Ace Trading

Final Year Project

Session 2019-2023

A project submitted in partial fulfillment of the degree of

BS in Information Technology

Department of Information Technology

Faculty of Computer Science & Information Technology

The Superior University, Lahore

Spring 2023

Type (Nature of project)	[✓] **D**evelopment [ ] **R**esearch [ ] **R**&**D**
Area of specialization	
FYP ID	FYP-BITM-F22-009

Project Group Members

Sr.#	Reg. #	Student Name	Email ID	*Signature
(i)	Bitm-f19-040	Mahrunisa Naveed	Bitm-f19-040@superior.edu.pk	
(ii)	Bitm-f19-028	Abdullah	Bitm-f19-028@superior.edu.pk	
(iii)	Bitm-f19-023	Haider Hassan	Bitm-f19-023@superior.edu.pk	

*The candidates confirm that the work submitted is their own and appropriate credit has been given where reference has been made to work of others

Plagiarism Free Certificate

This is to certify that, I Mahrunisa S/D of Naveed Alam, group leader of FYP under registration no ___ FYP-BITM-F22-009_____at information Technology Department, The Superior University, Lahore. I declare that my FYP report is checked by my supervisor.

Date: Name of Group Leader: Mahrunisa Signature: _____

Name of Supervisor: Sir Fiaz Ahmed Manager: Sir Javaid

Designation: Lecturer Designation: Lecturer

Signature: _____ Signature: _____

HoD: Dr. Asad Ali Naqvi

Signature: _____

Ace Trading

Change Record

Author(s)	Version	Date	Notes	Supervisor's Signature
Mahrunisa, Abdullah , Haider	1.0		Template 1	
Mahrunisa, Abdullah , Haider	1.2		Documentation	
Mahrunisa, Abdullah , Haider	1.3		Review of layouts	
Mahrunisa, Abdullah , Haider	1.4		Review of Coding	
Mahrunisa, Abdullah , Haider	1.5		Review of database	
Mahrunisa, Abdullah , Haider	1.0		Template 1	
Mahrunisa, Abdullah , Haider	1.2		Documentation	

APPROVAL

PROJECT SUPERVISOR

Comments: _____

_____

Name:_____

Date:_____ Signature:_____

PROJECT MANAGER

Comments: _____

_____

Date:_____ Signature:_____

HEAD OF THE DEPARTMENT

Comments: _____

_____

Date:_____ Signature:_____

Dedication

This dissertation is dedicated to my family, who passed on a love of reading and respect of education. Without their huge support, it would not have been possible. And also to my lovely friends, your inspiration, encouragement and support in diverse forms would always be remembered and cherished.

Acknowledgements

Here I am gladly present this project report on "**Ace Trading** "as a FINAL project of BSIT. At this time of submitting this report I use this opportunity to mention those people who with me along the work. I take this occasion to thank God, almighty for blessing us with his grace and taking me Endeavour to a successful culmination. I extend my sincere and heartfelt thanks to our esteemed guide **Mr. Fiaz Ahmed** for providing us with the right guidance, advice and for showing us the right way. I also extend my sincere thanks to our respected Head Of The department giving me an opportunity to polish my capabilities. I would like to thank the other faculty members also who lead me to a successful development, at this occasion. Last but not the least; I would like to thank friends for the support and encouragement they have given me during the course.

Executive Summary

Ace Trading is an Online Currency Exchange Platform Developed with Laravel. With our script you can create your own website for exchange of electronic currency (Such As: PayPal, Bitcoin, Skrill, WebMoney, Payeer, Perfect Money ETC). Included wonderful and friendly interface for use by the customer and practical admin panel for use in the administration of the site. Crypto currency, an encrypted, peer-to-peer network for facilitating digital barter, is a technology developed eight years ago. Bitcoin, the first and most popular crypto currency, is paving the way as a disruptive technology to long standing and unchanged financial payment systems that have been in place for many decades. While crypto currencies are not likely to replace traditional fiat currency, they could change the way Internet-connected global markets interact with each other, clearing away barriers surrounding normative national currencies and exchange rates. Technology advances at a rapid rate, and the success of a given technology is almost solely dictated by the market upon which it seeks to improve. Crypto currencies may revolutionize digital trade markets by creating a free flowing trading system without fees.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Chapter 1: Introduction

Investing in the stock market, forex, and cryptocurrency can be a challenging and time-consuming process. However, with the advent of online trading platforms, it has become easier for individuals to manage their investments and monitor their portfolio. One such platform is ACE Trading, a full responsive and dynamic investment platform built with the Laravel In addition, ACE Trading utilizes the power of block chain technology, which is the backbone of cryptocurrency. By using block chain, the platform ensures secure and transparent transactions while providing users with a decentralized and fee-free trading system.

Overall, ACE Trading is a comprehensive solution for anyone looking to create their own online investment system. With its user-friendly interface and powerful features, it makes it easy to get started and manage your investments in a simple and efficient way.

1.1. Background

Bitcoin, the worlds most common and well-known cryptocurrency, has been increasing in popularity. It has the same basic structure as it did when created in 2008, but repeat instances of the world market changing has created a new demand for cryptocurrencies much greater than its initial showing. By using a cryptocurrency, users are able to exchange value digitally without third party oversight. Cryptocurrency works on the theory of solving encryption algorithms to create unique hashes that are finite in number. Combined with a network of computers verifying transactions, users are able to exchange hashes as if exchanging physical currency. There is a finite number of bitcoins that will ever be generated, preventing an overabundance and ensuring its rarity.

As of March 18 2018 there are 1564 Cryptocurrencies available & traded in about 9422 exchanges. The market capitalization of all the cryptocurrencies is $275,797,435,861 i.e. $275 Billions. & 24-hour volume was $ 18,207,953,654 i.e.$18 Billions

1.2. Motivations and Challenges

Focus on offering a wider range of investment options, more advanced risk management tools, or a more user-friendly interface. The platform can fulfill, and how it will provide a better solution for your target audience.

1.3. Goals and Objectives

The main goal and objective is system must be capable to meet the desired objective and goals of the organization. System must be capable to change accordingly to the situation or requirements of the organization. The develop system is very secure and have a great sufficiency effect on the business. The develop system easy to understand and have the full capability to manage the system and it can minimize the different data security threats. There will no stationary expense and work will be done with minimum manpower. Means that it is too much economical.

1.4. Literature Review/Existing Solutions

E-Trade: One of the oldest and most popular online trading platforms, E-Trade offers a wide range of investment options, including stocks, options, ETFs, and mutual funds. It also provides tools for research, charting, and risk management.

TD Ameritrade: Another well-established platform, TD Ameritrade offers similar features to E-Trade but also includes a virtual trading platform that allows users to test out different strategies before investing real money.

Robinhood: A newer platform that has gained popularity due to its commission-free trading and easy-to-use mobile app. It offers options trading and cryptocurrency trading.

Coinbase: A cryptocurrency exchange that also offers trading of bitcoin, ethereum, and other cryptocurrency. It's easy to use and has a clean interface.

Binance: A cryptocurrency exchange that offers a wide range of trading options, including margin trading and staking, and also have a large variety of crypto-asset.

1.5. Gap Analysis

ACE Trading aims to provide a decentralized and fee-free trading system that ensures secure and transparent transactions. The combination of these features and functionalities will provide a unique and compelling solution for individual investors looking to make informed decisions and achieve their investment goals.

Identifying these gaps can provide a significant advantage over the competition, and help to position your platform as a more compelling option for potential users.

1.6. Proposed Solution

The proposed solution is the development of an online investment platform, ACE Trading, that addresses the problems of the existing systems by providing users with a comprehensive, feature-rich, and cost-effective platform.

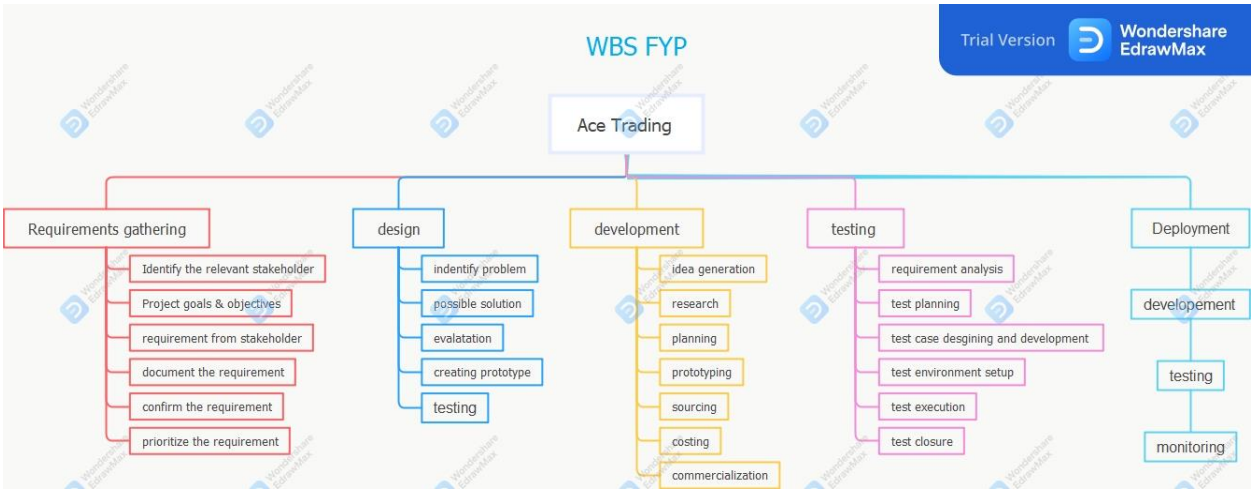
The platform will be built using the Laravel framework to ensure that it is responsive and dynamic. It will also include real-time monitoring of investments, automated trading, and risk m In addition, by utilizing blockchain technology, ACE Trading aims to provide a decentralized and fee-free trading system that ensures secure and transparent transactions. The combination of these features and functionalities will provide a unique and compelling solution for individual investors looking to make informed decisions and achieve their investment goals. Management tools, which will allow individual investors to manage and monitor their portfolio more efficiently.

1.7. Project Plan

A project plan, Work Breakdown Structure (WBS), Roles & Responsibility and Gantt chart are essential tools for managing and organizing a project. They help to define and organize the project scope, identify tasks and dependencies, and track progress.

1.7.1. Work Breakdown Structure

Work Breakdown Structure (WBS): The WBS breaks down the project into smaller, manageable tasks and organizes them into a hierarchical structure. The WBS for the ACE Trading platform development project could include tasks such as requirements gathering, design, development, testing, and deployment.



1.7.2. Roles & Responsibility Matrix

WBS #	WBS Deliverable	Activity #	Activity to Complete the Deliverable	Duration (# of Days)	Responsible Team Member(s) & Role(s)
1.	**Requirements gathering**	1	Submit Proposal	3 day	Mahrunisa
1.1	Define stakeholders	3	Research	1 day	Haider
1.2	Define goals	4	Research and proposal	2 days	Haider
2.	**Planning and design**	5	Surveys and meetings	10 days	Mahrunisa and Abdullah
2.1	Develop Requirments	6	Research and surveys	2 days	Mahrunisa
2.2	Research Hosting Options	7	Testing	5 days	Abdullah and Haider
2.3	Design Site User Interface	8	Making report	3 days	Mahruisa

3.	**Development**	9	Coding, testing and execution	1 month	Whole team
3.1	Development the Architecture	10	Coding and execute	15 days	Whole Team
3.2	Develop Site	11	Coding and execute	15 days	Whole team
3.2.1	Develop Product Database	14	Analysis and testing	5 days	Mahrunisa
3.2.2	Develop Web Site				Whole team
3.2.3	Test the Site				Haider
4.	**Testing**	16	Perform tracking	20 days	Mahrunisa and Abdullah
4.1	Perform Risk Management	17	Testing and analyzing	3 days	Haider
4.2	Perform Quality Asuurance				Abdullah
4.3	Perform Issue Management				Haider and Abdullah
5.	**Deployment**	19	Testing and delievery	1 month	Whole team
5.1	Project report	20	reporting	5 days	Mahrunisa
5.2	Perform Deployment	21	Testing and delievery	3 days	Whole team

1.7.3. Gantt Chart

Gantt Chart: The Gantt chart is a visual representation of the project schedule that displays the start and end dates of tasks and their dependencies. It is a powerful tool for tracking progress and identifying potential issues. A Gantt chart for the ACE Trading platform development project would include tasks such as coding, testing, and deployment, and show the dependencies and duration of each task.

- Project initiation and planning: 10 days
- Requirements gathering and analysis: 15 days

- Design and development: 90 days
- Testing and quality assurance: 40 days
- Deployment and user acceptance testing: 19 days
- Post-deployment maintenance: 60 days

It's important to note that a Project plan, WBS and Gantt chart are not set in stone and it should be flexible as the project proceeds, and should be updated as necessary to reflect any changes in scope, schedule, or resources.

ID	Task Name	Start	Finish	Duration	Complete
1	Project initiation and planning	2022-09-02	2022-09-15	10.0 d.	100.0%
2	Requirements gathering and analysis	2022-09-19	2022-10-07	15.0 d.	100.0%
3	Design and Development	2022-10-10	2023-02-10	90.0 d.	29.7%
4	Testin and quality assurance	2023-02-10	2023-04-06	40.0 d.	0.0%
5	deployment and user acceptance	2023-04-07	2023-05-03	19.0 d.	0.0%
6	post-deployment	2023-05-05	2023-07-27	60.0 d.	0.0%

1.8. Report Outline

The ACE Trading project includes the development of a comprehensive online investment platform that allows individual investors to manage and monitor their forex, stock, and cryptocurrency investments.

The platform will be built using the Laravel framework to ensure that it is responsive and dynamic. It will include a user-friendly interface and a range of features and functionalities, such as real-time monitoring of investments, automated trading, and risk management tools.

In addition, the platform will utilize blockchain technology to provide secure and transparent transactions and a decentralized, fee-free trading system. The project will cover the design, development, testing, and deployment of the platform, and a maintenance period of 6 months after the deployment to ensure the smooth operation of the platform.

Chapter 2

Software Requirement Specifications

Chapter 2: Software Requirement Specifications

2.1. Introduction

2.1.1. Purpose

ACE Trading aims to provide a decentralized and fee-free trading system that ensures secure and transparent transactions. The combination of these features and functionalities will provide a unique and compelling solution for individual investors looking to make informed decisions and achieve their investment goals.

In addition, the platform will utilize blockchain technology to provide secure and transparent transactions and a decentralized, fee-free trading system. The project will cover the design, development, testing, and deployment of the platform, and a maintenance period of 6 months after the deployment to ensure the smooth operation of the platform.

2.1.2. Document Conventions

The whole document is written in MLA format. Topics introduced in section and subsection headings in a bold text. Highlighting is to point out words in the glossary and Calibri text and includes various diagrams like UML, ERD, and Architecture Diagram.

2.1.3. Intended Audience and Reading Suggestions

- End User

Every user should be comfortable of working with computer and net browsing. He must have basic knowledge of English too. He has must be some knowledge of how to use any websites.

- Administrator

Administrator is an entity that will manage entire system. An administrator can cover areas such as database, security and integration. He can view, modify or delete records. He can do all kind of alterations in the database.

- Project Manager

2.1.4. Product Scope

The scope of the ACE Trading project includes the development of a comprehensive online investment platform that allows individual investors to manage and monitor their forex, stock, and cryptocurrency investments.

The platform will be built using the Laravel framework to ensure that it is responsive and dynamic. It will include a user-friendly interface and a range of features and functionalities, such as real-time monitoring of investments, automated trading, and risk management tools.

In addition, the platform will utilize blockchain technology to provide secure and transparent transactions and a decentralized, fee-free trading system. The project will cover the design, development, testing, and deployment of the platform, and a maintenance period of 6 months after the deployment to ensure the smooth operation of the platform.

2.1.5. References

We use the existing system as the reference of the project

- E-trade
- TD Ameritrade

2.2. Overall Description

2.2.1. Product Perspective

There are many types of Cryptocurrency that are implemented in different platforms including Cryptocurrency in social networks, Cryptocurrency in social games, loyalty points and Cryptocurrency in peer to peer networks. These platforms can be classified into two main categories, centralized cryptocurrency platforms and decentralized cryptocurrency platforms. The centralized cryptocurrency can be defined as a Cryptocurrency system that has a centralized repository which is similar to the central bank. The administrator of that repository has full control of transferring the Cryptocurrency value between persons or from location to another. Whereas the decentralized cryptocurrency can be defined as the Cryptocurrency system that has no centralized repository and has no single administrator. De- centralized Cryptocurrency can be obtain by computing or manufacturing effort.

2.2.2. User Classes and Characteristics

- Supports Litecoin, Bitcoin and Ethereum.
- 2FA verification on login and registration
- Levels Referral System (Stable)
- KYC (Know your customers) Compliance
- Advanced users management

- Signup Bonus
- Email notification on registration, when ROI drops and at the end of investment circle.
- Live chat management
- Mass email to all users
- Referral system

2.2.3. Operating Environment

The platform will utilize blockchain technology to provide secure and transparent transactions and a decentralized and built using the Laravel framework to ensure that it is responsive and dynamic. It will include a user-friendly interface and a range of features and functionalities, such as real-time monitoring of investments, automated trading, and risk management tools.

2.2.4. Design and Implementation Constraints

Design Constraints:

- User interference design must be simple and attractive.
- User Friendly

Implementation Constraints:

- Code must be clean and readable.
- Product must me developed within time and budget.

2.2.5. Assumptions and Dependencies

Connection of Wi-Fi is important and can handle through computer or laptop and should know about the trading while using this.

2.3. External Interface Requirements

2.3.1. User Interfaces

- ✓ Currency Exchange Facility.
- ✓ Currency BUY Facility.
- ✓ Currency SELL Facility.
- ✓ Blog & Contact Facility.
- ✓ Infront Exchanger.

- ✓ Easy Signup & Signing.
- ✓ Sell Currency Activity.
- ✓ BUY Currency Activity.
- ✓ Exchange Currency Activity.
- ✓ 16+ Deposit Method.
- ✓ Deposit History.
- ✓ Easy to Withdraw.
- ✓ Withdraw History.
- ✓ Total Transaction Report.
- ✓ Profile Management.
- ✓ Reference System.
- ✓ And More.
- ✓ Cross Browser Optimized.

2.3.2. Hardware Interfaces

- Computer or laptop

2.3.3. Software Interfaces

- Laravel framework
- HTML, CSS, and JavaScript
- MySQL
- Object-Relational Mapping (ORM) tool

2.3.4. Communications Interfaces

- Email

2.4. System Features

<This template illustrates organizing the functional requirements for the product by system features, the major services provided by the product. You may prefer to organize this section by use case, mode of operation, user class, object class, functional hierarchy, or combinations of these, whatever makes the most logical sense for your product.>

2.4.1. Registration

2.4.1.1. Description and Priority

User must have to create account t before using the system and admin also have their own login.

2.4.1.2. Stimulus/Response Sequences

After login the user can access the all the functionality.

2.4.1.3. Functional Requirements

REQ-SF2-1: Registration

REQ-SF2-2: Add register person in database

2.4.2. Currency Exchange Faculty

2.4.2.1. Description and Priority

Cryptocurrency, a virtual currency designed to act as money and a form of payment outside the control of any one person, group, or entity, thus removing the need for third-party involvement in financial transactions.

2.4.2.2. Stimulus/Response Sequences

User can buy bitcoin by using this platform.

2.4.2.3. Functional Requirements

However, here are some common functional requirements you might consider when designing a Bitcoin trading platform:

User Registration: Allow users to create accounts and provide necessary information for identification, security, and compliance purposes.

Account Management: Enable users to manage their account settings, including profile information, security preferences, and notification settings.

Wallet Integration: Provide users with individual Bitcoin wallets to store and manage their Bitcoin balances securely.

Trading Dashboard: Offer a user-friendly interface that displays real-time market data, charts, and trading tools, allowing users to analyze market trends and make informed trading decisions.

Order Placement: Enable users to place various types of orders, such as market orders, limit orders, stop orders, and conditional orders, to buy or sell Bitcoin.

Order Book: Display a live order book that shows the current buy and sell orders, order depth, and order history to help users assess market liquidity and make trading decisions.

Trade Execution: Facilitate the execution of trades by matching buy and sell orders, ensuring fair price discovery and efficient order fulfillment.

Order Management: Provide users with the ability to manage their open orders, including modifying or canceling orders as needed.

Trading History: Maintain a comprehensive history of users' trading activities, including executed trades, open orders, and transaction details.

Portfolio Overview: Display a summary of users' Bitcoin holdings, including current balances, transaction history, profit/loss calculations, and portfolio performance metrics.

Deposit and Withdrawal: Allow users to deposit Bitcoin into their platform wallets and withdraw Bitcoin to external wallets securely.

Security Measures: Implement robust security measures, such as two-factor authentication (2FA), encryption, and cold storage, to safeguard user accounts and funds.

Customer Support: Offer a support system to assist users with inquiries, issues, and dispute resolution, such as a ticketing system, live chat, or email support.

Reporting and Analytics: Provide users with reporting tools and analytics to track their trading performance, generate transaction history reports, and analyze market data.

Mobile Compatibility: Develop mobile applications or responsive web design to allow users to access the platform on various devices.

It's important to note that these requirements are not exhaustive, and additional features and considerations may be necessary based on your specific trading platform's target audience and business objectives.

2.4.3. System Feature 3 – Sell Log

2.4.4. Live Chat

2.4.5. Manage all withdraws

2.5. Nonfunctional Requirements

A non-functional requirement is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviors. Non-functional requirements are "constraints", "quality attributes", and "non-behavioral requirements. Non-Functional Requirements includes following.

2.5.1. Performance Requirements

Our display system works 24/7. Our display will not be downgraded while running.

2.5.2. Safety Requirements

Using Blockchain technology to provide safety.

2.5.3. Security Requirements

Unauthorized person cannot access system software side and a person with fake ID & password cannot access any activity.

2.5.4. Portability Requirements

We can transfer a system or component of our project from one environment to another. We can deploy our project in any organization.

2.5.5 Interactive

Our system is very interactive to use, all functionalities and features are visible and easy to use.

2.6. Domain Requirements

- Windows XP and earlier

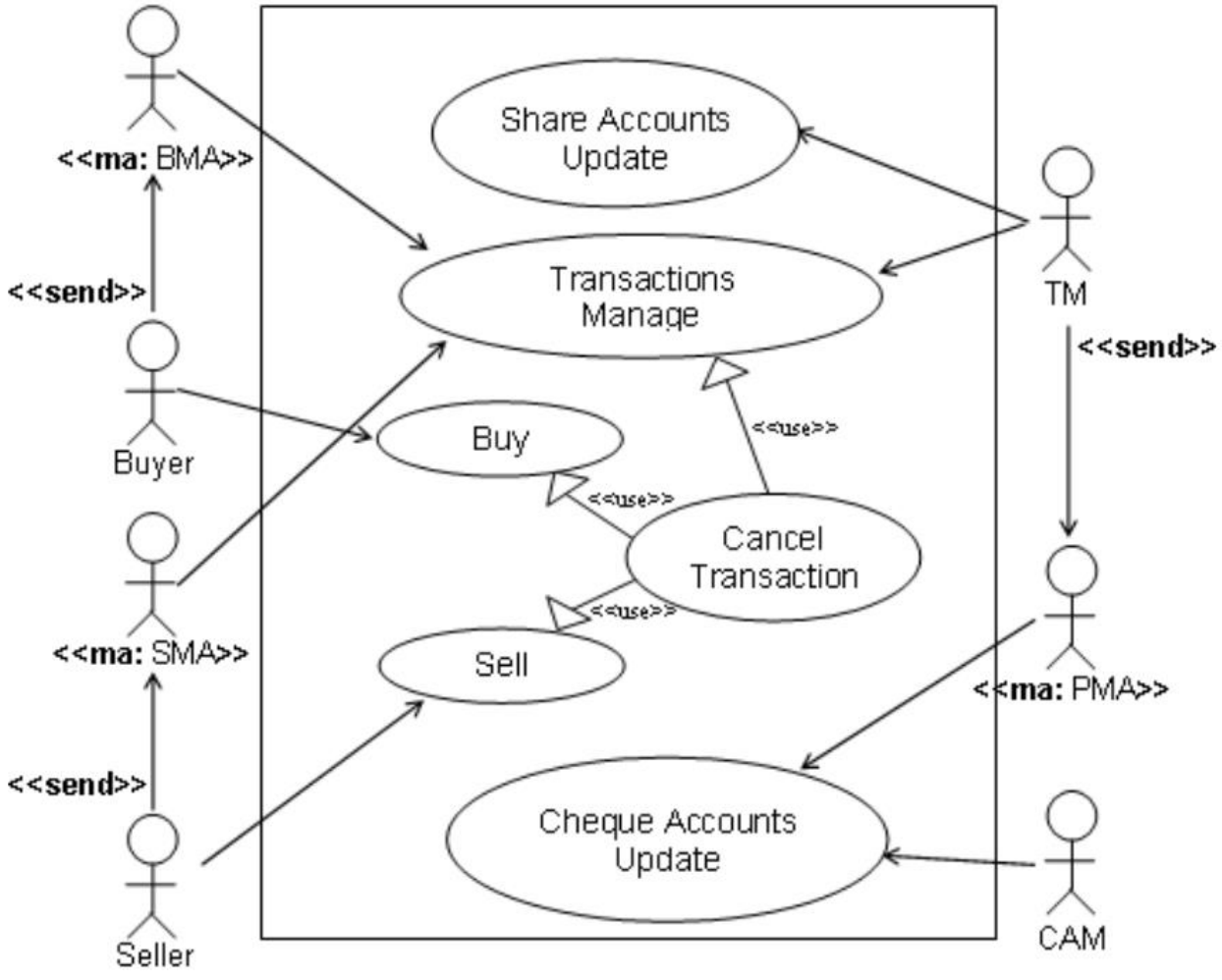
Chapter 3

Use Case Analysis

Chapter 3: Use Case Analysis

A use case diagram is a graphical depiction of a user's possible interactions with a system. A use case diagram shows various use cases and different types of users the system has and will often be accompanied by other types of diagrams as well. The use cases are represented by either circles or ellipses.

3.1. Use Case Model



3.2. Use Cases Description

UCDs have only 4 major elements: The actors that the system you are describing interacts with, the system itself, the use cases, or services, that the system knows how to perform, and the lines that represent relationships between these elements.

Chapter 4

System Design

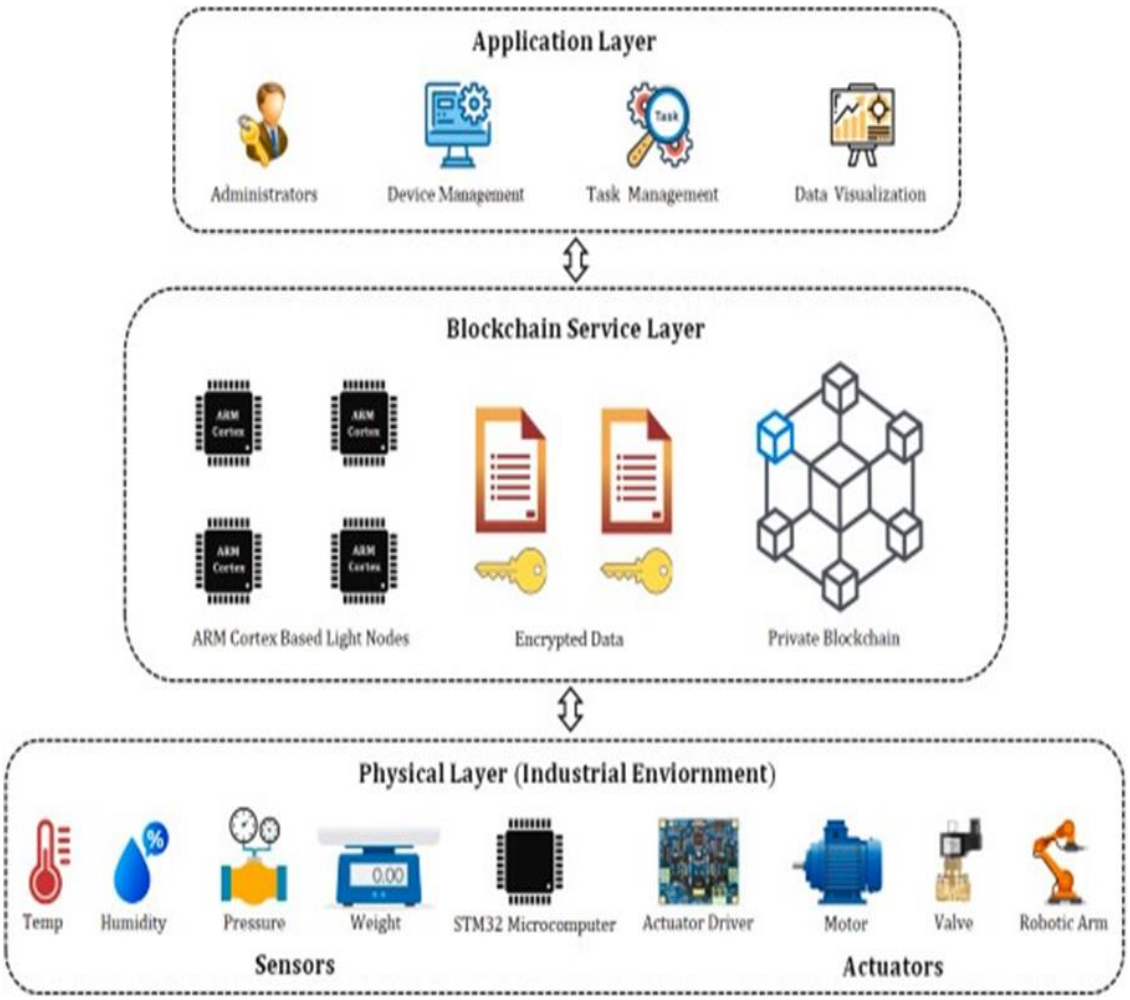
Chapter 4: System Design

Systems design interfaces, and data for an electronic control system to satisfy specified requirements. System design could be seen as the application of system theory to product development. There is some overlap with the disciplines of system analysis, system architecture and system engineering.

4.1. Architecture Diagram

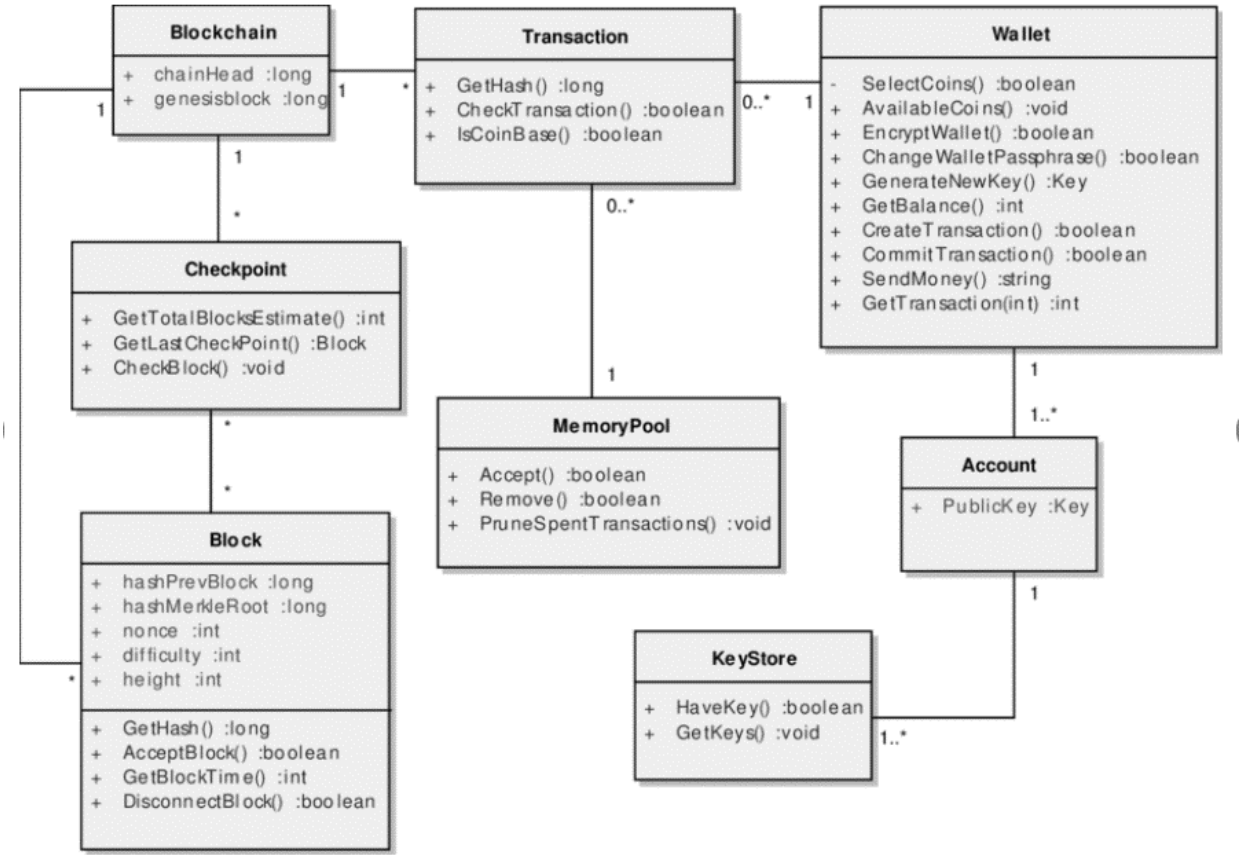
The ACE Trading platform will have a multi-tier architecture, which consists of the following layers:

- **Presentation Layer:** This layer will be responsible for displaying the user interface and handling user interactions. It will be built using HTML, CSS, and JavaScript to ensure a responsive and user-friendly interface.
- **Business Layer:** This layer will be responsible for handling business logic and implementing core functionality. It will be built using Laravel framework, a PHP-based web application framework, to ensure scalability and maintainability.
- **Data Access Layer:** This layer will be responsible for interacting with the database to retrieve and store data. It will be built using an object-relational mapping (ORM) tool, such as Eloquent, which is included in the Laravel framework, to handle database operations.
- **Database:** The platform will use a relational database management system, such as MySQL, to store data such as user information, investment details, and transaction records
- **Blockchain Layer:** For crypto trading, the platform will use blockchain technology to ensure secure and transparent transactions. This layer will be responsible for handling transactions, creating new blocks and interacting with the blockchain network.


Application Layer
Administrators
Device Management
Task Management
Data Visualization
Blockchain Service Layer
ARM Cortex Based Light Nodes
Encrypted Data
Private Blockchain
Physical Layer (Industrial Enviornment)
Temp
Humidity
Pressure
Weight
STM32 Microcomputer
Actuator Driver
Motor
Valve
Robotic Arm
Sensors
Actuators

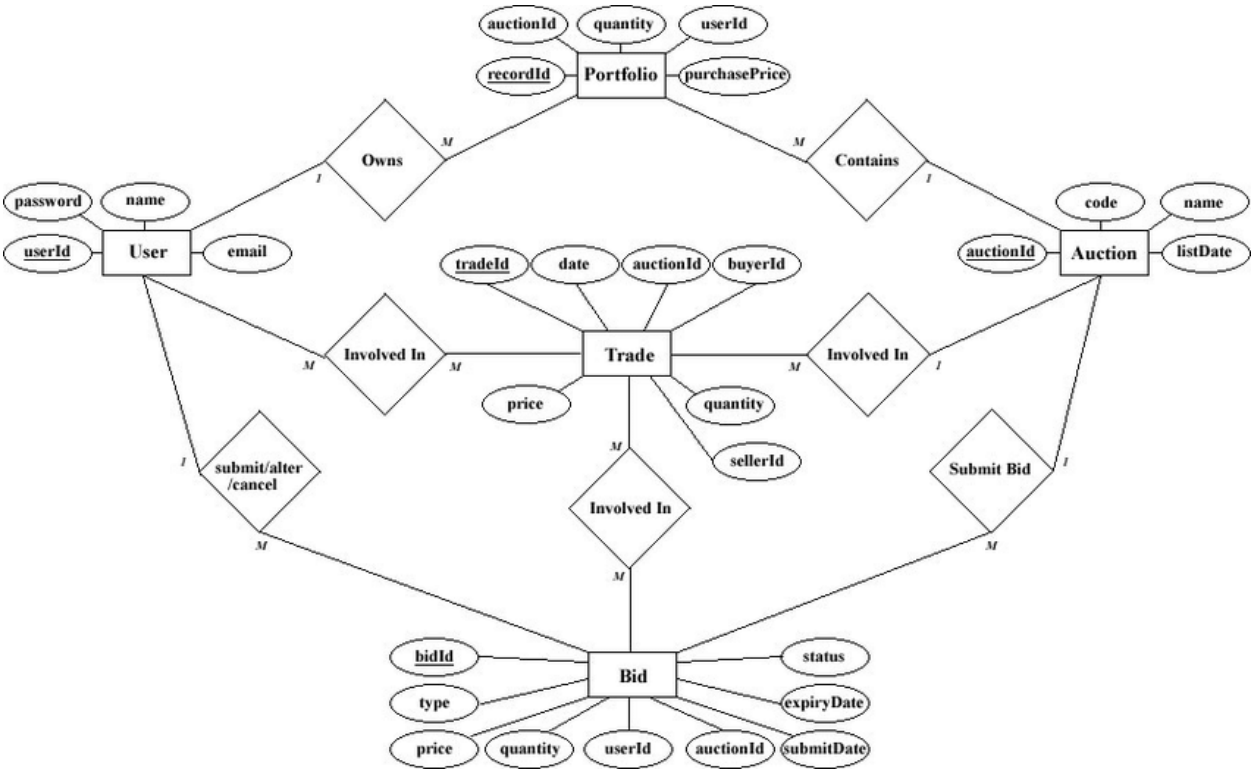
4.2. Domain Model

In software engineering, a domain model is a conceptual model of the domain that incorporates both behavior and data. In ontology engineering, a domain model is a formal representation of a knowledge domain with concepts, roles, datatypes, individuals, and rules, typically grounded in a description logic.



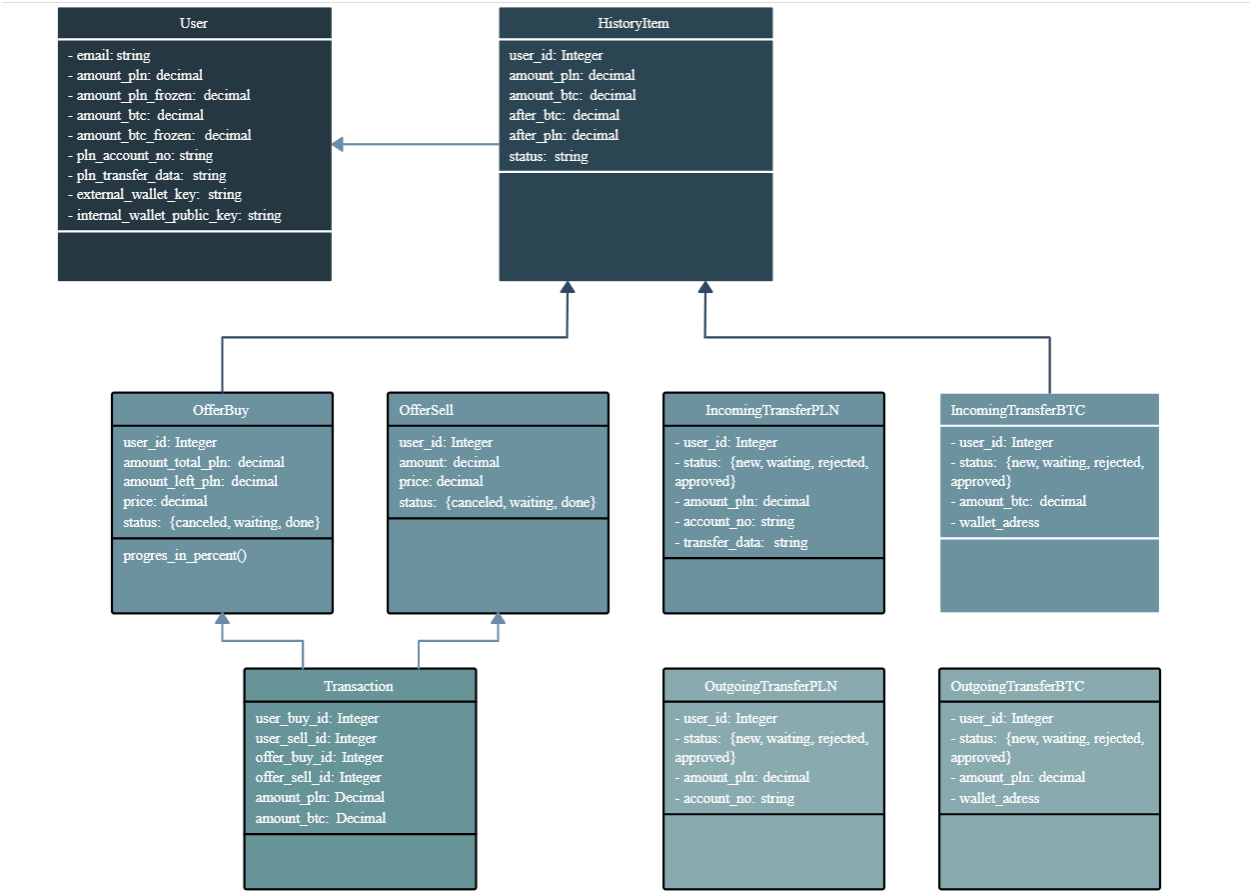
4.3. Entity Relationship Diagram with data dictionary

While a conceptual or logical Entity Relationship Diagram (ERD) will focus on the high-level business concepts, a Data Dictionary will provide more detail about a business concept, such as standard definitions of data elements, their meanings, and allowable values.



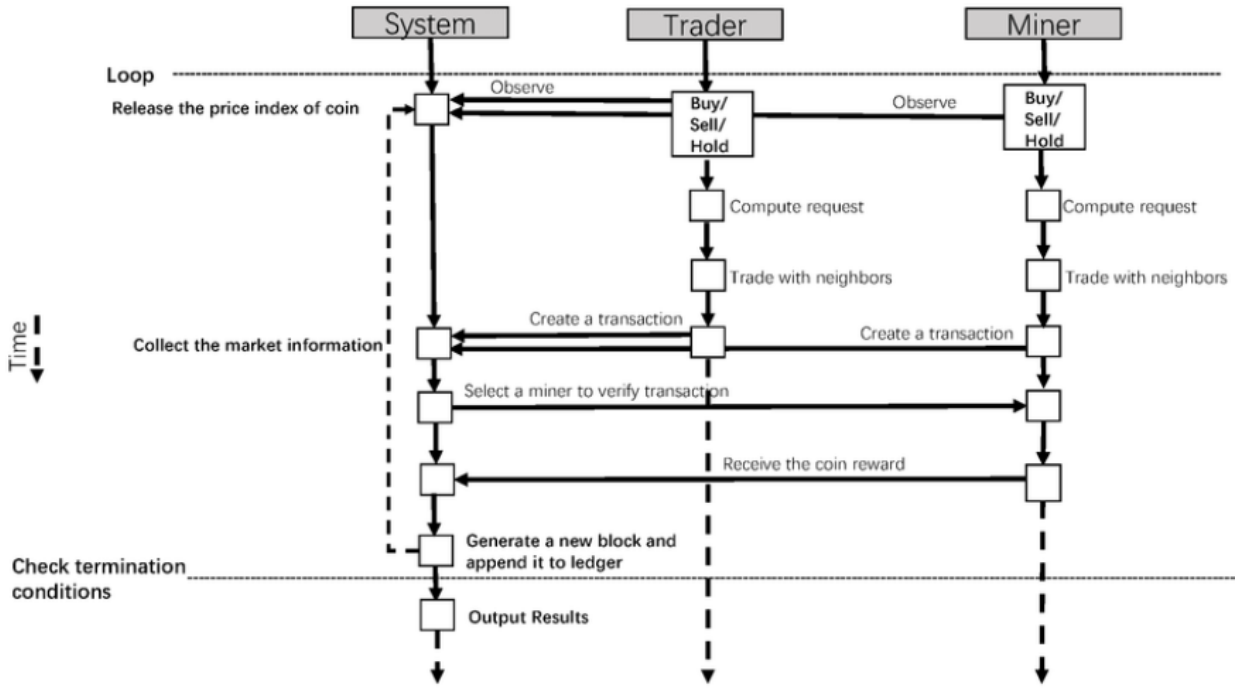
4.4. Class Diagram

A class diagram resembles a flowchart in which classes are portrayed as boxes, each box having three rectangles inside. The top rectangle contains the name of the class; the middle rectangle contains the attributes of the class; the lower rectangle contains the methods, also called operations, of the class.



4.5. Sequence / Collaboration Diagram

The sequence diagram are used to represent the sequence of messages that are flowing from one object to another. The collaboration diagram are used to represent the structural organization of the system and the messages that are sent and received.



4.6. Operation contracts

Operation contracts for a Bitcoin trading platform typically outline the terms and conditions governing the platform's operation and the relationship between the platform and its users. Here are some key elements that are often included:

Parties: Identify the parties involved in the contract, including the platform operator and the users.

Purpose: Clearly state the purpose of the contract, which is to establish the terms of operation for the Bitcoin trading platform.

Definitions: Provide definitions for key terms used throughout the contract, such as "platform," "user," "Bitcoin," "trading," and any other relevant terms.

Platform Services: Specify the services provided by the platform, such as facilitating the buying and selling of Bitcoin, providing a trading interface, order matching, and account management.

User Obligations: Outline the responsibilities and obligations of the users, such as complying with platform rules and regulations, providing accurate information during registration, and maintaining the security of their accounts.

Platform Obligations: Describe the responsibilities and obligations of the platform operator, such as maintaining the security and stability of the platform, ensuring the privacy of user data, and providing customer support.

Trading Rules: Establish the rules and regulations related to trading on the platform, including order types, trading fees, transaction limits, and any other relevant guidelines.

Account Management: Specify the procedures for account creation, verification, and management. This may include guidelines for password protection, two-factor authentication, and account suspension or termination.

Data Privacy and Security: Address the platform's policies regarding user data privacy, storage, and protection. This section may include details on data encryption, data retention, and compliance with relevant data protection laws.

Liability and Dispute Resolution: Define the liabilities of both parties in case of system failures, security breaches, or other issues. Include a clause on dispute resolution, such as arbitration or mediation, to handle conflicts between the platform and its users.

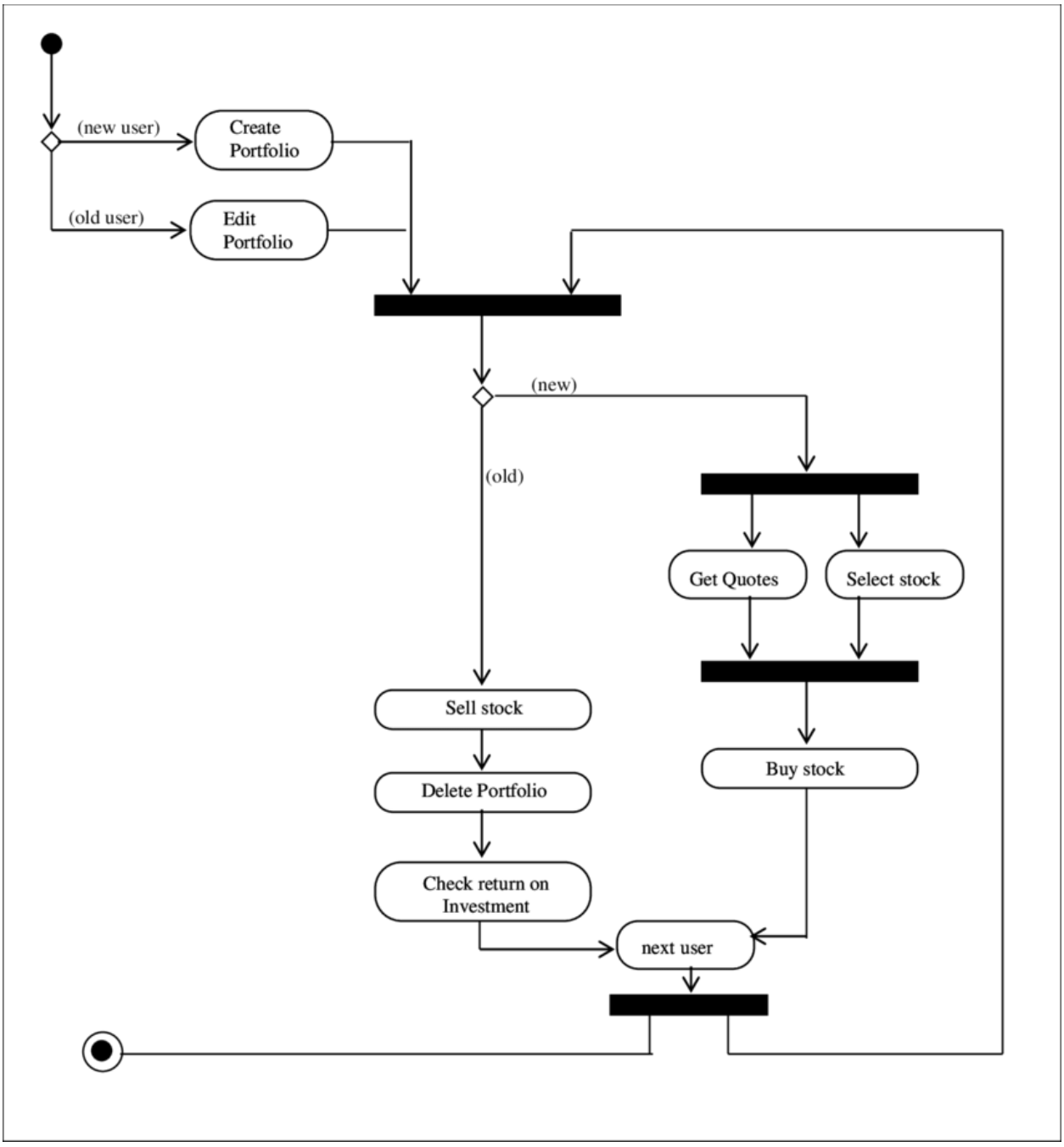
Termination: Specify the conditions under which either party can terminate the contract, including reasons for termination and any associated procedures.

Governing Law and Jurisdiction: Determine the applicable law governing the contract and identify the jurisdiction in which any disputes will be resolved.

It's important to note that the above points provide a general overview and should not be considered as legal advice. When creating operation contracts for a Bitcoin trading platform, it's crucial to consult with legal professionals familiar with the relevant laws and regulations in your jurisdiction to ensure compliance and protection for all parties involved.

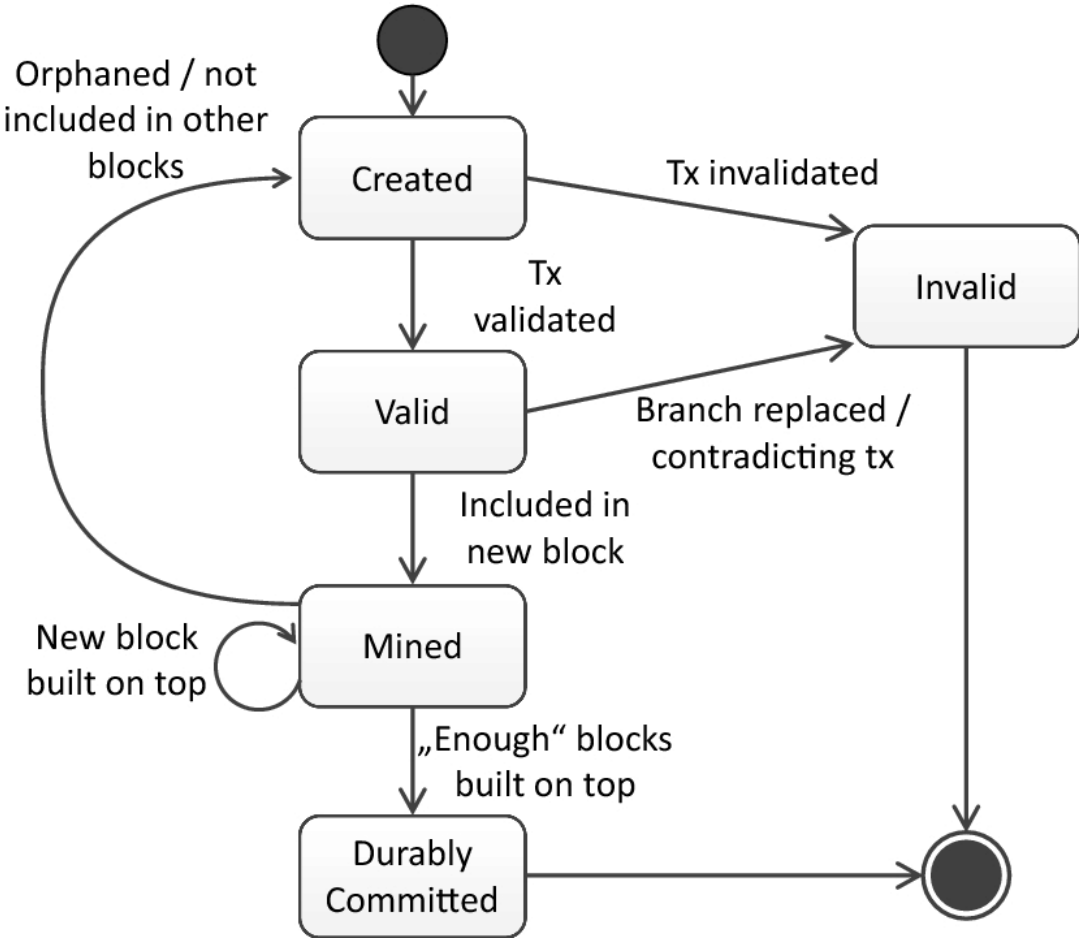
4.7. Activity Diagram

An activity diagram visually presents a series of actions or flow of control in a system similar to a flowchart or a data flow diagram. Activity diagrams are often used in business process modeling.



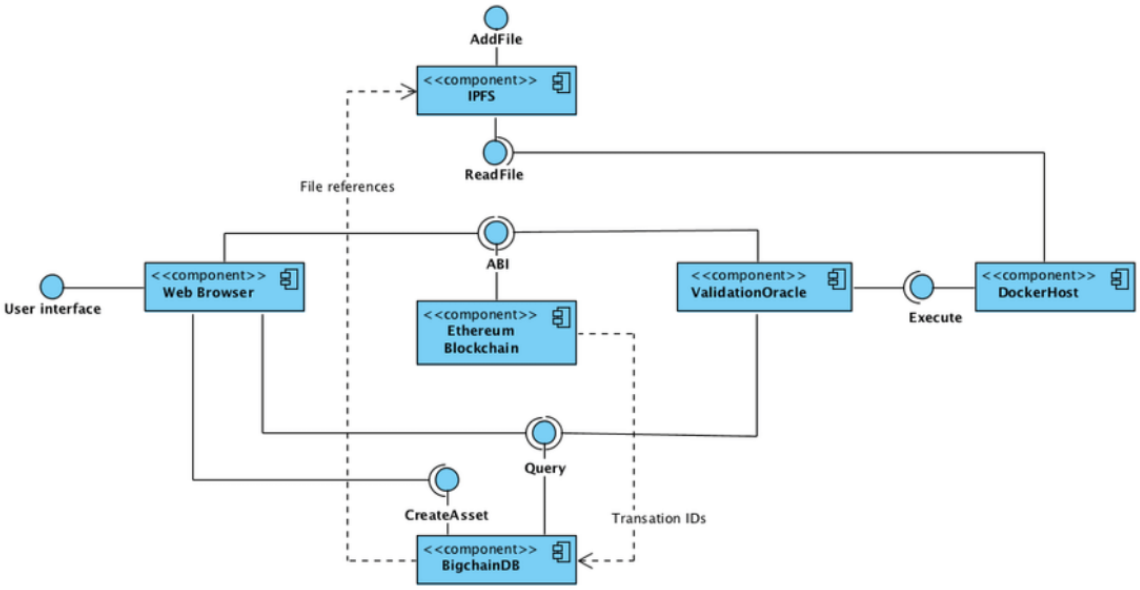
4.8. State Transition Diagram

A state transition diagram is used to represent a finite state machine. These are used to model objects which have a finite number of possible states and whose interaction with the outside world can be described by its state changes in response to a finite number of events



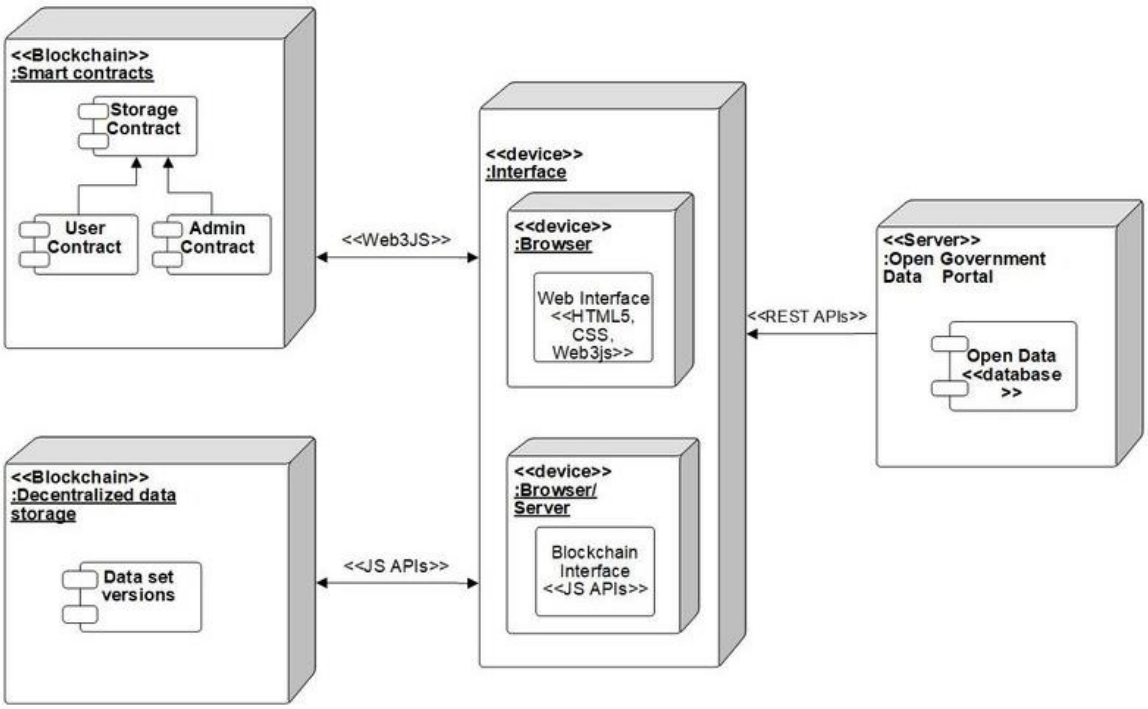
4.9. Component Diagram

A component is a logical unit block of the system, a slightly higher abstraction than classes. It is represented as a rectangle with a smaller rectangle in the upper right corner with tabs or the word written above the name of the component to help distinguish it from a class.



4.10. Deployment Diagram

In UML, deployment diagrams model the physical architecture of a system. Deployment diagrams show the relationships between the software and hardware components in the system and the physical distribution of the processing.



4.11. Data Flow diagram

Description of the Data Flow Diagram for a Bitcoin Trading Platform:

User Interaction:

Users interact with the trading platform through the user interface.

They can perform actions such as registering an account, logging in, placing buy/sell orders, and viewing their portfolio.

User Management:

User registration and login requests are processed by the user management system.

User data is stored in a database and can be accessed for authentication and authorization purposes.

Market Data Retrieval:

The trading platform fetches real-time market data from various sources, such as exchanges or price data providers.

The market data includes information about Bitcoin prices, order book depth, and recent trades.

Order Placement:

When users place buy or sell orders, the orders are processed and validated.

The trading platform checks if the user has sufficient funds and if the order is within the defined trading rules.

Order Execution:

The trading platform matches buy and sell orders based on price and other criteria.

Executed trades are recorded and updated in the order book.

Users' balances are adjusted accordingly, reflecting the executed trades.

Account Management:

Users can view their account information, including their portfolio balance, transaction history, and open orders.

Account-related actions, such as depositing or withdrawing funds, are processed securely.

Reporting and Analytics:

The trading platform generates reports and analytics based on user activities, trading volumes, and market data.

These reports can provide insights into trading patterns, user behavior, and platform performance.

External Integrations:

The trading platform may integrate with external systems, such as payment gateways, identity verification services, or third-party APIs.

These integrations enable functionalities such as fund deposits, KYC verification, or accessing external market data.

Security and Compliance:

The trading platform ensures security measures, such as encryption, secure connections, and user authentication.

Compliance procedures, such as KYC/AML checks, may be performed during user registration or fund transfers.

This high-level data flow diagram outlines the major components and data flows within a Bitcoin trading platform. It represents the flow of data between various system modules and external components. For a more detailed and comprehensive diagram, it is recommended to consult with a professional who can understand the specific requirements and design a diagram tailored to the platform's architecture.

Chapter 5

Implementation

Chapter 5: Implementation

In this chapter we will discuss about the implementation and the procedure. The process of making formal plans often highly intricate, widely influential plans into reality.

5.1. Important Flow Control/Pseudo codes

Flow Control:

1. Start website.
2. Introduction Screen.
3. Optional Screen (Log in or Sign Up)
4. If Sign Up.

   4.1. User enters all the required info fields..
5. If Log in.
6. Fills the required field
7. Next to the main screen.
8. All features of application are on main screen.

   For user:

- Currency Exchange Facility.
- Currency BUY Facility.
- Currency SELL Facility.
- Blog & Contact Facility.
- Infront Exchanger.
- Easy Signup & Signing.
- Sell Currency Activity.
- BUY Currency Activity.
- Exchange Currency Activity.
- 16+ Deposit Method.
- Deposit History.
- Easy to Withdraw.
- Withdraw History.
- Total Transaction Report.
- Profile Management.

- Reference System.
- And More.
- Cross Browser Optimized.

9. User can choose any option according to his/her requirement.

5.2. Components, Libraries, Web Services and stubs

Bitcoin Core: Bitcoin Core is the reference implementation of the Bitcoin protocol. It provides the necessary functionality to interact with the Bitcoin network, including transaction verification, wallet management, and network communication.

Trading Engine: The trading engine is the core component responsible for order matching, executing trades, and managing the order book. It handles functions such as order placement, order cancellation, and trade settlement.

Wallet Integration: Wallet integration is crucial for managing user funds and enabling deposits and withdrawals. Libraries such as BitcoinJS or BitcoinJ can be used to interact with Bitcoin wallets and perform operations like creating addresses, generating transactions, and signing transactions.

Price Data Providers: Access to real-time price data is essential for accurate market analysis and order execution. Web services and APIs from reliable price data providers like CoinMarketCap, CoinGecko, or exchanges with public APIs can be used to fetch up-to-date market data.

Exchange APIs: If the trading platform allows users to trade on multiple exchanges, integrating exchange APIs is necessary. Popular exchanges like Binance, Coinbase, or Kraken provide APIs that allow fetching market data, placing orders, and managing user accounts.

Payment Gateway Integration: Integrating a payment gateway enables users to deposit funds using various payment methods such as credit cards, bank transfers, or cryptocurrencies.

Payment service providers like BitPay, CoinGate, or Stripe offer APIs that facilitate seamless payment processing.

KYC/AML Services: To comply with regulatory requirements, Know Your Customer (KYC) and Anti-Money Laundering (AML) services can be integrated. Providers like Jumio, Onfido, or Thomson Reuters offer identity verification services that help onboard users securely and perform necessary compliance checks.

Analytics and Reporting Tools: To monitor platform performance, user activity, and generate reports, analytics and reporting tools can be employed. Services like Google Analytics, Mixpanel, or custom-built analytics solutions can provide valuable insights into platform usage and user behavior.

Email and Notification Services: Sending transactional emails, order notifications, or account-related notifications is crucial for keeping users informed. Email service providers like SendGrid, Mailgun, or custom-built email systems can be utilized for reliable email delivery.

Stubs and Mock APIs: During development and testing, stubs or mock APIs can be used to simulate external dependencies or components that are not yet fully implemented. Tools like WireMock, Nock, or custom stub implementations can be helpful in creating dummy responses for testing purposes.

5.3. Deployment Environment

AGILE methodology is a practice that promotes continuous iteration of development and testing throughout the software development lifecycle of the project. Both development and testing activities are concurrent unlike the Waterfall model.

The agile software development emphasizes on four core values.

- Individual and team interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

5.4. Tools and Techniques

- **Laravel framework:** A PHP-based web application framework that will be used to implement the business logic and core functionality of the platform. Laravel provides a range of tools and features that facilitate the development of a responsive and dynamic platform.
- **HTML, CSS, and JavaScript:** These will be used to design and implement the presentation layer of the platform, ensuring that it is user-friendly and responsive.
- **MySQL:** This will be used as the relational database management system to store and manage data such as user information, investment details, and transaction records.
- **Object-Relational Mapping (ORM) tool:** An ORM tool, such as Eloquent, which is included in the Laravel framework, will be used to handle database operations.
- Blockchain technology: This will be used to ensure secure and transparent transactions on the platform
- **Git:** Git will be used for version control and collaboration during the development process.
- **Cloud-based Infrastructure:** Platform will be hosted on cloud-based infrastructure like AWS,Azure,Google cloud to ensure high availability, scalability and data security.
- **Agile Methodology:** Agile development methodology will be used to manage the project, which emphasizes flexibility and collaboration, which allows teams to quickly respond to changes and deliver features incrementally.
- **Continuous Integration/Continuous Deployment**: Automated testing, and building and deployment process with the help of tools like Travis CI, Jenkins, CircleCI etc will be used to ensure the seamless integration of code changes.

5.5. Best Practices / Coding Standards

The best practices for designing and developing with Flutter to improve code quality, readability, maintainability, and productivity.

- Refactor code into widgets rather than methods.
- Make build function pure.
- Use state management.
- Have a well-defined Architecture.
- Follow effective dart style guide.

5.6. Version Control

- Xamp Control Pannel v.3.2.
- Laravel version 9

Chapter 6

Testing and Evaluation

Chapter 6: Testing and Evaluation

Testing and evaluation are crucial steps in the development and assessment of various systems, products, or processes. These steps help determine the performance, effectiveness, and quality of the subject under consideration. Let's explore testing and evaluation in more detail.

6.1. Use Case Testing

Use Case Testing is a technique used to verify the behavior of a system by testing its interactions with different actors or users. It is based on the concept of use cases, which represent specific scenarios or interactions between the system and its users. Use Case Testing helps ensure that the system meets the functional requirements and handles different user interactions correctly.

1. Use Case: User Registration
    - Description: Users create an account on the Bitcoin trading platform.
    - Test Cases:
        - Test Case 1: User provides valid information and successfully registers.
        - Test Case 2: User provides invalid or incomplete information, and the system displays appropriate error messages.
        - Test Case 3: User tries to register with an already existing username or email, and the system handles it correctly.
2. Use Case: User Login
    - Description: Users authenticate themselves and log into their accounts.
    - Test Cases:
        - Test Case 1: User enters correct credentials and successfully logs in.
        - Test Case 2: User enters incorrect credentials, and the system denies access.
        - Test Case 3: User's account is locked after multiple failed login attempts, and the system handles the lockout mechanism properly.
3. Use Case: Buy Bitcoin
    - Description: Users purchase Bitcoin on the trading platform.
    - Test Cases:

- Test Case 1: User selects the desired Bitcoin quantity and completes the purchase successfully.
- Test Case 2: User tries to buy Bitcoin with insufficient funds, and the system displays an appropriate error message.
- Test Case 3: User encounters a technical issue during the purchase process, and the system handles it gracefully.

4. Use Case: Sell Bitcoin
    - Description: Users sell their Bitcoin on the trading platform.
    - Test Cases:
        - Test Case 1: User selects the Bitcoin quantity to sell and completes the transaction successfully.
        - Test Case 2: User tries to sell more Bitcoin than they own, and the system prevents the transaction.
        - Test Case 3: User encounters an error during the selling process, and the system handles it appropriately.
5. Use Case: Account Security
    - Description: Users manage their account security settings.
    - Test Cases:
        - Test Case 1: User changes their password successfully.
        - Test Case 2: User enables two-factor authentication and verifies its functionality.
        - Test Case 3: User requests a password reset and successfully completes the process.

6.2. Equivalence partitioning

Equivalence Partitioning is a software testing technique that involves dividing the input data into groups or partitions based on the assumption that if one input within a partition behaves a certain way, all other inputs in the same partition will exhibit the same behavior. It is based on the concept that testing one representative value from each partition is sufficient to validate the behavior of all other values within the same partition.

The key steps involved in equivalence partitioning are as follows:

1. Identify input variables: Determine the input variables or parameters that influence the behavior of the system or component under test.
2. Define equivalence classes: Divide the input values for each variable into groups or equivalence classes based on their expected behavior. Each equivalence class represents a set of inputs that should result in the same behavior.
3. Select representative values: Choose one or more representative values from each equivalence class. These values should be typical of the inputs within the class.
4. Design test cases: Create test cases using the representative values. Each test case should cover one equivalence class and represent the behavior of all other values within that class.
5. Execute the test cases: Run the designed test cases and observe the system's response or behavior.
6. Validate results: Compare the observed results with the expected behavior for each equivalence class. If the behavior is consistent within a class, it indicates that the partitioning was successful.

The main advantage of equivalence partitioning is that it allows for a more efficient testing process by reducing the number of test cases needed to achieve reasonable coverage. By selecting representative values from each equivalence class, it is possible to identify potential defects or issues within a specific class without testing every single value.

6.3. Boundary value analysis

Boundary value analysis is a testing technique used to identify defects and errors at the boundaries of input values. When it comes to a Bitcoin trading platform, here are some potential boundary values to consider:

1. Minimum and maximum transaction amounts: Determine the minimum and maximum values allowed for Bitcoin transactions on the platform. Test the platform with values just below the minimum and just above the maximum to ensure that the system handles these boundary conditions correctly.

2. Minimum and maximum account balances: Identify the minimum and maximum values for account balances on the platform. Test scenarios where the account balance is slightly below the minimum and slightly above the maximum to verify that the platform handles these situations appropriately.
3. Minimum and maximum price limits: Establish the minimum and maximum price limits for Bitcoin trades. Test cases where the trade price is just below the minimum and just above the maximum to ensure that the system responds correctly to these boundary conditions.
4. Maximum number of transactions: Determine the maximum number of transactions that can be executed within a specific timeframe or for a particular user account. Test scenarios where the number of transactions is one less than the maximum and one more than the maximum to validate the system's behavior under these conditions.
5. Minimum and maximum input field lengths: Identify the minimum and maximum lengths allowed for input fields such as usernames, passwords, or transaction descriptions. Test cases with values just below the minimum and just above the maximum to ensure that the system handles these boundary conditions properly.
6. Minimum and maximum withdrawal limits: Define the minimum and maximum limits for Bitcoin withdrawals on the platform. Test withdrawal requests just below the minimum and just above the maximum to verify that the system processes them correctly.

Remember that these are general examples, and the specific boundary values may vary depending on the requirements and constraints of the Bitcoin trading platform you are testing. It's crucial to thoroughly analyze the system's requirements and consult with the development team to identify the appropriate boundary values for your testing efforts.

6.4. Data flow testing

Data flow testing is a technique used to analyze and validate the flow of data within a software system. In the case of a Bitcoin trading platform, data flow testing helps ensure that data is correctly transmitted and processed throughout the various components of the platform. While I can describe the general concept of data flow testing, I'm unable to generate specific diagrams.

However, I can provide you with a textual representation of how data flows within a typical Bitcoin trading platform.

In a Bitcoin trading platform, the data flow can be divided into several key components:

1. User Interface:
   - User input: Traders input their buy/sell orders, account details, and other relevant information through the user interface.
   - User output: The platform provides feedback and updates to the traders, displaying their account balance, transaction history, order status, etc.

2. Order Processing:
   - Order submission: The user's buy/sell orders are transmitted to the order processing system.
   - Order validation: The order processing system validates the orders for correctness, checking factors such as available balance, order type, price, etc.
   - Order execution: Valid orders are executed, which involves matching buy and sell orders and updating the order book.
   - Order confirmation: The status of the executed order is communicated back to the user interface.

3. Account Management:
   - Account information: User account details, including balance, transaction history, and personal information, are stored and managed within the platform's account management system.
   - Account updates: Account balance and transaction history are updated based on order execution and other relevant activities.

4. Market Data:
   - Price feeds: The platform receives market data, including the current Bitcoin price and trading volume, from external sources such as exchanges or market data providers.
   - Data processing: The received market data is processed and used for various purposes, including order matching and displaying real-time market information on the user interface.

5. Integration with External Services:

- External exchanges: The platform may integrate with external cryptocurrency exchanges to access liquidity and execute orders.
- Payment gateways: If the platform supports fiat currency transactions, integration with payment gateways allows users to deposit and withdraw funds.

These components represent a high-level overview of the data flow within a Bitcoin trading platform. The actual implementation and architecture may vary depending on the specific platform and its design choices. It's essential to thoroughly test the data flow at each stage to ensure that data is transmitted accurately and consistently, avoiding potential issues such as order mismatches, incorrect balance calculations, or data corruption.

6.5. Unit testing

Unit testing for a Bitcoin trading platform involves testing the individual components or units of the system to ensure they function correctly and meet the desired specifications. Here are some steps you can follow to perform unit testing for a Bitcoin trading platform:

1. Identify the units: Break down the platform into smaller units such as classes, functions, or modules that can be tested independently. For example, you may have units for order placement, trade execution, user authentication, and so on.

2. Write test cases: Create a set of test cases for each unit that cover different scenarios and use cases. Test cases should include both valid and invalid inputs to ensure the units handle various situations correctly. For instance, you can test placing buy/sell orders, verifying account balances, and handling edge cases like network failures or invalid user inputs.

3. Set up test data: Prepare the necessary test data, including mock data or test fixtures, to simulate different scenarios. For a Bitcoin trading platform, you might need mock user accounts, sample order data, market data, and simulated wallet balances.

4. Implement test code: Write the unit tests using a suitable testing framework or library. Popular options for unit testing in various programming languages include frameworks like JUnit (Java), NUnit (.NET), Pytest (Python), and Mocha (JavaScript). Write test methods to execute the test cases defined earlier and verify the expected outcomes.

5. Execute the unit tests: Run the unit tests and observe the results. Most testing frameworks provide ways to execute tests and generate reports indicating which tests passed, failed, or encountered errors. Investigate failed tests to identify and fix any issues in the code.

6. Test coverage analysis: Consider using a code coverage tool to assess the effectiveness of your unit tests. Code coverage helps you determine which parts of the code were exercised during testing and identify any areas that may require additional testing.

7. Refine and iterate: Analyze the test results, update or add new test cases as necessary, and repeat the unit testing process. It is important to refine your tests continually to cover different scenarios and catch potential bugs or vulnerabilities.

Remember that unit tests are focused on isolating and testing individual units of code, so you may need to use techniques like mocking or stubbing to simulate dependencies or external services (e.g., market data APIs, blockchain interactions) during testing.

Additionally, it's crucial to complement unit tests with integration tests, end-to-end tests, and security testing to ensure the overall functionality, performance, and security of your Bitcoin trading platform.

6.6. Integration testing

Integration testing for a Bitcoin trading platform involves testing the interaction between various components of the system to ensure they work together seamlessly. Here's a diagram illustrating the integration testing process for a Bitcoin trading platform:

```
+-----------------------+
|   Bitcoin Trading     |
|      Platform         |
+-----------------------+
       |  |  |
       |  |  |
       |  |  |
       |  |  |
       |  |  |
+--------|---|---|--------+
|       |  |  |       |
|   +---v---v---v---+   |
|   |   Trading    |   |
|   |   Engine     |   |
|   +-------|--------+   |
|           |            |
|           |            |
|           |            |
|           |            |
|   +-------v--------+   |
|   |  Order Book   |   |
|   |  Management   |   |
|   +-------|--------+   |
|           |            |
|           |            |
```

```
|         |         |
|         |         |
|   +-------v--------+   |
|   |  User Wallet  |   |
|   |  Management   |   |
|   +---------------+   |
|                 |
|   +---------------+   |
|   |  External     |   |
|   |  Services     |   |
|   +---------------+   |
+-----------------------+
```

In this diagram, we have several components of the Bitcoin trading platform:

1. Trading Engine: This component handles the execution of buy and sell orders on the platform. It interacts with the Order Book Management component to match orders and execute trades.

2. Order Book Management: This component manages the order book, which keeps track of all buy and sell orders. It receives order updates from the Trading Engine and ensures proper order matching and execution.

3. User Wallet Management: This component manages user wallets, which store their Bitcoin balances and transaction history. It handles operations such as depositing, withdrawing, and transferring Bitcoins.

4. External Services: This represents various external services that the platform interacts with, such as price feeds, payment gateways, and blockchain explorers.

During integration testing, the interactions between these components are tested to ensure they function correctly together. The following scenarios can be tested:

1. Placing buy/sell orders and verifying that they are reflected in the order book and trading engine.
2. Matching buy and sell orders and executing trades.
3. Verifying that user balances are updated correctly after executing trades.
4. Depositing Bitcoins into user wallets and verifying the updated balance.
5. Withdrawing Bitcoins from user wallets and ensuring the correct deduction from the balance.
6. Interacting with external services, such as retrieving real-time prices and performing payment transactions.

By thoroughly testing these integration points and ensuring the seamless functioning of the components, the Bitcoin trading platform can be validated for its overall functionality and reliability.

6.7. Performance testing

When it comes to performance testing for a Bitcoin trading platform, there are several key aspects that you should consider. Here is a general outline of the steps you can follow to perform performance testing for a Bitcoin trading platform:

1. Define Performance Objectives: Start by defining clear performance objectives for your trading platform. These objectives could include response time expectations, throughput requirements, maximum concurrent user load, and any other relevant metrics.

2. Identify Performance Scenarios: Identify the critical scenarios that you want to test. These scenarios could include placing trades, executing transactions, viewing real-time market data, and other common user actions. Consider both normal and peak load scenarios.

3. Set Up Test Environment: Set up a test environment that closely resembles your production environment. This includes configuring servers, network infrastructure, and any third-party

services used by the platform. Consider using realistic data sets to simulate real-world conditions.

4. Define Test Metrics: Determine the performance metrics you want to measure during the testing process. This could include response times, transaction throughput, server resource utilization (CPU, memory, network bandwidth), and any other relevant metrics that help evaluate the system's performance.

5. Design Test Cases: Based on the identified scenarios, design specific test cases that simulate user actions. These test cases should reflect different user profiles, load levels, and transaction volumes. Consider using automated testing tools to simulate concurrent user activity.

6. Execute Performance Tests: Run the performance tests using the defined test cases. Monitor and measure the performance metrics in real-time. Gather data on response times, transaction success rates, server resource utilization, and other relevant performance indicators.

7. Analyze Test Results: Analyze the test results to identify any performance bottlenecks or issues. Look for areas where the system may be experiencing delays or exhibiting poor performance. Pinpoint the root causes of these issues.

8. Optimize and Retest: Based on the analysis of test results, make necessary optimizations to address the identified performance bottlenecks. This could involve optimizing code, database queries, server configurations, or network infrastructure. Once the optimizations are implemented, rerun the performance tests to verify the improvements.

9. Load Testing: In addition to performance testing, consider conducting load testing to evaluate how the system handles high user loads. Load testing involves simulating a large number of concurrent users to assess the system's scalability and stability under heavy load conditions.

10. Monitor and Fine-Tune: Continuously monitor the performance of the trading platform in production to ensure it meets the desired performance objectives. Regularly review performance metrics and user feedback to identify areas for further optimization and improvement.

Remember that performance testing is an iterative process, and it's important to perform testing at different stages of development to catch performance issues early on. Additionally, it's crucial to keep up with the evolving needs and demands of your users, as performance requirements may change over time.

6.8. Stress Testing

Stress testing a Bitcoin trading platform is an essential step to ensure its reliability, scalability, and performance under demanding conditions. Here are some steps you can take to conduct stress testing for a Bitcoin trading platform:

1. Define Testing Objectives: Clearly define the goals and objectives of the stress testing process. Determine what aspects of the trading platform you want to evaluate, such as transaction processing, system response times, user load capacity, or system stability under heavy loads.

2. Create Test Scenarios: Develop a set of realistic test scenarios that simulate high-stress situations. These scenarios should include a wide range of user activities, such as placing orders, executing trades, withdrawing funds, or handling multiple concurrent users. Consider various stress factors like high transaction volumes, rapid market fluctuations, and sudden spikes in user activity.

3. Identify Key Performance Indicators (KPIs): Determine the performance metrics you will measure during the stress testing process. Some important KPIs for a Bitcoin trading platform may include transaction processing speed, order execution time, latency, system uptime, and user response times. Define acceptable thresholds for each KPI to identify performance bottlenecks.

4. Test Environment Setup: Set up a testing environment that closely resembles the production environment of the trading platform. Ensure that the hardware, software, and network

configurations are similar to the real-world environment. Use realistic data, including historical trading data, to create an accurate testing environment.

5. Test Execution: Execute the stress tests according to the predefined scenarios and test cases. Monitor the system's performance and record key metrics during the test execution. Measure the platform's performance under different user load levels, including normal, peak, and excessive load conditions.

6. Analyze Results: Analyze the collected data and performance metrics to identify any performance bottlenecks or weaknesses in the system. Look for areas where the platform struggles to handle the stress and determine the underlying causes. Identify any scalability issues, performance degradation, or stability problems and prioritize them based on their impact and severity.

7. Optimization and Retesting: Based on the analysis, make necessary improvements and optimizations to address the identified issues. This may involve optimizing code, enhancing database performance, upgrading hardware, or optimizing network configurations. Once the optimizations are implemented, retest the platform to ensure that the changes have effectively resolved the performance bottlenecks.

8. Monitor and Maintain: Even after stress testing, ongoing monitoring and maintenance are crucial to ensure the platform's continuous performance and reliability. Implement monitoring tools to keep track of critical metrics and proactively detect any emerging issues. Regularly review system logs, conduct periodic performance testing, and apply necessary updates to maintain optimal performance.

By following these steps, you can effectively stress test a Bitcoin trading platform and identify areas that require improvement to enhance its performance, scalability, and overall stability.

Chapter 7

Summary, Conclusion and Future Enhancements

Chapter 7: Summary, Conclusion & Future Enhancements

7.1. Project Summary

ACE Trading aims to provide a decentralized and fee-free trading system that ensures secure and transparent transactions. The combination of these features and functionalities will provide a unique and compelling solution for individual investors looking to make informed decisions and achieve their investment goals.

In addition, the platform will utilize blockchain technology to provide secure and transparent transactions and a decentralized, fee-free trading system. The project will cover the design, development, testing, and deployment of the platform, and a maintenance period of 6 months after the deployment to ensure the smooth operation of the platform.

7.2. Achievements and Improvements

User-Friendly Interfaces: Platforms have focused on creating intuitive and user-friendly interfaces to attract a wider audience and make trading more accessible. They have improved the design and layout of their platforms, making it easier for users to navigate through various features and execute trades seamlessly.

Advanced Trading Features: Bitcoin trading platforms have introduced advanced trading features to cater to both novice and experienced traders. This includes stop-loss orders, limit orders, margin trading, and even automated trading bots. These features allow users to execute trades based on predetermined conditions and strategies.

Increased Liquidity: Liquidity is crucial for a smooth trading experience, and many platforms have made efforts to increase liquidity by partnering with liquidity providers or implementing strategies to attract more traders. Higher liquidity ensures that users can buy or sell Bitcoin quickly without significantly impacting the market price.

Mobile Apps: Bitcoin trading platforms have developed mobile applications to allow users to trade on the go. These apps provide real-time market data, trading charts, and the ability to execute trades directly from mobile devices. Mobile apps have increased accessibility and convenience for traders.

Regulatory Compliance: As the cryptocurrency industry has faced increased regulatory scrutiny, trading platforms have made efforts to comply with relevant regulations. They have implemented know-your-customer (KYC) procedures to verify user identities and prevent fraudulent activities, thus ensuring a safer trading environment.

Customer Support: Bitcoin trading platforms have invested in improving their customer support services. They have expanded their support teams, implemented live chat features, and provided comprehensive FAQs and educational resources to help users navigate the platform and address any issues they may encounter.

Integration of Altcoins: Initially, Bitcoin trading platforms primarily focused on Bitcoin trading pairs. However, many platforms have expanded their offerings to include a wide range of altcoins, allowing users to trade various cryptocurrencies against Bitcoin or other digital assets.

Market Analysis Tools: To assist traders in making informed decisions, platforms have integrated advanced market analysis tools. These tools provide real-time charts, technical indicators, historical data, and market insights to help users analyze market trends and make more informed trading strategies.

7.3. Critical Review

User Interface and Experience: Assess the platform's design, layout, and user-friendliness. Is the interface intuitive and easy to navigate? Are the necessary features easily accessible? A well-designed and user-friendly interface can greatly enhance the trading experience.

Security Measures: Examine the platform's security protocols. Does it employ strong encryption, two-factor authentication, and cold storage for user funds? Robust security measures are crucial to protect user assets from potential threats.

Trading Features: Evaluate the range of trading features offered by the platform. Does it provide advanced order types, such as stop-loss and limit orders? Are there options for margin trading or automated trading? A diverse set of trading features can cater to different trading strategies and preferences.

7.4. Lessons Learnt

Use of Larvel

7.5. Future Enhancements/Recommendations

Mobile trading has become increasingly popular, and platforms should continue to invest in improving their mobile apps. This includes optimizing performance, adding new features, and providing a seamless and intuitive mobile trading experience.

Reference and Bibliography

.

Reference and Bibliography

[1] G. Andresen. March 2013 Chain Fork Post-Mortem. BIP 50.

[2] G. Andresen. Pay to Script Hash. BIP 16, 1 2012.

[3] G. Andresen. Blocksize Economics. bitcoinfoundation.org, October 2014.

[4] E. Androulaki, G. O. Karame, M. Roeschlin, T. Scherer, and S. Capkun. Evaluating User Privacy in Bitcoin. In Financial Cryptography, 2013.

[5] M. Andrychowicz, S. Dziembowski, D. Malinowski, and L. Mazurek. Secure Multiparty Computations on Bitcoin. In IEEE Symposium on Security and Privacy, 2014.

[6] M. Andrychowicz, S. Dziembowski, D. Malinowski, and L. Mazurek. On the Malleability of Bitcoin Transactions. In Workshop on Bitcoin Research, 2015.

[7] J. Aspnes, C. Jackson, and A. Krishnamurthy. Exposing computationally-challenged Byzantine impostors. Technical report, Yale, 2005.

[8] M. Babaioff, S. Dobzinski, S. Oren, and A. Zohar. On Bitcoin and Red Balloons. In SIGecom Exchanges, pages 56–73. ACM, 2012.

[9] A. Back, M. Corallo, L. Dashjr, M. Friedenbach, G. Maxwell, A. Miller, A. Poelstra, J. Tim´on, and P. Wuille. Enabling blockchain innovations with pegged sidechains, 2014. [10] A. Back et al. Hashcash-a denial of service counter-measure, 2002.

[11] L. Bahack. Theoretical Bitcoin Attacks with less than Half of the Computational Power (draft). Technical Report abs/1312.7013, CoRR, 2013.

[12] T. Bamert, C. Decker, L. Elsen, R. Wattenhofer, and S. Welten. Have a snack, pay with Bitcoins. In IEEE P2P, 2013.

[13] S. Barber, X. Boyen, E. Shi, , and E. Uzun. Bitter to Better—How to Make Bitcoin a Better Currency. In Financial Cryptography, 2012.

[14] J. Becker, D. Breuker, T. Heide, J. Holler, H. Rauer, and R. B¨ohme. Can We Afford Integrity by Proof-of-Work? Scenarios Inspired by the Bitcoin Currency. In WEIS, 2012.

[15] M. Belenkiy. E-Cash. In Handbook of Financial Cryptography and Security. CRC, 2011. [16] E. Ben-Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza. Zerocash:

Decentralized anonymous payments from Bitcoin. In IEEE Symposium on Security and Privacy, 2014.

[17] E. Ben-Sasson, A. Chiesa, D. Genkin, E. Tromer, and M. Virza. SNARKs for C: Verifying program executions succinctly and in zero knowledge. In CRYPTO, 2013.

[18] E. Ben-Sasson, A. Chiesa, M. Green, E. Tromer, and M. Virza. Secure Sampling of Public Parameters for Succinct Zero Knowledge Proofs. In IEEE Symposium on Security and Privacy, 2015.

[19] I. Bentov and R. Kumaresan. How to Use Bitcoin to Design Fair Protocols. In CRYPTO, 2014.

[20] I. Bentov, C. Lee, A. Mizrahi, and M. Rosenfeld. Proof of Activity: Extending Bitcoin's Proof of Work via Proof of Stake. Cryptology ePrint Archive, Report 2014/452, 2014.

[21] A. Biryukov and I. Pustogarov. Bitcoin over Tor isn't a good idea. In IEEE Symposium on Security and Privacy, 2015. [22] G. Bissias, A. P. Ozisik, B. N. Levine, and M. Liberatore. SybilResistant Mixing for Bitcoin. In WPES'14: Workshop on Privacy in the Electronic Society, 2014.

Index

Index

[A]

[B]

[C]