# Abstract

**An effective coin tracking system is designed and implemented to track the movement of any digital currency equipped at any time.**

**The proposed system has made good use of a common technology that combines the Smartphone app with Internet access**

***Chapter 1-***

*Introduction*

*Background*

*Business model*

*System vision document \_*

*Devolpment\_Enviorment\_*

*project management****-***

***Chapter 2***

*Gathering Requirements*

*System Problem Statement*

*Functional & non functional requirements*

*System Model*

*Use Case Model*

*Domain Class Model*

*Use Case Description Scenarios*

*Activity & System Sequence Models*

*System Activity Diagrams*

*System Sequence Diagrams*

***Chapter 3***

*Introduction*

*Architecture and Deployment environment Design*

*Software Architecture Design*

*Sequence Models*

*Design Class Models*

*Design the Database*

***Chapter 4***

*IMPLEMENTATION*

*Building code* ***6***

***Chapter 5***

*Unit Testing*

*Set cases test*

*Integration Testing*

*System Testing*

***List of Tables:***

*Table1:Stakeholder Analysis*

*Table 2:Risk Analysis*

*Table3:Use Case Discriptions*

*Table4:Testing*

***List of Figers:***

*Figer1:Context level digrams*

*Fifer2:first level(DFD)*

*Figer3:*

*Figer4:Gant Chart*

*Figer5:Use Case Digram*

*Figer6:Login*

*Giger7:Signup*

*Figer8:Sequence Digram*

*Figer9:Network environment*

*Figer10:Software architecture Design*

*Figer11:System and User interface*

*Figer12:System Security*

*Figer13:Testing*

***Chapter one***

***introduction***

* 1. **Introduction**

**Our application works to provide all digital currencies of different types and available in the markets and to increase trading opportunities, so we have provided this application that helps to enter this world and keep up with it and know prices at any time and in the easiest and fastest ways through buying, selling and trading ,** **as the application provides an electronic wallet that depends There is a pre-existing balance in it through which the owner can use this balance electronically, and the owner can put any financial balance in it for later use.**

**So we would like to introduce you:**

**coins verse App**

**A way for a new life to enter the virtual world of digital currencies, see their types, prices and names, track the change in prices, and enter this world and keep up with it**

## 1.2Background

**Our goal in this project is to help people know the prices of digital currencies and facilitate the process of trading in easy, fast, efficient and safe ways, as well as attract people to enter the world of trading. The application also contains an electronic wallet that helps in the ability to pay easily across devices and its spread, in addition to a competitive advantage in this field. The application was developed using the “Flutter” programming language.**

**• Advantages**

**1. Usability of design and user interface.**

**2. Facilitate the process of knowing currency rates faster, easier and in a more flexible way.**

**• Disadvantages**

**1. With a large number of users, the application may become slower.**

**2. If the customer does not have an Internet connection, he may not be able to use the application**

**1.3 Business Model**

**. Business needs**

**The application provides:**

**1) Knowing currency rates and the value of change in them.**

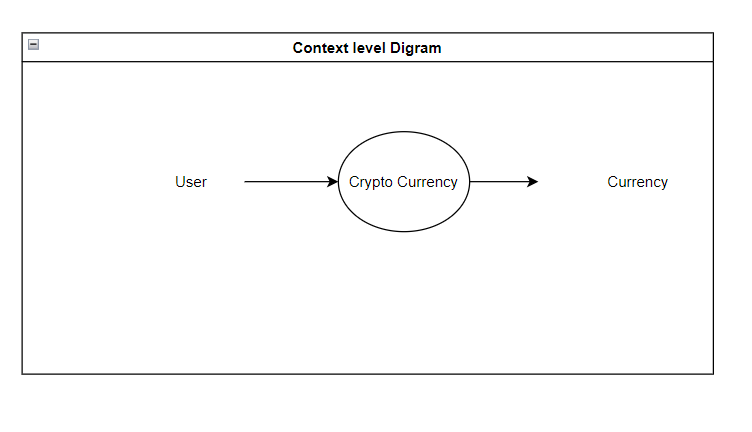
**2) The ability of people to trade quickly and safely.**

**3) Greater ease in buying, selling and trading.**

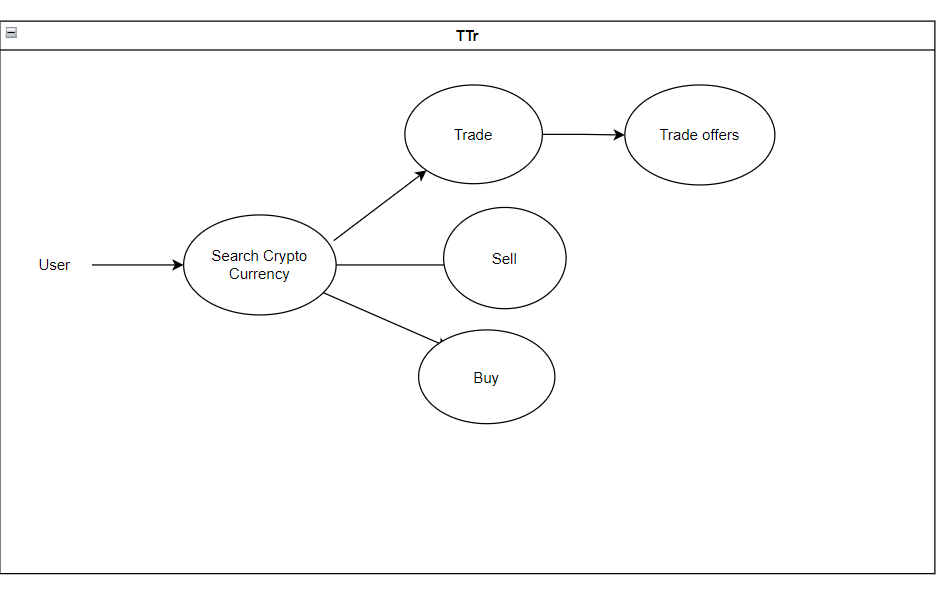
### . Business Environment

**This environment includes all the information of currency types and rates all in one application.**

#### Software Context Diagram

****

*Figer1:Context level Digram*

****

*Figer2:First level(DFD)*

**. System Scope**

**Coins Verse app will solve the problem of wallet file loss and payment information spoofing**

### Stakeholder Analysis

|  |  |
| --- | --- |
| **Stakeholders** | **Responsibilities** |
|  | **Login/logout** |
|  | **Register** |
| **User** | **Buy/Sell** |
|  | **Trade/Trade offers** |
|  | **Wallet** |

*Table 1: stakeholder analysis*

#### 1.4 System Vision Document

**•** Goals

**Building an application for currencies to present the types and prices of currencies in the rise and fall that occurred and will occur in the future.**

**•** Benefits

**1. Entering the world of trading and keeping up with it**

**2. Knowing currency rates accurately and quickly**

**3. Collect money through safe methods**

**•** Capabilities

**1. Easy access to the application**

**2. Easy access to currency type**

**3. Ease of searching for currencies and their prices**

**4. Easy to track currency movement**

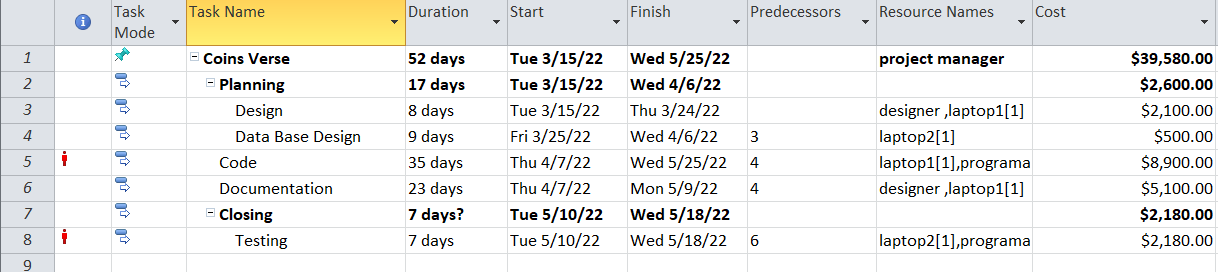
**5. ease trade and trade offers**

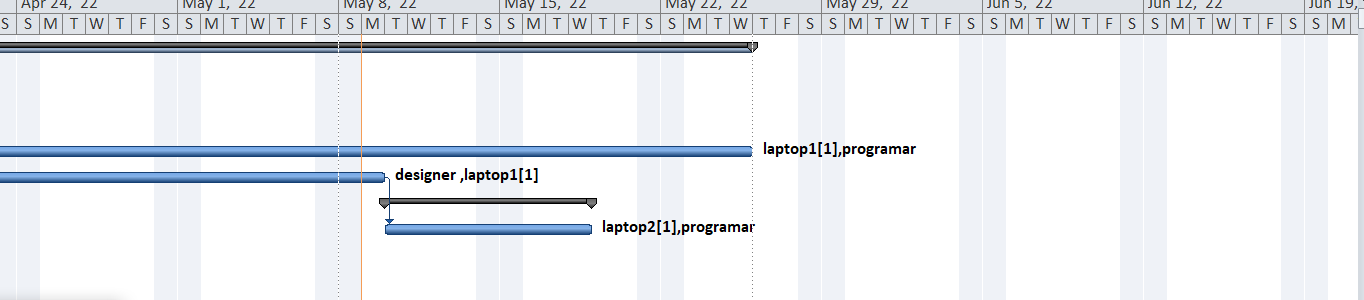
**6.** **Possibility to pay by visa**

**1.6 Project management**

**Regardless of the size of the project, there are specific basic elements that a project manager must follow that fails that will frustrate his project with many problems and issues. Whether the intent of the project is to manage change, improve services, or implement systems, a successful project must include such basic considerations as strategic planning. Planning defines and understands the needs of the project and what it wants to achieve. Through this process, project managers are able to set specific, measurable and achievable goals**

#### Project Iteration and Schedule

****

****

*Figer4:Gant Chart*

#### Project Financial Feasibility and Risk Analysis

**Economic Feasibility**

The project is economically feasible as the only cost involved is having a smartphone with the minimum requirements mentioned earlier. For the users to access the application, the only cost involved will be in getting access to the Internet.

**Risk Analysis**

|  |  |  |  |
| --- | --- | --- | --- |
| **Opportunities** | **Strengths** | **weaknesses** | **Threats** |
| Make your trading experience smoother and easier. | 1. Faster access to currency types and rates  2. Fast tracking of price changes | 1.The customer is unable to try and test the product.  2.Poor ability to deceive or cheat | like any other technological instrument can face for difficulties using especially the one with little to none tech skills. |

*Table2:Risk Analysis*

#### 1.5 Development Environment

.The application was developed using. Firstly, the Flutter programing language because it was the most feasible language to build our application as it will ensure a faster code writing time and it provides a variety of documents and resources. Secondly, FireBase to build our database because it produces a reliable and extensive database as well as fast & safe hosting.

#### Development Tools

* **The only technical aspects needed are mentioned below:**
* **Operating Environment**

1. **Android studio (Flutter) .**
2. **FireBase .**

* **Data base**

**1. Software name: firebase**

**2. Purpose**

**provided in software and system development, needs to improve the profitability of a software intensive product or service by allowing more users in without witnessing any latency, and improve project productivity.**

**System Development Process**

**Information Gathering**

In this important step, the problems, objectives, and resources needed are outlined. Participating stakeholders come together and engage in brainstorming. If the software is not brand new there will be less information and data gathering and more focus on improvements.

**Analysis**

At this point, end-user requirements have been clearly formulated. Information on competitive products is also collected.

**Design**

In this phase, the system design is developed. The overall software structure, layout, and business rules are designed. The overall software development model is created. Then the output of this phase must classification the new system as a collection of models.

**Implementation**

This is the actual construction phase of the system. The logical part of the system is formulated and the building of any hardware is accomplished. The programming language is already decided and the codes are written.

**System Testing**

The different workable parts of the system are brought together, making a whole integrated system. Various inputs are collected, analysed, and fed into the system. The main aim of the testing phase is to ensure that the system Goals are met and overall customers satisfaction is achieved.

***Chapter two***

***requirements***

**Requirements analysis is critical to the success of a systems or software project, The Requirements should be documented, actionable, measurable, testable, traceable, related to identified business needs or opportunities, and defined to a level of detail sufficient for system design.**

## 2.1 Gathering Requirements

The techniques we have used to gather requirements are Brainstorming, user observations, Prototyping.

## 2.2 System Problem Statement

## The real goal of this project is to popularize the idea of digital currencies in Jordan and increase the chances of using them and trading opportunities in them, where the customer can create his own account and log in to it at any time to know everything new on the site. Concern about your information, the site monitors the change in prices periodically

## 2.3 Functional, Non-Functional Requirments

**Functional Requirments**

**Client Requirments**

The Customer should be able to sign in for the system by entering (username, password).

The Customer must be able to purchase coins

The customer must be able to give an opinion.

The customer must be able to deal with the electronic wallet.

**Non-Functional Requirments**

**Non- functional requirements are conditions and constraints to which the program must conform …**

**Execution qualities**

**Usability**

The system shall be easy to use.

The system shall provide easy navigation for the user.

No technical skills shall be required to use the system.

**Security**

The Email and password shall be required to the active screen, only authorized people can access to their accounts.

**Availability**

System is required to have 95% available.

**Responsiveness**

For every invalid input from the user, the system shall display a meaningful error message explaining the input format expected.

**Efficiency**

The system should respond in 2 seconds or less.

**Evolution qualities**

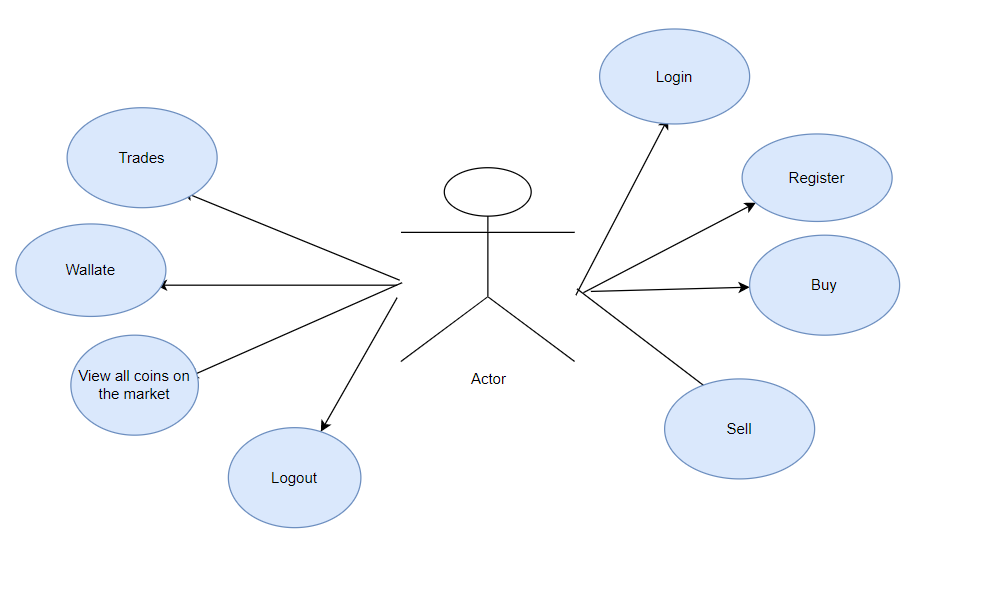
**Maintainability**

The system components shall be able to change without breaking the whole system**.**

## System model

### 2.4.1Use Case Model

### 



Figer5:Use case Digrams

### 2.4.2 Domain Class Model

Domain class model is the first model created during analysis because it is easier and convenient to define static entities which are independent of the application and are more stable in the progression of the software development. It is impossible to construct the entire software at once and develop every aspect uniformly. The first constructed model always has flaws and it is refined with several iterations. We will discuss each step of constructing a domain class model in detail. In the design of a system, a number of classes are identified and grouped together in a class diagram which helps to determine the static relations between those objects. With detailed modeling, the classes of the conceptual design are often split into a number of subclasses. In order to further describe the behavior of systems, these class diagrams can be complemented by a state diagram or UML state machine. Also instead of class diagrams object role modeling can be used if you just want to model the classes and their relationships.

### 2.4.3 Use Cases Descriptions "Scenarios"

A use case scenario is a description that illustrates, step by step, how a user is intending to use a system, essentially capturing the system behavior from the user’s point of view. A use case scenario can include stories, examples, and drawings. Use cases are extremely useful for describing the problem domain in unambiguous terms and communicating with the potential users of a system.

**Use Cases Descriptions**

|  |  |
| --- | --- |
| **Use case** | **Description** |
| **Register** | **To enter the application** |
| **Login** | **to access the application's services** |
| **Wallet** | **Can add a credit Card** |
| **Buy** | **He can buy if he has money** |
| **View all coins in the market** | **View all types and rates of currencies** |
| **Trade** | **Submit a trade offers or approve the offer** |
| **Sell** |  |
| **Logout** | **Log out the application** |

**Table3: Use Case Discriptions**

### 2.4.4 Activity & System Sequence Models

1. **System Activity Model**

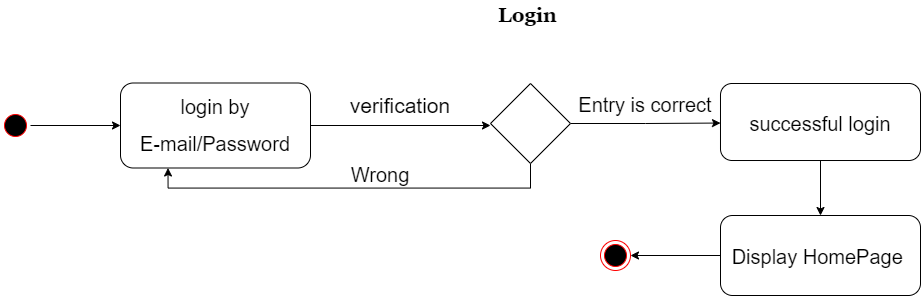
Is used to display the sequence of activities and show the workflow from a start point to the finish point detailing the many decision paths that exist in the progression of events contained in the activity.

1. **System Sequence Model**

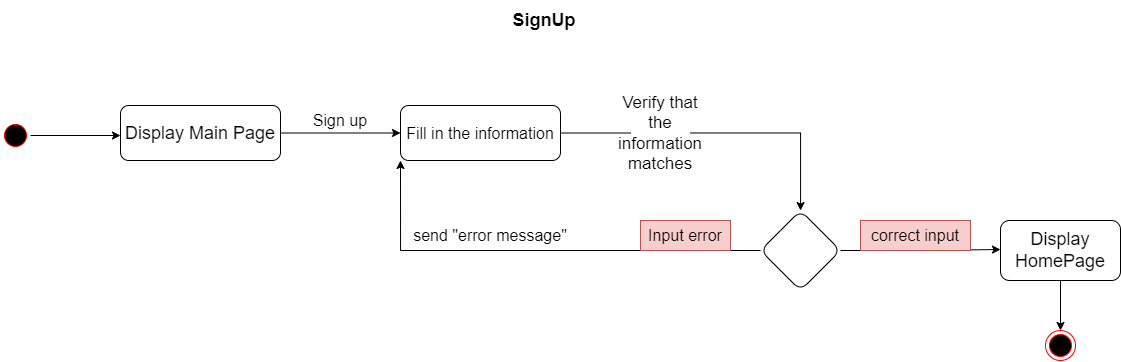
A sequence model elaborates a use case and describes the interaction between the objects in a sequence over a period of time. The sequence model conceptualizes the interaction between the objects by displaying the exchange of messages between them over time.

**2.4.5 System activity diagram**

Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system.

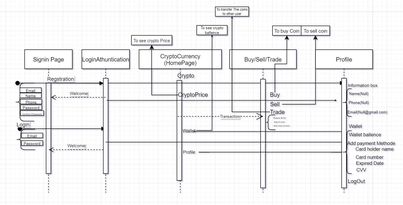


*Figer6:login*



Figer7:Signup

### 2.4.6 System Sequence Diagram



*Figer8:Sequence Digram*

***Chapter three***

***Design***

**Introduction**

In previous chapter we have discussed all part of requirements such as functional or non-functional requirement defined how gathering data, now we know what exactly the problem.

In this chapter we will give a description of solution with in a set a diagram like class sequence and active diagram and other techniques to show how the design will be

**3..1 Architecture and Deployment Environment Design**

Before you design your system, you need to understand the physical and logical aspects of your current environment. From a physical standpoint, your design depends on the type and integrity of your network infrastructure

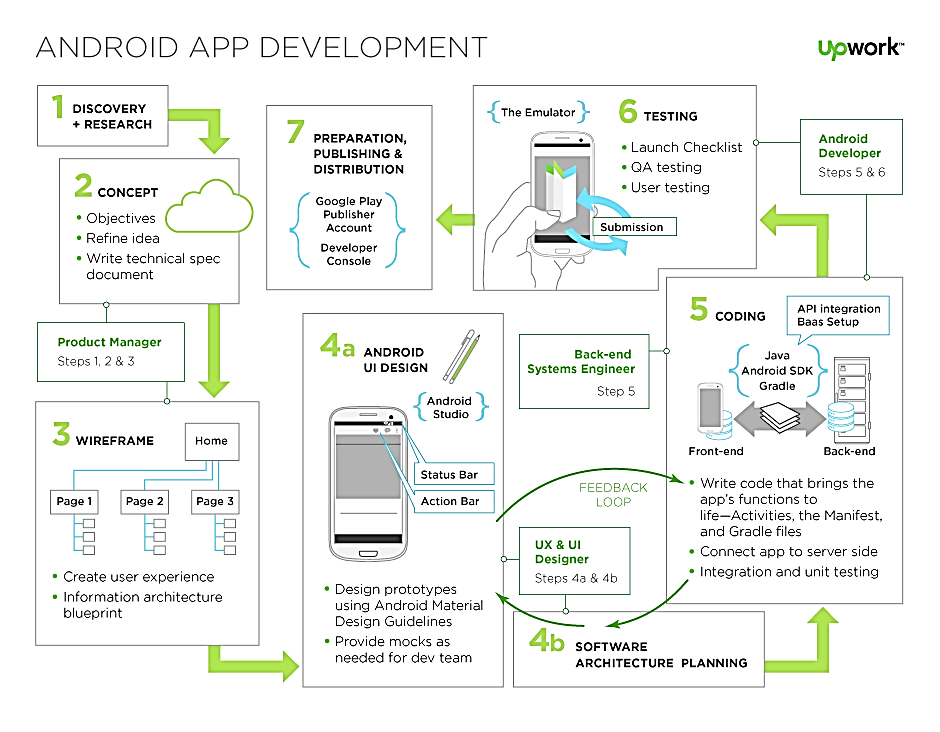
Architecture and Deployment Environment Design

## 

*Figer9: Network environment*

## 3.2 Software Architecture Design

The software architecture of a program or computing system is a depiction of the system that aids in the understanding of how the system will behave. Software architecture serves as the blueprint for both the system and the project developing it, defining the work assignments that must be carried out by design and implementation teams, the architecture is the primary carrier of system qualities such as performance, modifiability, and security, none of which can be achieved without a unifying architectural vision. Architecture is an artifact for early analysis to make sure that a design approach will yield an acceptable system



*Fige10: Software Architecture Design*

## 3.3 Sequence Models

## A sequence model elaborates a use case and describes the interaction between the objects in a sequence over a period of time. The sequence model conceptualizes the interaction between the objects by displaying the exchange of messages between them over time.

**3.4 Design Class Model**

**https://coinmarketcap.com/api/**

## 3.5 Design the database

Firebase is a mobile and web application development platform developed by Firebase, Inc. in 2011, then acquired by Google in 2014. As of October 2018, the Firebase platform has 18 products which are used by 1.5 million apps.

Firebase evolved from Evolve, a prior startup founded by James Tamplin and Andrew Lee in 2011. Evolve provided developers an API that enables the integration of online chat functionality into their websites. After releasing the chat service, Tamplin and Lee found that it was being used to pass application data that weren't chat messages. Developers were using Evolve to sync application data such as game state in real time across their users. Tamplin and Lee decided to separate the chat system and the real-time architecture that powered it. They founded Firebase as a separate company in April 2012.

Firebase Inc. raised seed funding in May 2012. The company further raised Series a funding in June 2013. In October 2014, Firebase was acquired by Google. In October 2015, Google acquired Divshot to merge it with the Firebase team. Since the acquisition, Firebase has grown inside Google and expanded their services to become a unified platform for mobile developers. Firebase now integrates with various other Google services to offer broader products and scale for developers. In January 2017, Google acquired Fabric and Crashlytics from Twitter to join those services to the Firebase team. Firebase launched Cloud Fire store, a Document Database, in October 2017.

**Firebase has many services as following**

**Services**

* + - * 1. **Analytics**

Firebase Analytics is a cost-free app measurement solution that provides insight into app usage and user engagement.

* + - * 1. **Develop**

Firebase Cloud Messaging

Formerly known as Google Cloud Messaging (GCM), Firebase Cloud Messaging (FCM) is a cross-platform solution for messages and notifications for Android, iOS, and web applications, which as of 2016 can be used at no cost

* + - * 1. **Firebase Auth**

Firebase Auth is a service that can authenticate users using only client-side code. It supports social login providers Facebook, GitHub, Twitter and Google (and Google Play Games). Additionally, it includes a user management system whereby developers can enable user authentication with email and password login stored with Firebase

* + - * 1. **Real time Database**

Firebase provides a real time database and backend as a service. The service provides application developers an API that allows application data to be synchronized across clients and stored on Firebase's cloud The company provides client libraries that enable integration with Android, iOS, JavaScript, Java, Objective-C, Swift and Node.js applications. The database is also accessible through a REST API and bindings for several JavaScript frameworks such as AngularJS, React, Ember.js and Backbone.js. The REST API uses the Server-Sent Events protocol, which is an API for creating HTTP connections for receiving push notifications from a server. Developers using the real time database can secure their data by using the company's server-side-enforced security rules.

* + - * 1. **Firebase Storage**

Firebase Storage provides secure file uploads and downloads for Firebase apps, regardless of network quality. The developer can use it to store images, audio, video, or other user-generated content. Firebase Storage is backed by Google Cloud Storage

* + - * 1. **Firebase Hosting**

Firebase Hosting is a static and dynamic web hosting service that launched on May 13, 2014. It supports hosting static files such as CSS, HTML, JavaScript and other files, as well as support through Cloud Functions. The service delivers files over a content delivery network (CDN) through HTTP Secure (HTTPS) and Secure Sockets Layer encryption (SSL). Firebase partners with fastly, a CDN, to provide the CDN backing Firebase Hosting. The company states that Firebase Hosting grew out of customer requests; developers were using Firebase for its real-time database but needed a place to host their content.

* + - * 1. **Firebase App Indexing**

Firebase App Indexing, formerly Google App Indexing, gets an app into Google Search. Adding App Indexing promotes both types of app results within Google Search and also provides query auto completions.

* + - * 1. **Firebase Dynamic Links**

Firebase Dynamic Links are smart URLs that dynamically change behavior to provide the best experience across different platforms (website/iOS/Android) as well as deep link to APP.

* + - * 1. **Firebase Invites**

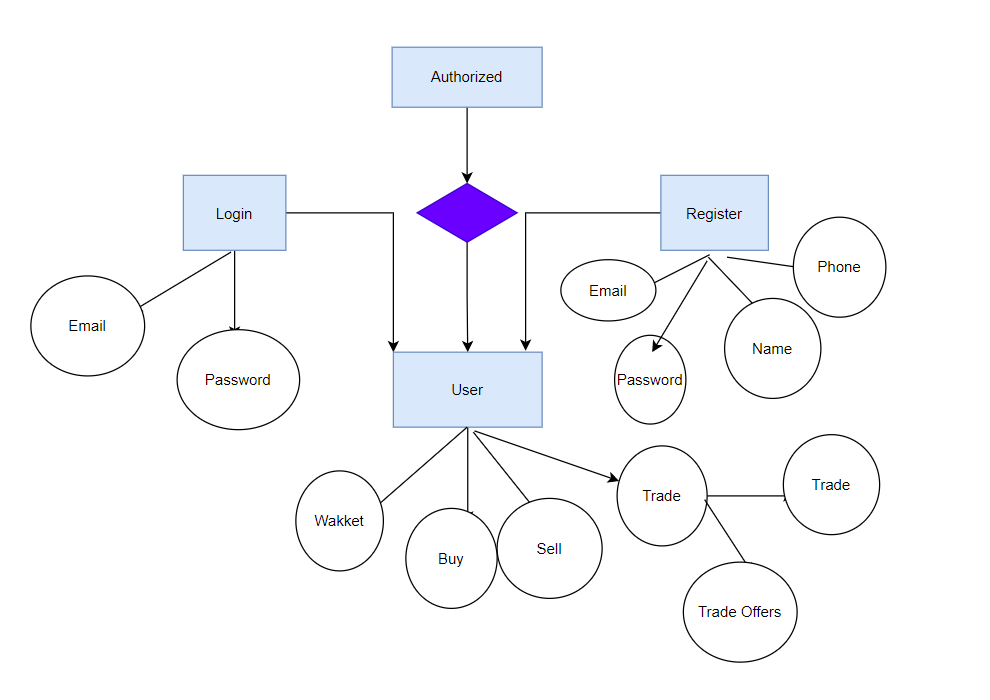
Firebase Invites is a cross-platform solution for sending personalized email and SMS invitations, on-boarding users, and measuring the impact of invitations

* + - * 1. **Firebase Remote Configuration**

Firebase Remote Configuration is a cloud service that lets developers change the behavior and appearance of their apps without requiring users to download an app update.

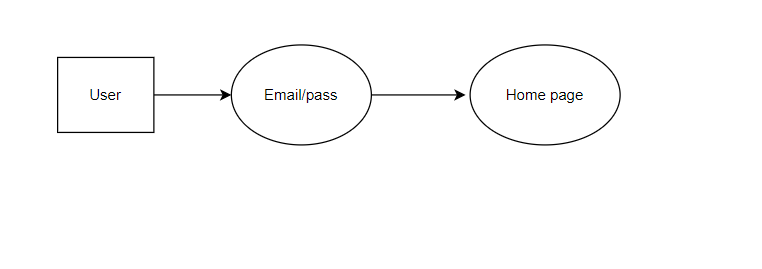
### Design Entity-Relationship Model

## Design the system and user interfaces



*Figer11:System and user interface*

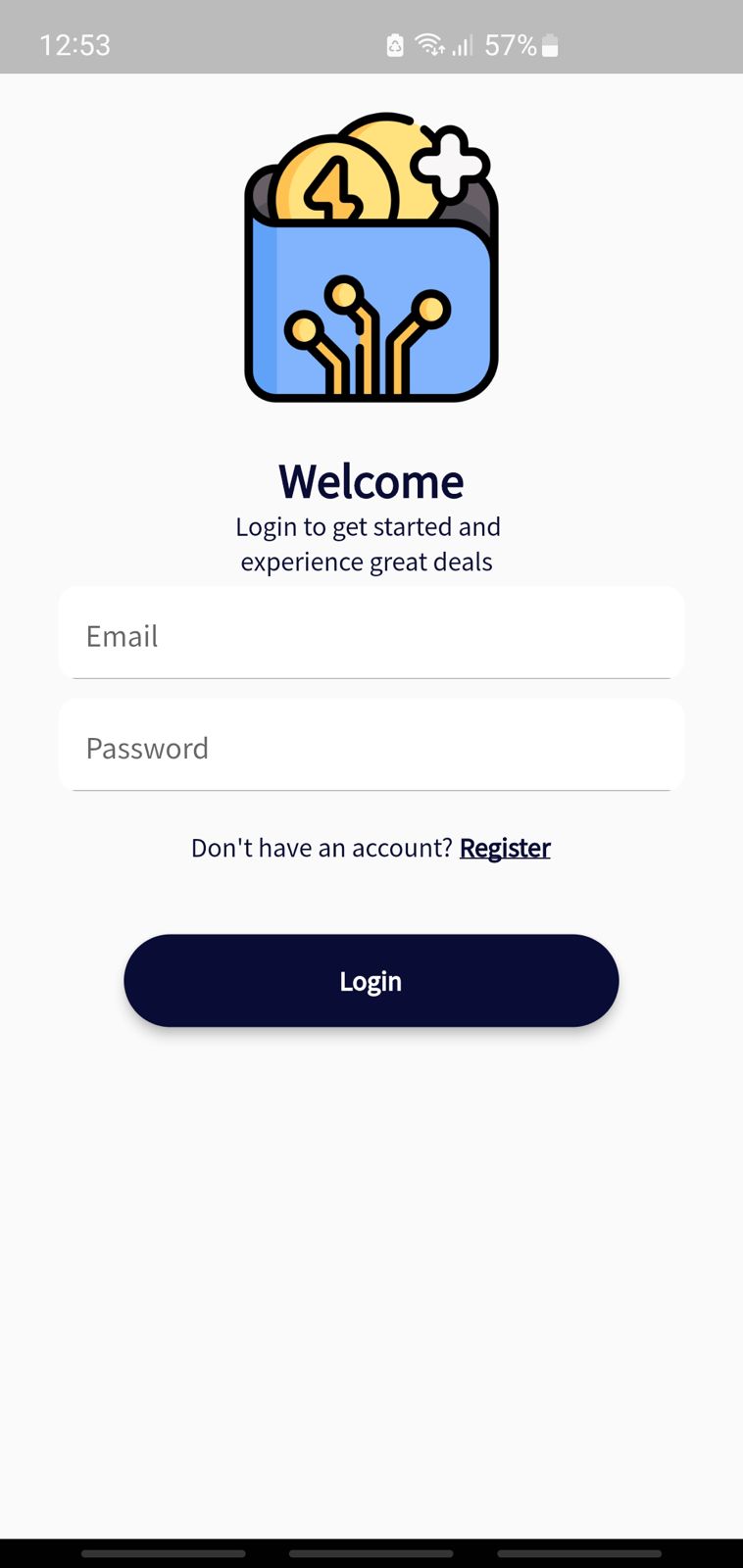
**Design The System security**

****

*Figer12:System Security*

**Design**

**Coins Verse App**

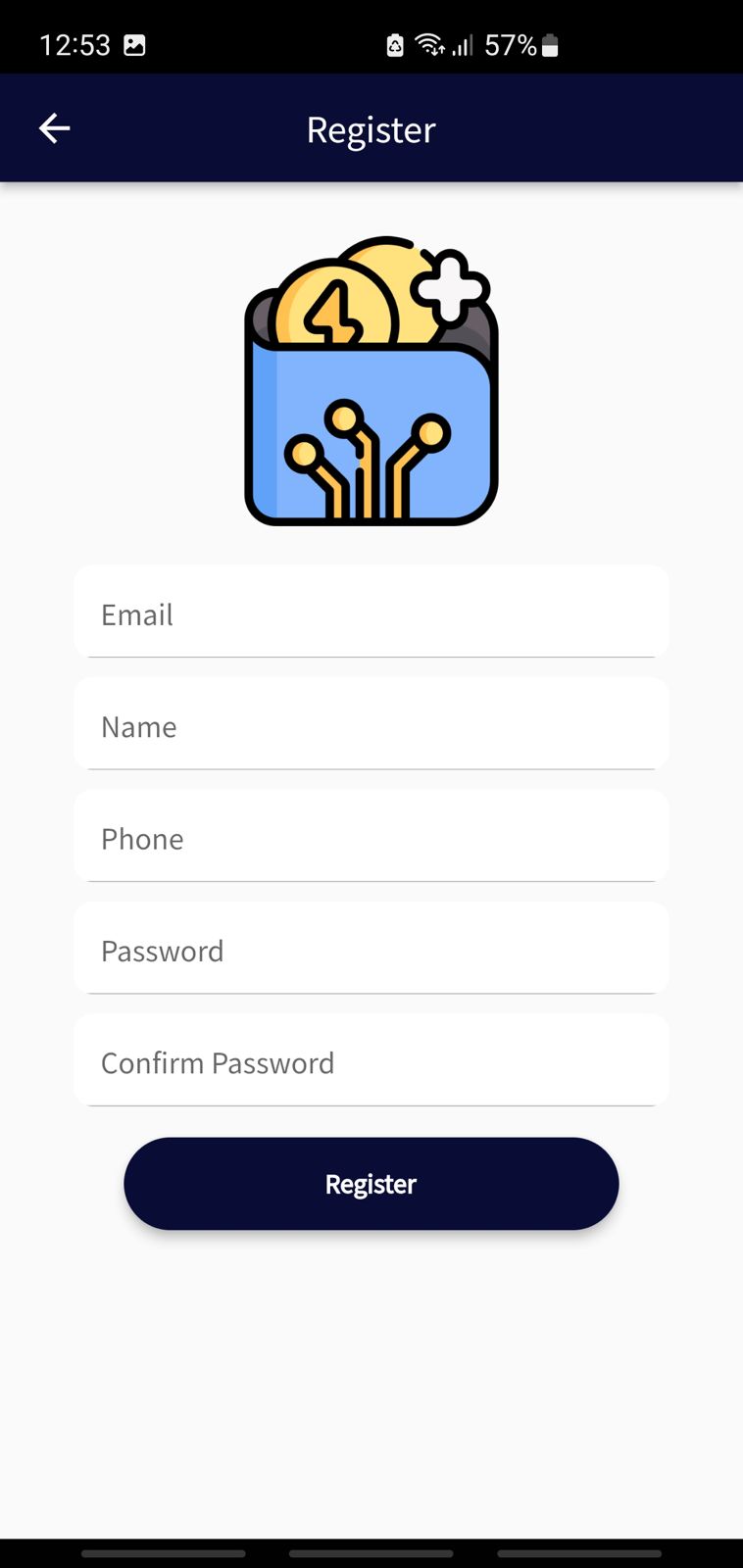
****

Login page

first page will show when you open the app with you can sign in if you have an account

and if you have an account already you can sign up easily using the

(sign up here button)



**3.6.2 Register**

Is where you create a new account in the app with easily filling the information under

**Contents**

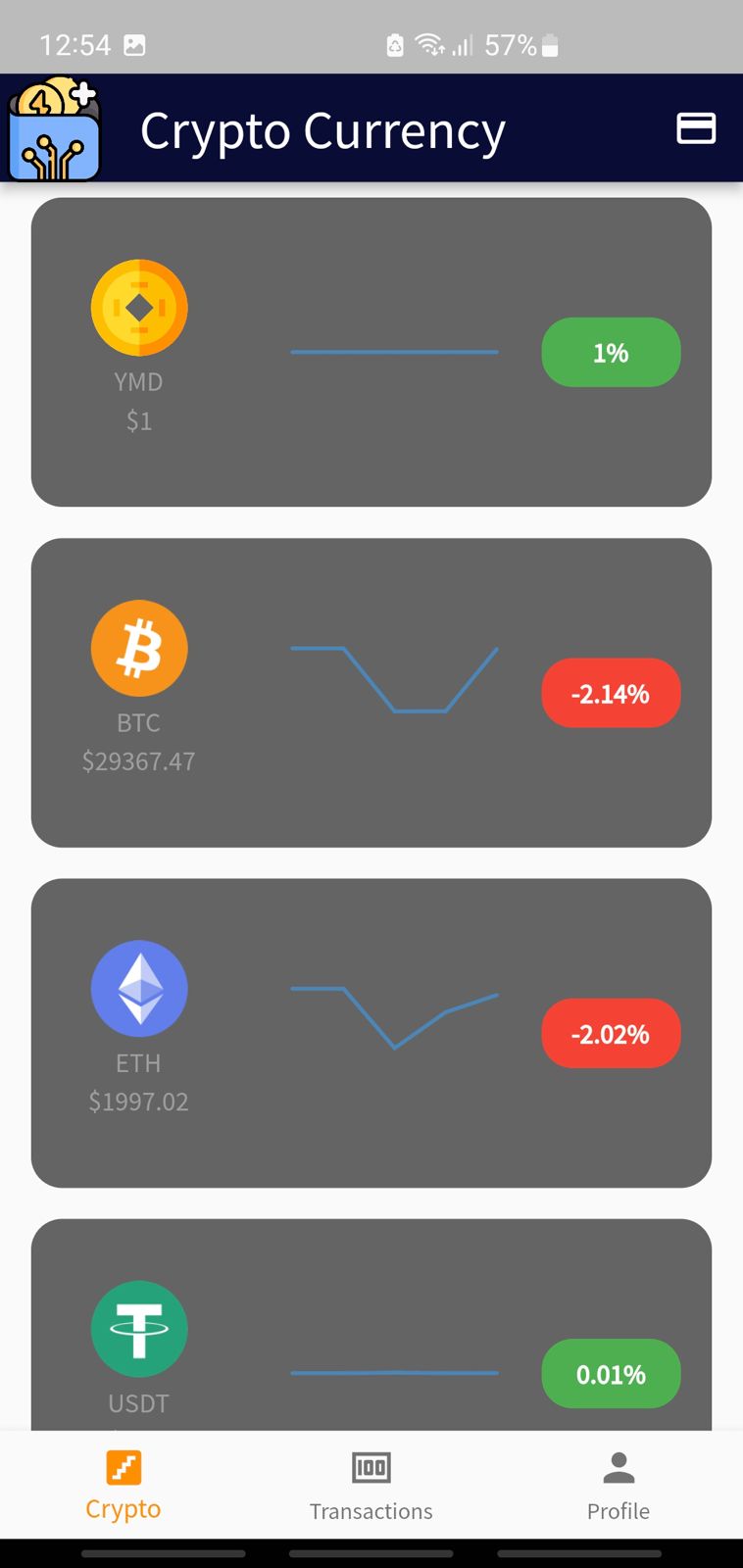
**1) Email**

**2) Name**

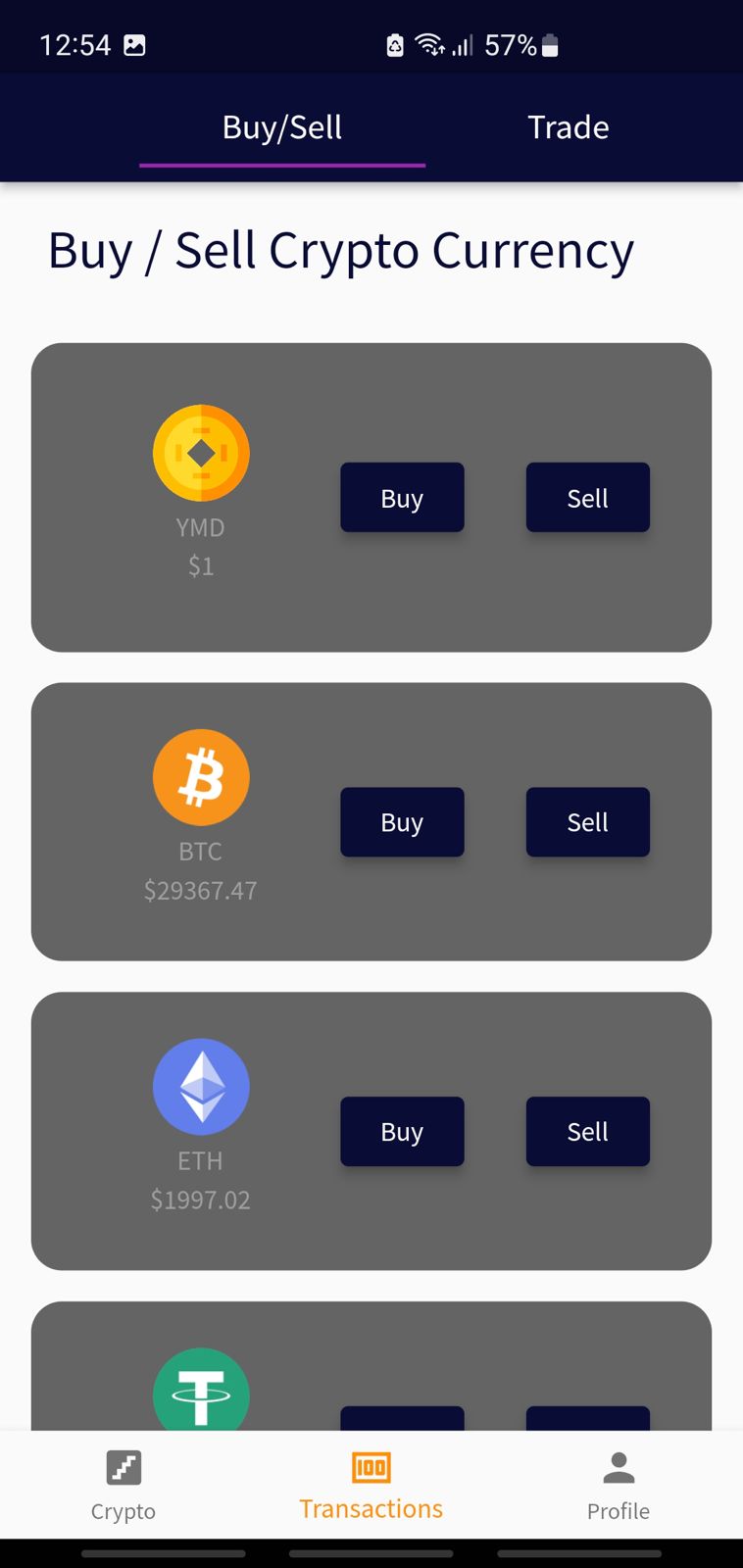
**3) Phone**

**4) Password**

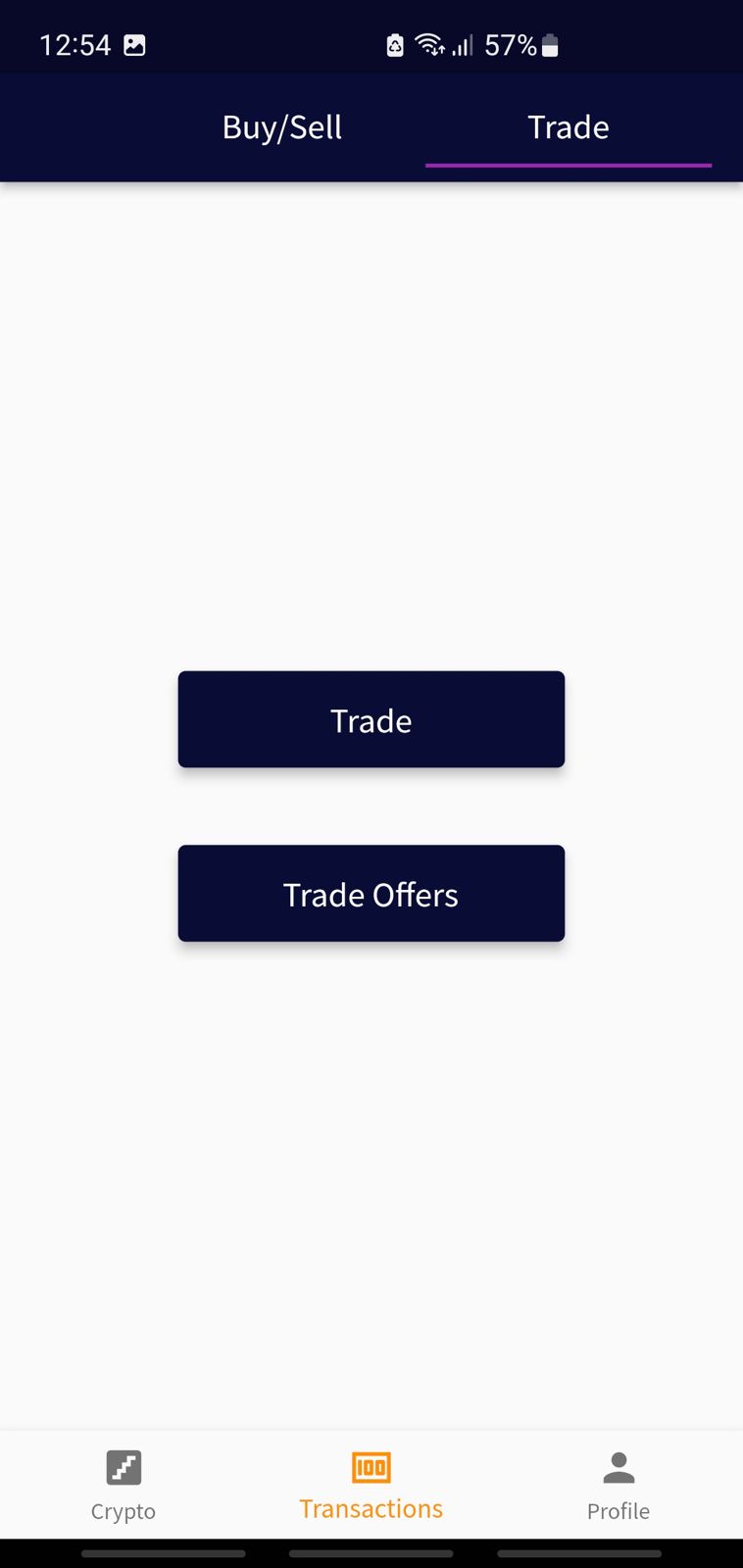
**5) Confirm Password**

****

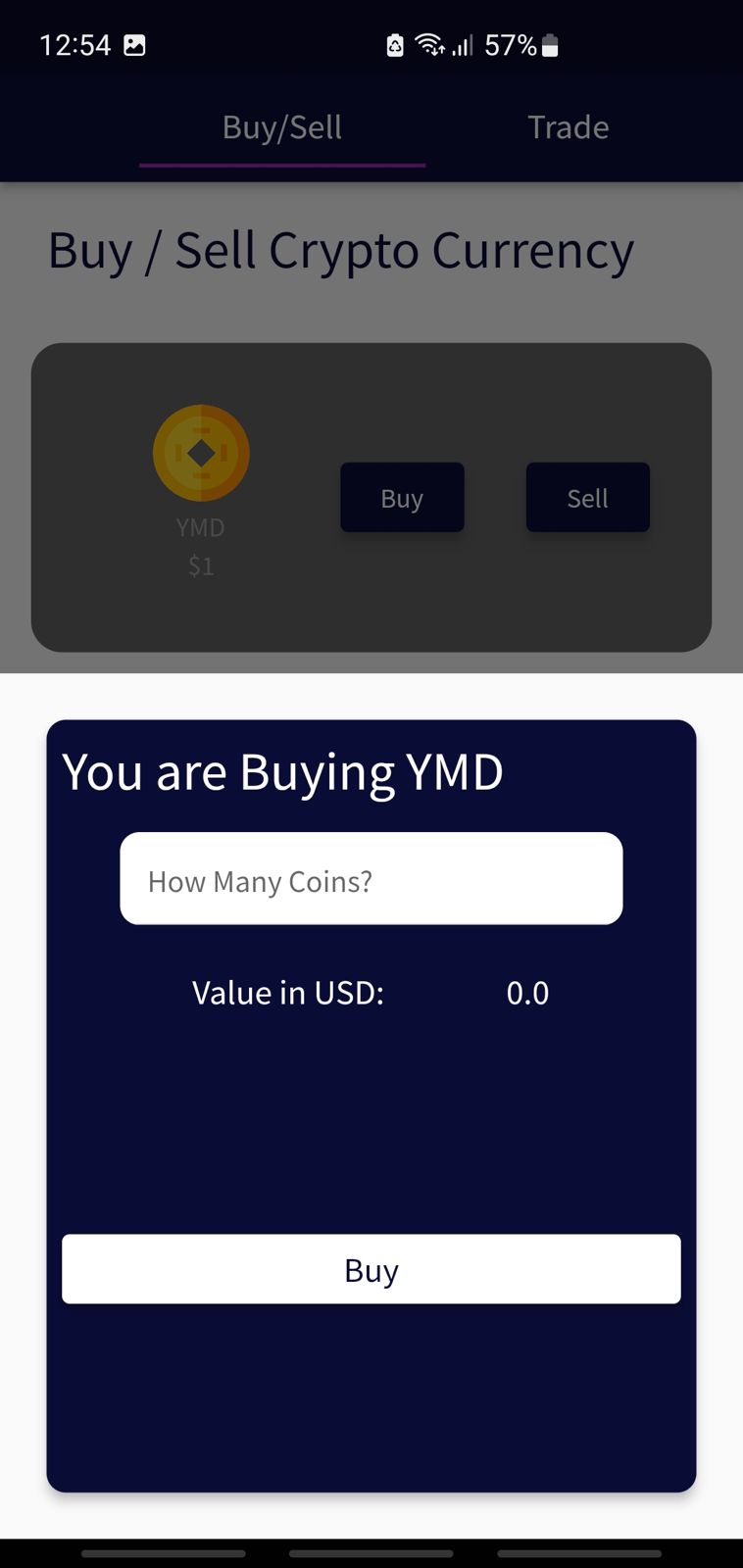
View all coins in the market

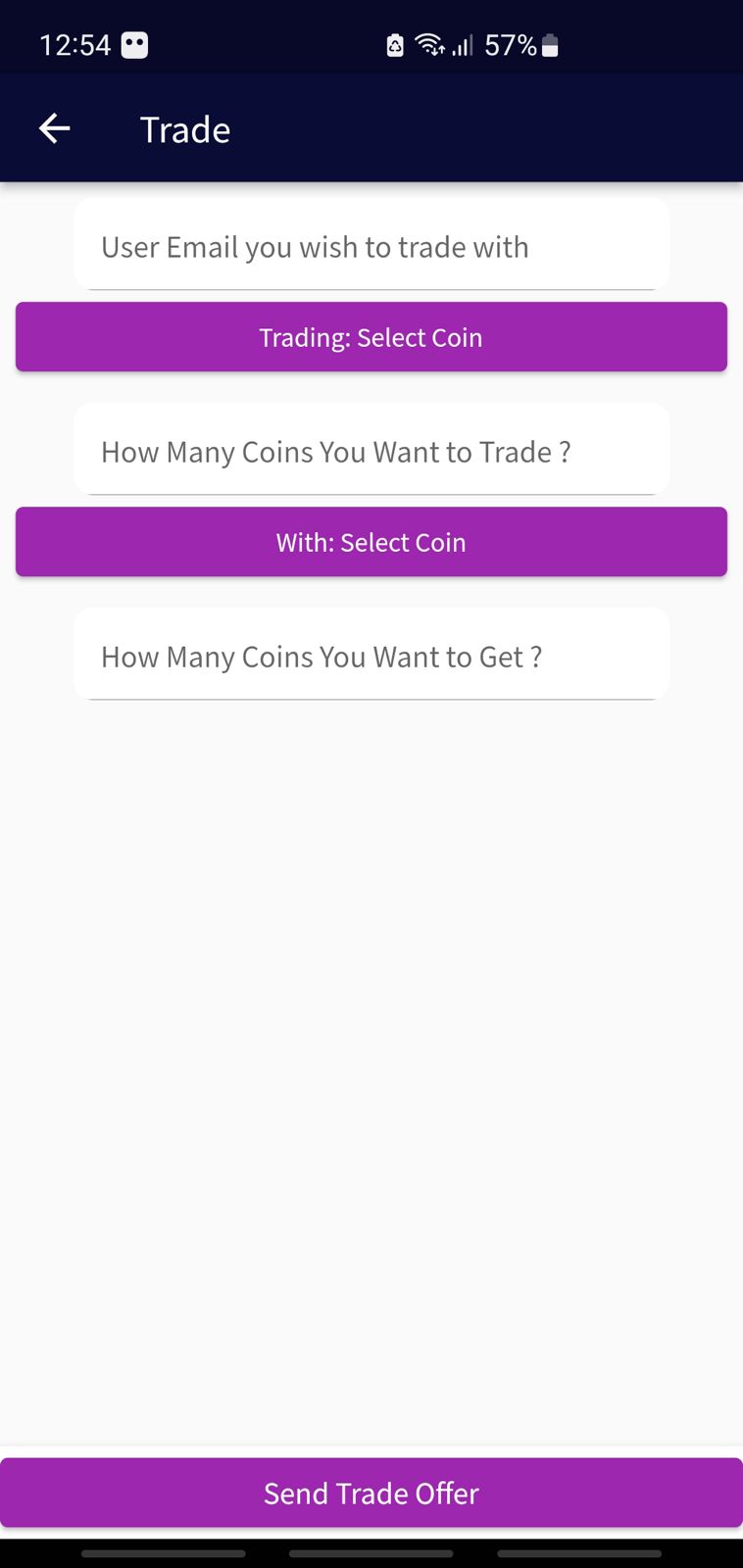


Choose if you’re buy or sell the currency

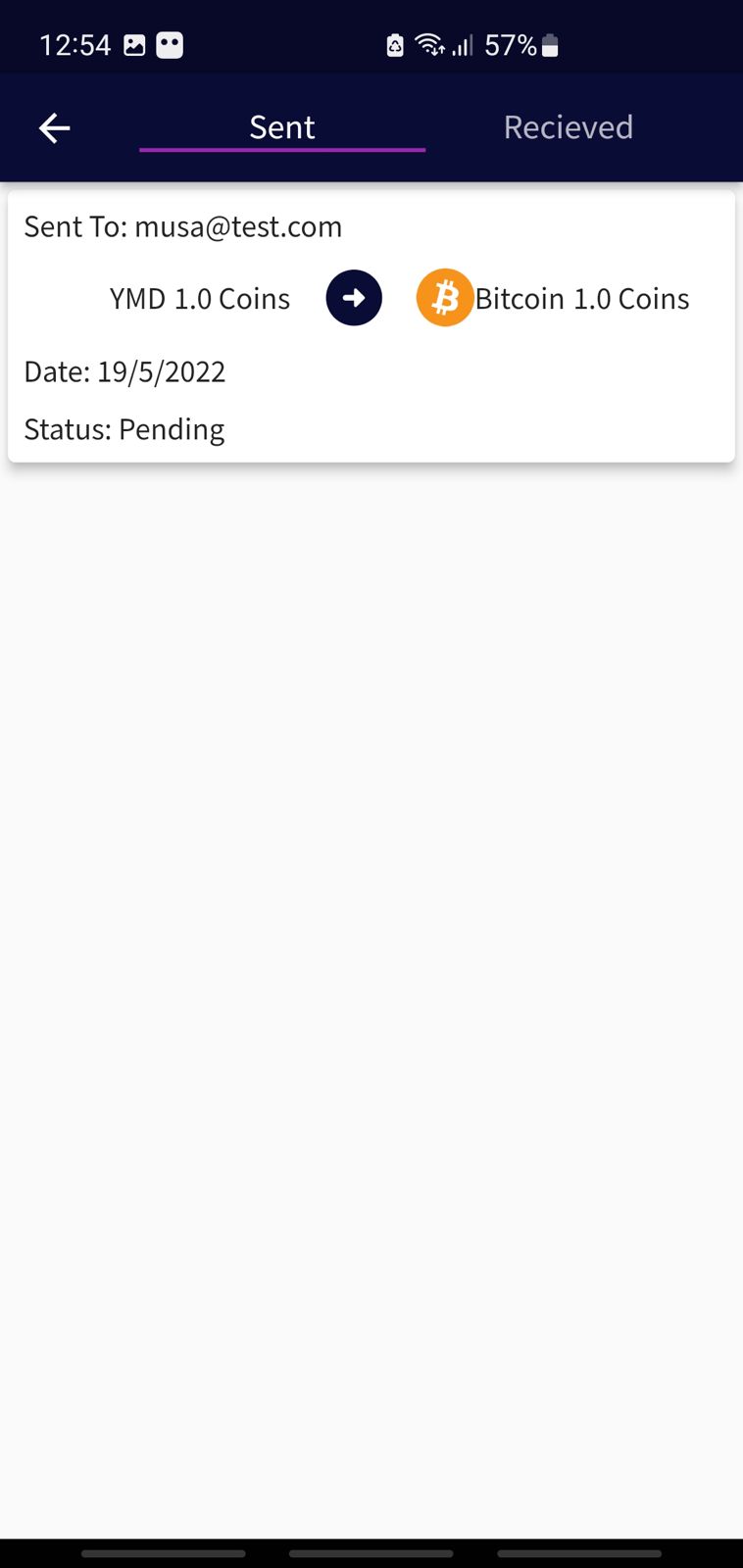


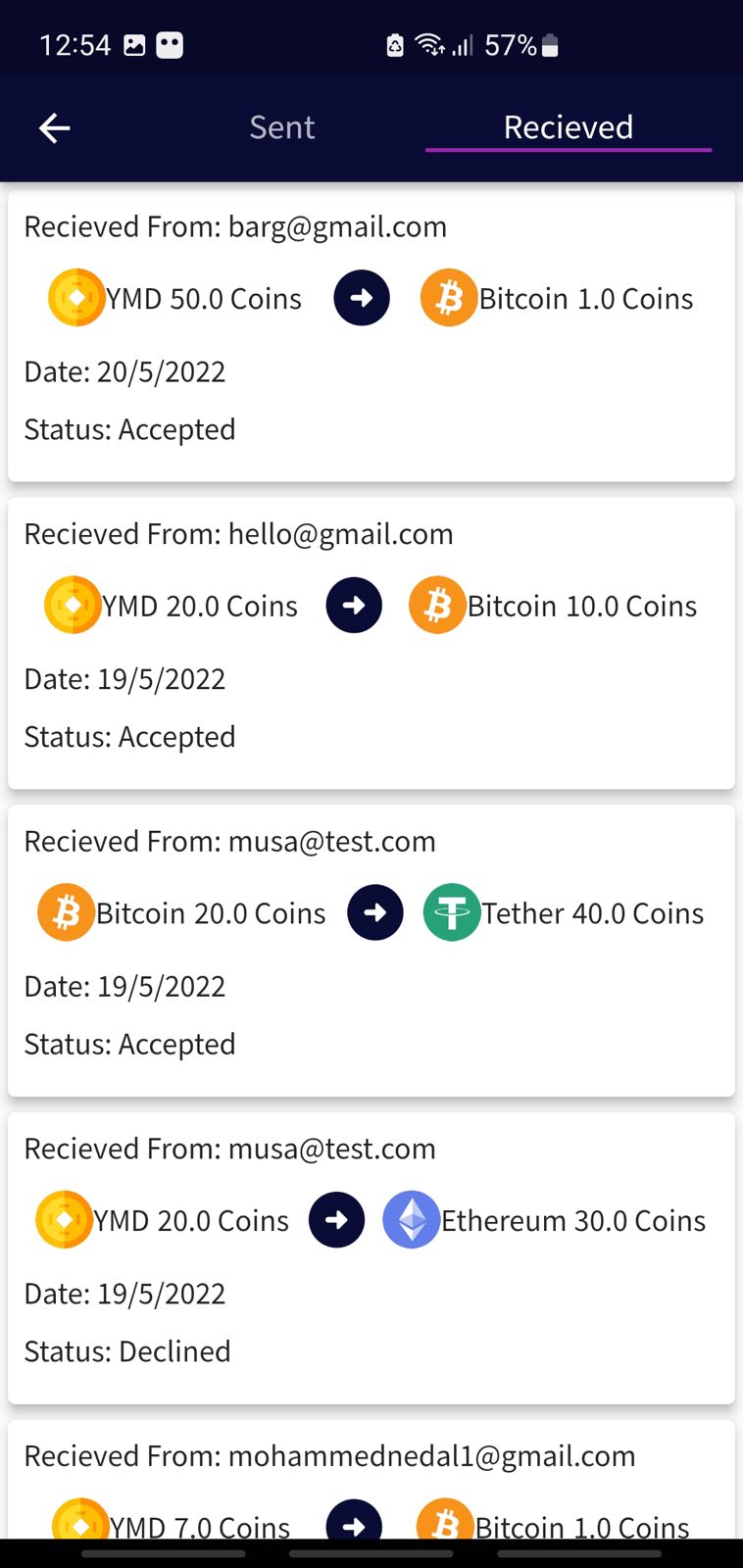
This page is based on the process of trading or accepting offers to trade

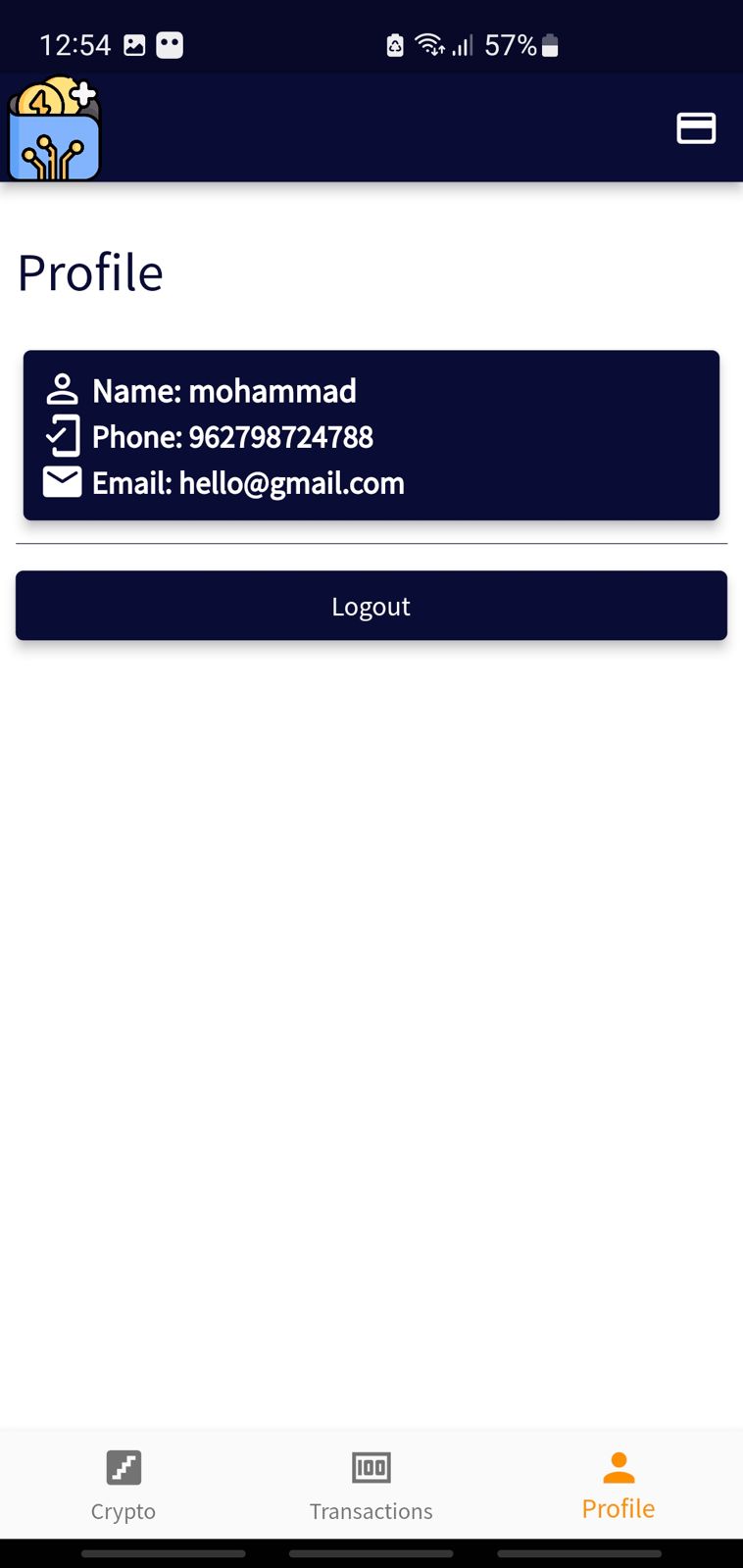


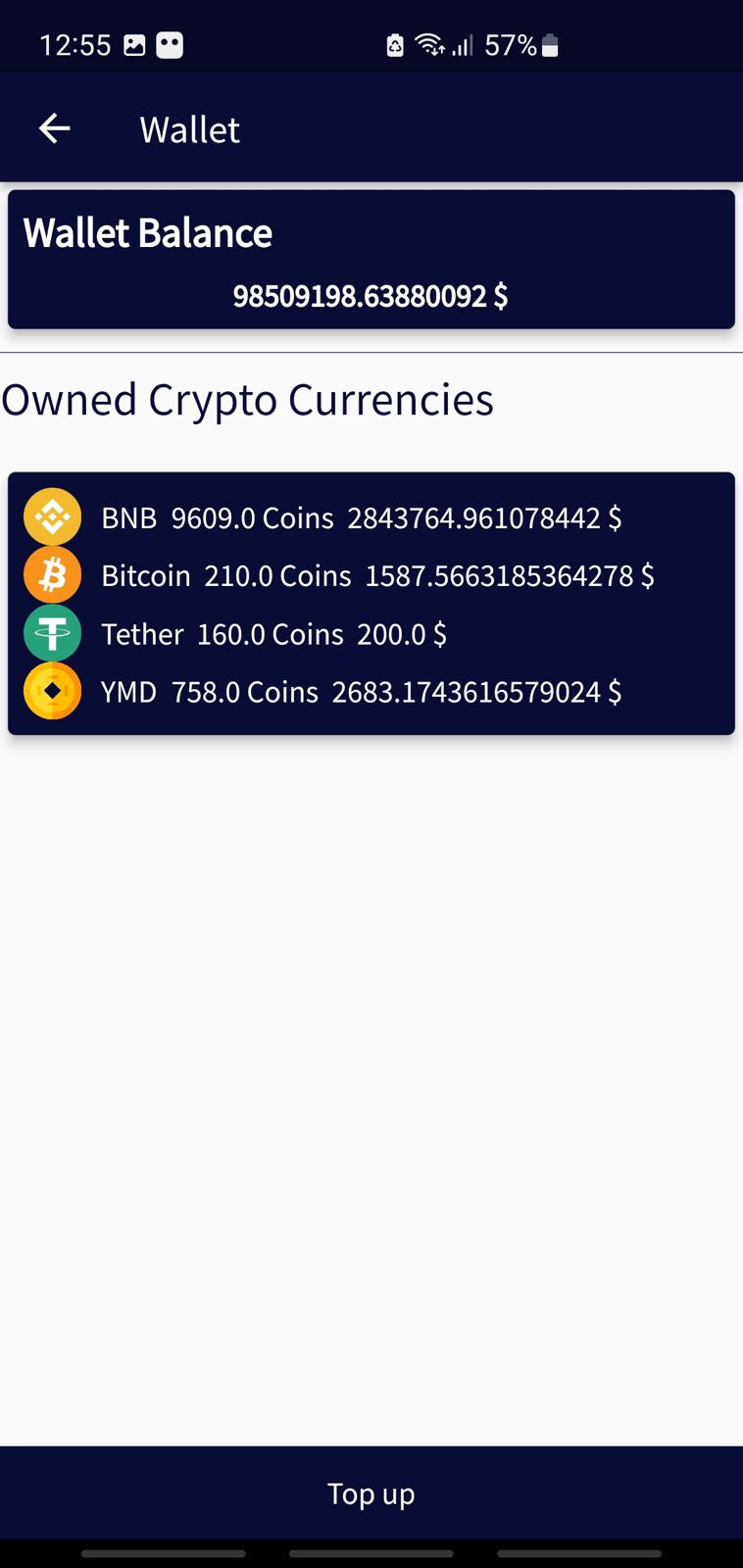


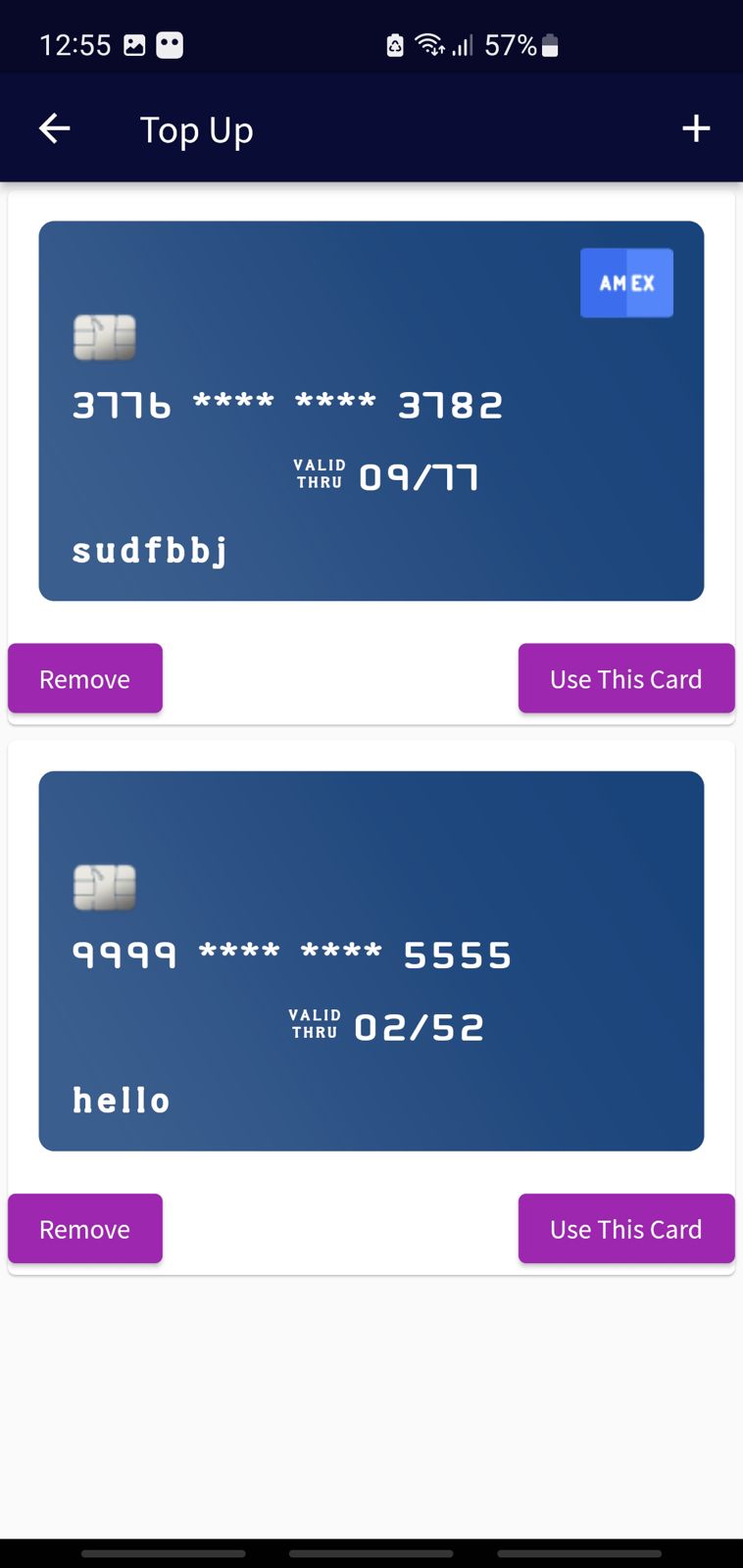
This page determines the e-mail of the person you want to trade with and determines the type of currency and the number of currencies to be traded

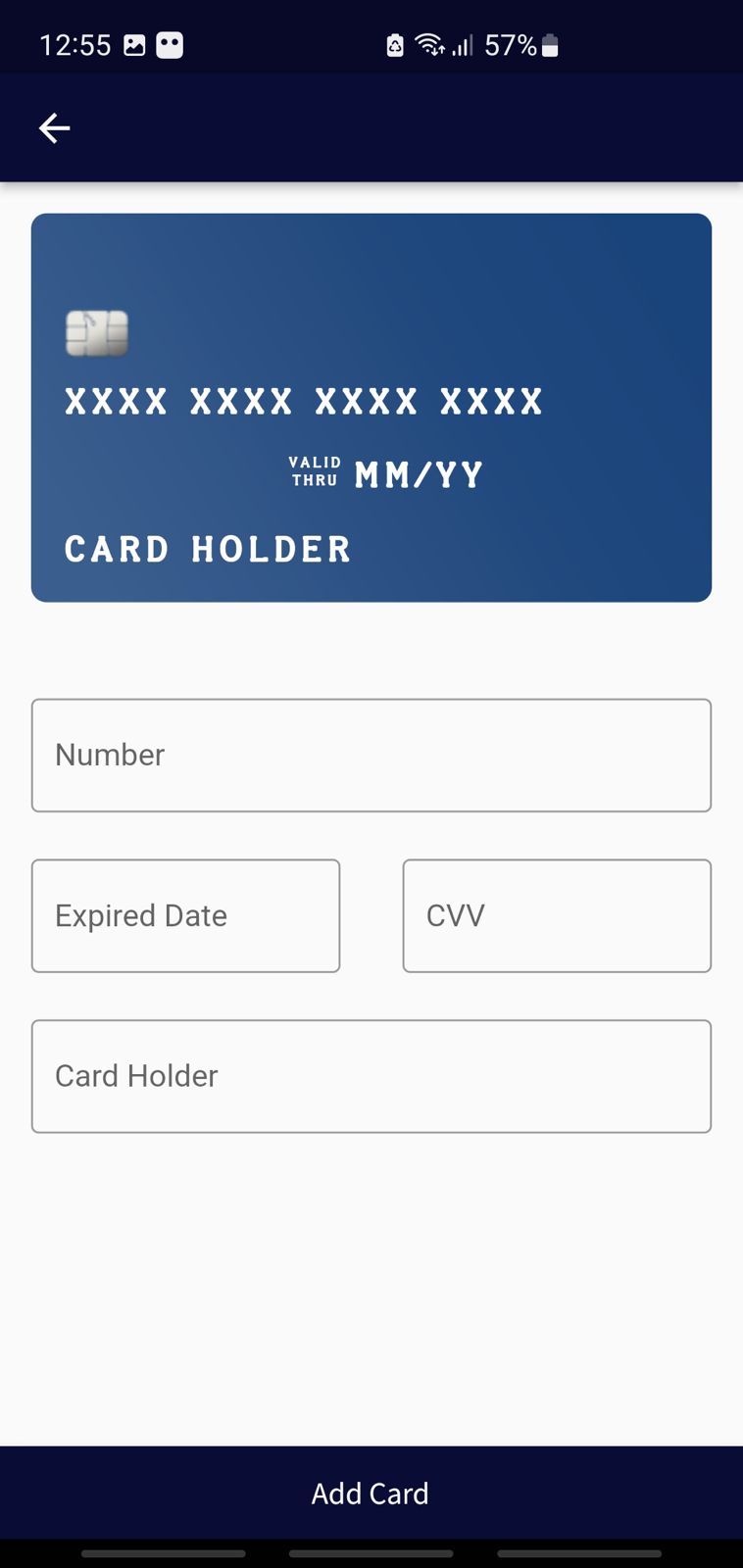












***Chapter Four***

***Implementation***

**Login:**

import 'package:crypto\_application/Screens/my\_home\_screen.dart';  
import 'package:crypto\_application/Screens/signup\_screen.dart';  
import 'package:crypto\_application/widgets/loader2.dart';  
import 'package:firebase\_auth/firebase\_auth.dart';  
import 'package:flutter/material.dart';  
import 'package:fluttertoast/fluttertoast.dart';  
*//import 'package:fluttertoast/fluttertoast.dart';  
  
//import 'package:shared\_preferences/shared\_preferences.dart';*class LoginScreen extends StatefulWidget {  
 static String *id* = 'loginScreen';  
  
 const LoginScreen({Key? key}) : super(key: key);  
  
 @override  
 \_LoginScreenState createState() => \_LoginScreenState();  
}  
  
class \_LoginScreenState extends State<LoginScreen> {  
 String email = "";  
 String password = "";  
 bool isLoading = false;  
 final \_auth = FirebaseAuth.*instance*;  
 final \_key = GlobalKey<FormState>();  
 bool loader = false;  
 @override  
 Widget build(BuildContext context) {  
 return Scaffold(  
 *//backgroundColor: Color.fromRGBO(11, 12, 54, 1),* body: SafeArea(  
 child: SingleChildScrollView(  
 child: Center(  
 child: Column(  
 crossAxisAlignment: CrossAxisAlignment.center,  
 children: [  
 Form(  
 key: \_key,  
 child: Column(  
 children: [  
 Padding(  
 padding: const EdgeInsets.symmetric(vertical: 20),  
 child: Image.asset(  
 'assets/images/crypto.png',  
 height: 150,  
 width: 150,  
 ),  
 ),  
 const Padding(  
 padding: EdgeInsets.symmetric(vertical: 0),  
 child: ListTile(  
 title: Center(  
 child: Text(  
 'Welcome',  
 style: TextStyle(  
 fontWeight: FontWeight.*bold*,  
 fontSize: 25,  
 color: Color.fromRGBO(11, 12, 54, 1)),  
 )),  
 subtitle: Center(  
 child: Text(  
 "Login to get started and \n experience great deals",  
 style: TextStyle(  
 color: Color.fromRGBO(11, 12, 54, 1)),  
 )),  
 ),  
 ),  
 loader  
 ? Loader2()  
 : Column(  
 children: [  
 Padding(  
 padding: const EdgeInsets.symmetric(  
 horizontal: 30.0),  
 child: TextFormField(  
 style: TextStyle(  
 color: Color.fromRGBO(11, 12, 54, 1)),  
 keyboardType: TextInputType.*emailAddress*,  
 cursorColor: Colors.*grey*,  
 onChanged: (val) {  
 email = val;  
 },  
 validator: (value) {  
 if (value == null ||  
 value.isEmpty ||  
 !value.toString().contains("@")) {  
 return 'Please enter a Vaild Email';  
 }  
 return null;  
 },  
 decoration: InputDecoration(  
 border: InputBorder.*none*,  
 hintText: 'Email',  
 filled: true,  
 fillColor: Colors.*white*,  
 contentPadding: const EdgeInsets.only(  
 left: 14.0,  
 bottom: 6.0,  
 top: 8.0),  
 focusedBorder: OutlineInputBorder(  
 borderSide:  
 BorderSide(color: Colors.*red*),  
 borderRadius:  
 BorderRadius.circular(10.0),  
 ),  
 enabledBorder: UnderlineInputBorder(  
 borderSide:  
 BorderSide(color: Colors.*grey*),  
 borderRadius:  
 BorderRadius.circular(10.0),  
 )),  
 ),  
 ),  
 const SizedBox(  
 height: 10.0,  
 ),  
 Padding(  
 padding: const EdgeInsets.symmetric(  
 horizontal: 30.0),  
 child: TextFormField(  
 style: TextStyle(color: Colors.*black*),  
 obscureText: true,  
 cursorColor: Colors.*grey*,  
 onChanged: (val) {  
 password = val;  
 },  
 validator: (value) {  
 if (value == null || value.isEmpty) {  
 return 'Please enter a Vaild Password';  
 }  
 return null;  
 },  
 decoration: InputDecoration(  
 border: InputBorder.*none*,  
 hintText: 'Password',  
 filled: true,  
 fillColor: Colors.*white*,  
 contentPadding:  
 const EdgeInsets.only(  
 left: 14.0,  
 bottom: 6.0,  
 top: 8.0),  
 focusedBorder: OutlineInputBorder(  
 borderSide:  
 BorderSide(color: Colors.*red*),  
 borderRadius:  
 BorderRadius.circular(10.0),  
 ),  
 enabledBorder: UnderlineInputBorder(  
 borderSide: BorderSide(  
 color: Colors.*grey*),  
 borderRadius:  
 BorderRadius.circular(10.0),  
 )),  
 )),  
 const SizedBox(  
 height: 20,  
 ),  
 Row(  
 mainAxisAlignment: MainAxisAlignment.center,  
 mainAxisSize: MainAxisSize.min,  
 children: [  
 Text(  
 'Don\'t have an account? ',  
 style: TextStyle(  
 color:  
 Color.fromRGBO(11, 12, 54, 1)),  
 ),  
 InkWell(  
 onTap: () {  
 Navigator.*pushNamed*(  
 context, SignUpScreen.*id*);  
 },  
 child: Text(  
 'Register',  
 style: TextStyle(  
 color: Color.fromRGBO(  
 11, 12, 54, 1),  
 decoration:  
 TextDecoration.*underline*,  
 fontWeight: FontWeight.*bold*),  
 ))  
 ],  
 ),  
 const SizedBox(  
 height: 20,  
 ),  
 Padding(  
 padding: const EdgeInsets.symmetric(  
 vertical: 16.0),  
 child: Material(  
 color: Color.fromRGBO(11, 12, 54, 1),  
 elevation: 5.0,  
 borderRadius: BorderRadius.circular(30.0),  
 child: MaterialButton(  
 onPressed: () async {  
 setState(() {  
 loader = true;  
 });  
 try {  
 await \_auth  
 .signInWithEmailAndPassword(  
 email: email,  
 password: password);  
 setState(() {  
 loader = false;  
 });  
 Navigator.*pushNamed*(  
 context, MyHomeScreen.*id*);  
 } catch (e) {  
 setState(() {  
 loader = false;  
 });  
 Fluttertoast.*showToast*(  
 msg: e.toString());  
 }  
 },  
 minWidth:  
 MediaQuery.*of*(context).size.width /  
 1.5,  
 height: 42.0,  
 child: const Text(  
 'Login',  
 style: TextStyle(  
 color: Colors.*white*,  
 fontWeight: FontWeight.*bold*),

**Signup:**

import 'package:crypto\_application/data\_base.dart';  
import 'package:crypto\_application/widgets/loader2.dart';  
import 'package:flutter/material.dart';  
import 'package:fluttertoast/fluttertoast.dart';  
  
import 'my\_home\_screen.dart';  
  
class SignUpScreen extends StatefulWidget {  
 const SignUpScreen({Key? key}) : super(key: key);  
 static String *id* = 'SignupScreen';  
 @override  
 \_SignUpScreenState createState() => \_SignUpScreenState();  
}  
  
class \_SignUpScreenState extends State<SignUpScreen> {  
 String email = '';  
 String name = '';  
 String phone = '';  
 String password = '';  
 String confirmPassword = '';  
 bool loader = false;  
 @override  
 Widget build(BuildContext context) {  
 return SafeArea(  
 child: Scaffold(  
 *// backgroundColor: Color.fromRGBO(11, 12, 54, 1),* appBar: AppBar(  
 backgroundColor: Color.fromRGBO(11, 12, 54, 1),  
 centerTitle: true,  
 title: Text(  
 'Register',  
 style: TextStyle(color: Colors.*white*),  
 ),  
 ),  
 body: Padding(  
 padding: const EdgeInsets.all(8.0),  
 child: Form(  
 child: Column(  
 children: [  
 Padding(  
 padding: const EdgeInsets.symmetric(vertical: 20),  
 child: Image.asset(  
 'assets/images/crypto.png',  
 height: 150,  
 width: 150,  
 ),  
 ),  
 Padding(  
 padding: const EdgeInsets.symmetric(horizontal: 30.0),  
 child: TextFormField(  
 style: TextStyle(color: Color.fromRGBO(11, 12, 54, 1)),  
 keyboardType: TextInputType.*emailAddress*,  
 cursorColor: Colors.*grey*,  
 onChanged: (val) {  
 email = val;  
 },  
 validator: (value) {  
 if (value == null ||  
 value.isEmpty ||  
 !value.toString().contains("@")) {  
 return 'Please enter a Vaild Email';  
 }  
 return null;  
 },  
 decoration: InputDecoration(  
 border: InputBorder.*none*,  
 hintText: 'Email',  
 filled: true,  
 fillColor: Colors.*white*,  
 contentPadding: const EdgeInsets.only(  
 left: 14.0, bottom: 6.0, top: 8.0),  
 focusedBorder: OutlineInputBorder(  
 borderSide: BorderSide(color: Colors.*red*),  
 borderRadius: BorderRadius.circular(10.0),  
 ),  
 enabledBorder: UnderlineInputBorder(  
 borderSide: BorderSide(color: Colors.*grey*),  
 borderRadius: BorderRadius.circular(10.0),  
 )),  
 ),  
 ),  
 const SizedBox(  
 height: 10.0,  
 ),  
 Padding(  
 padding: const EdgeInsets.symmetric(horizontal: 30.0),  
 child: TextFormField(  
 style: TextStyle(color: Color.fromRGBO(11, 12, 54, 1)),  
 keyboardType: TextInputType.*text*,  
 cursorColor: Colors.*grey*,  
 onChanged: (val) {  
 name = val;  
 },  
 validator: (value) {  
 if (value == null || value.isEmpty) {  
 return 'Please enter a Name';  
 }  
 return null;  
 },  
 decoration: InputDecoration(  
 border: InputBorder.*none*,  
 hintText: 'Name',  
 filled: true,  
 fillColor: Colors.*white*,  
 contentPadding: const EdgeInsets.only(  
 left: 14.0, bottom: 6.0, top: 8.0),  
 focusedBorder: OutlineInputBorder(  
 borderSide: BorderSide(color: Colors.*red*),  
 borderRadius: BorderRadius.circular(10.0),  
 ),  
 enabledBorder: UnderlineInputBorder(  
 borderSide: BorderSide(color: Colors.*grey*),  
 borderRadius: BorderRadius.circular(10.0),  
 )),  
 ),  
 ),  
 const SizedBox(  
 height: 10.0,  
 ),  
 Padding(  
 padding: const EdgeInsets.symmetric(horizontal: 30.0),  
 child: TextFormField(  
 style: TextStyle(color: Color.fromRGBO(11, 12, 54, 1)),  
 keyboardType: TextInputType.*number*,  
 cursorColor: Colors.*grey*,  
 onChanged: (val) {  
 phone = val;  
 },  
 validator: (value) {  
 if (value == null || value.isEmpty) {  
 return 'Please enter a Phone Number';  
 }  
 return null;  
 },  
 decoration: InputDecoration(  
 border: InputBorder.*none*,  
 hintText: 'Phone',  
 filled: true,  
 fillColor: Colors.*white*,  
 contentPadding: const EdgeInsets.only(  
 left: 14.0, bottom: 6.0, top: 8.0),  
 focusedBorder: OutlineInputBorder(  
 borderSide: BorderSide(color: Colors.*red*),  
 borderRadius: BorderRadius.circular(10.0),  
 ),  
 enabledBorder: UnderlineInputBorder(  
 borderSide: BorderSide(color: Colors.*grey*),  
 borderRadius: BorderRadius.circular(10.0),  
 )),  
 ),  
 ),  
 const SizedBox(  
 height: 10.0,  
 ),  
 Padding(  
 padding: const EdgeInsets.symmetric(horizontal: 30.0),  
 child: TextFormField(  
 obscureText: true,  
 cursorColor: Colors.*grey*,  
 style: TextStyle(color: Color.fromRGBO(11, 12, 54, 1)),  
 onChanged: (val) {  
 password = val;  
 },  
 validator: (value) {  
 if (value == null || value.isEmpty) {  
 return 'Please enter a Vaild Password';  
 }  
 return null;  
 },  
 decoration: InputDecoration(  
 border: InputBorder.*none*,  
 hintText: 'Password',  
 filled: true,  
 fillColor: Colors.*white*,  
 contentPadding: const EdgeInsets.only(  
 left: 14.0, bottom: 6.0, top: 8.0),  
 focusedBorder: OutlineInputBorder(  
 borderSide: BorderSide(color: Colors.*red*),  
 borderRadius: BorderRadius.circular(10.0),  
 ),  
 enabledBorder: UnderlineInputBorder(  
 borderSide: BorderSide(color: Colors.*grey*),  
 borderRadius: BorderRadius.circular(10.0),  
 )),  
 )),  
 const SizedBox(  
 height: 10.0,  
 ),  
 Padding(  
 padding: const EdgeInsets.symmetric(horizontal: 30.0),  
 child: TextFormField(  
 style: TextStyle(color: Color.fromRGBO(11, 12, 54, 1)),  
 obscureText: true,  
 cursorColor: Colors.*grey*,  
 onChanged: (val) {  
 confirmPassword = val;  
 },  
 validator: (v) =>  
 v == password ? null : "Password not match",  
 decoration: InputDecoration(  
 border: InputBorder.*none*,  
 hintText: 'Confirm Password',  
 filled: true,  
 fillColor: Colors.*white*,  
 contentPadding: const EdgeInsets.only(  
 left: 14.0, bottom: 6.0, top: 8.0),  
 focusedBorder: OutlineInputBorder(  
 borderSide: BorderSide(color: Colors.*red*),  
 borderRadius: BorderRadius.circular(10.0),  
 ),  
 enabledBorder: UnderlineInputBorder(  
 borderSide: BorderSide(color: Colors.*grey*),  
 borderRadius: BorderRadius.circular(10.0),  
 )),  
 )),  
 loader  
 ? Loader2()  
 : Padding(  
 padding: const EdgeInsets.symmetric(vertical: 16.0),  
 child: Material(  
 color: Color.fromRGBO(11, 12, 54, 1),  
 elevation: 5.0,  
 borderRadius: BorderRadius.circular(30.0),  
 child: MaterialButton(  
 onPressed: () async {  
 setState(() {  
 loader = true;  
 });  
 try {  
 await Database()  
 .createUserData(name, email, phone, password);  
 setState(() {  
 loader = false;  
 });  
 Navigator.*pushNamed*(context, MyHomeScreen.*id*);  
 } catch (e) {  
 setState(() {  
 loader = false;  
 });  
 Fluttertoast.*showToast*(msg: e.toString());  
 }  
 },  
 minWidth: MediaQuery.*of*(context).size.width / 1.5,  
 height: 42.0,  
 child: const Text(  
 'Register',  
 style: TextStyle(  
 color: Colors.*white*,  
 fontWeight: FontWeight.*bold*),

**Buy trade:**

import 'package:crypto\_application/Screens/second\_screen.dart';  
import 'package:crypto\_application/Screens/trade\_Screen.dart';  
import 'package:flutter/material.dart';  
  
class SwitchBuyTrade extends StatefulWidget {  
 const SwitchBuyTrade({Key? key}) : super(key: key);  
  
 @override  
 State<SwitchBuyTrade> createState() => \_SwitchBuyTradeState();  
}  
  
class \_SwitchBuyTradeState extends State<SwitchBuyTrade>  
 with TickerProviderStateMixin {  
 late TabController \_tabController;  
  
 @override  
 void initState() {  
 super.initState();  
 \_tabController = TabController(length: 2, vsync: this);  
 \_tabController.animateTo(0);  
 }  
  
 @override  
 Widget build(BuildContext context) {  
 return Scaffold(  
 appBar: AppBar(  
 backgroundColor: Color.fromRGBO(11, 12, 54, 1),  
 leading: Text(''),  
 title: TabBar(  
 controller: \_tabController,  
 tabs: [  
 Padding(  
 padding: const EdgeInsets.all(8.0),  
 child: Text(  
 'Buy/Sell',  
 style: TextStyle(fontSize: 18, color: Colors.*white*),  
 ),  
 ),  
 Padding(  
 padding: const EdgeInsets.all(8.0),  
 child: Text(  
 'Trade',  
 style: TextStyle(fontSize: 18, color: Colors.*white*),  
 ),  
 ),  
 ],  
 ),  
 ),  
 body: TabBarView(  
 controller: \_tabController,  
 children: [SecondScreen(), TradeScreen()]),

Coin detail:

import 'dart:math';  
  
import 'package:crypto\_application/models/models.dart';  
import 'package:crypto\_application/widgets/widgets.dart';  
import 'package:flutter/material.dart';  
  
import 'package:crypto\_application/models/fetchCoins\_models/fetch\_coins\_models.dart';  
import 'package:intl/intl.dart';  
  
class CoinDetailScreen extends StatelessWidget {  
 final DataModel coin;  
 const CoinDetailScreen({  
 Key? key,  
 required this.coin,  
 }) : super(key: key);  
  
 @override  
 Widget build(BuildContext context) {  
 Random random = Random();  
 int next(int min, int max) => random.nextInt(max - min);  
 var coinIconUrl =  
 "https://raw.githubusercontent.com/spothq/cryptocurrency-icons/master/128/color/";  
 var coinPrice = coin.quoteModel.usdModel;  
 DateTime parseDate = new DateFormat("yyyy-MM-dd'T'HH:mm:ss.SSS'Z`'")  
 .parse(coinPrice.lastUpdated);  
 var inputDate = DateTime.*parse*(parseDate.toString());  
 var outputFormat = DateFormat('MM/dd/yyyy hh:mm a');  
 var outputDate = outputFormat.format(inputDate);  
 var data = [  
 ChartData(next(110, 140), 1),  
 ChartData(next(9, 41), 2),  
 ChartData(next(140, 200), 3),  
 ChartData(coinPrice.percentChange\_24h, 4),  
 ChartData(coinPrice.percentChange\_1h, 5),  
 ChartData(next(110, 140), 6),  
 ChartData(next(9, 41), 7),  
 ChartData(next(140, 200), 8),  
 ChartData(coinPrice.percentChange\_24h, 9),  
 ChartData(coinPrice.percentChange\_1h, 10),  
 ChartData(next(110, 140), 12),  
 ChartData(next(9, 41), 13),  
 ChartData(coinPrice.percentChange\_1h, 14),  
 ChartData(next(9, 41), 15),  
 ChartData(next(140, 200), 16),  
 ChartData(coinPrice.percentChange\_24h, 17),  
 ChartData(coinPrice.percentChange\_1h, 18),  
 ChartData(next(110, 140), 19),  
 ChartData(next(9, 41), 20),  
 ChartData(next(140, 200), 21),  
 ChartData(coinPrice.percentChange\_24h, 22),  
 ChartData(next(110, 140), 23),  
 ];  
  
 return Scaffold(  
 *// backgroundColor: Color.fromRGBO(11, 12, 54, 1),* body: CustomScrollView(  
 slivers: [  
 CoinDetailAppBar(coin: coin, coinIconUrl: coinIconUrl),  
 CoinRandomedChartWidget(  
 coinPrice: coinPrice, outputDate: outputDate, data: data),  
 SliverToBoxAdapter(  
 child: Container(  
 height: 600,  
 child: Column(  
 mainAxisAlignment: MainAxisAlignment.center,  
 children: [  
 Container(  
 margin: const EdgeInsets.symmetric(horizontal: 16.0),  
 height: 400.0,  
 width: double.*infinity*,  
 child: Column(  
 children: [  
 Row(  
 mainAxisAlignment: MainAxisAlignment.spaceAround,  
 children: [  
 Text(  
 "Circulating Supply: ",  
 *// style: Theme.of(context).textTheme.subtitle1,* style: TextStyle(  
 color: Color.fromRGBO(11, 12, 54, 1)),  
 ),  
 Text(  
 coin.circulatingSupply.toString(),  
 *// style: Theme.of(context).textTheme.headline6,* style: TextStyle(  
 color: Color.fromRGBO(11, 12, 54, 1)),  
 ),  
 ],  
 ),  
 const SizedBox(height: 8.0),  
 Row(  
 mainAxisAlignment: MainAxisAlignment.spaceAround,  
 children: [  
 Text(  
 "Max Supply: ",  
 *// style: Theme.of(context).textTheme.subtitle1,* style: TextStyle(  
 color: Color.fromRGBO(11, 12, 54, 1)),  
 ),  
 Text(  
 coin.maxSupply.toString(),  
 *// style: Theme.of(context).textTheme.headline6,* style: TextStyle(  
 color: Color.fromRGBO(11, 12, 54, 1)),  
 ),  
 ],  
 ),  
 const SizedBox(height: 8.0),  
 Row(  
 mainAxisAlignment: MainAxisAlignment.spaceAround,  
 children: [  
 Text(  
 "Market pairs: ",  
 *// style: Theme.of(context).textTheme.subtitle1,* style: TextStyle(  
 color: Color.fromRGBO(11, 12, 54, 1)),  
 ),  
 Text(  
 coin.numMarketPairs.toString(),  
 *// style: Theme.of(context).textTheme.headline6,* style: TextStyle(  
 color: Color.fromRGBO(11, 12, 54, 1)),  
 ),  
 ],  
 ),  
 const SizedBox(height: 8.0),  
 Row(  
 mainAxisAlignment: MainAxisAlignment.spaceAround,  
 children: [  
 Text(  
 "Market Cap: ",  
 *// style: Theme.of(context).textTheme.subtitle1,* style: TextStyle(  
 color: Color.fromRGBO(11, 12, 54, 1)),  
 ),  
 Text(  
 coin.quoteModel.usdModel.marketCap  
 .toStringAsFixed(2),  
 *// style: Theme.of(context).textTheme.headline6,* style: TextStyle(  
 color: Color.fromRGBO(11, 12, 54, 1)),  
 ),  
 ],  
 ),  
 const SizedBox(height: 8.0),

**First Screen:**

import 'package:flutter/material.dart';  
  
import 'package:crypto\_application/models/fetchCoins\_models/big\_data\_model.dart';  
import 'package:crypto\_application/repository/repository.dart';  
import 'package:crypto\_application/widgets/widgets.dart';  
  
class FirstScreen extends StatefulWidget {  
 const FirstScreen({  
 Key? key,  
 }) : super(key: key);  
  
 @override  
 \_FirstScreenState createState() => \_FirstScreenState();  
}  
  
class \_FirstScreenState extends State<FirstScreen> {  
 late Future<BigDataModel> \_futureCoins;  
 late Repository repository;  
 @override  
 void initState() {  
 repository = Repository();  
 \_futureCoins = repository.getCoins();  
 super.initState();  
 }  
  
 @override  
 Widget build(BuildContext context) {  
 return FutureBuilder<BigDataModel>(  
 future: \_futureCoins,  
 builder: (context, snapshot) {  
 if (snapshot.connectionState == ConnectionState.done) {  
 if (snapshot.hasData) {  
 var coinsData = snapshot.data!.dataModel;  
 return CoinListWidget(coins: coinsData);  
 } else if (snapshot.hasError) {  
 return Text('${snapshot.error}');  
 }  
 }  
  
 return const Center(  
 child: CircularProgressIndicator(),

**Second Screen:**

import 'package:flutter/material.dart';  
  
import '../models/fetchCoins\_models/big\_data\_model.dart';  
import '../repository/repository.dart';  
import '../widgets/coin\_list\_widget.dart';  
  
class SecondScreen extends StatefulWidget {  
 const SecondScreen({Key? key}) : super(key: key);  
  
 @override  
 State<SecondScreen> createState() => \_SecondScreenState();  
}  
  
class \_SecondScreenState extends State<SecondScreen> {  
 late Future<BigDataModel> \_futureCoins;  
 late Repository repository;  
 @override  
 void initState() {  
 repository = Repository();  
 \_futureCoins = repository.getCoins();  
 super.initState();  
 }  
  
 @override  
 Widget build(BuildContext context) {  
 return FutureBuilder<BigDataModel>(  
 future: \_futureCoins,  
 builder: (context, snapshot) {  
 if (snapshot.connectionState == ConnectionState.done) {  
 if (snapshot.hasData) {  
 var coinsData = snapshot.data!.dataModel;  
 return CoinListWidgetForBuy(coins: coinsData);  
 } else if (snapshot.hasError) {  
 return Text('${snapshot.error}');  
 }  
 }  
  
 return const Center(  
 child: CircularProgressIndicator(),

**Top up:**

import 'package:crypto\_application/data\_base.dart';  
import 'package:crypto\_application/models/credit\_card.dart';  
import 'package:crypto\_application/widgets/loader2.dart';  
import 'package:flutter/material.dart';  
import 'package:flutter\_credit\_card/credit\_card\_form.dart';  
import 'package:flutter\_credit\_card/credit\_card\_model.dart';  
import 'package:flutter\_credit\_card/credit\_card\_widget.dart';  
import 'package:fluttertoast/fluttertoast.dart';  
import 'package:provider/provider.dart';  
  
class TopUpWallet extends StatefulWidget {  
 final String uid;  
 TopUpWallet({Key? key, required this.uid}) : super(key: key);  
 static String *id* = 'top\_up';  
  
 @override  
 State<TopUpWallet> createState() => \_TopUpWalletState();  
}  
  
class \_TopUpWalletState extends State<TopUpWallet> {  
 @override  
 Widget build(BuildContext context) {  
 return Scaffold(  
 appBar: AppBar(  
 backgroundColor: Color.fromRGBO(11, 12, 54, 1),  
 title: Text('Top Up'),  
 actions: [  
 IconButton(  
 onPressed: () {  
 Navigator.*push*(  
 context,  
 MaterialPageRoute(  
 builder: (context) => AddNewCard(uid: widget.uid)));  
 },  
 icon: Icon(Icons.*add*)),  
 ],  
 ),  
 body: StreamProvider<List<CreditCard>>.value(  
 value: Database().streamCreditCards(widget.uid),  
 initialData: [],  
 child: SelectCreditCard(  
 uid: widget.uid,  
 ),  
 ),  
 );  
 }  
}  
  
class SelectCreditCard extends StatefulWidget {  
 final String uid;  
 const SelectCreditCard({Key? key, required this.uid}) : super(key: key);  
  
 @override  
 State<SelectCreditCard> createState() => \_SelectCreditCardState();  
}  
  
class \_SelectCreditCardState extends State<SelectCreditCard> {  
 void \_topUp(String uid, String cardHolder, String cardNum) {  
 showModalBottomSheet(  
 context: context,  
 builder: (context) {  
 return Container(  
 *// color: Color.fromRGBO(11, 12, 54, 1),* padding: EdgeInsets.symmetric(horizontal: 20, vertical: 20),  
 child: AddBalanceToUrWallet(  
 cardHolder: cardHolder, cardNum: cardNum, uid: uid),  
 );  
 });  
 }  
  
 @override  
 Widget build(BuildContext context) {  
 final creditCardinf = Provider.*of*<List<CreditCard>>(context);  
 return ListView.builder(  
 itemCount: creditCardinf.length,  
 itemBuilder: ((context, i) {  
 if (creditCardinf.isEmpty) {  
 return Center(  
 child: CircularProgressIndicator(),  
 );  
 } else {  
 return Card(  
 child: Column(  
 children: [  
 CreditCardWidget(  
 isHolderNameVisible: true,  
 cardNumber: creditCardinf[i].cardNumber,  
 expiryDate: creditCardinf[i].expirydate,  
 cardHolderName: creditCardinf[i].cardHolder,  
 cvvCode: creditCardinf[i].cvv,  
 showBackView: false,  
 onCreditCardWidgetChange: (creditCardBrand) {},  
 ),  
 *// SizedBox(  
 // height: 5,  
 // ),* Row(  
 mainAxisAlignment: MainAxisAlignment.spaceBetween,  
 children: [  
 ElevatedButton(  
 onPressed: () async {  
 try {  
 await Database().removeCreditCard(  
 widget.uid, creditCardinf[i].id);  
 Fluttertoast.*showToast*(  
 msg: 'Card Has Been Removed!');  
 } catch (e) {  
 Fluttertoast.*showToast*(msg: e.toString());  
 }  
 },  
 child: Text('Remove')),  
 ElevatedButton(  
 onPressed: () {  
 \_topUp(widget.uid, creditCardinf[i].cardHolder,  
 creditCardinf[i].cardNumber);  
 },  
 child: Text('Use This Card')),  
 ],  
 )  
 ],  
 ),  
 );  
 }  
 }));  
 }  
}  
  
class AddBalanceToUrWallet extends StatefulWidget {  
 final String uid, cardHolder, cardNum;  
 AddBalanceToUrWallet(  
 {Key? key,  
 required this.cardHolder,  
 required this.cardNum,  
 required this.uid})  
 : super(key: key);  
  
 @override  
 State<AddBalanceToUrWallet> createState() => \_AddBalanceToUrWalletState();  
}  
  
class \_AddBalanceToUrWalletState extends State<AddBalanceToUrWallet> {  
 double amount = 0;  
 bool loader = false;  
 @override  
 Widget build(BuildContext context) {  
 return Container(  
 child: ListView(children: [  
 Card(  
 shape:  
 RoundedRectangleBorder(borderRadius: BorderRadius.circular(10)),  
 elevation: 5,  
 child: Padding(  
 padding: const EdgeInsets.all(8.0),  
 child: Column(  
 mainAxisSize: MainAxisSize.min,  
 children: [  
 Text(  
 'Card Number: ${widget.cardNum}',  
 style: TextStyle(fontSize: 16),  
 ),  
 SizedBox(  
 height: 5,  
 ),  
 Text(  
 'Card Holder: ${widget.cardHolder}',  
 style: TextStyle(fontSize: 16),  
 ),  
 ],  
 ),  
 ),  
 ),  
 SizedBox(  
 height: 20,  
 ),  
 Padding(  
 padding: const EdgeInsets.symmetric(horizontal: 30.0),  
 child: TextFormField(  
 style: TextStyle(color: Color.fromRGBO(11, 12, 54, 1)),  
 keyboardType: TextInputType.*number*,  
 cursorColor: Colors.*grey*,  
 maxLength: 9,  
 onChanged: (val) {  
 setState(() {  
 amount = double.*parse*(val);  
 });  
 },  
 validator: (value) {  
 if (value == null || value.isEmpty) {  
 return 'Please enter a Value';  
 }  
 return null;  
 },  
 decoration: InputDecoration(  
 border: OutlineInputBorder(  
 borderSide: BorderSide(color: Colors.*red*),  
 borderRadius: BorderRadius.circular(10.0),  
 ),  
 hintText: 'Amount in \$',  
 filled: true,  
 fillColor: Colors.*white*,  
 contentPadding:  
 const EdgeInsets.only(left: 14.0, bottom: 6.0, top: 8.0),  
 disabledBorder: OutlineInputBorder(  
 borderSide: BorderSide(color: Colors.*red*),  
 borderRadius: BorderRadius.circular(10.0),  
 ),  
 focusedBorder: OutlineInputBorder(  
 borderSide: BorderSide(color: Colors.*red*),  
 borderRadius: BorderRadius.circular(10.0),  
 ),  
 enabledBorder: UnderlineInputBorder(  
 borderSide: BorderSide(color: Colors.*grey*),  
 borderRadius: BorderRadius.circular(10.0),  
 )),  
 ),  
 ),  
 SizedBox(  
 height: 30,  
 ),  
 loader  
 ? Loader2()  
 : Padding(  
 padding: const EdgeInsets.symmetric(vertical: 16.0),  
 child: Material(  
 color: Color.fromRGBO(11, 12, 54, 1),  
 elevation: 5.0,  
 borderRadius: BorderRadius.circular(30.0),  
 child: MaterialButton(  
 onPressed: () async {  
 setState(() {  
 loader = true;  
 });  
 try {  
 await Database().topUpWallet(widget.uid, amount);  
 Fluttertoast.*showToast*(  
 msg: '$amount \$ has been added to your account');  
 setState(() {  
 loader = true;  
 });  
 Navigator.*pop*(context);  
 } catch (e) {  
 setState(() {  
 loader = true;  
 });  
 Fluttertoast.*showToast*(msg: e.toString());  
 }  
 },  
 minWidth: MediaQuery.*of*(context).size.width / 1.5,  
 height: 42.0,  
 child: const Text(  
 'Top Up',  
 style: TextStyle(  
 color: Colors.*white*, fontWeight: FontWeight.*bold*),  
 *////////* ),  
 ),  
 ),  
 ),  
 ]),  
 );  
 }  
}  
  
class AddNewCard extends StatefulWidget {  
 final String uid;  
 AddNewCard({Key? key, required this.uid}) : super(key: key);  
  
 @override  
 State<AddNewCard> createState() => \_AddNewCardState();  
}  
  
class \_AddNewCardState extends State<AddNewCard> {  
 String cardNumber = '', expiryDate = '', cardHolderName = '', cvvCode = '';  
 bool isCvvFocused = false;  
 final \_key = GlobalKey<FormState>();  
  
 @override  
 Widget build(BuildContext context) {  
 return Scaffold(  
 appBar: AppBar(  
 backgroundColor: Color.fromRGBO(11, 12, 54, 1),  
 ),  
 body: ListView(  
 children: [  
 CreditCardWidget(  
 isHolderNameVisible: true,  
 cardNumber: cardNumber,  
 expiryDate: expiryDate,  
 cardHolderName: cardHolderName,  
 cvvCode: cvvCode,  
 showBackView: isCvvFocused,  
 onCreditCardWidgetChange: (creditCardBrand) {},  
 ),  
 SizedBox(  
 height: 10,  
 ),  
 CreditCardForm(  
 formKey: \_key, *// Required* onCreditCardModelChange: onCreditCardModelChange, *// Required* themeColor: Colors.*red*,  
 obscureCvv: true,  
 obscureNumber: false,  
 isHolderNameVisible: true,  
 isCardNumberVisible: true,  
 isExpiryDateVisible: true,  
 cardNumberDecoration: const InputDecoration(  
 border: OutlineInputBorder(),  
 labelText: 'Number',  
 hintText: 'XXXX XXXX XXXX XXXX',  
 ),  
 expiryDateDecoration: const InputDecoration(  
 border: OutlineInputBorder(),  
 labelText: 'Expired Date',  
 hintText: 'XX/XX',  
 ),  
 cvvCodeDecoration: const InputDecoration(  
 border: OutlineInputBorder(),  
 labelText: 'CVV',  
 hintText: 'XXX',  
 ),  
 cardHolderDecoration: const InputDecoration(  
 border: OutlineInputBorder(),  
 labelText: 'Card Holder',  
 ),  
 cardHolderName: cardHolderName,  
 cardNumber: cardNumber,  
 cvvCode: cvvCode,  
 expiryDate: expiryDate),  
 ],  
 ),  
 bottomNavigationBar: BottomAppBar(  
 color: Color.fromRGBO(11, 12, 54, 1),  
 child: TextButton(  
 child: Text(  
 'Add Card',  
 style: TextStyle(color: Colors.*white*, fontSize: 16),  
 ),  
 onPressed: () async {  
 try {  
 await Database().saveCreditCard(  
 widget.uid, cardNumber, expiryDate, cvvCode, cardHolderName);  
 Fluttertoast.*showToast*(msg: 'Your Card has been Save');  
 Navigator.*pop*(context);  
 } catch (e) {  
 Fluttertoast.*showToast*(msg: e.toString());  
 }  
 },  
 ),  
 ),  
 );  
 }  
  
 void onCreditCardModelChange(CreditCardModel? creditCardModel) {  
 setState(() {  
 cardNumber = creditCardModel!.cardNumber;  
 expiryDate = creditCardModel.expiryDate;  
 cardHolderName = creditCardModel.cardHolderName;  
 cvvCode = creditCardModel.cvvCode;  
 isCvvFocused = creditCardModel.isCvvFocused;

});  
 }  
}

**Trade Screen:**

import 'package:crypto\_application/data\_base.dart';  
import 'package:crypto\_application/widgets/ymd.dart';  
import 'package:firebase\_auth/firebase\_auth.dart';  
import 'package:flutter/material.dart';  
import 'package:fluttertoast/fluttertoast.dart';  
import 'package:provider/provider.dart';  
  
import '../models/chart\_data\_model.dart';  
import '../models/fetchCoins\_models/big\_data\_model.dart';  
import '../models/fetchCoins\_models/data\_model.dart';  
import '../models/trade.dart';  
import '../repository/repository.dart';  
import '../streams/stream\_trades.dart';  
import '../widgets/coin\_chart\_widget.dart';  
import '../widgets/coin\_logo\_widget.dart';  
import '../widgets/loader2.dart';  
  
class TradeScreen extends StatefulWidget {  
 const TradeScreen({Key? key}) : super(key: key);  
  
 @override  
 State<TradeScreen> createState() => \_TradeScreenState();  
}  
  
class \_TradeScreenState extends State<TradeScreen> {  
 @override  
 Widget build(BuildContext context) {  
 return Scaffold(  
 body: Padding(  
 padding: const EdgeInsets.all(8.0),  
 child: Column(mainAxisAlignment: MainAxisAlignment.center, children: [  
 Center(  
 child: Container(  
 height: 50,  
 width: 200,  
 child: ElevatedButton(  
 style: ElevatedButton.*styleFrom*(  
 primary: Color.fromRGBO(11, 12, 54, 1),  
 elevation: 5,  
 ),  
 onPressed: () {  
 Navigator.*pushNamed*(context, TradeSc.*id*);  
 },  
 child: Text(  
 'Trade',  
 style: TextStyle(fontSize: 18, color: Colors.*white*),  
 ),  
 ),  
 ),  
 ),  
 SizedBox(  
 height: 40,  
 ),  
 Center(  
 child: Container(  
 height: 50,  
 width: 200,  
 child: ElevatedButton(  
 style: ElevatedButton.*styleFrom*(  
 primary: Color.fromRGBO(11, 12, 54, 1),  
 elevation: 5,  
 ),  
 onPressed: () {  
 Navigator.*push*(  
 context,  
 MaterialPageRoute(  
 builder: ((context) => SwitchTrade())));  
 },  
 child: Text(  
 'Trade Offers',  
 style: TextStyle(fontSize: 18, color: Colors.*white*),  
 ),  
 ),  
 ),  
 ),  
 ])),  
 );  
 }  
}  
  
class TradeSc extends StatefulWidget {  
 static String *id* = 'tradeSc';  
  
 TradeSc({Key? key}) : super(key: key);  
  
 @override  
 State<TradeSc> createState() => \_TradeScState();  
}  
  
class \_TradeScState extends State<TradeSc> {  
 String userEmail = '';  
 String selectedCoin1 = 'Select Coin';  
 String selectedCoin2 = 'Select Coin';  
 String coinSym = '';  
 String coinSym2 = '';  
  
 double firstCoin = 0, seconedCoin = 0;  
  
 void \_showSelect1() {  
 showModalBottomSheet(  
 context: context,  
 builder: (context) {  
 return Container(  
 *//color: Color.fromRGBO(11, 12, 54, 1),* padding: EdgeInsets.symmetric(horizontal: 20, vertical: 20),  
 child: SelectCoin1(  
 selectFirstCoin: selectFirstCoin,  
 ),  
 );  
 });  
 }  
  
 void \_showSelect2() {  
 showModalBottomSheet(  
 context: context,  
 builder: (context) {  
 return Container(  
 *//color: Color.fromRGBO(11, 12, 54, 1),* padding: EdgeInsets.symmetric(horizontal: 20, vertical: 20),  
 child: SelectCoin2(  
 selectSeconedCoin: selectSeconedCoin,  
 ),  
 );  
 });  
 }  
  
 selectFirstCoin(String coinName, String coinSymbol) {  
 setState(() {  
 selectedCoin1 = coinName;  
 coinSym = coinSymbol;  
 });  
 }  
  
 selectSeconedCoin(String coinName, String coinSymbol2) {  
 setState(() {  
 selectedCoin2 = coinName;  
 coinSym2 = coinSymbol2;  
 });  
 }  
  
 final FirebaseAuth auth = FirebaseAuth.*instance*;  
 String uid = '';  
 @override  
 void initState() {  
 final User? user = auth.currentUser;  
 uid = user!.uid;  
 super.initState();  
 }  
  
 @override  
 Widget build(BuildContext context) {  
 return Scaffold(  
 appBar: AppBar(  
 backgroundColor: Color.fromRGBO(11, 12, 54, 1), title: Text('Trade')),  
 body: Padding(  
 padding: const EdgeInsets.all(8.0),  
 child: ListView(  
 children: [  
 Padding(  
 padding: const EdgeInsets.symmetric(horizontal: 30.0),  
 child: TextFormField(  
 style: TextStyle(color: Color.fromRGBO(11, 12, 54, 1)),  
 keyboardType: TextInputType.*emailAddress*,  
 cursorColor: Colors.*grey*,  
 onChanged: (val) {  
 userEmail = val;  
 },  
 validator: (value) {  
 if (value == null ||  
 value.isEmpty ||  
 !value.toString().contains("@")) {  
 return 'Please enter a Vaild Email';  
 }  
 return null;  
 },  
 decoration: InputDecoration(  
 border: InputBorder.*none*,  
 hintText: 'User Email you wish to trade with',  
 filled: true,  
 fillColor: Colors.*white*,  
 contentPadding: const EdgeInsets.only(  
 left: 14.0, bottom: 6.0, top: 8.0),  
 focusedBorder: OutlineInputBorder(  
 borderSide: BorderSide(color: Colors.*red*),  
 borderRadius: BorderRadius.circular(10.0),  
 ),  
 enabledBorder: UnderlineInputBorder(  
 borderSide: BorderSide(color: Colors.*grey*),  
 borderRadius: BorderRadius.circular(10.0),  
 )),  
 ),  
 ),  
 ElevatedButton(  
 onPressed: () {  
 \_showSelect1();  
 },  
 child: Text('Trading: $selectedCoin1')),  
 SizedBox(  
 height: 10,  
 ),  
 Padding(  
 padding: const EdgeInsets.symmetric(horizontal: 30.0),  
 child: TextFormField(  
 style: TextStyle(color: Color.fromRGBO(11, 12, 54, 1)),  
 keyboardType: TextInputType.*number*,  
 cursorColor: Colors.*grey*,  
 onChanged: (val) {  
 firstCoin = double.*parse*(val);  
 },  
 validator: (value) {  
 if (value == null || value.isEmpty) {  
 return 'Please enter a number';  
 }  
 return null;  
 },  
 decoration: InputDecoration(  
 border: InputBorder.*none*,  
 hintText: 'How Many Coins You Want to Trade ?',  
 filled: true,  
 fillColor: Colors.*white*,  
 contentPadding: const EdgeInsets.only(  
 left: 14.0, bottom: 6.0, top: 8.0),  
 focusedBorder: OutlineInputBorder(  
 borderSide: BorderSide(color: Colors.*red*),  
 borderRadius: BorderRadius.circular(10.0),  
 ),  
 enabledBorder: UnderlineInputBorder(  
 borderSide: BorderSide(color: Colors.*grey*),  
 borderRadius: BorderRadius.circular(10.0),  
 )),  
 ),  
 ),  
 ElevatedButton(  
 onPressed: () {  
 \_showSelect2();  
 },  
 child: Text('With: $selectedCoin2')),  
 SizedBox(  
 height: 10,  
 ),  
 Padding(  
 padding: const EdgeInsets.symmetric(horizontal: 30.0),  
 child: TextFormField(  
 style: TextStyle(color: Color.fromRGBO(11, 12, 54, 1)),  
 keyboardType: TextInputType.*number*,  
 cursorColor: Colors.*grey*,  
 onChanged: (val) {  
 seconedCoin = double.*parse*(val);  
 },  
 validator: (value) {  
 if (value == null || value.isEmpty) {  
 return 'Please enter a number';  
 }  
 return null;  
 },  
 decoration: InputDecoration(  
 border: InputBorder.*none*,  
 hintText: 'How Many Coins You Want to Get ?',  
 filled: true,  
 fillColor: Colors.*white*,  
 contentPadding: const EdgeInsets.only(  
 left: 14.0, bottom: 6.0, top: 8.0),  
 focusedBorder: OutlineInputBorder(  
 borderSide: BorderSide(color: Colors.*red*),  
 borderRadius: BorderRadius.circular(10.0),  
 ),  
 enabledBorder: UnderlineInputBorder(  
 borderSide: BorderSide(color: Colors.*grey*),  
 borderRadius: BorderRadius.circular(10.0),  
 )),  
 ),  
 ),  
 ],  
 ),  
 ),  
 bottomNavigationBar: BottomAppBar(  
 child: ElevatedButton(  
 onPressed: () async {  
 try {  
 await Database().sendTradeOffer(  
 selectedCoin1,  
 selectedCoin2,  
 firstCoin,  
 seconedCoin,  
 uid,  
 userEmail,  
 coinSym,  
 coinSym2);  
 Navigator.*pop*(context);  
 } catch (e) {  
 Fluttertoast.*showToast*(msg: e.toString());  
 }  
 },  
 child: Text(  
 'Send Trade Offer',  
 style: TextStyle(fontSize: 16),  
 ))),  
 );  
 }  
}  
  
class SelectCoin1 extends StatefulWidget {  
 final selectFirstCoin;  
 const SelectCoin1({Key? key, required this.selectFirstCoin})  
 : super(key: key);  
  
 @override  
 State<SelectCoin1> createState() => \_SelectCoin1State();  
}  
  
class \_SelectCoin1State extends State<SelectCoin1> {  
 late Future<BigDataModel> \_futureCoins;  
 late Repository repository;  
 @override  
 void initState() {  
 repository = Repository();  
 \_futureCoins = repository.getCoins();  
 super.initState();  
 }  
  
 @override  
 Widget build(BuildContext context) {  
 return FutureBuilder<BigDataModel>(  
 future: \_futureCoins,  
 builder: (context, snapshot) {  
 if (snapshot.connectionState == ConnectionState.done) {  
 if (snapshot.hasData) {  
 var coinsData = snapshot.data!.dataModel;  
 return CoinListForTrade(  
 coins: coinsData,  
 selectCoin: widget.selectFirstCoin,  
 );  
 } else if (snapshot.hasError) {  
 return Text('${snapshot.error}');  
 }  
 }  
  
 return const Center(  
 child: CircularProgressIndicator(),  
 );  
 },  
 );  
 }  
}  
  
class SelectCoin2 extends StatefulWidget {  
 final selectSeconedCoin;  
 const SelectCoin2({Key? key, required this.selectSeconedCoin})  
 : super(key: key);  
  
 @override  
 State<SelectCoin2> createState() => \_SelectCoin2State();  
}  
  
class \_SelectCoin2State extends State<SelectCoin2> {  
 late Future<BigDataModel> \_futureCoins;  
 late Repository repository;  
 @override  
 void initState() {  
 repository = Repository();  
 \_futureCoins = repository.getCoins();  
 super.initState();  
 }  
  
 @override  
 Widget build(BuildContext context) {  
 return FutureBuilder<BigDataModel>(  
 future: \_futureCoins,  
 builder: (context, snapshot) {  
 if (snapshot.connectionState == ConnectionState.done) {  
 if (snapshot.hasData) {  
 var coinsData = snapshot.data!.dataModel;  
 return CoinListForTrade(  
 coins: coinsData,  
 selectCoin: widget.selectSeconedCoin,  
 );  
 } else if (snapshot.hasError) {  
 return Text('${snapshot.error}');  
 }  
 }  
  
 return const Center(  
 child: CircularProgressIndicator(),  
 );  
 },  
 );  
 }  
}  
  
*//select coin*class CoinListForTrade extends StatefulWidget {  
 final List<DataModel> coins;  
 final selectCoin;  
 const CoinListForTrade(  
 {Key? key, required this.coins, required this.selectCoin})  
 : super(key: key);  
  
 @override  
 State<CoinListForTrade> createState() => \_CoinListForTradeState();  
}  
  
class \_CoinListForTradeState extends State<CoinListForTrade> {  
 var coinIconUrl =  
 "https://raw.githubusercontent.com/spothq/cryptocurrency-icons/master/128/color/";  
 @override  
 Widget build(BuildContext context) {  
 return Column(  
 crossAxisAlignment: CrossAxisAlignment.start,  
 children: [  
 YMDcoinTrade(selectCoin: widget.selectCoin),  
 Expanded(  
 child: ListView.builder(  
 itemExtent: 160,  
 itemCount: widget.coins.length,  
 itemBuilder: (context, index) {  
 var coin = widget.coins[index];  
 var coinPrice = coin.quoteModel.usdModel;  
 var data = [  
 ChartData(coinPrice.percentChange\_90d, 2160),  
 ChartData(coinPrice.percentChange\_60d, 1440),  
 ChartData(coinPrice.percentChange\_30d, 720),  
 ChartData(coinPrice.percentChange\_24h, 24),  
 ChartData(coinPrice.percentChange\_1h, 1),  
 ];  
 return GestureDetector(  
 onTap: () {  
 widget.selectCoin(widget.coins[index].name,  
 '${(coinIconUrl + widget.coins[index].symbol).toLowerCase()}.png');  
 Navigator.*pop*(context);  
 },  
 child: Container(  
 height: 160.0,  
 width: double.*infinity*,  
 padding: const EdgeInsets.symmetric(vertical: 4.0),  
 margin: const EdgeInsets.symmetric(  
 vertical: 8.0, horizontal: 16.0),  
 decoration: BoxDecoration(  
 color: Color.fromRGBO(0, 0, 0, 0.6),  
 borderRadius: BorderRadius.circular(16.0),  
 ),  
 child: Row(  
 crossAxisAlignment: CrossAxisAlignment.center,  
 mainAxisAlignment: MainAxisAlignment.spaceBetween,  
 children: [  
 CoinLogoWidget(coin: coin),  
 CoinChartWidget(  
 data: data,  
 coinPrice: coinPrice,  
 color: Colors.*grey*,  
 ),  
 ],  
 ),  
 ),  
 );  
 },  
 ),  
 ),  
 ],  
 );  
 }  
}  
  
class SwitchTrade extends StatefulWidget {  
 const SwitchTrade({Key? key}) : super(key: key);  
  
 @override  
 State<SwitchTrade> createState() => \_SwitchTradeState();  
}  
  
class \_SwitchTradeState extends State<SwitchTrade>  
 with TickerProviderStateMixin {  
 late TabController \_tabController;  
  
 @override  
 void initState() {  
 super.initState();  
 \_tabController = TabController(length: 2, vsync: this);  
 \_tabController.animateTo(0);  
 }  
  
 @override  
 Widget build(BuildContext context) {  
 return Scaffold(  
 appBar: AppBar(  
 backgroundColor: Color.fromRGBO(11, 12, 54, 1),  
 title: TabBar(  
 controller: \_tabController,  
 tabs: [  
 Text(  
 'Sent',  
 style: TextStyle(fontSize: 18),  
 ),  
 Text(  
 'Recieved',  
 style: TextStyle(fontSize: 18),  
 ),  
 ],  
 ),  
 ),  
 body: TabBarView(controller: \_tabController, children: [  
 StreamTrades(),  
 StreamRecieved(),  
 ]),  
 );  
 }  
}  
  
class TradeOffers extends StatefulWidget {  
 TradeOffers({Key? key}) : super(key: key);  
  
 @override  
 State<TradeOffers> createState() => \_TradeOffersState();  
}  
  
class \_TradeOffersState extends State<TradeOffers> {  
 @override  
 Widget build(BuildContext context) {  
 final trade = Provider.*of*<List<Trade>>(context);  
  
 return Scaffold(  
 body: ListView.builder(  
 itemCount: trade.length,  
 itemBuilder: (context, i) {  
 DateTime dateTime = trade[i].timestamp.toDate();  
  
 return Card(  
 elevation: 5,  
 child: Padding(  
 padding: const EdgeInsets.all(8.0),  
 child:  
 Column(mainAxisAlignment: MainAxisAlignment.start, children: [  
 Align(  
 alignment: Alignment.*centerLeft*,  
 child: Text(  
 'Sent To: ${trade[i].recieverEmail}',  
 style: TextStyle(fontSize: 16),  
 ),  
 ),  
 SizedBox(  
 height: 10,  
 ),  
 Row(  
 mainAxisAlignment: MainAxisAlignment.spaceEvenly,  
 children: [  
 Row(  
 children: [  
 Image.network(  
 trade[i].symbol1,  
 height: 30,  
 width: 30,  
 ),  
 Text(  
 '${trade[i].coinName1} ',  
 style: TextStyle(fontSize: 16),  
 ),  
 Text(  
 trade[i].coinCount1.toString() + ' Coins',  
 style: TextStyle(fontSize: 16),  
 ),  
 ],  
 ),  
 Icon(  
 Icons.*arrow\_circle\_right\_rounded*,  
 size: 35,  
 color: Color.fromRGBO(11, 12, 54, 1),  
 ),  
 Row(  
 children: [  
 Image.network(  
 trade[i].symbol2,  
 height: 30,  
 width: 30,  
 ),  
 Text(  
 '${trade[i].coinName2} ',  
 style: TextStyle(fontSize: 16),  
 ),  
 Text(  
 trade[i].coinCount2.toString() + ' Coins',  
 style: TextStyle(fontSize: 16),  
 ),  
 ],  
 ),  
 ],  
 ),  
 SizedBox(  
 height: 10,  
 ),  
 Align(  
 alignment: Alignment.*centerLeft*,  
 child: Text(  
 'Date: ${dateTime.day}/${dateTime.month}/${dateTime.year}',  
 style: TextStyle(fontSize: 16),  
 ),  
 ),  
 SizedBox(  
 height: 10,  
 ),  
 Align(  
 alignment: Alignment.*centerLeft*,  
 child: Text(  
 'Status: ${trade[i].status}',  
 style: TextStyle(fontSize: 16),  
 ),  
 ),  
 ]),  
 ),  
 );  
 },  
 ),  
 );  
 }  
}  
  
class TradeOffersRec extends StatefulWidget {  
 TradeOffersRec({Key? key}) : super(key: key);  
  
 @override  
 State<TradeOffersRec> createState() => \_TradeOffersRec();  
}  
  
class \_TradeOffersRec extends State<TradeOffersRec> {  
 bool checkAccept = true;  
 checkAccepted(bool check) {  
 if (check == true) {  
 checkAccept = false;  
 } else {  
 checkAccept = true;  
 }  
 }  
  
 @override  
 Widget build(BuildContext context) {  
 final trade = Provider.*of*<List<Trade>>(context);  
 bool loader = false;  
 return Scaffold(  
 body: ListView.builder(  
 itemCount: trade.length,  
 itemBuilder: (context, i) {  
 DateTime dateTime = trade[i].timestamp.toDate();  
 checkAccepted(trade[i].accepted);  
 return Card(  
 elevation: 5,  
 child: Padding(  
 padding: const EdgeInsets.all(8.0),  
 child:  
 Column(mainAxisAlignment: MainAxisAlignment.start, children: [  
 Align(  
 alignment: Alignment.*centerLeft*,  
 child: Text(  
 'Recieved From: ${trade[i].senderEmail}',  
 style: TextStyle(fontSize: 16),  
 ),  
 ),  
 SizedBox(  
 height: 10,  
 ),  
 Row(  
 mainAxisAlignment: MainAxisAlignment.spaceEvenly,  
 children: [  
 Row(  
 children: [  
 Image.network(  
 trade[i].symbol1,  
 height: 30,  
 width: 30,  
 ),  
 Text(  
 '${trade[i].coinName1} ',  
 style: TextStyle(fontSize: 16),  
 ),  
 Text(  
 trade[i].coinCount1.toString() + ' Coins',  
 style: TextStyle(fontSize: 16),  
 ),  
 ],  
 ),  
 Icon(  
 Icons.*arrow\_circle\_right\_rounded*,  
 size: 35,  
 color: Color.fromRGBO(11, 12, 54, 1),  
 ),  
 Row(  
 children: [  
 Image.network(  
 trade[i].symbol2,  
 height: 30,  
 width: 30,  
 ),  
 Text(  
 '${trade[i].coinName2} ',  
 style: TextStyle(fontSize: 16),  
 ),  
 Text(  
 trade[i].coinCount2.toString() + ' Coins',  
 style: TextStyle(fontSize: 16),  
 ),  
 ],  
 ),  
 ],  
 ),  
 SizedBox(  
 height: 10,  
 ),  
 Align(  
 alignment: Alignment.*centerLeft*,  
 child: Text(  
 'Date: ${dateTime.day}/${dateTime.month}/${dateTime.year}',  
 style: TextStyle(fontSize: 16),  
 ),  
 ),  
 SizedBox(  
 height: 10,  
 ),  
 Align(  
 alignment: Alignment.*centerLeft*,  
 child: Text(  
 'Status: ${trade[i].status}',  
 style: TextStyle(fontSize: 16),  
 ),  
 ),  
 SizedBox(  
 height: 10,  
 ),  
 loader  
 ? Loader2()  
 : Visibility(  
 visible: checkAccept,  
 child: Row(  
 mainAxisAlignment: MainAxisAlignment.spaceBetween,  
 children: [  
 Padding(  
 padding:  
 const EdgeInsets.symmetric(vertical: 16.0),  
 child: Material(  
 color: Color.fromRGBO(11, 12, 54, 1),  
 elevation: 5.0,  
 borderRadius: BorderRadius.circular(30.0),  
 child: MaterialButton(  
 onPressed: () async {  
 setState(() {  
 loader = true;  
 });  
 try {  
 await Database().tradeAccepted(  
 trade[i].sender,  
 trade[i].reciever,  
 trade[i].tradeId,  
 trade[i].coinName1,  
 trade[i].coinName2,  
 trade[i].coinCount1,  
 trade[i].coinCount2,  
 trade[i].symbol1,  
 trade[i].symbol2);  
 setState(() {  
 loader = false;  
 });  
 } catch (e) {  
 setState(() {  
 loader = false;  
 });  
 Fluttertoast.*showToast*(msg: e.toString());  
 }  
 },  
 child: const Text(  
 'Accept',  
 style: TextStyle(  
 color: Colors.*white*,  
 fontWeight: FontWeight.*bold*),  
 ),  
 ),  
 ),  
 ),  
 Padding(  
 padding:  
 const EdgeInsets.symmetric(vertical: 16.0),  
 child: Material(  
 color: Color.fromRGBO(11, 12, 54, 1),  
 elevation: 5.0,  
 borderRadius: BorderRadius.circular(30.0),  
 child: MaterialButton(  
 onPressed: () async {  
 setState(() {  
 loader = true;  
 });  
 await Database().declineTrade(  
 trade[i].sender,  
 trade[i].reciever,  
 trade[i].tradeId);  
 setState(() {  
 loader = false;  
 });  
 },  
 child: const Text(  
 'Decline',  
 style: TextStyle(  
 color: Colors.*white*,  
 fontWeight: FontWeight.*bold*),

**Wallet:**

mport 'package:crypto\_application/Screens/top\_up\_wallet.dart';  
import 'package:crypto\_application/models/user.dart';  
import 'package:firebase\_auth/firebase\_auth.dart';  
import 'package:flutter/material.dart';  
  
import '../data\_base.dart';  
import 'third\_scree.dart';  
  
class Wallet extends StatefulWidget {  
 Wallet({Key? key}) : super(key: key);  
 static String *id* = 'wallet';  
  
 @override  
 State<Wallet> createState() => \_WalletState();  
}  
  
class \_WalletState extends State<Wallet> {  
 final FirebaseAuth auth = FirebaseAuth.*instance*;  
 String uid = '';  
 @override  
 void initState() {  
 final User? user = auth.currentUser;  
 uid = user!.uid;  
 super.initState();  
 }  
  
 @override  
 Widget build(BuildContext context) {  
 return Scaffold(  
 appBar: AppBar(  
 backgroundColor: Color.fromRGBO(11, 12, 54, 1),  
 title: Text('Wallet'),  
 ),  
 body: ListView(  
 children: [  
 StreamBuilder<UserData>(  
 stream: Database().userData(uid),  
 builder: (context, snapshot) {  
 UserData? userData = snapshot.data;  
 if (snapshot.hasData) {  
 return Card(  
 color: Color.fromRGBO(11, 12, 54, 1),  
 elevation: 5,  
 child: Padding(  
 padding: const EdgeInsets.all(8.0),  
 child: Column(  
 mainAxisSize: MainAxisSize.min,  
 crossAxisAlignment: CrossAxisAlignment.start,  
 mainAxisAlignment: MainAxisAlignment.spaceEvenly,  
 children: [  
 Text(  
 'Wallet Balance',  
 style: TextStyle(  
 fontSize: 21,  
 color: Colors.*white*,  
 fontWeight: FontWeight.*bold*),  
 ),  
 SizedBox(  
 height: 10,  
 ),  
 Center(  
 child: Text(  
 '${userData!.wallet} \$',  
 style: TextStyle(  
 fontSize: 16,  
 color: Colors.*white*,  
 fontWeight: FontWeight.*bold*),  
 ),  
 ),  
 ],  
 ),  
 ),  
 );  
 } else {  
 return CircularProgressIndicator();  
 }  
 }),  
 Divider(  
 color: Color.fromRGBO(11, 12, 54, 1),  
 ),  
 Text(  
 'Owned Crypto Currencies',  
 style:  
 TextStyle(fontSize: 24, color: Color.fromRGBO(11, 12, 54, 1)),  
 ),  
 SizedBox(  
 height: 20,  
 ),  
 Card(  
 color: Color.fromRGBO(11, 12, 54, 1),  
 elevation: 5,  
 child: Padding(  
 padding: const EdgeInsets.all(8.0),  
 child: StreamOwnedCoins(  
 uid: uid,  
 ),  
 ),  
 ),  
 ],  
 ),  
 bottomNavigationBar: BottomAppBar(  
 color: Color.fromRGBO(11, 12, 54, 1),  
 child: TextButton(  
 onPressed: () {  
 Navigator.*push*(  
 context,  
 MaterialPageRoute(  
 builder: (context) => TopUpWallet(uid: uid)));  
 },  
 child: Text('Top up',  
 style: TextStyle(  
 fontSize: 16,  
 color: Colors.*white*,

***Chapter Five***

***Testing***

# TESTING

Software testing is any activity aimed at evaluating an attribute or capability of a program or system and determining that it meets its required results. Although crucial to software quality and widely deployed by programmers and testers, software testing still remains an art, due to limited understanding of the principles of software. The difficulty in software testing stems from the complexity of software: we can not completely test a program with moderate complexity. Testing is more than just debugging. The purpose of testing can be quality assurance, and validation, or reliability estimation. Testing can be used as a generic metric as well. Correctness testing and reliability testing are two major areas of testing. Software testing is a trade-off between budget, time, and quality.

\*Unit Testing1

*Figer13:Testing*



## 5.1 Unit Testing

Unit testing is one of the software testing types which includes the initial testing phase where the smallest components or the modules of a software are tested individually. both testers and developers can isolate each module, identify and fix the system defects at a very early stage of the software development lifecycle (SDLC).

Eg. “Test every single major function such as login, Adding and removing Product to Favourite list, Cancelation a product from the list

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Steps | Variable | Test Case1 | Test Case2 | Expected result | Actual result | Pass/fail |
| 1 | **Login to System** | Enter valid username and password | ---- | entering to the system | ---- |  |
| 2 | **Login to System** | ---- | Enter invalid username and password | Displaying message “Failed to login”  and staying in  login page | ---- |  |

*Table 4:Testing*

### 5.2 Set cases test

You have to show where you used the requirement in your design as well as in your implementation by conducting a test for the full system and showing the input / process / output procedures within design and implementation stages)

* 1. **Integration Testing**

Integration testing is the testing of various modules of the software under development together as a group. This determines whether or not they function together seamlessly as part of the system or whole.

let admin add a new product then go to firebase to see if the addition is accepted "added successfully" and showing a new table for the new product

* 1. **System Testing**

Software testing is a level of testing that validates the complete and fully integrated software product. At this level, we test the system as "one unit" to check if it encompasses all the main Functions.

**Conclusion**

**and Future Work**

**conclusion**

The Coins application is designed to provide a convenient market for currency trading. The search engine provides an easy and convenient way to search for available currencies. also , . The application will display currencies with pictures and prices for each currency. The user can then do the buying or selling process. They can also trade by accepting a request for a trading and buying process, or sending a request to carry out trading and selling. It also provides an electronic wallet, buying and selling through a visa.

We also developed this application using the "Flutter" programming language on Android Studio. FireBase is used to manage the database

**Future Work**

Due to time constraints, As we found During the project implementation process that there is much progress can be made in the future. As we know, It aims to provide the most suitable environment for the market and target groups.

Here are some future supplements ....

1. Make the application more flexible and easier to use
2. Support other languages
3. Login with a Facebook, Google, iCloud account

d) The item will be ranked on the basis of the most traded currency