Experiment No. 1

Name: Jigar Siddhpura
SAP ID: 60004200155
Batch: C22
Date: 23/1/2024

Aim: To identify a suitable life cycle model for your case study and justify your choice

Abstract

It is a type of online marketplace that leverages blockchain technology and decentralised applications to facilitate secure and transparent transactions between buyers and sellers. Unlike traditional ecommerce platforms, web3 ecommerce platforms enable direct peer-to-peer transactions without intermediaries such as banks or payment processors. This is possible through the use of smart contracts, which are self-executing contracts with the terms of the agreement directly written into code. Web3 ecommerce platforms also offer enhanced privacy and security features, such as the ability to encrypt personal and financial data using cryptographic protocols. In addition, they enable buyers and sellers to maintain control over their own data and funds, as opposed to having to rely on centralised third-party entities.

Product:
Our Web3 ecommerce platform is a decentralised online marketplace that leverages blockchain technology and smart contracts to facilitate secure and transparent transactions between buyers and sellers. The platform provides a direct peer-to-peer environment, eliminating intermediaries such as banks or payment processors, and allowing users to maintain control over their own data and funds. The platform also offers enhanced privacy and security features, such as the ability to encrypt personal and financial data using cryptographic protocols.

Users:
Our platform is designed for anyone who wants to buy or sell goods or services online in a secure and decentralised environment. This includes individuals, small businesses, and large corporations looking to reduce transaction fees and increase user control over data and funds.

Features:

- Direct peer-to-peer transactions: The platform enables buyers and sellers to transact directly without intermediaries, reducing fees and increasing user control over data and funds.
- Smart contracts: Self-executing contracts with the terms of the agreement directly written into code, ensuring that transactions are executed securely and transparently.

- Enhanced privacy and security: The platform offers encryption of personal and financial data using cryptographic protocols, ensuring that data is secure and private.
- Lower fees: By eliminating intermediaries, the platform reduces transaction fees for buyers and sellers, making it more cost-effective than traditional ecommerce platforms.
- Faster transactions: The platform leverages blockchain technology to enable faster transactions, reducing the time it takes to complete a transaction and increasing overall efficiency.
- Greater user control: The platform enables users to maintain control over their own data and funds, reducing the reliance on centralised third-party entities and increasing user autonomy.

The scope:

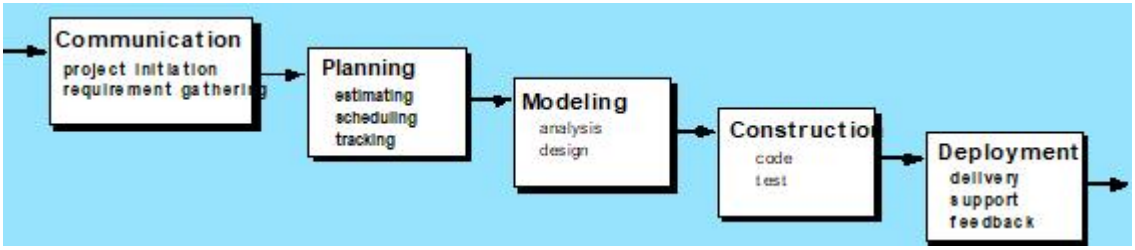
- Requirements gathering: This includes identifying the functional and non-functional requirements of the platform, such as security, scalability, and performance. Some requirements for our project include **integration with blockchain technology, decentralised storage and hosting, support for cryptocurrency payments and transactions, integration of smart contracts (for verifying the authenticity of products, managing inventory, and processing payments)**
- Design: This includes the user interface design, database design, and system design. The design should be aligned with the platform's requirements. Our project must have a **decentralised design, multi currency support, enhanced privacy and security. The system architecture must accommodate blockchain technology, smart contracts.**
- Development: This includes selecting the programming languages, frameworks, and tools that are best suited for the platform. **Programming languages could include Solidity, RUST, Python, and other tools like MetaMask wallets, etc. could be used.**
- Testing: This includes functional, performance, security, and usability testing.
- Deployment: This involves deploying the platform to a production environment, ensuring that it is secure and scalable. **Blockchain Network hosts like Ethereum, Celo, Filecoin could be used.**
- Maintenance: This involves ongoing maintenance of the platform to ensure that it continues to function properly and meets the changing needs of the business.

In addition to these processes, it is important to consider the unique challenges of building a web3 e-commerce platform, such as managing digital assets, ensuring security,

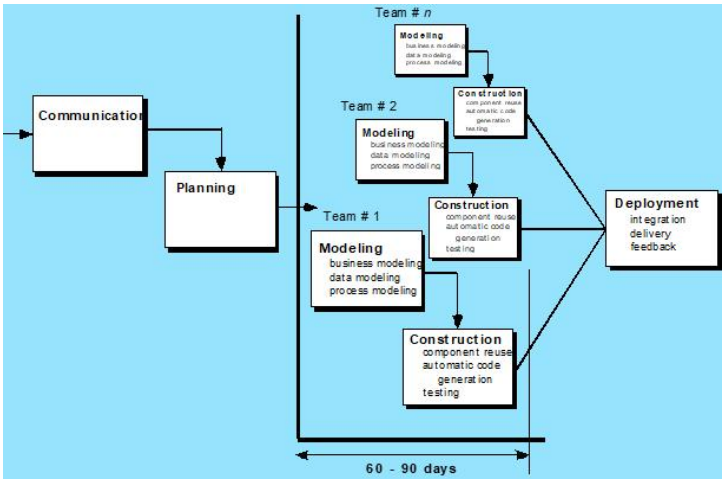
and integrating with blockchain networks. The scope of the software engineering process for a web3 e-commerce platform should take into account these challenges to ensure a successful platform.

Every Life Cycle and best fit

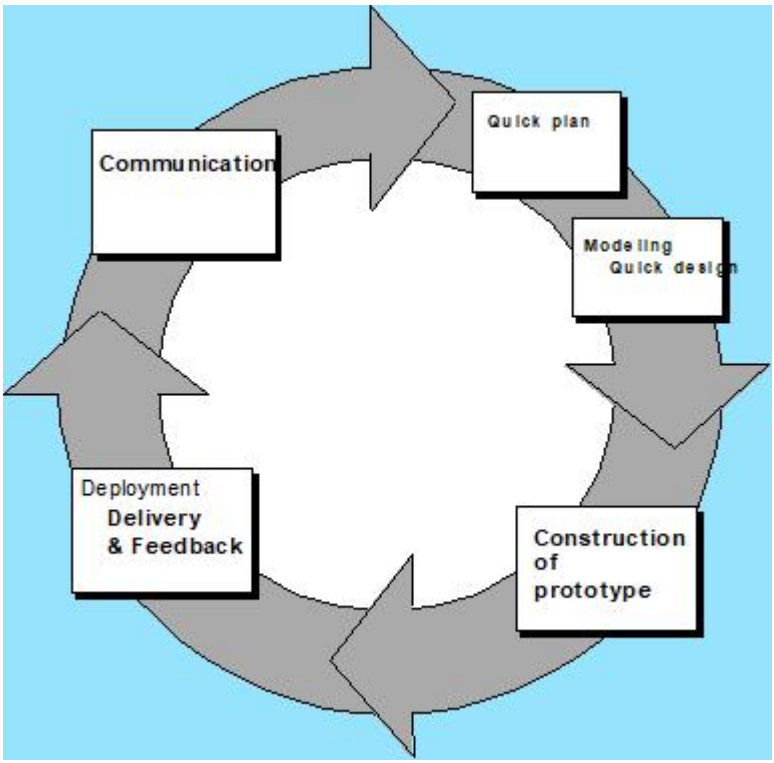
1) **Waterfall Model:** The waterfall model is a linear sequential approach to software development, where each stage of the development cycle follows the previous one. While this model was widely used in the past for its linear and sequential approach, it may not be the best fit for web3 based e-commerce due to its Lack of flexibility. In web3 e-commerce projects, requirements are often subject to change due to the fast-evolving nature of the technology. Higher risk of failure, Delayed testing, and limited customer involvement In a web3 e-commerce project, it is essential to have continuous customer feedback and involvement to ensure that the final product meets their needs and expectations.



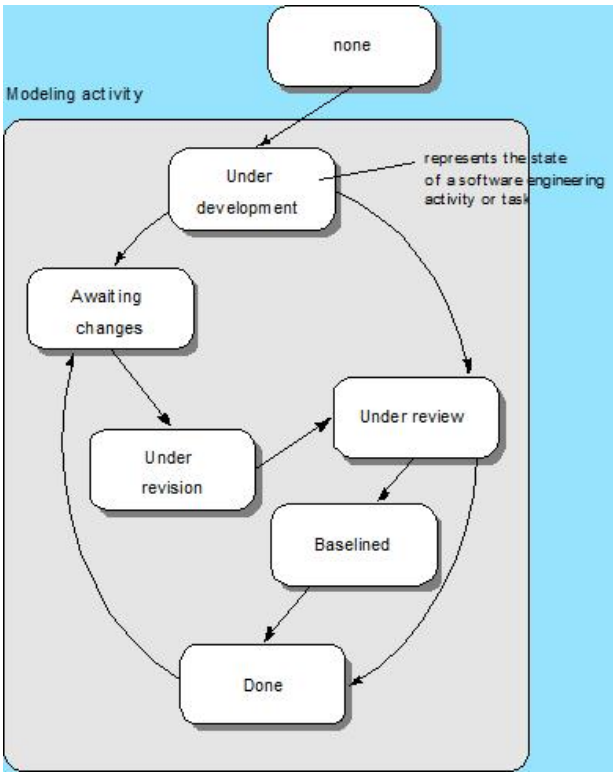
2) **RAD Model:** The Rapid Application Development (RAD) model is a software development approach where the emphasis is on speed and flexibility. The Rapid Application Development (RAD) model prioritises fast prototyping and iterative development, but its disadvantages are Lack of documentation. In web3 e-commerce projects, documentation is essential for security and audit purposes, and the lack of it may create vulnerabilities and compliance issues. Limited testing, Limited scalability, Limited collaboration, . In web3 e-commerce projects, it is essential to have a collaborative and cross-functional team to ensure that the product meets the needs of both customers and the business.



3) **Prototyping Model**: The prototyping model, which involves creating a working model of a system to test and refine its functionality before building the final product, has several disadvantages in the context of web3 e-commerce. Time and Cost, Creating prototypes can be time-consuming and expensive. Limited Scope, Risk of Misinterpretation, Technical Complexity Web3 e-commerce systems can be highly complex, with many different components and technologies involved. Creating accurate prototypes that reflect this complexity can be challenging, and errors or oversights in the prototyping stage can have significant consequences later on.



4) **Concurrent Model**: The concurrent model is a software development approach where multiple stages of the development cycle are executed simultaneously. Disadvantages include performance bottlenecks, difficult to debug, data inconsistency, longer development time

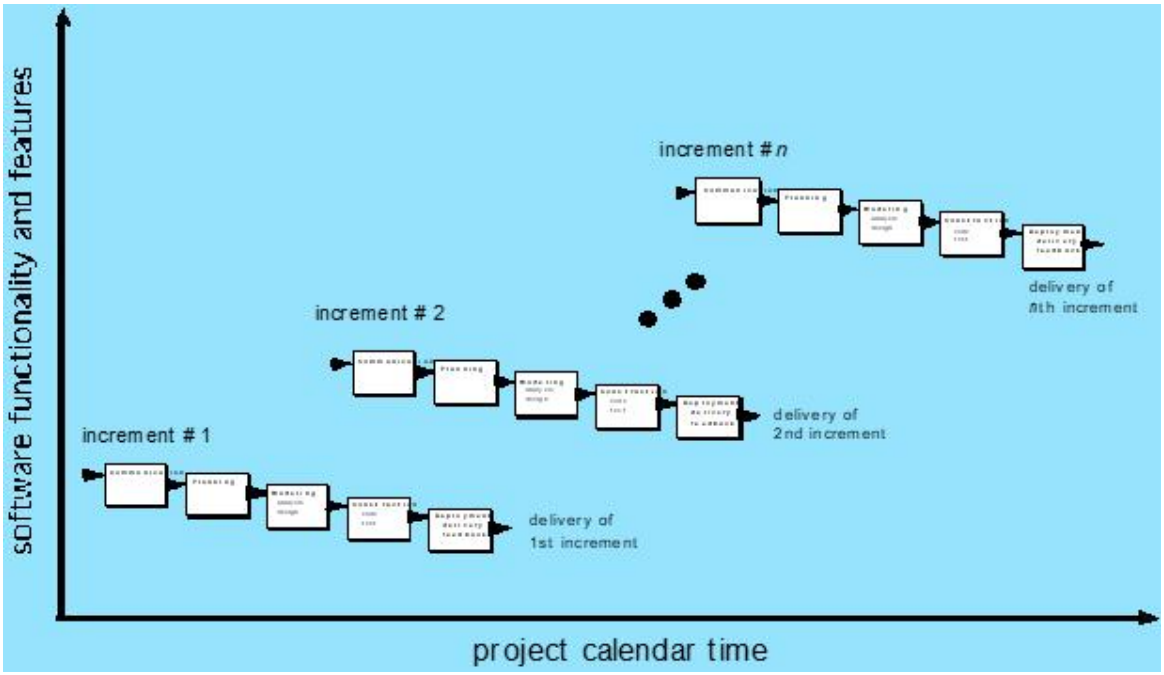


5) **Agile-XP**: Agile XP may not scale well to larger teams or more complex projects.Agile XP requires a high level of collaboration and engagement from all team members, which can be a disadvantage when working with a team that is not experienced with or committed to Agile development practices.Limited documentation:software over comprehensive documentation, which can be a disadvantage for Web3 e-commerce websites that require detailed business rules and security requirements.
6) **Agile-FDD**: FDD requires a high level of collaboration and engagement from all team members, which can be a disadvantage when working with a team that is not experienced with or committed to Agile development practices.FDD prioritises flexibility and adaptability over predictability, which can be a disadvantage for Web3 e-commerce websites that require strict adherence to project timelines and budgets.
7) **Agile-Crystal**: Crystal requires a high level of collaboration and engagement from all team members, which can be a disadvantage when working with a team that is not experienced with or committed to Agile development practices.Crystal focuses on a limited scope of work in each iteration, which can be a disadvantage when developing a Web3 e-commerce website that involves multiple complex components and features.
8) **Agile-Adaptive software development**:ASD may not scale well to larger teams or more complex projects. This can be a disadvantage for Web3 e-commerce websites, which may involve many complex and interconnected components and features.ASD requires a high level of collaboration and engagement from all team

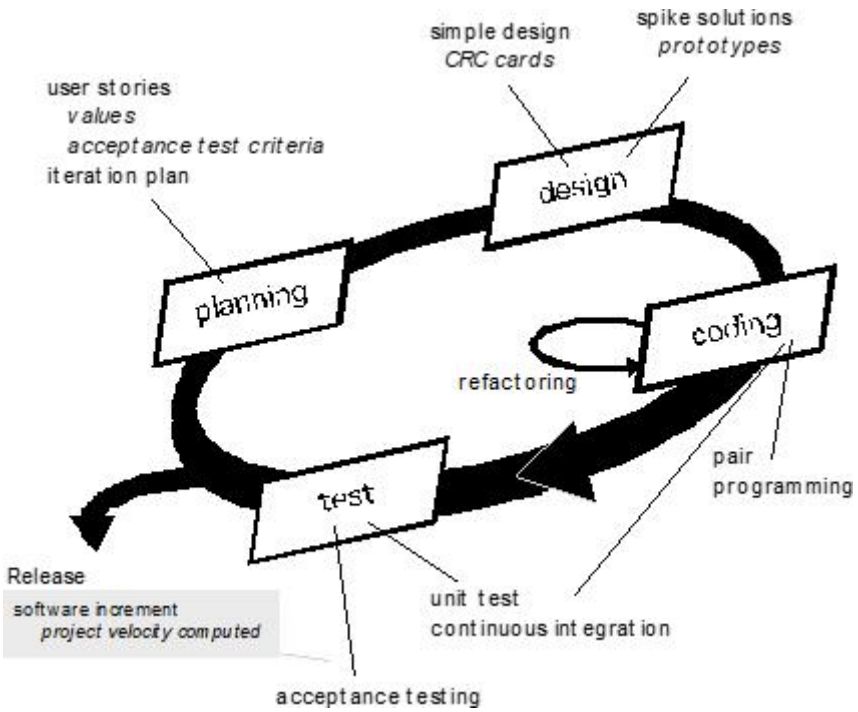
members, which can be a disadvantage when working with a team that is not experienced with or committed to Agile development practices.

Models that we choose and why:
Out of the given life cycle models, the three best models that fit a web3 ecommerce platform are the Incremental Model, the Agile Model - Scrum, and the Spiral Model. Here's why:

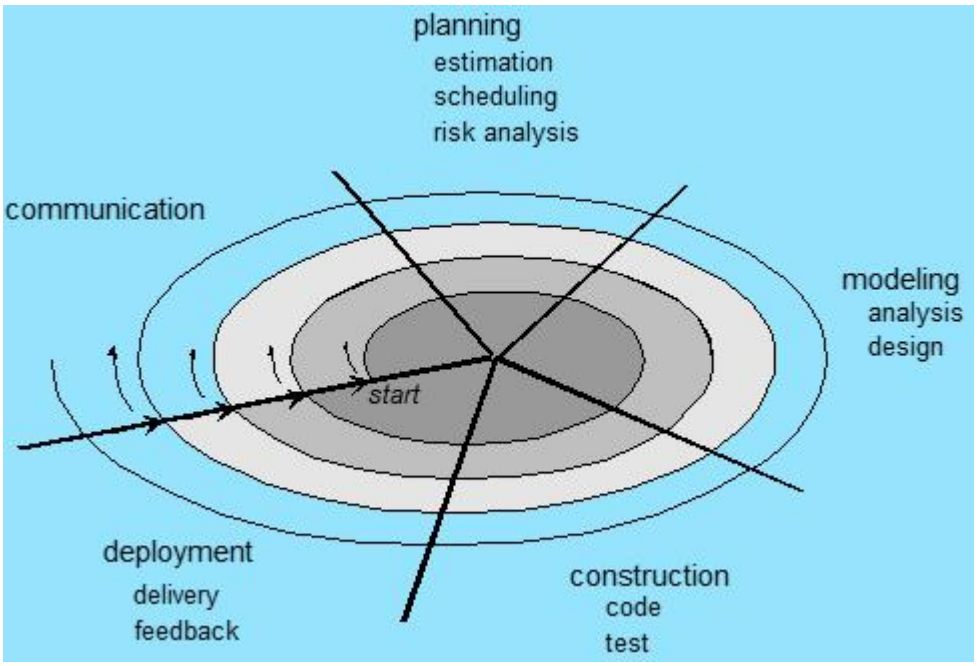
- **Incremental Model**: The Incremental Model is a software development model in which the development process is divided into small, manageable phases or increments. Each increment builds on the previous increment, and the system is developed and tested incrementally. This model is suitable for web3 ecommerce platforms because it allows for iterative development, which means that the platform can be continuously improved and adapted to meet changing user needs and business requirements.Using an incremental model allows for the rapid delivery of new features, which can help keep the website competitive and up-to-date.



- **Agile Model-Scrum**: Scrum emphasises collaboration and communication among team members, which is particularly important in developing a Web3 e-commerce website. This type of website requires expertise in multiple areas, including blockchain development, smart contract programming, and e-commerce functionality. Scrum is suitable for web3 ecommerce platforms because it allows for frequent releases and rapid feedback, which is critical for keeping up with the fast-changing market and customer demands.



- **Spiral Model**: The Spiral Model is a software development model that combines the iterative nature of the Incremental Model with the risk management and prototyping of the Prototyping Model. The Spiral Model is suitable for web3 ecommerce platforms because it allows for early identification and management of risks, which is important in the context of an online platform that deals with sensitive user data and financial transactions.



In summary, the Incremental Model, Agile Model - Scrum, and Spiral Model are the three best life cycle models that fit a web3 ecommerce platform due to their iterative and

flexible nature, ability to manage risks and adapt to changing requirements, and support for frequent releases and rapid feedback.

Since a web3 ecommerce platform involves various technologies, such as blockchain and cryptocurrency, the development process may involve several uncertainties, and requirements may change quickly. The Agile Model is well-suited for such scenarios, as it emphasises flexibility, collaboration, and continuous feedback, allowing the development team to make changes quickly and efficiently.

Conclusion:

In this experiment we have studied various process models and are able to apply suitable process model for our application.

Software Requirements Specification

for

ChainStore

Version 1.0

Prepared by

Group Name: ChainStore

Jigar Siddhpura	**60004210155**	**jsiddhpura2812@gmail.com**
Aman Nambisan	**60004210166**	**aman2003@gmail.com**
Falguni Parmar	**60004220130**	**falguni.p0205@gmail.com**

Instructor: ***Dr. Meera Narvekar***

Course: **Computer Engineerig**

Lab Section: ***L4***

Teaching Assistant: **NA**

Date: **30-01-2024**

1 Introduction

1.1 Document Purpose

This SRS document outlines the software requirements for an ecommerce platform that operates on the web3 blockchain. The current release version of the platform is v1.0. It aims to provide a decentralized and secure online marketplace for buyers and sellers. The scope of this SRS covers the entire platform, including all subsystems and functionalities.

1.2 Product Scope

This Web3 ecommerce platform, Mirrorball, is a decentralised online marketplace that leverages blockchain technology and smart contracts to facilitate secure and transparent transactions between buyers and sellers. The platform provides a direct peer-to-peer environment, eliminating intermediaries such as banks or payment processors, and allowing users to maintain control over their own data and funds. The platform also offers enhanced privacy and security features, such as the ability to encrypt personal and financial data using cryptographic protocols.

Mirrorball will be a decentralized application (dApp) built on blockchain technology. The platform will allow users to buy and sell products using cryptocurrencies and other digital assets. The platform will provide a secure and transparent environment for transactions by leveraging blockchain's immutable ledger technology. The platform will support various features, including a user-friendly interface, product listings, product search, shopping cart, checkout, and order management.

The platform will have two types of users: buyers and sellers. Buyers will be able to browse and search for products, add items to their cart, and place orders. Sellers will be able to list products for sale, manage their inventory, and fulfill orders. The platform will also have an admin panel that will enable the platform owner to manage user accounts, monitor transactions, and perform other administrative tasks.

1.3 Intended Audience and Document Overview

The primary audience for this SRS document includes the development team responsible for building the web3 ecommerce platform. Other stakeholders, such as product owners, project managers, investors, quality assurance testers and regulatory authorities, may also refer to this document to understand the requirements and functionalities of the platform along with the client.

We recommend that readers begin with the Overview section and then proceed to the sections that are most relevant to their role in the development process. Developers may be most interested in the Technical Requirements section, while project managers may find the General Requirements and User Requirements sections most pertinent. Quality assurance professionals may wish to read all sections to ensure comprehensive testing and validation of the platform.

When reading this document, it is recommended to start with the Introduction section to understand the purpose and scope of the document. After that, the Overall Description section provides a broad understanding of the platform and its components. The Specific Requirements section provides detailed requirements for developers and project managers, while the Non-functional Requirements section is pertinent to both technical and business stakeholders. Finally, the Other Requirements section provides any additional information that may be relevant to specific stakeholders.

1.4 Definitions, Acronyms and Abbreviations

1.4.1 API

Application Programming Interface

1.4.2 Blockchain Technology

A system in which a record of transactions, especially those made in a cryptocurrency, is maintained across computers that are linked in a peer-to-peer network.

1.4.3 dApp

Decentralized applications (dApps) are digital applications or programs that exist and run on a blockchain or peer-to-peer (P2P) network of computers instead of a single computer. DApps (also called "dapps") are thus outside the purview and control of a single authority.

1.4.4 Ecommerce

Ecommerce (electronic commerce) refers to all online activity that involves the buying and selling of products and services.

1.4.5 Ethereum

Ethereum is a decentralized blockchain platform that establishes a peer-to-peer network that securely executes and verifies application code, called smart contracts.

1.4.6 HTTPS

Hypertext Transfer Protocol Secure

1.4.7 IPFS

The InterPlanetary File System (IPFS) is a protocol, hypermedia and file sharing peer-to-peer network for storing and sharing data in a distributed file system.

1.4.8 MetaMask

MetaMask is a software cryptocurrency wallet used to interact with the Ethereum blockchain. It allows users to access their Ethereum wallet through a browser extension or mobile app, which can then be used to interact with decentralized applications.

1.4.9 Smart Contracts

Smart contracts are simply programs stored on a blockchain that run when predetermined conditions are met.

1.4.10 Solidity

Solidity is an object-oriented programming language for implementing smart contracts on various blockchain platforms, most notably, Ethereum.

1.4.11 Web3

Web3 (also known as Web 3.0) is an idea for a new iteration of the World Wide Web which incorporates concepts such as decentralization, blockchain technologies, and token-based economics.

1.5 References and Acknowledgments

1. P. Shamili and B. Muruganantham, "Enhancing the Decentralized Application (Dapp) for E-commerce by using the Ethereum Blockchain," in Proceedings of International Conference on Recent Trends in Computing: ICRTC 2021, Singapore: Springer Nature Singapore, Jan. 2022, pp. 679-694.
2. P. SS and A. Kumar, "Web3.0 E-Commerce Decentralized Application," 2022.
3. S. Kulkarni, "Framework for Blockchain Based Decentralized Ecommerce Application Using Smart Contracts," 2020.
4. R. Infante, "Building Ethereum Dapps: decentralized applications on the Ethereum blockchain," Simon and Schuster, 2019.
5. R.S. Kumar, "An overview of the expected influence of web 3.0 on e-commerce and allied domains."

2 Overall Description

2.1 Product Perspective

The web3 ecommerce platform "Mirrorball", is a new, self-contained product designed to provide a secure and decentralized ecommerce experience for both buyers and sellers. Mirrorball is built on top of blockchain technology and utilizes smart contracts to facilitate transactions, ensuring that all parties involved in a transaction can trust the authenticity of the exchange.

Mirrorball is not a replacement for any existing systems but is instead an innovative solution that provides a new way of conducting e-commerce transactions. Its aim is to provide a platform that is secure, transparent, and cost-effective. The platform will enable small businesses and individuals to participate in the global economy without the need for intermediaries.

Mirrorball will allow buyers to purchase goods and services with cryptocurrency, providing a new level of financial privacy and security. Sellers will be able to receive payments in cryptocurrency, which they can then exchange for their local currency or hold as an investment.

2.1.1 System Interface

The platform will be built on top of a blockchain network such as Ethereum or Binance Smart Chain, which will serve as the underlying technology for recording all transactions and interactions. The platform's smart contracts will be written in Solidity, while the frontend will be developed using modern web development frameworks such as React.

2.1.2 User Interface

The new system shall provide a very intuitive and user-friendly interface to the users, where they can easily browse and search for products, view product descriptions, and make purchases using cryptocurrencies. The platform shall also provide a seller dashboard where sellers can easily manage their inventory, view their sales reports, and fulfill orders.

2.1.3 Hardware Interface

a) Server-side

The platform will be hosted on a decentralized network, which means that the hardware requirements will be minimal. The platform's smart contracts will be deployed on the blockchain network, while the frontend will be hosted on IPFS.

b) Client-side

Users will require an internet-connected device with a modern web browser that supports Web3 technologies such as MetaMask or WalletConnect.

2.1.4 Software Interface

a) Server-side

The platform's smart contracts will interact with the blockchain network to record all transactions and interactions. The platform shall also use a decentralized storage solution such as IPFS to store product images and descriptions.

b) Client-side

The platform shall interact with Web3 wallets such as MetaMask or WalletConnect to enable users to make transactions using cryptocurrencies.

2.1.5 Communication Interfaces

The platform shall use the Web3 API to communicate with the blockchain network and Web3 wallets. The platform shall also use HTTPS to encrypt all communications between the client and server.

2.1.6 Memory Constraints

Memory constraints will come into play when the size of the IPFS storage grows to a considerable size. However, since IPFS is a decentralized storage solution, the platform can easily scale up its storage capacity by adding more nodes to the network.

2.1.7 Operations

The platform shall have operations to ensure the security and privacy of all transactions. The platform shall also have backup and recovery procedures in place to protect against data loss or corruption.

2.1.8 Site Adaptation Requirements

The platform shall be designed to be easily adaptable to different languages and currencies, enabling users from different countries to use the platform without any difficulty.

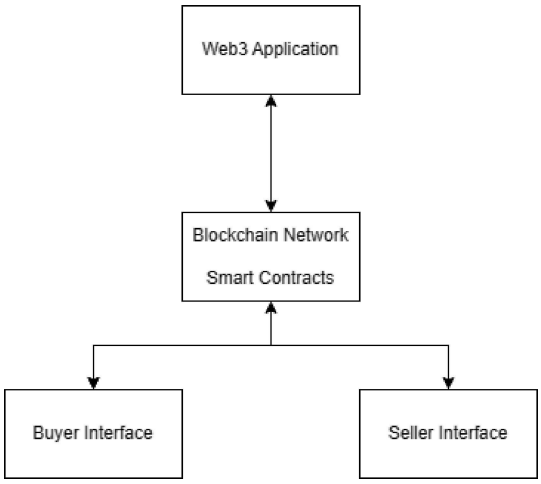


Figure 1: Major components of Mirrorball and how it interacts with the environment

2.2 Product Functionality

2.2.1 Major Functions of the System

Major Functions of the System include:

- User registration and account management
- Product browsing and search functionality
- Shopping cart management
- Order placement and payment processing
- Order tracking and shipment management
- Review and rating system for products
- Seller account management and product listing
- Inventory management for sellers
- Customer support and feedback management
- Marketing and promotional tools for sellers

2.2.2 Data Flow of the Application

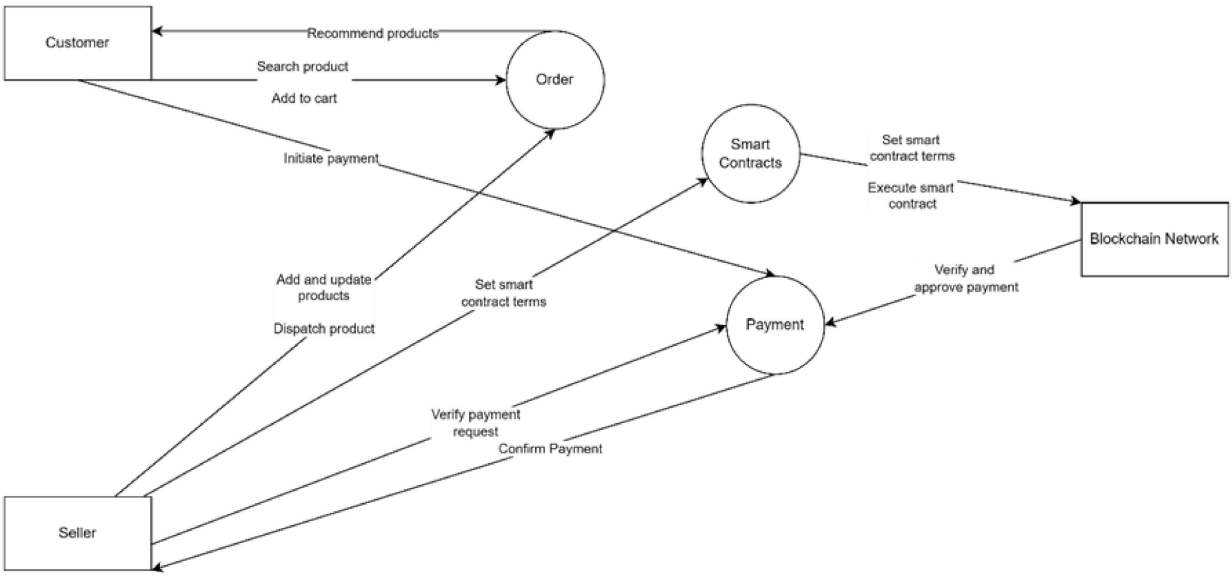


Figure 2: Data Flow Diagram of the Application

2.3 Users and Characteristics

2.3.1 Buyers

Buyers are the primary users of the ecommerce platform. They may have varying levels of technical expertise and educational background, but all share the common goal of purchasing goods or services. They may also differ in terms of their frequency of use, with some using the

platform on a daily basis and others only occasionally. Security is an important concern for buyers, as they want to be assured that their personal and financial information is kept confidential and secure.

2.3.2 Sellers

Sellers are the merchants who use the platform to sell their products or services. They may be individuals or businesses, and may have varying levels of technical expertise. They are responsible for managing their own product listings, inventory, and pricing on the platform. Security and privacy are also important concerns for sellers, as they want to protect their own financial and proprietary information.

2.3.3 Customer Support Representatives

Customer support representatives are responsible for providing assistance to buyers and sellers who encounter issues while using the ecommerce platform. They may have varying levels of technical expertise, but should be skilled in communicating with users in a helpful and professional manner. They may also require access to sensitive user information in order to resolve certain issues, so security and privacy are also important concerns for them.

2.3.4 Technical Support Representatives

Technical support representatives are responsible for providing assistance to users who encounter technical issues while using the platform. They require a high level of technical expertise and familiarity with the platform's underlying technology. They may also be responsible for maintaining the platform's infrastructure and resolving issues related to scalability and reliability.

2.3.5 Administrators

Administrators are responsible for managing the platform and ensuring that it is running smoothly. They may have varying levels of technical expertise, but should be familiar with the platform's underlying technology and able to make changes to its configuration as needed. They are also responsible for ensuring the security and privacy of user data, as well as maintaining compliance with relevant regulations and standards.

2.3.6 Marketing and Sales Personnel

Marketing and sales personnel are responsible for promoting the platform and attracting new users. They may have varying levels of technical expertise, but should be skilled in communicating the platform's value proposition to potential users. They may also be responsible for developing and managing marketing campaigns and partnerships with other businesses or organizations.

2.3.7 Data Analysts

Data analysts are responsible for analyzing user data and providing insights to help improve the platform's performance and user experience. They require a high level of technical expertise

and familiarity with data analysis tools and techniques. They may also be responsible for developing and maintaining the platform's analytics infrastructure.

The most important users for the ecommerce platform are the buyers and sellers, as they are the primary users who generate revenue for the platform. Customer support representatives and technical support representatives are also critical for ensuring a positive user experience and resolving issues that may arise. Administrators are important for ensuring the platform's overall security, compliance, and reliability. While marketing and sales personnel and data analysts are important for promoting and improving the platform, they are generally less critical for the day-to-day functioning of the platform compared to the other user groups.

2.4 Operating Environment

Mirrorball is a decentralized application that aims to provide a secure, transparent and efficient ecommerce solution using blockchain technology. The platform allows buyers and sellers to interact with each other directly, without the need for intermediaries, such as payment gateways or escrow services. The platform will be built using the following technologies:

- Ethereum blockchain
- Solidity smart contract language
- React.js framework

2.4.1 Hardware Platform

Mirrorball is a decentralized application that runs on the Ethereum blockchain. Therefore, it can be accessed from any device that supports an Ethereum wallet, such as MetaMask or MyEtherWallet.

2.4.2 Operating System and Versions

Mirrorball can be accessed from any operating system that supports an Ethereum wallet. This includes but is not limited to:

- Windows (7, 8, 10)
- MacOS (10.13 High Sierra and later)
- Linux (Ubuntu, Debian, Fedora, etc.)

2.4.3 Software Components

Mirrorball relies on several software components and applications to function properly. These include:

- Ethereum network nodes
- Ethereum wallet software (MetaMask, MyEtherWallet, etc.)
- Web3.js library for Ethereum interaction

- React.js framework for frontend development
- Solidity smart contract language for smart contract development
- IPFS (InterPlanetary File System) for file storage and retrieval

2.4.4 Subsystem Interconnections

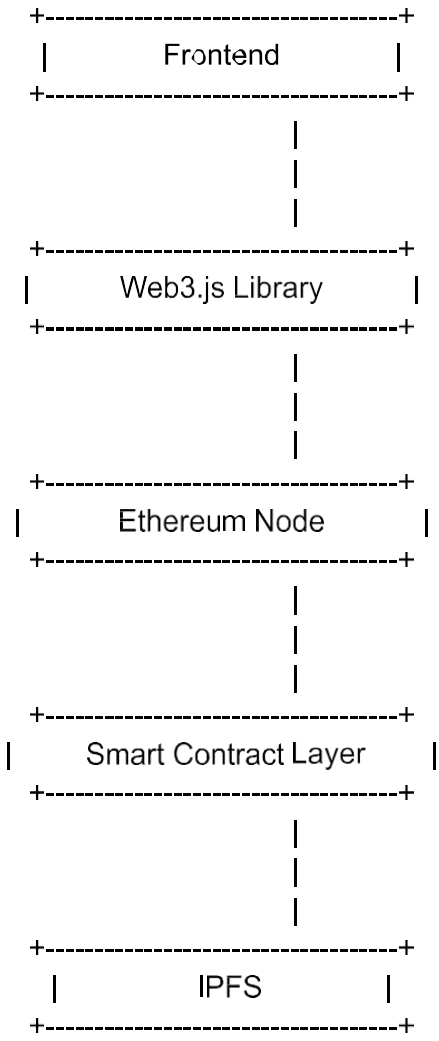


Figure 3: Major Components of Mirrorball and their Interconnections

2.4.5 External Interfaces

The Web3 Ecommerce Platform interacts with several external interfaces, including:

- Ethereum network nodes

- Ethereum wallet software (MetaMask, MyEtherWallet, etc.)
- Payment gateways (PayPal, Stripe, etc.) for fiat currency payments
- Shipping carriers (UPS, FedEx, etc.) for product shipments
- APIs for product information retrieval

Thus, the Web3 Ecommerce Platform operates in a decentralized environment that relies on several software components and applications to function properly. The platform leverages blockchain technology to provide a secure and transparent ecommerce solution that eliminates the need for intermediaries. The platform can be accessed from any device that supports an Ethereum wallet, and it interacts with several external interfaces to provide a seamless ecommerce experience.

2.5 Design and Implementation Constraints

2.5.1 Memory Constraints

Since the platform will be built on a blockchain network, memory constraints will be an important consideration. Smart contract code on the blockchain network must be kept to a minimum to ensure efficient use of resources.

2.5.2 Timing Constraints

The platform will need to ensure that all transactions are processed in a timely manner. The blockchain network's speed will affect how quickly transactions are processed, and the platform will need to take this into account to ensure a smooth user experience.

2.5.3 Security Considerations

The platform will need to ensure that user data and transactions are secure. This will involve implementing security measures such as encryption and authentication, as well as ensuring that smart contracts are secure and free from vulnerabilities.

2.5.4 Communication Protocols

The platform will need to communicate with the blockchain network, as well as with other external systems such as payment processors. The platform will need to use appropriate communication protocols to ensure that data is transmitted securely and reliably.

2.5.5 Language Requirements

Smart contracts will be written in Solidity, and the frontend will be developed using modern web development frameworks such as React. Developers working on the platform will need to be proficient in these languages and frameworks.

2.5.6 Design Conventions/Programming Standards

The platform will need to adhere to design conventions and programming standards to ensure consistency and maintainability. The customer's organization may have their own standards that

the platform will need to adhere to, or the platform may need to follow industry-standard conventions and best practices.

2.5.7 Parallel Operations

The platform will need to handle multiple transactions and interactions simultaneously. The blockchain network's ability to handle parallel operations will be an important consideration when designing the platform.

2.5.8 Third-Party Interfaces

The platform may need to interface with third-party systems such as payment processors or shipping providers. The platform will need to ensure that these interfaces are secure, reliable, and meet the requirements of the external systems.

2.6 User Documentation

For the SRS of a web3 ecommerce platform, the user documentation components that will be delivered along with the software include user manuals, on-line help, and tutorials. The user manuals will provide step-by-step instructions on how to use the ecommerce platform, including how to create an account, browse products, add items to the cart, checkout, and track orders. The on-line help will be available through the platform itself, providing users with quick access to answers to common questions and issues they may encounter while using the platform. The tutorials will provide users with more in-depth information on how to use specific features of the platform, such as how to set up and manage a store or how to use the platform's smart contract functionality. The user documentation will adhere to established standards and delivery formats, such as the W3C's Web Content Accessibility Guidelines (WCAG) and the Portable Document Format (PDF).

2.7 Assumptions and Dependencies

2.7.1 Assumptions

- Users of the platform are familiar with using cryptocurrencies and are comfortable using digital wallets to make purchases.
- The platform will have reliable access to the blockchain network it is built on and that there will be no major technical issues with the network.
- The platform will be used primarily by users with internet-connected devices and modern web browsers that support Web3 technologies.
- The platform will not need to integrate with any legacy systems or platforms.
- Sellers will be able to handle cryptocurrency payments and are comfortable exchanging them for their local currency or holding them as investments.

- The platform will not have to comply with any specific legal or regulatory requirements beyond standard ecommerce regulations.

2.7.2 Dependencies

- The platform will rely on the stability and security of the chosen blockchain network and any third-party software libraries used in the development process.
- The platform will need to integrate with digital wallets such as MetaMask or WalletConnect, and any other third-party payment providers that may be used.
- The platform's frontend will be hosted on IPFS, so the availability and performance of IPFS will be a critical dependency.
- The platform will rely on modern web development frameworks such as React, which will need to be kept up-to-date and maintained to ensure compatibility and security.

3 Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

3.1.1.1 Buyer Interface

The buyer interface will allow buyers to browse products, search for specific items, add items to their cart, and checkout. The interface should have a clean and simple design, with clear navigation and search functionality. Sample screens may include a home page featuring top products, a search results page, a product detail page, and a shopping cart and checkout page. The checkout page will include a crypto payment interface that will allow buyers to pay for their purchases using cryptocurrencies. The crypto payment interface should be designed to be user-friendly, secure, and efficient.

3.1.1.2 Seller Interface

The seller interface will allow sellers to manage their product listings, view sales and revenue data, and interact with customers. The interface should have a dashboard-like design, with an overview of sales data and product performance. Sample screens may include a product listing management page, a sales dashboard, and a customer interaction page. The seller interface should also include a crypto payment interface that will allow sellers to receive payments for their products in cryptocurrencies. The crypto payment interface should be designed to be easy to use and secure.

3.1.1.3 Customer Support Interface

The customer support interface will allow customer support representatives to view and manage support requests from buyers and sellers. The interface should have clear and organized ticket management functionality, with the ability to assign and escalate tickets as needed. Sample screens may include a ticket management dashboard, a ticket detail page, and a customer profile page. The customer support interface should also include a crypto payment interface that will allow customer support representatives to issue refunds or process payments to buyers or sellers using cryptocurrencies.

3.1.1.4 Technical Support Interface

The technical support interface will allow technical support representatives to view and manage technical support requests from buyers and sellers. The interface should have clear and organized ticket management functionality, with the ability to escalate tickets to engineering teams as needed. Sample screens may include a ticket management dashboard, a ticket detail page, and a system configuration page. The technical support interface should also include a crypto payment interface that will allow technical support representatives to issue refunds or process payments to buyers or sellers using cryptocurrencies.

3.1.1.5 Administrator Interface

The administrator interface will allow administrators to manage the overall platform configuration, security, and compliance. The interface should have a hierarchical design, with different levels of access and functionality for different types of administrators. Sample screens may include a system configuration page, a security and compliance management page, and a user management page. The administrator interface should also include a crypto payment

interface that will allow administrators to manage and monitor crypto transactions on the platform.

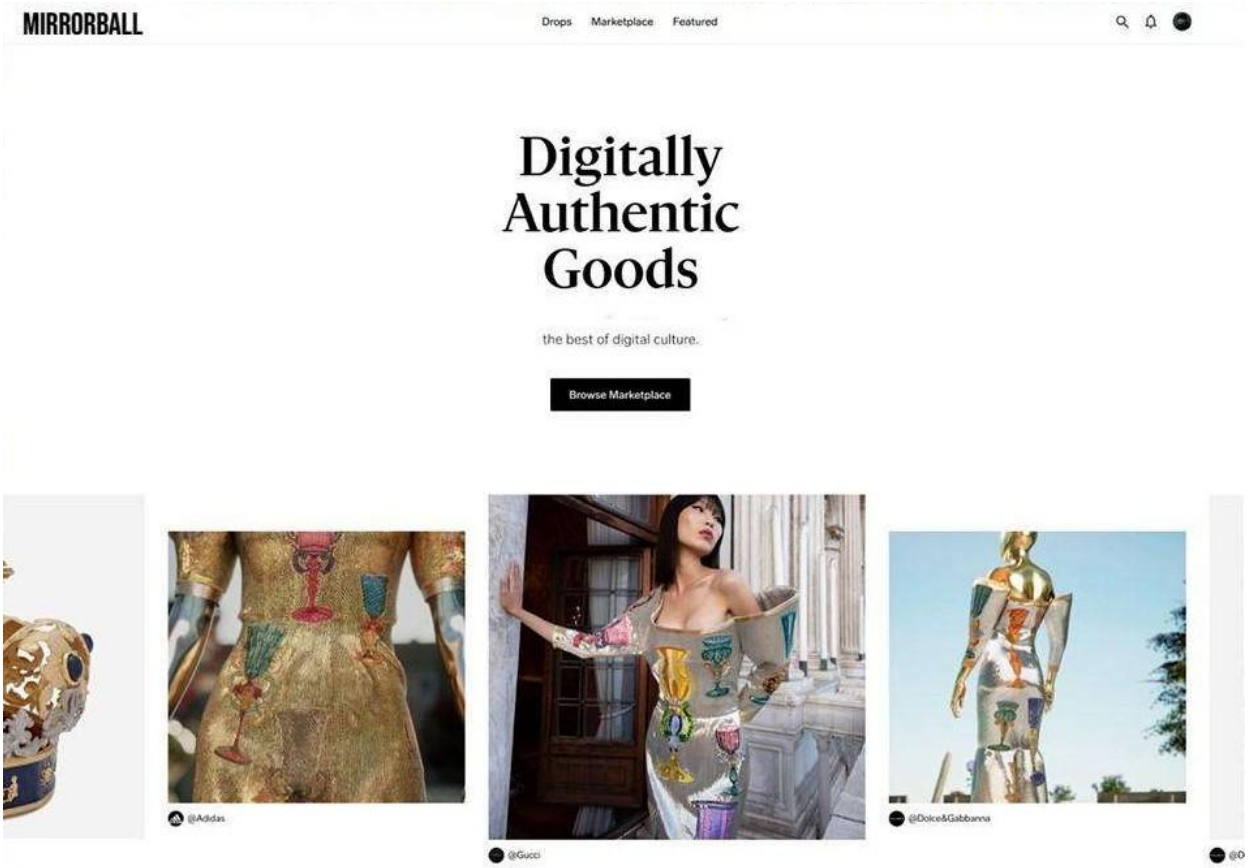


Figure 4.1 Landing Page

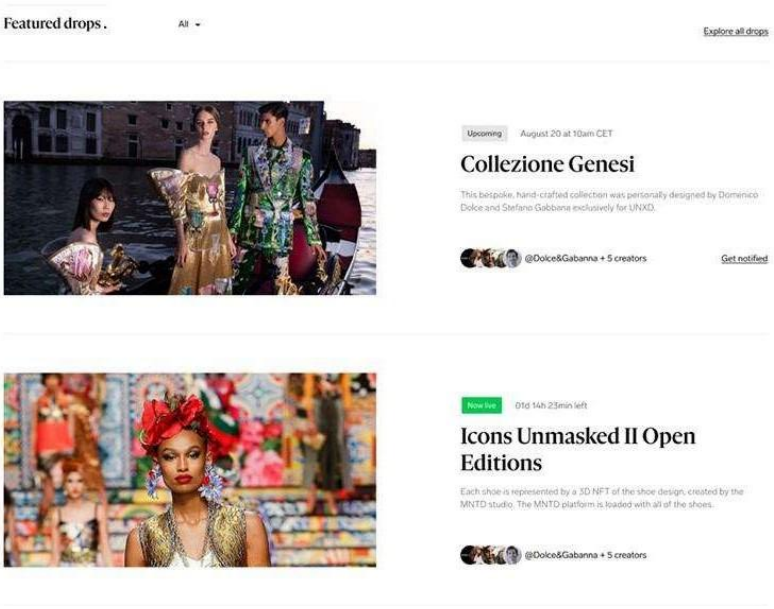


Figure 4.2 Featured Page

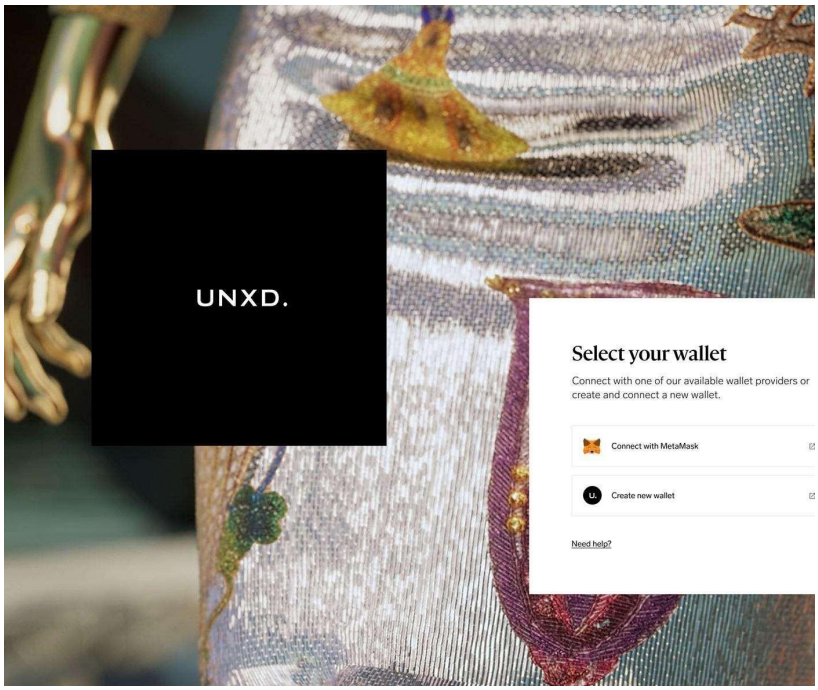


Figure 4.3 Wallet Connection Page

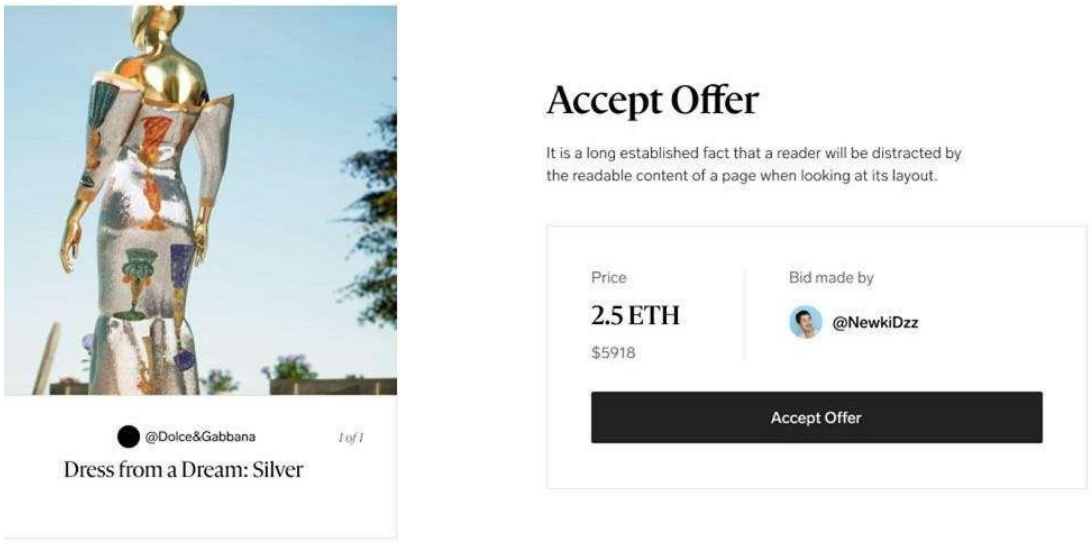


Figure 4.4 View Product Page

3.1.2 Hardware Interfaces

3.1.2.1 Payment gateway interface

This interface allows the web3 ecommerce platform to communicate with the payment gateway to process payments securely and efficiently.

The interface supports various payment methods, including credit cards, debit cards, and cryptocurrencies.

The interface uses secure protocols to ensure the safety of transactions.

The interface integrates with the platform's checkout system to provide a seamless payment experience for users.

3.1.2.2 Cryptocurrency wallet interface

This interface allows the web3 ecommerce platform to communicate with users' cryptocurrency wallets to process payments using digital currencies.

The interface supports various types of cryptocurrencies, including Bitcoin, Ethereum, and Litecoin.

The interface uses secure protocols to ensure the safety of transactions and user data.

The interface integrates with the platform's checkout system to provide a seamless payment experience for users.

3.1.2.3 Barcode scanner interface

This interface allows the web3 ecommerce platform to communicate with barcode scanners to scan product barcodes for inventory management and order fulfillment purposes.

The interface supports various types of barcode scanners, including handheld and stationary devices.
The interface uses libraries that provide easy integration with the platform's inventory management and order fulfillment systems.

3.1.2.4 RFID reader interface

This interface allows the web3 ecommerce platform to communicate with RFID readers to track products and manage inventory.
The interface supports various types of RFID readers, including handheld and stationary devices.
The interface uses libraries that provide easy integration with the platform's inventory management system.

3.1.2.5 Printer interface

This interface allows the web3 ecommerce platform to communicate with printers to generate receipts, shipping labels, and other documents.
The interface supports various types of printers, including thermal and inkjet printers.
The interface uses libraries that provide easy integration with the platform's order fulfillment and shipping systems.

3.1.2.6 Mobile device interface

This interface allows the web3 ecommerce platform to communicate with users' mobile devices to provide a mobile-responsive experience and push notifications.
The interface supports various types of mobile devices, including smartphones and tablets.
The interface uses libraries that provide easy integration with the platform's mobile app and notification systems.

3.1.3 Software Interfaces

The software interface will be designed to work with a web3 ecommerce platform, which will run on a server with an operating system such as Linux or Windows. The interface will be built using programming languages and frameworks such as JavaScript, React, and Node.js.
The interface will communicate with the operating system through system calls and APIs provided by the operating system. For example, to facilitate crypto payments, the interface may need to interact with the operating system's cryptographic libraries to generate and verify digital signatures.
The interface will also interact with databases that store information about products, orders, and users. The database may be hosted on the same server as the web3 ecommerce platform or on a separate server.
In terms of data items or messages coming into the system and going out, the interface will receive requests from the web3 ecommerce platform to initiate crypto payments, and it will send responses back to the platform indicating the status of the payment transaction.
The interface will also need to communicate with external services, such as crypto wallets and payment processors, to facilitate the payment transactions. The nature of these communications will depend on the specific services being used.

Data that will be shared across software components will include information about the products being sold, user account information, and order information. This data may be stored in a database or exchanged between the web3 ecommerce platform and the interface using APIs.

3.1.4 Communications Interfaces

Mirrorball will require various communication interfaces to enable interaction with external entities. These interfaces will include email, web browser, and network server communication protocols such as HTTP and FTP. The platform will also need to support electronic forms to facilitate the entry of user information. To ensure secure communication, the platform will use encryption for data transfer. The specific encryption standards will be determined based on industry best practices and will be regularly reviewed and updated to maintain the highest level of security.

In addition to encryption, the platform will also implement authentication and authorization mechanisms to ensure that only authorized users have access to sensitive information. The platform will use standard communication protocols such as SSL/TLS to establish secure connections between the web server and the client browser. To ensure that data is transferred efficiently, the platform will implement data transfer rates that are consistent with industry standards. The platform will also use synchronization mechanisms to ensure that data is transferred correctly and that any changes made are reflected in real-time across all systems.

3.2 Functional Requirements

3.2.1 User Management

- The platform shall allow users to register for an account with email and password.
- The platform shall provide users with the ability to edit and update their personal information.
- The platform shall allow users to view their order history and track the status of their current orders.
- The platform shall provide users with the ability to leave reviews and ratings for products.

3.2.2 Product Management

- The platform shall allow sellers to add and remove products from the catalog.
- The platform shall allow sellers to update product information, such as price and availability.
- The platform shall allow sellers to manage product categories and subcategories.
- The platform shall allow sellers to search for products using keywords and filters.

3.2.3 Ordering and Payment

- The platform shall allow users to add products to a shopping cart and checkout.
- The platform shall support multiple payment methods, including cryptocurrency payments.

- The platform shall generate a unique address for each cryptocurrency transaction and store the transaction data on the blockchain.
- The platform shall allow users to view the status of their payment transactions and receive confirmation of successful payments.

3.2.3 Blockchain Integration

- The platform shall integrate with a blockchain to securely store transaction data and provide transparency and immutability.
- The platform shall allow administrators to view transaction data on the blockchain and verify the authenticity of payments.
- The platform shall use smart contracts to automate payment processes and ensure the fair exchange of value.

3.2.4 Security and Privacy

- The platform shall implement secure communication protocols to protect user data and prevent unauthorized access.
- The platform shall use encryption to ensure the confidentiality of user data and payment transactions.
- The platform shall implement authentication and authorization mechanisms to control access to sensitive data and functions.
- The platform shall comply with relevant data protection regulations and guidelines, such as GDPR and CCPA.

3.2.5 Analytics and Reporting

- The platform shall provide administrators with insights and analytics on user behavior and sales performance.
- The platform shall generate reports on product popularity, sales trends, and revenue.
- The platform shall allow administrators to export data for further analysis and processing.

3.3 Behaviour Requirements

3.3.1 Use Case View

3.3.1.1 Use case diagram

Figure 5: Use Case Diagram

4 Other Non-functional Requirements

4.1 Performance Requirements

4.1.1 Speed

Users expect the ecommerce application to be fast and responsive. Transactions should be processed quickly, and users should not experience long wait times.

4.1.2 Scalability

The application should be able to handle a large number of users and transactions without experiencing performance issues. This is especially important during peak times such as holidays or flash sales.

4.1.3 Security

Users want to know that their personal and financial information is secure. The application should have robust security features, such as encryption and multi-factor authentication, to prevent hacking and other forms of cyberattacks.

4.1.4 Transparency

Blockchain technology allows for increased transparency in ecommerce transactions. Users should be able to track their purchases and ensure that they are getting what they paid for.

4.1.5 Low fees

Blockchain technology can potentially reduce the fees associated with ecommerce transactions. Users expect to pay reasonable fees for their transactions and do not want to be charged exorbitant amounts.

4.1.6 Reliability

Users expect the ecommerce application to be available and accessible whenever they need it. The application should have high uptime and minimal downtime.

4.1.7 Ease of use

The ecommerce application should be intuitive and easy to navigate. Users should be able to find what they are looking for quickly and easily, and the checkout process should be straightforward and streamlined.

4.2 Safety and Security Requirements

4.2.1 Privacy

Users want their personal and financial information to be kept private and secure. The ecommerce application should have robust privacy features, such as encryption and data protection, to prevent unauthorized access to user information.

4.2.2 Data integrity

Users expect the ecommerce application to be tamper-proof and to maintain data integrity. Blockchain technology provides a secure, decentralized ledger that ensures the accuracy and immutability of data.

4.2.3 Authentication

Users want to know that their transactions and interactions with the ecommerce application are secure. The application should have strong authentication mechanisms, such as multi-factor authentication, to prevent unauthorized access to user accounts.

4.2.4 Fraud prevention

Users expect the ecommerce application to have measures in place to prevent fraudulent activities, such as fake transactions or counterfeit products. The blockchain technology provides an auditable, tamper-proof record of all transactions, which helps prevent fraud.

4.2.5 Compliance

Users want to know that the ecommerce application is compliant with relevant laws and regulations, such as anti-money laundering (AML) and know-your-customer (KYC) regulations. The application should have measures in place to ensure compliance and prevent illegal activities.

4.2.6 Secure payments

Users expect the ecommerce application to have secure payment options, such as cryptocurrency payments, that prevent fraudulent activities and protect their financial information.

4.2.7 Protection against cyberattacks

Users expect the ecommerce application to be protected against cyberattacks, such as hacking and phishing attempts. The application should have strong security features, such as firewalls and intrusion detection systems, to prevent and mitigate cyberattacks.

4.3 Software Quality Attributes

4.3.1 Reliability

Users expect the ecommerce application to be reliable and available whenever they need it. To achieve reliability, the application should be tested thoroughly and have a high uptime percentage. Also, a fault-tolerant architecture should be adopted that ensures that the system remains operational even in the event of hardware or software failures.

4.3.2 Scalability

The application should be able to handle a large number of users and transactions without experiencing performance issues. To achieve scalability, the application architecture should be designed to scale horizontally and vertically, and load testing should be performed to ensure that the system can handle a high volume of traffic.

4.3.3 Maintainability

The ecommerce application should be easy to maintain and update. To achieve maintainability, the application should be well-documented, modular, and have a clear separation of concerns. Also, continuous integration and delivery (CI/CD) practices should be adopted to facilitate the release of updates and new features.

4.3.4 Usability

The ecommerce application should be easy to use and navigate. To achieve usability, the application should have an intuitive user interface, clear navigation, and provide users with helpful feedback. User testing should be performed to ensure that the application is user-friendly and meets the needs of the target audience.

4.3.5 Security

The ecommerce application should be secure and protect user data and transactions. To achieve security, the application should implement best practices for security, such as encryption, multi-factor authentication, and regular security audits. Also, compliance with relevant laws and regulations should be ensured.

4.3.6 Performance

The ecommerce application should perform well and respond quickly to user requests. To achieve performance, the application should be optimized for speed and have a high throughput. Also, caching, indexing, and other performance-enhancing techniques should be used.

4.3.7 Testability

The ecommerce application should be easy to test and debug. To achieve testability, the application should be designed with testing in mind, and automated testing should be used to reduce the time and effort required for testing and debugging. Also, unit testing, integration

testing, and end-to-end testing should be performed to ensure that the application meets the required quality standards.

SHRI VILEPARLE KELAVANI MANDAL'S
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC ACCREDITED with "A" GRADE (CGPA : 3.18)

Software Engineering - Experiment 3

SAP ID: **60004200166 , 60004210155**
Name: **Aman Nambisan, Jigar Siddhpura**
Div: **C**
Batch: **C22**

Aim
Identify scenarios & develop UML Use case and Class Diagram for the project

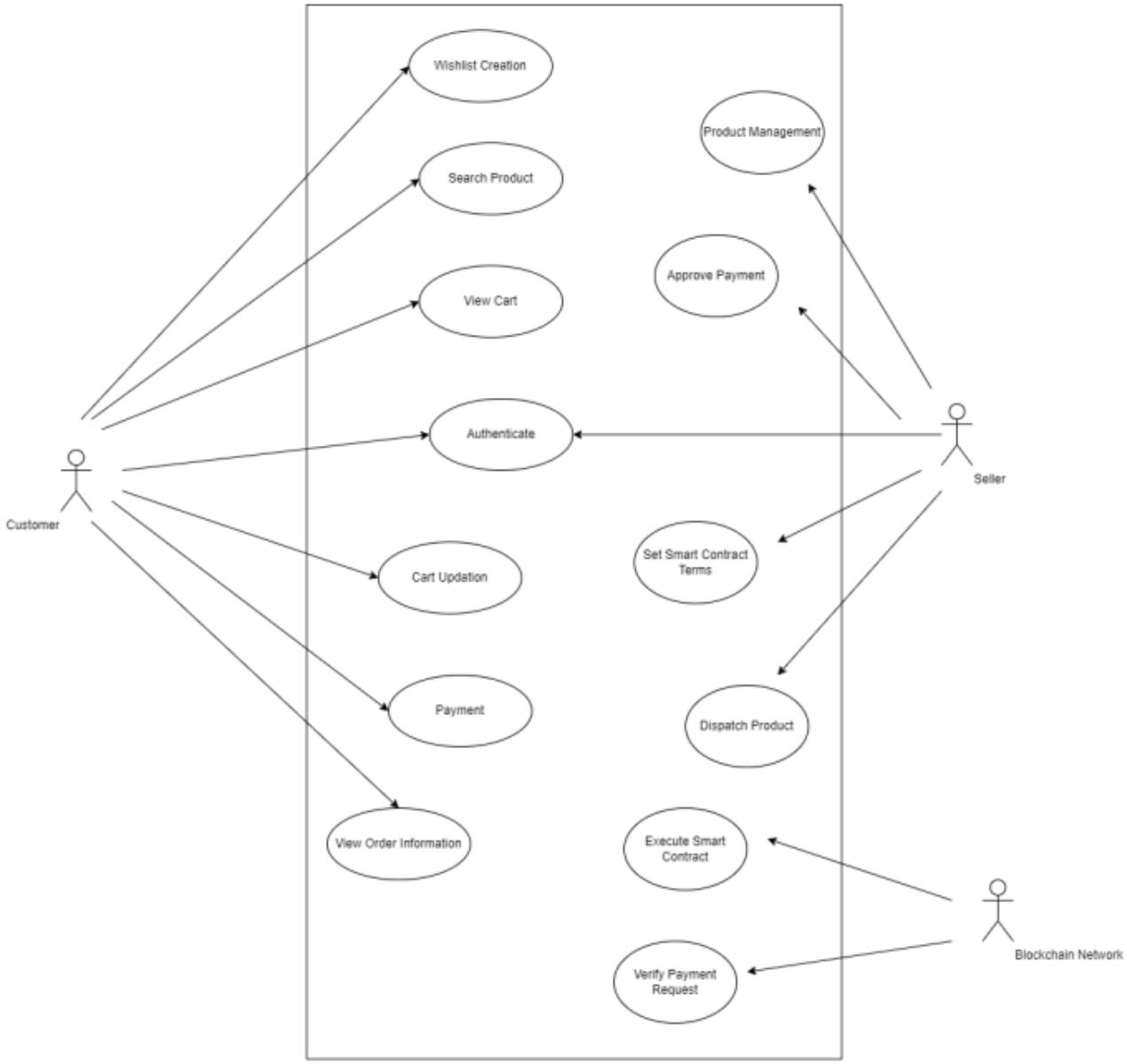
Theory
Use case diagrams and class diagrams are two important tools used in the field of software engineering to model and design software systems.

A use case diagram is a visual representation of the functional requirements of a system. It consists of actors, use cases, and the relationships between them. The actors are the external entities that interact with the system, while the use cases are the actions or services provided by the system. The relationships between them depict how the actors and use cases interact with each other. The use case diagram provides an overview of the system's functionality and can be used to communicate the system's behaviour to stakeholders.

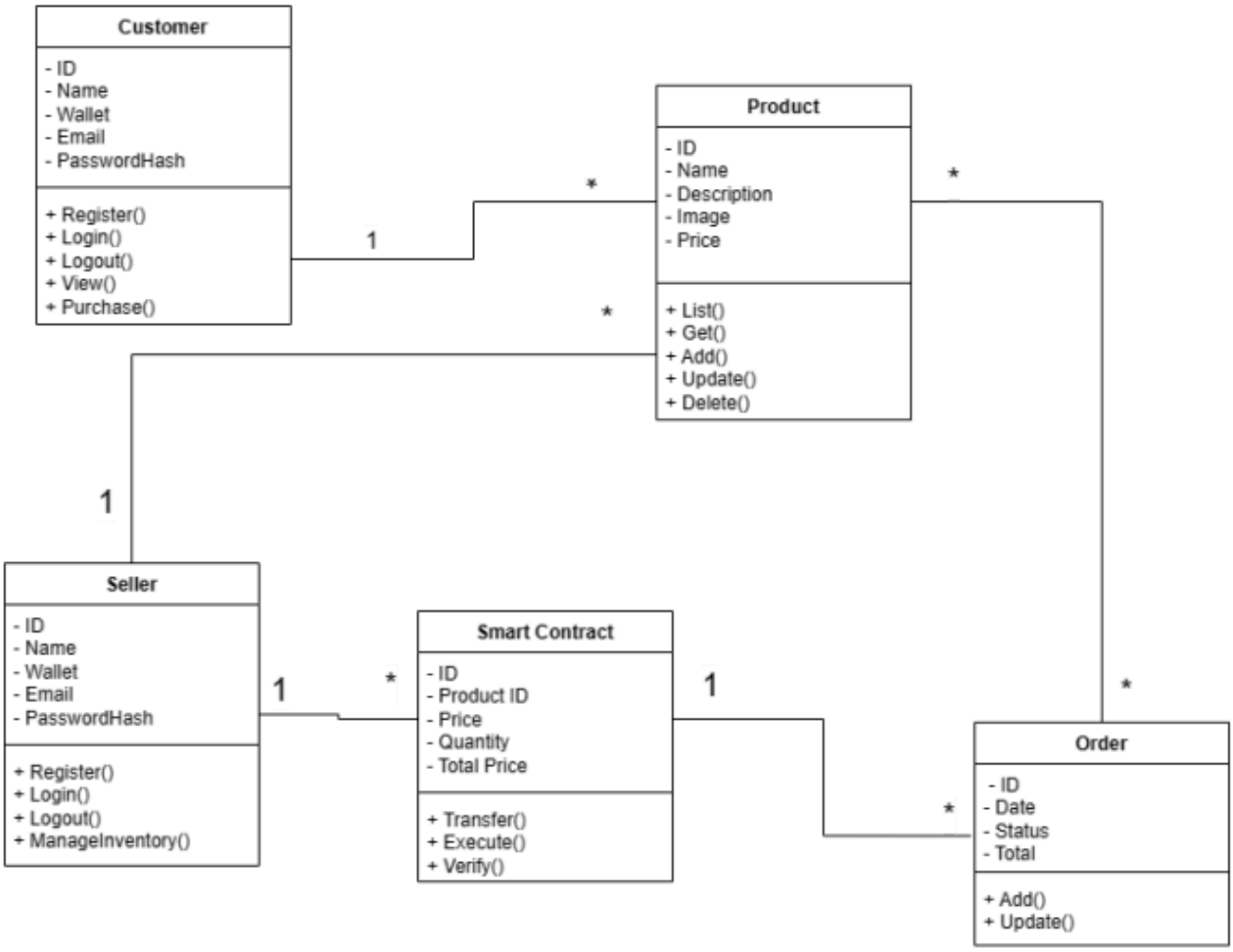
On the other hand, a class diagram is a structural diagram that describes the classes, attributes, and methods of a system. It provides a detailed view of the system's internal structure and is used to model the behaviour of the system's objects. A class diagram consists of classes, relationships between classes, attributes, and methods. Classes represent the objects in the system, and the relationships between them represent the associations, dependencies, and inheritance relationships. Attributes describe the properties of the objects, while methods define the behaviour of the objects.

Together, use case diagrams and class diagrams provide a comprehensive view of a software system's functionality and structure. They help software developers to communicate their design decisions to stakeholders and ensure that everyone has a clear understanding of the system's behaviour and structure. By using these diagrams, software developers can model and design software systems that are robust, reliable, and meet the needs of their users.

Use Case Diagram



Class Diagram



Conclusion

In conclusion, use case diagrams and class diagrams are essential tools for software engineers in the design and development of software systems. Use case diagrams provide a high-level view of a system's functionality and the interactions between the system and its users, while class diagrams provide a detailed view of a system's internal structure, including the classes, attributes, and methods that define its behaviour.

Use case diagrams and class diagrams are complementary tools that help software developers to communicate their design decisions to stakeholders, ensuring that everyone has a clear understanding of the system's behaviour and structure. By using these diagrams, software developers can model and design software systems that are robust, reliable, and meet the needs of their users.

Overall, the use of use case diagrams and class diagrams promotes effective communication, enhances system design and development, and ultimately leads to the creation of high-quality software systems.

SHRI VILEPARLE KELAVANI MANDAL'S
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC ACCREDITED with "A" GRADE (CGPA : 3.18)

Software Engineering - Experiment 4

SAP ID: **60004210155, 60004210166**
Name: Jigar Siddhpura, Aman Nambisan
Div: **C2**
Batch: **C22**

Aim

Develop Activity diagram and DFD (up to 2 levels) for the project.

Theory

Activity Diagram:

A UML activity diagram depicts the dynamic behavior of a system or part of a system through the flow of control between actions that the system performs. It is similar to a flowchart except that an activity diagram can show concurrent flows. The main component of an activity diagram is an action node, represented by a rounded rectangle, which corresponds to a task performed by the software system. Arrows from one action node to another indicate the flow of control. That is, an arrow between two action nodes means that after the first action is complete the second action begins. A solid black dot forms the initial node that indicates the starting point of the activity. A black dot surrounded by a black circle is the final node indicating the end of the activity. A fork represents the separation of activities into two or more concurrent activities. It is drawn as a horizontal black bar with one arrow pointing to it and two or more arrows pointing out from it. Each outgoing arrow represents a flow of control that can be executed concurrently with the flows corresponding to the other outgoing arrows. These concurrent activities can be performed on a computer using different threads or even using different computers.

Solid one-sided arrows represent the control flow or activity flow, showing the sequence and direction of activities.

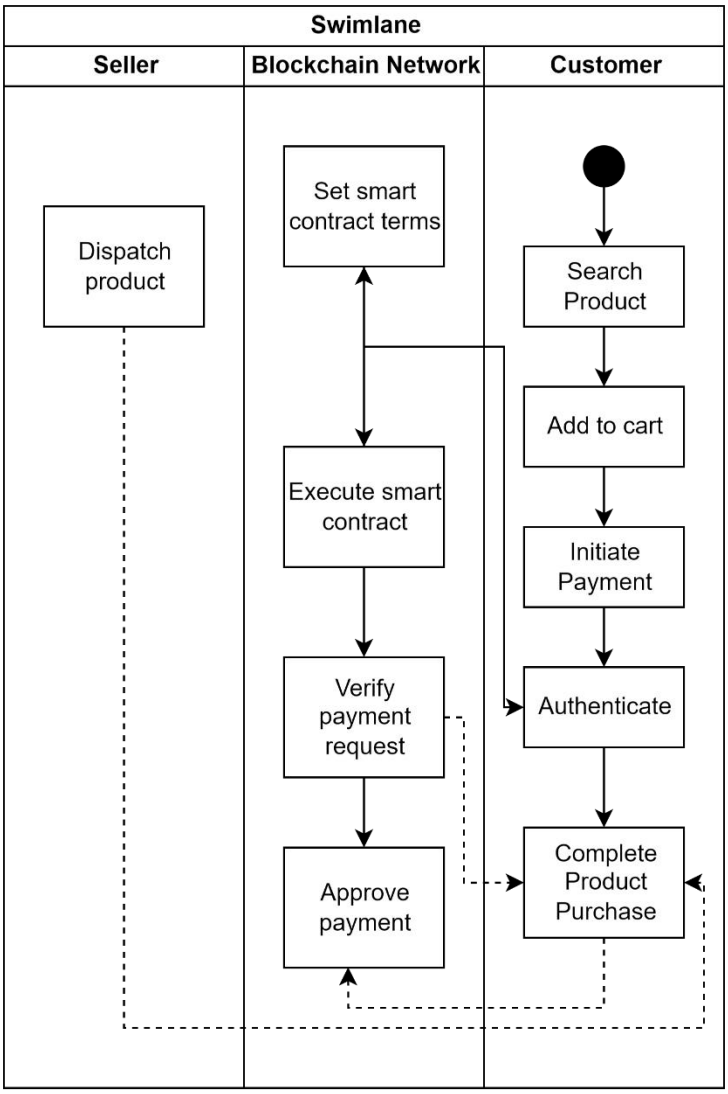
Solid two-sided arrows represent two-way communication or transitions between activities.

Dotted arrows represent the flow of data or objects between activities, indicating the input and output of the activities.

For more notations: https://www.geeksforgeeks.org/unified-modeling-language-uml-activity-diagrams/

An Activity is used to represent the invocation of operation, a step in an entire business process.

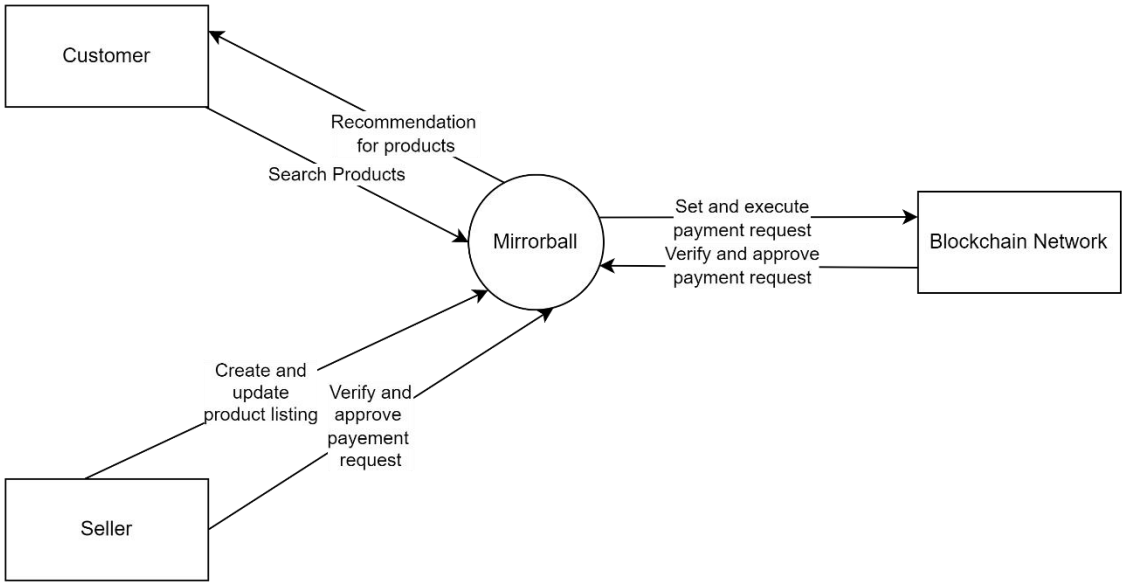
Swimlane is a way in which the performed activities can be grouped by the same actor on an Activity diagram



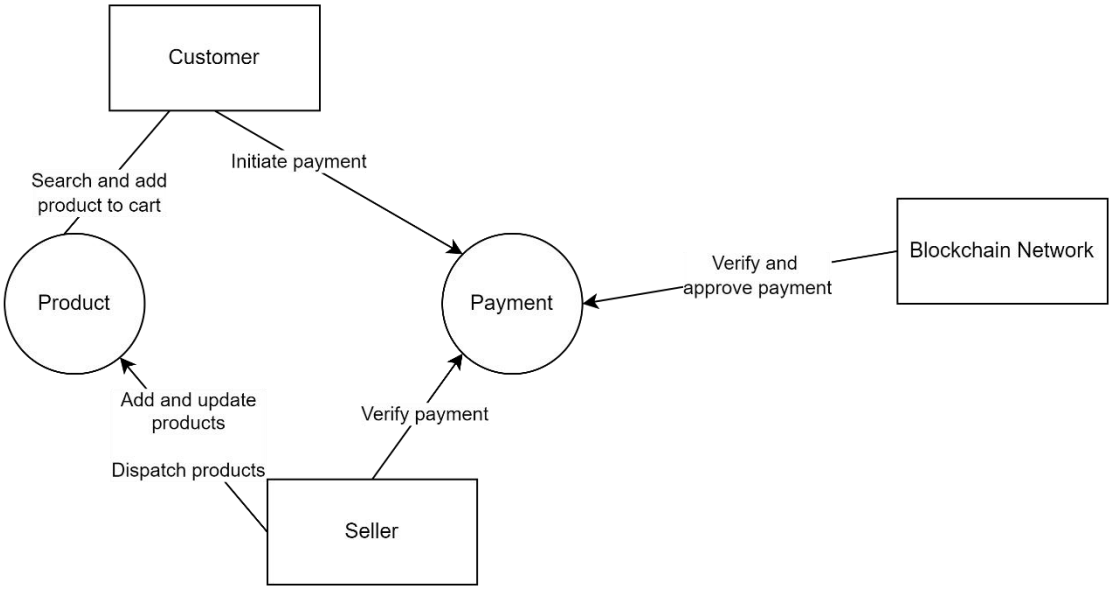
Data Flow Diagrams:

The data flow diagram enables you to develop models of the information domain and functional domain. As the DFD is refined into greater levels of detail, you perform an implicit functional decomposition of the system. At the same time, the DFD refinement results in a corresponding refinement of data as it moves through the processes that embody the application.

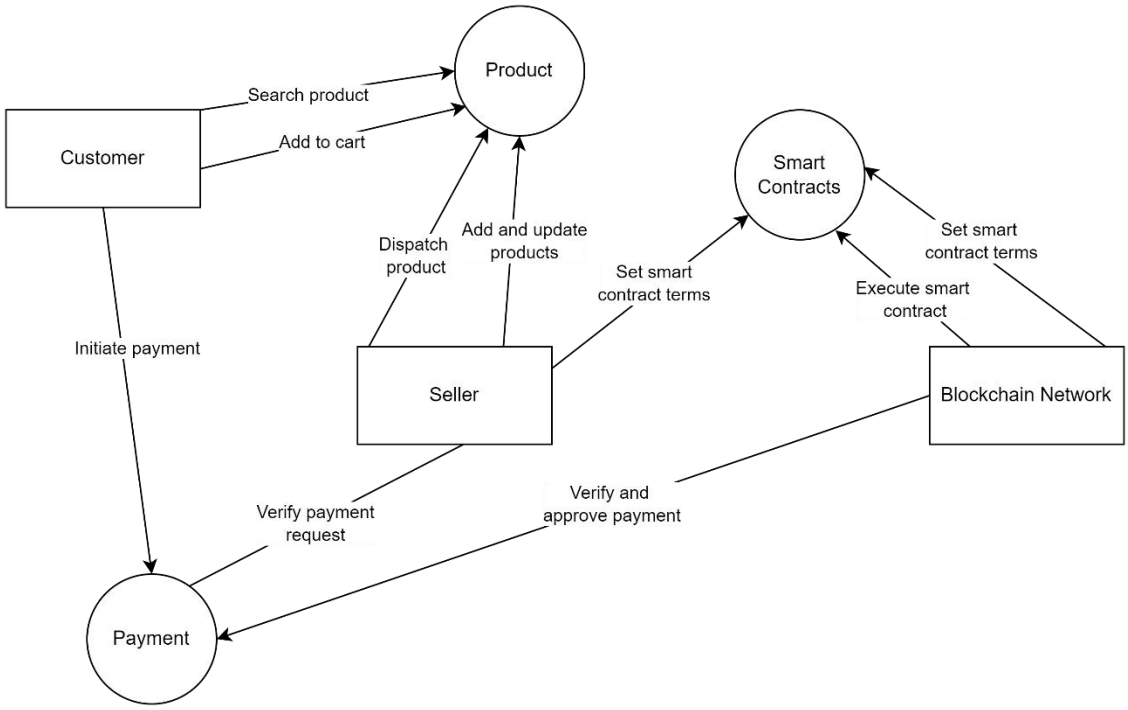
DFD Level 0



DFD Level 1



DFD Level 2



Conclusion

Thus, we are able to draw Activity and Swim lane diagram for our case study. We are also able to depict the flow of data through various processes through different level DFDs.

SHRI VILEPARLE KELAVANI MANDAL'S
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC ACCREDITED with "A" GRADE (CGPA : 3.18)

Software Engineering - Experiment 5

SAP ID: **60004200155, 60004210166**
Name: **Jigar Siddhpura, Aman Nambisan**
Div: **C2**
Batch: **C22**

Aim

Develop Sequence and Collaboration diagram for the project.

Theory

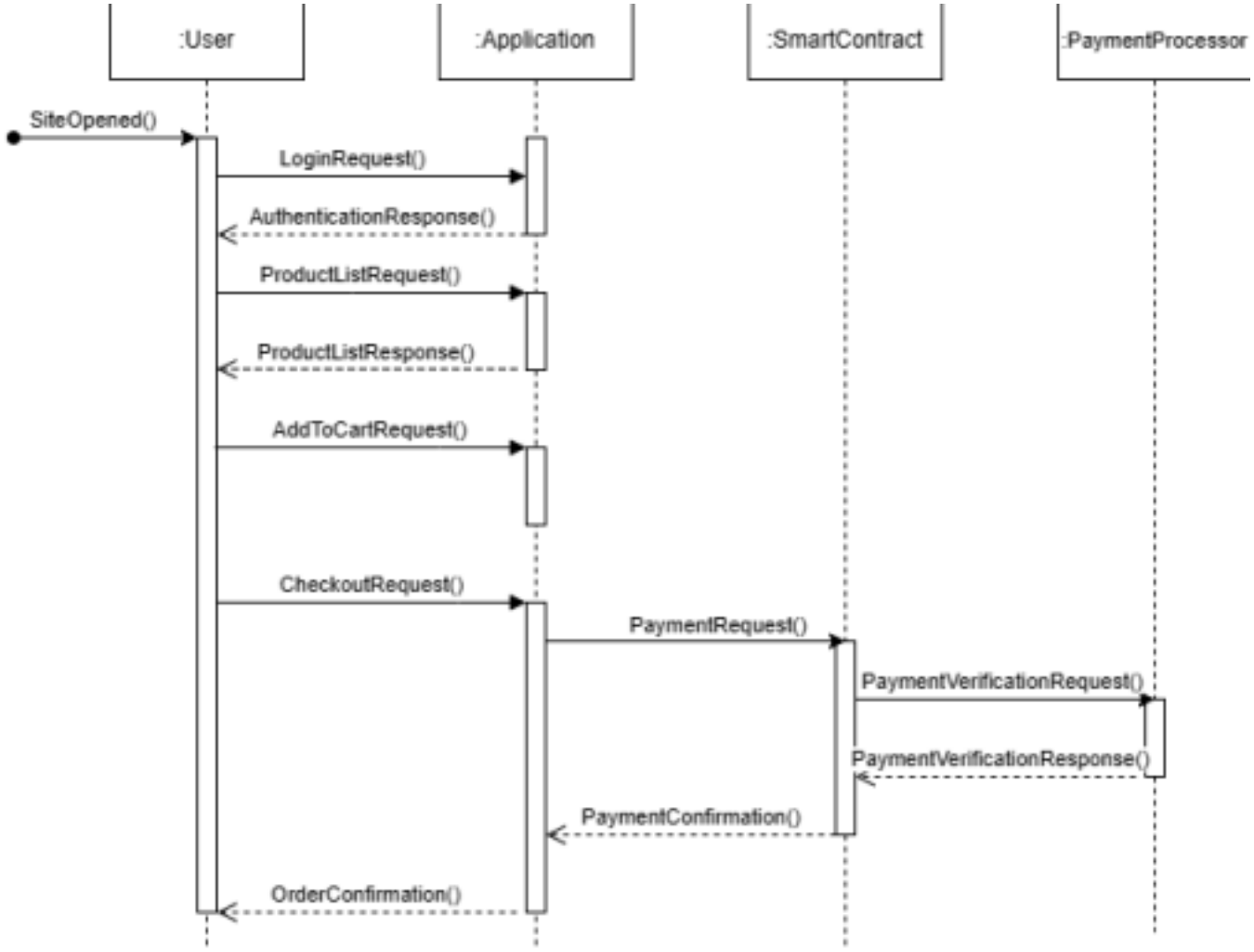
A sequence diagram is used to show the dynamic communications between objects during execution of a task. It shows the temporal order in which messages are sent between the objects to accomplish that task. One might use a sequence diagram to show the interactions in one use case or in one scenario of a software system.

A sequence diagram shows method calls using horizontal arrows from the caller to the callee, labeled with the method name and optionally including its parameters, their types, and the return type. Each box in the row at the top of the diagram usually corresponds to an object, although it is possible to have the boxes model other things, such as classes. If the box represents an object (as is the case in all our examples), then inside the box you can optionally state the type of the object preceded by the colon. You can also precede the colon and type by a name for the object. Below each box there is a dashed line called the lifeline of the object. The vertical axis in the sequence diagram corresponds to time, with time increasing as you move downward. If logical control structures are required, it is probably best to draw a separate sequence diagram for each case. That is, if the message flow can take two different paths depending on a condition, then draw two separate sequence diagrams, one for each possibility

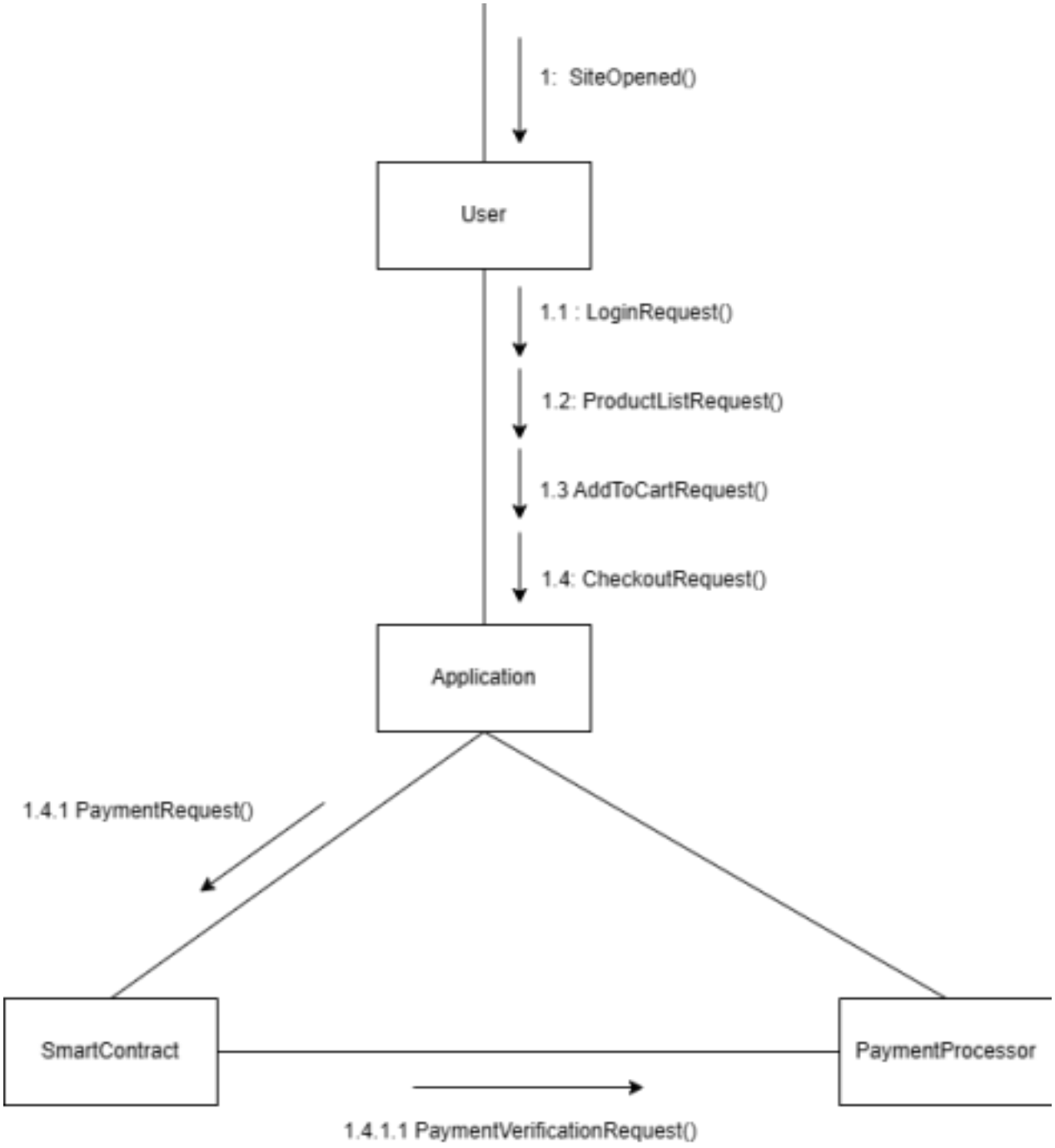
A collaboration diagram displays the same actions shown in the sequence diagram in Figure. In a collaboration diagram the interacting objects are represented by rectangles. Associations between objects are represented by lines connecting the rectangles. There is typically an incoming arrow to one object in the diagram that starts the sequence of message passing. That arrow is labeled with a number and a message name. If the incoming message is labeled with the number 1 and if it causes the receiving object to invoke other messages on other objects, then those messages are represented by arrows from the sender to the receiver along an association line and are given numbers 1.1, 1.2, and so forth, in the order they are called. If those messages in turn invoke other messages,

another decimal point and number are added to the number labeling these messages, to indicate further nesting of the message passing.

Sequence Diagram



Collaboration Diagram



Conclusion

In conclusion, sequence and collaboration diagrams are both powerful tools for modelling and visualising the behaviour of software systems, particularly in the context of object-oriented programming.

Sequence diagrams show the temporal ordering of messages between objects, providing a clear representation of the flow of control and the interactions between objects over time. This makes them useful for modelling complex scenarios involving multiple objects or actors, and for identifying potential issues or bottlenecks in the system.

Collaboration diagrams, on the other hand, show the relationships between objects and their interactions in a static view, allowing developers to see how objects collaborate to accomplish specific tasks. Collaboration diagrams are especially useful for identifying complex relationships between objects, and for visualising how objects interact with one another to implement a particular functionality.

Experiment No. 6

SAP ID: 60004210155, 60004210166

Name: Jigar Siddhpura, Aman Nambisan

Div: C2

Batch: C22

Aim: Estimate effort and cost required using FP/COCOMO for the project. Create WBS and Gantt Chart for the same. Use a PM Tool to depict a project plan.

Theory:

Work Breakdown Structure:

Work Breakdown Statement

A work breakdown statement (WBS) is a categorized list of tasks with an estimate of resources required to complete the task. An example WBS appears below.

WBS #	Task Description	Est Person -Hrs	Who	Resources	M&S
5	Profile motor power				
5.1	Design test stand	20	SE, JM	Pro/E	
5.2	Build test stand	15	SE, JM	Frame & brake parts	$35
5.3	Test 3 motors	3	SE, JM	Stroboscope	$75

5.4	Plot torque vs. speed	2	JM	Excel	

(M&S = Materials & Supplies)

Gantt Chart Basics

Gantt charts are a project planning tool that can be used to represent the timing of tasks required to complete a project. Because Gantt charts are simple to understand and easy to construct, they are used by most project managers for all but the most complex projects.
In a Gantt chart, each task takes up one row. Dates run along the top in increments of days, weeks or months, depending on the total length of the project. The expected time for each task is represented by a horizontal bar whose left end marks the expected beginning of the task and whose right end marks the expected completion date. Tasks may run sequentially, in parallel or overlapping.

As the project progresses, the chart is updated by filling in the bars to a length proportional to the fraction of work that has been accomplished on the task. This way, one can get a quick reading of project progress by drawing a vertical line through the chart at the current date. Completed tasks lie to the left of the line and are completely filled in. Current tasks cross the line and are behind schedule if their filled-in section is to the left of the line and ahead of schedule if the filled-in section stops to the right of the line. Future tasks lie completely to the right of the line.
In constructing a Gantt chart, keep the tasks to a manageable number (no more than 15 or 20) so that the chart fits on a single page. More complex projects may require subordinate charts which detail the timing of all the subtasks which make up one of the main tasks. For team projects, it often helps to have an additional column containing numbers or initials which identify who on the team is responsible for the task.

Often the project has important events which you would like to appear on the project timeline, but which are not tasks. For example, you may wish to highlight when a prototype is complete or the date of a design review. You enter these on a Gantt chart as "milestone" events and mark them with a special symbol, often an upside-down triangle.

Practical:

1) For Estimation:

FP ESTIMATION :

External inputs: Product details, Order placement, Payment details
External Inquiry: Order status, Product availability, Product price comparison
Internal Logical Files: Product database, Order database, User account details
External Outputs: Order confirmation, Shipping details
External interface files: Blockchain transactions

Information Domain Value	Count	Simple	Average	Complex	Total
External inputs	3	3	4	6	3*6=18
External enquiry	3	4	5	7	3*7=21
Internal Logical Files	3	3	4	6	3*6=18
External Outputs	2	7	10	15	2*15=30
External interface files	1	5	7	10	1*10=10
Total					97

Total-Count = 97

Value Adjustment Factors:

Does the system require reliable backup and recovery?

3 - The system requires reliable backup and recovery as it deals with important transactions and user data.

Are specialized data communications required to transfer information to or from the application?

4 - Specialized data communications are required as the platform utilizes blockchain technology for secure transactions.

Are there distributed processing functions?

3 - The platform may require distributed processing functions to handle high volume transactions.

Is performance critical?

5 - Performance is critical as the platform needs to handle real-time transactions and user interactions.

Will the system run in an existing, heavily utilized operational environment?

0 - The platform is a new product with no existing operational environments.

Does the system require online data entry?

4 - The platform requires online data entry for various transactions and user interactions.

Does the online data entry require the input transaction to be built over multiple screens or operations?

3 - Some functionalities may require input transactions over multiple screens.

Are the ILFs updated online?

5 - The ILFs are updated online as the platform deals with real-time transactions.

Are the inputs, outputs, files, or inquiries complex?

2 - Most of the inputs, outputs, files, and inquiries are simple.

Is the internal processing complex?

2 - The internal processing is not complex as the platform focuses on user interactions and transactions.

Is the code designed to be reusable?

3 - The code can be reused over multiple functionalities with minor modifications.

Are conversion and installation included in the design?

3 - Conversion and installation are included in the design.

Is the system designed for multiple installations in different organizations?

4 - The system is designed to be easily installed in different organizations.

Is the application designed to facilitate change and ease of use by the user?

5 - The application is designed to be user-friendly and easy to use.

Hence Σ (Fi) = 46
The estimated number of FP is derived:

FP estimated = count-total x [0.65 + 0.01 x Σ (Fi)]

= 97* [0.65 + 0.01 x 46]

= 107.67

Therefore, FP estimated is **107.67** pm

For WBS and Gantt Chart

WBS:

Level 1: Project Management

- **1.1 Project Initiation**
- **1.2 Project Planning**
- **1.3 Project Execution**
- **1.4 Project Monitoring and Control**
- **1.5 Project Closure**

Level 2: Requirements Analysis and Design

- **2.1 Define Business Requirements**
- **2.2 Analyze User Requirements**
- **2.3 Develop System Architecture**
- **2.4 Design User Interface**
- **2.5 Design Database Schema**

Level 3: Frontend Development

- **3.1 Develop Home Page**
- **3.2 Develop Category Pages**
- **3.3 Develop Product Pages**
- **3.4 Develop Cart and Checkout Pages**
- **3.5 Develop User Account Pages**
- **3.6 Develop Help and Support Pages**
- **3.7 Develop Marketing Pages**

Level 4: Backend Development

- **4.1 Develop User Management System**
- **4.2 Develop Payment Gateway Integration**
- **4.3 Develop Order Management System**
- **4.4 Develop Inventory Management System**
- **4.5 Develop Shipping and Delivery System**
- **4.6 Develop Customer Support System**
- **4.7 Develop Reporting and Analytics System**

Level 5: Blockchain Integration

- **5.1 Determine Appropriate Blockchain Platform**
- **5.2 Integrate Blockchain with Payment Gateway**
- **5.3 Integrate Blockchain with Order Management System**
- **5.4 Integrate Blockchain with Inventory Management System**
- **5.5 Integrate Blockchain with Shipping and Delivery System**

Level 6: Testing and Quality Assurance

- **6.1 Develop Test Plan**
- **6.2 Perform Unit Testing**
- **6.3 Perform Integration Testing**
- **6.4 Perform System Testing**
- **6.5 Perform Acceptance Testing**
- **6.6 Perform Security Testing**

Level 7: Deployment and Maintenance

- **7.1 Deploy to Production Environment**
- **7.2 Monitor and Maintain Production Environment**
- **7.3 Upgrade System as Necessary**
- **7.4 Provide User Support and Help Desk Services**

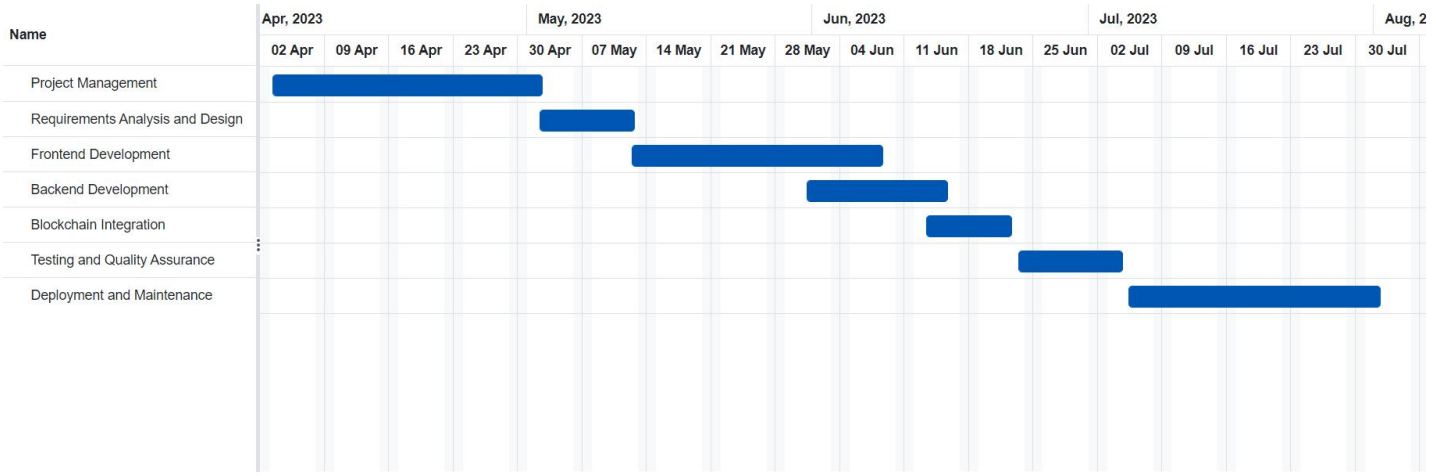
Team members and tasks:

Project Manager: oversees the entire project and ensures timely delivery
Business Analyst: responsible for requirements gathering and analysis
UI/UX Designer: designs the platform's user interface and user experience
Frontend Developer: develops the frontend of the platform using HTML, CSS, JavaScript and ReactJS
Backend Developer: develops the backend of the platform using a web framework such as Node.js or Django
Blockchain Developer: integrates Web3 features into the platform such as blockchain payments, decentralized storage, and smart contracts
Quality Assurance Engineer: tests the platform for functionality, performance, and security
Deployment Engineer: deploys the platform to a test environment and then to production
Support Engineer: provides maintenance and support for the platform

Estimated effort:

Based on the estimated effort of 107.67 person months, we can distribute the effort as follows:
40% of the effort (43.068 person months) can be allocated to analysis, design, and planning
20% of the effort (21.534 person months) can be allocated to coding and implementation
40% of the effort (43.068 person months) can be allocated to testing, deployment, and maintenance.

GANTT TIMELINE CHART:



ID	Name	Duration	Start Date	End Date
1	Project Management	22 days	Apr 03, 2023	May 02, 2023
2	Requirements Analysis and Design	9 days	May 02, 2023	May 12, 2023
3	Frontend Development	20 days	May 12, 2023	Jun 08, 2023
4	Backend Development	12 days	May 31, 2023	Jun 15, 2023
5	Blockchain Integration	8 days	Jun 13, 2023	Jun 22, 2023
6	Testing and Quality Assurance	8 days	Jun 23, 2023	Jul 04, 2023
7	Deployment and Maintenance	20 days	Jul 05, 2023	Aug 01, 2023

Conclusion:

Thus, we are able to estimate the effort required for our project and also create a Gantt Chart.

Software Engineering Experiment 7
Names: Jigar Siddhpura, Aman Nambisan, Falguni Parmar
SAP IDs: 60004210155, 60004210166 60004220130

Project Name	**MirrorBall**
Reference Document	**Project Functional Requirement Specification , Version 1**
Created by	**Jigar Siddhpura, Aman Nambisan, Falguni Parmar**
Date of creation	**26-Mar-24**
Date of review	**01-Apr-24**

Test scenario ID	Requirement-reference document index	Test scenario description	Impor-tance	No. of test cases
TS_01	3.2.1	validate if the user is able register and login using email and password and also updating their personal details	High	
TS_02	3.2.1	Validate if user is able to view and add products to cart in addition to being able to view their order history and tracking status of current orders.	High	
TS_03	3.2.2	Validate if seller is able to add, update and remove products from his/her catalogue	High	
TS_04	3.2.3	Validate if admins are able to view the transactions	High	
TS_05	3.2.5	Validate if admins can view the analytics and generate reports	Medium	

Project Name	**Mirrorball**
Reference Document	**Project Functional Requirement Specification , Version 1**
Created by	Jigar Siddhpura, Aman Nambisan, Falguni Parmar
Date of creation	26-03-2024
Date of review	01-04-2024

Test case ID	Test Objective	Precondition	Steps:	Test data	Expected result	Post-condition
TC_01_1	Successful customer login to the Mirrorball website	1. A valid Mirrorball User account for login to be available 2. Mirrorball site is launched on a compatible browser	1. In the login Panel, enter the username	"A valid username"	The user is logged in successlfully.	For first time users, personal information is displayed.
			2. Enter the Password for the User account in the password field	"A valid Password"		
			3. Click "Login" button			
TC_01_2	Error message on unsuccessful customer login to Mirrorball website	1. A valid Mirrorball User name for login to be available 2. Mirrorball site is launched on a compatible browser	1. In the login Panel, enter the username	"A valid username"	The user is not logged in to the Mirrorball website and an error message is displayed - "Your password is incorrect" .	The user is given the option to recover the password by clicking on the "Forgot password?" button.
			2. Enter the Password for the Mirrorball account in the password field	"An invalid Password"		
			3. Click "Login" button			
TC_01_3	Error message on unsuccessful customer login to Mirrorball website due to absence of account	1. An invalid Mirrorball User name for login to be available 2. Mirrorball site is launched on a compatible browser	1. In the login Panel, enter the username	"An invalid username"	The user is not logged in to the Mirrorball website and an error message is displayed - "This account does not exist" .	The user is prompted to create a Mirrorball user account.

			2. Enter the Password for the Mirrorball account in the password field	"An invalid Password"		
			3. Click "Login" button			
TC_01_4	First time user login- information display check	1. A valid Mirrorball User name for login to be available 2. Mirrorball site is launched on a compatible browser	1. In the login Panel, enter the username	"A valid username"	The user is logged in successfully and the personal information page is displayed	These fields are grayed out and cannot be modified
			2. Enter the Password for the Mirrorball-User account in the password field	"A valid Password"	Check if the following fields are disabled for entry in Personal Details: 1. Customer ID 2. First name 3. Last name 4. Address 5. Phone No 6. Date of Birth	
			3. Click "Login" button			
			4. Check the fields on the "Personal information Page"			
TC_01_5	Personal details- modification with valid values- "First Name"	1. Mirrorball site is launched on a compatible browser and a Mirrorball User account holder is logged in to the site	1. Check the fields on the "Personal information Page"	"A valid new value for the first name field"	The first name field needs to now show the new value entered	The database is updated with the new details of the user.
			2. Change the field, "First Name" with enter a valid new name in this field			
			3. Click on "Save"			

TC_02_1	Check the current cart details	1. Mirrorball site is launched on a compatible browser and a Mirrorball User account holder is logged in to the site. 2. User has added products to his/her Cart.	Click on the Cart icon displayed at the top right corner of the page	The product data stored in the database for the logged in user (non-empty value)	The user's added products are displayed on the screen	1. The user can go to checkout and buy the items in the cart by clicking on the "Checkout" button. 2.The user is given a button on the bottom of the Cart page - "Add products to your Cart from your wishlist" which will redirect to the user's wishlist.
TC_02_2	Check the current cart details without having add any product yet	1. Mirrorball site is launched on a compatible browser and a Mirrorball User account holder is logged in to the site 2. User has not added any products to the cart. 3. Previously added products to the cart, if any, have been sent to Checkout.	Click on the Cart icon displayed at the top right corner of the page	The product data stored in the database for the logged in user (empty value)	The message "No products in the Cart right now"	1. The "Checkout" button will be disabled. 2. A prompt would be dispalyed on clicking the "Checkout" button - "Add products to your Cart to go to Checkout :)" 3.The user is given a button on the bottom of the Cart page - "Add products to your Cart from your wishlist" which will redirect to the user's wishlist.
TC_02_3	Check the order history of the user	1. Mirrorball site is launched on a compatible browser and a Mirrorball User account holder is logged in to the site 2. User has made purchases from the site.	Click on the Orders icon displayed at the top right corner of the page	The order data stored in the database for the logged in user (empty value)	1. All the delivered orders' of the logged in User are displayed in a timely fashion. 2. Undelivered orders are displayed with the option of tracking.	1. The user can click on the delivered orders to view more details about the same. 2. The user can click on the "Track" button beside the undelivered orders to track their delivery status.
TC_03_1	Check if seller can add product(s) to his/her catalogue	1. Mirrorball site is launched on a compatible browser and a Mirrorball Seller account holder is logged in to the site.	Click on the "My Catalogue" icon on the top right corner of the page.	Valid data for the new product	The product data is saved in the database.	The product will be displayed to the users.
			Click on the "Add product" button under the dropdown.			

			Fill the form to add the details of the new product and click on the "Save" button.			
TC_03_2	Check if seller can update product(s) in his/her catalogue	1. Mirrorball site is launched on a compatible browser and a Mirrorball Seller account holder is logged in to the site. 2. User has at least one product in his/her catalogue	Click on the "My Catalogue" icon on the top right corner of the page.	Valid updated data for an existing catalogue product	The product data is updated in the database	The product will be displayed to the users with its updated information.
			Click on the "View catalogue" button under the dropdown.			
TC_03_3	Check if seller can remove product(s) his/her catalogue	1. Mirrorball site is launched on a compatible browser and a Mirrorball Seller account holder is logged in to the site. 2. User has at least one product in his/her catalogue	Click on the "My Catalogue" icon on the top right corner of the page.	Valid data for an exisiting catalogue product	The product data is deleted from the database	The product will no longer be displayed to the users.
			Click on the "View catalogue" button under the dropdown.			
TC_04_1	Validate if an admin is able to view the transactions	1. Mirrorball website is launched on a compatible browser 2. An admin user account is available for login	In the login panel, enter the admin username "admin"	Valid admin username: "admin"	The admin user should be able to view the transactions table, which includes the following columns: transaction ID, customer name, date, and total amount.	The admin user is logged out of the system or remains logged in, depending on the test case design.
			Enter the valid password for the non-admin user account in the password field.	Valid admin password: "mypassword"		
			Click the "Login" button.			
			Once the admin user is logged in, navigate to the "Transactions" tab.			
			Verify that the admin is able to view the transactions table.			

TC_04_2	Validate if a non-admin user is restricted from viewing the transactions	1. Mirrorball website is launched on a compatible browser 2. An non-admin account is available for login	In the login panel, enter the non-admin username "user1".	Valid non-admin username: "user1"	The non-admin user should not be able to view the transactions table and an error message should be displayed stating that the user is not authorized to view the transactions.	The non-admin user is logged out of the system or remains logged in, depending on the test case design.
			Enter the valid password for the admin user account in the password field.	Valid non-admin password: "mypassword"		
			Click the "Login" button.			
			Once the non-admin user is logged in, check if the user is unable to navigate to the "Transactions" tab and is unable to view the transactions table.			
TC_04_3	Validate if an admin user is able to search for a specific transaction	1. Mirrorball website is launched on a compatible browser 2. An admin user account is available for login 3. Transactions have been made on the website	In the login panel, enter the admin username "admin"	Valid admin username: "admin"	The admin user should be able to search for a specific transaction by transaction ID or customer name and view the filtered table.	The admin user is logged out of the system or remains logged in, depending on the test case design.
			Enter the valid password for the non-admin user account in the password field.	Valid admin password: "mypassword"		
			Click the "Login" button.			
			Enter a transaction ID or customer name in the search bar.			
			Verify that the transaction table is filtered to show only the transactions that match the search criteria.			
TC_05_1	Verify if admin user is able to view analytics and generate reports	1. Mirrorball website is launched on a compatible browser 2. An admin user account is available for login and is logged in	Navigate to the Analytics section of the Mirrorball website.	Admin user with appropriate privileges	The report is generated successfully.	The report is displayed on the screen and is available for download.

			Click on the "Generate Report" button.	Date range for generating the report	The report contains the required analytics data.	
			Select the desired date range for the report.			
			Click on the "Generate" button to generate the report.			
TC_05_2	Verify if a non-admin user is unable to view analytics and generate reports.	1. Mirrorball website is launched on a compatible browser 2. A non-admin account is available for login and is logged in	Navigate to the Analytics section of the Mirrorball website.	Non-admin user with appropriate privileges	The user is unable to generate the report.	The user is redirected back to the Analytics section of the website.
			Click on the "Generate Report" button.	Date range for generating the report	An error message is displayed, stating that the user does not have the required privileges.	
			Try to select the desired date range for the report.			
			Click on the "Generate" button to generate the report.			
TC_05_3	Verify if admin user is able to view analytics without generating reports.	1. Mirrorball website is launched on a compatible browser 2. An admin user account is available for login and is logged in	Navigate to the Analytics section of the Mirrorball website.	Admin user with appropriate privileges	The user is able to view the analytics data without generating a report.	The user is still in the Analytics section of the website.
			View the analytics data.			

SHRI VILEPARLE KELAVANI MANDAL'S
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC ACCREDITED with "A" GRADE (CGPA : 3.18)

Software Engineering - Experiment 8

SAP ID: **60004210155, 60004210166, 60004220130**
Name: **Jigar Siddhpura, Aman Nambisan, Falguni Parmar**
Div: **C2**
Batch: **C22**

Aim: Create a Risk Mitigation, Monitoring and Management Plan

Theory: RMMM, also known as Risk Mitigation, Monitoring, and Management, is a framework or methodology used to effectively identify, assess, and address risks in a systematic manner throughout the lifecycle of a project or system. It provides a structured approach for managing risks to minimize their potential negative impact and ensure the successful completion of a project. Here's a theoretical overview of the RMMM framework:

- Risk Identification: The first step in the RMMM process is to identify potential risks that could impact the project or system. This involves systematically examining various aspects, such as technology, resources, stakeholders, requirements, and external factors, to identify potential risks and their sources.

- Risk Assessment: Once risks are identified, they need to be assessed to understand their likelihood of occurrence and potential impact on the project. This involves evaluating the probability, severity, and detectability of each risk to prioritize them based on their significance.

- Risk Mitigation: Risk mitigation involves developing strategies and actions to reduce the probability or impact of identified risks. This could include implementing preventive measures, such as improving security controls, conducting thorough testing, or implementing redundancy. The goal is to minimize the likelihood or severity of risks and enhance the project's resilience.

- Risk Monitoring: Throughout the project lifecycle, risks need to be continuously monitored to detect any changes or new risks that may arise. This involves tracking and analyzing risk indicators, metrics, and triggers to identify warning signs or deviations from expected outcomes. Regular monitoring enables proactive risk management and timely intervention.

- Risk Management: Risk management is an ongoing process that involves decision-making, planning, and coordination to address risks effectively. This includes assigning responsibilities, establishing communication channels, and

defining escalation procedures to ensure that risks are managed and mitigated in a coordinated and timely manner.

- Documentation and Lessons Learned: It is essential to document all identified risks, their assessment, mitigation strategies, and the outcomes of risk management activities. This documentation serves as a valuable reference for future projects and enables the organization to learn from past experiences, continuously improve risk management practices, and apply lessons learned.

By following the RMMM framework, organizations can systematically address risks throughout the project lifecycle, enhance decision-making, and improve overall project success rates. It provides a structured approach to identify, assess, mitigate, and monitor risks, enabling stakeholders to have better visibility and control over potential threats and uncertainties.

Risk Table:

Risks	Category	Probability	Impact
Equipment failure	TI	70%	1
Late delivery	BU	30%	1
Technology will not meet expectations	TE	25%	1
End users resist system	BU	20%	1
Changes in requirements	PS	20%	2
Fluctuations in cryptocurrency prices	BU	50%	2
Regulatory changes in Cryptocurrency	PS	15%	3
Less reuse than planned	PS	60%	3
Payment disputes	BU	30%	3

Smart contract vulnerabilities	TE	25%	4

Poor comments in code	TI	20%	4
Cybersecurity breaches	TI	40%	4

Risk Information Sheet (RIS) for risk no. 10 :

Risk Information Sheet		
Risk ID : P02-4-32	**Date** : 3/4/24	**Prob** : 25%
Description : Smart contracts are self-executing computer programs that are stored on a blockchain. They can be used to automate processes , create decentralized applications, and facilitate transactions. However, smart contracts are also vulnerable to security risks including coding erros, design flaws, and other vulnerabilities that could lead to exploitation and financial losses.		
Refinement /Context Subcondition 1- The smart contract code is not thoroughly reviewed before deployment. Subcondition 2- The smart contract is not tested under different conditions before deployment. Subcondition 3- The smart contract is not audited by an external security team. Subcondition 4- The smart contract is not updated regularly to address newly discovered vulnerabilities. Subcondition 5- The smart contract is not deployed on a secure and reliable blockchain platform. Subcondition 6- The deployment team lacks knowledge and experience in smart contract security.		
Mitigation /Monitoring: 1. Use of well-tested and audited smart contract libraries and frameworks. 2. Ensure secure coding practices, such as input validation and proper error handling, and followed during smart contract development. 3. Conduct regular security assessments and penetration testing to identify and mitigate vulnerabilities. 4. Implement a bug bounty program to incentivize researchers to identify and report vulnerabilities. 5. Monitor the blockchain network and smart contracts for suspicious activity using blockchain monitoring tools. 6. Have a disaster recovery plan in place to address security incidents and minimize impact. 7. Stay up to date on the latest security threats and developments in the blockchain and smart contract space.		
Management /Contingency Plan /Trigger: Allocate budget for regular security audits and penetration testing of smart contracts. Ensure smart contract developers undergo thorough security training. Develop a contingency plan for response to security breaches or incidents involving smart contracts. **Trigger**: security breaches or incidents involving smart contracts		
Current Status : No significant smart contract vulnerabilities have been identified in the system as of the last security audit conducted on April 3, 2024. However, continuous monitoring and testing are being carried out to ensure the system remains secure against potential vulnerabilities		
Originator : Jigar Siddhpura	**Assigned** : Aman Nambisan	

Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

Academic Year: 2023_24

Name: Jigar Siddhpura **SAPID:** 60004200155

DIV: C/C2 **Branch:** Computer Engineering

SE - Experiment 9 - Configuration Management

Aim: Study of Configuration Management using GitHub

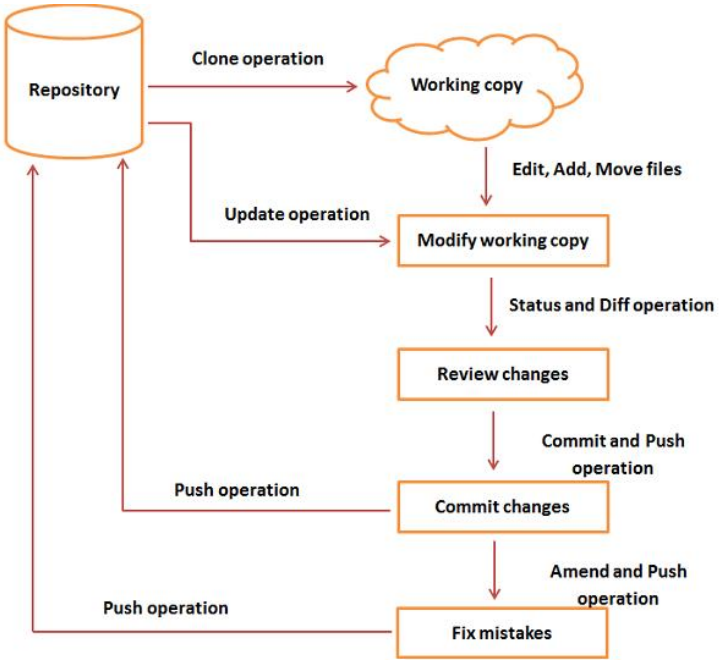
Theory:

Git is a distributed revision control and source code management system with an emphasis on speed. Git was initially designed and developed by Linus Torvalds for Linux kernel development. Git is a free software distributed under the terms of the GNU General Public License version 2.

Git Life Cycle

General workflow is as follows −

1. Clone the Git repository as a working copy.
2. Modify the working copy by adding/editing files.
3. If necessary, update the working copy by taking other developer's changes.
4. Review the changes before commit.
5. Commit changes. If everything is fine, then push the changes to the repository.
6. After committing, if something is wrong, then correct the last commit and push the changes to the repository.



Academic Year: 2023_24

Git Life Cycle

1. Creating Git Repository

   Initialize a new repository by using **init** command followed by **--bare** option. It initializes the repository without a working directory. By convention, the bare repository must be named as **.git**.

```
[gituser@CentOS ~]$ pwd
/home/gituser

[gituser@CentOS ~]$ mkdir project.git

[gituser@CentOS ~]$ cd project.git/

[gituser@CentOS project.git]$ ls

[gituser@CentOS project.git]$ git --bare init
Initialized empty Git repository in /home/gituser-
m/project.git/

[gituser@CentOS project.git]$ ls
branches config description HEAD hooks info objects refs
```

2. Generate Public-Private RSA Key Pair

```
User1@CentOS ~]$ pwd
/home/user1

[user1@CentOS ~]$ ssh-keygen
```

3. Adding keys to authorized keys

   Suppose there are two developers working on a project. Both users have generated public keys.

   Both add their public key to the server by using ssh-copy-id command as given below

```
[user1@CentOS ~]$ pwd
/home/user1

[user2@CentOS ~]$ ssh-copy-id -i ~/.ssh/id_rsa.pub
gituser@git.server.com
```

4. Push changes to the repository

   We have created a bare repository on the server and allowed access for two users. Both users can push their changes to the repository by adding it as a remote.

Git init command creates **.git** directory to store metadata about the repository every time it reads the configuration from the .**git/config** file.

User1 creates a new directory, adds README file, and commits his change as initial commit. After commit, he verifies the commit message by running the **git log** command.

```
[user1@CentOS ~]$ pwd
/home/user1

[user1@CentOS ~]$ mkdir user1_repo

[user1@CentOS ~]$ cd user1_repo/

[user1@CentOS user1_repo]$ git init
Initialized empty Git repository in
/home/user1/user1_repo/.git/

[user1@CentOS user1_repo]$ echo 'TODO: Add contents for
README' > README

[user1@CentOS user1_repo]$ git status -s
?? README

[user1@CentOS user1_repo]$ git add .

[user1@CentOS user1_repo]$ git status -s
A README

[user1@CentOS user1_repo]$ git commit -m 'Initial commit'
```

5. Checking log message by executing the git log command.

```
[user1@CentOS user1_repo]$ git log
```

6. Commit changes

To commit the changes, he used the git commit command followed by –m option. If we omit –m option. Git will open a text editor where we can write multiline commit message

```
[user2@CentOS project]$ git commit -m 'Implemented
my_strlen function'
```

Academic Year: 2023_24

Performance:

1. Perform all the commands using Git

```
C:\Users\Admin>cd ../..

C:\>cd .vscode

C:\.vscode>cd college

C:\.vscode\college>cd experiments

C:\.vscode\college\experiments>git clone https://github.com/Greninja28/SE-10.git
Cloning into 'SE-10'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 3 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.

C:\.vscode\college\experiments>cd SE-10

C:\.vscode\college\experiments\SE-10>code .

C:\.vscode\college\experiments\SE-10>
```

2. Take screenshots of each of the command and respective output

```
PS C:\.vscode\college\experiments\SE-10> git add .
PS C:\.vscode\college\experiments\SE-10> git status -s
M  start.py
PS C:\.vscode\college\experiments\SE-10> git commit -m 'file changed'
[main 8992e87] file changed
 1 file changed, 1 insertion(+)
PS C:\.vscode\college\experiments\SE-10> git status -s
PS C:\.vscode\college\experiments\SE-10> git log
commit 8992e871e8f52ea791ea992f0dcd717ec04516fb (HEAD -> main)
Author: Sahej Jain <89766122+Greninja28@users.noreply.github.com>
Date:   Wed May 3 09:20:21 2023 +0530

    file changed

commit 2accd86c64e87864ffb442b5e68bb33432cb82bc (origin/main, origin/HEAD)
Author: Sahej Jain <89766122+Greninja28@users.noreply.github.com>
Date:   Wed May 3 09:16:49 2023 +0530

    Initail commit
```

3. Explore the commands for merging the documents and show the screenshots.

```
PS C:\.vscode\college\experiments\SE-10> git pull origin main
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused
Unpacking objects: 100% (3/3), 665 bytes | 41.00 KiB/s, done.
From https://github.com/Greninja28/SE-10
 * branch            main       -> FETCH_HEAD
   8992e87..7ff546c  main       -> origin/main
Updating 8992e87..7ff546c
Fast-forward
 README.md | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 README.md
PS C:\.vscode\college\experiments\SE-10> git add .
PS C:\.vscode\college\experiments\SE-10> git commit -m 'Added a sum program'
[main f20de92] Added a sum program
 1 file changed, 3 insertions(+), 2 deletions(-)
PS C:\.vscode\college\experiments\SE-10> git push origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 343 bytes | 171.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/Greninja28/SE-10.git
   7ff546c..f20de92  main -> main
PS C:\.vscode\college\experiments\SE-10>
```

Conclusion:

In conclusion, the study experiment on Git has provided valuable insights into the benefits of using Git for version control in software development. The experiment demonstrated that Git offers powerful tools and features to manage code changes, collaborate more efficiently, and ensure codebase consistency.

One of the significant advantages of Git is its branching and merging capabilities, which enable developers to work on multiple versions of a project concurrently without causing conflicts. This functionality is essential when working with large teams or when making significant changes to a codebase. Additionally, Git's revert and reset features make it easy to undo changes and recover previous versions of code, which can save developers time and effort in troubleshooting.

Furthermore, the study experiment confirmed that Git enhances collaboration and facilitates communication between team members by providing a centralized repository where all project files can be stored, accessed, and shared. This, in turn, reduces the likelihood of errors and inconsistencies in the codebase

A.Y 2022-23

Experiment No. 10

Name: **Jigar Siddhpura**
SAP ID: **60004210155**
Class/Batch: **TE C/C2**

Aim: Study of Azure Devops

Theory:

Azure DevOps provides developer services for allowing teams to plan work, collaborate on code development, and build and deploy applications. Azure DevOps supports a collaborative culture and set of processes that bring together developers, project managers, and contributors to develop software. It allows organizations to create and improve products at a faster pace than they can with traditional software development approaches.

Azure DevOps provides integrated features that you can access through your web browser or IDE client.

Azure Repos:

Azure Repos is a set of version control tools that you can use to manage your code. Version control systems are software that help you track changes you make in your code over time. As you edit your code, you tell the version control system to take a snapshot of your files. The version control system saves that snapshot permanently so you can recall it later if you need it. Use version control to save your work and coordinate code changes across your team.

Azure Repos provides two types of version control:

1. Git repositories: Git is the most commonly used version control system today and is quickly becoming the standard for version control. Git is a distributed version control system, meaning that your local copy of code is a complete version control repository. These fully functional local repositories make it is easy to work offline or remotely. You commit your work locally, and then sync your copy of the repository with the copy on the server.
2. Team Foundation Version Control (TFVC): Azure Repos also supports Team Foundation Version Control (TFVC). TFVC is a centralized version control system.

Typically, team members have only one version of each file on their dev machines.

Historical data is maintained only on the server. Branches are path-based and created on the server.

Azure Pipelines:

Azure Pipelines automatically builds and tests code projects to make them available to others. It works with just about any language or project type. Azure Pipelines combines continuous integration (CI) and continuous delivery (CD) to test and build your code and ship it to any target.

Continuous Integration (CI) is the practice used by development teams of automating merging and testing code. Implementing CI helps to catch bugs early in the development cycle, which makes them less expensive to fix. Automated tests execute as part of the CI process to ensure quality. Artifacts are produced from CI systems and fed to release processes to drive frequent deployments. The Build service in Azure DevOps Server helps you set up and manage CI for your applications.

Continuous Delivery (CD) is a process by which code is built, tested, and deployed to one or more test and production environments. Deploying and testing in multiple environments increases quality. CI systems produce deployable artifacts, including infrastructure and apps. Automated release processes consume these artifacts to release new versions and fixes to existing systems. Monitoring and alerting systems run continually to drive visibility into the entire CD process.

Azure Boards:

Delivers a suite of Agile tools to support planning and tracking work, code defects, and issues using Kanban and Scrum methods. Azure Boards provides software development teams with the interactive and customizable tools they need to manage their software projects. It provides a rich set of capabilities including native support for Agile, Scrum, and Kanban processes, calendar views, configurable dashboards, and integrated reporting. These tools scale as your

business grows.

Quickly and easily track work, issues, and code defects associated with your project. The Kanban board, shown in the following image, is just one of several tools that allows you to add, update, and filter user stories, bugs, features, and epics.

Azure Test Plans:

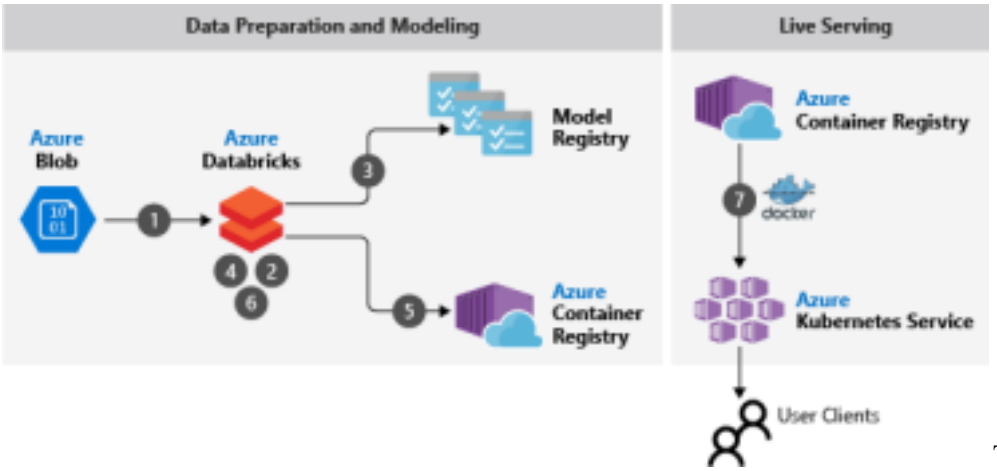
Azure Test Plans provides rich and powerful tools everyone in the team can use to drive quality and collaboration throughout the development process. The easy-to-use, browser-based test management solution provides all the capabilities required for planned manual testing, user acceptance testing, exploratory testing, and gathering feedback from stakeholders.

Azure Artifacts:

Azure Artifacts enable developers to consume and publish different types of packages to Artifacts feeds and public registries such as NuGet.org and npmjs.com. You can use Azure Artifacts in conjunction with Azure Pipelines to deploy packages, publish build artifacts, or integrate files between your pipeline stages to build, test, or deploy your application.

Azure Architecture Solutions:

1. Azure Architecture for Content based Recommendation System:



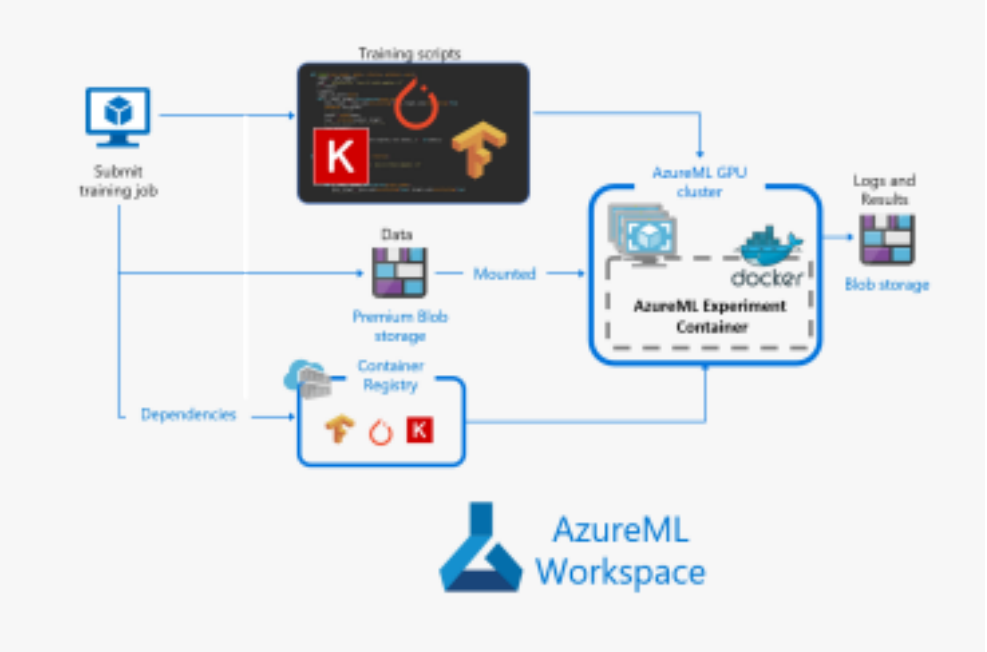
This example scenario covers the training, evaluation, and deployment of a machine learning model for content-based personalization on Apache Spark using Azure Databricks. In this case, a model

is trained with a supervised classification algorithm on a dataset containing user and item features. The label for each example is a binary value indicating that the user engaged

with (for example, clicked) an item. This scenario covers a subset of the steps required for a full end-to-end recommendation system workload. The broader context of this scenario is based on a generic e-commerce website with a front end that serves rapidly changing content to its users. This website uses cookies and user profiles to personalize the content for that user. Along with user profiles, the website may have information about every item it serves to each user.

2. Azure Architecture for Distributed Training Deep learning models:

This reference architecture shows how to conduct distributed training of deep learning models across clusters of GPU-enabled VMs. The scenario is image classification, but the solution can be generalized to other deep learning scenarios such as segmentation or object detection.



Workflow

This architecture consists of the following services:

Azure Machine Learning Compute plays the central role in this architecture by scaling resources up and down according to need. Azure ML Compute is a service that helps provision and manage clusters of VMs, schedule jobs, gather results, scale resources, and

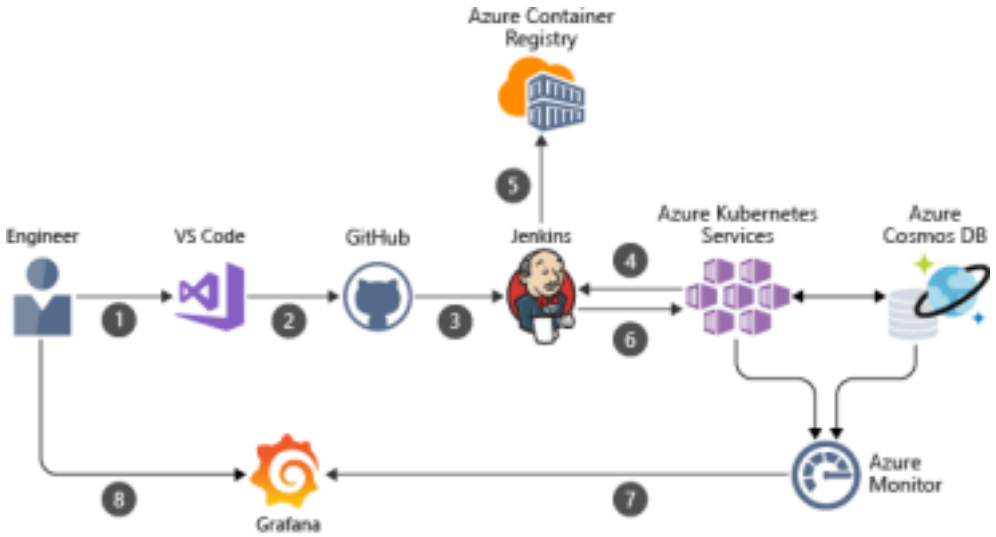
handle failures. It supports GPU-enabled VMs for deep learning workloads.

Standard Blob storage is used to store the logs and results. Premium Blob storage is used to store the training data and is mounted in the nodes of the training cluster using blobfuse. The

Premium tier of Blob storage offers better performance than the Standard tier and is recommended for distributed training scenarios. When mounted using blobfuse, during first the epoch, the training data is downloaded to the local disks of the training cluster and cached. For every subsequent epoch, the data is read from the local disks, which is the most performant option.

Container Registry is used to store the Docker image that Azure Machine Learning Compute uses to run the training.

3. Azure Architecture for CI/CD:



Azure Web Apps is a fast and simple way to create web apps using ASP.NET, Java, Node.js, or PHP. Deliver value faster to your customers with a continuous integration and continuous deployment (CI/CD) pipeline that pushes each of your changes automatically to Web Apps.