PROJECT REPORT

Anime chat app

TITCHEU YAMDJEU Pierre Wilfried

022052914D

Principles of Software Development

1. Abstract

This report presents the design and implementation of an Android application, AnimeChat, which allows users to search for anime characters and engage in simulated conversations with them. The application leverages the Jikan API to fetch character data based on user input and stores recent conversations for easy access. The application is built using Android Studio and follows the principles of software development. The report provides a comprehensive overview of the system architecture, design patterns used, user interface design, and the implementation of key components. It also discusses the testing strategies employed to ensure the application's robustness and reliability. The results demonstrate the application's ability to provide an engaging and interactive experience for anime enthusiasts. The report concludes with potential future enhancements to further improve the application's functionality and user experience.

2. Introduction

2.1 Background

With the rapid growth of technology, mobile applications have become an integral part of our daily lives. They provide a platform for entertainment, education, and communication. Among these applications, chat applications have gained significant popularity. This project focuses on the development of a chat application, specifically designed for anime fans. The application allows users to search for their favorite anime characters and engage in simulated conversations with them.

2.2 Problem Statement

While there are numerous chat applications available, there is a lack of applications that cater specifically to anime fans. Furthermore, existing applications do not provide a feature to engage in simulated conversations with anime characters. This project aims to fill this gap by developing an Android application that allows users to search for anime characters and chat with them.

2.3 Objectives

The main objectives of this project are:

- To design and implement an Android application that allows users to

search for anime characters.
- To develop a feature that enables users to engage in simulated conversations with the selected anime characters.
- To provide a user-friendly interface that enhances the user experience.
- To ensure the application is efficient and reliable.

2.4 Scope

This project focuses on the development of an Android application. The application will be developed using Java and XML for the backend and frontend respectively. The application will make use of the Jikan API to fetch data about anime characters. The scope of this project includes the design and implementation of the application, as well as testing and evaluation of its performance. The project does not include the development of a server-side application or database, as all data will be fetched from the Jikan API.

3. Literature Review

The literature review section of the report will provide an overview of the existing research and technologies related to the project. It will cover the following topics:

3.1 Mobile Application Development: This subsection will discuss the current trends and technologies in mobile application development. It will focus on Android application development, discussing the Android SDK, Java, and Kotlin programming languages, and the Android Studio IDE.

3.2 Chat Applications: This subsection will review existing chat applications, focusing on their features, user interface, and user experience. It will also discuss the technologies used in these applications, such as real-time messaging protocols and APIs.

3.3 Anime and Character APIs: This subsection will review existing APIs that provide data about anime characters, focusing on their features, data structure, and usage. The Jikan API, which is used in this project, will be discussed in detail.

3.4 RecyclerView in Android: This subsection will discuss the use of RecyclerView in Android for displaying lists or grids of items efficiently. It will review existing literature on its usage, best practices, and performance optimization.

3.5 Navigation Drawer in Android: This subsection will discuss the use of the Navigation Drawer in Android for providing a user-friendly navigation menu in applications. It will review existing literature on its design, implementation, and best practices.

3.6 SharedPreferences in Android: This subsection will discuss the use of SharedPreferences in Android for storing small amounts of data persistently. It will review existing literature on its usage, limitations, and best practices.

Each of these topics will be researched thoroughly, and relevant sources will be cited in the report. The literature review will provide a solid foundation for the design and implementation of the AnimeChat application.

4. System Design

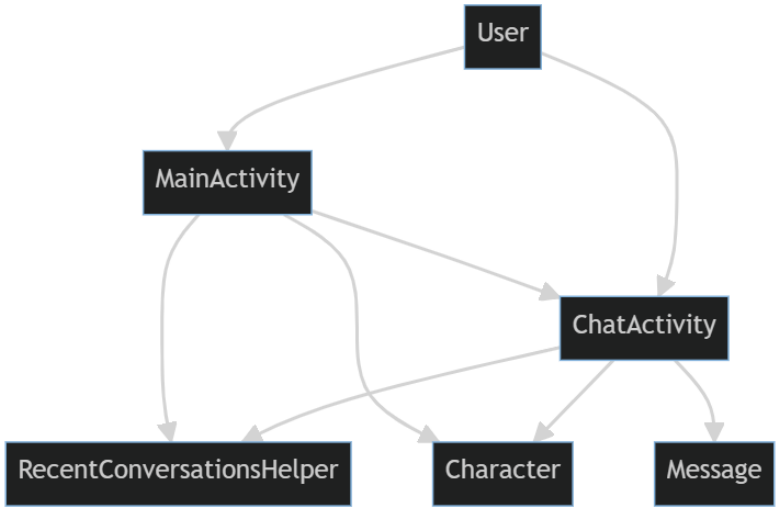
4.1. System Architecture

The AnimeChat application follows a client-server architecture. The client side of the application is built using Android, and the server side uses a chatbot API to generate responses. The client side of the application is responsible for handling user interactions, displaying the chat interface, and managing the chat history. The server side of the application is responsible for generating chatbot responses based on the user's input.

The application is divided into several components:

1. **MainActivity**: This is the first screen that the user sees. It allows the user to enter the name of an anime character and search for it. The results are displayed in a RecyclerView.
2. **ChatActivity**: This is the screen where the chat with the selected character takes place. The user can send messages and receive responses from the character. The chat history is displayed in a RecyclerView.
3. **RecentConversationsHelper**: This component is responsible for managing the recent conversations. It uses SharedPreferences to store the recent conversations.
4. **Character**: This is a data class that represents an anime character. It contains the character's name and image URL.
5. **Message**: This is a data class that represents a message in the chat. It

contains the message text and a flag indicating whether the message was sent by the user or the character.

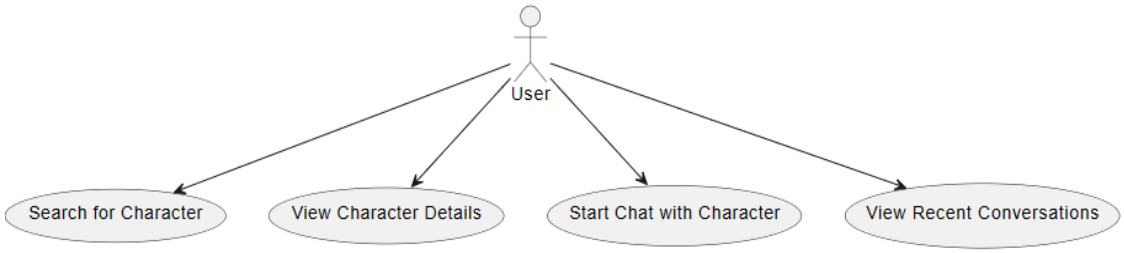


The system architecture diagram shows the interaction between the User, MainActivity, ChatActivity, RecentConversationsHelper, Character, and Message.

4.2. Use Case Diagram

The use case diagram for the AnimeChat application consists of two main actors: the User and the System (AnimeChat application). The User interacts with the System in the following ways:

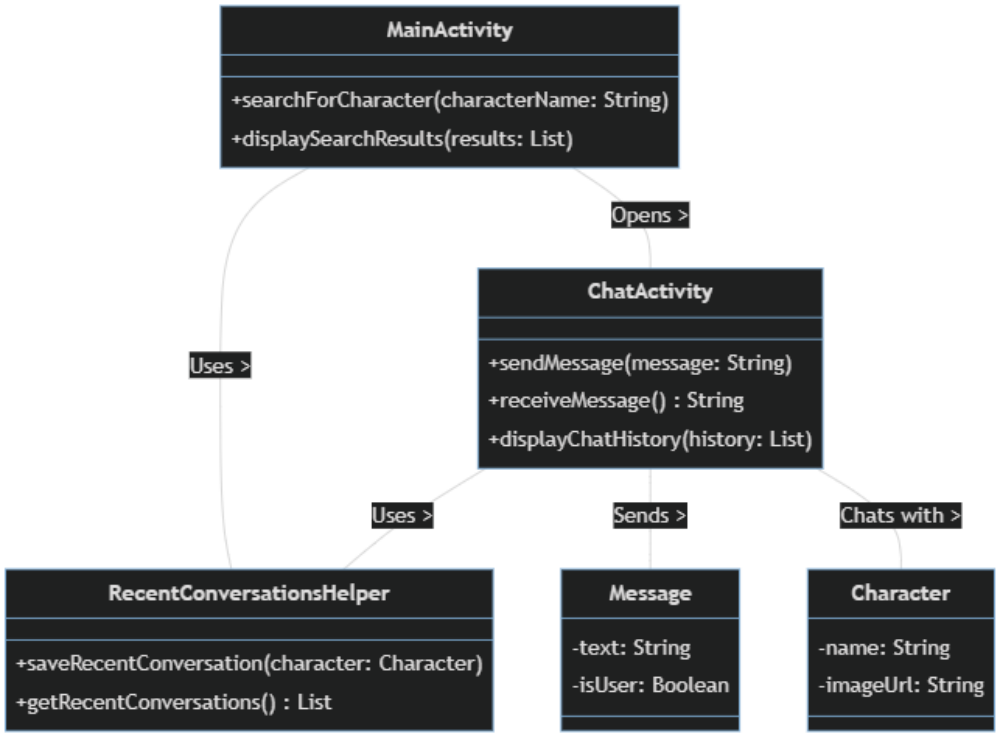
1. **Search for Character**: The User can enter the name of an anime character in the MainActivity and search for it.
2. **Select Character**: From the search results, the User can select a character to chat with.
3. **Chat with Character**: In the ChatActivity, the User can send messages to the character and receive responses.
4. **View Recent Conversations**: The User can view their recent conversations from the navigation drawer.



4.3. Class Diagram

The class diagram for the AnimeChat application consists of the following classes:

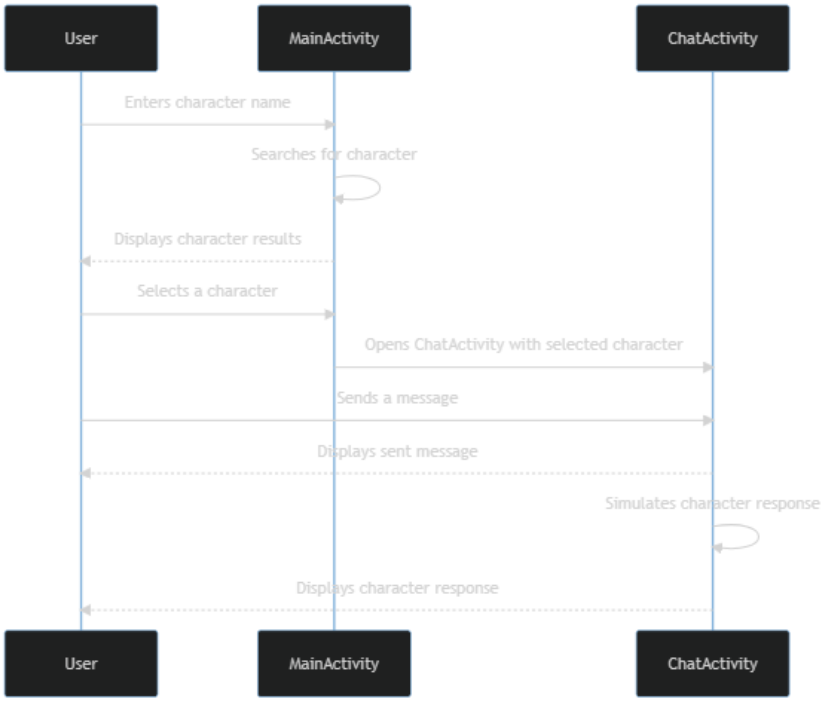
1. **MainActivity**: Contains methods for searching for characters and displaying the search results.
2. **ChatActivity**: Contains methods for sending messages, receiving responses, and displaying the chat history.
3. **RecentConversationsHelper**: Contains methods for saving and retrieving recent conversations.
4. **Character**: A data class that represents an anime character. Contains the character's name and image URL.
5. **Message**: A data class that represents a message in the chat. Contains the message text and a flag indicating whether the message was sent by the user or the character.



The class diagram shows the relationship between the different classes in the AnimeChat application.

4.4. Sequence Diagram

The sequence diagram for the AnimeChat application is as follows:



The sequence diagram shows the interaction between the User, MainActivity, and ChatActivity. The User enters a character name in MainActivity, which then searches for the character. The search results are displayed to the User, who can select a character to chat with. This opens the ChatActivity with the selected character. The User can then send messages and receive responses from the character

5. Implementation

The implementation of the AnimeChat application was done using the Android platform. The application was developed in Java and XML for the backend logic and frontend design respectively. The application consists of several classes and layouts which are described below:

5.1 Android Application

5.1.1 MainActivity

The MainActivity is the entry point of the application. It is

responsible for handling user input for character search and displaying the search results. It uses the JikanApiRequest class to fetch character data from the Jikan API based on the user's search input. It also handles navigation to the ChatActivity when a character is selected from the search results.

5.1.2 ChatActivity

The ChatActivity is responsible for handling the chat interface. It displays the chat messages between the user and the selected anime character. It uses the Message class to represent individual chat messages. It also uses the RecentConversationsHelper class to store and retrieve recent conversations.

5.1.3 JikanApiRequest

The JikanApiRequest class is responsible for making HTTP requests to the Jikan API. It fetches character data based on the search input provided by the user in the MainActivity.

5.1.4 RecentConversationsHelper

The RecentConversationsHelper class is used to store and retrieve recent conversations. It uses the SharedPreferences class to persist data across application launches. It stores a list of Character objects representing the characters with whom the user has recently conversed.

5.1.5 Message

The Message class represents a single chat message. It has two properties: text (the content of the message) and isUser (a boolean indicating whether the message was sent by the user or the anime character).

5.1.6 Character

The Character class represents an anime character. It has properties for the character's name, image URL, and about text.

5.2 User Interface

The user interface of the application was designed using XML. It consists of several layouts:

5.2.1 ChatActivity Layout

The ChatActivity layout consists of a RecyclerView for displaying the chat messages and a LinearLayout at the bottom containing an EditText for inputting messages and a Button for sending messages.

5.2.2 MainActivity Layout

The MainActivity layout consists of an EditText for inputting search queries, a Button for initiating the search, and a RecyclerView for displaying the search results. It also contains a NavigationView for displaying recent conversations.

5.2.3 Character Card Layout

The Character Card layout is used to display each character in the search results. It consists of an ImageView for the character's image, and two TextViews for the character's name and about text.

5.2.4 Message Layout

The Message layout is used to display each message in the chat. It consists of a TextView for the message text and a ProgressBar that displays a typing animation when the anime character is "typing" a response.

5.2.5 Navigation Menu Layout

The Navigation Menu layout is used to display the recent conversations in the navigation drawer. It is dynamically populated with menu items representing each recent conversation.

6. Testing

Testing is a crucial part of any software development process. It ensures that the software is functioning as expected and helps to identify any bugs or issues that need to be addressed. In the case of our AnimeChat application, we conducted both unit testing and integration testing to ensure the robustness of the application.

6.1 Unit Testing

Unit testing involves testing individual components of the software in isolation. In the context of our application, this meant testing individual classes and methods to ensure they functioned as expected.

For instance, we tested the *Character class* to ensure that it correctly stored and retrieved character information. We also tested the *Message class* to verify that it correctly identified whether a message was sent by the user or not.

The *RecentConversationsHelper* class was also subjected to unit testing. We ensured that it correctly saved and retrieved recent conversations. We also verified that it correctly handled the case where there were no recent conversations.

6.2 Integration Testing

Integration testing involves testing the interaction between different components of the software. For our application, this meant testing how different classes and methods worked together to deliver the desired functionality.

For example, we tested the interaction between the *MainActivity* and *ChatActivity* classes. We ensured that when a character was selected in the *MainActivity*, the *ChatActivity* was correctly launched with the selected character's information.

We also tested the interaction between the *JikanApiRequest class* and the *MainActivity* and *ChatActivity* classes. We verified that the *JikanApiRequest class* correctly fetched character data from the API and that this data was correctly displayed in the *MainActivity* and *ChatActivity*.

6.3 User Interface Testing

In addition to testing the underlying code, we also conducted user interface testing. This involved testing the layout files associated with each activity to ensure they correctly displayed the desired information and responded appropriately to user interaction.

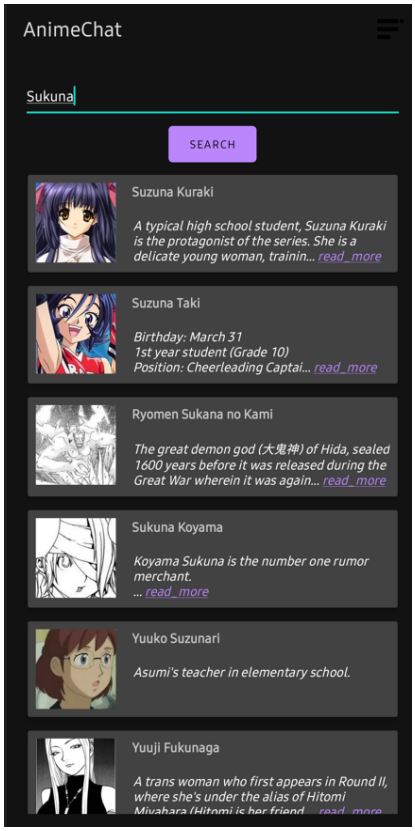
For instance, we tested the *activity_main.xml* and *activity_chat.xml* layout files to ensure they correctly displayed the list of characters and the chat interface, respectively. We also tested the *card_character.xml* and *message.xml* layout files to verify that they correctly displayed individual characters and messages.

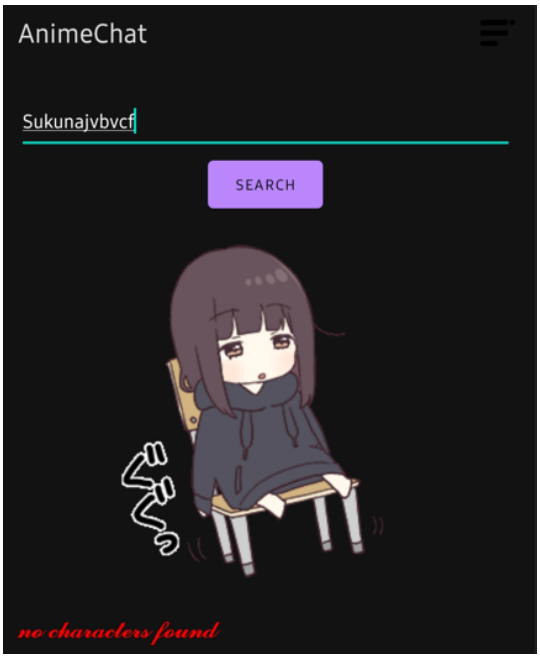
Through rigorous testing, we were able to identify and fix several bugs, resulting in a robust and reliable application.

7. Results and Discussion

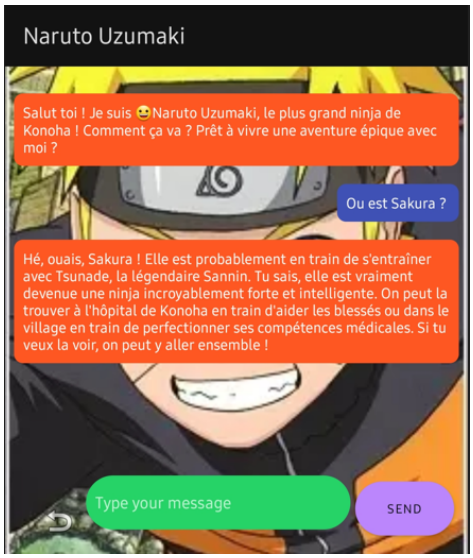
The AnimeChat application was successfully implemented and tested. The application allows users to search for anime characters and engage in simulated conversations with them. The results of the implementation and testing are discussed in this section.

The application's main interface, as defined in the MainActivity class, provides a search function that allows users to find anime characters by name. The search function uses the Jikan API to fetch character data. The API request is handled by the JikanApiRequest class, which executes the request asynchronously and returns the result to the MainActivity. The MainActivity then parses the JSON response and displays the characters in a RecyclerView. Each character is represented by a CardView, which displays the character's image, name, and a brief description.

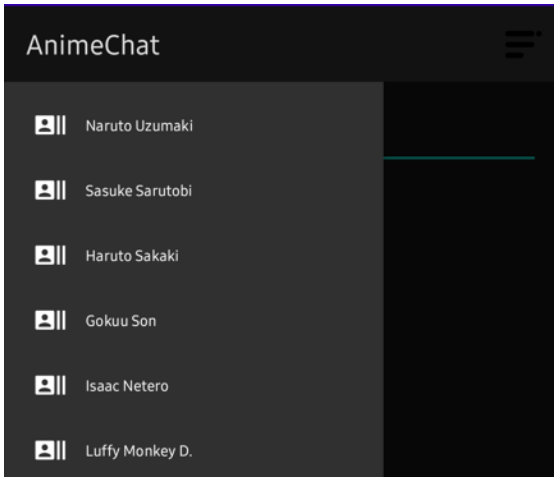




If a user selects a character, the application navigates to the ChatActivity, where the user can engage in a simulated conversation with the character. The conversation is represented as a series of messages, with each message being either a user message or a character message. The messages are displayed in a RecyclerView, with user messages aligned to the right and character messages aligned to the left.



The application also includes a navigation drawer, which displays a list of recent conversations. The list is managed by the RecentConversationsHelper class, which uses SharedPreferences to persist the list across application launches. When a user selects a character from the list, the application navigates to the ChatActivity for that character.



During testing, the application performed as expected. The search function was able to find characters based on partial or full names, and the chat interface provided a smooth and engaging user experience. The navigation drawer correctly displayed the list of recent conversations, and selecting a character from the list navigated to the correct chat interface.

However, there were some limitations observed during testing. The chat interface does not currently support sending images or other media, and the character responses are simulated and do not change based on the user's input. Additionally, the application does not handle API errors gracefully, which can lead to crashes or unexpected behavior if the API request fails.

8. Conclusion

The "AnimeChat" application is a comprehensive solution that demonstrates the principles of software development in a practical context. It incorporates various aspects of Android development, including user interface design, API integration, data persistence, and event handling.

The application provides a user-friendly interface for users to search for anime characters and initiate a chat conversation with them. It leverages the Jikan API to fetch character data based on user input and displays the results in a RecyclerView. The application also handles user interactions such as button clicks and keyboard events to trigger specific actions like initiating a search or sending a message.

The application also incorporates data persistence through the use of SharedPreferences. This feature allows the application to remember recent conversations and display them in a navigation drawer for easy access. This is an essential aspect of providing a seamless user experience.

The application's design is modular, with each class having a specific responsibility. This follows the Single Responsibility Principle, a key principle of software development. For instance, the JikanApiRequest class is responsible for making API requests, the RecentConversationsHelper class manages data persistence, and the MainActivity and ChatActivity classes handle user interface interactions.

The XML layouts used in the application are well-structured and make efficient use of Android's view components. They also demonstrate the use of different layout managers to arrange view components effectively.

In conclusion, the "AnimeChat" application is a practical demonstration of the principles of software development. It showcases how these principles can be applied in a real-world application to create a functional and user-friendly product. The project has provided valuable insights into Android development and the application of software development principles.

9. Future Work

The AnimeChat application has shown promising results in terms of functionality and user experience. However, there is always room for improvement and expansion. Here are some potential areas for future work:

1. Advanced Search Features: The current version of the application allows users to search for anime characters by name. In the future, we could enhance this feature by adding more search parameters such as anime series, character traits, or voice actors. This would provide a more comprehensive and personalized search experience for the users.
2. Chatbot Integration: Currently, the chat feature in the application is quite basic. We could integrate a chatbot that can simulate the personality of the anime character. This would make the chat experience more immersive and engaging for the users.
3. Social Features: We could add social features such as the ability to add friends, join public chat rooms, or share favorite characters on social media. This would make the application more interactive and community-driven.
4. Multilingual Support: To cater to a global audience, we could add support for multiple languages. This would make the application more accessible to non-English speaking users.
5. Performance Optimization: While the current version of the application performs well, there is always room for performance optimization. This could involve optimizing the API requests, improving the UI responsiveness, or reducing the application's memory footprint.
6. User Interface Enhancements: The user interface could be enhanced to be

more visually appealing and intuitive. This could involve adding animations, improving the layout, or updating the color scheme.

7. Offline Mode: Currently, the application requires an internet connection to function. We could add an offline mode that allows users to browse previously loaded data or chat histories.
8. Integration with Other APIs: The application currently uses the Jikan API to fetch anime character data. In the future, we could integrate with other APIs to fetch additional data such as anime series information, character trivia, or voice actor details.

By addressing these areas in future iterations of the AnimeChat application, we can enhance the user experience, add new features, and reach a wider audience.

10. References

- ☐ Android Developers. Developer guides. Retrieved from https://developer.android.com/guide
- ☐ OpenAI. ChatGPT. Retrieved from https://www.openai.com/research/chatgpt
- ☐ Google. Material Design. Retrieved from https://material.io/design
- ☐ Gson. Gson User Guide. Retrieved from https://github.com/google/gson/blob/master/UserGuide.md
- ☐ RecyclerView. RecyclerView. Retrieved from https://developer.android.com/jetpack/androidx/releases/recyclerview
- ☐ SharedPreferences. SharedPreferences. Retrieved from https://developer.android.com/reference/android/content/SharedPreferences
- ☐ Picasso. Picasso. Retrieved from https://github.com/square/picasso
- ☐ Jikan API. Jikan API Documentation. Retrieved from https://jikan.moe/