E-COMMERCE PLATFORM: SELLER AND BUYER MANAGEMENT

PROJECT REPORT

Submitted by

RESHMA J K [211422243267]

THARANI T [211422243334]

SATHYA SREE R [211422243295]

in partial fulfillment of the requirements for the award of degree

of

BACHELOR OF ENGINEERING

in

ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

PANIMALAR ENGINEERING COLLEGE, CHENNAI-600123

ANNA UNIVERSITY: CHENNAI-600 025

November, 2024

BONAFIDE CERTIFICATE

Certified that this project report titled "**E-COMMERCE PLATFORM: SELLER AND BUYER MANAGEMENT**" is the bonafide work of **RESHMA J K [211422243267], THARANI T[211422243334], SATHYA SREE R[211422243295]** who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Dr.S. MALATHI M.E., Ph.D
HEAD OF THE DEPARTMENT

DEPARTMENT OF AI & DS,
PANIMALAR ENGINEERING COLLEGE,
NASARATHPETTAI,
POONAMALLEE,
CHENNAI - 600123.

SIGNATURE

K. CHARULATHA M.E.,
SUPERVISOR
ASSISTANT PROFESSOR

DEPARTMENT OF AI & DS,
PANIMALAR ENGINEERING COLLEGE,
NASARATHPETTAI,
POONAMALLEE,
CHENNAI - 600123.

Certified that the above candidates were examined in the End Semester mini Project Viva-Voce Examination held on.....

INTERNAL EXAMINER

EXTERNAL EXAMINER

DECLARATION BY THE STUDENT

We **RESHMA J K [211422243267] ,THARANI T [211422243334], SATHYA SREE R [211422243295]** hereby declare that this mini project report titled **"E-commerce Platform: Seller and Buyer Management",** under the guidance of **K.CHARULATHA, M.E** is the original work done by us and I have not plagiarized or submitted to any other degree in any university by me.

RESHMA J K [211422243267]

THARANI T [211422243334]

SATHYA SREE R [211422243295]

ACKNOWLEDGEMENT

I also take this opportunity to thank all the Faculty and Non-Teaching Staff Members of Department of Artificial Intelligence and Data Science for their constant support. Finally I thank each and every one who helped me to complete this project. At the outset we would like to express our gratitude to our beloved respected Chairman, **Dr.Jeppiaar M.A.,Ph.D,** Our beloved correspondent and Secretary **Mr.P.Chinnadurai M.A., M.Phil., Ph.D.,** and our esteemed director for their support.

We would like to express thanks to our Principal, **Dr. K. Mani M.E., Ph.D.,** for having extended his guidance and cooperation.

We would also like to thank our Head of the Department, **Dr.S.Malathi M.E.,Ph.D.,** of Artificial Intelligence and Data Science for her encouragement.

Personally we thank K. Charulatha, M.E, Assistant Professor, Department of Artificial Intelligence and Data Science for the persistent motivation and support for this project, who at all times was the mentor of germination of the project from a small idea.

We express our thanks to the project coordinator **Ms. C.M HILDA JERLIN M.E., Assistant Professor** in Department of Artificial Intelligence and Data Science for their Valuable suggestions from time to time at every stage of our project.

Finally, we would like to take this opportunity to thank our family members, friends, and well-wishers who have helped us for the successful completion of our project.

We also take the opportunity to thank all faculty and non-teaching staff members in our department for their timely guidance in completing our project.

RESHMA J K [211422243267]

THARANI T [211422243334]

SATHYA SREE R [211422243295]

ABSTRACT

Goods trade is a supply chain transaction that involves shippers buying goods from suppliers and carriers providing goods transportation. Various business documents like purchase order, despatch advice, invoices, and receive advice get exchanged among the trade participants during any trade transaction. Similarly, various business processes like freight transport, invoice generation, goods receiving, invoice processing, and payment processing get executed by the participants in a trade transaction. Discrepancy during the execution of any of these processes leads to disputes between the participants involved, and the time consumed in resolving the disputes causes a delay in the process execution resulting in cost overhead for all the participants involved. Shippers are issued invoices from suppliers for the goods provided and from carriers for goods transportation. The shipper carries out goods receiving and invoice processing before proceeding to payment processing of bills for suppliers and carriers, where invoice processing includes tasks like processing claims and adjusting the payments. Goods receiving involves verification of received goods by Shipper's receiving team. Processing claims and adjusting the payments are carried out by Shipper's accounts payable team, which in turn is verified by the accounts receivable teams of suppliers and carriers. This paper presents a blockchain-based accounts payable system for shippers, which generates claims for deficiency in the goods received and accordingly adjusts the payment in the bills for suppliers and carriers. Primary motivations for these supply chain organizations to adopt blockchain-based accounts payable systems are to eliminate the process redundancies (accounts payable vs. accounts receivable), to reduce the number of disputes among the transacting participants, to reduce the dispute resolution time, and to accelerate the accounts payable processes via optimizations in the claims generation and blockchain-based dispute reconciliation.

Keywords : *Blockchain Technology, Accounts Payable and Receivable, Supply Chain Optimization,*

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

Any trade transaction, be it domestic or global, involves exercising certain processes to complete. Domestic trade is the exchange of goods within country boundaries in contrast to between different countries in global/international trade. We describe the different processes involved in goods trade using a global trade transaction in figure Fig. 2. Shippers initiate a trade transaction by sending a purchase order (PO) which consists of details of the requested goods to the suppliers. Suppliers typically package the goods into intermodal containers either by themselves or with the help of Origin Cargo Management (OCM) team. Suppliers issue despatch advice (DA) that describes the goods packed details, and commercial invoice (CI) that describes the terms together with the details of the amount that shipper must pay for the goods supplied. Since global trade involves freight transportation across country borders, a typical freight journey involves multiple modes (e.g., road, rail, or sea) of carriers contributing to the container movement from origin to the destination. Moreover, freight transportation may also involve drayage providers to move containers a short distance via ground fright (e.g., move containers from truck to a ship). Once freight reaches the delivery center at destination, the goods receiving team of the shipper verifies if the received goods can be accepted or not. If there are any damages to the received goods or discrepancies in terms of received quantity/price against PO, then the receiving team records the same via receiving advice (RA). The different carriers involved in freight movement also issue their respective invoices for their services. Once the shipper has access to invoices of carriers and the supplier, its accounts payable team needs to process the invoices. First, the accounts payable team raises a claim for the discrepancies reported in RA in the form of claim advice (CA). Second, the

accounts payable team deducts the amount captured in CA from the appropriate invoice (either from the supplier's invoice or from a carrier's invoice whoever is accountable) and generate payment advices (PAs), where each PA captures the net amount payable by the shipper either to the supplier or to a carrier.

1.2 PROBLEM DEFINITION

- **EXISTING SYSTEM**
- In before years the data does not store but here, we are using block chain it is more secure for the data
- In may transaction area does not send a money in safe mode.

- **DISADVANTAGES**
- Lows of security
- The user without login, Encrypted data cannot store in block chain

CHAPTER-2

LITERATURE SURVEY

1. Title: Security and privacy for the internet of medical things enabled healthcare systems: A survey

Author: Y singh

Abstract: Goods trade is a supply chain transaction that involves shippers buying goods from suppliers and carriers providing goods transportation. Shippers are issued invoices from suppliers and carriers. Shippers carry out goods receiving and invoice processing before payment processing of bills for suppliers and carriers, where invoice processing includes tasks like processing claims and adjusting the bill payments. Goods receiving involves verification of received goods by the Shipper's receiving team. Invoice processing is carried out by the Shipper's accounts payable team, which in turn is verified by the accounts receivable teams of suppliers and carriers. This paper presents a blockchain-based accounts payable system that generates claims for the deficiency in the goods received and accordingly adjusts the payment in the bills for suppliers and carriers.

2. Title: Blockchain based accounts payable platform for goods trade

Author: VP Yanambaka

Abstract: This paper presents a blockchain-based accounts payable system that generates claims for the deficiency in the goods received and accordingly adjusts the payment in the bills for suppliers and carriers. Primary motivations for these supply chain organizations to adopt blockchain-based accounts payable systems are to eliminate the process redundancies (accounts payable vs. accounts receivable), to

reduce the number of disputes among the transacting participants, and to accelerate the accounts payable processes via optimizations in the claims generation and blockchain-based dispute reconciliation.

3. Title: All Aboard! Major Shipping Lines Secure Antitrust Immunity for TradeLens Blockchain Agreement

Author: En Cheng

Abstract: Despite the growing interest in blockchain technology, there are few examples of business value being delivered by live solutions. One exception is TradeLens, a blockchain- enabled platform for tracking shipping containers and related documentation in global supply chains. This article describes the TradeLens journey from initial prototypes and pilots to its live deployment. Although TradeLens still has a long way to go, its vision to substantially improve global supply chains has kept participants engaged and committed to adopting and growing the ecosystem.

4. Title: Ultra low power ECG processing system for IoT devices

Author: Guihua Er

Abstract: The Internet of Things (IoT) represents a set of interconnected smart objects and people at any time and at any place. The IoT incorporates wide spectrum that can impact businesses, healthcare, social and political aspects. It is a platform that extends from sensors, local processors, wireless transmitters, and central management stations

CHAPTER 3

SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

In existing e-commerce systems, data security and transaction safety are major concerns. Data is often stored in centralized servers, making it vulnerable to breaches and tampering. Additionally, financial transactions in many platforms lack a safe and transparent mode of operation, leaving users at risk. To address these issues, our project integrates blockchain technology for enhanced security and trust. Blockchain allows decentralized data storage, ensuring that information is immutable and protected from unauthorized access. This technology also secures financial transactions by utilizing a transparent, tamper-resistant system that guarantees the safe transfer of funds between buyers and sellers. By adopting blockchain, our platform creates a more secure and reliable environment for e-commerce activities. This improves user confidence, enhances transaction safety, and ensures data integrity, addressing the limitations of existing systems while offering a robust solution for the future of online shopping.

DISADVANTAGES

The system faces some disadvantages, such as reduced security if users fail to log in, which leaves data vulnerable. Additionally, blockchain has limitations in storing encrypted data directly, as it requires off-chain solutions for managing sensitive information. These factors may impact the overall security and efficiency.

3.2 PROPOSED SYSTEM

The proposed system focuses on analyzing the impact of increasing transaction send rates on the throughput and latency of Compute CA and Compute PA transactions. Our findings indicate that as the transaction send rate increases, latency for both transaction types tends to rise as well. This phenomenon presents challenges

for maintaining efficient processing times in high-demand scenarios. However, the system is specifically designed to effectively manage shipment throughput within the current global trade ecosystem. By employing advanced algorithms and optimization techniques, it can handle significant transaction volumes without compromising performance. Furthermore, the proposed system is built with scalability in mind, allowing it to adapt to future workloads as global trade continues to evolve. This ensures that the system remains relevant and efficient in a dynamic environment, providing reliable support for businesses looking to optimize their operations in the fast-paced world of international commerce.

ADVANTAGES

The proposed system ensures data security through the use of blockchain technology, which provides a decentralized and tamper-resistant storage solution. Additionally, it utilizes unique blocks created from different fields, enabling efficient and precise searches tailored to user needs. This structure enhances data retrieval speed and accuracy, allowing users to access relevant information quickly while maintaining the integrity and confidentiality of their data.

3.3 DEVELOPMENT ENVIROMENT

SOFTWARE REQUIREMENT

- Front End Language : HTML,CSS,JSP,SERVELTS
- Backend Language : MYSQL, JAVA
- Operating System : Windows 10 or 11
- IDE : Javadevelipemenkit

HARDWARE REQUIREMENT

- System : Intel i5 and above
- Hard Disk : 40 GB.
- Ram : Minimum 4GB
- Processor : 64-bit, four-core, 2.5 GHz minimum per core

INPUT DESIGN

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

- What data should be given as input?
- How the data should be arranged or coded?
- The dialog to guide the operating personnel in providing input.
- Methods for preparing input validations and steps to follow when error occur.

OBJECTIVES

1. Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.
2. It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.
3. When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus, the objective of input design is to create an input layout that is easy to follow

OUTPUT DESIGN

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system's relationship to help user decision-making.

1. Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When

analysis design computer output, they should Identify the specific output that is needed to meet the requirements.

2.Select methods for presenting information.

3.Create document, report, or other formats that contain information produced by the system.

The output form of an information system should accomplish one or more of the following objectives.

- Convey information about past activities, current status or projections of the
- Future.
- Signal important events, opportunities, problems, or warnings.
- Trigger an action.
- Confirm an action.

CHAPTER 4

SYSTEM DESIGN

4.1 UML DIAGRAMS

4.1.1 USE CASE DIAGRAM

Fig 4.1.1 Use case diagram

This use case diagram illustrates an e-commerce system where **Sellers** can manage products and view reviews, **Buyers** can search, view, and purchase items, and **Account Payable** can authorize and manage billing tasks. The system supports features like registration, login, and account management for all users

4.1.2 CLASS DIAGRAM

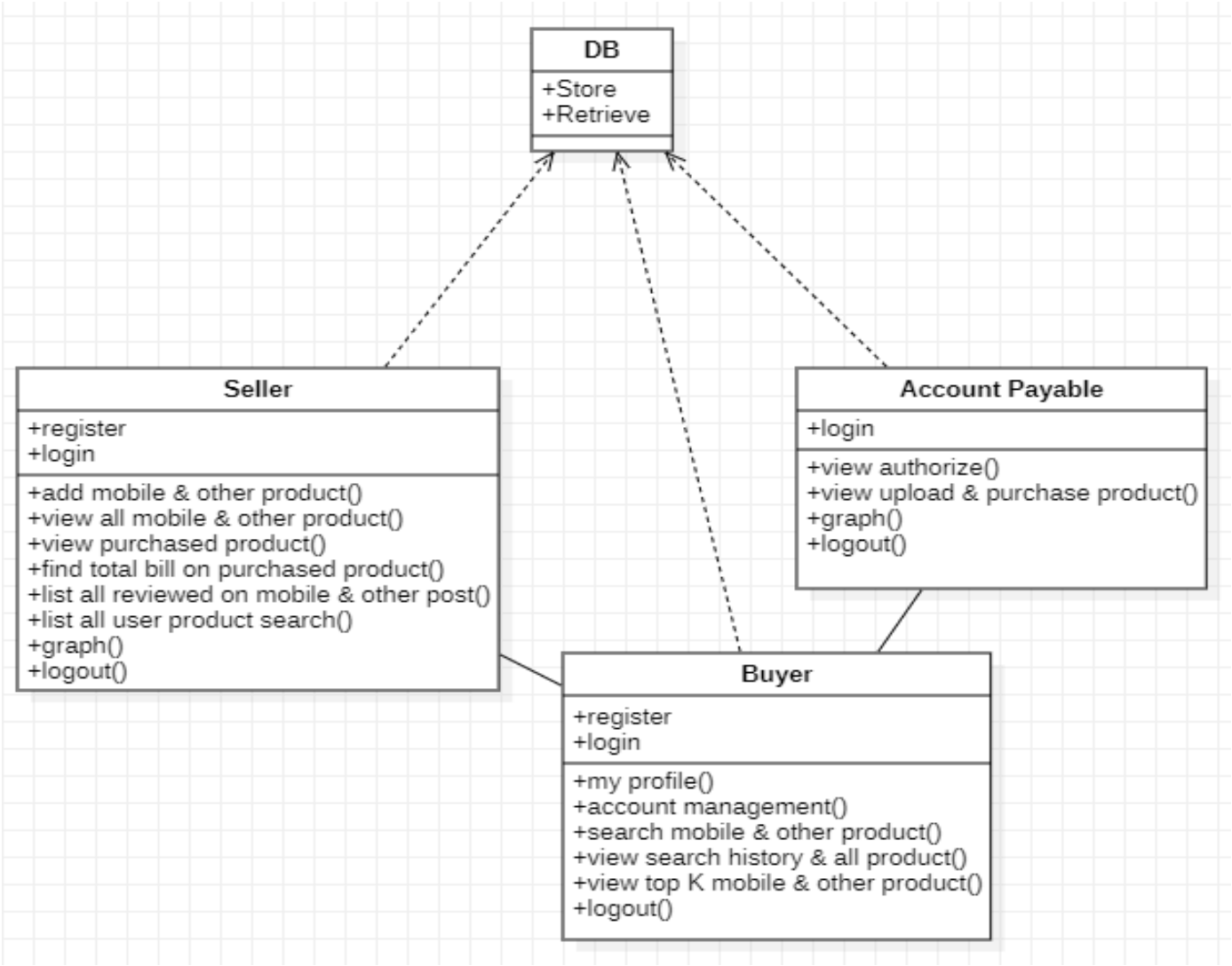

Fig 4.1.2 Class diagram

This class diagram represents an e-commerce system with classes for **Seller**, **Buyer**, **Account Payable**, and a **Database (DB)**. Each class has specific methods for actions like product management, viewing and purchasing items, and account management, while the **DB** class manages data storage and retrieval for all users.

4.1.3 SEQUENCE DIAGRAM

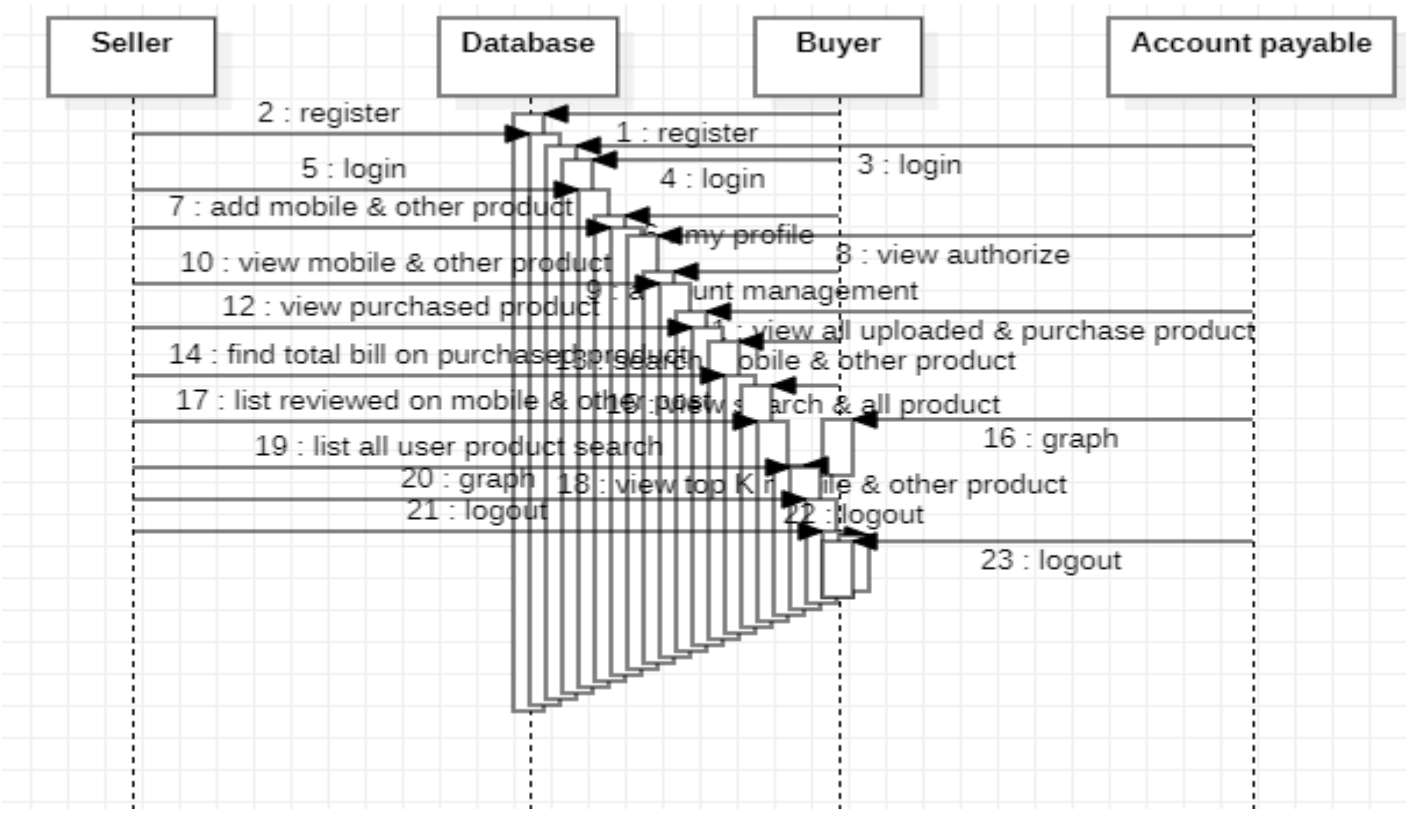

Fig 4.1.3 Sequence Diagram

This sequence diagram illustrates interactions within an e-commerce system among **Seller**, **Buyer**, **Account Payable**, and **Database**. Key actions include **registering** and **logging in** for all users, with the **Database** facilitating data storage and retrieval. Sellers can manage products, buyers can view and search for items, and Account Payable can authorize and view purchase data. Each actor communicates with the database to perform tasks, following a sequential flow from registration to logout.

4.1.4. DEPLOYMENT DIAGRAM

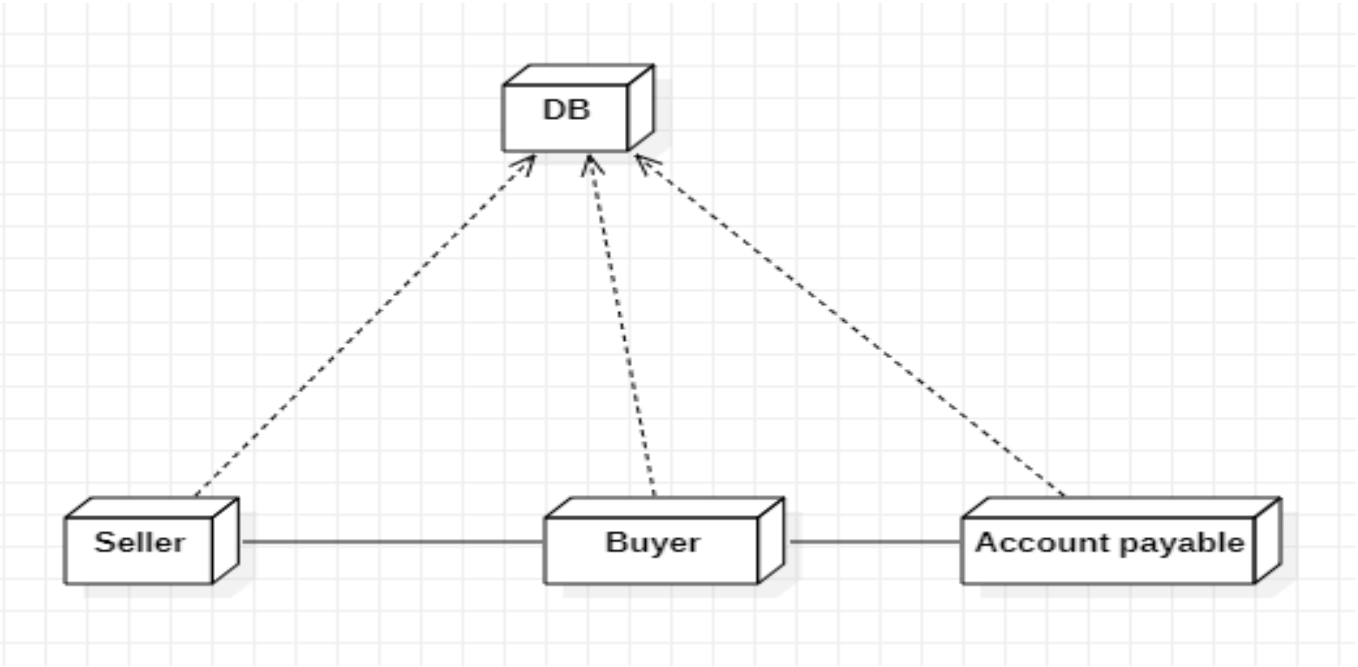

Fig 4.1.4 Deployment Diagram

This diagram illustrates the interaction between a **Seller**, **Buyer**, and **Account Payable** system, all of which connect to a central **Database** to store and retrieve transaction-related data. The solid lines show direct relationships, while dashed lines represent indirect interactions with the database.

4.2 ER DIAGRAM

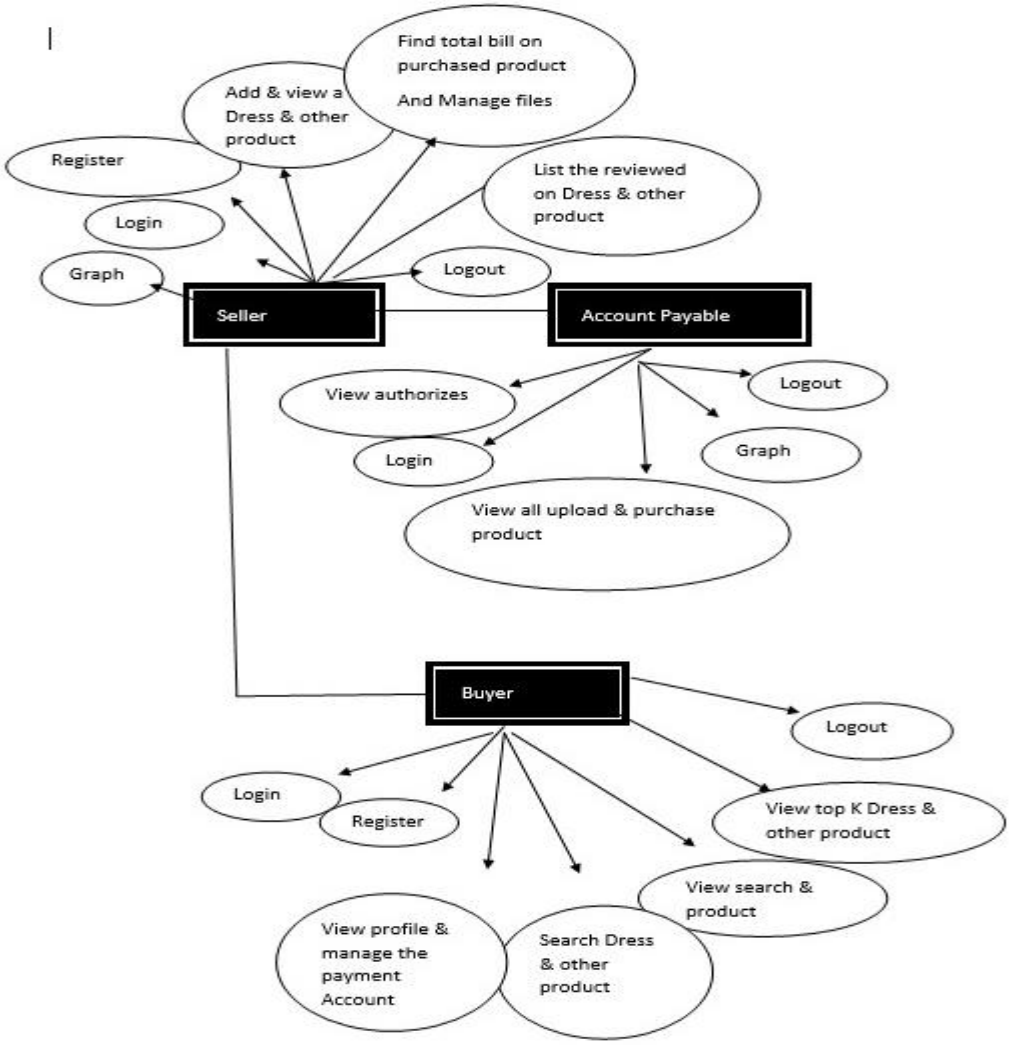

Fig 4.2 ER Diagram

This ER diagram shows the relationships and actions between **Seller**, **Buyer**, and **Account Payable** entities in a system, where each entity can perform various functions like login, register, view products, and manage accounts. Each entity has specific interactions, such as adding/viewing products, managing payments, and generating purchase reports.

4.3 DATAFLOW DIAGRAM

4.3.1 0 LEVEL DFD

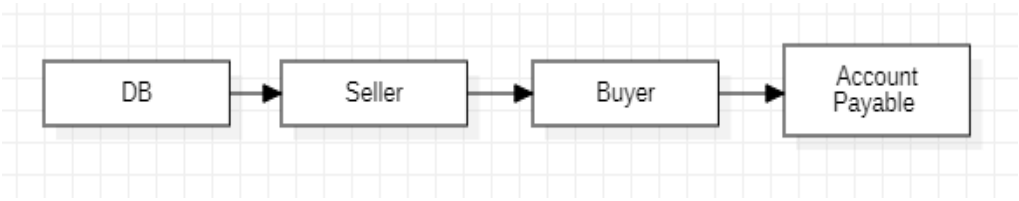

Fig 4.3.1 Data flow diagram level 0

4.3.2 FIRST LEVEL DFD

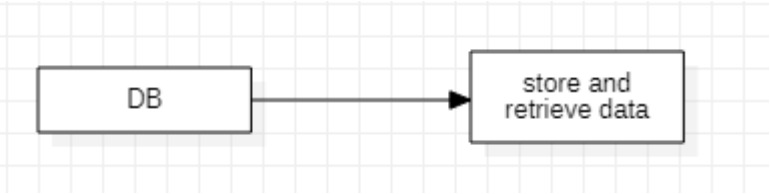

Fig 4.3.2 Dataflow diagram level 1

4.3.3 SECOND LEVEL DFD

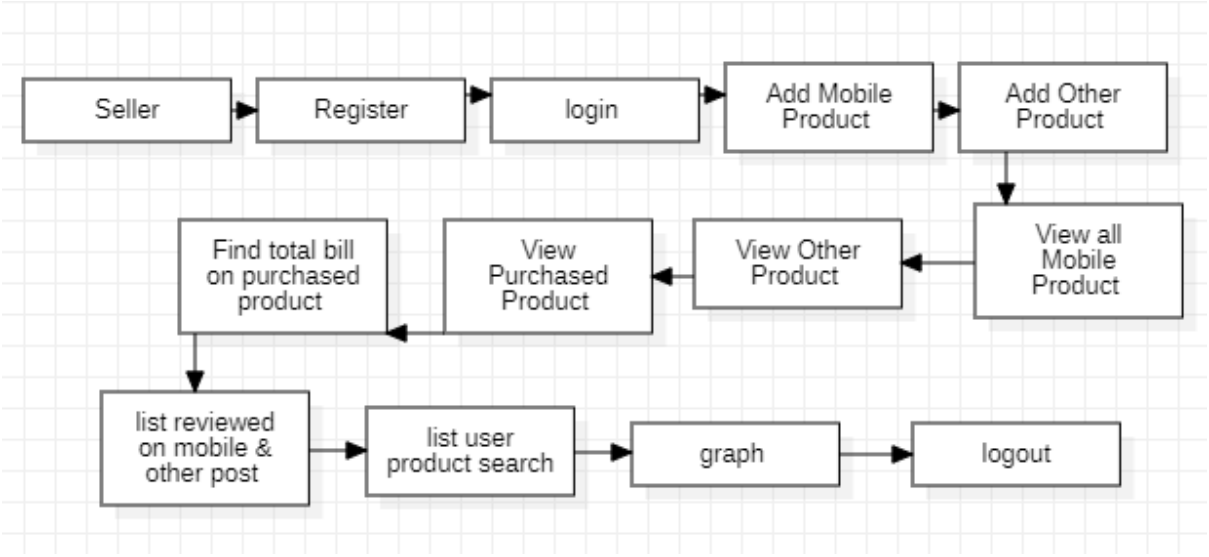

Fig 4.3.3 Dataflow diagram level 2

4.3.4 THIRD LEVEL DFD

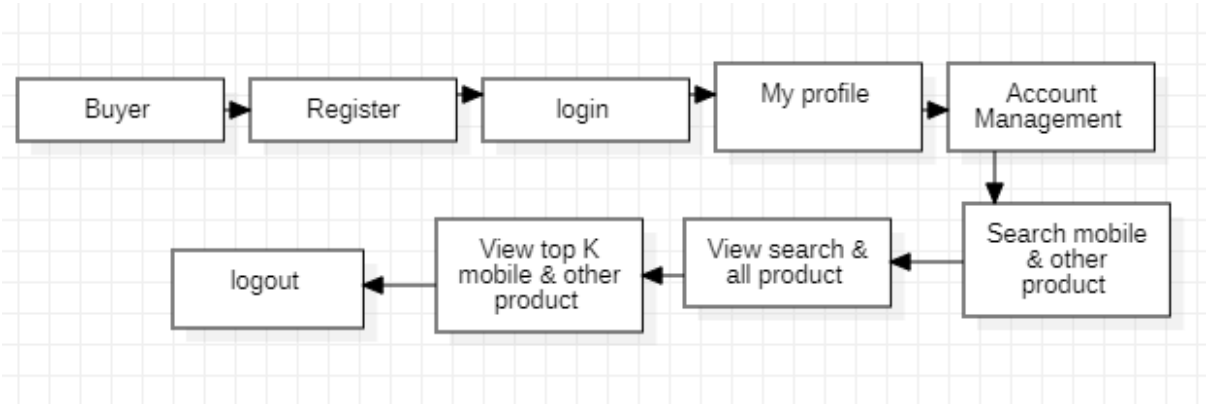

Fig 4.3.4 Dataflow diagram level 3

4.3.5 FOURTH LEVEL DFD

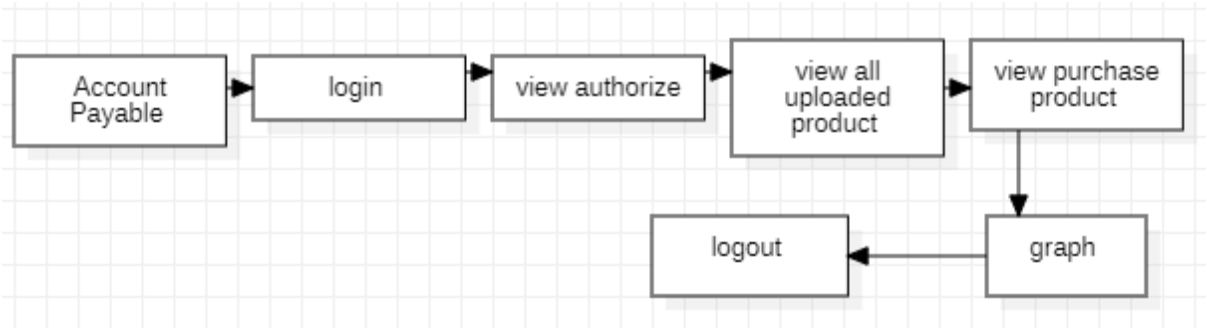

Fig 4.3.5 Dataflow diagram level 4

CHAPTER 5

SYSTEM ARCHITECTURE

5.1 ARCHITECTURE OVERVIEW

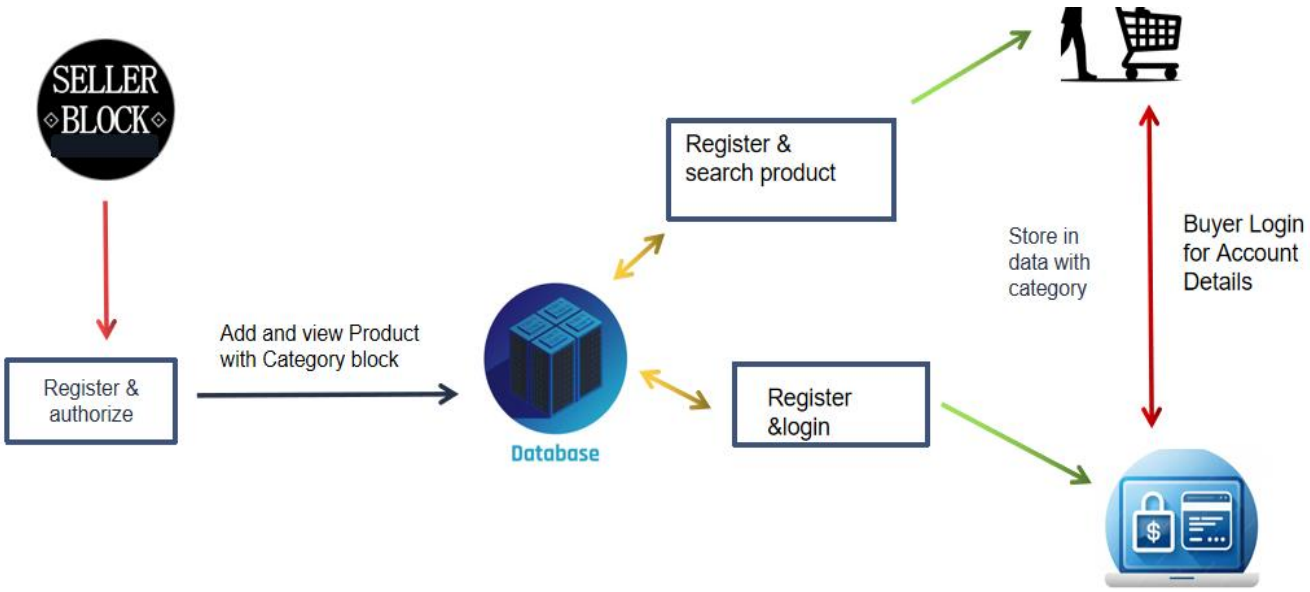

Fig 5.1 Architecture diagram

Overall Functionality

The diagram depicts a simplified e-commerce system where users can register, log in, and interact with products. It seems to focus on the seller's perspective, showcasing how they can add and view products, while buyers can search, register, log in, and make purchases.

Key Components

1. **Seller Block:**
    - **Register & Authorize:** This block likely represents the process where a seller registers their account and is authorized to add and manage products.
    - **Add and View Product with Category Block:** This indicates the seller's ability to add new products, categorize them, and view existing products in their

inventory.

2. **Database:**
    - This central component stores all product information, including categories. It also likely stores user data, order details, and other relevant information.
3. **Buyer Interactions:**
    - **Register & Search Product:** This block represents the actions a buyer takes to register a new account or search for existing products.
    - **Register & Login:** This block shows the options for a buyer to either create a new account or log in to an existing one.
    - **Buyer Login for Account Details:** This indicates that once a buyer logs in, they can access their account details, order history, and other personalized information.
    - **Store in Data with Category:** This step likely involves the system storing the buyer's selected products, along with their categories, in preparation for checkout.
    - **Shopping Cart:** The final arrow points to a shopping cart icon, suggesting that the system allows buyers to add products to their cart and proceed to checkout.

Possible Workflow

1. **Seller Side:**
    - A seller registers and authorizes their account.
    - They add products to the database, categorizing them as needed.
2. **Buyer Side:**
    - A buyer registers or logs in to their account.
    - They search for products using the search functionality.
    - They select desired products and add them to their cart.
    - They proceed to checkout, where their account details and selected products are processed.

Assumptions & Limitations

- The diagram doesn't explicitly show the payment processing or order fulfillment steps.
- It's unclear how the system handles inventory management or product availability.

- The diagram doesn't indicate any error handling or security measures.

Additional Considerations

- The system could benefit from user-friendly interfaces for both sellers and buyers.
- Robust security measures should be in place to protect user data and prevent unauthorized access.
- A clear and efficient checkout process is crucial for a positive user experience.

I hope this explanation is helpful! Feel free to ask if you have any further questions.

5.2 MODULE DESCRIPTION

In this project have three modules:

- 1. Seller Block
- 2. Buyer Block
- 3. Account payable block

Seller Block

- List All Users & Authorize
- Add Category
- Add Sub - Category
- Add dress posts
- Add other posts
- View all Dress Products
- View all Other Products
- View Purchased product
- Find total Bill on Purchased Products
- List All Reviewed on Dress post
- List All Reviewed on Other post
- List all users products searched
- View Dress Product Rank in chart
- View other Product rank in chart
- View product search ratio in chart
- Logout

Buyer Block

- Register and authorize by seller
- Login
- My Profile
- Account Management
- Search Dress Product
- Search Other Product
- View My search History
- View all product
- View Top K Dress Product Result
- View Top K Other Product Result
- Logout

Account Payable Login

- Login
- My Profile
- View & Process Order
- View Purchased Dress Product by block chain
- View Purchased other product by blockchain
- Logout

CHAPTER 6

SYSTEM STUDY

FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

- ECONOMICAL FEASIBILITY
- TECHNICAL FEASIBILITY
- SOCIAL FEASIBILITY

ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical

resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

CHAPTER-7

SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

TYPES OF TESTS

7.1 Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration.unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

7.2 Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit

testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination components.

Functional test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box. you cannot "see" into it. The test provides inputs and responds to outputs without considering how the software works.

Unit Testing:

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail. Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

CHAPTER 8

CONCLUSION AND FUTURE ENHANCEMENTS

8.1 CONCLUSION

We discussed the need for a block chain-based accounts payable system that eliminates the process redundancies (accounts payable vs. accounts receivable), enables efficient invoice processing, and reduces the amount of time spent in reconciling disputes between the transacting participants in goods trade (domestic and global)

8.2 FUTURE ENHANCEMENTS

In Future show that its practical to deploy one such system in real-world customer environments. We are currently experimenting with the use of the proposed system in collaboration with the participants of the Trade Lens platform.

APPENDICES

CODING

```
package algo;
import javax.crypto.Cipher;
import javax.crypto.SecretKey;
import javax.crypto.spec.SecretKeySpec;

public class CipherHelper {

    // Algorithm used
    private final static String ALGORITHM = "DES";

    /**
     * Encrypt data
     * @param secretKey -   a secret key used for encryption
     * @param data      -   data to encrypt
     * @return  Encrypted data
     * @throws Exception
     */
    public String cipher(String secret, String data) throws Exception {
        // Key has to be of length 8
        if (secretKey == null || secretKey.length() != 8)
            throw new Exception("Invalid key length - 8 bytes key needed!");

        SecretKey key = new SecretKeySpec(secretKey.getBytes(), ALGORITHM);
        Cipher cipher = Cipher.getInstance(ALGORITHM);
        cipher.init(Cipher.ENCRYPT_MODE, key);
```

```
    return toHex(cipher.doFinal(data.getBytes()));
}

/**
 * Decrypt data
 * @param secretKey -   a secret key used for decryption
 * @param data      -   data to decrypt
 * @return  Decrypted data
 * @throws Exception
 */
public  String decipher(String secret, String data) throws Exception {
    // Key has to be of length 8
    if (secretKey == null || secretKey.length() != 8)
        throw new Exception("Invalid key length - 8 bytes key needed!");

    SecretKey key = new SecretKeySpec(secretKey.getBytes(), ALGORITHM);
    Cipher cipher = Cipher.getInstance(ALGORITHM);
    cipher.init(Cipher.DECRYPT_MODE, key);
     System.out.println(cipher.doFinal(toByte(data)));
    return new String(cipher.doFinal(toByte(data)));
}

// Helper methods

private static byte[] toByte(String hexString) {
    int len = hexString.length()/2;

    byte[] result = new byte[len];
```

```
        for (int i = 0; i < len; i++)
            result[i] = Integer.valueOf(hexString.substring(2*i, 2*i+2), 16).byteValue();
        return result;
    }

    public static String toHex(byte[] stringBytes) {
        StringBuffer result = new StringBuffer(2*stringBytes.length);

        for (int i = 0; i < stringBytes.length; i++) {
            result.append(HEX.charAt((stringBytes[i]>>4)&0x0f)).append(HEX.charAt(stringByte
s[i]&0x0f));
        }

        return result.toString();
    }
    String secretKey="12345678";
    private final static String HEX = "0123456789ABCDEF";

    public static void main(String[] args) {
        try {
            String secretKey    = "01234567";
            String data="test";
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

}
package block;
```

```
import java.util.List;

public class Block {

    private int previousHash;
    private List<Transaction> transactions;

    public Block(int previousHash, List<Transaction> transactions) {
        this.previousHash = previousHash;
        this.transactions = transactions;
    }

    public int getPreviousHash() {
        return previousHash;
    }

    public void setPreviousHash(int previousHash) {
        this.previousHash = previousHash;
    }

    public List<Transaction> getTransactions() {
        return transactions;
    }

    public void setTransactions(List<Transaction> transactions) {
        this.transactions = transactions;
    }
```

```
    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;

        Block block = (Block) o;

        if (previousHash != block.previousHash) return false;
        return transactions != null ? transactions.equals(block.transactions) : block.transactions ==
null;
    }

    @Override
    public int hashCode() {
        int result = previousHash;
        result = 31 * result + (transactions != null ? transactions.hashCode() : 0);
        return result;
    }
}
package block;


public class Transaction {

    private String sourceName;
    private String destinationName;
    private Long sum;
```

```
public String getSourceName() {
    return sourceName;
}

public void setSourceName(String sourceName) {
    this.sourceName = sourceName;
}

public String getDestinationName() {
    return destinationName;
}

public void setDestinationName(String destinationName) {
    this.destinationName = destinationName;
}

public Long getSum() {
    return sum;
}

public void setSum(Long sum) {
    this.sum = sum;
}

public Transaction(String sourceName, String destinationName, Long sum) {
    this.sourceName = sourceName;
    this.destinationName = destinationName;
    this.sum = sum;
}
```

```
    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;

        Transaction that = (Transaction) o;

        if (sourceName != null ? !sourceName.equals(that.sourceName) : that.sourceName !=
null) return false;
        if (destinationName != null ? !destinationName.equals(that.destinationName) :
that.destinationName != null)
            return false;
        return sum != null ? sum.equals(that.sum) : that.sum == null;
    }

    @Override
    public int hashCode() {
        int result = sourceName != null ? sourceName.hashCode() : 0;
        result = 31 * result + (destinationName != null ? destinationName.hashCode() : 0);
        result = 31 * result + (sum != null ? sum.hashCode() : 0);
        return result;
    }
}
```

```
package dataset;
import javax.crypto.Cipher;
import javax.crypto.spec.SecretKeySpec;
import sun.misc.BASE64Encoder;
public class AesEncryption {

    byte[] keyBytes = new byte[] {
                0x73, 0x2f, 0x2d, 0x33, (byte) 0xc8, 0x01, 0x73, 0x2b, 0x72,
          0x06, 0x75, 0x6c, (byte) 0xbd, 0x44, (byte) 0xf9, (byte) 0xc1
   };
    SecretKeySpec skeySpec;
    Cipher cipher;
    public AesEncryption() {
          skeySpec = new SecretKeySpec(keyBytes, "AES");
          try {
                cipher = Cipher.getInstance("AES");
                cipher.init(Cipher.ENCRYPT_MODE, skeySpec);
          } catch (Exception e) {
                e.printStackTrace();
          }

    }

    public String  toEncrypt(byte[] content) {
          try {

                byte[] utf8 = cipher.doFinal(content);
                return (new BASE64Encoder()).encode(utf8);
```

```
            } catch (Exception e) {
                  e.printStackTrace();
                  return null;
            }

      }

}

package dataset;
import javax.crypto.Cipher;
import javax.crypto.spec.SecretKeySpec;
public class AESDecryption {
      byte[] keyBytes = new byte[] {
                  0x73, 0x2f, 0x2d, 0x33, (byte) 0xc8, 0x01, 0x73, 0x2b, 0x72,
            0x06, 0x75, 0x6c, (byte) 0xbd, 0x44, (byte) 0xf9, (byte) 0xc1
   };
      SecretKeySpec skeySpec;
      Cipher cipher;
      public AESDecryption() {
            skeySpec = new SecretKeySpec(keyBytes, "AES");
            try {
                  cipher = Cipher.getInstance("AES");
                  cipher.init(Cipher.DECRYPT_MODE, skeySpec);
            } catch (Exception e) {
                  e.printStackTrace();
            }
      }
      public String toDeycrypt(String  encrypted) {
```

```
            try {
                byte[] dec = new sun.misc.BASE64Decoder().decodeBuffer(encrypted);
                byte[] utf8 = cipher.doFinal(dec);
                return new String(utf8, "UTF8");
            } catch (Exception e) {
                e.printStackTrace();
                return null;
            }
        }
}
```

A.1 Sample Screenshots

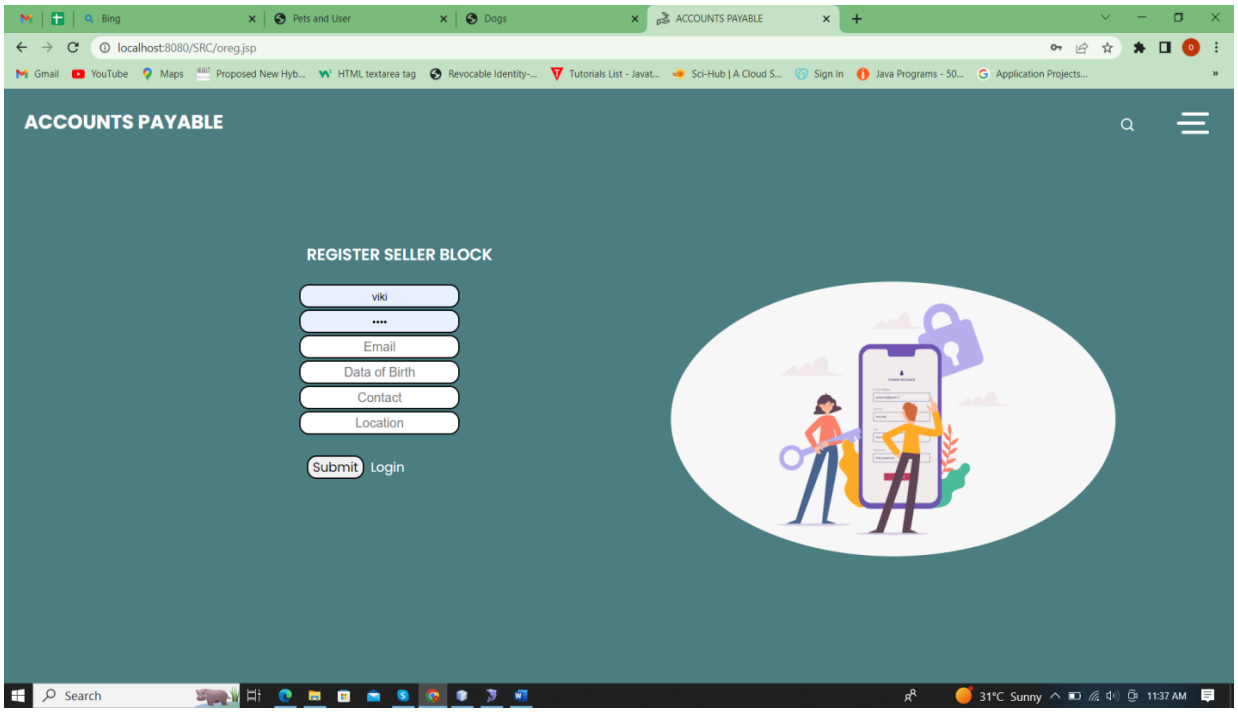

Fig 1. Seller Registration Page

`

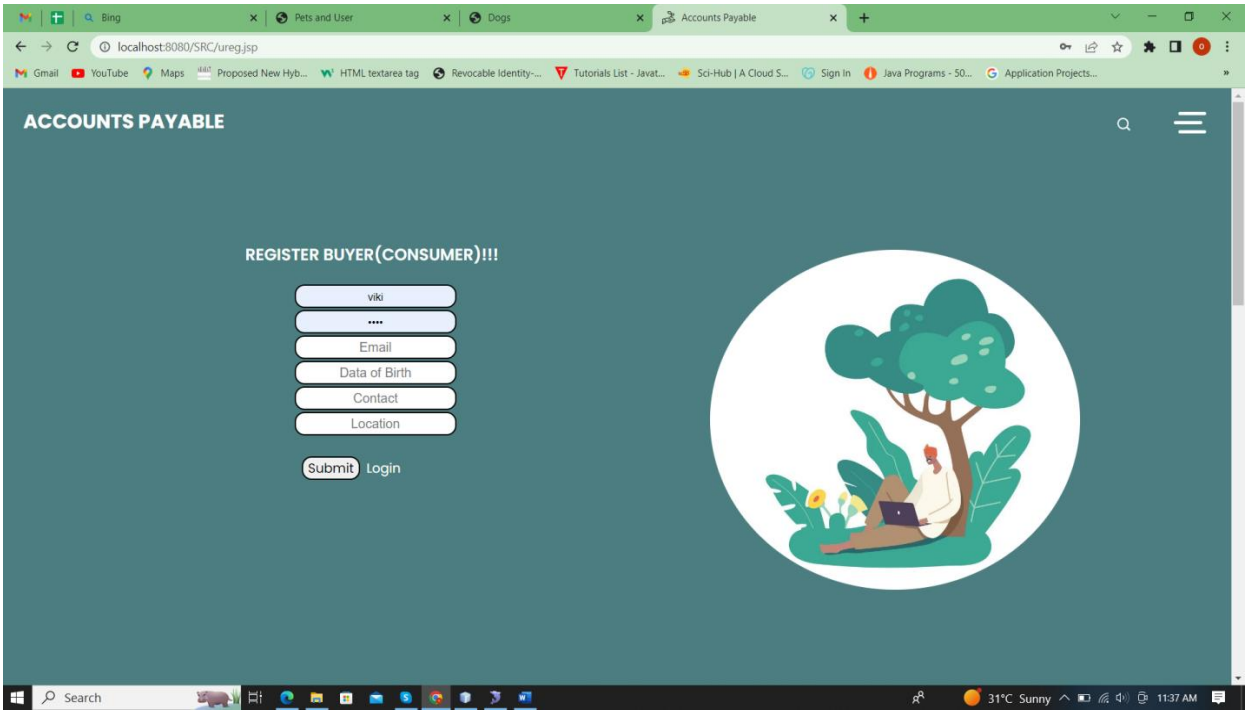

Fig 2. Buyer Registration Page

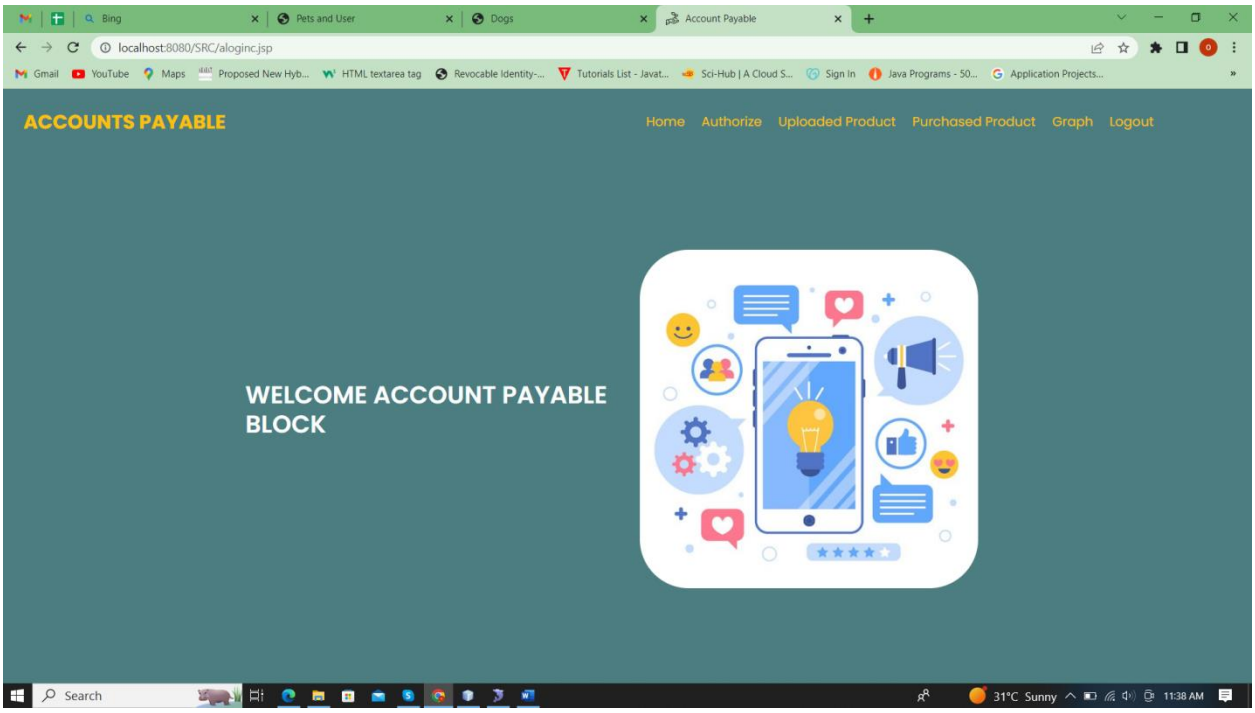

Fig 3.Account Payable

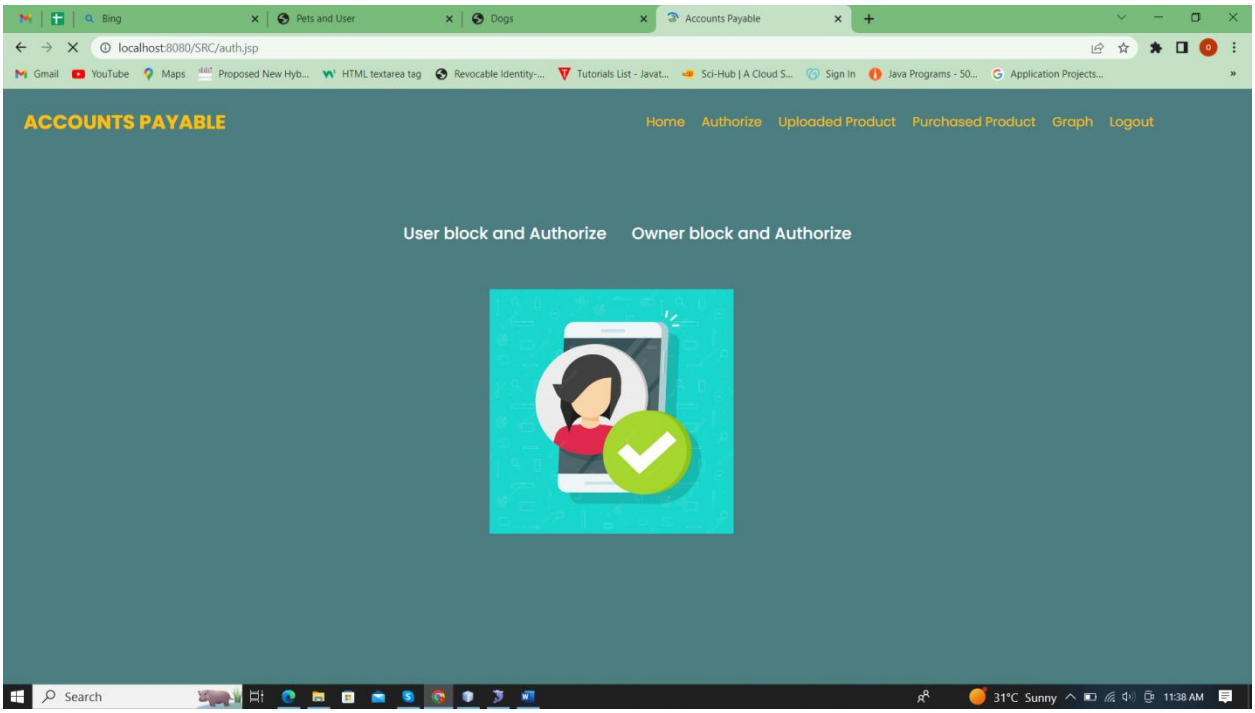

Fig 4. Authorization Page

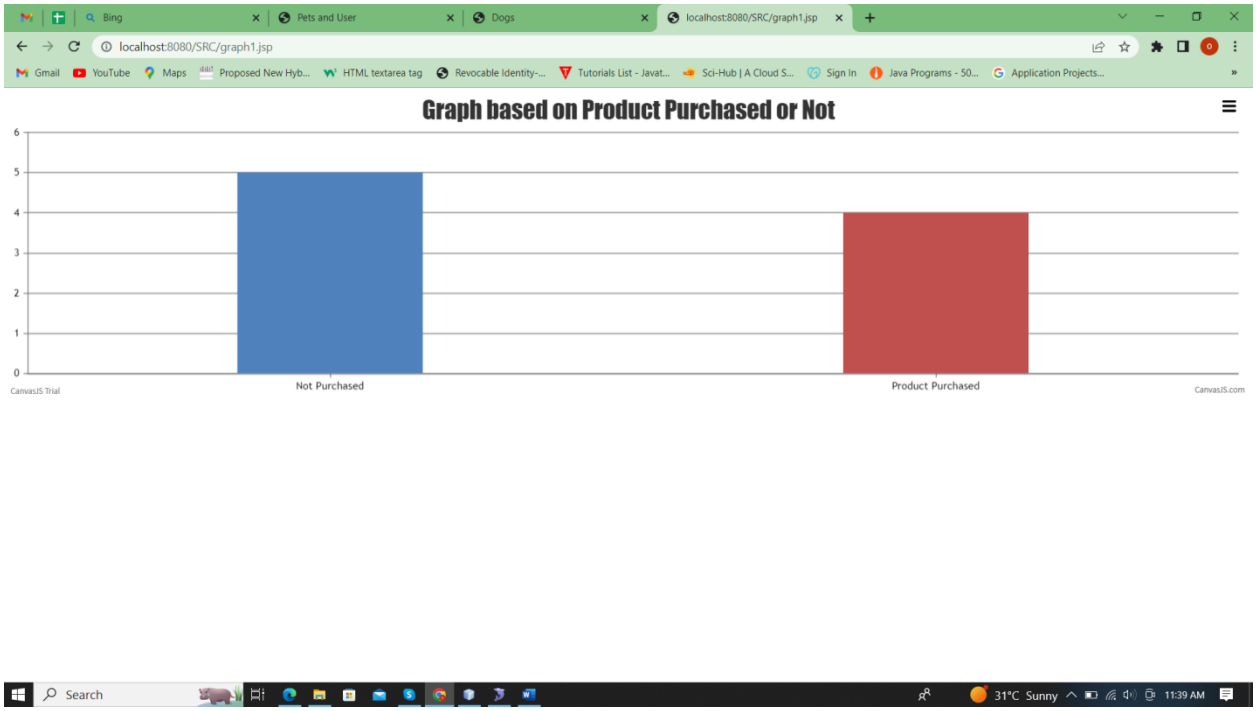

Fig 5. Graph Based on Product Purchased or Not

REFERENCES

[1] “Letter of credit,” https://en.wikipedia.org/wiki/Letter of credit/.

[2] “Invoice discounting,” https://en.wikipedia.org/wiki/Factoring (finance).

[3] “Reverse factoring (supply chain finance),” https://en.wikipedia.org/wiki/ Reverse factoring.

[4] “GS1 XML standards 3.2,” https://www.gs1.org/standards/gs1-xml/3-2/.

[5] “TradeLens,” https://www.tradelens.com.

[6] K. Narayanam, S. Goel, A. Singh, Y. Shrinivasan, S. Chakraborty, P. Selvam, V. Choudhary, and M. Verma, “Blockchain based e-invoicing platform for global trade,” in 2020 IEEE International Conference on Blockchain (Blockchain), 2020, pp. 385–392.

[7] http://ibm.biz/BlockchainSolutionFromWalmartCanadaAndDLTLabs.

[8] S. E. Chang, H. L. Luo, and Y. Chen, “Blockchain-enabled trade finance innovation: A potential paradigm shift on using letter of credit,” Sustainability, vol. 12, no. 1, p. 188, 2020.

[9] J. Chiu and T. V. Koeppl, “Blockchain-based settlement for asset trading,” The Review of Financial Studies, vol. 32, no. 5, pp. 1716– 1753, 2019.

[10] A. Bogucharskov, I. Pokamestov, K. Adamova, and Z. N. Tropina, “Adoption of blockchain technology in trade finance process,” Journal of Reviews on Global Economics, vol. 7, pp. 510–515, 2018.

[11] “we.trade,” https://we-trade.com/.

[12] “Invoice Processing by Accounts Payable,” https://www.purchasing.ucla. edu/accounts-payable/invoice-processing-by-accounts-payable.

[13] “Blockchain for Accounts Payable,” https://cporising.com/2017/05/25/ blockchain-for-accounts-payable-an-introduction/.

[14] “Blockchain in Accounts Receivable,” https://netsend.com/blog/ blockchain-

accounts-receivable/.

[15] “Our Prediction: Blockchain WILL Replace the Supplier Invoice,” https://www.netnetweb.com/content/blog/ blockchain-will-replace-the-supplier-invoice.