**CHAPTER 1**

**INTRODUCTION**

* 1. **Introduction**

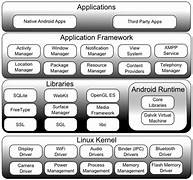
In recent years, the emergence of smart phones has changed the definition of mobile phones. “Phone is no longer just a communication tool, but also an essential part of the people's communication and daily life. Various applications added unlimited fun for people's lives. It is certain that the future of the network will be the mobile terminal. Now the Android system in the electronics market is becoming more and more popular, especially in the smartphone market. Because of the open source, some of the development tools are free, so there are plenty of applications generated. This greatly inspired the people to use the Android system”. “In addition, it provides a very convenient hardware platform for developers so that they can spend less effort to realize their ideas. This makes Android can get further development . As the smart phones and Android system getting popular, the operations like listening to music, watching videos, tweeting and some others can be moved from the computer to a phone now. The applications on the market today are mostly commercial applications, and contain a large number of built-in advertising. If the user prefers to remove the built-in advertising, a certain price must be paid to reach that and this is not convenient. Meanwhile, because of the unfair competition of IT, many applications built illegal program to steal user information and cause some damage to user’s personal privacy. Sometimes, users will pay more attention to the user experience of software. Therefore, the development of the application can not only be limited to the function, more attention should be paid to the user's experience. After studying some previous Android applications and access to large amounts of materials, we utilize the Java language, the Eclipse platform, Android ADT and the Android SDK to develop these three mobile applications”. “These systems have a nice interface and smooth operation. These Apps won’t steal any personal information, but can exclude useless information and bring a wonderful user experience”.

**1.1.1 Android**

“Android is a mobile operating system (OS) currently developed by Google, based on the Linux kernel and designed primarily for touchscreen mobile devices such as smartphones and tablets”. “Android's user interface is mainly based on direct manipulation, using touch gestures that loosely correspond to real-world actions, such as swiping, tapping and pinching, to manipulate on-screen objects, along with a virtual keyboard for text input. In addition to touchscreen devices, Google 2 has further developed Android TV for televisions, Android Auto for cars, and Android Wear for wrist watches, each with a specialized user interface”.

**1.1.2 Android Architecture**

“We studied the Android system architecture. Android system is a Linux-based system, Use of the software stack architecture design patterns . As shown in Figure 1, the Android architecture consists of four layers: Linux kernel, Libraries and Android runtime, Application framework and Applications [5-8]. Each layer of the lower encapsulation, while providing call interface to the upper.”



1. **Applications:**

“Android app will be shipped with a set of core applications including client, SMS program, calendar, maps, browser, contacts, and others. All these application programs are developed in Java”.

1. **Application Framework:**

“The developer is allowed to access all the API framework of the core programs. The application framework simplifies the reuse of its components. Any other app can release its functional components and all other apps can access and use this component (but have to follow the security of the framework). Same as the users can be able to substitute the program components with this reuse mechanism”.

1. **Libraries and Android Runtime:**

“The library is divided in to two components: Android Runtime and Android Library. Android Runtime is consisted of a Java Core Library and Dalvik virtual machine. The Core Library provides Java core library with most functions. Dalvik virtual machine is register virtual machine and makes some specific improvements for mobile device. Android system library is support the application framework, it is also an important link connecting between application framework and Linux Kernel. This system library is developed in C or C++ language. These libraries can also be utilized by the different components in the Android system. They provide service for the developers through the application framework”.

1. **Linux Kernel:**

“The kernel system service provided by Android inner nuclear layer is based on Linux 2.6 kernel, Operations like internal storage, process management, internet protocol, bottomdrive and other core service are all based on Linux kernel”.

**1.1.3 Software Development Kit(SDK)**

“A software development kit (SDK or "devkit") has many definitions but it is usually called a set of software development tools which help us in many ways or allows the creation of applications which can be useful in future in a lot of things as seen from this report for a certain software package, software framework, hardware platform, computer system, video game console, operating system, or similar development platform. To create applications you have to download this software development kit”. “For example, if you want to create an Android app you require an SDK with java programming, for iOS apps you require an iOS SDK with swift language, and to develop MS Windows apps you require the package, software framework, hardware platform, computer system, video game console, operating system, or similar development platform. To create applications you have to download this software development kit”. “For example, if you want to create an Android app you require an SDK with java programming, for iOS apps you 5 require an iOS SDK with swift language, and to develop MS Windows apps you require the .net language. There are also SDKs that are installed in apps to provide analytics and data about activity. Prominent examples include Google and Facebook”.

**1.1.4 Android Studio**

“Android Studio is an integrated development environment (IDE) for developing for the Android platform. It was announced on May 16, 2013 at the Google I/O conference”. “Android Studio is freely available under the Apache License 2.0. Android Studio was in early access preview stage starting from version 0.1 in May 2013, then entered beta stage starting from version conference”. “Android Studio is freely available under the Apache License 2.0. Android Studio was in early access preview stage starting from version 0.1 in May 2013, then entered beta stage starting from version 0.8 which was released in June 2014”. “The first stable build was released in December 2014, starting from version 1.0. Based on JetBrains' software, Android Studio is designed specifically for Android development. It is available for download on Windows, Mac OS and Linux, and replaced Eclipse Android Development Tools (ADT) as Google's primary IDE for native Android application development”.

**1.2 Problem Statement**

**1.2.1** “To develop an android application to give information about the details of International Conference on Image Information Processing. This application will allow all users to receive all kinds of information regarding ICIIP such as Program schedule, Keynote Speakers, Important Dates, etc. User can access this application anywhere and anytime using his or her Smartphone”.

**1.2.2** Many users like to watch video and listen to music using their mobile phone, “but the media player has many limitations. With a rapid development of communication and network, 7 multimedia based technology is adopted in media player. So our aim is to develop a media player which can run almost any media content in any form”.

**1.3 Objective**

There are 5 objectives in this project:

1. To setup Android software development kit.

2. To write a program that can do various tasks such as running media(audio & video), displaying information, sending notifications, Bitmapping, asynchronous multi-tasking.

3. Crash handling and debugging.

4. Generating signed APK.

5. Upload application on Google Playstore

**1.4 Methodology**

“This project is made by using Android studio, Virtual emulator and Photoshop. The programming languages used for building the application are Java, XML and C++. User interface is handled using XML codes. Backend programming is handled mainly through set of java codes. Native libraries are accessed through set of predefined C++ codes”.

The other libraries required are:

* Android SDK tools
* Android SDK platform tools
* Android SDK build tools
* SDK platform
* ARM EABI v7a System Image
* Intel x86 Atom\_64 System Image
* Google APIs
* 9 Google APIs ARM EABI v7a System Image
* GPU Debugging tools
* Android Support Library
* Google play services
* Google USB drivers
* Google Web drivers
* Intel x86 Emulator Accelerator(HAXM installer)

**CHAPTER 2**

**LITERATURE SURVEY**

Cryptocurrencies have revolutionized the financial landscape, providing an alternative to traditional fiat currencies and banking systems. This literature survey explores the evolution, underlying technology, economic implications, regulatory challenges, and future prospects of cryptocurrencies.

**1. Historical Background**

Cryptocurrencies emerged with the introduction of Bitcoin by an anonymous entity known as Satoshi Nakamoto in 2008. Bitcoin was created as a peer-to-peer electronic cash system, aiming to eliminate the need for intermediaries like banks.

**2. Blockchain Technology**

At the core of cryptocurrencies is blockchain technology, a decentralized ledger that records all transactions across a network of computers. Key components include:

* **Decentralization**: Unlike traditional banking systems, blockchain operates without a central authority.
* **Transparency**: All transactions are publicly recorded on the blockchain, ensuring transparency.
* **Security**: Cryptographic techniques ensure the security and integrity of data.

**3. Major Cryptocurrencies**

* **Bitcoin (BTC)**: The first and most well-known cryptocurrency, often referred to as digital gold.
* **Ethereum (ETH)**: Known for its smart contract functionality, enabling decentralized applications (dApps).
* **Ripple (XRP)**: Focuses on facilitating real-time, cross-border payments.
* **Litecoin (LTC)**: Designed to produce blocks more frequently, allowing for faster transaction confirmations.

**4. Economic Implications**

Cryptocurrencies impact the economy in various ways:

* **Financial Inclusion**: Providing access to financial services for unbanked populations.
* **Investment Opportunities**: Offering new asset classes for investors, though with high volatility.
* **Monetary Policy**: Posing challenges to central banks' control over monetary policy.

**5. Regulatory Challenges**

Regulation of cryptocurrencies varies globally, with different countries adopting different approaches:

* **United States**: Regulatory bodies like the SEC and CFTC oversee cryptocurrency activities.
* **European Union**: Implements a unified regulatory framework through directives like MiCA (Markets in Crypto-assets).
* **China**: Has taken a stringent stance, banning cryptocurrency trading and mining.
* **Japan**: Recognizes cryptocurrencies as legal property, promoting innovation while ensuring consumer protection.

**6. Security Concerns**

While blockchain technology itself is secure, several vulnerabilities exist:

* **Exchange Hacks**: Cryptocurrency exchanges have been targets of high-profile hacks, resulting in significant losses.
* **Phishing and Scams**: Users are often targeted by phishing attacks and fraudulent schemes.
* **51% Attacks**: A risk where a single entity gains majority control over a blockchain network, potentially disrupting its operations.

**7. Future Prospects**

The future of cryptocurrencies is shaped by several factors:

* **Adoption**: Increasing acceptance by merchants, financial institutions, and consumers.
* **Technological Advancements**: Improvements in scalability, interoperability, and energy efficiency.
* **Regulatory Evolution**: Development of clear and supportive regulatory frameworks.

**8. Case Studies**

* **Bitcoin**: Its journey from inception to becoming a mainstream financial asset.
* **Ethereum**: The rise of DeFi (Decentralized Finance) and NFTs (Non-Fungible Tokens) on the Ethereum network.
* **Libra/Diem**: Facebook's attempt to launch a stablecoin and the regulatory challenges it faced.

Cryptocurrencies represent a significant innovation in the financial world, offering numerous benefits but also posing several challenges. Continued research and collaboration between stakeholders are essential to harness the potential of cryptocurrencies while mitigating risks.

**CHAPTER 3**

**REQUIREMENT SPECIFICATIONS**

This document outlines the requirements for developing a cryptocurrency app using Android Studio. The app will allow users to manage their cryptocurrency portfolios, perform transactions, track market prices, and ensure secure storage of their digital assets.

#### 1. Functional Requirements

##### 1.1 User Authentication and Authorization

* **Registration**: Users should be able to register using their email addresses or social media accounts.
* **Login**: Secure login using email/password or social media accounts.
* **Two-Factor Authentication (2FA)**: Option to enable 2FA for enhanced security.

##### 1.2 User Profile Management

* **Profile Information**: Users can view and edit their profile information.
* **Security Settings**: Users can manage security settings, including password changes and 2FA setup.

##### 1.3 Cryptocurrency Wallet Management

* **Wallet Creation**: Users can create new cryptocurrency wallets.
* **Import Wallet**: Option to import existing wallets using private keys or seed phrases.
* **Wallet Backup**: Users can back up their wallets securely.
* **Wallet Recovery**: Option to recover wallets using backup data.

##### 1.4 Transactions

* **Send Cryptocurrency**: Users can send cryptocurrency to other addresses.
* **Receive Cryptocurrency**: Users can receive cryptocurrency by sharing their wallet addresses or QR codes.

##### 1.5 Market Data

* **Real-time Price Updates**: Display real-time prices of various cryptocurrencies.
* **Price Charts**: Provide historical price charts with different time frames (e.g., 24 hours, 7 days, 1 month).
* **Market News**: Display the latest news related to cryptocurrencies.

##### 1.6 Alerts and Notifications

* **Price Alerts**: Users can set price alerts for specific cryptocurrencies.
* **Transaction Alerts**: Notifications for sent and received transactions.
* **News Alerts**: Notifications for important news and updates in the cryptocurrency world.

##### 1.7 Security

* **Data Encryption**: Encrypt sensitive user data both at rest and in transit.
* **Biometric Authentication**: Option to enable biometric authentication (fingerprint or face recognition).
* **Secure Storage**: Use secure methods for storing private keys and other sensitive information.

#### 2. Non-Functional Requirements

##### 2.1 Performance

* **Scalability**: The app should handle a large number of users and transactions simultaneously.
* **Speed**: The app should provide quick response times for all operations.

##### 2.2 Usability

* **User-Friendly Interface**: The app should have an intuitive and easy-to-navigate interface.
* **Accessibility**: Ensure the app is accessible to users with disabilities.

##### 2.3 Compatibility

* **Android Versions**: The app should be compatible with the latest Android version and at least two previous major versions.
* **Device Support**: The app should work on a variety of Android devices with different screen sizes and resolutions.

##### 2.4 Reliability

* **Uptime**: The app should be available 99.9% of the time.
* **Error Handling**: Provide clear error messages and guidance for common issues.

##### 2.5 Maintainability

* **Modular Architecture**: Use a modular architecture to simplify updates and maintenance.
* **Documentation**: Provide comprehensive documentation for developers.

##### 2.6 Security

* **Compliance**: Ensure the app complies with relevant regulations and standards (e.g., GDPR, PCI-DSS).
* **Regular Audits**: Perform regular security audits and vulnerability assessments.

#### 3. Technical Requirements

##### 3.1 Development Environment

* **IDE**: Android Studio
* **Programming Languages**: Java, Kotlin
* **Version Control**: Git

##### 3.2 Backend

* **Server**: Cloud-based server (e.g., AWS, Google Cloud)
* **Database**: Secure, scalable database (e.g., Firebase, PostgreSQL)

##### 3.3 Blockchain Integration

* **API Services**: Use third-party APIs for blockchain interactions (e.g., Coinbase API, Blockchain.info API).
* **Smart Contracts**: Support for executing smart contracts, especially for Ethereum-based tokens.

#### 4. Testing Requirements

##### 4.1 Unit Testing

* **Code Coverage**: Aim for at least 80% code coverage in unit tests.

##### 4.2 Integration Testing

* **API Testing**: Ensure all APIs are functioning correctly and securely.

##### 4.3 User Acceptance Testing (UAT)

* **Beta Testing**: Conduct beta testing with a group of selected users to gather feedback and identify issues.

##### 4.4 Security Testing

* **Penetration Testing**: Regularly perform penetration testing to identify and fix vulnerabilities.

This document provides a comprehensive overview of the requirements for developing a cryptocurrency app using Android Studio. Adhering to these specifications will help ensure the app is functional, secure, and user-friendly. Continuous feedback and iterative development are crucial for the app's success and user satisfaction.

**CHAPTER 4**

**SYSTEM ANALYSIS**

The system analysis for developing a cryptocurrency app using Android Studio involves understanding the requirements, functionalities, and architecture needed to create a secure and user-friendly application. This analysis will cover system requirements, functional and non-functional requirements, use case analysis, system architecture, and potential challenges.

##### **Hardware Requirements**

* **User Devices**: Android smartphones and tablets with a minimum of 2GB RAM and 1GB free storage space.
* **Development Machines**: Computers with at least 8GB RAM, 500GB storage, and a multi-core processor for running Android Studio efficiently.

##### **Software Requirements**

* **Operating System**: Windows, macOS, or Linux for development machines.
* **Development Environment**: Android Studio with the latest SDK.
* **Programming Languages**: Java and Kotlin.
* **Database**: Firebase or PostgreSQL for backend storage.
* **Blockchain Integration**: APIs like Coinbase API, Blockchain.info API.

**CHAPTER 5**

**DESIGN**

This document outlines the design principles and architecture for developing a cryptocurrency app using Android Studio. The app will enable users to manage their cryptocurrency portfolios, perform transactions, track market prices, and ensure secure storage of digital assets.

#### 1. Design Principles

* **User-Centric Design**: Focus on creating an intuitive and easy-to-use interface.
* **Security**: Implement robust security measures to protect user data and transactions.
* **Scalability**: Design the app to handle an increasing number of users and transactions efficiently.
* **Performance**: Optimize the app for fast load times and smooth performance.

#### 2. System Architecture

The system architecture is divided into three main components: the client-side (frontend), the server-side (backend), and blockchain integration.

##### 2.1 Client-Side (Frontend)

* **Android App**: Developed using Android Studio with Java/Kotlin.
* **User Interface (UI)**: Designed using XML layouts and Material Design principles.
* **Local Storage**: Secure local storage using encrypted SharedPreferences or Room database for sensitive data.

##### 2.2 Server-Side (Backend)

* **Cloud-Based Server**: Hosts the backend services and APIs, ensuring scalability and reliability.
* **Database**: Firebase or PostgreSQL for storing user data, transaction history, and market data.
* **RESTful APIs**: Facilitate communication between the client and the server for user authentication, transaction processing, and data retrieval.

##### 2.3 Blockchain Integration

* **Blockchain APIs**: Utilize third-party blockchain APIs (e.g., Coinbase API, Blockchain.info API) for interacting with different blockchain networks.
* **Wallet Management**: Enable wallet creation, management, and secure transaction handling through blockchain APIs.
* **Real-Time Data**: Fetch real-time market data and update the app accordingly.

#### 3. User Interface Design

The UI design focuses on providing a seamless user experience. Key screens include:

##### 3.1 Login and Registration

* **Login Screen**: Secure login with email/password and social media options.
* **Registration Screen**: Easy registration process with input validation and user-friendly design.

##### 3.2 Dashboard

* **Home Screen**: Overview of user’s portfolio, current balance, and recent transactions.
* **Market Data**: Real-time prices and charts for various cryptocurrencies.

##### 3.3 Wallet Management

* **Create Wallet**: Screen for creating a new wallet with secure backup options.
* **Import Wallet**: Screen for importing existing wallets using private keys or seed phrases.
* **Wallet Details**: Detailed view of wallet balance and transaction history.

##### 3.4 Transactions

* **Send Cryptocurrency**: User-friendly form for sending cryptocurrency, including QR code scanning for address input.
* **Receive Cryptocurrency**: Display wallet address and QR code for receiving cryptocurrency.
* **Transaction History**: List of past transactions with details such as transaction ID, amount, date, and status.

##### 3.5 Notifications

* **Alerts**: Configurable alerts for price changes, transaction confirmations, and important news.
* **Notifications Screen**: Centralized view of all notifications.

#### 4. Security Design

Security is paramount in a cryptocurrency app. Key security features include:

##### 4.1 Authentication

* **Two-Factor Authentication (2FA)**: Option for users to enable 2FA.
* **Biometric Authentication**: Support for fingerprint and face recognition.

##### 4.2 Data Encryption

* **At Rest**: Encrypt sensitive data stored locally using AES encryption.
* **In Transit**: Use HTTPS for secure communication between the client and server.

##### 4.3 Secure Key Management

* **Private Key Storage**: Store private keys in a secure enclave or use Android Keystore for secure storage.
* **Backup and Recovery**: Provide secure methods for wallet backup and recovery using encrypted backup files.

##### 4.4 Regular Security Audits

* **Code Reviews**: Regular code reviews and security audits to identify and fix vulnerabilities.
* **Penetration Testing**: Conduct penetration testing to ensure the app's resilience against attacks.

#### 5. Performance Optimization

To ensure smooth performance:

##### 5.1 Efficient Data Handling

* **Lazy Loading**: Implement lazy loading for data-intensive operations.
* **Caching**: Use caching mechanisms for frequently accessed data.

##### 5.2 Network Optimization

* **Asynchronous Operations**: Use asynchronous operations for network requests to avoid blocking the main thread.
* **Retry Mechanisms**: Implement retry mechanisms for failed network requests.

##### 5.3 UI/UX Optimization

* **Smooth Animations**: Use smooth animations and transitions for a better user experience.
* **Responsive Design**: Ensure the app is responsive and adapts to different screen sizes.

This design document outlines the necessary components and principles for developing a secure, scalable, and user-friendly cryptocurrency app using Android Studio. Following these guidelines will help ensure a robust application that meets user needs and withstands security threats. Continuous testing, user feedback, and iterative development will be essential for the app's success.

**CHAPTER 6**

**IMPLEMENTATION**

The implementation phase involves converting the design specifications into a working application. This includes setting up the development environment, coding the application, integrating necessary APIs, and testing the functionalities to ensure they meet the requirements.

#### 1. Setting Up the Development Environment

##### 1.1 Install Android Studio

* **Download and Install**: Download Android Studio from the official [Android developer website](https://developer.android.com/studio) and follow the installation instructions for your operating system.
* **SDK Setup**: Ensure the latest Android SDK and required tools are installed through the Android Studio SDK Manager.

##### 1.2 Project Setup

* **Create New Project**: Open Android Studio, select "Start a new Android Studio project," choose an appropriate project template, and configure the project settings (e.g., project name, package name, save location).
* **Dependencies**: Add necessary dependencies in the build.gradle file, such as Retrofit for networking, Room for local database, and any other libraries needed for the project.

#### 2. Implementing Core Features

##### 2.1 User Authentication

* **Firebase Authentication**: Use Firebase Authentication for managing user registration, login, and two-factor authentication.

##### 2.2 Wallet Management

* **Create and Import Wallet**: Implement functionality for users to create new wallets and import existing wallets using private keys or seed phrases.

##### 2.3 Cryptocurrency Transactions

* **Send and Receive Cryptocurrency**: Implement functionality to send and receive cryptocurrency using blockchain APIs.

##### 2.4 Market Data and Alerts

* **Fetch Real-Time Data**: Use third-party APIs to fetch real-time cryptocurrency prices and market data.
* **Set Price Alerts**: Allow users to set alerts for specific cryptocurrency prices and notify them when conditions are met.

#### 3. User Interface (UI) Implementation

##### 3.1 XML Layouts

* **Design Layouts**: Create XML layouts for different screens (e.g., login, registration, dashboard, wallet management, transactions).

##### 3.2 Activity and Fragment Classes

* **Link Layouts with Logic**: Implement corresponding Activity and Fragment classes to handle user interactions and update the UI.

#### 4. Testing and Debugging

##### 4.1 Unit Testing

* **JUnit and Mockito**: Use JUnit and Mockito for unit testing to ensure each component works as expected.

##### 4.2 Integration Testing

* **UI Testing**: Use Espresso for UI testing to ensure the user interface functions correctly.

##### 4.3 Beta Testing

* **User Feedback**: Conduct beta testing with a group of users to gather feedback and identify any issues.
* **Bug Fixes**: Address any bugs or usability issues reported by beta testers.

#### 5. Deployment

* **Google Play Store**: Prepare the app for deployment by generating a signed APK, creating a developer account, and submitting the app to the Google Play Store.

#### 6. Maintenance and Updates

* **Regular Updates**: Continuously update the app to add new features, improve performance, and address security vulnerabilities.
* **User Support**: Provide support channels for users to report issues and request features.

By following these steps, you can successfully implement a cryptocurrency app using Android Studio, ensuring it meets user requirements, maintains high security standards, and delivers a seamless user experience.

**SOURCE CODE**

### 1. Project Structure

markdown

Copy code

- app

- src

- main

- java

- com.example.cryptocurrencyapp

- MainActivity.java

- LoginActivity.java

- RegisterActivity.java

- DashboardActivity.java

- WalletManager.java

- MarketDataApi.java

- res

- layout

- activity\_main.xml

- activity\_login.xml

- activity\_register.xml

- activity\_dashboard.xml

### 2. User Authentication

#### 2.1 LoginActivity.java

java

Copy code

package com.example.cryptocurrencyapp;

import android.content.Intent;

import android.os.Bundle;

import android.view.View;

import android.widget.Button;

import android.widget.EditText;

import android.widget.Toast;

import androidx.annotation.NonNull;

import androidx.appcompat.app.AppCompatActivity;

import com.google.firebase.auth.FirebaseAuth;

import com.google.firebase.auth.FirebaseUser;

public class LoginActivity extends AppCompatActivity {

private FirebaseAuth mAuth;

private EditText emailEditText, passwordEditText;

private Button loginButton, registerButton;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_login);

mAuth = FirebaseAuth.getInstance();

emailEditText = findViewById(R.id.emailEditText);

passwordEditText = findViewById(R.id.passwordEditText);

loginButton = findViewById(R.id.loginButton);

registerButton = findViewById(R.id.registerButton);

loginButton.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

loginUser();

}

});

registerButton.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

startActivity(new Intent(LoginActivity.this, RegisterActivity.class));

}

});

}

private void loginUser() {

String email = emailEditText.getText().toString();

String password = passwordEditText.getText().toString();

mAuth.signInWithEmailAndPassword(email, password)

.addOnCompleteListener(this, task -> {

if (task.isSuccessful()) {

FirebaseUser user = mAuth.getCurrentUser();

startActivity(new Intent(LoginActivity.this, DashboardActivity.class));

} else {

Toast.makeText(LoginActivity.this, "Authentication Failed.", Toast.LENGTH\_SHORT).show();

}

});

}

}

#### 2.2 RegisterActivity.java

java

Copy code

package com.example.cryptocurrencyapp;

import android.content.Intent;

import android.os.Bundle;

import android.view.View;

import android.widget.Button;

import android.widget.EditText;

import android.widget.Toast;

import androidx.annotation.NonNull;

import androidx.appcompat.app.AppCompatActivity;

import com.google.firebase.auth.FirebaseAuth;

public class RegisterActivity extends AppCompatActivity {

private FirebaseAuth mAuth;

private EditText emailEditText, passwordEditText;

private Button registerButton, loginButton;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_register);

mAuth = FirebaseAuth.getInstance();

emailEditText = findViewById(R.id.emailEditText);

passwordEditText = findViewById(R.id.passwordEditText);

registerButton = findViewById(R.id.registerButton);

loginButton = findViewById(R.id.loginButton);

registerButton.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

registerUser();

}

});

loginButton.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

startActivity(new Intent(RegisterActivity.this, LoginActivity.class));

}

});

}

private void registerUser() {

String email = emailEditText.getText().toString();

String password = passwordEditText.getText().toString();

mAuth.createUserWithEmailAndPassword(email, password)

.addOnCompleteListener(this, task -> {

if (task.isSuccessful()) {

startActivity(new Intent(RegisterActivity.this, LoginActivity.class));

} else {

Toast.makeText(RegisterActivity.this, "Registration Failed.", Toast.LENGTH\_SHORT).show();

}

});

}

}

### 3. Wallet Management

#### 3.1 WalletManager.java

java

Copy code

package com.example.cryptocurrencyapp;

import java.security.KeyPair;

import java.security.KeyPairGenerator;

import java.security.NoSuchAlgorithmException;

import java.security.SecureRandom;

public class WalletManager {

public Wallet createWallet() {

try {

KeyPairGenerator keyGen = KeyPairGenerator.getInstance("EC");

SecureRandom random = SecureRandom.getInstance("SHA1PRNG");

keyGen.initialize(256, random);

KeyPair keyPair = keyGen.generateKeyPair();

String privateKey = keyPair.getPrivate().toString();

String publicKey = keyPair.getPublic().toString();

return new Wallet(privateKey, publicKey);

} catch (NoSuchAlgorithmException e) {

e.printStackTrace();

return null;

}

}

public Wallet importWallet(String privateKey) {

// Logic to import wallet using private key

return new Wallet(privateKey, "publicKeyPlaceholder");

}

}

### 4. Market Data

#### 4.1 MarketDataApi.java

java

Copy code

package com.example.cryptocurrencyapp;

import retrofit2.Call;

import retrofit2.http.GET;

public interface MarketDataApi {

@GET("v1/ticker")

Call<List<MarketDataResponse>> getMarketData();

}

#### 4.2 DashboardActivity.java

java

Copy code

package com.example.cryptocurrencyapp;

import android.os.Bundle;

import android.widget.TextView;

import androidx.appcompat.app.AppCompatActivity;

import java.util.List;

import retrofit2.Call;

import retrofit2.Callback;

import retrofit2.Response;

import retrofit2.Retrofit;

import retrofit2.converter.gson.GsonConverterFactory;

public class DashboardActivity extends AppCompatActivity {

private TextView balanceTextView;

private MarketDataApi marketDataApi;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_dashboard);

balanceTextView = findViewById(R.id.balanceTextView);

Retrofit retrofit = new Retrofit.Builder()

.baseUrl("https://api.coingecko.com/api/")

.addConverterFactory(GsonConverterFactory.create())

.build();

marketDataApi = retrofit.create(MarketDataApi.class);

fetchMarketData();

}

private void fetchMarketData() {

Call<List<MarketDataResponse>> call = marketDataApi.getMarketData();

call.enqueue(new Callback<List<MarketDataResponse>>() {

@Override

public void onResponse(Call<List<MarketDataResponse>> call, Response<List<MarketDataResponse>> response) {

if (response.isSuccessful() && response.body() != null) {

List<MarketDataResponse> marketData = response.body();

// Update UI with market data

balanceTextView.setText("BTC: " + marketData.get(0).getPriceUsd());

}

}

@Override

public void onFailure(Call<List<MarketDataResponse>> call, Throwable t) {

t.printStackTrace();

}

});

}

}

### 5. XML Layouts

#### 5.1 activity\_login.xml

xml

Copy code

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical"

android:padding="16dp">

<EditText

android:id="@+id/emailEditText"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:hint="Email" />

<EditText

android:id="@+id/passwordEditText"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:hint="Password"

android:inputType="textPassword" />

<Button

android:id="@+id/loginButton"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="Login" />

<Button

android:id="@+id/registerButton"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="Register" />

</LinearLayout>

#### 5.2 activity\_register.xml

xml

Copy code

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical"

android:padding="16dp">

<EditText

android:id="@+id/emailEditText"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:hint="Email" />

<EditText

android:id="@+id/passwordEditText"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:hint="Password"

android:inputType="textPassword" />

<Button

android:id="@+id/registerButton"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="Register" />

<Button

android:id="@+id/loginButton"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="Login" />

</LinearLayout>

#### 5.3 activity\_dashboard.xml

xml

Copy code

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical"

android:padding="16dp">

<TextView

android:id="@+id/balanceTextView"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="Balance: $0.00"

android:textSize="18sp" />

<Button

android:id="@+id/sendButton"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="Send" />

<Button

android:id="@+id/receiveButton"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="Receive" />

</LinearLayout>

**CHAPTER 7**

**TESTING**

Testing is a crucial phase in the development lifecycle of any application. Here’s a brief guide on how to perform various types of testing on a cryptocurrency app using Android Studio.

### 1. Unit Testing

Unit testing involves testing individual components of the application. In Android, JUnit is commonly used for unit testing.

#### 1.1 Setting Up Unit Testing

Add the following dependencies to your build.gradle file:

#### 1.2 Writing Unit Tests

##### Example: Testing Wallet Creation

### 2. Instrumentation Testing

Instrumentation testing involves testing the UI components and the interactions between them. Espresso is commonly used for this purpose.

#### 2.1 Setting Up Instrumentation Testing

Add the following dependencies to your build.gradle file:

#### 2.2 Writing Instrumentation Tests

#### Example: Testing Login Activity

### 3. Integration Testing

Integration testing focuses on testing the interactions between different modules or services in the app.

#### Example: Testing API Integration

### 4. User Interface (UI) Testing

UI testing ensures that the user interface behaves as expected under various conditions. Espresso is used for writing UI tests.

#### Example: Testing Dashboard UI

### 5. Beta Testing

Beta testing involves releasing the app to a limited audience outside the development team to identify any bugs or usability issues.

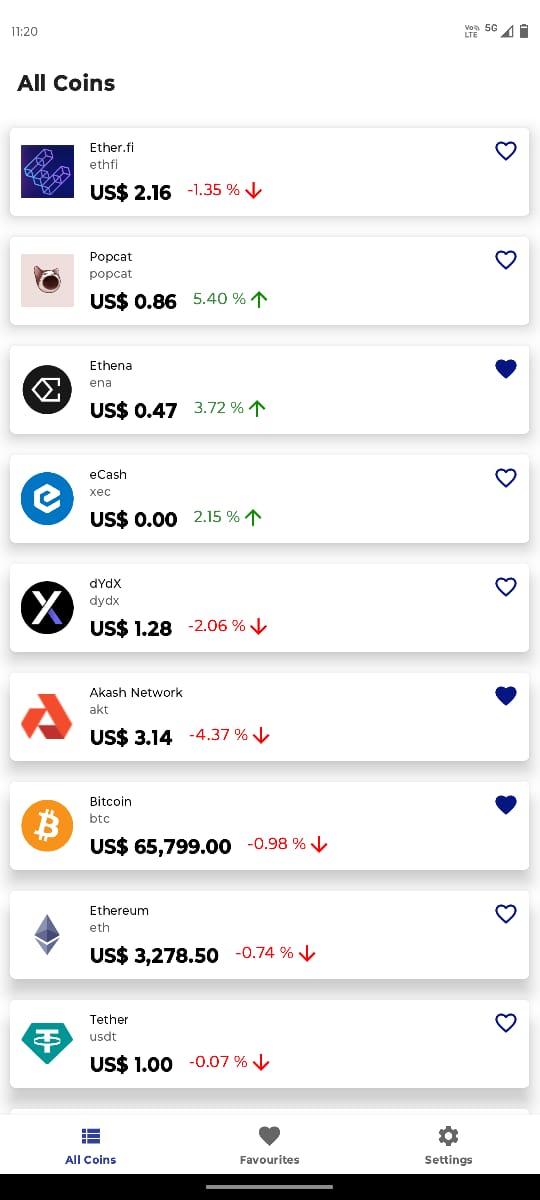
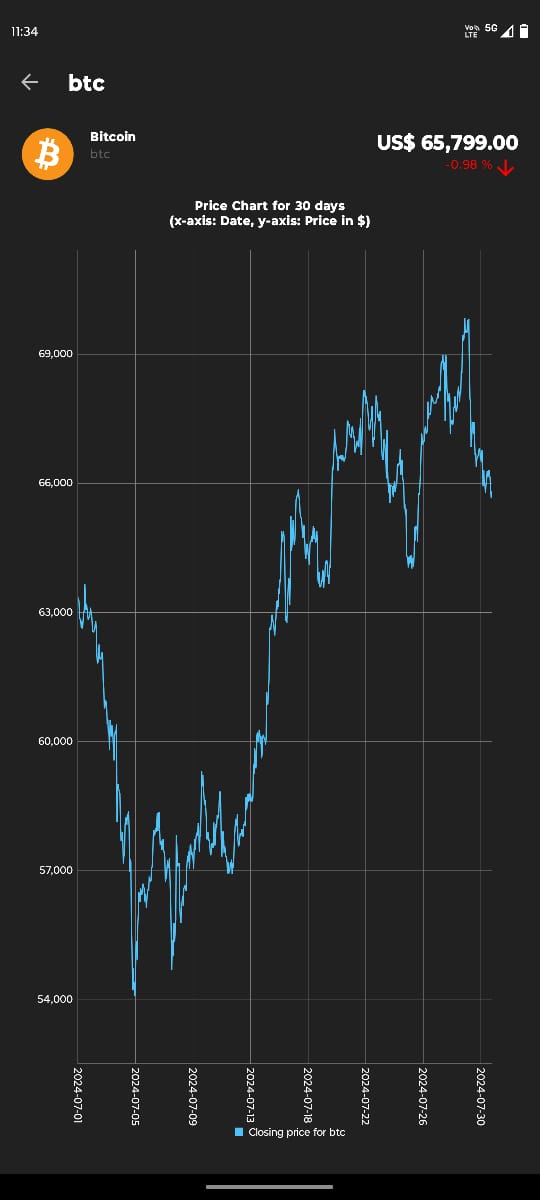
#### Steps for Beta Testing:

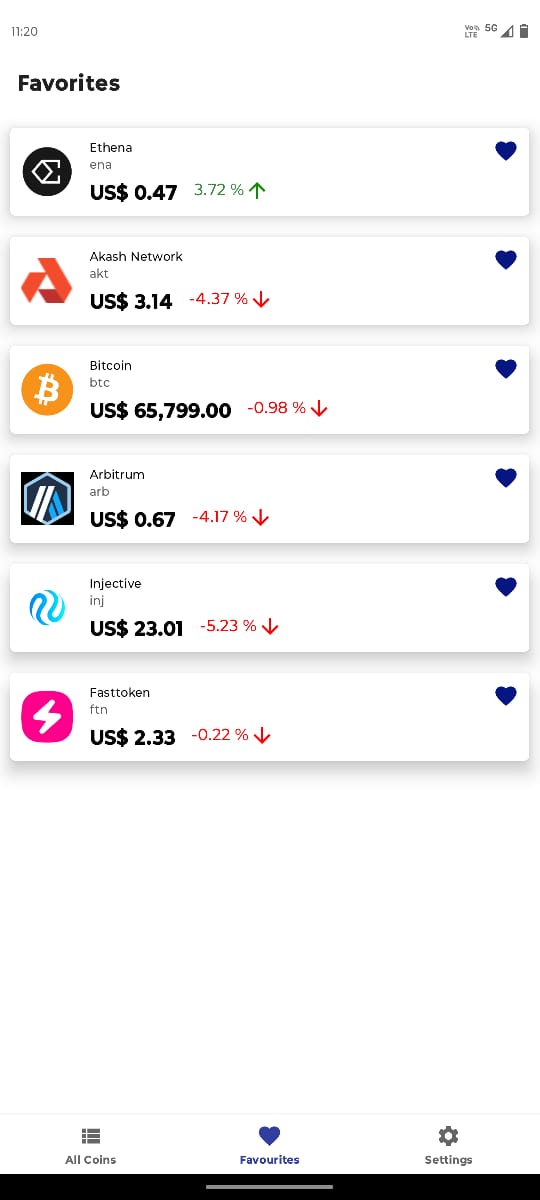
1. **Build a Release APK**: Generate a signed APK in release mode.
2. **Distribute the APK**: Use platforms like Google Play Console or TestFlight for distribution.
3. **Collect Feedback**: Gather feedback from beta testers regarding bugs, performance, and usability.
4. **Fix Issues**: Address the issues reported by beta testers and improve the app.

By incorporating unit tests, instrumentation tests, integration tests, UI tests, and beta testing, you can ensure that your cryptocurrency app is robust and user-friendly. Additionally, setting up a CI pipeline helps maintain code quality and streamline the development process.

**CHAPTER 8**

**SNAPSHOTS**

**** Fig.1: Display of all coins Fig.2 : Graph representing the coin

****Fig.3 : Favorites Fig.4 : Dark modes

****

Fig.5 : App logo

**CHAPTER 9**

**CONCLUSION**

Developing a cryptocurrency app using Android Studio involves a structured approach from conception through deployment. Initially, it’s essential to gather and analyze user requirements to understand what features are necessary, such as secure user authentication, wallet management, and real-time cryptocurrency market data. System design follows, where a scalable and secure architecture is crafted to handle these requirements efficiently. During implementation, Android Studio’s comprehensive development tools are utilized to build a robust and intuitive user interface, integrate with blockchain technologies, and ensure data security. Testing is a critical phase that includes unit tests to validate individual components, instrumentation tests to ensure UI functionality, and integration tests to verify that different parts of the app work together seamlessly. Beta testing with real users can further identify any issues and gather feedback for improvements. Finally, the app is prepared for deployment by generating a signed APK and submitting it to the Google Play Store. Ongoing maintenance and updates are crucial to address any emerging issues, enhance features, and ensure continued compatibility with evolving technologies. By meticulously following these steps, developers can deliver a high-quality cryptocurrency app that meets user expectations and stands out in a competitive market.

**FUTURE ENHANCEMENTS**

1. Enhanced Security Features
2. Improved User Interface (UI) and Experience (UX)
3. Integration with Decentralized Finance (DeFi)
4. Advanced Analytics and Insights
5. Expanded Cryptocurrency Support
6. Integration with Traditional Financial Systems
7. Enhanced Customer Support
8. Regulatory Compliance and Transparency

**REFERENCES**

1. **"Android Programming: The Big Nerd Ranch Guide"** by Bill Phillips, Chris Stewart, and Kristin Marsicano
2. **"Android Development with Kotlin"** by Marcin Moskala and Igor Wojda
3. **Google’s Android Developer Documentation**
4. **"Building a Cryptocurrency Portfolio App with Android Studio"** on Medium
5. **"How to Integrate Cryptocurrency APIs into Your Android App"** by DZone
6. **"Android Development for Beginners"** on Udacity
7. **"Kotlin for Android Development"** on Coursera