Project 2:Flutter Tik Tok Clone App

**Prepared By:**

**Yaakoub Hamad 202106364**

**Cynthia Daou 202104972**

Overview

This report provides an analysis of the Flutter application codebase”Tik Tok Clone”. The application is designed to function as a social media platform, featuring user authentication, video upload and playback, comments, and search functionalities. The codebase is organized into multiple files, including screens, models, controllers, and integration with Firebase services.

SignUp & Login Functionality

In the Flutter application, the signup and login functionalities are implemented in the AuthController along with the associated screens (LoginScreen and HomeScreen). Let's delve into the details of how these features are structured and executed.

1. AuthController (auth\_controller.dart)

* Description:
  + The AuthController is a crucial part of user authentication management in the application.
  + It utilizes Firebase Authentication services for user registration, login, and logout.
  + Manages the user's authentication state and provides functionality for profile picture handling.
* Key Features:
  + Initialization and Authentication State:
    - The controller initializes with the application and binds to the Firebase authentication state changes.
    - The onReady method ensures that the controller is ready to manage authentication state changes.
    - The controller tracks the user's authentication status using the \_user observable.
  + User Registration:
    - The registerUser method takes parameters such as username, email, password, and a profile picture.
    - It creates a new user using Firebase authentication services and uploads the profile picture to Firebase Storage.
    - The user's data is stored in Firebase Firestore under the 'users' collection.
  + User Login:
    - The loginUser method handles user login with email and password.
    - It utilizes Firebase authentication services for user login.
  + User Logout:
    - The signOut method is responsible for logging the user out.
  + Profile Picture Handling:
    - The pickImage method uses the ImagePicker library to allow users to choose a profile picture.
    - The selected image is stored in the \_pickedImage observable.
* Integration with Screens:
  + The AuthController is connected to various screens, including the LoginScreen and HomeScreen.
  + It directs users to the appropriate screen based on their authentication status.

2. LoginScreen (login\_screen.dart)

* Description:
  + The LoginScreen provides the user interface for logging in to the application.
  + It includes input fields for email and password, as well as options for user registration.
* Key Components:
  + Utilizes the AuthController to handle user login and navigation.
  + The TextFormFields capture user input for email and password.
  + Buttons trigger the loginUser method from the AuthController.
  + Navigates to the HomeScreen upon successful login.

3. HomeScreen (home\_screen.dart)

* Description:
  + The HomeScreen serves as the initial screen upon launching the application.
  + It checks the authentication status and directs users to either the login screen or the main content.
* Key Components:
  + The HomeScreen depends on the AuthController to determine the user's authentication status.
  + If the user is authenticated, they are directed to the main content (e.g., VideoScreen); otherwise, they are directed to the LoginScreen.

1. SearchScreen (search\_screen.dart)

* Description:
  + Implements the user search functionality.
  + Displays a search bar in the app bar and a list of users based on search results.
* Key Components:
  + Utilizes the SearchController for managing search-related operations.
  + Integrates with Firebase Firestore for real-time user data updates.
  + Navigates to the user's profile screen upon tapping a search result.

2. VideoScreen (video\_screen.dart)

* Description:
  + Represents the main screen for video playback and interaction.
  + Displays a vertically scrolling list of videos with user information and engagement options.
* Key Components:
  + Uses the VideoController to fetch and manage the list of videos.
  + Implements video playback, user engagement buttons (like, comment), and navigation to comment screens.
  + Integrates with Firebase Firestore for real-time video updates.

3. CommentScreen (comment\_screen.dart)

* Description:
  + Displays comments related to a specific video.
  + Allows users to read and interact with comments, including liking and replying.
* Key Components:
  + Relies on the CommentController for managing comments.
  + Provides options for liking comments and navigating to user profiles.

File: confirm\_screen.dart

Description:

The ConfirmScreen is responsible for confirming and processing the selected video. It displays the selected video with options for providing additional information like the song name and caption before sharing it.

Key Components:

1. VideoPlayer Widget:
   * Uses the VideoPlayer widget to play the selected video.
2. TextInputField Widgets:
   * Uses custom TextInputField widgets for entering the song name and caption.
3. uploadVideoController Instance:
   * Uses an instance of the UploadVideoController to handle the video upload process.
4. build Method:
   * Constructs the UI with the video player, song name input field, caption input field, and a "Share" button.
   * Invokes the uploadVideoController.uploadVideo method when the "Share" button is pressed.

File: add\_video\_screen.dart

Description:

The AddVideoScreen is responsible for allowing users to add a new video to the application. It provides a simple interface with a button to trigger the video selection process.

Key Components:

1. pickVideo Method:
   * Takes an ImageSource (camera or gallery) and a BuildContext.
   * Uses the ImagePicker library to allow users to choose a video.
   * Navigates to the ConfirmScreen with the selected video file.
2. showOptionsDialog Method:
   * Shows a dialog with options for selecting a video from the gallery or capturing it using the camera.
   * Invokes the pickVideo method based on the user's selection.
3. build Method:
   * Constructs the UI with a button labeled "Add Video."
   * When tapped, it opens the options dialog for video selection.

File: confirm\_screen.dart

Description:

The ConfirmScreen is responsible for confirming and processing the selected video. It displays the selected video with options for providing additional information like the song name and caption before sharing it.

Key Components:

1. VideoPlayer Widget:
   * Uses the VideoPlayer widget to play the selected video.
2. TextInputField Widgets:
   * Uses custom TextInputField widgets for entering the song name and caption.
3. uploadVideoController Instance:
   * Uses an instance of the UploadVideoController to handle the video upload process.
4. build Method:
   * Constructs the UI with the video player, song name input field, caption input field, and a "Share" button.
   * Invokes the uploadVideoController.uploadVideo method when the "Share" button is pressed.

File: profile\_screen.dart

Description:

The ProfileScreen displays the user's profile information, including their profile picture, follower/following counts, and a grid of their uploaded video thumbnails.

Key Components:

1. ProfileController Instance:
   * Uses an instance of the ProfileController to manage and update user profile information.
2. Profile Information Display:
   * Displays the user's profile picture, follower/following counts, and a button for following/unfollowing or signing out.
3. Grid of Video Thumbnails:
   * Utilizes a GridView.builder to display the user's uploaded video thumbnails.
4. build Method:
   * Constructs the UI with various sections for displaying profile information and the grid of video thumbnails.
   * Utilizes CachedNetworkImage for efficient image loading.

Models

1. Comment Model (comment.dart)

* Description:
  + Represents comment-related data, including username, comment text, date, likes, and user details.
* Key Features:
  + Provides a method (fromJson) for converting Firestore document snapshots to Comment objects.
  + Defines a toJson method for converting Comment objects to Firestore-compatible JSON.

2. User Model (user.dart)

* Description:
  + Represents user-related data, including name, email, UID, and profile photo.
* Key Features:
  + Provides methods (fromJson and toJson) for converting Firestore document snapshots to and from User objects.

3. Video Model (video.dart)

* Description:
  + Represents video-related data, including username, UID, video ID, likes, comment count, share count, song name, caption, video URL, thumbnail, and profile photo.
* Key Features:
  + Offers methods (fromJson and toJson) for converting Firestore document snapshots to and from Video objects.

Controllers

1. AuthController (auth\_controller.dart)

* Description:
  + Manages user authentication, including registration, login, logout, and profile picture handling.
* Key Features:
  + Uses Firebase services for authentication and Firestore for storing user data.
  + Handles navigation based on the user's authentication status.
  + Enables profile picture selection and upload.

2. CommentController (comment\_controller.dart)

* Description:
  + Manages comments for videos, including retrieval, posting, liking, and updating comment counts.
* Key Features:
  + Utilizes Firebase Firestore for real-time comment updates.
  + Provides methods for posting comments, liking comments, and updating comment counts.

3. SearchController (search\_controller.dart)

* Description:
  + Handles user search functionality, including retrieving and displaying search results.
* Key Features:
  + Utilizes Firebase Firestore to search for users based on name.
  + Updates the UI in real-time with search results.

4. UploadVideoController (upload\_video\_controller.dart)

* Description:
  + Manages the uploading of videos, including video compression, storage, and updating video data.
* Key Features:
  + Uses Video Compress library for video compression.
  + Integrates with Firebase Storage for video and thumbnail uploads.
  + Updates Firestore with video-related data.

5. VideoController (video\_controller.dart)

* Description:
  + Handles video-related functionalities, including fetching videos, liking videos, and updating video data.
* Key Features:
  + Utilizes Firebase Firestore for real-time updates on video-related data.
  + Provides methods for liking videos.

Firebase Integration (firebase.dart)

* Description:
  + Integrates the application with Firebase services, including Authentication, Firestore, and Storage.
* Key Features:
  + Configures and initializes Firebase services.
  + Manages authentication status, user data, and Firestore data updates.

Conclusion

The Flutter application codebase demonstrates a well-organized and modular structure, making use of various screens, models, controllers, and Firebase integration to deliver a seamless social media experience. The use of fire Store updates from Firebase enhances the responsiveness of the application, ensuring users receive the latest information. Overall, the codebase exhibits best practices in Flutter app development, promoting readability, maintainability, and scalability.