Currency Converter Report

Name : Lanka Pranay

Roll No : AM.EN.U4CSE21168

1. Introduction

1.1 Purpose

This report elaborates on the Software Requirements Specification (SRS) for a **Currency Converter Application** designed for Android devices. The app will enable users to convert between different currencies in real-time by leveraging external APIs for live exchange rates. The report outlines functional and non-functional requirements, system architecture, and key design constraints.

1.2 Scope

The Currency Converter Application provides a platform for quick and accurate currency conversion. The app integrates real-time exchange rate data and offers offline access to recently retrieved rates. The key functionalities include:

- Real-time currency conversion.
- Comprehensive currency support.
- Historical exchange rate views (optional).
- Offline functionality for last-accessed rates.
- User-friendly features such as favorite currencies.

This document serves as a reference for developers, testers, and stakeholders involved in the application lifecycle.

1.3 Definitions, Acronyms, and Abbreviations

- **SRS**: Software Requirements Specification.
- **API**: Application Programming Interface.
- **UI**: User Interface.
- **JSON**: JavaScript Object Notation.
- **IDE**: Integrated Development Environment.

1.4 References

1. Android Developer Documentation.
2. Java SE Documentation.
3. API Documentation (e.g., Frankfurter, Open Exchange Rates).
4. Android Material Design Guidelines.

---

2. Overall Description

2.1 Product Perspective

The application is a standalone tool designed for Android devices. It integrates external APIs to fetch live exchange rates and supports offline functionality using cached data. This ensures reliability and quick access for users, regardless of connectivity.

2.2 Product Features

1. **Currency Conversion**: Allows users to convert between any two currencies by entering an amount.
2. **Real-Time Exchange Rates**: Retrieves live data from APIs like Frankfurter or Open Exchange Rates.
3. **Currency List**: A comprehensive list of currencies is available for user selection.
4. **Favorites**: Users can mark commonly used currencies for faster access.
5. **Offline Mode**: Enables conversions using the most recently fetched rates when offline.
6. **Historical Exchange Rates (Optional)**: Provides access to exchange rate trends for specified date ranges.

2.3 User Classes and Characteristics

1. **Regular Users**: Individuals requiring currency conversion for travel, business, or financial purposes.
2. **System Administrators**: Backend managers ensuring smooth API integration and overall performance.

2.4 Operating Environment

- Android devices running version **5.0 (Lollipop)** or higher.
- Internet connectivity for real-time updates.
- Integration with APIs like Frankfurter or Open Exchange Rates.

2.5 Design and Implementation Constraints

- **Performance**: Target response time for API calls is under 2 seconds.
- **Currency Support**: The app will support at least 100 currencies.
- **Data Security**: Cached data for offline use must be encrypted.

2.6 Assumptions and Dependencies

- The user has a stable internet connection for real-time updates.
- The external API service is operational and reliable.

---

3. Functional Requirements

3.1 Currency Conversion

- Users can input an amount and select two currencies to convert.
- The app retrieves live exchange rates from APIs for calculations.

3.2 Real-Time Exchange Rates

- Fetches live exchange rates through API calls.
- Displays cached rates if the API is inaccessible.

3.3 Offline Mode

- Stores the latest rates locally for offline use.
- Users can perform conversions without internet access using cached data.

3.4 Currency List and Favorites

- A detailed list of currencies for selection.
- Option to mark currencies as favorites for quick access.

3.5 Historical Exchange Rates (Optional)

- Provides historical data for specified date ranges.

---

4. Non-Functional Requirements

4.1 Performance

- Exchange rates fetched and displayed within **2 seconds**.
- Conversion calculations must be instantaneous post-fetch.

4.2 Security

- Encrypt API keys and cached data.
- Securely handle sensitive user data.

4.3 Usability

- Simple and intuitive UI with clear error messages for invalid inputs.
- Prompts to guide users through the conversion process.

4.4 Reliability

- **99% availability**, excluding maintenance or API downtimes.

4.5 Scalability

- Must handle increased API calls as the user base grows.

---

5. System Architecture

5.1 Architecture Design

The application will follow an **MVC (Model-View-Controller)** design pattern:

1. **Model**: Handles data operations, including API calls and data caching.
2. **View**: Displays the UI, including currency selection and results.
3. **Controller**: Manages interactions between the Model and View.

5.2 Data Flow

1. **API Integration**: Fetches exchange rates in JSON format.
2. **Data Caching**: Saves fetched data locally for offline use.
3. **Conversion Engine**: Computes currency conversions based on the selected inputs.

---

6. Testing Plan

6.1 Functional Testing

- Validate currency conversion accuracy.
- Test real-time and offline functionality.

6.2 Performance Testing

- Measure API response times under varying loads.

6.3 Usability Testing

- Evaluate the UI for ease of navigation and clarity.

---

7. Deployment Plan

1. Develop using **Android Studio** with Java.
2. Integrate with APIs for real-time exchange rates.
3. Publish the app on the Google Play Store.

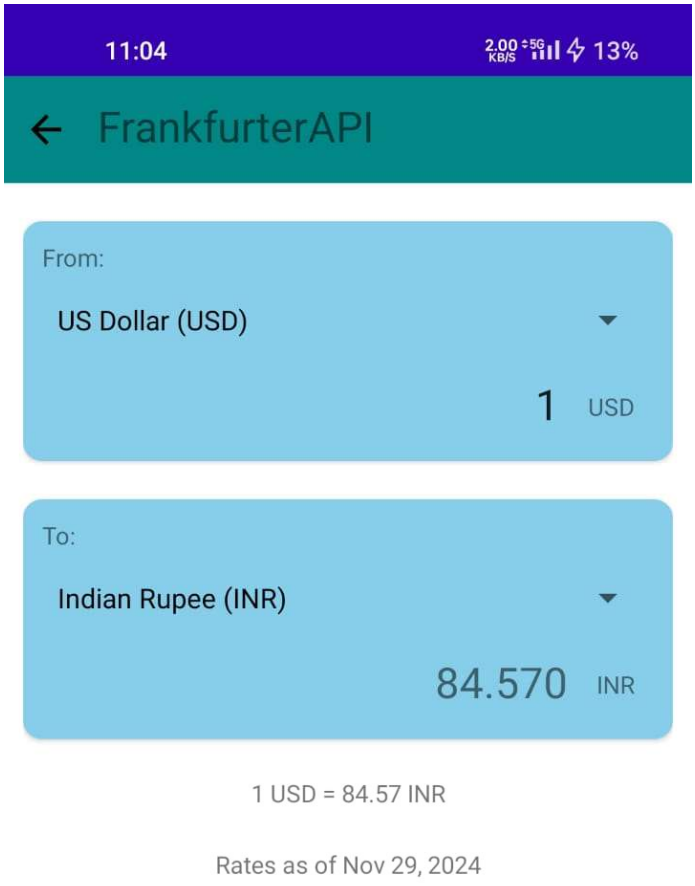
---

8. Conclusion

This report outlines a detailed roadmap for building the Currency Converter Application. With its robust features and user-focused design, the app will meet the needs of users requiring efficient currency conversion.

Screen Shots:

1.Using Frankfurter API



2.Using Currency API