase SDK)

🔄 Google Sign-In Flow:

1. User taps “Sign in with Google”
2. App opens Google Sign-In sheet
3. On success, token is exchanged with Firebase
4. Firebase issues a UID and creates user if new

🔧 Firebase SDKs:

- `FirebaseAuth`
- `GoogleSignIn`
- `FirebaseCore`

🧩 Core Features

Feature	Description
✅ Login/Register	Email/Password + Google OAuth
✅ Real-Time Chat	Firestore with live updates
✅ Media Sharing	Upload/download images and videos
✅ Typing Indicator	Detect when user is typing
✅ Last Seen	Update on logout or app exit
✅ Search Users	Search registered users to chat
✅ Profile Edit	Change name and profile picture

📁 Firebase Firestore Structure

🔐 `users` Collection

```
classDiagram
    class User {
        +String uid
        +String name
        +String email
        +String profileImageURL
        +Timestamp lastSeen
        +Boolean isOnline
    }
    %% Firebase User Document Structure
    note for User "Firestore users collection"
```

💬 `conversations` Collection

```
classDiagram
    class Conversation {
        +String conversation_id
        +Array members
        +String lastMessage
        +Timestamp lastUpdated
    }
    class Message {
        +String message_id
        +String senderId
        +String text
        +String mediaURL
        +Timestamp timestamp
        +String type
        +Boolean isRead
    }
    Conversation "1" *-- "many" Message : contains
    %% Conversation structure showing one-to-many relationship with messages
    %% Each conversation can have multiple messages
    %% Messages are stored as a subcollection
```

📦 Firebase Storage Structure

```
graph TD
    A["chat_media"] --> B["conversation_id_1"]
    A --> C["conversation_id_2"]
    A --> D["conversation_id_n"]

    B --> E["message_1.jpg"]
    B --> F["message_2.mp4"]
    B --> G["message_3.jpg"]

    C --> H["message_1.mp4"]
    C --> I["message_2.jpg"]

    D --> J["..."]

    %% Each conversation has its own folder
    %% Messages are stored with unique IDs
    %% Supports both images (.jpg) and videos (.mp4)
```

Used for image/video uploads.

📱 UI Screens

Screen Name	Description
`LoginView`	Sign in with Email or Google
`RegisterView`	Register with Email
`HomeView`	Shows recent chats
`UserSearchView`	Start new chats with other users
`ChatView`	Real-time messaging interface

`ProfileView`	Edit name and profile photo
`SettingsView`	Logout and view app info

🔁 App Flow

1. **Login/Register**:
    - Firebase Auth verifies and creates user.
    - New users added to Firestore `users` collection.
2. **HomeView (Conversations)**:
    - Fetches user's conversations.
    - Realtime listener updates for new messages.
3. **ChatView**:
    - Real-time listener on `messages` subcollection.
    - Shows messages instantly.
    - Upload image → store in Firebase Storage → save mediaURL in Firestore.
4. **Typing Indicator**:
    - Updates a `isTyping` flag in `conversations` document.
    - Other user listens for that flag and displays UI.
5. **Last Seen**:
    - Updated when the app goes to background or on logout using `AppLifecycle`.

🔧 Tech Stack & Dependencies

Component	Tools/Frameworks
UI	SwiftUI
Architecture	MVVM
Auth	FirebaseAuth, GoogleSignIn
DB	Firestore
Storage	Firebase Storage

Image Handling	`SDWebImageSwiftUI` or `Kingfisher`
Media Access	`PhotosUI`, `AVFoundation`
Video Support	Optional, via `AVPlayer`

📈 Scalability Considerations

- Use pagination for messages to avoid long Firestore loads
- Index Firestore fields (e.g. `members`, `lastUpdated`)
- Avoid reading/writing entire documents repeatedly
- Use listeners with care to reduce reads

🧪 Testing

Area	Method
Authentication	Manual + UI Tests
Firestore Access	Unit tests via mocks
Media Uploads	Manual QA
Lifecycle	App background/foreground testing

📝 README Tips for GitHub

- Include **app screenshots or GIFs**
- List features in markdown
- Add “How to Set Up Firebase” section
- Provide full walkthrough of Firebase Console config
- Mention that this app **does not require an Apple Developer account**