A PROJECT REPORT

On

ZeroByte

(Block-Chain Based File Storage App)

Submitted by

Mr. Abheek Shailesh Shah

In partial fulfilment for the award of the degree Of

BACHELOR OF SCIENCE

In

COMPUTER SCIENCE

Under the Guidance of

Mrs. Vinita Gupta

Department of Computer Science

NES Ratnam College of Arts, Science & Commerce

(Sem VI)

(2024-2025)

'A' GRADE
NAAC Re-accredited

NATIONAL EDUCATION SOCIETY'S

Ratnam College of Arts, Science & Commerce
Bhandup, Mumbai – 400 078.

Department of Computer Science

CERTIFICATE

Class: ***Roll / Seat No.:*** ***Year:***

This is to certify that Mr. Abheek Shailesh Shah has satisfactorily completed the project ZeroByte to be submitted in the partial fulfilment of the 3 years Degree Course Bachelor in Computer Science for the University of Mumbai for the year 2024 to2025.

*Place:*_____

*Date:*_____

_____

Project Guide

_____

In-charge,
Dept. of Computer Science

_____

Signature of Examiner with Date

DECLARATION

I, Abheek Shailesh Shah, hereby declare that the project entitled "ZeroByte" submitted in the partial fulfilment for the award of **Bachelor of Science** in **Computer Science** during the academic year **2024 – 2025** is my original work and the project has not formed the basis for the award of any degree, associateship, fellowship or any other similar titles.

Signature of the Student:

Place:

Date:

Acknowledgement

To list who all have helped me is difficult because they are so numerous and the depth is so enormous.

I would like to acknowledge the following as being idealistic channels and fresh dimensions inthe completion of this project.

I take this opportunity to thank the **University of Mumbai** for giving me a chance to do thisproject.

I would like to thank my **Principal, Dr. Vinita Dhulia** for providing the necessary facilities required for completion of this project.

I would also like to express my sincere gratitude towards my **Project guide Mrs. Vinita Gupta** whose guidance and care made the project successful.

I would like to thank my college, for having provided various reference books and magazinesrelated to my project.

Lastly, I would like to thank each and every person who directly or indirectly helped me in thecompletion of the project, especially M**y Parents and Peers** who supported me throughout my project.

Date:

Mr. Abheek Shailesh Shah

ZeroByte

Block-Chain Based File Storage App

Abheek Shah | TYCS

INDEX

Preliminary Investigation

INTRODUCTION

In today's digital era, **data security and storage reliability** have become major concerns for individuals and businesses alike. Traditional cloud storage solutions, such as **Google Drive and Dropbox**, rely on **centralized servers**, making them vulnerable to **cyber-attacks, data breaches, and unauthorized access**. These security risks have led to the emergence of **blockchain-based storage solutions**, which provide **decentralization, security, and transparency** as an alternative.

The **Blockchain-Based File Storage System App** is designed to **leverage the power of InterPlanetary File System (IPFS) and Blockchain Technology** to store and retrieve files securely. Unlike conventional cloud storage, which relies on a **single trusted entity**, this system ensures that files are **stored in a decentralized manner**, reducing reliance on a single point of failure. Additionally, it integrates **Firebase Authentication**, allowing users to securely log in using **Email/Password, Google Sign-In, or Anonymous Authentication**. This ensures secure access control **without compromising user privacy**.

The **key differentiator** of this system is its **decentralized nature**. Unlike traditional storage systems, which require users to trust **third-party providers**, the proposed system **encrypts, fragments, and distributes files across a decentralized network**. This approach guarantees:

- **Enhanced security** against unauthorized access.
- **Faster retrieval speeds** due to distributed file access.
- **Cost-effective storage** by utilizing decentralized infrastructure.
-

Users can **upload files**, retrieve them using **unique IPFS hashes**, and verify data integrity using **blockchain mechanisms**. The integration of **IPFS for storage and Firebase for authentication** ensures that the app remains **secure, decentralized, and user-friendly**.

This application is **particularly useful** for:

- **Privacy-conscious individuals** looking for a **secure storage alternative**.
- **Businesses handling sensitive data**, such as legal firms or healthcare institutions.
- **Researchers and developers** needing censorship-resistant storage.
- **Blockchain enthusiasts** who want to experience decentralized storage firsthand.
-

By combining **blockchain and cloud authentication**, this project bridges the gap between **advanced security mechanisms and practical usability**. The **open-source nature** of the project ensures **continuous improvement** while giving users full control over their **data privacy and ownership**.

This project aims to demonstrate the **real-world application of blockchain technology** in the field of file storage. By providing a **fully functional, decentralized, and easy-to-use storage solution**, the **Blockchain-Based File Storage System App** empowers users to **store and access their data without relying on centralized storage providers**.

OVERVIEW

ZeroByte is an Android application that allows users to **store and retrieve files securely** using **IPFS and blockchain technology**. It provides **decentralized storage** to eliminate the risks of **data loss and unauthorized access**. The app integrates **Firebase Authentication**, enabling users to log in via **Email/Password, Google Sign-In, or Anonymous Authentication**. Users can **upload files to IPFS**, retrieve them using **unique hashes (CIDs)**, check the **IPFS server status**, and **securely log out**. This app serves as an alternative to **centralized cloud storage** solutions.

Key Features:

1. **Decentralized File Storage:**
   Traditional cloud storage services rely on centralized servers, which are prone to hacking, data breaches, and single points of failure. Our app leverages InterPlanetary File System (IPFS) to store files in a distributed manner, eliminating the need for central servers. This ensures higher security, availability, and censorship resistance while giving users complete control over their data.
2. **Multiple Authentication Options:**
   Users can securely access the app using three authentication methods:
   - Email/Password Authentication (for registered users).
   - Google Sign-In (for easy and quick login via Google accounts).
   - Anonymous Authentication (allowing users to access storage without providing personal details).

     This flexibility ensures a secure and personalized user experience while maintaining privacy protection.
3. **File Upload & Retrieval:**
   Users can upload files directly to IPFS, where each file is assigned a unique Content Identifier (CID). This CID acts as a reference that allows users to retrieve files from any device without relying on centralized storage providers.
4. **Security & Data Integrity:**
   The decentralized nature of IPFS makes files tamper-proof and immutable, ensuring that no unauthorized changes can be made. Data integrity is maintained as every file is stored in multiple nodes across the network, making it resistant to data loss and cyber threats.
5. **User-Friendly Interface:**
   The app features a modern, minimalist UI with a clean color palette (white, grey, black, and red), ensuring an intuitive experience. Simple navigation and easy access to file upload, retrieval, and authentication options make it accessible to users of all levels.

OBJECTIVES

In an era where data privacy and security concerns are at an all-time high, traditional cloud storage solutions often fall short due to their reliance on centralized servers. **ZeroByte** aims to **overcome these challenges** by leveraging blockchain and IPFS technology to provide a **secure, decentralized, and efficient file storage solution**. The key objectives of this project include:

1. **Provide a Decentralized Alternative to Centralized Cloud Storage Solutions**
   The primary goal of this project is to **eliminate dependence on centralized storage providers** like Google Drive and Dropbox, which pose security and privacy risks. By using **IPFS (InterPlanetary File System)**, files are stored in a **distributed network** rather than a single server, reducing vulnerabilities associated with **data breaches, censorship, and server failures**.
2. **Ensure Secure and Encrypted File Storage Using IPFS**
   The app guarantees that **files are securely stored and accessed** via **unique Content Identifiers (CIDs)**, ensuring **tamper-proof and immutable storage**. Since IPFS distributes files across multiple nodes, unauthorized modifications or data corruption become virtually impossible.
3. **Offer Multiple Authentication Methods for User Access Control**
   Security and **ease of access** are core priorities of this system. The app integrates **Firebase Authentication**, allowing users to **log in through multiple methods**, including:
   - **Email/Password Authentication** for users who want a personalized, secure login.
   - **Google Sign-In** for those who prefer a seamless, quick authentication process.
   - **Anonymous Authentication** for privacy-focused users who do not wish to provide personal credentials.

     This ensures **flexibility and inclusivity** for a wide range of users.
4. **Improve Data Privacy by Eliminating Third-Party Control Over User Data**
   Unlike traditional cloud storage solutions that require users to **trust third-party providers**, this system ensures **users retain complete ownership** of their files. With blockchain and IPFS, **data is not stored on a central server**, meaning **no external entity** has access to users' private files or personal information.
5. **Develop an Open-Source, Accessible, and Efficient Storage System**
   The project aims to be **fully open-source**, allowing developers and tech enthusiasts to **contribute, modify, and enhance** the system. This ensures **continuous improvements**, wider adoption, and a **cost-effective** storage solution accessible to everyone.
6. **Enhance Data Availability and Reliability Through Decentralization**
   Traditional cloud storage providers are prone to **downtime, data loss, and access restrictions** due to centralized failures. By utilizing **IPFS and blockchain**, the app ensures **continuous availability of files**, even if certain nodes go offline. This enhances **data reliability, availability, and fault tolerance**, making it a **robust and scalable** storage solution.

SCOPE

ZeroByte is an innovative Android application that leverages **IPFS (InterPlanetary File System) and Firebase Authentication** to provide **a secure, decentralized, and user-friendly file storage solution**. Unlike traditional cloud services, this system **eliminates reliance on centralized storage providers**, ensuring **enhanced privacy, security, and data integrity**.

Scope of the Project

1. **User Authentication and Secure Access**
    - Users will be able to **sign in** using three authentication methods:
        - **Email/Password Authentication** for registered users.
        - **Google Sign-In** for seamless login.
        - **Anonymous Authentication** for privacy-conscious users.
    - Firebase Authentication will be integrated to **manage user credentials securely**.
2. **Decentralized File Storage & Retrieval**
    - Users can **upload files** to the **IPFS network** through **Infura's IPFS API**.
    - Each uploaded file will generate a **unique Content Identifier (CID)** that serves as a permanent reference for retrieval.
    - Users can **retrieve their files** by entering the CID, ensuring **efficient access** without centralized dependencies.
3. **IPFS Node Connectivity Check**
    - A built-in feature allows users to **check the status of the IPFS node**, ensuring that the decentralized network is active before uploading or retrieving files.
4. **Secure Logout Mechanism**
    - Users can log out securely, preventing unauthorized access to their stored data.

Technology Stack

- **Android Studio (Java)** – Primary framework for Android development.
- **Firebase Authentication** – Ensures secure and flexible user authentication.
- **IPFS (Infura Node)** – Decentralized file storage mechanism.

Limitations & Exclusions

- The system **does not** integrate with **traditional cloud storage solutions (Google Drive, Dropbox, etc.)**, as the goal is to promote **purely decentralized storage**.
- **Smart contract integration** for access control is not part of the initial implementation but may be considered in future updates.
- **End-to-end encryption** will be **a future enhancement**, ensuring further security improvements.

This project serves as a **proof-of-concept**, demonstrating **the viability of blockchain-based file storage** and setting a foundation for future improvements such as **private IPFS networks, on-chain authentication, and advanced cryptographic encryption mechanisms**.

FEATURES

ZeroByte: Secure, Decentralized File Storage on Blockchain

ZeroByte is an Android app that revolutionizes file storage by combining blockchain technology with IPFS (InterPlanetary File System) for secure, tamper-proof, and decentralized file management. Built with Java and Firebase Authentication, it offers a minimalist yet powerful solution for users who prioritize privacy and data integrity.

Key Features:

1. **Secure Authentication**

   ZeroByte supports multiple login methods, including Email/Password, Google Sign-In, and Anonymous login via Firebase. This ensures flexibility while maintaining robust security.

2. **IPFS-Powered Storage**

   Files are stored on IPFS, a peer-to-peer decentralized network, ensuring no single point of failure. Each uploaded file generates a unique Content Identifier (CID), guaranteeing authenticity and preventing unauthorized modifications.

3. **Virus Scanning for Safe Downloads**

   Before downloading, files are scanned via VirusTotal. If malware is detected, users receive a warning and can choose whether to proceed, adding an extra layer of security.

4. **Real-Time IPFS Status Checks**

   Users can verify if their local or remote IPFS node is online before uploading or downloading files, ensuring smooth operations.

5. **Simple & Intuitive Workflow**
   - **Upload:** Select a file, store it on IPFS, and receive a retrievable CID.
   - **Download:** Enter a CID to fetch files securely from the IPFS network.

6. **Modern UI with Minimalist Design**

   The app features a sleek, user-friendly interface with a monochrome theme (#FFFFFF, #D3D3D3, #000000) for clarity and ease of use.

7. **Future-Ready Upgrades**

   Planned enhancements include file encryption, storage management tools (view/delete), and support for multi-format previews.

8.

Why ZeroByte?

Unlike traditional cloud storage, ZeroByte eliminates reliance on centralized servers, reducing risks of data breaches. Its integration with blockchain and IPFS ensures files remain immutable and globally accessible without third-party control.

Ideal for privacy-conscious users, developers, and businesses, ZeroByte redefines secure file storage with cutting-edge decentralization.

RESOURCES

ZeroByte: Project Resources

ZeroByte is built using a robust tech stack designed for security, decentralization, and performance. Here are the key components powering our blockchain-based file storage solution:

Core Infrastructure

- IPFS (InterPlanetary File System) – The backbone for decentralized storage, using both local nodes for testing and Infura's IPFS API for production. Files are hashed into unique CIDs (Content Identifiers) to ensure tamper-proof retrieval.
- Firebase Authentication – Handles secure user logins via email/password, Google Sign-In, and anonymous sessions. Firebase Realtime Database (optional) stores metadata like saved file CIDs.

Development & Security

- Android Studio – Primary IDE for Java-based development, with Kotlin support planned for future updates.
- IPFS-Java (ipfs-http-client) – Enables direct interaction with IPFS nodes.
- Retrofit – Processes API calls to Infura IPFS and VirusTotal for file scanning.
- VirusTotal Integration – Scans downloaded files for malware before allowing local storage, adding a critical security layer.

UI/UX & Performance

- Material Design Components – Ensures a clean, intuitive interface with a monochrome theme (#FFFFFF, #D3D3D3, #000000).
- Glide/Picasso – Optimizes image loading for future file previews.
- Android Profiler & Logcat – Monitors app performance and debugs issues in real time.

Deployment & Future Scaling

- Google Play Store – Main distribution channel, using Android App Bundles (AAB) for efficient delivery.
- Firebase App Distribution – Facilitates beta testing before public release.
- Blockchain Expansion (Future) – Potential integration with Ethereum/Solana smart contracts for access control and Filecoin for incentivized storage.

Why This Stack?

ZeroByte combines decentralized storage (IPFS) with enterprise-grade security (Firebase, VirusTotal) while maintaining a user-friendly experience. The modular architecture allows seamless upgrades, from encryption to full blockchain integration.

By leveraging these technologies, ZeroByte delivers a secure, private, and scalable alternative to traditional cloud storage—all within a lightweight Android app.

SYSTEM REQUIREMNTS

Minimum Supported Devices

- **Android OS:** Android 8.0 (Oreo) or later (API level 26+)
- **RAM:** 2GB recommended (1GB may work but with potential instability)
- **Storage:** 500MB+ free space for app and cached files
- **Processor:** Any modern ARM-based chip (no high-end requirements)
- **Network:** Stable internet (Wi-Fi or 4G+) for IPFS/VirusTotal operations

Key Dependencies

- **Firebase Authentication** (Google/email login)
- **IPFS Node/Infura API** (decentralized file storage)
- **VirusTotal API** (malware scanning)
- **AndroidX Libraries** (core, lifecycle, appcompat)

Required Permissions

- INTERNET (for IPFS/VirusTotal API calls)
- READ_EXTERNAL_STORAGE (to access user-selected files)

Optimization Notes

- **Lightweight Operations:** Efficient file handling with OkHttp/Retrofit
- **Error Handling:** Graceful timeouts for slow networks
- **Screen Compatibility:** Works on all standard resolutions (480x800+)

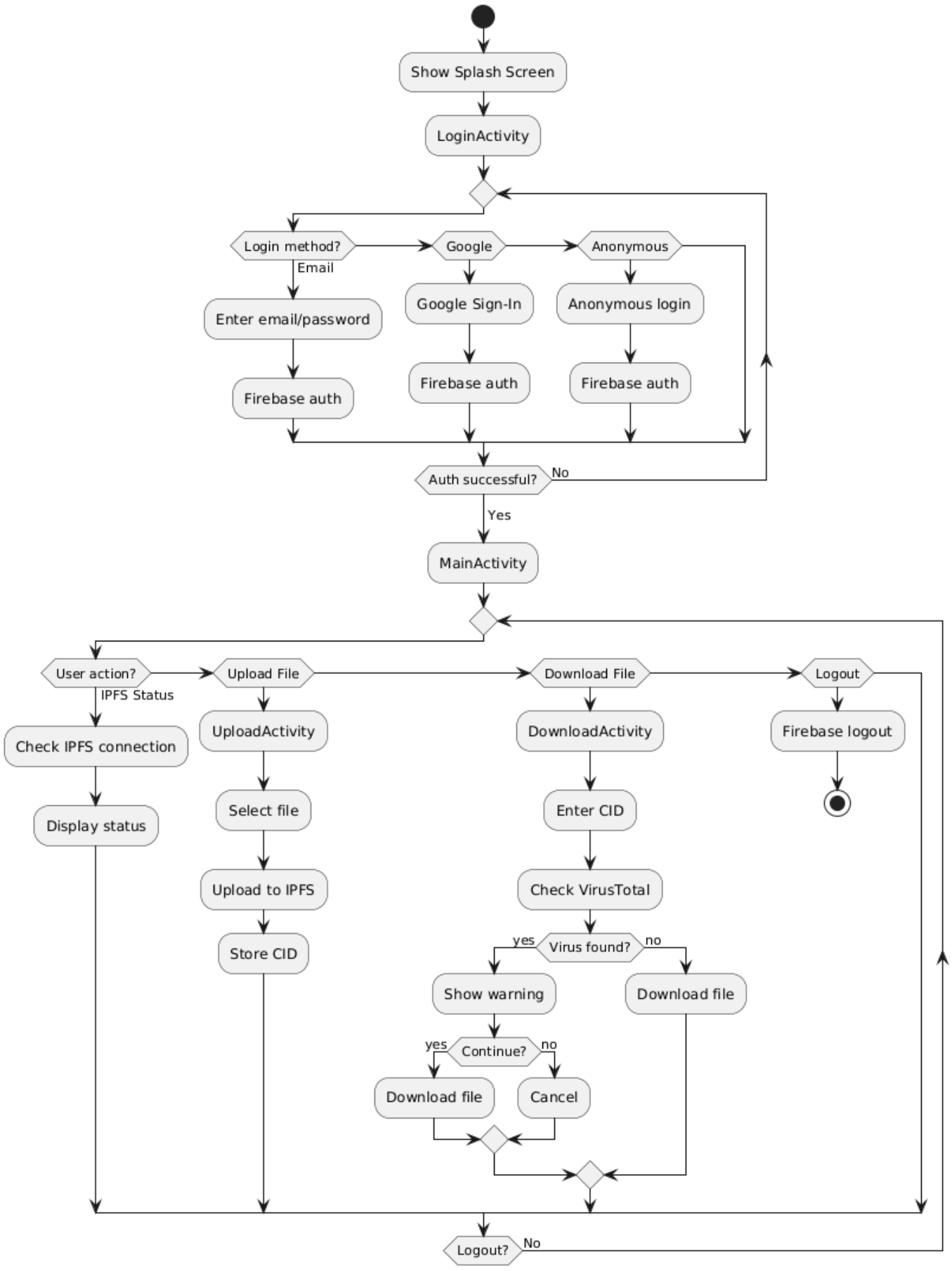
Testing Recommended: Emulate low-RAM devices (2GB or less) to ensure stability.

System Analysis

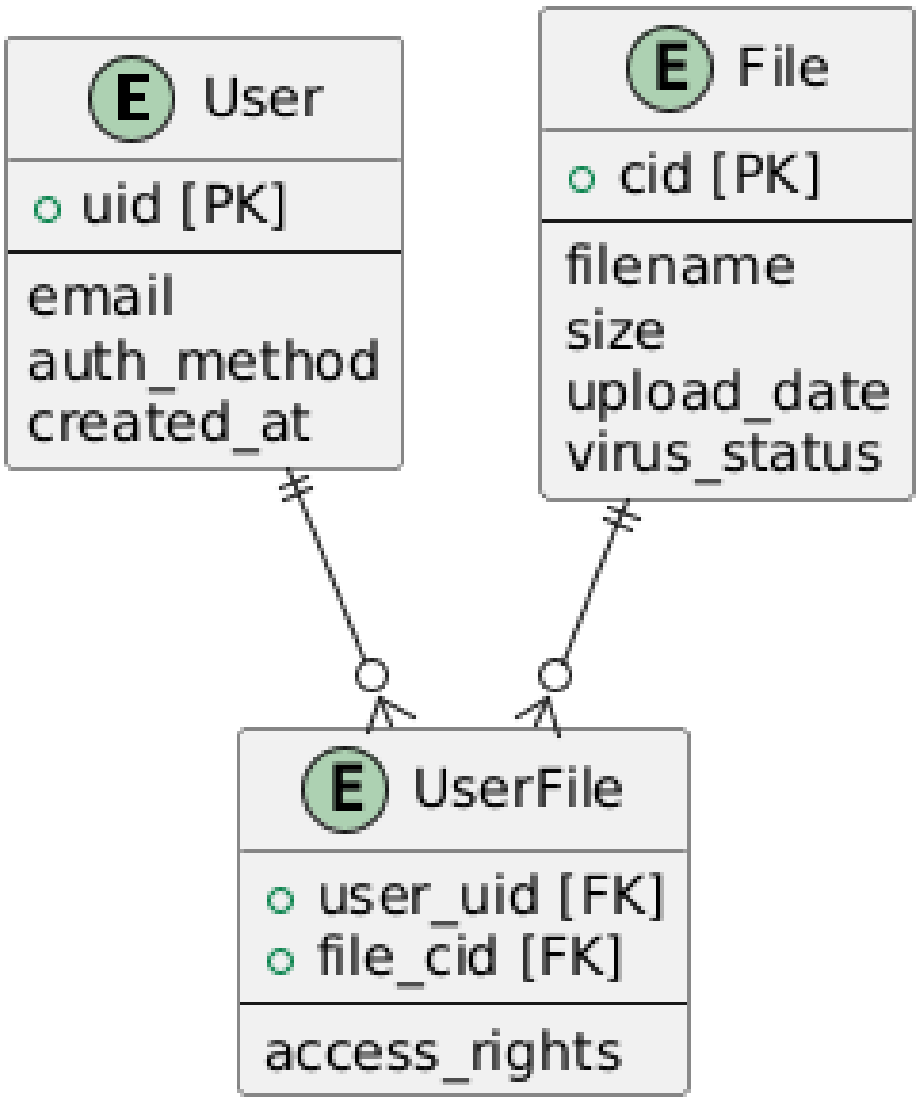
&

Design

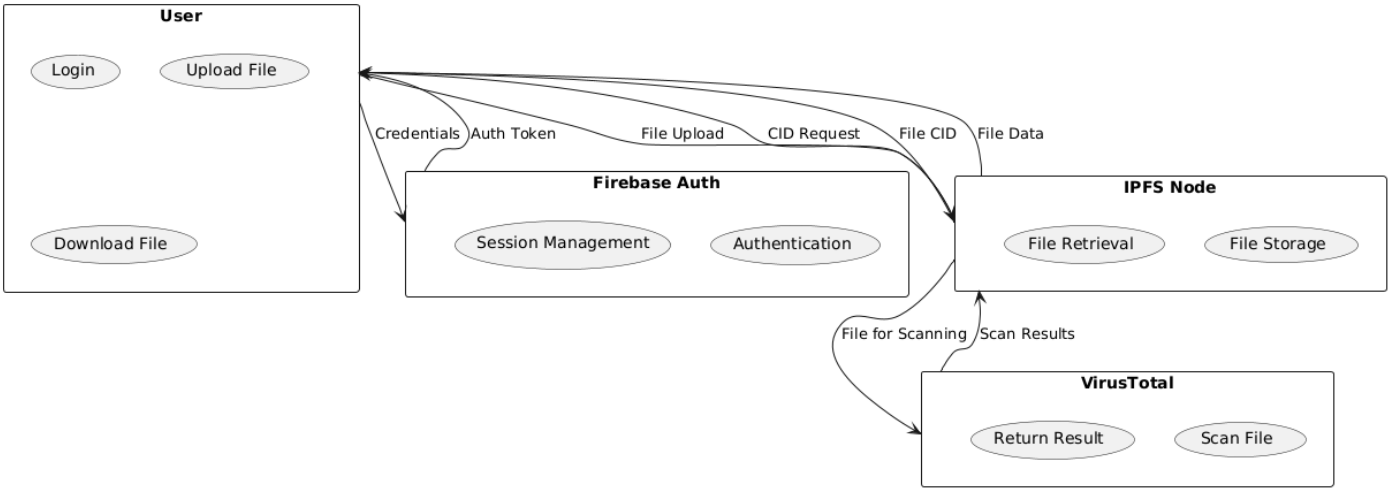
FLOW CHART


Show Splash Screen
LoginActivity
Login method?
Email
Google
Anonymous
Enter email/password
Google Sign-In
Anonymous login
Firebase auth
Firebase auth
Firebase auth
Auth successful?
No
Yes
MainActivity
User action?
IPFS Status
Upload File
Download File
Logout
Check IPFS connection
UploadActivity
DownloadActivity
Firebase logout
Display status
Select file
Enter CID
Upload to IPFS
Check VirusTotal
Store CID
yes
Virus found?
no
Show warning
Download file
yes
Continue?
no
Download file
Cancel
Logout?
No

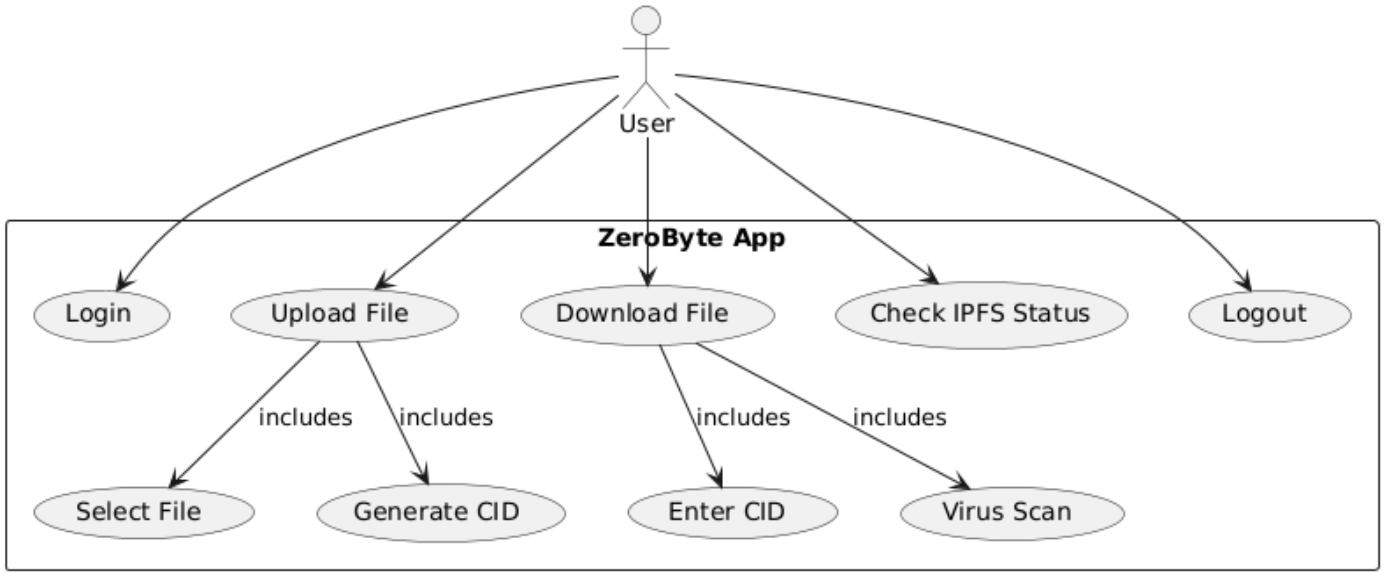
ER DIAGRAM


User
uid [PK]
email
auth_method
created_at
File
cid [PK]
filename
size
upload_date
virus_status
UserFile
user_uid [FK]
file_cid [FK]
access_rights

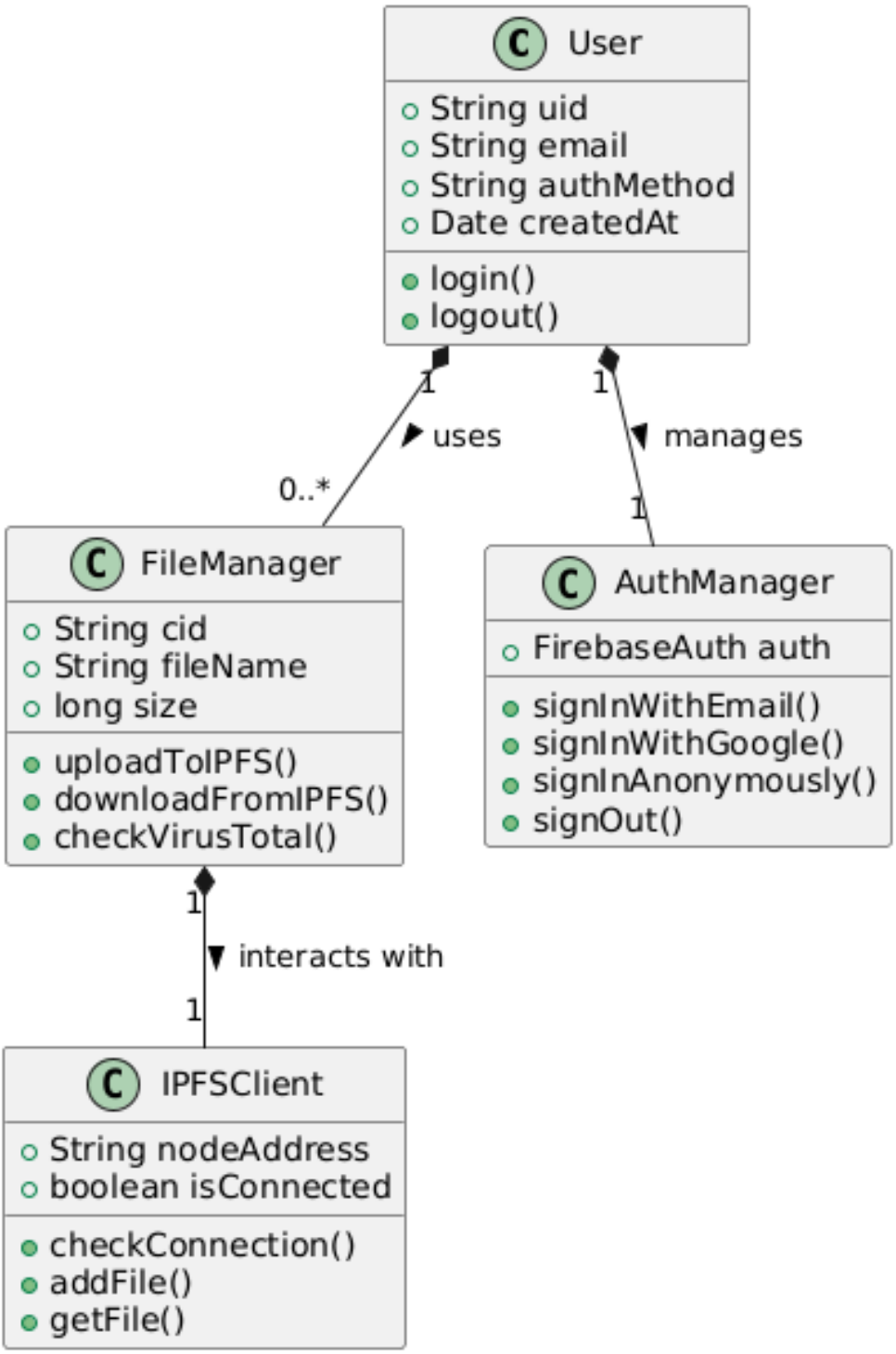
DATA FLOW DIAGRAM



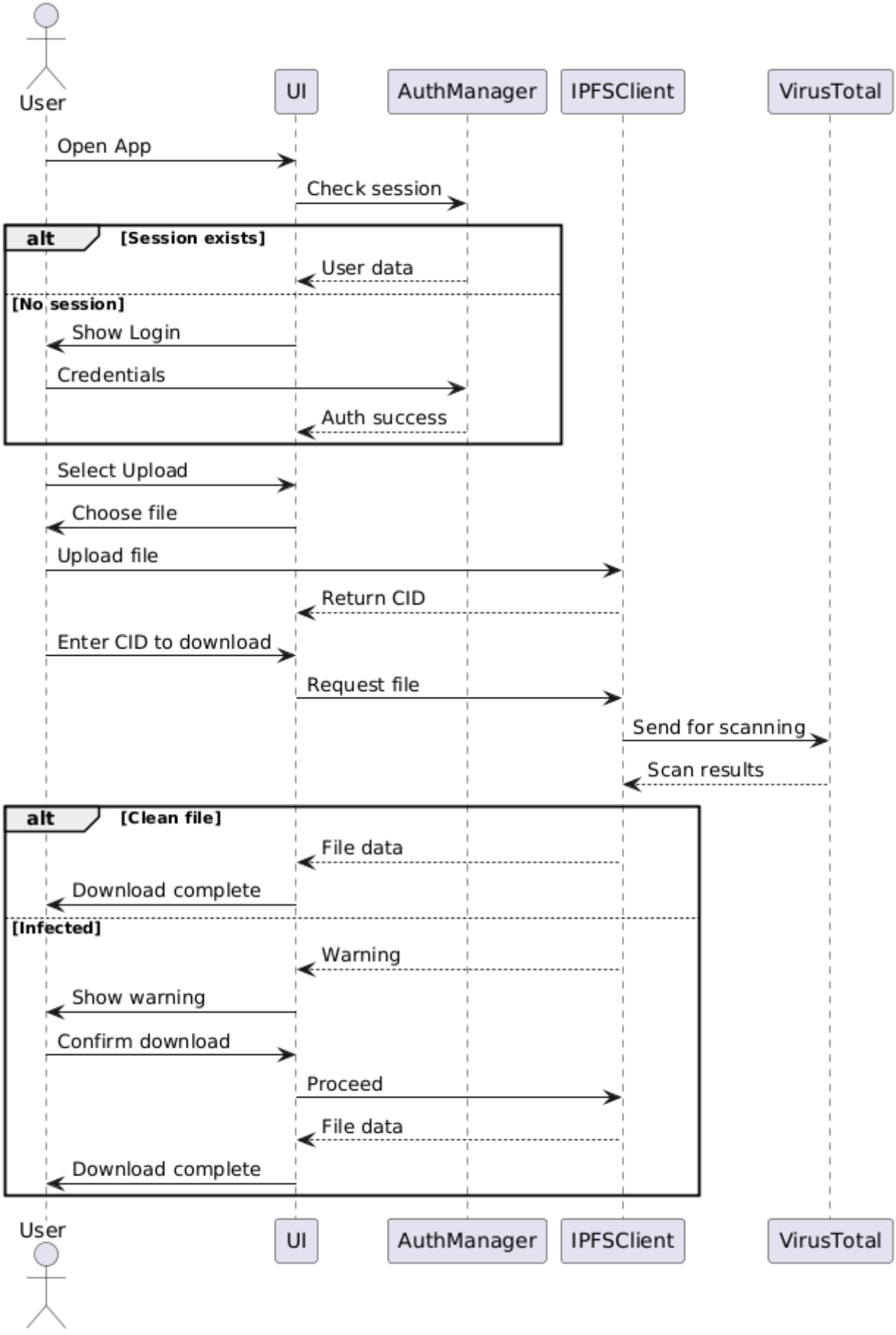
USE CASE DIAGRAM



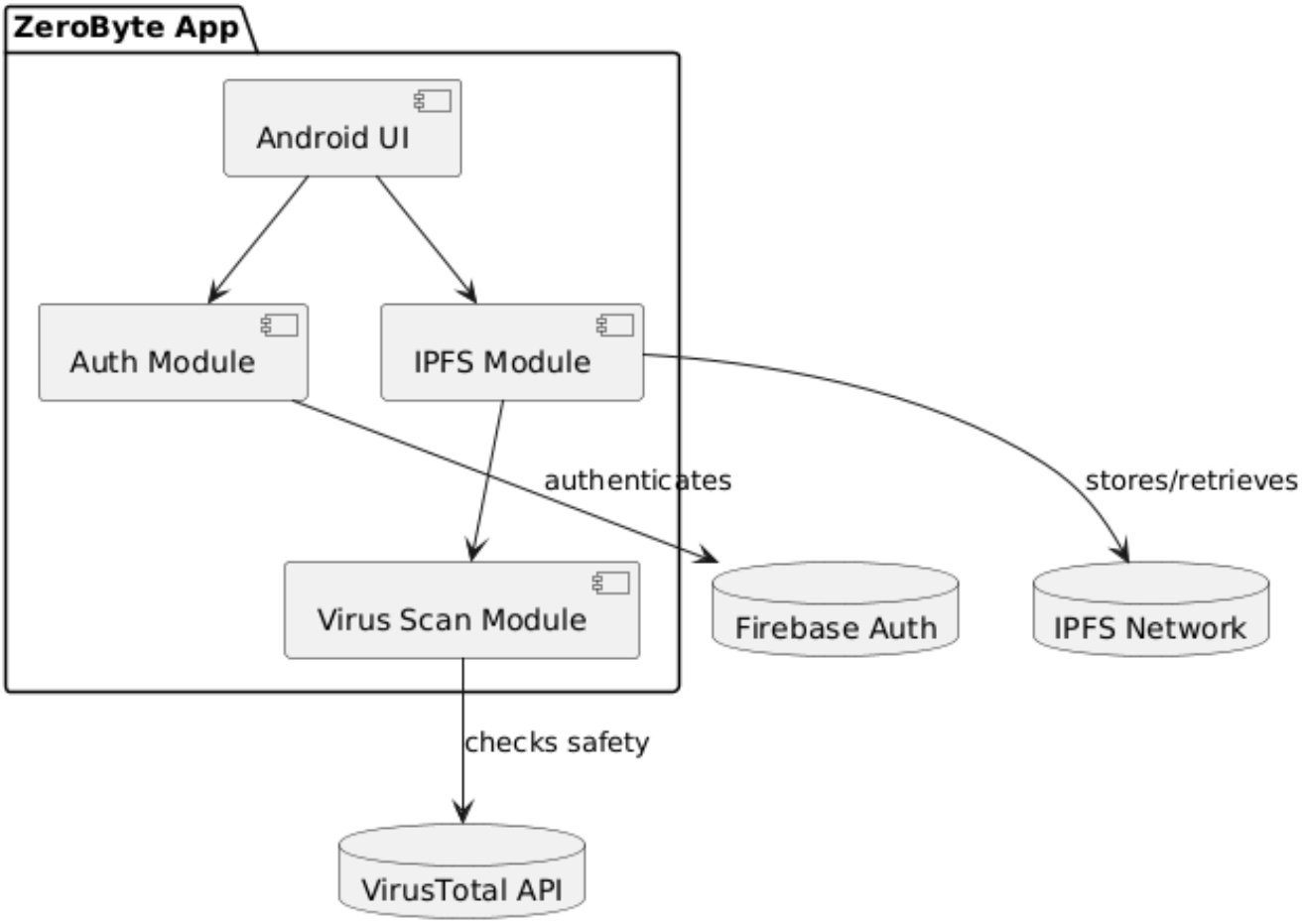
CLASS DIAGRAM



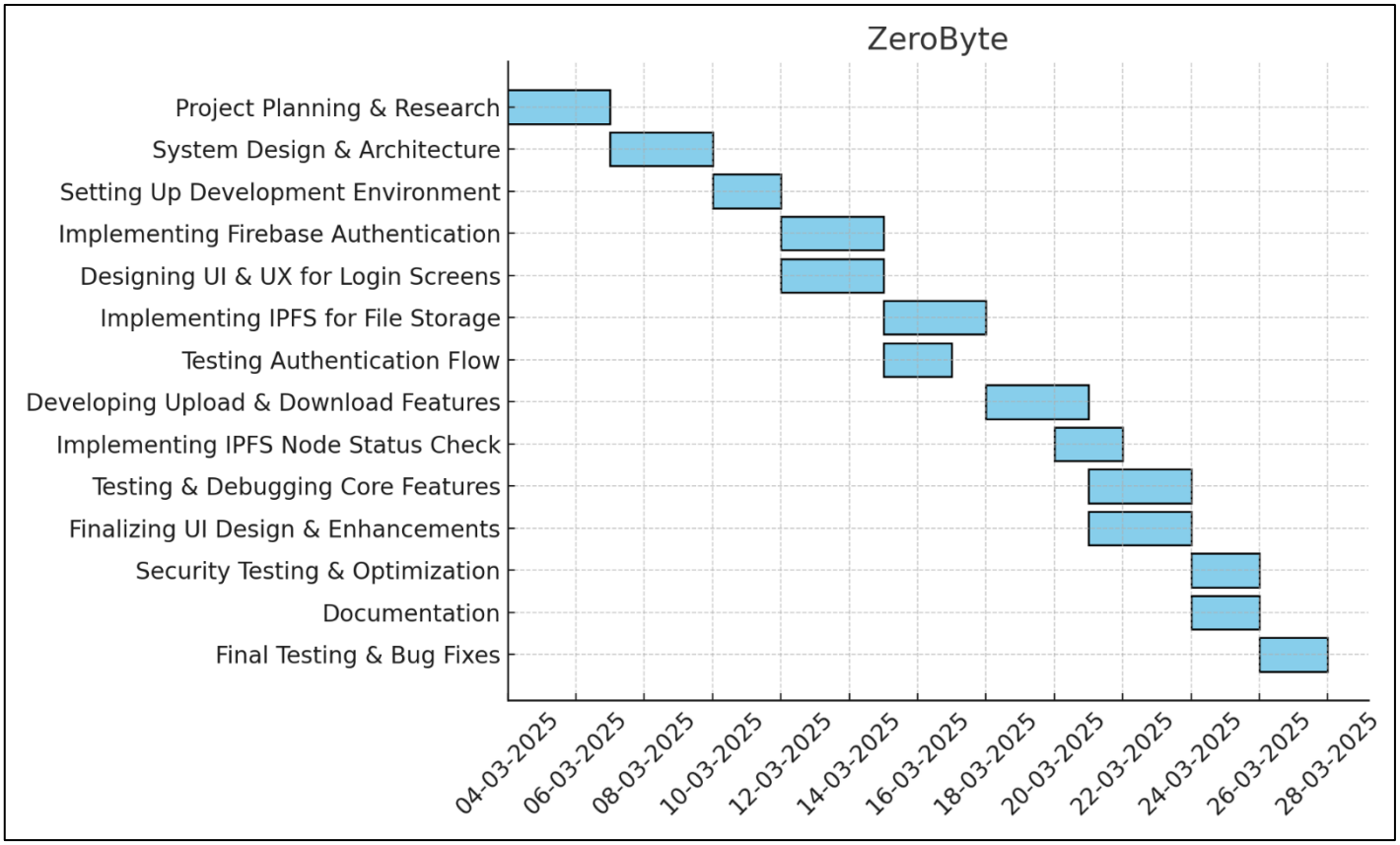
SEQUENCE DIAGRAM



COMPONENT DIAGRAM



GANTT CHART


ZeroByte
Project Planning & Research
System Design & Architecture
Setting Up Development Environment
Implementing Firebase Authentication
Designing UI & UX for Login Screens
Implementing IPFS for File Storage
Testing Authentication Flow
Developing Upload & Download Features
Implementing IPFS Node Status Check
Testing & Debugging Core Features
Finalizing UI Design & Enhancements
Security Testing & Optimization
Documentation
Final Testing & Bug Fixes
04-03-2025
06-03-2025
08-03-2025
10-03-2025
12-03-2025
14-03-2025
16-03-2025
18-03-2025
20-03-2025
22-03-2025
24-03-2025
26-03-2025
28-03-2025

App Structure & Programming

SOURCE CODE

AndroidManifest.xml

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    package="com.example.zerobyte">

    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />

    <application
        android:usesCleartextTraffic="true"
        android:networkSecurityConfig="@xml/network_security_config"
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.Zerobyte"
        tools:targetApi="31">

        <activity
            android:name=".SplashActivity"
            android:theme="@style/Theme.Zerobyte"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

        <activity android:name=".LoginActivity" android:exported="true"/>
        <activity android:name=".MainActivity" android:exported="true"
android:theme="@style/Theme.Zerobyte"/>
        <activity android:name=".UploadActivity" android:exported="true"/>
        <activity android:name=".DownloadActivity" android:exported="true"/>
    </application>
</manifest>
```

SplashActivity.java

```
package com.example.zerobyte;
import android.content.Intent;
import android.os.Bundle;
import android.os.Handler;
import androidx.appcompat.app.AppCompatActivity;

public class SplashActivity extends AppCompatActivity {
    private static final int SPLASH_TIME_OUT = 2000;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_splash);

        new Handler().postDelayed(() -> {
            Intent intent = new Intent(SplashActivity.this, LoginActivity.class);
            startActivity(intent);
            finish();
        }, SPLASH_TIME_OUT);
    }
}
```

activity_splash.xml

```
<?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:gravity="center"
    android:background="@color/dark_black">
    <ImageView
        android:layout_width="125dp"
        android:layout_height="130dp"
        android:contentDescription="App Logo"
        android:src="@mipmap/ic_launcher" />
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="ZeroByte"
        android:textSize="24sp"
        android:textStyle="bold"
        android:textColor="@color/pure_white"
        android:paddingTop="20dp"/>
</LinearLayout>
```

LoginActivity.java

```
package com.example.zerobyte;
import android.content.Intent;
import android.os.Bundle;
import android.util.Log;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;
import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import com.google.android.gms.auth.api.signin.GoogleSignIn;
import com.google.android.gms.auth.api.signin.GoogleSignInAccount;
import com.google.android.gms.auth.api.signin.GoogleSignInClient;
import com.google.android.gms.auth.api.signin.GoogleSignInOptions;
import com.google.android.gms.common.api.ApiException;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.AuthCredential;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.auth.GoogleAuthProvider;

public class LoginActivity extends AppCompatActivity {
    private EditText etEmail, etPassword;
    private Button btnEmailLogin;
    private ImageView btnGoogleSignIn, btnAnonymousLogin;
    private TextView tvToggleSignUp;
    private FirebaseAuth mAuth;
    private GoogleSignInClient mGoogleSignInClient;
    private static final int RC_SIGN_IN = 100;
    private boolean isSignUpMode = false;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);
        mAuth = FirebaseAuth.getInstance();
        etEmail = findViewById(R.id.etEmail);
        etPassword = findViewById(R.id.etPassword);
        btnEmailLogin = findViewById(R.id.btnEmailLogin);
        btnGoogleSignIn = findViewById(R.id.btnGoogleSignIn);
        btnAnonymousLogin = findViewById(R.id.btnAnonymousLogin);
        tvToggleSignUp = findViewById(R.id.tvToggleSignUp);
        updateButtonText();
        GoogleSignInOptions gso = new GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT_SIGN_IN)
                .requestIdToken(getString(R.string.default_web_client_id))
                .requestEmail()
                .build();
        mGoogleSignInClient = GoogleSignIn.getClient(this, gso);
        btnEmailLogin.setOnClickListener(v -> {
            if (isSignUpMode) {
                signUpWithEmail();
```

```
        } else {
            emailLogin();
        }
    });
    btnGoogleSignIn.setOnClickListener(v -> googleSignIn());
    btnAnonymousLogin.setOnClickListener(v -> anonymousSignIn());
    tvToggleSignUp.setOnClickListener(v -> toggleSignUpMode());
}

private void emailLogin() {
    String email = etEmail.getText().toString().trim();
    String password = etPassword.getText().toString().trim();
    if (email.isEmpty() || password.isEmpty()) {
        Toast.makeText(this, "Please enter email and password", Toast.LENGTH_SHORT).show();
        return;
    }
    mAuth.signInWithEmailAndPassword(email, password)
            .addOnCompleteListener(this, task -> {
                if (task.isSuccessful()) {
                    navigateToMain();
                } else {
                    Toast.makeText(this, "Authentication failed.", Toast.LENGTH_SHORT).show();
                }
            });
}

private void signUpWithEmail() {
    String email = etEmail.getText().toString().trim();
    String password = etPassword.getText().toString().trim();
    if (email.isEmpty() || password.isEmpty()) {
        Toast.makeText(this, "Please enter email and password", Toast.LENGTH_SHORT).show();
        return;
    }
    mAuth.createUserWithEmailAndPassword(email, password)
            .addOnCompleteListener(this, task -> {
                if (task.isSuccessful()) {
                    navigateToMain();
                } else {
                    Toast.makeText(this, "Sign up failed.", Toast.LENGTH_SHORT).show();
                }
            });
}

private void googleSignIn() {
    Intent signInIntent = mGoogleSignInClient.getSignInIntent();
    startActivityForResult(signInIntent, RC_SIGN_IN);
}

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (requestCode == RC_SIGN_IN) {
        Task<GoogleSignInAccount> task = GoogleSignIn.getSignedInAccountFromIntent(data);
        try {
            GoogleSignInAccount account = task.getResult(ApiException.class);
```

```
            firebaseAuthWithGoogle(account.getIdToken());
        } catch (ApiException e) {
            Log.w("LoginActivity", "Google sign-in failed", e);
        }
    }
}

private void firebaseAuthWithGoogle(String idToken) {
    AuthCredential credential = GoogleAuthProvider.getCredential(idToken, null);
    mAuth.signInWithCredential(credential)
            .addOnCompleteListener(this, task -> {
                if (task.isSuccessful()) {
                    navigateToMain();
                } else {
                    Toast.makeText(this, "Google sign-in failed", Toast.LENGTH_SHORT).show();
                }
            });
}

private void anonymousSignIn() {
    mAuth.signInAnonymously()
            .addOnCompleteListener(this, task -> {
                if (task.isSuccessful()) {
                    navigateToMain();
                } else {
                    Toast.makeText(this, "Anonymous sign-in failed", Toast.LENGTH_SHORT).show();
                }
            });
}

private void toggleSignUpMode() {
    isSignUpMode = !isSignUpMode;
    updateButtonText();
}

private void updateButtonText() {
    if (isSignUpMode) {
        btnEmailLogin.setText("Sign Up");
        tvToggleSignUp.setText("Already have an account? Sign In");
    } else {
        btnEmailLogin.setText("Sign In");
        tvToggleSignUp.setText("Don't have an account? Sign Up");
    }
}

private void navigateToMain() {
    Intent intent = new Intent(LoginActivity.this, MainActivity.class);
    startActivity(intent);
    finish();
}
}
```

activity_login.xml

```
<?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:gravity="center"
   android:padding="24dp"
   android:background="@color/dark_black">
   <ImageView
      android:layout_width="120dp"
      android:layout_height="120dp"
      android:src="@mipmap/ic_launcher"
      android:contentDescription="@string/app_name"
      android:layout_gravity="center"/>
   <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="@string/app_name"
      android:textSize="24sp"
      android:textColor="@color/dark_black"
      android:textStyle="bold"
      android:layout_gravity="center"
      android:paddingTop="16dp"/>
   <EditText
      android:id="@+id/etEmail"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:hint="Enter Email"
      android:inputType="textEmailAddress"
      android:padding="12dp"
      android:background="@color/light_grey"
      android:textColor="@color/dark_black"
      android:textColorHint="@color/dark_black"
      android:drawableStart="@drawable/ic_email"
      android:drawablePadding="12dp"
      android:layout_marginTop="12dp"/>
   <EditText
      android:id="@+id/etPassword"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:hint="Enter Password"
      android:inputType="textPassword"
      android:padding="12dp"
      android:background="@color/light_grey"
      android:textColor="@color/dark_black"
      android:textColorHint="@color/dark_black"
      android:drawableStart="@drawable/ic_key"
      android:drawablePadding="12dp"
      android:layout_marginTop="8dp"/>
   <Button
      android:id="@+id/btnEmailLogin"
      android:layout_width="match_parent"
```

```
        android:layout_height="wrap_content"
        android:text="SignIn with Email"
        android:backgroundTint="@color/light_grey"
        android:textColor="@color/dark_black"
        android:layout_marginTop="12dp"/>
    <TextView
        android:id="@+id/tvToggleSignUp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Don't have an account? Sign Up"
        android:textColor="@color/pure_white"
        android:layout_marginBottom="12dp"
        android:layout_marginTop="12dp"
        android:clickable="true"
        android:focusable="true"
        android:textStyle="bold"/>
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="OR"
        android:layout_marginTop="10dp"
        android:layout_marginBottom="10dp"
        android:textColor="@color/pure_white"
        android:textAlignment="center"
        android:textSize="20dp"/>
    <ImageView
        android:id="@+id/btnGoogleSignIn"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:src="@drawable/ic_google"
        android:contentDescription="Sign in with Google"
        android:padding="12dp"
        android:background="?attr/selectableItemBackground"
        android:scaleType="centerInside"
        android:layout_marginTop="8dp"/>
    <ImageView
        android:id="@+id/btnAnonymousLogin"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:src="@drawable/ic_anonymous"
        android:contentDescription="Continue as Guest"
        android:padding="12dp"
        android:background="?attr/selectableItemBackground"
        android:scaleType="centerInside"
        android:backgroundTint="@color/pure_white"
        android:layout_marginTop="8dp"/>
    <ProgressBar
        android:id="@+id/progressBar"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:visibility="gone"
        android:layout_gravity="center"
        android:layout_marginTop="16dp"/>
</LinearLayout>
```

MainActivity.java

```
package com.example.zerobyte;

import android.content.Intent;
import android.os.Bundle;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;
import androidx.appcompat.app.AppCompatActivity;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;

public class MainActivity extends AppCompatActivity {
    private TextView tvWelcome;
    private EditText etIPAddress, etPortNumber;
    private Button btnUploadFile, btnDownloadFile, btnCheckIPFSStatus, btnLogout;
    private FirebaseAuth mAuth;
    private IPFSManager ipfsManager;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        mAuth = FirebaseAuth.getInstance();
        tvWelcome = findViewById(R.id.tvWelcome);
        etIPAddress = findViewById(R.id.etIPAddress);
        etPortNumber = findViewById(R.id.etPortNumber);
        btnUploadFile = findViewById(R.id.btnUploadFile);
        btnDownloadFile = findViewById(R.id.btnDownloadFile);
        btnCheckIPFSStatus = findViewById(R.id.btnCheckIPFSStatus);
        btnLogout = findViewById(R.id.btnLogout);

        FirebaseUser currentUser = mAuth.getCurrentUser();
        if (currentUser != null) {
            String welcomeMessage = "Welcome, " + currentUser.getEmail();
            tvWelcome.setText(welcomeMessage);
        } else {
            tvWelcome.setText("Welcome to ZeroByte");
        }

        btnUploadFile.setOnClickListener(v -> navigateToUploadActivity());
        btnDownloadFile.setOnClickListener(v -> navigateToDownloadActivity());
        btnCheckIPFSStatus.setOnClickListener(v -> checkIPFSStatus());
        btnLogout.setOnClickListener(v -> logoutUser());
    }

    private String getIPFSAddress() {
        String ip = etIPAddress.getText().toString().trim();
        String port = etPortNumber.getText().toString().trim();
        return "/ip4/" + ip + "/tcp/" + port;
    }

    private void navigateToUploadActivity() {
        String ipfsAddress = getIPFSAddress();
        if (ipfsAddress.isEmpty() || !isValidIPFSAddress(ipfsAddress)) {
```

```
        Toast.makeText(this, "Please enter valid IP and Port", Toast.LENGTH_SHORT).show();
        return;
    }
    Intent intent = new Intent(MainActivity.this, UploadActivity.class);
    intent.putExtra("IPFS_ADDRESS", ipfsAddress);
    startActivity(intent);
}

private void navigateToDownloadActivity() {
    String ipfsAddress = getIPFSAddress();
    if (ipfsAddress.isEmpty() || !isValidIPFSAddress(ipfsAddress)) {
        Toast.makeText(this, "Please enter valid IP and Port", Toast.LENGTH_SHORT).show();
        return;
    }
    Intent intent = new Intent(MainActivity.this, DownloadActivity.class);
    intent.putExtra("IPFS_ADDRESS", ipfsAddress);
    startActivity(intent);
}

private boolean isValidIPFSAddress(String address) {
    return address.startsWith("/ip4/") && address.contains("/tcp/");
}

private void checkIPFSStatus() {
    String ipfsAddress = getIPFSAddress();
    if (ipfsAddress.isEmpty() || !isValidIPFSAddress(ipfsAddress)) {
        Toast.makeText(this, "Please enter valid IP and Port", Toast.LENGTH_SHORT).show();
        return;
    }

    ipfsManager = new IPFSManager(this, ipfsAddress, new IPFSManager.IPFSInitListener() {
        @Override
        public void onIPFSInitialized() {
            ipfsManager.isIPFSOnline(new IPFSManager.IPFSStatusListener() {
                @Override
                public void onIPFSStatusChecked(boolean isOnline, String message) {
                    runOnUiThread(() -> {
                        Toast.makeText(
                                MainActivity.this,
                                isOnline ? "Online" : "Offline",
                                Toast.LENGTH_SHORT
                        ).show();
                    });
                }
            });
        }

        @Override
        public void onIPFSInitFailed(String error) {
            runOnUiThread(() -> Toast.makeText(
                    MainActivity.this,
                    "Offline",
                    Toast.LENGTH_SHORT
            ).show());
        }
    });
}
```

```
    private void logoutUser() {
        mAuth.signOut();
        Toast.makeText(this, "Logged out successfully", Toast.LENGTH_SHORT).show();
        startActivity(new Intent(MainActivity.this, LoginActivity.class));
        finish();
    }
}
```

activity_main.xml

```
<?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:gravity="center"
    android:background="@color/dark_black"
    android:padding="20dp">

    <TextView
        android:id="@+id/tvWelcome"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Welcome to ZeroByte"
        android:textSize="24sp"
        android:textColor="@color/pure_white"
        android:textStyle="bold"
        android:layout_marginBottom="20dp"/>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:layout_marginBottom="12dp">

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="/ip4/"
            android:textColor="@color/pure_white" />

        <EditText
            android:id="@+id/etIPAddress"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:hint="     IP ADDRESS"
            android:textColor="@color/pure_white"
            android:textColorHint="@color/light_grey" />

        <TextView
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="/tcp/"
        android:textColor="@color/pure_white" />

    <EditText
        android:id="@+id/etPortNumber"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:hint="      PORT NO"
        android:textColor="@color/pure_white"
        android:textColorHint="@color/light_grey" />
</LinearLayout>

<Button
    android:id="@+id/btnCheckIPFSStatus"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Check IPFS Server Status"
    android:backgroundTint="@color/light_grey"
    android:textColor="@color/dark_black"
    android:layout_marginBottom="12dp"/>

<Button
    android:id="@+id/btnUploadFile"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Upload File"
    android:backgroundTint="@color/light_grey"
    android:textColor="@color/dark_black"
    android:layout_marginBottom="12dp"/>

<Button
    android:id="@+id/btnDownloadFile"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Download File"
    android:backgroundTint="@color/light_grey"
    android:textColor="@color/dark_black"
    android:layout_marginBottom="12dp"/>

<Button
    android:id="@+id/btnLogout"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Logout"
    android:backgroundTint="@color/light_grey"
    android:textColor="@color/dark_black"/>
</LinearLayout>
```

IPFSManger.java

```
package com.example.zerobyte;

import android.content.ContentResolver;
import android.net.Uri;
import android.os.Handler;
import android.os.Looper;
import android.util.Log;
import java.io.File;
import java.io.FileOutputStream;
import java.io.InputStream;
import java.io.OutputStream;
import java.util.List;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import io.ipfs.api.IPFS;
import io.ipfs.api.MerkleNode;
import io.ipfs.api.NamedStreamable;
import io.ipfs.multihash.Multihash;
import io.ipfs.multiaddr.MultiAddress;

public class IPFSManager {
    private IPFS ipfs;
    private final ExecutorService executorService;
    private final Handler mainHandler;
    private final String ipfsAddress;

    public IPFSManager(Object context, String ipfsAddress, IPFSInitListener listener) {
        this.ipfsAddress = ipfsAddress;
        this.executorService = Executors.newFixedThreadPool(3);
        this.mainHandler = new Handler(Looper.getMainLooper());
        initializeIPFS(listener);
    }

    private void initializeIPFS(IPFSInitListener listener) {
        executorService.execute(() -> {
            try {
                ipfs = new IPFS(new MultiAddress(ipfsAddress));
                ipfs.refs.local();
                mainHandler.post(listener::onIPFSInitialized);
            } catch (Exception e) {
                mainHandler.post(() -> listener.onIPFSInitFailed(e.getMessage()));
            }
        });
    }

    public void uploadFileToIPFS(Uri fileUri, ContentResolver contentResolver, IPFSUploadListener
```

```
listener) {
        executorService.execute(() -> {
            File tempFile = null;
            try {
                InputStream inputStream = contentResolver.openInputStream(fileUri);
                if (inputStream == null) {
                    mainHandler.post(() -> listener.onUploadFailed("Cannot open file"));
                    return;
                }
                tempFile = File.createTempFile("ipfs_upload", ".tmp");
                try (OutputStream outputStream = new FileOutputStream(tempFile)) {
                    byte[] buffer = new byte[8192];
                    int bytesRead;
                    while ((bytesRead = inputStream.read(buffer)) != -1) {
                        outputStream.write(buffer, 0, bytesRead);
                    }
                }
                NamedStreamable.FileWrapper fileWrapper = new NamedStreamable.FileWrapper(tempFile);
                List<MerkleNode> result = ipfs.add(fileWrapper);
                String cid = result.get(0).hash.toBase58();
                mainHandler.post(() -> listener.onUploadSuccess(cid));
            } catch (Exception e) {
                mainHandler.post(() -> listener.onUploadFailed(e.getMessage()));
            } finally {
                if (tempFile != null && tempFile.exists()) {
                    tempFile.delete();
                }
            }
        });
    }

    public void downloadFileFromIPFS(String cid, Uri saveLocationUri,
                                     ContentResolver contentResolver, IPFSDownloadListener listener) {
        executorService.execute(() -> {
            try (InputStream inputStream = ipfs.catStream(Multihash.fromBase58(cid));
                 OutputStream outputStream = contentResolver.openOutputStream(saveLocationUri)) {
                if (outputStream == null) {
                    mainHandler.post(() -> listener.onDownloadFailed("Cannot create output file"));
                    return;
                }
                byte[] buffer = new byte[8192];
                int bytesRead;
                while ((bytesRead = inputStream.read(buffer)) != -1) {
                    outputStream.write(buffer, 0, bytesRead);
                }
                mainHandler.post(() -> listener.onDownloadSuccess(null));
            } catch (Exception e) {
                mainHandler.post(() -> listener.onDownloadFailed(e.getMessage()));
```

```
            }
        });
    }

    public void isIPFSOnline(IPFSStatusListener listener) {
        executorService.execute(() -> {
            try {
                ipfs.refs.local();
                mainHandler.post(() -> listener.onIPFSStatusChecked(true, "Connected"));
            } catch (Exception e) {
                mainHandler.post(() -> listener.onIPFSStatusChecked(false,
                        e.getMessage() != null ? e.getMessage() : "Connection failed"));
            }
        });
    }

    public interface IPFSInitListener {
        void onIPFSInitialized();
        void onIPFSInitFailed(String error);
    }

    public interface IPFSStatusListener {
        void onIPFSStatusChecked(boolean isOnline, String message);
    }

    public interface IPFSUploadListener {
        void onUploadSuccess(String cid);
        void onUploadFailed(String error);
    }

    public interface IPFSDownloadListener {
        void onDownloadSuccess(byte[] fileData);
        void onDownloadFailed(String error);
    }

    public void shutdown() {
        executorService.shutdownNow();
    }
}
```

UploadActvity.java

```
package com.example.zerobyte;

import android.content.ClipData;
import android.content.ClipboardManager;
import android.content.ContentResolver;
import android.content.Context;
import android.content.Intent;
import android.database.Cursor;
import android.net.Uri;
import android.os.Bundle;
import android.provider.OpenableColumns;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;

public class UploadActivity extends AppCompatActivity {
    private static final int PICK_FILE_REQUEST = 1;
    private Button btnUpload, btnCheckIPFS;
    private EditText txtFileStatus;
    private Uri fileUri;
    private IPFSManager ipfsManager;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_upload);

        btnUpload = findViewById(R.id.btnUpload);
        btnCheckIPFS = findViewById(R.id.btnCheckIPFS);
        txtFileStatus = findViewById(R.id.txtFileStatus);

        String ipfsAddress = getIntent().getStringExtra("IPFS_ADDRESS");
        if (ipfsAddress == null || ipfsAddress.isEmpty() || !isValidIPFSAddress(ipfsAddress)) {
            Toast.makeText(this, "Invalid IPFS address", Toast.LENGTH_SHORT).show();
            finish();
            return;
        }

        ipfsManager = new IPFSManager(this, ipfsAddress, new IPFSManager.IPFSInitListener() {
            @Override
```

```
            public void onIPFSInitialized() {
            }

            @Override
            public void onIPFSInitFailed(String error) {
                runOnUiThread(() -> Toast.makeText(UploadActivity.this, "Offline",
Toast.LENGTH_SHORT).show());
            }
        });
        txtFileStatus.setOnClickListener(view -> openFileChooser());
        btnUpload.setOnClickListener(view -> uploadFileToIPFS());
        btnCheckIPFS.setOnClickListener(view -> checkIPFSStatus());
    }

    private boolean isValidIPFSAddress(String address) {
        return address.startsWith("/ip4/") && address.contains("/tcp/");
    }

    private void openFileChooser() {
        Intent intent = new Intent(Intent.ACTION_GET_CONTENT);
        intent.setType("*/*");
        startActivityForResult(intent, PICK_FILE_REQUEST);
    }

    @Override
    protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        if (requestCode == PICK_FILE_REQUEST && resultCode == RESULT_OK && data != null) {
            fileUri = data.getData();
            if (fileUri != null) {
                String fileName = getFileName(fileUri);
                txtFileStatus.setText(fileName != null ? fileName : "Unknown file");
            }
        }
    }

    private void uploadFileToIPFS() {
        if (fileUri == null) {
            Toast.makeText(this, "Select a file first", Toast.LENGTH_SHORT).show();
            return;
        }
        try {
            File file = getFileFromUri(fileUri);
            if (file == null || !file.exists()) {
                Toast.makeText(this, "File not found", Toast.LENGTH_SHORT).show();
                return;
            }
            ipfsManager.uploadFileToIPFS(fileUri, getContentResolver(), new
```

```
IPFSManager.IPFSUploadListener() {
            @Override
            public void onUploadSuccess(String cid) {
                runOnUiThread(() -> {
                    txtFileStatus.setText(cid);
                    copyToClipboard(cid);
                    Toast.makeText(UploadActivity.this, "Uploaded! CID copied",
Toast.LENGTH_SHORT).show();
                });
            }

            @Override
            public void onUploadFailed(String error) {
                runOnUiThread(() -> Toast.makeText(UploadActivity.this, "Upload failed",
Toast.LENGTH_SHORT).show());
            }
        });
    } catch (Exception e) {
        Toast.makeText(this, "Error: " + e.getMessage(), Toast.LENGTH_SHORT).show();
    }
}

private void checkIPFSStatus() {
    if (ipfsManager == null) {
        Toast.makeText(this, "Offline", Toast.LENGTH_SHORT).show();
        return;
    }
    ipfsManager.isIPFSOnline(new IPFSManager.IPFSStatusListener() {
        @Override
        public void onIPFSStatusChecked(boolean isOnline, String message) {
            runOnUiThread(() -> {
                Toast.makeText(UploadActivity.this, isOnline ? "Online" : "Offline",
Toast.LENGTH_SHORT).show();
            });
        }
    });
}

private void copyToClipboard(String text) {
    ClipboardManager clipboard = (ClipboardManager)
getSystemService(Context.CLIPBOARD_SERVICE);
    ClipData clip = ClipData.newPlainText("CID", text);
    clipboard.setPrimaryClip(clip);
}

private String getFileName(Uri uri) {
    String result = null;
    if (uri.getScheme().equals("content")) {
```

```
        try (Cursor cursor = getContentResolver().query(uri, null, null, null, null)) {
            if (cursor != null && cursor.moveToFirst()) {
                int nameIndex = cursor.getColumnIndex(OpenableColumns.DISPLAY_NAME);
                if (nameIndex >= 0) {
                    result = cursor.getString(nameIndex);
                }
            }
        }
        if (result == null) {
            result = uri.getLastPathSegment();
        }
        return result;
    }

    private File getFileFromUri(Uri uri) throws IOException {
        InputStream inputStream = getContentResolver().openInputStream(uri);
        if (inputStream == null) throw new IOException("Cannot open file");
        File tempFile = File.createTempFile("ipfs_upload", null, getCacheDir());
        try (FileOutputStream outputStream = new FileOutputStream(tempFile)) {
            byte[] buffer = new byte[4 * 1024];
            int read;
            while ((read = inputStream.read(buffer)) != -1) {
                outputStream.write(buffer, 0, read);
            }
            outputStream.flush();
        }
        return tempFile;
    }
}
```

activity_upload.xml

```
<?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"
    android:background="@color/dark_black">

    <TextView
        android:id="@+id/txtUploadTitle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="UPLOAD FILE"
        android:textSize="20sp"
```

```
        android:textStyle="bold"
        android:textColor="@color/pure_white"
        android:layout_gravity="center_horizontal"
        android:paddingBottom="30dp"
        android:paddingTop="30dp"/>

    <EditText
        android:id="@+id/txtFileStatus"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="No file selected"
        android:textSize="20sp"
        android:textColor="@color/dark_black"
        android:background="@color/light_grey"
        android:padding="15dp"
        android:focusable="false"
        android:clickable="true"
        android:cursorVisible="false"
        android:maxLines="1"
        android:singleLine="true" />

    <Button
        android:id="@+id/btnUpload"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Upload to IPFS"
        android:backgroundTint="@color/light_grey"
        android:textColor="@color/dark_black"
        android:padding="10dp"
        android:layout_marginTop="20dp" />

    <Button
        android:id="@+id/btnCheckIPFS"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Check IPFS Status"
        android:backgroundTint="@color/light_grey"
        android:textColor="@color/dark_black"
        android:padding="10dp"
        android:layout_marginTop="10dp" />

</LinearLayout>
```

DownloadActivity.java

```
package com.example.zerobyte;

import android.app.AlertDialog;
import android.app.ProgressDialog;
import android.content.ContentResolver;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;

public class DownloadActivity extends AppCompatActivity {
    private static final int PICK_SAVE_LOCATION_REQUEST = 1001;

    private EditText edtFileHash;
    private Button btnDownload, btnCheckStatus;
    private ProgressDialog progressDialog;
    private IPFSManager ipfsManager;
    private Uri saveLocationUri;
    private String currentFileHash;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_download);
        edtFileHash = findViewById(R.id.edtFileHash);
        btnDownload = findViewById(R.id.btnDownloadFile);
        btnCheckStatus = findViewById(R.id.btnCheckIPFSStatus);
        btnDownload.setOnClickListener(v -> handleDownloadClick());
        btnCheckStatus.setOnClickListener(v -> checkIPFSStatus());
        initializeIPFS();
    }

    private void initializeIPFS() {
        String ipfsAddress = getIntent().getStringExtra("IPFS_ADDRESS");
        if (ipfsAddress == null || ipfsAddress.isEmpty()) {
            showToast("Invalid IPFS address");
            finish();
            return;
        }
        ipfsManager = new IPFSManager(this, ipfsAddress, new IPFSManager.IPFSInitListener() {
            @Override
            public void onIPFSInitialized() {
```

```
        }

        @Override
        public void onIPFSInitFailed(String error) {
            showToast("Offline: " + error);
        }
    });
}

private void checkIPFSStatus() {
    if (ipfsManager == null) {
        showToast("Offline: Not initialized");
        return;
    }
    showProgress("Checking connection...");
    ipfsManager.isIPFSOnline(new IPFSManager.IPFSStatusListener() {
        @Override
        public void onIPFSStatusChecked(boolean isOnline, String message) {
            dismissProgress();
            showToast(isOnline ? "Online" : "Offline: " + message);
        }
    });
}

private void handleDownloadClick() {
    String fileHash = edtFileHash.getText().toString().trim();
    if (isValidFileHash(fileHash)) {
        currentFileHash = fileHash;
        requestSaveLocation();
    } else {
        showToast("Invalid IPFS hash (must start with Qm)");
    }
}

private boolean isValidFileHash(String hash) {
    return hash != null && hash.startsWith("Qm") && hash.length() > 10;
}

private void requestSaveLocation() {
    Intent intent = new Intent(Intent.ACTION_CREATE_DOCUMENT);
    intent.addCategory(Intent.CATEGORY_OPENABLE);
    intent.setType("*/*");
    intent.putExtra(Intent.EXTRA_TITLE, "downloaded_file");
    startActivityForResult(intent, PICK_SAVE_LOCATION_REQUEST);
}

@Override
protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
```

```
        super.onActivityResult(requestCode, resultCode, data);
        if (requestCode == PICK_SAVE_LOCATION_REQUEST && resultCode == RESULT_OK && data != null) {
            saveLocationUri = data.getData();
            if (saveLocationUri != null) {
                startVirusScan();
            }
        }
    }

    private void startVirusScan() {
        showProgress("Scanning for viruses...");
        new Thread(() -> {
            VirusTotalAPI.ScanResult result = VirusTotalAPI.scanFile(saveLocationUri, getContentResolver());
            runOnUiThread(() -> {
                dismissProgress();
                handleScanResult(result);
            });
        }).start();
    }

    private void handleScanResult(VirusTotalAPI.ScanResult result) {
        if (result.isError) {
            showErrorDialog("Scan Failed", result.errorMessage, true);
        } else if (result.isInfected()) {
            showVirusWarning(result);
        } else {
            startFileDownload();
        }
    }

    private void showVirusWarning(VirusTotalAPI.ScanResult result) {
        String message = String.format(
                "Security Alert!\n\n" +
                        "• Malicious detections: %d\n" +
                        "• Suspicious detections: %d\n" +
                        "• Clean scans: %d\n\n" +
                        "This file may harm your device. Download anyway?",
                result.maliciousCount,
                result.suspiciousCount,
                result.cleanCount
        );

        new AlertDialog.Builder(this)
                .setTitle("Virus Detected")
                .setMessage(message)
                .setPositiveButton("Download", (d, w) -> startFileDownload())
                .setNegativeButton("Cancel", null)
```

```
            .setIcon(android.R.drawable.ic_dialog_alert)
            .show();
    }

    private void showErrorDialog(String title, String message, boolean allowRetry) {
        AlertDialog.Builder builder = new AlertDialog.Builder(this)
                .setTitle(title)
                .setMessage(message)
                .setIcon(android.R.drawable.ic_dialog_alert);
        if (allowRetry) {
            builder.setPositiveButton("Retry", (d, w) -> startVirusScan());
        }
        builder.setNegativeButton("OK", null).show();
    }

    private void startFileDownload() {
        showProgress("Downloading file...");
        ipfsManager.downloadFileFromIPFS(currentFileHash, saveLocationUri,
                getContentResolver(), new IPFSManager.IPFSDownloadListener() {
                    @Override
                    public void onDownloadSuccess(byte[] fileData) {
                        runOnUiThread(() -> {
                            dismissProgress();
                            showToast("Download completed successfully");
                        });
                    }

                    @Override
                    public void onDownloadFailed(String error) {
                        runOnUiThread(() -> {
                            dismissProgress();
                            showErrorDialog("Download Failed", error, true);
                        });
                    }
                });
    }

    private void showProgress(String message) {
        runOnUiThread(() -> {
            if (progressDialog == null) {
                progressDialog = new ProgressDialog(this);
                progressDialog.setCancelable(false);
            }
            progressDialog.setMessage(message);
            if (!progressDialog.isShowing()) {
                progressDialog.show();
            }
        });
```

```
    }

    private void dismissProgress() {
        runOnUiThread(() -> {
            if (progressDialog != null && progressDialog.isShowing()) {
                progressDialog.dismiss();
            }
        });
    }

    private void showToast(String message) {
        runOnUiThread(() -> Toast.makeText(this, message, Toast.LENGTH_SHORT).show());
    }
}
```

VirusTotalAPI.java

```
package com.example.zerobyte;

import android.content.ContentResolver;
import android.net.Uri;
import android.util.Log;
import org.json.JSONException;
import org.json.JSONObject;
import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.util.Locale;
import java.util.concurrent.TimeUnit;
import okhttp3.MediaType;
import okhttp3.MultipartBody;
import okhttp3.OkHttpClient;
import okhttp3.Request;
import okhttp3.RequestBody;
import okhttp3.Response;

public class VirusTotalAPI {
    private static final String API_KEY =
"c81338cbcf544fbf865db3975269ccfa28cb621e038541f5ea862571254c0e4c"; // Replace with actual key
    private static final String UPLOAD_URL = "https://www.virustotal.com/api/v3/files";
    private static final String REPORT_URL = "https://www.virustotal.com/api/v3/analyses/%s";

    private static final OkHttpClient client = new OkHttpClient.Builder()
            .connectTimeout(3, TimeUnit.SECONDS)
            .readTimeout(3, TimeUnit.SECONDS)
            .writeTimeout(3, TimeUnit.SECONDS)
            .build();
```

```
public static ScanResult scanFile(Uri fileUri, ContentResolver contentResolver) {
    try {
        String analysisId = uploadFile(fileUri, contentResolver);
        if (analysisId == null) {
            return new ScanResult("Upload failed", true);
        }
        String scanReport = pollScanResults(analysisId);
        if (scanReport == null) {
            return new ScanResult("Scan timeout", true);
        }
        return analyzeScanReport(scanReport);
    } catch (Exception e) {
        Log.e("VirusTotal", "Scan error", e);
        return new ScanResult("Scan failed: " + e.getMessage(), true);
    }
}

private static String uploadFile(Uri fileUri, ContentResolver contentResolver)
        throws IOException, JSONException {
    try (InputStream inputStream = contentResolver.openInputStream(fileUri)) {
        if (inputStream == null) {
            throw new IOException("Cannot open file stream");
        }
        ByteArrayOutputStream buffer = new ByteArrayOutputStream();
        byte[] data = new byte[16384];
        int bytesRead;
        while ((bytesRead = inputStream.read(data)) != -1) {
            buffer.write(data, 0, bytesRead);
        }
        RequestBody body = new MultipartBody.Builder()
                .setType(MultipartBody.FORM)
                .addFormDataPart("file", "file",
                        RequestBody.create(buffer.toByteArray(), MediaType.get("application/octet-stream")))
                .build();
        Request request = new Request.Builder()
                .url(UPLOAD_URL)
                .post(body)
                .addHeader("x-apikey", API_KEY)
                .build();
        try (Response response = client.newCall(request).execute()) {
            if (!response.isSuccessful()) {
                throw new IOException("Upload failed: " + response.code());
            }
            String json = response.body().string();
            return new JSONObject(json).getJSONObject("data").getString("id");
        }
    }
}
```

```
private static String pollScanResults(String analysisId) {
    int maxAttempts = 7;
    int initialDelayMs = 4000;
    for (int attempt = 0; attempt < maxAttempts; attempt++) {
        try {
            Thread.sleep(initialDelayMs * (int)Math.pow(2, attempt));
            String report = getScanReport(analysisId);
            JSONObject json = new JSONObject(report);
            String status = json.getJSONObject("data")
                    .getJSONObject("attributes")
                    .getString("status");
            if ("completed".equals(status)) {
                return report;
            }
        } catch (Exception e) {
            Log.w("VirusTotal", "Polling attempt " + (attempt + 1) + " failed", e);
        }
    }
    return null;
}

private static String getScanReport(String analysisId) throws IOException {
    Request request = new Request.Builder()
            .url(String.format(Locale.US, REPORT_URL, analysisId))
            .get()
            .addHeader("x-apikey", API_KEY)
            .build();
    try (Response response = client.newCall(request).execute()) {
        if (!response.isSuccessful()) {
            throw new IOException("API error: " + response.code());
        }
        return response.body().string();
    } catch (Exception e) {
        throw new IOException("Network error: " + e.getMessage());
    }
}

private static ScanResult analyzeScanReport(String scanReport) throws JSONException {
    JSONObject report = new JSONObject(scanReport);
    JSONObject stats = report.getJSONObject("data")
            .getJSONObject("attributes")
            .getJSONObject("stats");
    return new ScanResult(
            stats.getInt("malicious"),
            stats.getInt("suspicious"),
            stats.getInt("undetected")
    );
```

```
    }

    public static class ScanResult {
        public final int maliciousCount;
        public final int suspiciousCount;
        public final int cleanCount;
        public final boolean isError;
        public final String errorMessage;
        public ScanResult(int malicious, int suspicious, int clean) {
            this.maliciousCount = malicious;
            this.suspiciousCount = suspicious;
            this.cleanCount = clean;
            this.isError = false;
            this.errorMessage = null;
        }
        public ScanResult(String error, boolean isError) {
            this.maliciousCount = 0;
            this.suspiciousCount = 0;
            this.cleanCount = 0;
            this.isError = isError;
            this.errorMessage = error;
        }
        public boolean isInfected() {
            return maliciousCount > 0 || suspiciousCount > 0;
        }
    }
}
```

activity_download.xml

```
<?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"
    android:background="@color/dark_black">

    <TextView
        android:id="@+id/txtDownloadTitle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="DOWNLOAD FILE"
        android:textSize="20sp"
        android:textStyle="bold"
        android:textColor="@color/pure_white"
        android:layout_gravity="center_horizontal"
```

```
        android:paddingBottom="30dp"
        android:paddingTop="30dp"/>

    <EditText
        android:id="@+id/edtFileHash"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Enter File Hash"
        android:textSize="20sp"
        android:textColor="@color/dark_black"
        android:background="@color/light_grey"
        android:padding="15dp"
        android:inputType="text"
        android:layout_marginBottom="20dp"/>

    <Button
        android:id="@+id/btnDownloadFile"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Download to Device"
        android:backgroundTint="@color/light_grey"
        android:textColor="@color/dark_black"
        android:padding="10dp"
        android:layout_marginTop="20dp" />

    <Button
        android:id="@+id/btnCheckIPFSStatus"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Check IPFS Status"
        android:layout_marginBottom="12dp"
        android:backgroundTint="@color/light_grey"
        android:textColor="@color/dark_black"/>

    <TextView
        android:id="@+id/txtDownloadStatus"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Status: "
        android:textSize="20sp"
        android:paddingTop="20sp"
        android:textColor="@color/pure_white"/>

</LinearLayout>
```

colors.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources xmlns:tools="http://schemas.android.com/tools">
    <string name="app_name">zerobyte</string>
    <string name="default_web_client_id">800887044729-
488dbo9ejmiu1j200t6g6mejojduumu6.apps.googleusercontent.com</string>
    <string name="sign_in_with_google">Sign in with Google</string>
    <string name="sign_out">Sign Out</string>
    <string name="status_signed_in">Logged in as:</string>
    <string name="status_signed_out">Not signed in</string>
</resources>
```

strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources xmlns:tools="http://schemas.android.com/tools">
    <string name="app_name">zerobyte</string>
    <string name="default_web_client_id">800887044729-
488dbo9ejmiu1j200t6g6mejojduumu6.apps.googleusercontent.com</string>
    <string name="sign_in_with_google">Sign in with Google</string>
    <string name="sign_out">Sign Out</string>
    <string name="status_signed_in">Logged in as:</string>
    <string name="status_signed_out">Not signed in</string>
</resources>
```

themes.xml

```
<resources xmlns:tools="http://schemas.android.com/tools">
    <style name="Base.Theme.Zerobyte" parent="Theme.Material3.DayNight.NoActionBar">
    </style>
    <style name="Theme.Zerobyte" parent="Base.Theme.Zerobyte" />
</resources>
```

network_security_config.xml

```
<?xml version="1.0" encoding="utf-8"?>
<network-security-config>
    <domain-config cleartextTrafficPermitted="true">
        <domain includeSubdomains="true">192.168.1.104</domain>
        <domain includeSubdomains="true">192.168.1.106</domain>
        <domain includeSubdomains="true">10.0.0.2</domain>
        <domain includeSubdomains="true">localhost</domain>
        <domain includeSubdomains="true">10.0.2.2</domain>
        <domain includeSubdomains="true">172.20.10.2</domain>
    </domain-config>
</network-security-config>
```

values.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="default_web_client_id" translatable="false">800887044729-
guqeqf1e17f0oe7e7ps9mo9p0kh39oga.apps.googleusercontent.com</string>
    <string name="gcm_defaultSenderId" translatable="false">800887044729</string>
    <string name="google_api_key"
translatable="false">AIzaSyA42qsVKm7B20D9Z5aZuXDbVWAkXRWSxCw</string>
    <string name="google_app_id"
translatable="false">1:800887044729:android:a10d47e738fcdc646332a8</string>
    <string name="google_crash_reporting_api_key"
translatable="false">AIzaSyA42qsVKm7B20D9Z5aZuXDbVWAkXRWSxCw</string>
    <string name="google_storage_bucket" translatable="false">zerobyte-82ad4.firebasestorage.app</string>
    <string name="project_id" translatable="false">zerobyte-82ad4</string>
</resources>
```

build.gradle.kts (:app)

```
plugins {
    id("com.android.application")
    id("com.google.gms.google-services")
}

android {
    namespace = "com.example.zerobyte"
    compileSdk = 35
    defaultConfig {
        applicationId = "com.example.zerobyte"
        minSdk = 26
        targetSdk = 35
        versionCode = 1
        versionName = "1.0"
        testInstrumentationRunner = "androidx.test.runner.AndroidJUnitRunner"
    }
    buildTypes {
        release {
            isMinifyEnabled = false
            proguardFiles(
                getDefaultProguardFile("proguard-android-optimize.txt"),
                "proguard-rules.pro"
            )
        }
    }
    compileOptions {
        sourceCompatibility = JavaVersion.VERSION_11
        targetCompatibility = JavaVersion.VERSION_11
    }
    buildFeatures {
        viewBinding = true
    }
```

```
}

dependencies {
    implementation(libs.appcompat)
    implementation(libs.material.v190)
    implementation(libs.constraintlayout.v214)
    implementation(libs.navigation.fragment.ktx)
    implementation(libs.navigation.ui.ktx)
    implementation(libs.core.ktx)
    implementation(platform("com.google.firebase:firebase-bom:33.10.0"))
    implementation ("com.google.firebase:firebase-auth:22.1.1")
    implementation("com.google.firebase:firebase-analytics")
    implementation(libs.play.services.auth)
    implementation("com.github.ipfs:java-ipfs-http-client:v1.4.4")
    implementation(libs.okhttp)
    implementation(libs.gson)
    testImplementation(libs.junit)
    androidTestImplementation(libs.junit.v115)
    androidTestImplementation(libs.espresso.core.v351)
    implementation ("com.github.ipfs:java-ipfs-http-client:v1.3.3")
    implementation ("com.squareup.retrofit2:retrofit:2.9.0")
    implementation ("com.squareup.retrofit2:converter-gson:2.9.0")
}
```

settings.gradle.kts (zerobyte)

```
pluginManagement {
    repositories {
        google {
            content {
                includeGroupByRegex("com\\.android.*")
                includeGroupByRegex("com\\.google.*")
                includeGroupByRegex("androidx.*")
            }
        }
        mavenCentral()
        gradlePluginPortal()
        maven { url = uri("https://jitpack.io") }
    }
}

dependencyResolutionManagement {
    repositoriesMode.set(RepositoriesMode.PREFER_PROJECT)
    repositories {
        google()
        mavenCentral()
        maven { url = uri("https://jitpack.io") }
    }
}

rootProject.name = "zerobyte"
include(":app")
```

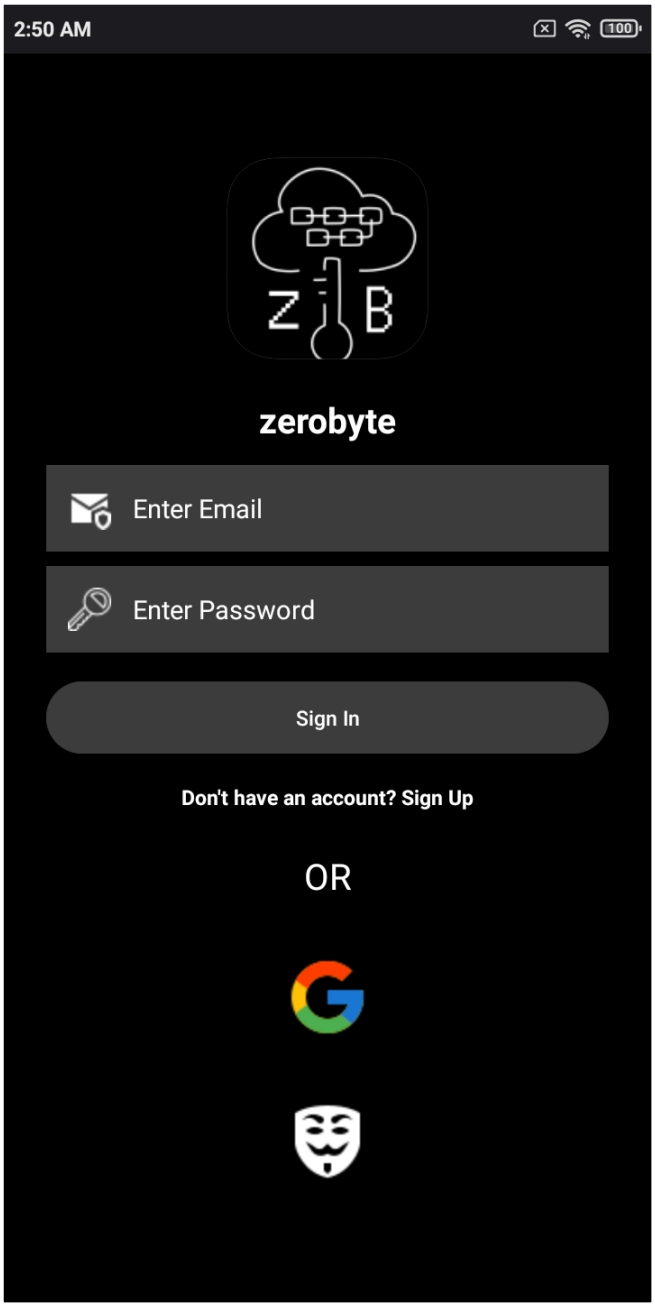
google-service.json


```
{
  "project_info": {
    "project_number": "800887044729",
    "project_id": "zerobyte-82ad4",
    "storage_bucket": "zerobyte-82ad4.firebasestorage.app"
  },
  "client": [
    {
      "client_info": {
        "mobilesdk_app_id": "1:800887044729:android:a10d47e738fcdc646332a8",
        "android_client_info": {
          "package_name": "com.example.zerobyte"
        }
      },
      "oauth_client": [
        {
          "client_id": "800887044729-488dbo9ejmiu1j200t6g6mejojduumu6.apps.googleusercontent.com",
          "client_type": 1,
          "android_info": {
            "package_name": "com.example.zerobyte",
            "certificate_hash": "23c31432fe6c5ad01b7818312f36f0c4b655d73b"
          }
        },
        {
          "client_id": "800887044729-guqeqf1e17f0oe7e7ps9mo9p0kh39oga.apps.googleusercontent.com",
          "client_type": 3
        }
      ],
      "api_key": [
        {
          "current_key": "AIzaSyA42qsVKm7B20D9Z5aZuXDbVWAkXRWSxCw"
        }
      ],
      "services": {
        "appinvite_service": {
          "other_platform_oauth_client": [
            {
              "client_id": "800887044729-guqeqf1e17f0oe7e7ps9mo9p0kh39oga.apps.googleusercontent.com",
              "client_type": 3
            }
          ]
        }
      }
    }
  ],
  "configuration_version": "1" }
```

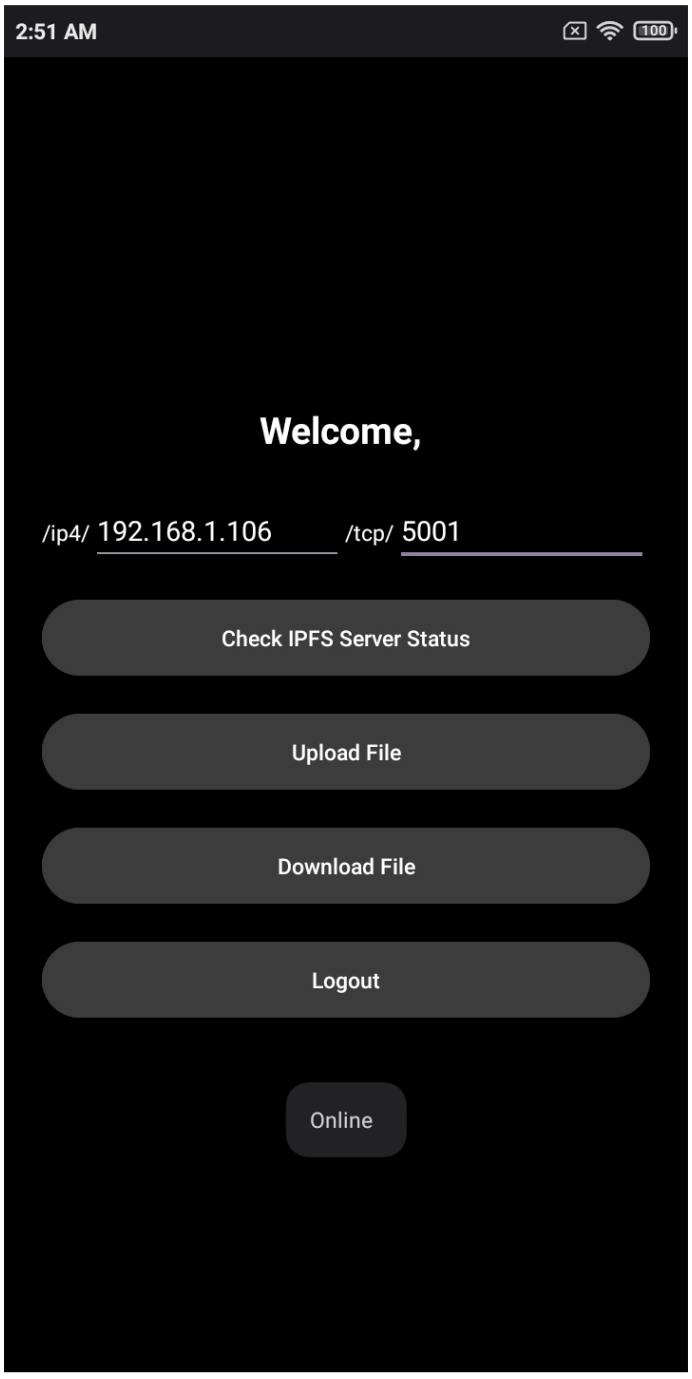
OUTPUT

Splash Screen

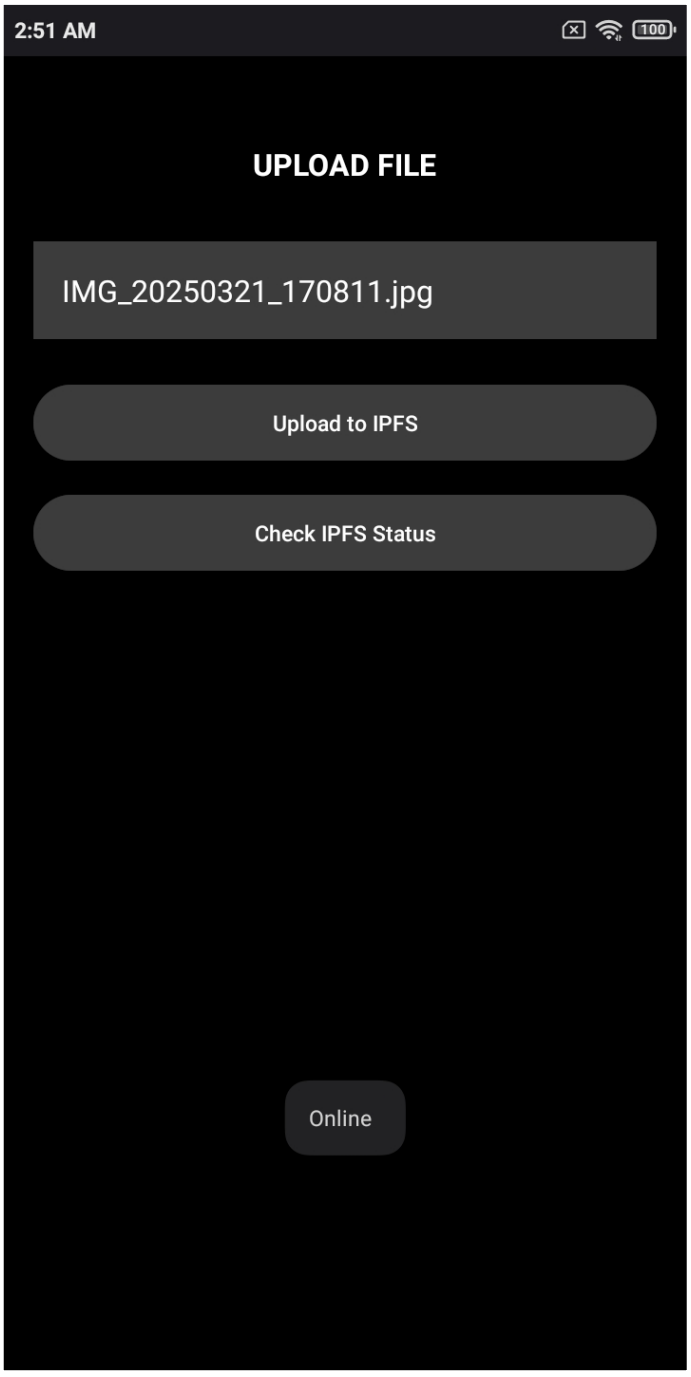
Login Screen



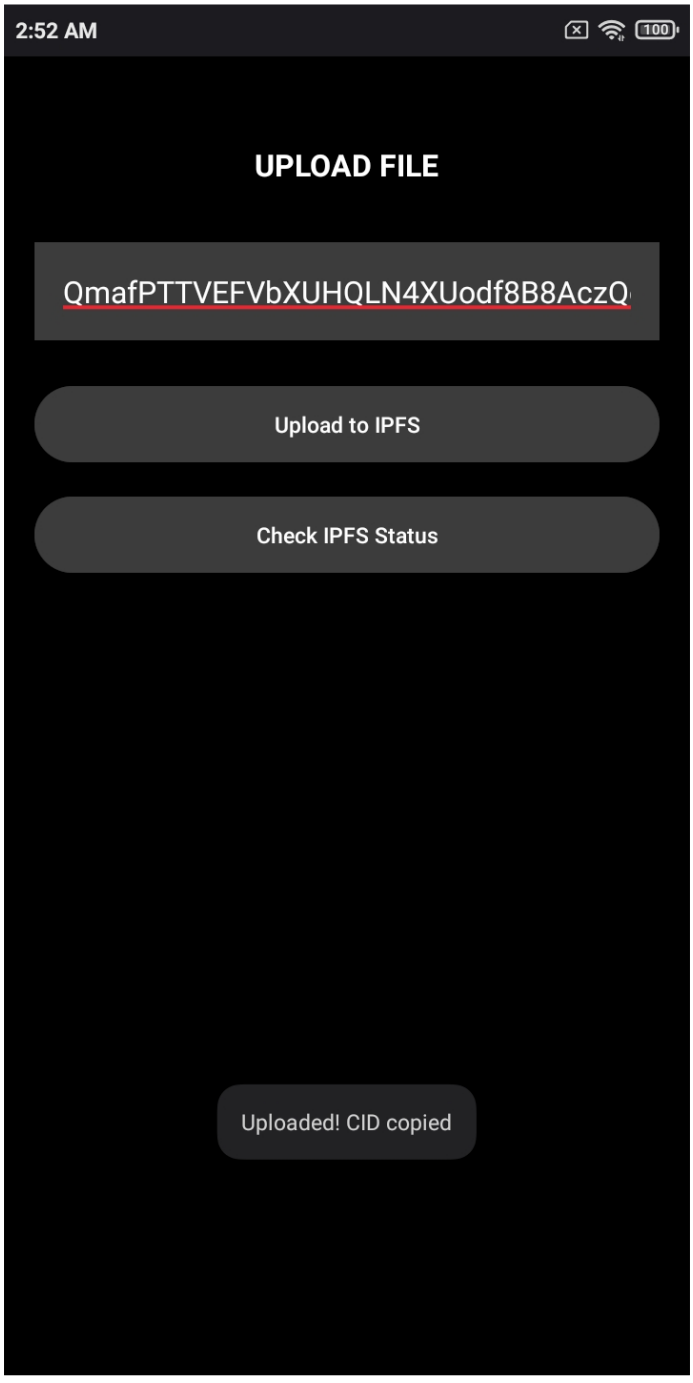
Welcome Screen



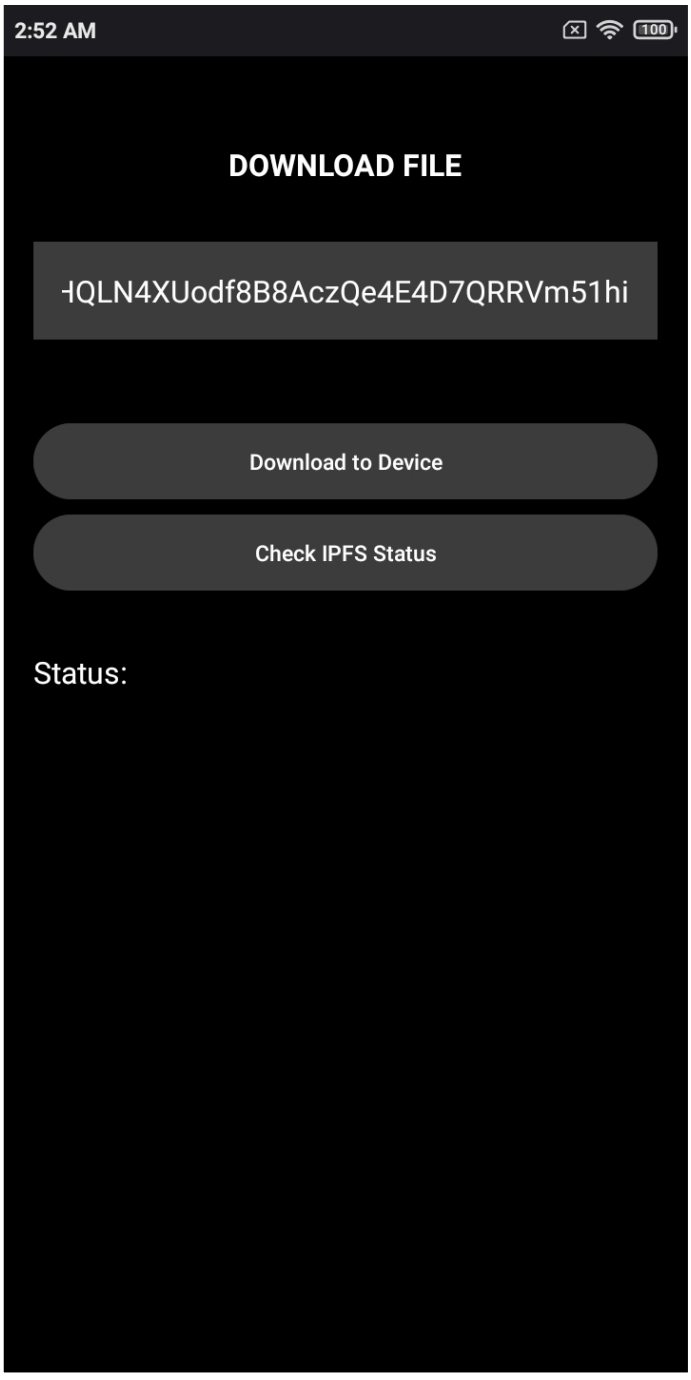
Upload Activity



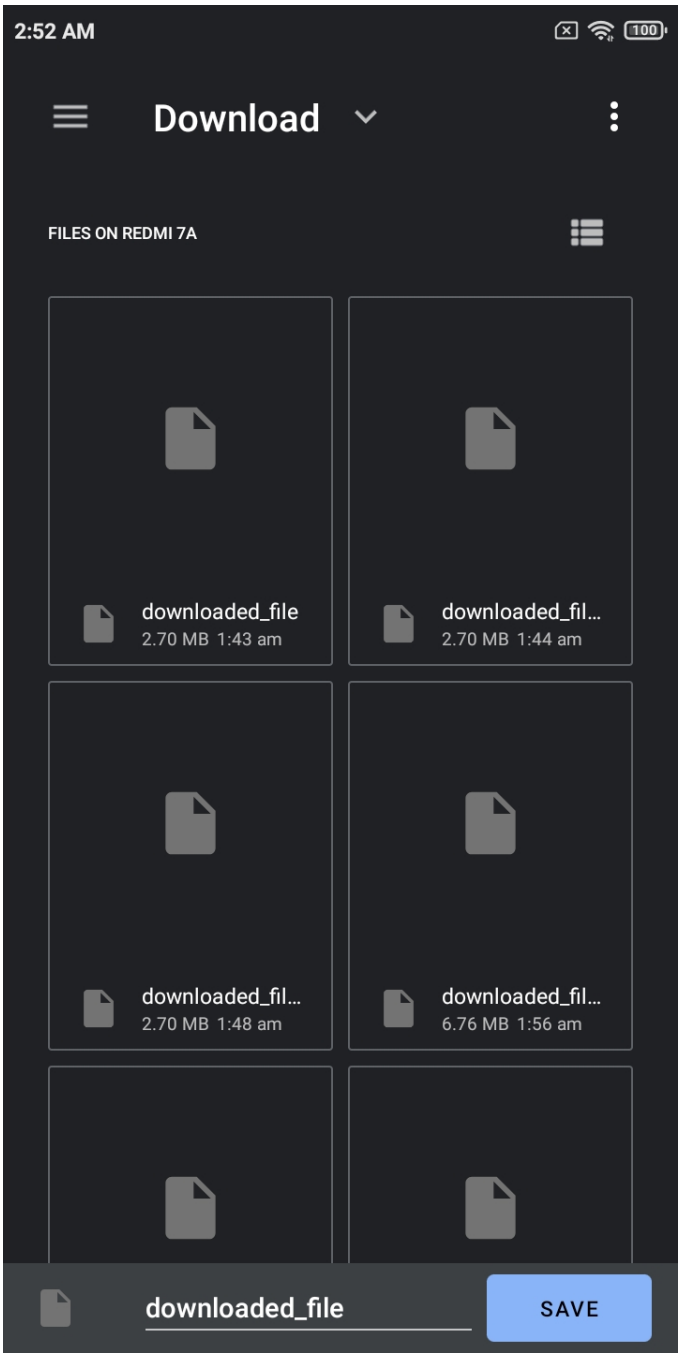
Upload Complete



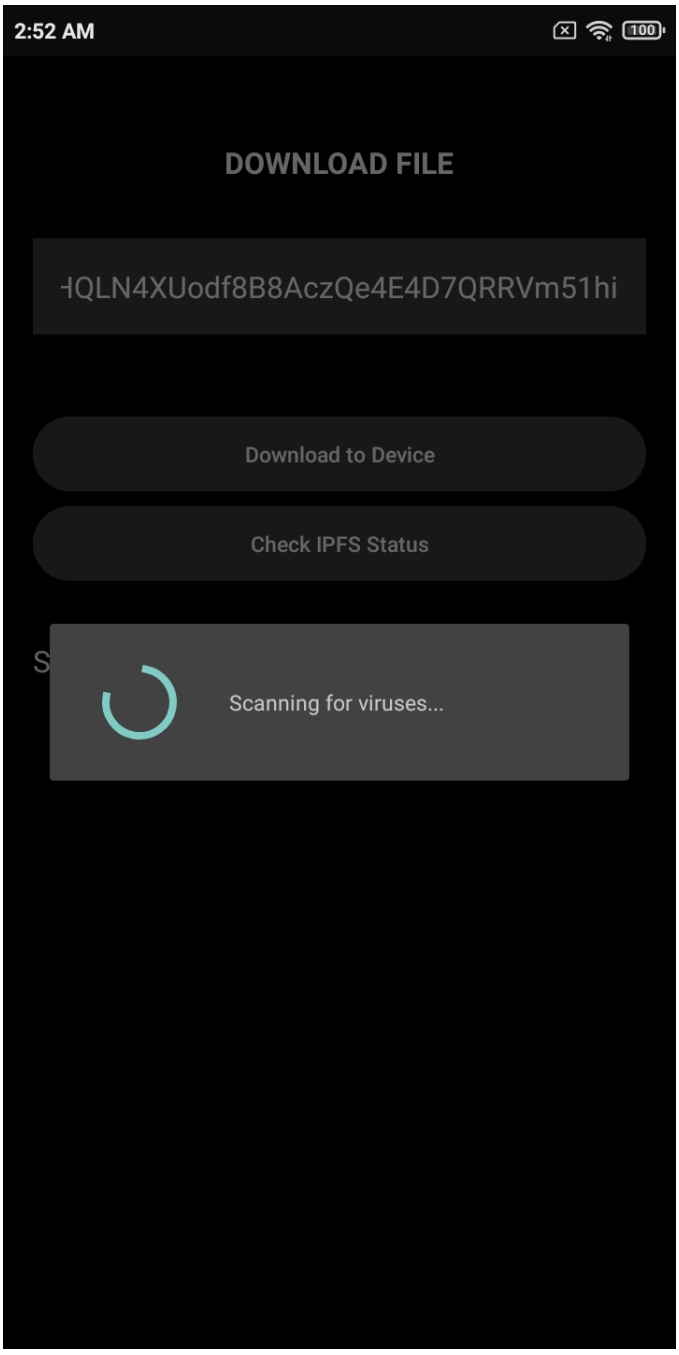
Download Activity



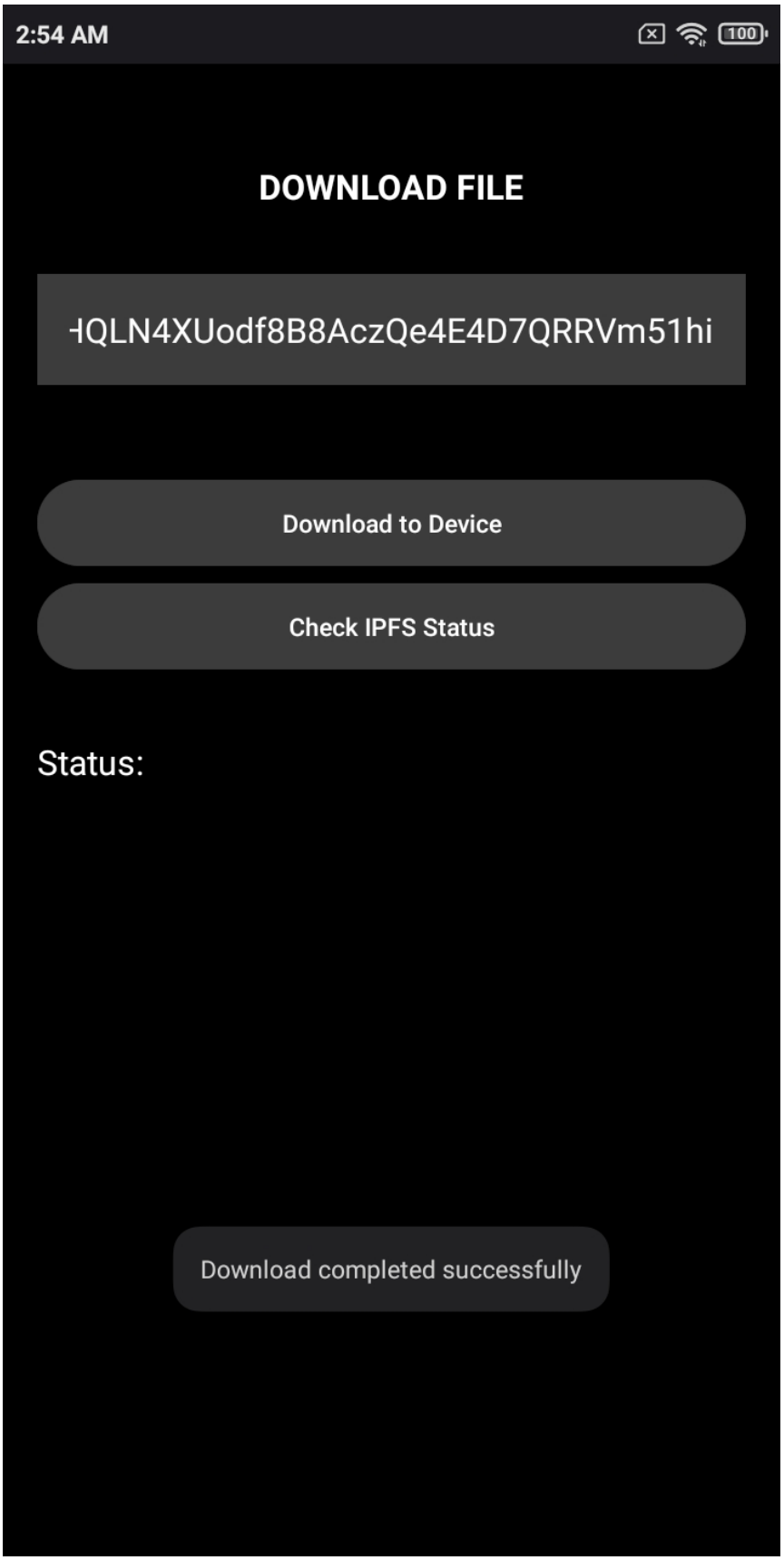
Select Download Location



Virus Total Scanning



Download Complete



REFERENCES

• **Android Studio**
o Android Studio is Google's official integrated development environment (IDE) for building Android apps, offering tools for coding, debugging, and testing.
o Available at: https://developer.android.com/studio

• **DeepSeek**
o DeepSeek is an advanced AI platform offering models for text generation, coding assistance, and research-oriented solutions.
o Available at: https://www.deepseek.com/

• **ChatGPT**
o ChatGPT is an AI language model that assists with generating text, answering questions, and solving problems across various topics.
o Available at: https://chat.openai.com/

• **Plant Text UML**
o PlantText is a web-based tool for quickly creating UML diagrams using simple text syntax, streamlining software design workflows.
o Available at: https://www.planttext.com/

• **Figma**
o Figma is a cloud-based design tool for creating and collaborating on user interface (UI) and user experience (UX) designs in real time.
o Available at: https://www.figma.com/

• **IPFS Documentation**
o The Interplanetary File System (IPFS) is a decentralized protocol for storing and sharing hypermedia in a distributed network.
o Available at: https://docs.ipfs.tech/

• **Web3**
o Web3 refers to the ecosystem of decentralized technologies, including blockchain and smart contracts, enabling user-owned internet applications.
o Available at: https://web3js.readthedocs.io/