PROJECT REPORT

>>

Report on
Game Recommendation App

Hurain-21sw122 , Malook - 21sw116

INTRODUCTION:

The "Game Recommend App" addresses the growing need for streamlined game discovery by offering tailored recommendations for platforms like PS5 and Xbox. Built in Flutter, this mobile app ensures a responsive, cross-platform experience and uses Firebase for secure user authentication, cloud data storage, and image handling. By allowing users to sign up, log in, and share games with descriptions and images, the app creates a centralized and engaging resource for gaming enthusiasts. This report outlines the design, development, and problem-solving strategies involved in creating a robust, user-friendly application that meets Complex Engineering Problem (CEP) standards.

Real-World Problem Identification

Objective:

The project aims to create a mobile application called "Game Recommend App" to help users discover games suitable for specific platforms like PS5 or Xbox, based on their preferences.

Problem Analysis: With the rapid growth of gaming platforms, users often struggle to find relevant recommendations. The app addresses this issue by centralizing game information in a single, easy-to-navigate space, streamlining the discovery process. The app's use of Firebase allows it to offer a personalized experience through authentication and save user data securely in the cloud.

Target Users: Gamers who want reliable, platform-specific recommendations, as well as individuals who might enjoy discovering new games based on user uploads.

Proposed Solution

Core Functionality:

User Authentication: Allows users to register, log in, and securely manage their accounts using Firebase Authentication.
Game Uploads: Users can submit games for others to view, including details like the game's name, platform (e.g., PS5, Xbox), and a brief description. Additionally, they can upload an image to enhance visual appeal.
Platform-Specific Categorization: The app organizes games by platform, making it easier for users to filter and find relevant games based on their devices.
Expected User Flow:
Users are greeted with an authentication page (authPage) that checks if they are logged in. New users can sign up via SignUpScreen, while returning users log in through LoginScreen.
Once authenticated, users navigate to the UploadGameScreen to add games, with platform selection and image upload options available.

Technical Specifications

Framework: Flutter

- Backend: Firebase for Data Storage
- Languages: Dart (for Flutter)
- Architecture: MVVM (Model-View-ViewModel) for code organization and scalability
- State Management: Provider package for managing the app state efficiently

Data Storage

Cloud Database:

Firebase Firestore stores game data in collections, organized by platform (ps5, xbox). This structure allows efficient querying and categorization of games.
Database Justification: Firestore's real-time synchronization capabilities make it ideal for apps with frequently updated data. It also allows offline persistence, meaning users can still see cached data even without an internet connection.
Image Storage: Firebase Storage handles image files for each game uploaded by users, creating a game_images folder that stores images with unique paths based on game names.
Storage Justification: Firebase Storage integrates seamlessly with Firestore and allows controlled, secure file access. It also provides auto-scaling, ensuring reliability as more images are uploaded.

APIs/Packages/Plug-ins Used

Firebase Core:

Initializes Firebase across different platforms, ensuring unified app behavior.
Firebase Auth: Manages authentication, allowing secure user registration, login, and error handling for various scenarios.
Firebase Firestore: Provides a structured, scalable database to store user-submitted games, organized for platform-based querying.
Firebase Storage: Manages image uploads and retrieval, with functionality to generate public URLs for image display in the app.
Image Picker: Allows users to select images from their device gallery, enhancing the app's usability by letting users add visuals to their game entries.

Issues and Bugs Encountered and Resolved

Image Selection Null Check:

Adding a null check in _pickImage() method to avoid app crashes when a user cancels image selection. This improves the user experience by handling unexpected actions gracefully.

Error Handling in Authentication:

Sign-Up Errors: Error cases like “email already in use” or “weak password” are managed in SignUpScreen using FirebaseAuthException catch blocks. The app displays descriptive messages via SnackBar so users know exactly why an error occurred.

Login Errors: For cases like “user not found” or “wrong password,” the app displays appropriate feedback, reducing user frustration and guiding them toward corrective actions.

Data Validation:

The app enforces required fields, ensuring users don’t leave critical information blank (e.g., game name, description). Password validation ensures a minimum length for better security.

Field Matching: Password confirmation fields ensure the two password entries match, preventing user errors during registration

CONCLUSION

In conclusion, the "Game Recommend App" successfully addresses the challenge of game discovery by providing a centralized platform for sharing and finding game recommendations across specific consoles. Using Flutter and Firebase, the app achieves a responsive, cross-platform design with secure data handling. Through structured implementation, robust error handling, and a user-friendly interface, the project fulfills its objectives and demonstrates the complexity and problem-solving skills required by Complex Engineering Problem (CEP) standards. This app not only simplifies game exploration for users but also showcases effective mobile app development practices.