PYCO - Final Report

Akemi Sai

December 17, 2024

Abstract

PYCO is an innovative social platform designed to help users organize their wardrobe, request outfit suggestions, and engage with a community of fashion enthusiasts. This report details the app's functionality, technical implementation, challenges encountered, and any modifications or omissions made to the original project proposal.

Contents

1 Project Overview

PYCO is a social platform that allows users to create an inventory of their clothing items, request personalized outfit suggestions, and interact with other users through responses and likes. Users can create requests for outfit ideas based on specific occasions or items they want to pair, and others in the community can respond with outfit suggestions. The app offers a dynamic interface, supporting the discovery of new styles and the exploration of various fashion trends.

The platform's core functionalities include:

- Clothing inventory management
- Outfit request creation
- Outfit suggestion responses from the community
- User profiles with likes, followers, and outfit bookmarks
- Social features such as liking outfits and following users

2 Installation

To run the PYCO app, follow these steps:

1. Clone the repository:

   ```
   git clone https://github.com/yourusername/PYCO.git
   ```

2. Navigate to the project directory:

   ```
   cd PYCO
   ```

3. Install dependencies:

   ```
   npm install
   ```

4. Run the application:

   ```
   npm start
   ```

Ensure that you have the necessary environment set up, including a Firebase account for backend services.

3 Tech Stack

The app is built with the following technologies:

- **Frontend:** React (TypeScript), Styled Components
- **Backend:** Firebase Firestore, Firebase Authentication
- **State Management:** React Context API, React Hooks
- **UI Development:** Jetpack Compose
- **Authentication:** Firebase Authentication
- **Navigation:** Jetpack Compose Navigation
- **Database:** Firebase Firestore
- **Image Handling:** CameraX, Image Capture, and Background Removal using Remove.bg
- **Deployment:** Firebase Hosting
- **Version Control:** Git, GitHub

4 Key Features

PYCO provides the following key features:

- **Clothing Inventory:** Users can add, view, and organize clothing items with attributes like type, color, material, and tags.
- **Outfit Requests:** Users can create outfit requests based on specific items or events and receive suggestions from others.
- **Outfit Suggestions:** Fashion enthusiasts can respond to requests with outfits composed of clothing items from the user's inventory.
- **User Profiles:** Each user has a profile with personal information, liked outfits, followers, and following counts.
- **Social Features:** Users can like outfits, follow others, and bookmark outfits for later reference.

5 APIs and Libraries Used

The following APIs and libraries were utilized in the project:

- **Firebase Firestore:** Used for storing user profiles, clothing items, outfit requests, and responses.
- **Firebase Authentication:** Provides secure user sign-up and sign-in.

- **Remove.bg API:** Used for background removal from images.
- **CameraX:** Used for capturing images through the app's camera.
- **Image Capture API:** Facilitates image capturing from the device's camera.
- **Styled Components:** Used for styling the app's components.
- **React Router:** Manages navigation between pages.
- **React Hooks:** Used for managing state and side-effects within the app.

6 Schema Overview

The database schema for PYCO is structured as follows:

- **ClothingItem:** Stores individual clothing items with attributes such as type, material, color, and tags.
- **Outfit:** Represents an outfit created by a user, consisting of references to `ClothingItem` documents.
- **Request:** Represents an outfit request created by a user with a description, title, and associated tags.
- **Response:** Stores the responses to requests with an outfit suggestion and any additional comments.
- **User:** Stores user profile information, including bookmarked outfits, followers, following, and activity data.

7 Database Structure

The Firestore database is organized into collections:

- **users:** Contains user profiles.
- **clothingItems:** Contains individual clothing items.
- **outfits:** Contains user-created outfits.
- **requests:** Contains user-created requests.
- **responses:** Contains responses to requests.

8 Folder Structure

The folder structure of the project is as follows:

- **src/**: Contains all source code for the application.
  - **components/**: React components for UI elements.
  - **pages/**: React components representing different app pages.
  - **context/**: React Context API for managing global state.
  - **services/**: Firebase-related services (authentication, database interactions).
- **public/**: Contains public assets like images and `index.html`.
- **package.json**: Contains project dependencies and scripts.

9 Challenges

The following challenges were encountered during the development of the PYCO app:

9.1 Debugging & Logging

Debugging and logging were challenging due to the large number of asynchronous operations, such as Firestore interactions, CameraX, and background threads. It was difficult to track data and pinpoint issues when ViewModels were interacting extensively with each other. A lot of time was spent identifying the right places to log and carefully managing the flow of data.

9.2 Handling Uri and CameraX

Managing URIs and integrating CameraX was complicated. Storing and fetching images, especially from the camera, was time-consuming as we had to figure out the proper caching mechanisms. Ensuring that images were correctly stored and accessible across app sessions involved multiple iterations of testing and debugging.

9.3 Push Notifications

Implementing push notifications was difficult because we were uncertain about where to place certain functions, particularly with Firebase Authentication not being guaranteed at the time of `MainActivity.kt`. Figuring out the correct order of operations and how to handle notifications triggered by background tasks took considerable effort.

9.4 Databases and ViewModels

Throughout the development process, the database schema underwent several changes due to trial and error. Adjusting the Firestore structure to optimize data retrieval and updates, while keeping the ViewModels efficient, was a constant challenge.

10 Lessons Learned

The following lessons were learned during the development of PYCO:

10.1 API Integration

We improved our understanding of API integration, especially managing asynchronous operations with callbacks. The process of connecting various services like Firebase, CameraX, and push notifications helped us better manage APIs and their responses.

10.2 Collaboration

Working with a team meant dealing with frequent conflicting merges, especially when multiple team members were working on the same files. We improved our communication and planning skills, which helped mitigate issues related to version control and streamline the development process.

10.3 Databases

10.3.1 Debugging Firestore

Understanding the Firestore structure and authentication workflows required some time, as we had to learn how to correctly manage real-time updates and interactions with nested collections. Debugging and modifying the Firestore structure involved a lot of trial and error to achieve the desired performance.

10.3.2 Gradle and LogCat

We had to familiarize ourselves with handling errors in Gradle files, which were common throughout the project. Additionally, LogCat became an essential tool for tracking and debugging issues, and writing detailed logs helped immensely in identifying and fixing problems.

11 Features Modified or Omitted

Several modifications were made to the initial project proposal:

- **Modified Feature:** The "Outfit Sharing" feature was initially planned to allow users to share outfits externally. Due to time constraints, this feature was omitted, but the app allows for sharing outfits within the platform.
- **Omitted Feature:** The app was initially planned to include a social media-like feed for posts related to fashion. This feature was removed for simplicity and to focus on core functionality.

12 Future Enhancements

Future versions of PYCO may include:

- **Improved Search:** Adding advanced search functionality to filter clothing items by multiple attributes such as type, color, and material.
- **Outfit Suggestions via AI:** Implementing AI-driven outfit suggestions based on user preferences, weather, or specific occasions.
- **Social Media Integration:** Allowing users to share outfits and posts on other social media platforms.

13 License