Internship Mobile Application Report

1. General Overview

As part of the internship task, I developed a **mobile application using SwiftUI** that integrates with **Firebase** for authentication, video storage, and live streaming.

The requirements stated that the application must contain **exactly 5 pages**:

1. **Login Screen**
2. **Sign-Up Screen**
3. **Video List Screen**
4. **Video Player Screen**
5. **Live Streaming Screen**

In addition, to handle user information, I added an **Account button** on the home screen which opens as a **popup**. This popup displays the logged-in user's email and provides a **Logout button**.

Normally, I wanted to extend this popup into a **Profile Page** (where users could upload/change a profile picture, update username, and change password). However, since the task instructions explicitly required "exactly 5 screens," I kept it as a popup to comply with the rules.

2. Firebase Configuration

The app is connected to **Firebase** services:

- **Firebase Authentication** → For secure login and registration.
- **Firebase Firestore** → To store and retrieve video data and live stream configuration.
- **Firebase Storage** (optional, extendable) → Can be used to store thumbnails or profile pictures.

Firestore Collections

- **videos** collection
    - title: String
    - videoURL: String
    - thumbURL: String
    - createdAt: Timestamp
- **config/live** document
    - isLive: Bool
    - liveURL: String (YouTube live stream link)
    - startedAt: Date
    - thumbURL: String (preview image for live stream)
    - title: String

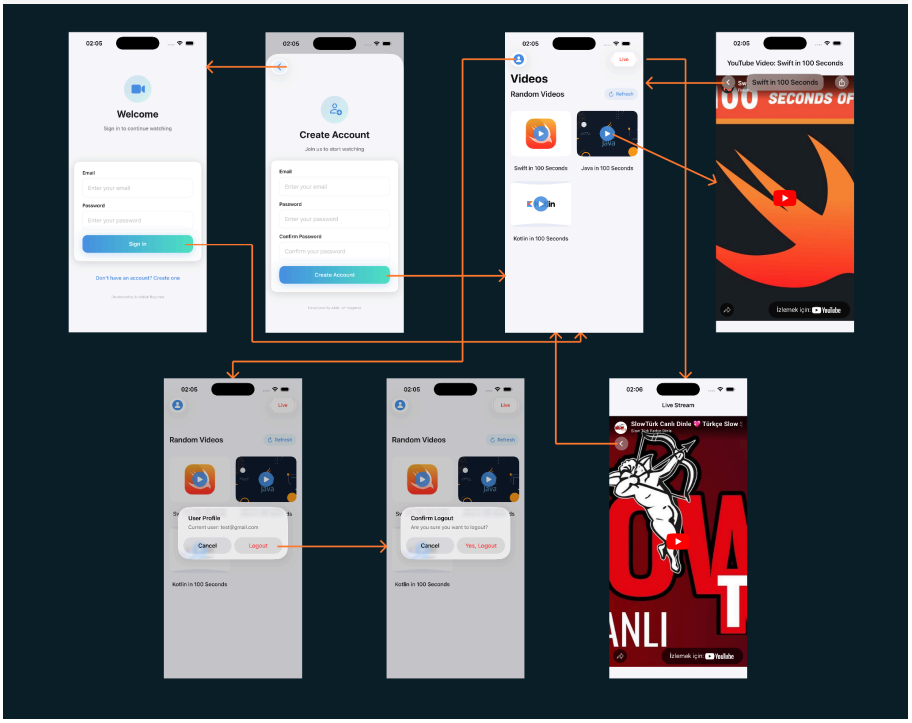
This structure allows me to fetch **3 random videos** from Firestore for the video list screen and dynamically display the **live stream** when isLive == true.

Firestore Rule

```
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {
    match /config/{docId} {
      allow read: if request.auth != null;
      allow write: if false; // yazmayı kapat
    }
    match /videos/{docId} {
      allow read: if request.auth != null;
      allow write: if false;
    }
  }
}
```

3. Application Flow

Screen Flowchart



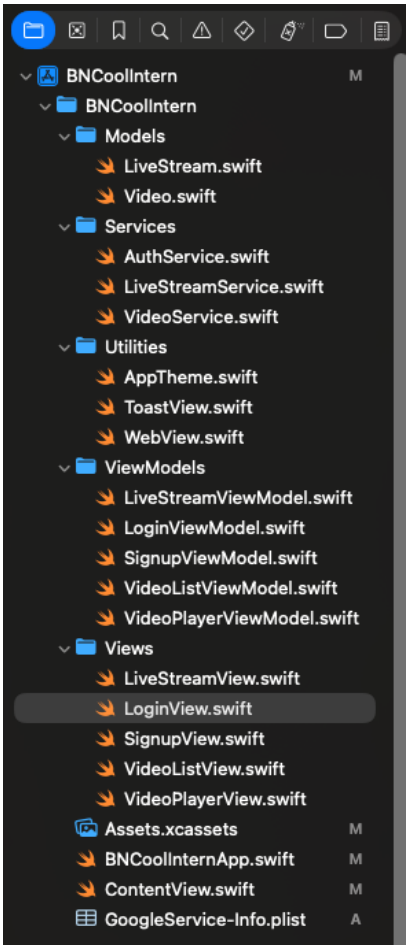
Screen Flowchart

- **Login Screen** → User enters email/password → AuthService authenticates → Redirects to Video List.
- **Sign-Up Screen** → User creates account → Data stored in Firebase Auth → Redirects to Video List.
- **Video List Screen** → Fetches 3 random videos from Firestore. Tapping on one opens the Video Player. The screen also has an **Account button**.
- **Video Player Screen** → Plays the selected video using AVPlayer.
- **Live Streaming Screen** → Embeds YouTube Live stream using WebView → Data fetched from Firestore config.
- **Account Popup** → Displays user email + Logout button.

4. Architecture

The project strictly follows **MVVM (Model – View – ViewModel)** with **Services** and **Utilities** for a clean, scalable architecture.

Folder Structure



Responsibilities

- **Models** → Define Firestore data structures (Video, LiveStream).
- **Services** → Manage Firebase Authentication, Firestore video data, and live stream configs.
- **Utilities** → Handle theme, colors, reusable toast notifications, and embedded web views.
- **ViewModels** → Business logic layer that binds data between services and views.
- **Views** → SwiftUI screens and UI elements, only responsible for presentation.

5. UI & Design

The design uses a **modern and minimalistic style**, with consistent theme colors:

- **Primary** → Blue (main action buttons)
- **Secondary** → Teal (secondary actions)
- **Danger** → Red (logout, destructive actions)
- **TextPrimary** → White/Black depending on mode
- **TextSecondary** → Gray shades for subtitles
- **Background** → Light/Dark depending on system appearance
- **CardBG** → Slightly elevated neutral background for cards

Dark mode equivalents were also defined to ensure accessibility and a professional look across devices.

6. Additional Notes

- **Account Popup**: Instead of a full profile page, the account button opens a popup due to the “5 screens only” restriction.
- **Live Streaming**: Fully dynamic – live status, title, thumbnail, and stream URL are all fetched from Firestore.
- **Scalability**: Thanks to MVVM + Services, new features (e.g., profile management, video upload) can be added easily.

7. Future Improvements

If there were no screen limitations, I would extend the **Account popup into a full Profile Page**, including:

- Upload/change profile picture.
- Splash screen
- Edit username and personal data.
- Change password functionality.
- Manage user preferences (e.g., notifications, theme).
- Offline video caching to allow viewing without internet.
- Firebase Analytics integration (track most viewed videos and active times).

This would provide a more user-friendly experience and make the app closer to a production-ready product.

8. Conclusion

Through this project, I gained hands-on experience with Firebase integration in SwiftUI. By applying the MVVM architecture, I built a scalable and maintainable structure. I also applied modern UI/UX principles, simulating a real product development process. With this foundation, I am confident in developing more advanced, production-level applications in the future.

Abdullah Başpınar
+90 551 343 29 10
aabdullahbaspinarr@gmail.com