WOLKITE UNIVERSITY

COLLEGE OF COMPUTING AND INFORMATICS

DEPARTMENT OF COMPUTER SCIENCE

A

Project

On

Web Based Inventory Management System for Gold Water Company

Prepared by:

NAME	ID
1. Daniel Bekele	NSR/0586/13
2. Dominik Loang Peter	NSR/2542/13
3. Abdlwahid Akrem	NSR/2561/13
4. Dagim Tekuash	NSR/0567/13

Project Advisor: Mr. Kolana Korigne (MSc.)

Wolkite University, Wolkite, Ethiopia

May 10, 2024

WOLKITE UNIVERSITY

COLLEGE OF COMPUTING AND INFORMATICS

DEPARTMENT OF COMPUTER SCIENCE

Web Based Inventory Management System for Gold Water Company

SUBMITED TO DEPARTMENT OF COMPUTER SCIENCE IN PARTIAL FULFILMENT OF THE REQUIREMENT FOR THE DEGREE OF BACHLOER OF SCIENCE IN COMPUTER SCIENCE

BY

Name	**ID**
1. Daniel Bekele	NSR/0586/13
2. Dominik Loang Peter	NSR/2542/13
3. Abdlwahid Akrem	NSR/2561/13
4. Dagim Tekuash	NSR/0567/13

***Project Advisor*:** Mr. Kolana Korigne (MSc.)

DECLARATION PAGE

This is to declare that this project work which is done under the supervision of Mr. Kolana Korigne having the title Web Based Inventory Management System for Gold Water Company is the sole contribution of: Daniel Bekele, Dominik Loang Peter, Abdlwahid Akrem, and Dagim Tekuash. No part of the project work has been reproduced illegally which can be considered as Plagiarism. All referenced parts have been used to argue the idea and have been cited properly. We will be responsible and liable for any consequence if violation of this declaration is proven.

Date: _____

Group Members:

Full Name	**Signature**
1. Daniel Bekele	_____
2. Dominik Loang Peter	_____
3. Abdlwahid Akrem	_____
4. Dagim Tekuash	_____

APPROVAL FORM

This is to confirm that the project report entitled Web Based Inventory Management System for Gold Water Company, College of Computing and Informatics Department of Computer Science by: Daniel Bekele, Dominik Loang Peter, Abdlwahid Akrem, and Dagim Tekuash is approved for submission.

---	---	---
Advisor Name	Signature	Date
---	---	---
Department Head Name	Signature	Date
---	---	---
Examiner 1 Name	Signature	Date
---	---	---
Examiner 2 Name	Signature	Date
---	---	---
Examiner 3 Name	Signature	Date

ACKNOWLEDGEMENT

First of all, we would like to thanks to God for his giving us strength and health to complete this project.

Secondly, we would like to thank our Advisor Mr. Kolana Korigne for those restless support on an edition of our document, input to the quality of this document, heart full guidance, their valuable advice.

Thirdly, we would like to thank Gold Water Company Asset and Production Manage Ato Samuel Bekele. For his partial willingness of the interview, patience in answering to our numerous questions, that help us to get full information.

Fourthly, we would like to thank to the Wolkite University, Department of Computer Science for their acquiescence of free internet service that make us to accomplish our project document without any interference's.

At the last the group members would like to thank each other. The main contributors to do this document project are teamwork, friendship and the belief that we may achieve something we set out to do. We also hope that this project and the documentation may be testaments to our continued friendship and better work. And it may also be the essence of success and symbolic witness of what we are going to work.

TABLE OF CONTENT

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS AND ACRONYMS

CD …….....Compact Disk

CPU……. ……Central Processing Unit

CSS…….....Cascaded Style Sheet

DB……... …….Data Base

GB ……..…….. Giga Bytes

GBG……..……Gold Business Group

HTML……...Hyper Text Markup Language

HTTP…….....Hyper Text Transfer Protocol

MySQL…… …… Structural Query Language

OOA……. …….Object Oriented Analysis

OOD……...Object Oriented Design

PHP……..... Hypertext Preprocessor

RAM……Random Access Memory

UML……. ……. Unified Modeling Language

ABSTRACT

Inventory management system deals primarily with determining designed to track and manage the inventory of raw materials, packaging materials, and finished products related to the production of drinking water or within multiple locations of a store.

Our project is about Gold water inventory management system to automate the major operations found in the company. This project is done to develop online web application. This document has a detailed description about the current manual system and the newly developed system. The project identifies the problems and detailed overview of the company using different data gathering methods. The current system is a manual based system. We want to change the existing manual system by newly developed automated system. The system will be developed using PHP programming language and uses html language to design an effective transferring of data and also MYSQL as a back end database with CSS implemented in the interface and java script for validation. The project has scope and objective. The newly automated system provides fast and reliable service for the office employees and minimizing time and resource wastage. Our project identifies the weakness of the existing system and by highly investigating the problems of the existing system the document has clear and concise solutions for those problems. Under this document we specify the functional and nonfunctional requirements of the system and by carefully applying the functional requirements users can get quality service from the newly automated system. Our project specifies the hardware and software requirements. The project is done by all the group members and each member has our own specified task and we are responsible for our tasks. Generally, the main goal of web-based inventory management system is to shorten data processing time, to reduce errors, to improve the accuracy of input and to provide data reliability of the item.

CHAPTER ONE

1. INTRODUCTION

Technology is spreading its wing almost every walks of human life activities. Nowadays it is better if every activity is done using new technology in order to fulfill the need of human being, organization, enterprise.In the dynamic global landscape of the beverage industry, the critical need for efficient inventory management is universally recognized. This project is crafted to introduce a bespoke inventory management system tailored for Gold Water, a key player in the production and distribution of bottled mineral water with its headquarters situated in Ethiopia. Understanding the pressing demand for streamlined processes in this rapidly evolving sector, seeks to streamline and automate its current processes. The proposed system aims to revolutionize the management of inventory by providing accurate monitoring, and seamless control. By optimizing stock availability and reducing operational costs, this system is designed to enhance overall efficiency, ensuring that Gold Water remains agile

1.1 Background of Organization

Gold Water is a member of the Gold Business Group (GBG) engaged in the production of bottled non-alcoholic beverages, mainly potable water under the brand name Gold Water, at its factory site in Tatek Industrial Zone, with a mission to provide unique and high-quality non-alcoholic beverages locally and to neighboring countries and beyond by expanding its products and going everywhere possible. Located in Shaggar city administration, the site sits in beautiful highlands where water is abundant and surrounded by a high workforce equipped with the necessary expertise and good working culture from Ethiopia and China. It took 2 years to study the area, and the water resources and build the site that has been fully operational since November 2019.

1.2 STATEMENT OF PROBLEM

The current state of the company's operations presents a significant challenge. Manual processes, reliance on spreadsheets, and paper-based documentation create inefficiencies and limitations. A primary concern is the manual handling of inventory management, leading to inaccuracies in recording data.

Inaccuracy in recording data the manual system is prone to errors or mistakes made when entering, storing, or reporting data. This can occur for various reasons such as human error. It can negatively impact the validity of production and input material. Delaying to processing data the use of spreadsheets and paper-based documentation contributes to delays in updating inventory information. The manual recording and reconciliation processes, triggered by transactions and inventory changes, consume valuable time.Limited reporting capabilities With a manual system, it can be challenging to generate reports that provide a comprehensive overview of stock and making it difficult to make informed decisions.Limited security Manual systems are often not secure, making it easy for unauthorized individuals to access or manipulate sensitive information Manual and error-prone processes which can lead to delays in decision-making and increased costs. Manual data entry can result in data duplication, incorrect data entry, and data inconsistencies, which can lead to inaccurate reporting.

1.3 OBJECTIVE OF THE PROJECT

1.3.1 General Objective

The general objective of this project is to develop an efficient and user-friendly inventory management system for Gold Water that enhances inventory control and optimizes stock levels.

1.3.2 Specific Objectives

To achieve the general objective mentioned above the following are specific objective:

The specific objectives of the project include:

- To shorten data-processing time

- Minimize data inaccuracies and errors
- Improve the accuracy of input
- Enhance reporting capabilities
- Reduce costs and delays
- Enhance data security
- Give information easily and efficiently
- Implementing the project with the flexible user interface

1.4 FEASIBILITY STUDY

To bring the successful completion of this project goals and objectives the feasibility issues listed below will determine the project viability or the discipline of planning, organizing, and managing resources.

1.4.1 Economic Feasibility

The proposed system aims to reduce operational costs through better resource utilization and minimize losses due to stock-outs or overstock situations. The initial investment is expected to be outweighed by the long-term cost savings

1.4.2 Technical Feasibility

The chosen technology stack ensures scalability, security, and compatibility with manual existing Gold Water systems. The technical infrastructure will be designed to accommodate future expansions and advancements.

1.4.3 Operational Feasibility

The proposed system will be designed to align with Gold Water's existing operational processes and can be easily integrated into the current workflow and work on all operating system.

1.5 SCOPE AND LIMITATION OF THE PROJECT

1.5.1 Scope of the project

The proposed project aims to provide a comprehensive Inventory Management System that caters to businesses, customers, and workers associated with the Gold Water Project.

With a particular focus on Ethiopia, the system address the specific needs and challenges faced by stakeholders in the Gold Water sector

So the project focus on the following

- To manage user account
- To manage items
- To search items.
- To request an item online.
- To generate report for plant manager.
- To control production

1.5.2 Limitation of the project

1. The system is unable to provide real-time visibility into inventory levels and stock movements. This limitation implies that users of the system do not have access to up-to-date information on current inventory quantities or real-time tracking of stock movements. The system may have delays or rely on manual updates, which can hinder efficient inventory management.

2. Due to time constraints, the project was unable to develop an Android-based system. This limitation suggests that the project team did not have sufficient time to create a version of the system that is specifically designed and optimized for Android devices, which are widely used in the market. As a result, the system may be limited to certain platforms or devices, potentially impacting its accessibility and reach.

1.6 SIGNIFICANCE AND BENEFICIARY OF THE PROJECT

1.6.1 SIGNIFICANCE OF THE PROJECT

The successful implementation of the inventory management system will bring several benefits to Gold Water, including:

- Improved inventory accuracy and control
- Decreased instances of inventory shortages
- Optimal stock levels and reduced inventory carrying costs
- Increased operational efficiency

- User of the system can automatically update the information an item or change its status.
- Better security and management of item file.

1.6.2 BENEFICIARY OF THE PROJECT

The primary beneficiaries of the project include Gold Water's management team, inventory managers, procurement personnel, sales representatives.

- Business owner
- For employees

  Work their job without any difficulties.
- For developers

  We are getting knowledge by developing this system (Industry-specific Knowledge).

  We get satisfaction.

  We earn grade by deploying and maintaining if failure occurs.

1.7 METHODOLOGY

1.7.1 Requirement Gathering

Interviews: To obtain fundamental qualitative and background information about the existing manual system, team members conduct interviews with marketers and select customers. This help in understanding the services provided, as well as identifying any problems associated with the current environment.

Direct Observation: While interviews are crucial for information gathering, direct observation offers a simpler approach. Physically observing information that may not be obtained through interviews can be vital. This method is particularly important when individuals face difficulties in communication due to professional terms or language barriers.

Questionnaires: Recognizing the heavy workload of employees and managers, who may find it challenging to respond to inquiries, the team prepare a set of sample questions. This approach aims to efficiently gather precise information.

Existing Documents: To gain additional insights into the project, the team refer to previous documents related to the project. During the analysis of these documents, special attention given to those that can contribute valuable features to the project

1.7.2 System Analysis and Design

During the system analysis and design phase of a project, we will adopt the object-oriented approach, which involves examining requirements through the lens of classes and objects within the problem domain.

The rationale for employing this object-oriented methodology includes the following:

- Simplifying Complex Program Design and Implementation:

  Reason: Streamlining the process of designing and implementing intricate programs.

- Inheriting Properties from Super Classes:

  Reason: Facilitating the inheritance of properties defined in super classes.

- Method Re usability to Avoid Redundancy:

  Reason: Enabling the reuse of methods to prevent redundancy in the code.

- Enhancing Collaboration in Software Projects:

  Reason: Facilitating teamwork among designers and programmers within a unified software project.

- Encapsulation of Data and Functions for Easy Debugging:

  Reason: Ensuring that data and functions are encapsulated in objects for simplified debugging.

- Increasing Consistency Across Activities:

  Reason: Promoting consistency among analysis, design, and programming activities.

- Improving Communication Among Stakeholders:

Reason: Enhancing communication among users, analysts, designers, and programmers.

- Comprehensive System Modeling Before Development:

Reason: Allowing for a comprehensive system model before the actual development phase.

- Ease of Object Implementation Modification:

Reason: Simplifying the modification of object implementations due to loose coupling.

- Easy Structure Understanding Through Real-World Entity Representation:

Reason: Providing an easy-to-understand structure by representing real-world entities in the modeling.

- Direct Manipulation of Architectural Components:

Reason: Facilitating the direct manipulation of architectural components, supported by various object-oriented programming languages.

In summary, the adoption of the object-oriented approach offers numerous advantages, ranging from simplifying development processes to improving collaboration and communication among project stakeholders.

1.7.3 System Development Model

- In the realm of system development models, opting for the agile model is our preferred approach for crafting high-quality software, driven by various factors such as:
- Iterative and Incremental Progress:

  We embrace a methodology where iterative and incremental advancements are made within each phase of development.
- Customer Collaboration:

  A direct and collaborative engagement with customers is a pivotal aspect of our strategy.
- Time and Cost Efficiency:

Leveraging the Agile Development Model translates to significant savings in both time and financial resources.

- Flexibility and Control:

  This model affords us a high degree of flexibility and control, catering to the dynamic needs of developers and users alike.

- Constant Delivery of Working Software:

  A key advantage lies in the consistent and timely delivery of functional software, typically on a weekly or monthly basis.

- Frequent Entrepreneur-Developer Interactions:

  Regular and frequent interactions between entrepreneurs and developers foster a swift pace of software development.

- Focus on Deliverable over Documentation:

  Agile places a primary emphasis on tangible deliverable, minimizing bureaucratic paperwork in favor of tangible outcomes.

- Continuous Collaboration Among Stakeholders:

  The ongoing interaction among customers, developers, and testers is a fundamental principle, ensuring continuous alignment and feedback throughout the development process.

1.7.4 Testing Methodology

Black Box Testing:-we use black box testing because it requires the tester to understand what the program is supposed to do, but not how it works. It performs by the end user and tester, testing is based on external exception but not on internal behavior. In case of white box testing requires the tester to know and understand how the software works. It performed only by developer and tester and testing is based on the internal behavior of the system. Black box testing is less time consumable than white box testing.

1.7.5 Development Tools and Technology

1.7.5.1 Front - End Technology

HTML: - Client-side coding, and it is the fundamental coding language that creates and organizes element such as form, button.

CSS: - for the formatting of the web. For example, style any HTML element to change its dimensions, colors, borders, spacing, and so on.

JavaScript: - used to input validation checking that fields have content if they are not to be left empty, ensuring that email addresses conform to the proper format, and ensuring that the values entered are within expected bounds.

Bootstrap (for front-end CSS decoration)

1.7.5.2 Back - End Technology

PHP: - server-side scripting language. And it is Platform independence, it can work with a variety of platforms. Also, it can run almost on any platforms.

MySQL: -used to create a database. They are some database like Oracle, SQL but use MySQL because it is fast, free to use, and supports more of the standard DB functions than others.

1.7.5.3. Documentation and Modeling Tools

Microsoft office word: -It is very useful because it takes less time to write and format the text, communicative effectively smart diagram and chart tools, quickly assemble the document. By looking its useful properties, we used Microsoft office word to type our project work to get all the above.

Power point: - was used to present the document forms that can be understood easily. We also used it to present our presentation in a short and brief way.

Draw Io: - used to design the UML diagram.

1.7.5.4. Deployment Environment

This Inventory Management System used or deployed for Gold Water Company office.

1.8 OPERATING ENVIRONMENT

- Hardware environment
  - Processing power
    - ✓ 64 bit operating system
    - ✓ Intel(R) core™i3 CPU 550@3.20GHz
  - Memory & Secondary storage
    - ✓ RAM: 4GB and above

- ✓ 500GB Hard-disk: and swap space (if RAM is insufficient).

- Peripherals
  - ✓ CD-ROM drives,
  - ✓ Network devices, etc.

1.9. DOCUMENT ORGANIZATION

The document outlining the proposed system structured into the following chapters, each presenting its content in the following manner:

Chapter One: In this chapter, we provide an overview of the existing system, discussing its introduction, identified issues, general and specific objectives, and the methodology employed for data collection and analysis.

Chapter Two: This section delves into a detailed examination of the existing system, highlighting its users, major functions, and drawbacks.

Chapter Three: Focused on the proposed system, this chapter outlines both its functional and non-functional requirements.

Chapter Four: This chapter is dedicated to the analysis of the proposed system, encompassing the flow of events in the scenario, the use case model featuring major use cases, and detailed and dynamic models of the proposed system.

Chapter Five: Addressing system design, this section covers an overview of the system, design considerations, goals, trade-offs, system architecture, subsystem decomposition, persistent data management, and user interface design.

CHAPTER TWO

2. DESCRIPTION OF EXISTING SYSTEM

2.1. Introduction of the existing system

In some of the non-networked Inventory management of fixed Gold water is done manually, and this manual system hinders the organization from managing its entire items, production in a well-organized manner. This manual system doesn’t provide

adequate and appropriate control over fixed items, production and input output and it's the major problem to the operation of the gold water company.

The following are the problems that arise from the manual system:-

- Errors in process and miscalculation caused by spreadsheets and human errors.
- Change in tax rules and regulations not being applied when released.
- Lack of audit trails and reporting.
- Inability to easily change critical depreciation information: like depreciation methods.
- Difficulty in exchanging data with other production manager and input purchaser.
- Inability to attach other information on production records.

2.2. Users of Existing System

Purchasing department: is responsible for managing the procurement process and acquiring the necessary goods and services to support the company's operations related to water supply

Store Manager: -A person who is responsible to approve the request, managing the items to be taken by the production department, request for raw material to purchasing department and report to plant department.

Production manager: - A person who is responsible request to store manager, return left input to store manager, request technician, control output product and send the product to store.

Quality checker: - A person who is responsible to check the quality of raw material (input) and product before stored to stock.

Finance manager: - Responsible for reporting to company manager.

Admin: - A person who is responsible for managing user account.

Plant Manager: - Receive report from other department.

2.3. Major Functions of the Existing System

- Register new item within their property.
- Categorize and request item based on the type of usage in a different store.
- Generate reports about item
- Store all item list on paper.
- Requesting for items.
- Approve the request of item
- Transfer item from one department to another department.

2.4. Form and Other Document of the Existing System

There are some form that describe the existing system. You can see in Appendix A. Appendix A1: Item transferring form, Appendix A2: Product Rejection Form and Appendix A3: Store Request Form

2.5. Drawback of the Existing System

Difficult in receiving information and controlling the status of items. It takes longer time to do one specific job.Inefficiency in Workflows Manual processes are often time-consuming and inefficient, impacting the speed at which inventory-related tasks are performed.Existing system needs much time to search individual file and it requires more space to store all files manually. And it needs many human power and resource supply Error-Prone Processes Manual data entry and calculations increase the likelihood of errors, leading to inaccuracies in inventory records and financial reporting.Dependence on Spreadsheets Reliance on spreadsheets for calculations and record-keeping introduces complexities and increases the risk of formula errors.Risk of Data Loss Manual record-keeping increases the risk of data loss due to physical damage, misplacement, or other unforeseen events.

2.6. Business Rules of the Existing System

A business rule is successfully an operating standard or policies that we have tried to specify for both the existing system and the proposed system of the inventory management for gold water company must satisfy. The existing system has rules or principles those are:

Business rule 1: when new items are received, they are typically recorded and assigned a unique code or ID by the storekeeper.

Business rule 2 if one department wants to transfer item into another department both department means department who transferred material and department that take material must sign and fill the form.

Business rule 3: only employee of the company is allowed to take an authenticated material. **Business rule 4:** in order to get the item, user must get permission from the department and the store manager.

Business Rule 5: The inventory management system should enforce a periodic inventory audit, conducted by authorized personnel, to ensure the accuracy of recorded stock levels Discrepancies discovered during audits must be promptly investigated, documented, and reported to the relevant stakeholders for resolution.

Business Rule 6: The system should maintain a comprehensive transaction log for all inventory movements, including transferred items. This log should be regularly reviewed by the designated personnel to ensure transparency and accountability in the inventory management processes.

Business Rule 6: The system should maintain expiration of product and items and trigger Or enforce user to remove expired product.

CHAPTER THREE

3. PROPOSED SYSTEM

Our proposed web-based Inventory Management System for gold water company is aims to replace the current manual system. The system is designed to record the issuance of new items, requested items with appropriate specifications and categories. It facilitates the search for items available in the storehouse and streamlines the step-by-step processes involved in the delivery or return of items. Additionally, the system generates up-to-date reports for decision-makers to aid in budget allocation and control.

It incorporates database security measures to ensure the confidentiality and integrity of data. Each worker in the storehouse is assigned specific privileges, allowing them to perform authorized operations on the database.

3.1 Functional Requirements

Functional requirement is a specification of a function that the system must support.

Request Approval: The system should support customizable approval workflows based on the organization's hierarchy or business rules.

Request Rejection: Approvers should have the option to reject a request, providing a reason.

Report Viewing: The system should have a centralized repository for storing and organizing reports.

Item Searching: The system should provide a user-friendly search interface that is easily accessible from relevant pages or sections.

Item Management: The system shall provide a user interface for authorized users to register new items. The system shall allow authorized users to delete items, Authorized users shall be able to update existing item details through the system and the system shall provide a user-friendly interface for users to view detailed information about items.

Account Management: The system shall provide a user registration process for new users. Authorized administrators shall have the ability to delete user accounts, Users shall

be able to update their account information through a user profile interface and Users, based on their permissions, shall be able to view their account details through a user profile interface.

Report Generation: Users should have the ability to customize and save their own report templates.

View Announcement: -the system allows see announcement in order to register and participate in the tender.

Item Request: Users shall have the ability to submit item requests through a user-friendly interface.

Item Transfer: The system facilitates the transfer of items from one department to another.

User Login: All users have the ability to log in to the system.

3.2 Nonfunctional Requirements

Nonfunctional requirements are constraints on the operation of the system that are not directly related to its functions.

The following outlines the nonfunctional requirements for the proposed system.

3.2.1. User Interface and Human Factors

The proposed system boasts a user-friendly graphical user interface that is intuitive for users. This is achieved through the utilization of HTML, CSS, and JavaScript for the front-end, coupled with System Analysis and Design principles.

3.2.2. Hardware Considerations

The proposed system is compatible with various hardware configurations. Implemented using the PHP server-side scripting language, which supports platform independence, the system can seamlessly run on any platform.

3.2.3. Security Measures

Our system ensures a high level of security by preventing unauthorized access to secured system pages. External security is reinforced through role-based authorization techniques and authentication, restricting system functionality and information access.

3.2.4. Performance Considerations

The proposed system is designed for optimal performance, ensuring speed, reliability, minimal response time, efficient user service, availability, and support for a minimum of a lot of simultaneous users.

3.2.5. Error Handling and Validation

The system remains operational even when user's input unsupported formats, thanks to an exception-handling mechanism implemented using JavaScript. The system handles various exceptions, providing clear error messages for actions such as incorrect data type, entry of empty strings, and incorrect format.

3.2.6 Quality issues

Ensuring the correctness and currency of information in the database is crucial. The system promotes reliability through a permanent database storage solution and guarantees availability within internet access. Robustness is a key attribute, allowing the system to adapt to changes and operate under stress or tolerate unpredictable inputs from users.

3.2.6. Backup and Recovery

The proposed system maintains data backups through copies stored on different devices, including hard disks. Nightly backups ensure data integrity and facilitate recovery in the event of server failures we used mysql database backup to restore from mysql to local machine..

3.2.7. Resource Requirements

Development of the system necessitates specific resources, including server specifications for Apache, client requirements for browsers and CPU, and essential software tools such as Adobe Photo shop. Time is also a crucial resource, alongside hardware like computers, flash disks, and CD/DVD storage for data backup.

3.2.8. Physical Environment

The inventory management system is set up at the Gold Water Company's store office, strategically positioned to handle potential risks and external factors.

3.2.9. Documentation

The new system provides comprehensive documentation, including help contents and tips, to ensure maintainability. Users are guided on system usage through detailed information and instructions provided by the system.

CHAPTER FOUR

4. SYSTEM ANALYSIS

4.1 SYSTEM MODEL

This chapter is dedicated to the analysis of the proposed system through the application of various Unified Modeling Language (UML) analysis modeling techniques. These techniques encompass the utilization of use case diagrams, the creation of use case descriptions (scenarios), development of sequence diagrams, and formulation of activity diagrams, construction of analysis class diagrams, and the generation of user interface prototypes. Following the identification of actors and use cases, the subsequent step involves the refinement and development of the identified use cases, accompanied by the articulation of textual descriptions (scenarios).The Sequence diagram is then crafted based on the evolved use cases designed for the proposed system. Additionally, activities within the system will be visually represented through the incorporation of activity diagrams.

4.1.1 Use case model

In the process of system modeling, capturing the dynamic behavior emerges as a paramount consideration. Dynamic behavior, in this context, refers to the operational or running state of the system. It is imperative to underscore that an exclusive focus on static behavior is insufficient for comprehensive system modeling; dynamic behavior assumes greater significance in this regard. Within the Unified Modeling Language (UML), specific diagrams cater to the modeling of dynamic nature, with the use case diagram being a prominent example. In delineating the dynamic attributes of a system through the

use case diagram, it is essential to acknowledge the presence of internal or external factors driving interactions. These influential factors, referred to as actors, form an integral part of use-case diagrams. Consequently, use-case diagrams encapsulate actors, use cases, and their interrelationships, providing a graphical representation to model the system or subsystem of an application. It is noteworthy that each individual use case diagram encapsulates a distinct functionality of the system. The ensuing discussion will expound upon the utilization of the following use case diagrams in the modeling of the system

4.1.1.1. Use Case Identification

Use case identification describes the use case and actor involve in the use case. Each use case describes the functionality of the proposed system, and actors represent the people or systems that provide or receive information from the system. The most important and basic use cases and actors of our system are the following:

Table 4.1 Use case identification

Actor Name	**Specific Role of Actors**
System Admin	Manage account: create account, update account and delete account
Store manager	Approve the request, manage items, report and search items
Quality checker	Check quality
Store keeper	Place items
Technician	Reject machine and approve machine
Purchaser	Accept request ,reject request ,generate report And announce

Production manager	Request for items, manage product, request technician, return left input and send product

4.1.1.2 Use Case Diagram

Use Case diagrams serve as a visual representation illustrating the dynamic interactions among use cases, actors, and their interrelations. Within this framework, use cases serve to depict the functionality inherent in the system, delineating the system's requirements from the user's perspective. Actors, on the other hand, represent the individuals or systems engaging in the exchange of information with the system, forming a pivotal component of the overall system dynamics.

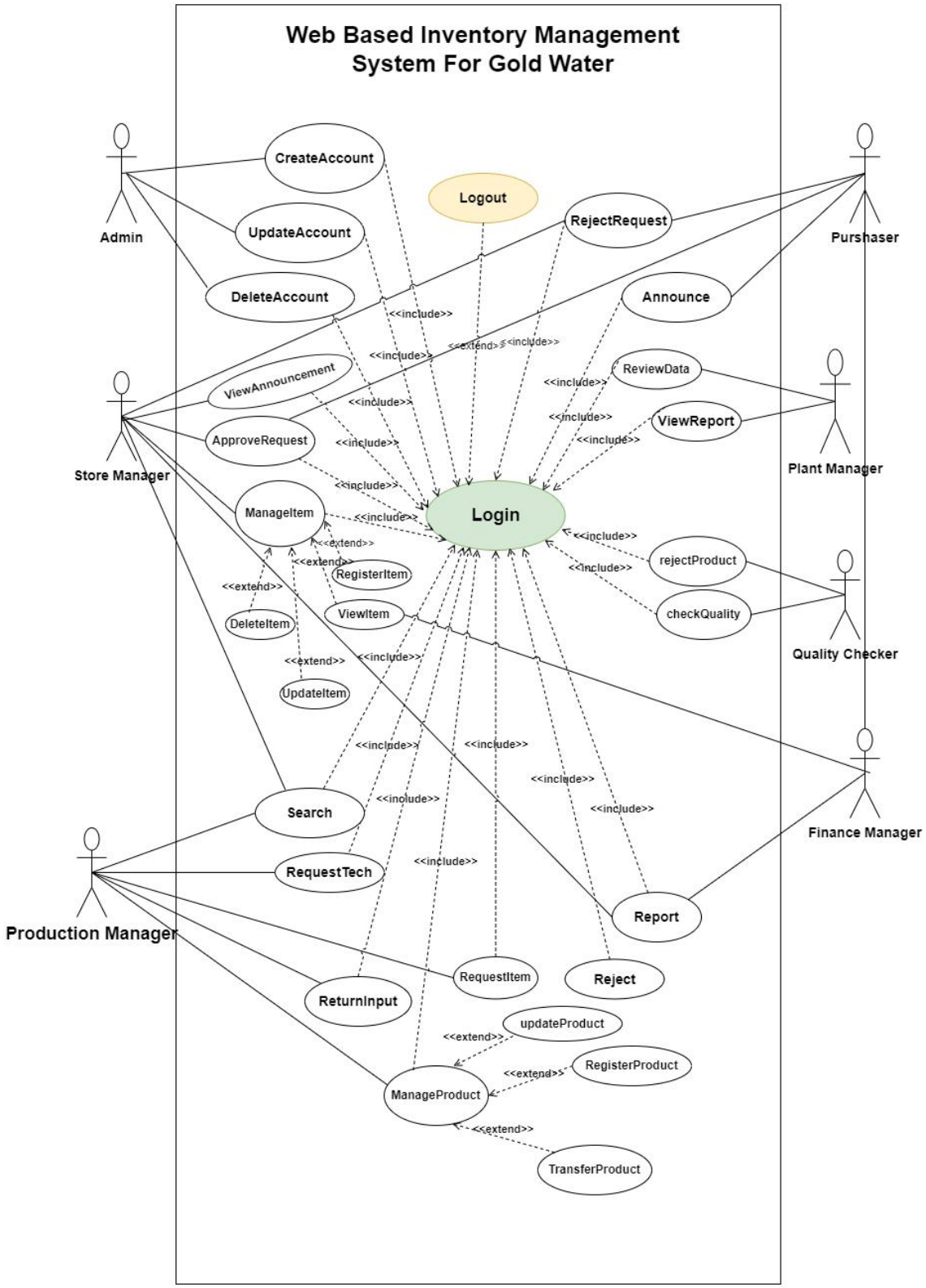


Figure 4. 1 Use case diagram

4.1.1.3 Use Case Description

Table 4. 2 Use case Description for login

Use Case ID	01	
Use Case Name	Login	
Actor	All Actor	
Pre-Condition	The user must have a username, password and Role.	
Description	This use case describes how to authenticate and validate system members and admin, so that only members and admin with correct username and password can access the system	
Normal Flow:	User Action	System Response
	1. Users open login page 3. The users fill the login form by providing phone/email and password 4. They click login button. 6. The user gets access to corresponding homepage	2. The system responses by display the login page 5. System checks whether all the fields are filled accurately and are valid and verify from database 7. Use case ends

Alternative Flows:	Alternate Course 5.1. If user enters invalid email/username or tries to login without filling out all the provided fields, try again or fill out all field 5.2 The System determines the Alternate course of Action invalidity of email/username and/or give a warning message to user to fill all available fields. Go to step 3.
Post Conditions	The user will be logged into the system if the user is registered in the system and inserted correct username and password

Table 4.3 Use case Description for Create Account

Use Case ID	02	
Use Case Name	Create Account	
Actor	Admin	
Pre-Condition	All Building Owners must browse to the system and open create account page	
Description	This use case describes how the owner create account (register) to the system.	
Normal Flow:	User Action	System Response

	1. Owners' open create account page 3. The owners fill the register form by providing all needed information. 4. They click sign up button. 7. The owner gets in to payment gateway integration).	2. The system responses by display the create account page 5. System checks whether all the fields are filled accurately and are valid. 6. store to database 8. Use case ends
Alternative Flows:	Alternate Course 5.1. If the owner enters invalid input or tries to register without filling out all the provided fields, try again or fill out all field	
Post Conditions	System should have successfully created a new user account with the provided information.	

Table 4.4 Use case Description for Delete Account

Use Case ID	03
Use Case Name	Delete Account
Actor	Admin
Pre-Condition	The system admin must be logged into the system
Description	This use case describes how the system admin will able to delete a user account

Normal Flow:	User Action	System Response
	1. The System admin will navigate to menu bar and click on accounts. 3. the system admin will be able to search for the account he/she wants with in the table displayed and can click on remove to delete individual account he/she wants if there is proper cause for termination	2. The system will load and display all the list of account that were created within the system in table format. 4. System will take the ID of the account and remove it from the database and any related data with the account. 5. Use Case exit.
Alternative Flows:	The system will display error message and reload the page and not make any changes to the request account and will continue with the same state.	
Post Conditions	System should have successfully, the system admin has successfully Delete a user's account	

Table 4. 5 Use case Description for Update Account

Use Case ID	04
Use Case Name	Update Account

Actor	Admin	
Pre-Condition	The system admin must be logged into the system	
Description	This use case describes how the system admin will able to update a user account	
Normal Flow:	User Action	System Response
	1. The System admin will navigate to menu bar and click on accounts. 3. the system admin will be able to search for the account he/she wants with in the table displayed and can click on update to Update individual account he/she wants if there is proper cause for termination	2. The system will load and display all the list of account that were created within the system in table format. 4. System will take the ID of the account and remove it from the database and any related data with the account. if ot#4.1 5. Use Case exit.
Alternative Flows:	The system will display error message and reload the page and not make any changes to the request account and will continue with the same state.	

Post Conditions	System should have successfully, the system admin has successfully update a user's account

Table 4. 6 Use case Description for Approve Request

Use Case ID	05	
Use Case Name	Approve Request	
Actor	Store Manager , Technician and purchaser	
Pre-Condition	The Store Manager , Technician and purchaser must have their account and must be logged into the system	
Description	This use case describes how the Store Manager and Technician will able to approve request	
Normal Flow:	User Action	System Response
	1.The user open home page 2. The actor should log in to the system. 4. The actor clicks approve request link. 6. The actor fill required information. 7. The click approve request	3. The system displays actor page. 5. The system displays request 8. The system validates the process and display successfully approved message. 9. Use Case exit.

	button.	
Alternative Flows:	If the request is not approved the system display the request is rejected or not approved, please try again and the actor goes into step 6.	
Post Conditions	Approve and save the data to the store database.	

Table 4.7 Use case Description for Reject Request

Use Case ID	06
Use Case Name	Reject Request
Actor	Store Manager , Technician and purchaser
Pre-Condition	The Store Manager , Technician and purchaser must have their account and must be logged into the system

Description	This use case describes how the Store Manager , Technician and purchaser will able to approve request	
Normal Flow:	User Action	System Response
	1.The user open home page 2. The actor should log in to the system. 4. The actor clicks approve request link. 6. The actor fill required information. 7. The click Reject request button.	3. The system displays actor page. 5. The system displays request 8. The system validates the process and display successfully Reject message. 9. Use Case exit.
Alternative Flows:	If the request is not Rejected the system display the request is not rejected, please try again and the actor goes into step 6.	
Post Conditions	Reject and save the data to the store database.	

Table 4.8 Use case Description for Manage Item

Use Case ID	07	
Use Case Name	Manage Item	
Actor	Store Manager	
Pre-Condition	The Store Manager must have their account and must be logged into the system	
Description	This use case describes how the Store Manager will be manage item in the stock	
Normal Flow:	User Action	System Response
	1.The user open home page 2. The actor should log in to the system. 4. The actor clicks item link. 6.The actor manage item	3. The system displays actor page. 5. The system displays item information 7. The system validates the process and display successfully 8. Use Case exit.
Alternative Flows:	If the item information is not managed properly, please try again and the actor goes into step 6.	

Post Conditions	Information managed and save the data to database.

Table 4.9 Use case Description for Manage Product

Use Case ID	08	
Use Case Name	Manage Product	
Actor	Production Manager	
Pre-Condition	The Production Manager must have their account and must be logged into the system	
Description	This use case describes how the Production Manager will be manage the product	
Normal Flow:	User Action	System Response
	1.The user open home page 2. The user should log in to the system. 4. The user clicks product link. 6. The user manage product	3. The system displays user page. 5. The system displays product information 7. The system validates the process and display successfully 8. Use Case exit.

Alternative Flows:	If the product information is not managed properly, please try again and the actor goes into step 6.
Post Conditions	Information managed and save the data to database.

Table 4. 10 Use case Description for Search

Use Case ID	09	
Use Case Name	Search	
Actor	Store Manager and Production Manager	
Pre-Condition	The Production Manager and store manager must have their account and must be logged into the system	
Description	This use case describes how the Production Manager and store manager will be search	
Normal Flow:	User Action	System Response
	1. User will type search query 3. The user click on of the suggestion or search button	2. The system will automatically suggest after each query 4. The system will display search result from database 5. Use case ends

Alternative Flows:	If user inputs invalid query or if the searched query result doesn't found in database, appropriate message will be shown
Post Conditions	The user is able to see searched information

Table 4.11 Use case Description for Report

Use Case ID	10	
Use Case Name	Report	
Actor	Store Manager, Production Manager and Finance Manager	
Pre-Condition	The Production Manager, Finance Manager and store manager must have their account and must be logged into the system	
Description	This use case describes how the Production Manager, Finance Manager and store manager will be report	
Normal Flow:	User Action	System Response
	1.The user open home page 2. The user should log in to the system. 4. The user clicks report link. 6. The user fill necessary information	3. The system displays user page. 5. The system displays report criteria. 7. The system validates the process and display successfully 8. Use Case exit.

Alternative Flows:	If the report information is not filled properly, please try again and the actor goes into step 6.	
Post Conditions	The user is able to generate report	

Table 4.12 Use case Description for check quality

Use Case ID	11	
Use Case Name	Check quality	
Actor	Quality checker	
Pre-Condition	The Quality checker must have their account and must be logged into the system	
Description	This use case describes how the Quality checker check the purchased item	
Normal Flow:	User Action	System Response

	1.The user open home page 2. The user should log in to the system. 4. The user clicks Item link. 6. The user approve the quality	3. The system displays user page. 5. The system displays item list 7. The system validates the process and display successfully 8. Use Case exit.
Alternative Flows:	If the item information not checked, please try again and the actor goes into step 6.	
Post Conditions	The user is able to check items quality	

Table 4.13 Use case Description for Request

Use Case ID	12	
Use Case Name	Request	
Actor	Production Manager	
Pre-Condition	The Production Manager must have their account and must be logged into the system	
Description	This use case describes how the Production Manager Request for Item and for technician	
Normal Flow:	User Action	System Response

	1.The user open home page 2. The user should log in to the system. 4. The user clicks request link. 6. The user select and sent a request	3. The system displays user page. 5. The system displays request option. 7. The system validates the process and display successfully 8. Use Case exit.
Alternative Flows:	If the request information is not filled properly, please try again and the actor goes into step 6.	
Post Conditions	The user is able to request for item and for technician	

Table 4.14 Use case Description for View Report

Use Case ID	13
Use Case Name	View Report
Actor	Plant Manager
Pre-Condition	The Plant Manager must have their account and must be logged into the system
Description	This use case describes how the Plant Manager will be View report

Normal Flow:	User Action	System Response
	1.The user open home page 2. The user should log in to the system. 4. The user clicks View report link. 6. The user View report	3. The system displays user page. 5. The system displays report List. 7. Use Case exit.
Alternative Flows:	If the report information is not filled properly, please try again and the actor goes into step 4.	
Post Conditions	The user is able to View report	

Table 4.15 Use case Description for Announce

Use Case ID	14
Use Case Name	Announce
Actor	Purchaser
Pre-Condition	The purchaser must have their account and must be logged into the system
Description	This use case describes how the purchaser will be announce information

Normal Flow:	User Action	System Response
	1.The user open home page 2. The user should log in to the system. 4. The user clicks announce link. 6. The user announce	3. The system displays user page. 5. The system displays announce page. 7. Use Case exit.
Alternative Flows:	If the announce information is not filled properly, please try again and the actor goes into step 4.	
Post Conditions	The user is able to announce information	

Table 4.16 Use case Description for Return Input

Use Case ID	15
Use Case Name	Return Input
Actor	Production Manager
Pre-Condition	The Production Manager must have their account and must be logged into the system.
Description	This use case describes how the Production Manager will be return input that left from production.

Normal Flow:	User Action	System Response
	1.The user open home page 2. The user should log in to the system. 4. The user clicks return input link. 6. The user return input	3. The system displays user page. 5. The system displays return input page. 7. Use Case exit.
Alternative Flows:	If the return input is not filled properly, please try again and the actor goes into step 4.	
Post Conditions	The user is able to return input to store	

4.1.1.4 Use case Scenario

Tells who is using the system and what they are trying to accomplish. Provides a realistic, fictional account of a user's constraints: when and where they are working, why they are using the system, and what they need the system to do for them. Describes any relevant aspects of the context in which the user is working with the system, including what information the user has on hand when beginning to use the system. The following are describing scenario of how the users use the systems to perform operations.

Scenario Name: User Login

Participating Actor: User

Flow of Events: To initiate the login process, the user begins by accessing the system's web page. Subsequently, the system presents the user with a login page. The user

proceeds to enter their username and password, followed by clicking the login button. The system then performs validation checks on the provided username and password. In the event of invalid credentials, the system responds with the message, "These credentials do not match our records." Conversely, if the entered data is valid, the system grants access and displays a page showcasing the user's assigned privileges. The use case concludes at this point.

Scenario Name: Account Creation Actor

Participating Actor: Administrator

Flow of Events: To initiate the account creation process, the administrator begins by launching the web browser. Afterward, they navigate to access the login page. Here, the administrator proceeds to input the user's name, password, and selects the designated role. Ensuring accuracy, the administrator enters valid information and clicks on the 'login' button. Upon successful login, the administrator gains access to the admin page, which encompasses various tasks, including a link to manage accounts.

To create an account for authorized users, the administrator clicks on the "manage account" link. This action opens the manage account page, where the admin can click on the "create account" button. Subsequently, a form titled "Create Account" is presented, prompting the administrator to input details such as the user's first name, last name, employee ID, email, phone number, block number, office number, registration date, status, username, and password.

Once the required information is accurately filled out, the administrator clicks the "create" button. If the provided user information is valid, the system proceeds to save the details into the database and generates a success message, indicating that a user account has been successfully created. Conversely, if there are any discrepancies or invalid information, the system displays an error message conveying "Invalid user information."

Scenario Name: Item Registration

Actors: Store Manager

The Store Manager initiates the login process in the system. After a successful login, the system presents the home page. The Store Manager navigates to the "Register Item" link, leading to the display of the "Register Item" form. Completing the form with detailed item information, the Store Manager clicks the "Register" button. The system then conducts validation checks on the provided information. Upon successful validation, the system displays a confirmation message indicating successful registration.

Extension: Incorrect Item Information

In the event of incorrect item information, the process resumes at the step of completing the item details, allowing the Store Manager to correct and re-submit the information.

4.2 Object Model

An object model is a conceptual framework that represents the structure and behavior of objects within a system. Objects encapsulate data and methods, interacting with one another to achieve specific functionalities. This model simplifies software design by organizing code into reusable and modular components, fostering efficient development and maintenance in object-oriented programming paradigms.

4.2.1 Class Diagram

A class diagram is a visual representation in UML depicting the structure and relationships of classes within a system. It illustrates the blueprint for object-oriented design, showcasing classes, their attributes, and methods. Class diagrams define the static aspects of a system, highlighting inheritance, associations, and multiplicity relationships between classes. Each class embodies the characteristics and behaviors of objects, promoting a clear understanding of system architecture. This diagram aids software developers, architects, and stakeholders in comprehending the overall structure of a software application, facilitating effective communication and collaboration during the design and development phases of object-oriented programming projects.

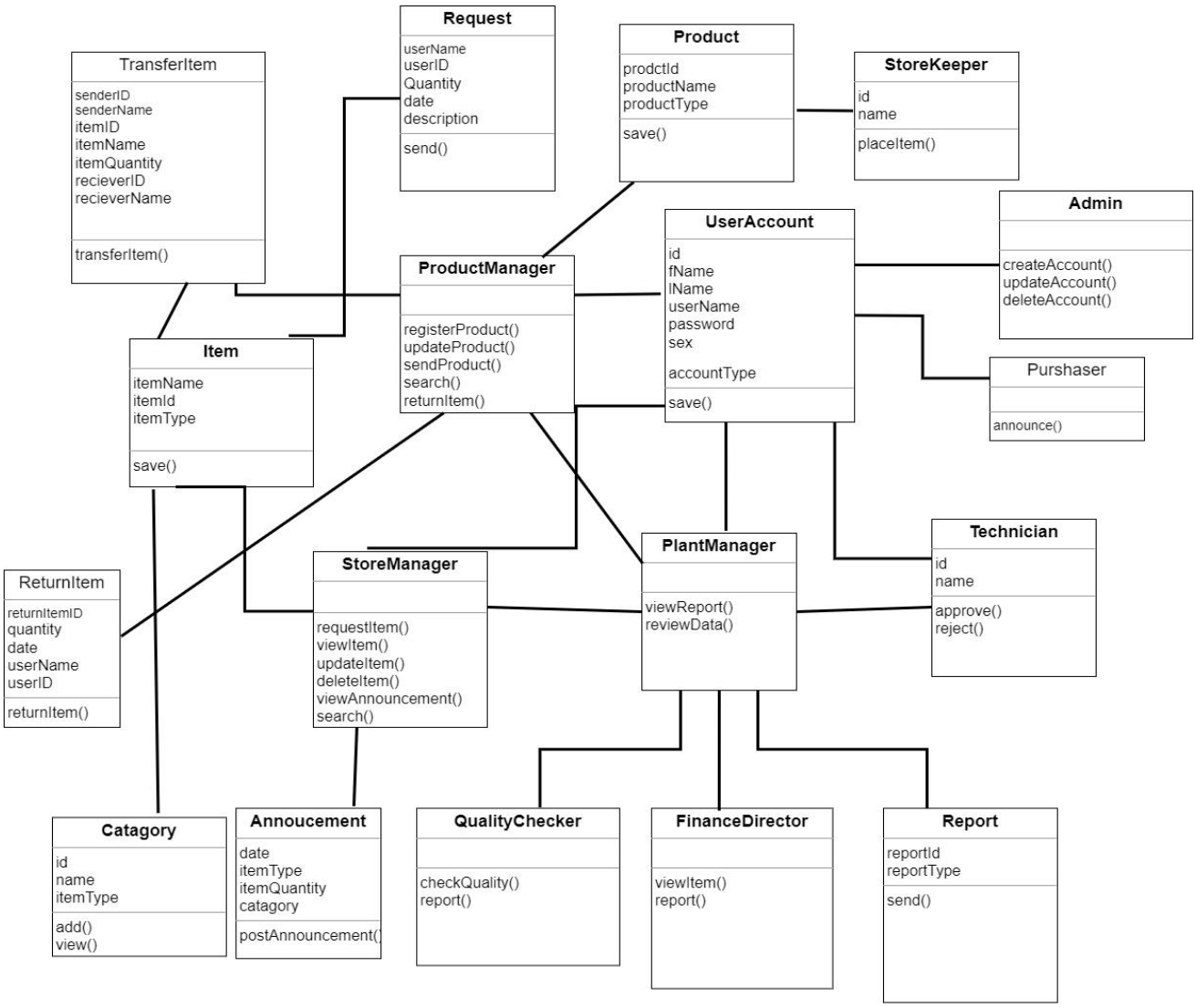


Figure 4. 2 Class diagram

4.2.2 Data Dictionary

Data dictionary of inventory management system for gold water database table are the following. Each table has its own attributes (dictionary of field elements) and each has a foreign key-primary-key relationship with other tables of the database. So we have organized the dictionary of IMS database tables as shown below.

Table 4.17 Data dictionary for Register item

Table Name: Register Item

Attributes	**Data Types**	**Constraint**
Item Name	varchar(20)	Not null
Item ID	int	Primary Key
Item Type	varchar(20)	Not null
Quantity	Int(10)	Not null
Reg_Date	Date	Not null

Table 4.18 Data dictionary for Request

Name Table: Request		
Attributes	Data Type	Constraint
UserName	varchar(20)	Not null
Item	varchar(20)	Not null
ItemQuantity	int(10)	Not null
ProductMnager ID	int	Foreign
Date	Date	Not null

Table 4.19 Data dictionary for Transfer Item

Name Table: TransferItem		
Attributes	Data Type	Constraint
senderName	Varchar(20)	Not null
productID	int	Foreign key (refers itemID)
productName	Varchar(20)	Not null
productQuantity	Int(20)	Not null

Table 4.20 Data dictionary for Return Item

Table Name: ReturnItem		
Attributes	Data Type	Constraint
returnItemID	int	Foreign key (refers itemID)
quantity	Int(10)	Not null
returnDate	Date	Not null
userID	int	Foreign key(refers itemID)
description	Varchar(30)	Not null

4.3 Dynamic Model

4.3.1 Sequence Diagram

A sequence diagram is a type of UML (Unified Modeling Language) diagram that illustrates the interactions and order of messages exchanged between objects or components in a system over a specific period of time. Sequence diagrams are particularly useful in the analysis level of software development, where the emphasis is on understanding and modeling the behavior of the system.

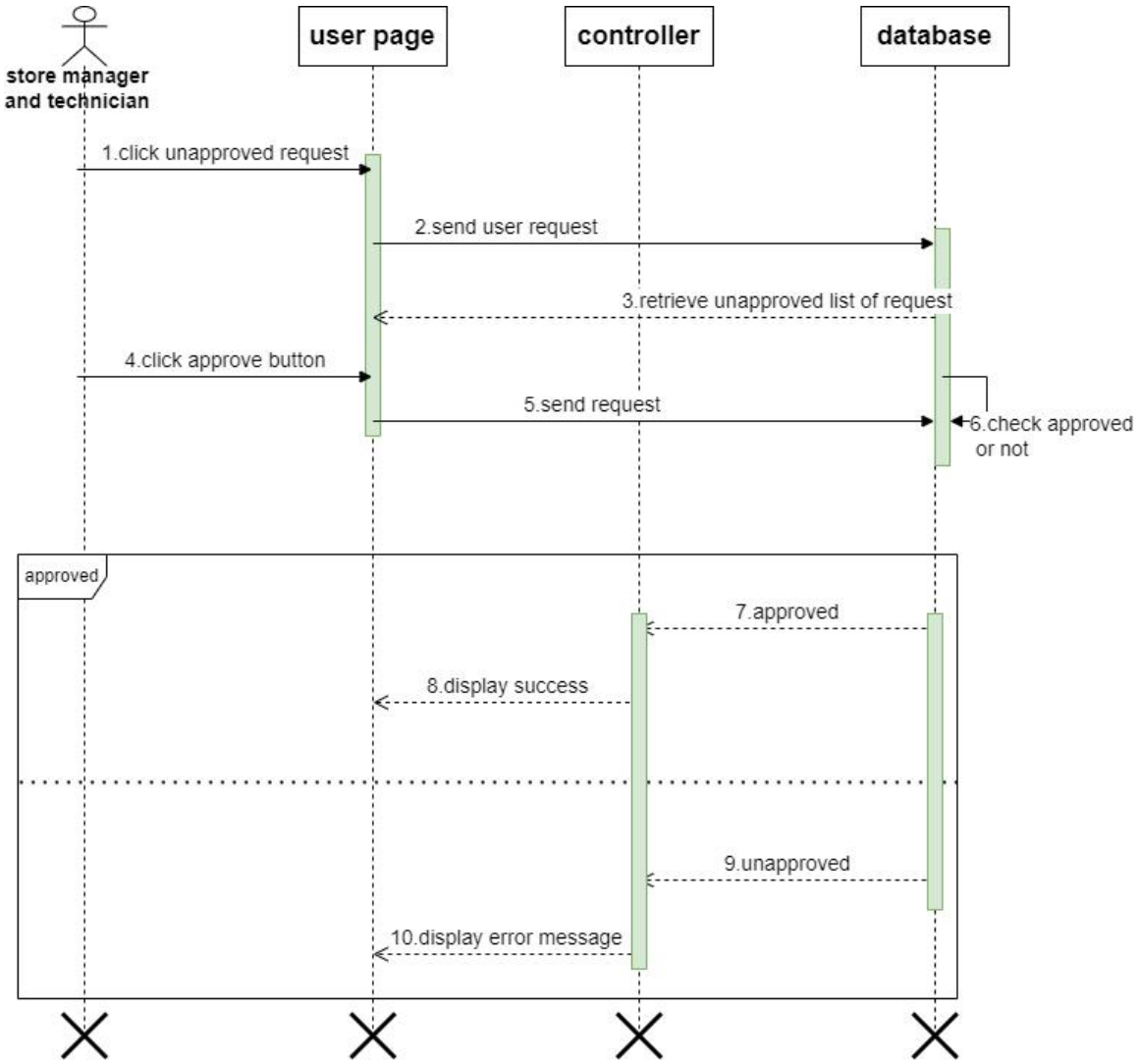


Figure 4. 3 Sequence Diagram for Approve

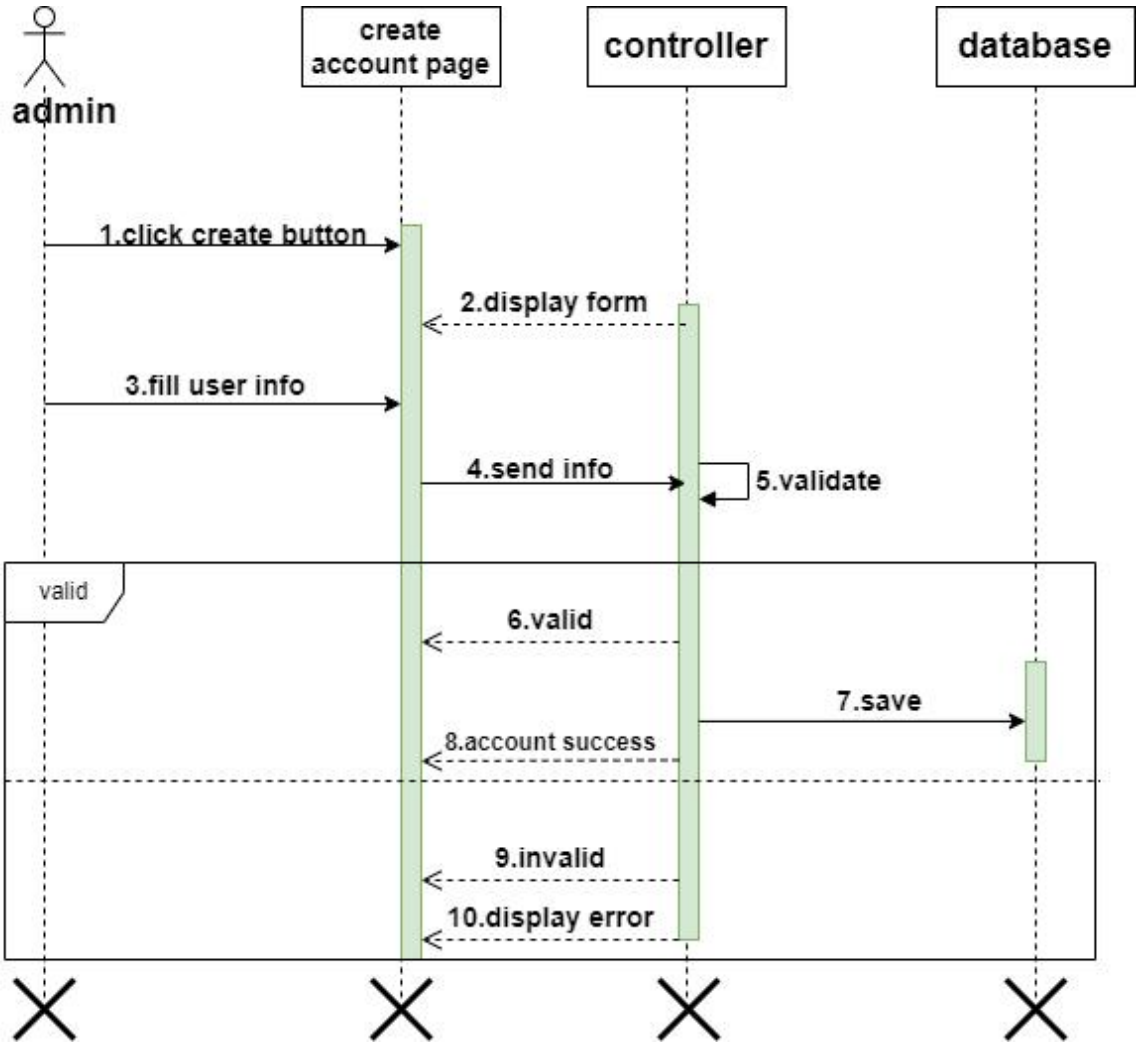


Figure 4. 4 Sequence Diagram For Create Account

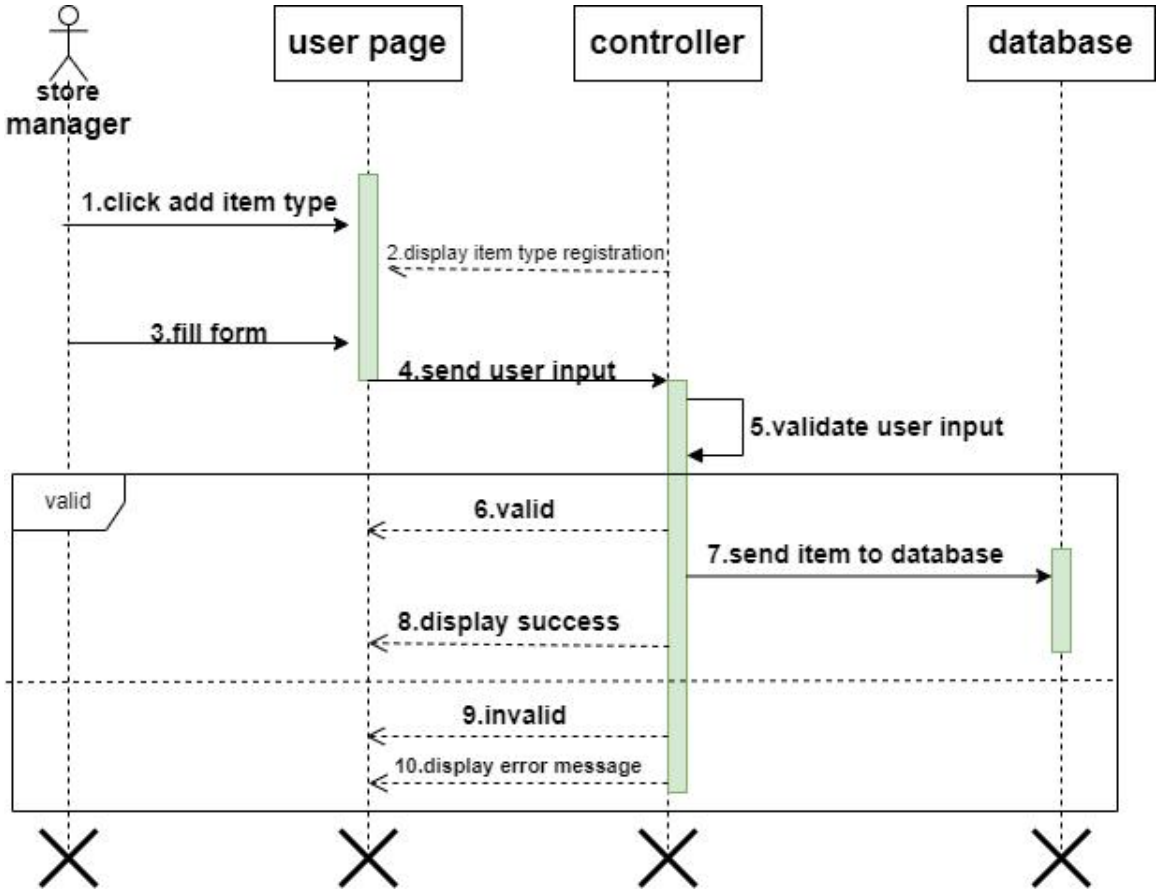


Figure 4. 5 Sequence Diagram for Add Items Types

4.3.2 Activity Diagram

An activity diagram is a type of UML (Unified Modeling Language) diagram that represents the dynamic aspects of a system. It is particularly useful for modeling the flow of activities or processes within a system, showing how different activities are coordinated and the sequence in which they occur.

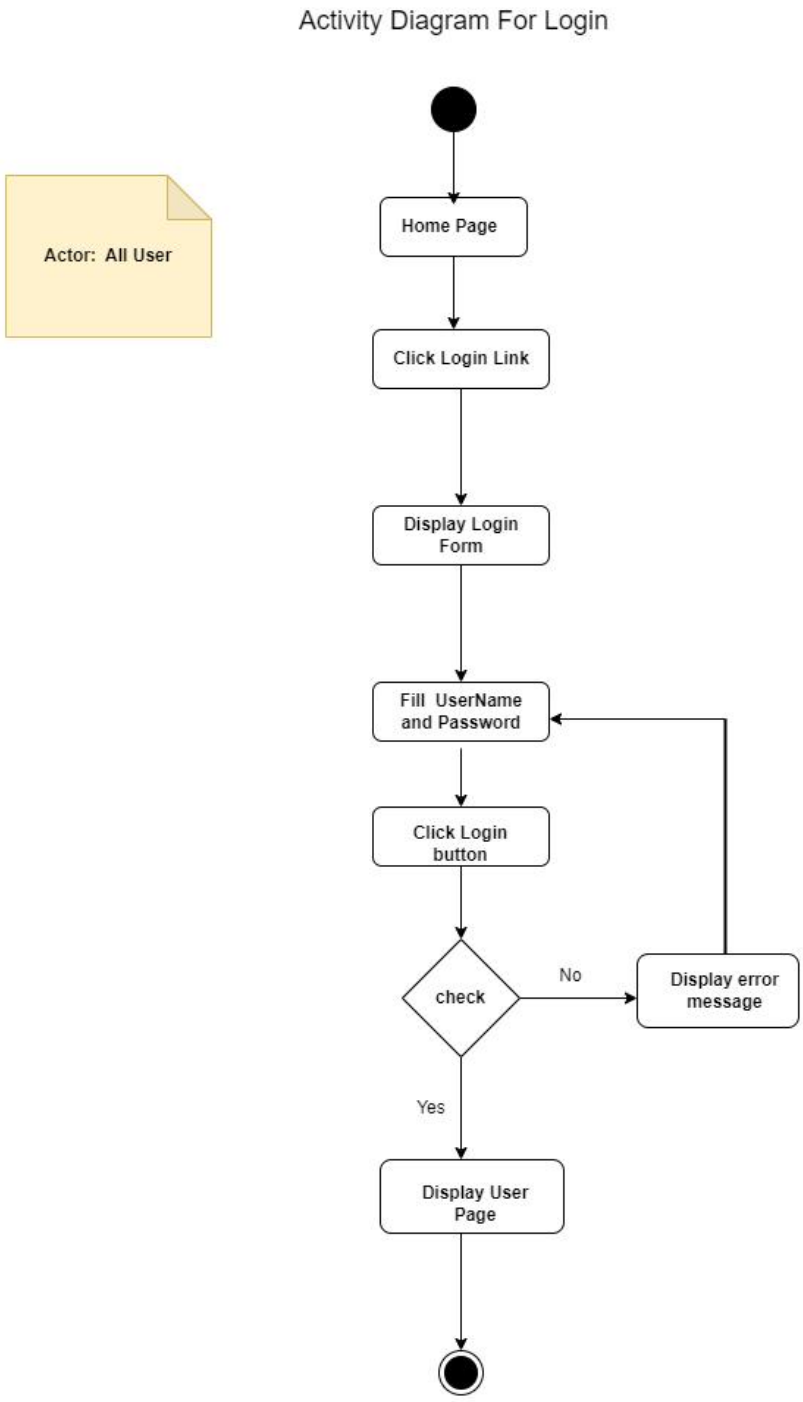


Figure 4. 6 Activity Diagram for Login

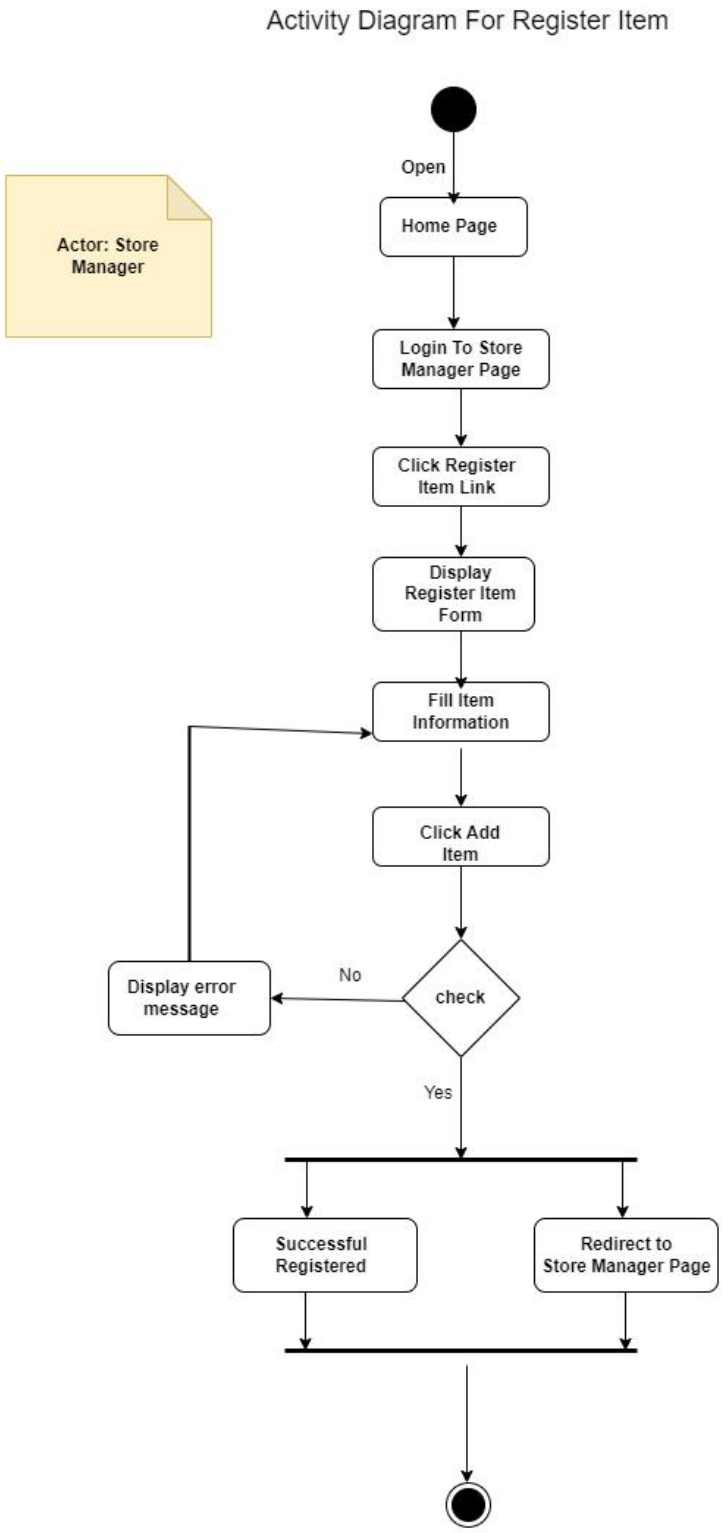


Figure 4. 7 Activity Diagram for Register Item

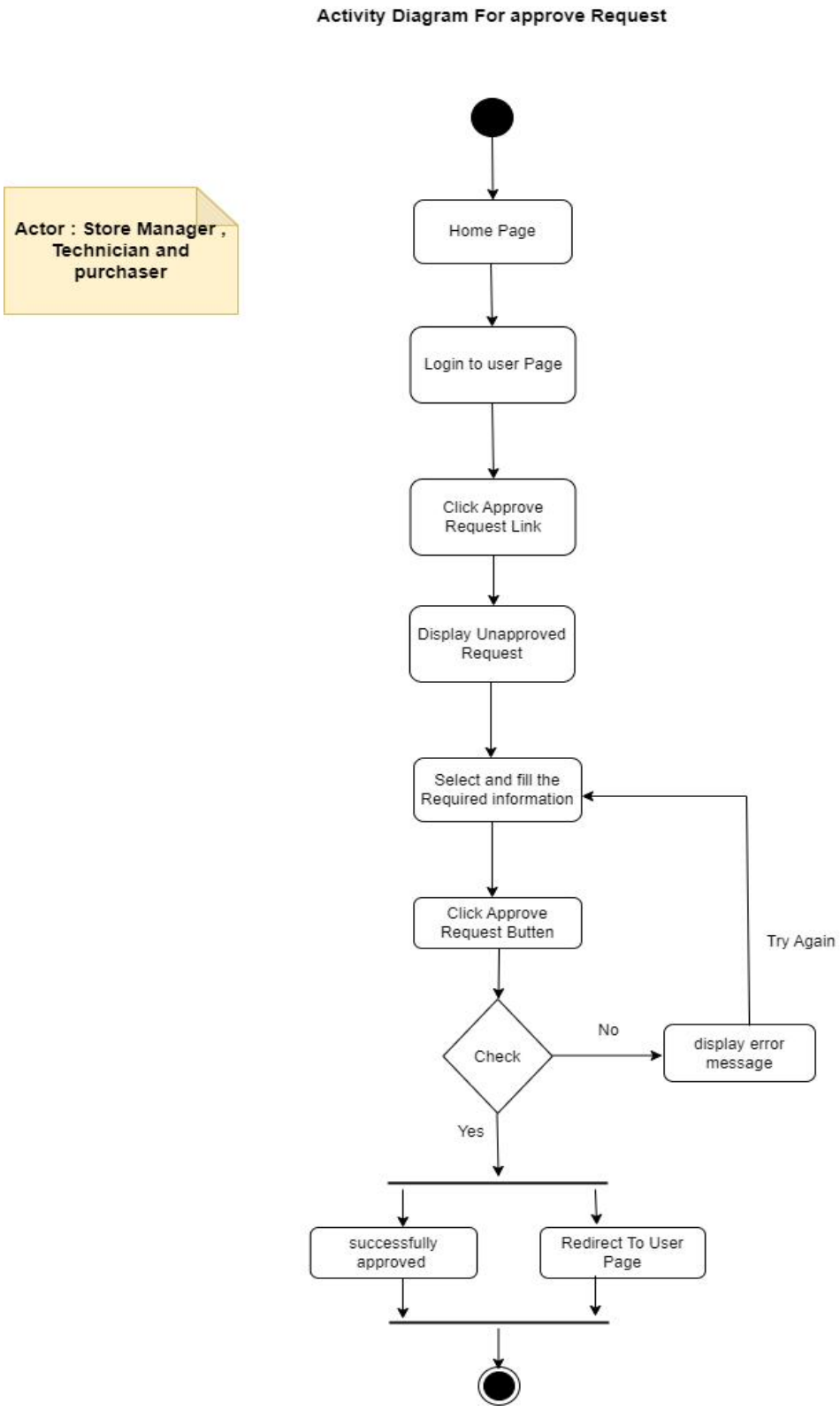


Figure 4. 8 Activity Diagram for Approve Request

4.3.3 State Chart Diagram

State chart diagram describes the flow of control of the proposed system from one state to another state to describe the system dynamically. States are defined as a condition in which an object exists and it changes when some event is triggered.

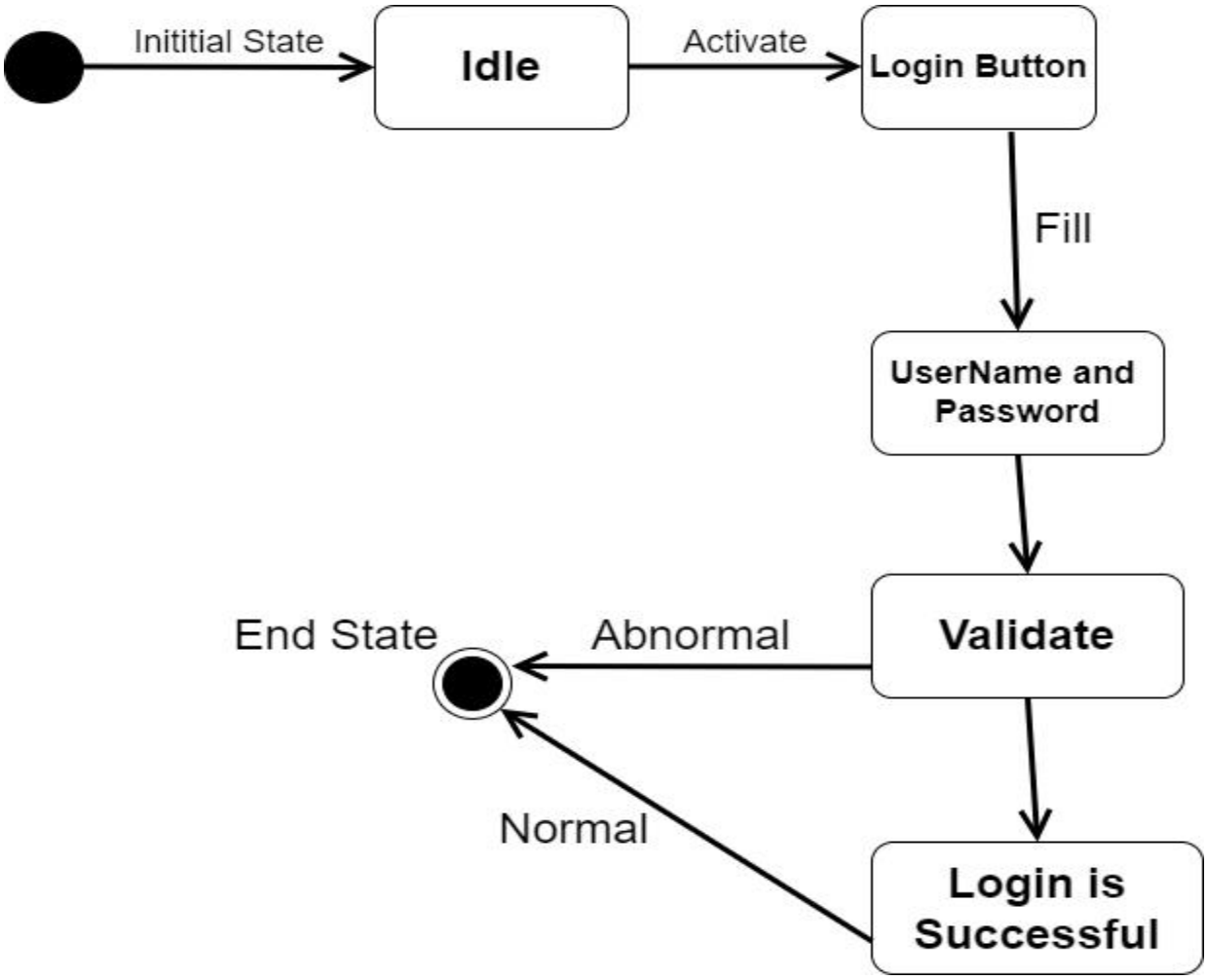


Figure 4. 9 State Chart Diagram for Login

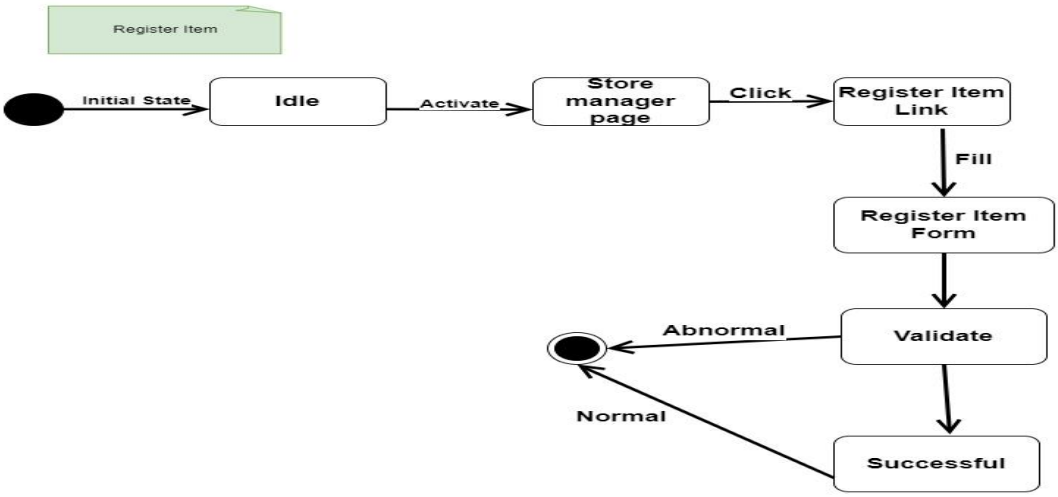


Figure 4. 10 State Chart Diagram for Register Item

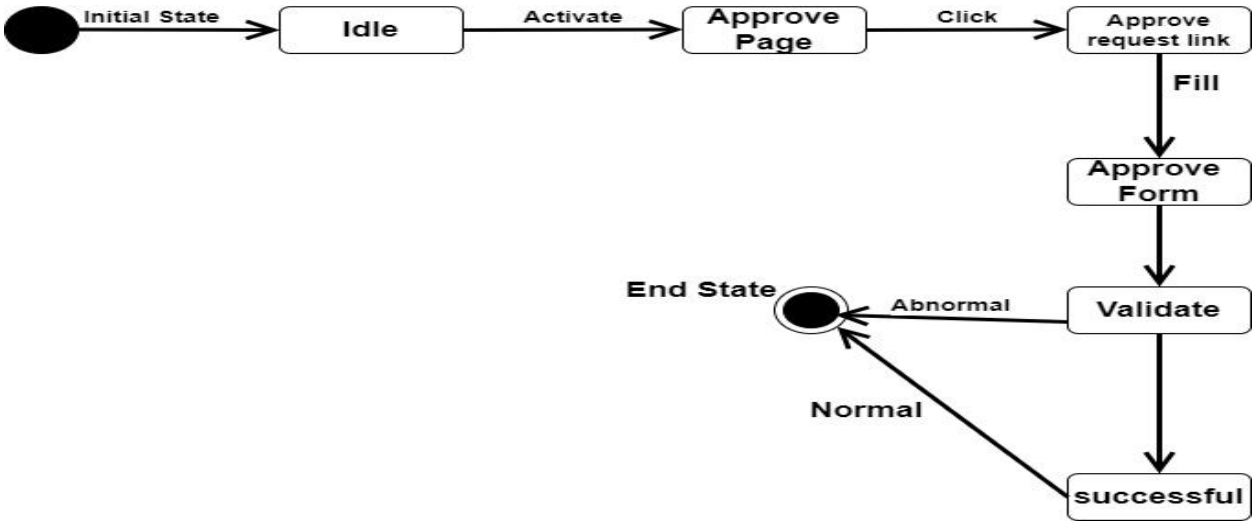


Figure 4. 11 State Chart Diagram for Approve Request

CHAPTER FIVE

5. SYSTEM DESIGN

5.1. Design Goals

Design goals are the high-level objectives that guide the development of the software architecture and components. These goals help ensure that the software meets specific criteria, such as performance, maintainability, and usability.

- **User Interface and Human Factors:** Design an intuitive and user-friendly interface to streamline user interactions, enhancing usability and reducing training overhead.
- **Hardware Consideration:** Optimize the software design to efficiently utilize available hardware resources, ensuring compatibility and performance on target devices.
- **Security Issues:** Implement robust security measures, including authentication, authorization, and data encryption, to safeguard sensitive information and protect against unauthorized access.
- **Performance Consideration:** Optimize system performance to ensure quick response times, efficient data retrieval, and scalability, meeting the demands of a growing inventory and user base.
- **Error Handling and Validation:** Incorporate comprehensive error handling mechanisms and data validation to prevent and gracefully handle errors, ensuring data integrity and a resilient system.
- **Quality Issues:** Strive for high software quality by adhering to coding standards, conducting thorough testing, and fostering a maintainable and modular design.
- **Backup and Recovery:** Implement reliable backup and recovery mechanisms to protect against data loss, ensuring business continuity in case of system failures or disasters.

- **Physical Environment:** Consider the physical environment where the system operates, accommodating factors like network conditions and device capabilities to ensure optimal performance.

- **Resource Issues**: Efficiently manage system resources, including memory and processing power, to prevent resource bottlenecks and ensure smooth operation under varying load conditions.

- **Documentation:** Develop comprehensive documentation, including user manuals and technical documentation, to facilitate system understanding, maintenance, and future enhancements.

5.2. Current System Architecture (if any)

The existing system of the inventory management system for gold water is manual system and hence there is no Existing software architecture that will be considered. As a result, we only describe the software architecture of the newly proposed system.

5.3 Proposed System Architecture

This section presents a general view of our system architecture and briefly describes the assignment of functionality to each subsystem. This system uses three-tier architecture. The user interface is implemented on a desktop PC and uses a standard graphical user interface with different modules running on the application server. The middle tiers are usually multitier. We use 3-tier for our project because 3- tier architecture provides scalability, performance, availability for the project.

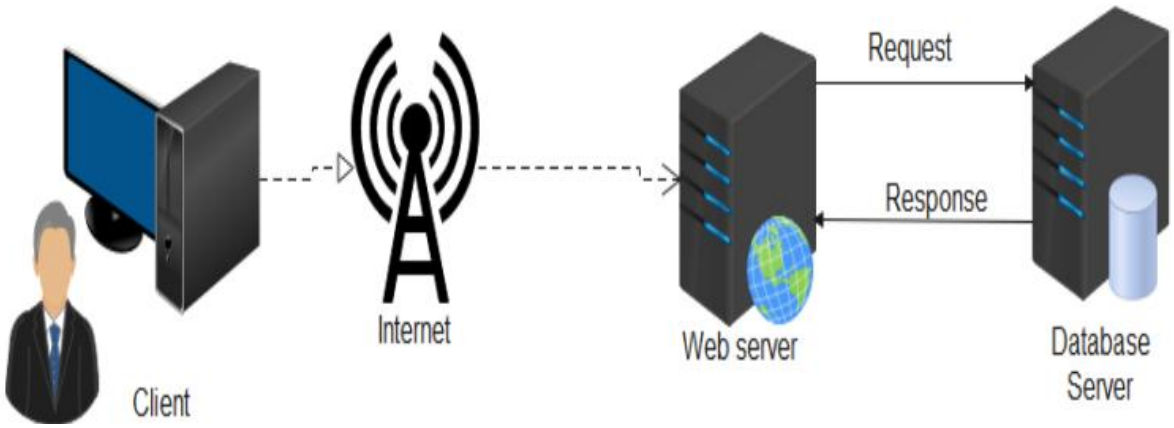


Figure 5. 1 Architecture of proposed system

5.3.1 Subsystem Decomposition and Description

To reduce the complexity of the solution domain, the project team decomposes a system into simpler parts, called subsystems, which are made of a number of solution domain classes. In the case of complex subsystems, the team recursively apply this principle and decompose a sub- system into simpler subsystems. Decomposition Results large systems into a set of loosely dependent parts which make up the system. Subsystem Decomposition is the way that helps us to distinguish the part of the operations that takes place under the organization.

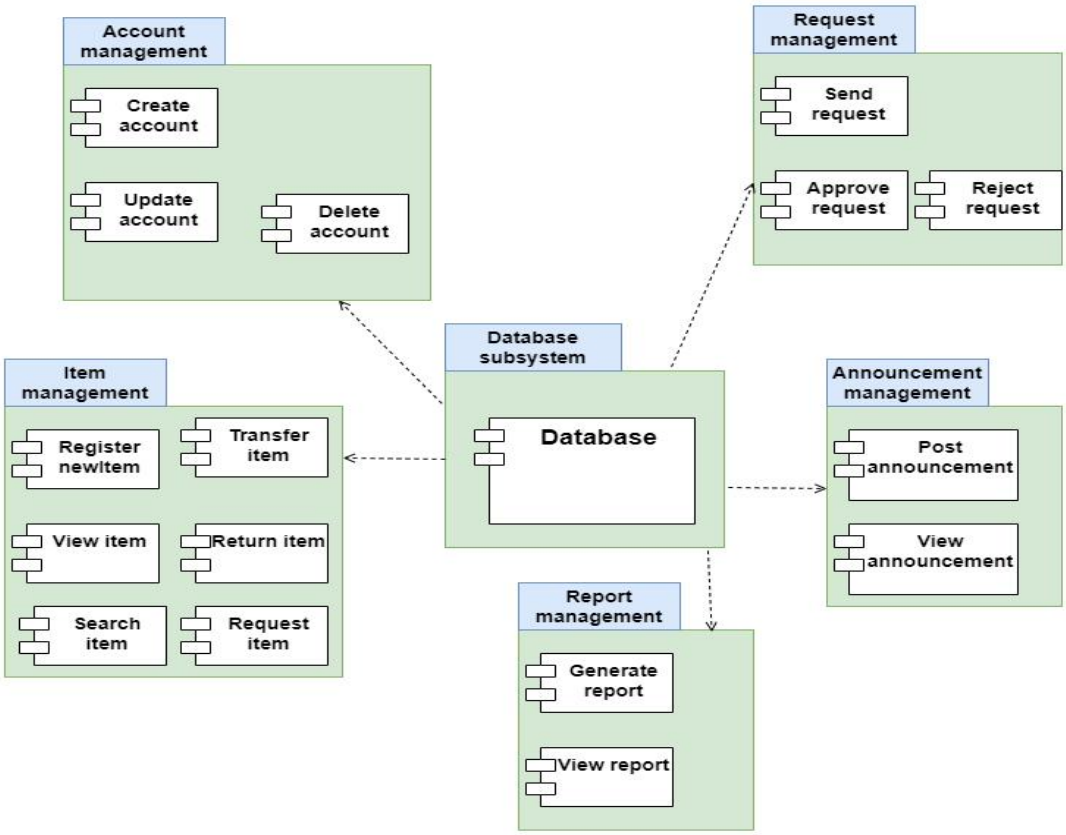


Figure 5. 2 System Decomposition

5.3.2 Hardware Software Mapping (Deployment diagram)

Deployment modeling is used to show the hardware of the system, the software that is installed in the hardware and also the middleware that is used to connect the disparate

machines to one and other. It also shows how the software and the hardware components work together.

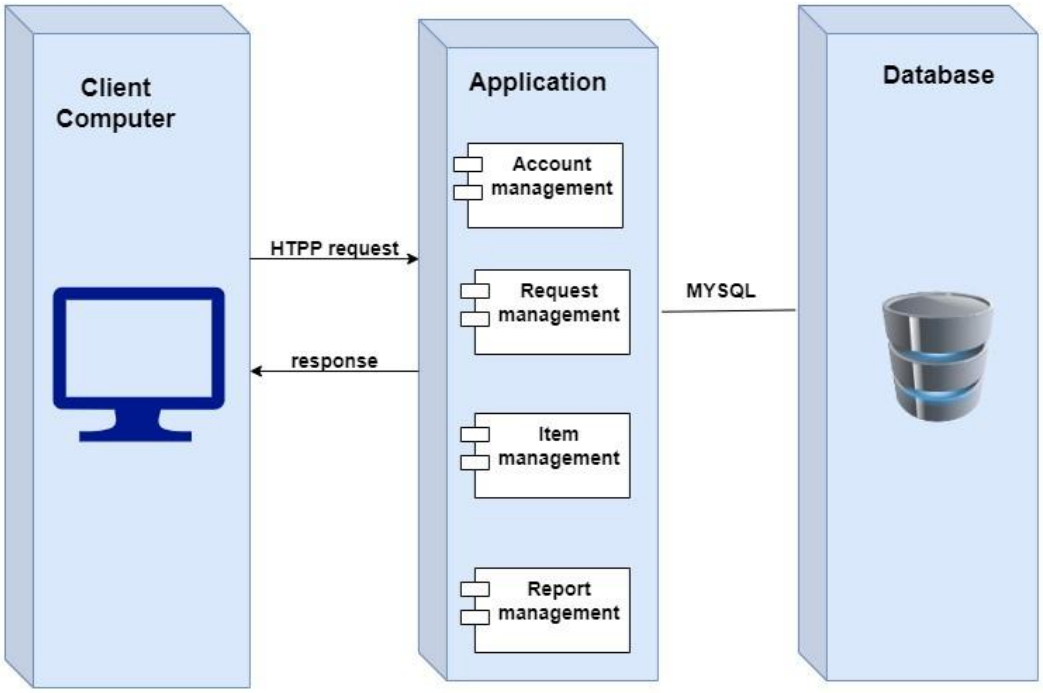


Figure 5. 3 Deployment diagram

5.3.3 Detailed Class Diagram

The class diagram is the main building block of object-oriented modeling. It is used for general conceptual modeling of the structure of the application, and for detailed modeling translating the models into programming code. Class diagrams can also be used for data modeling. The classes in a class diagram represent both the main elements, interactions in the application, and the classes to be programmed.

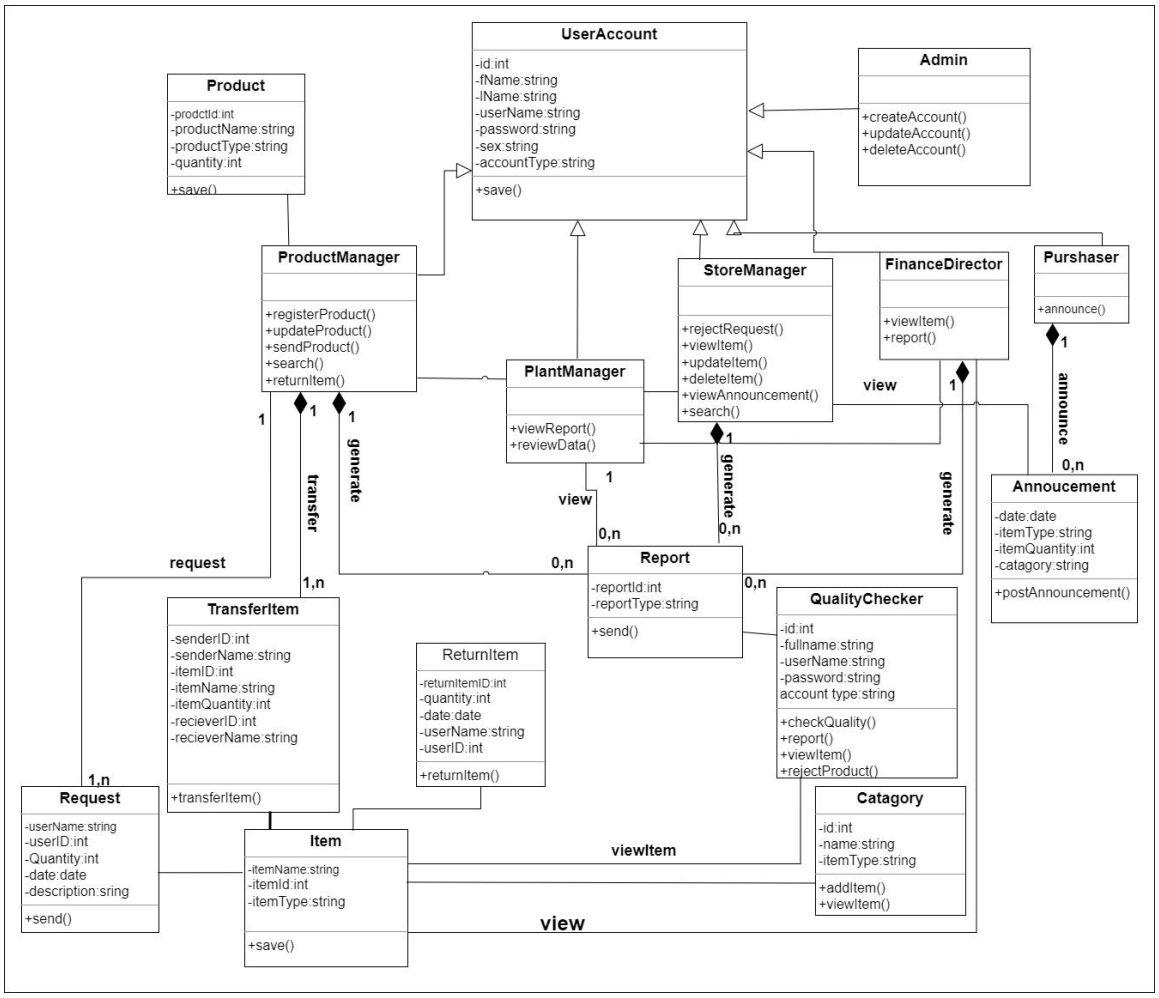


Figure 5. 4 Detailed Class Diagram

5.3.4 Persistent Data Management

When user interact with the system persistent data management is needed during this activity persistent object is identified, storage management strategy is selected, and description of database encapsulation is described, in order to store information persistently we map objects into tables and the attributes into fields to the specific table based on the objects found on the system. This part is to describe and show the necessary relationships among the tables, which are selected to store the data persistently in the system.

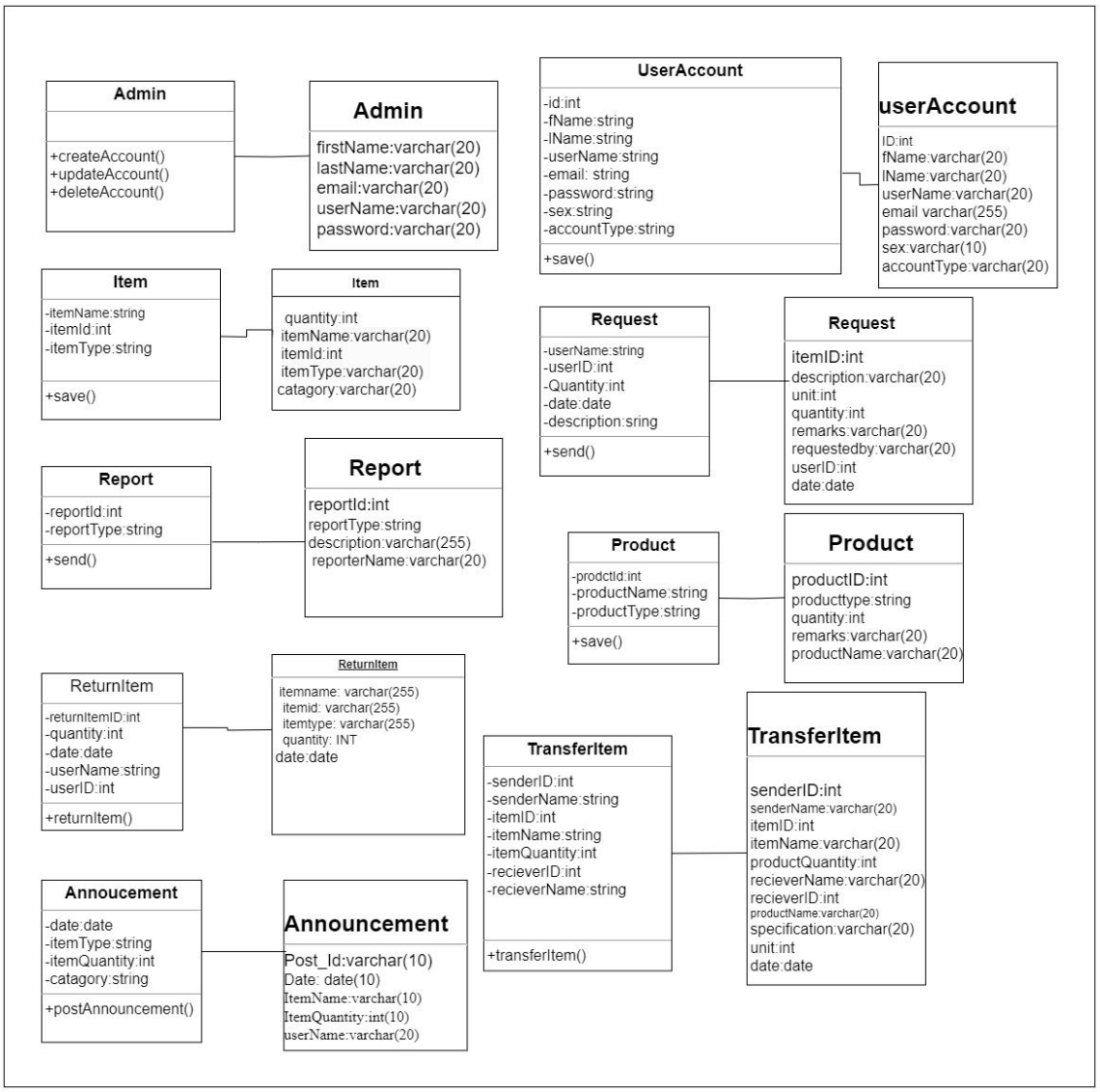


Figure 5. 5 Persistent Data Management

5.3.5 Access Control and Security

Access control and security describes the user model of the system in terms of an access privilege. We use Authentication (using password and user name), Authorization (access privilege) and auditing mechanism in order to perform access control in database. We describe privilege with actor by “yes” and “no”.

Table 5. 1 Access Control and Security

operation	admin	Storeman	Production manager	Technician	Quality check	purchaser	Finance director	Plant man	Store keeper
login	yes	yes	yes	yes	yes	yes	yes	yes	yes
Manage account	yes	no	no	no	no	no	no	no	no
Approve request	no	yes	no	yes	no	yes	no	no	no
Manage item	no	yes	no	no	no	no	no	no	no
Generate report	no	yes	yes	no	yes	no	yes	no	no
View report	no	no	no	no	no	no	no	yes	no
Post announce	no	no	no	no	no	yes	no	no	no
View announce	no	yes	no	no	no	no	no	no	no
Reject request	no	no	no	yes	no	yes	no	no	no
Search	no	yes	yes	no	no	no	no	no	no
Manage product	no	no	yes	no	no	no	no	no	no
Place item	no	no	no	no	no	no	no	no	yes

Check quality	no	no	no	no	yes	no	no	no	no
Return item	no	no	yes	no	no	no	no	no	no
transfer item	no	no	yes	no	no	no	no	no	no

5.3 Package

Package is groups of “basic elements”, e.g., classes or use cases represented as file folders. There is a relationship between packages. Example dependency: - Package A depends on package B if package A contains a class which depends on a class in package B.in our system have some

Account Management Package:

In this package, managing of information regard to account and perform:

Create account

Delete account

Update account

Report management Package: This Package allows for managing report information and perform this operation:

Generate report

View report

Request management Package: This subsystem allows for managing Request information and perform this operation:

Send request

Approve request

Item management Package: This also allows the following function.

Register new item

Register transferred item

Register returned item

Search item

Announcement Management Package

View Announcement

Post Announcement

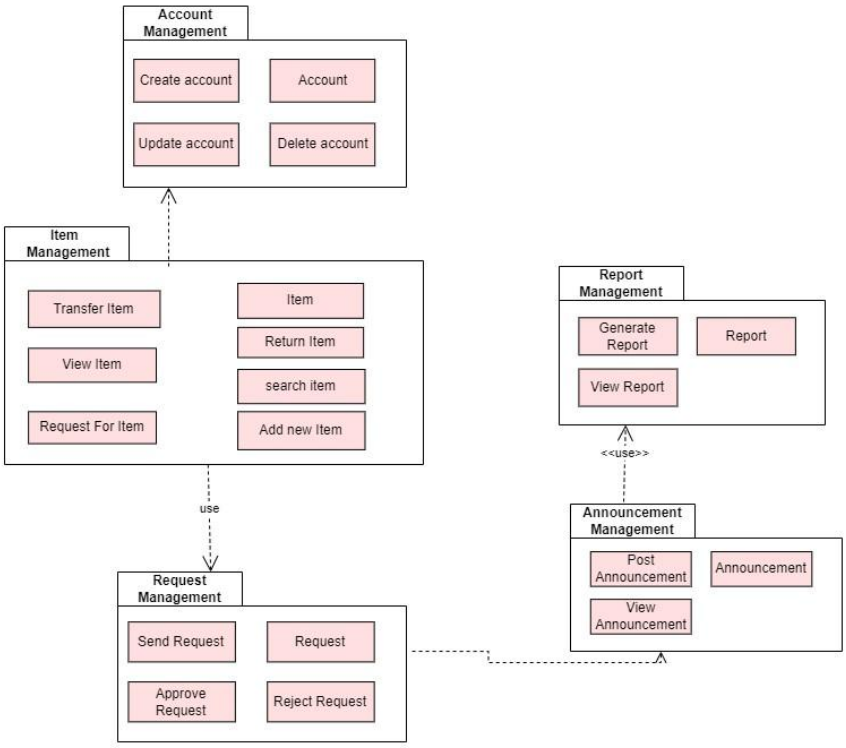


Figure 5. 6 Package Diagram

5.4 Algorithm Design

Algorithms used to solve the problem. It defines the algorithm required for each element of the architectural design to accomplish its tasks. It defines the algorithm required for each element of the architectural design to accomplish its tasks.

Algorithm for login

```
Algorithm LoginProcess
 Begin
// Step 1: Get Username
Input: Username
 // Step 2: Get Password
Input: Password
 // Step 3: Authenticate User
 If (AuthenticateUser(Username, Password))
 Then
// Step 4: Set Cookies for Successful Login
SetCookies(LoginSuccessful)
// Step 5: Display Login Successful
 Display "Login Successful"
Else
// Step 6: Display Login Failed Message
Display "Login failed, please check your username and password."
End
AuthenticateUser(username, password)
    ValidateCredentials(username, password)
    IF (Credentials are valid) THEN
        RETURN True
```

ELSE

RETURN False

END IF

Algorithm for logout

Begin:

IF (UserIsLoggedIn()) THEN

DeleteCookies

DisplayHomePage

Display "Logout Successful"

End

Algorithm for Item Registration

RegisterItem()

BEGIN

INPUT: Item Information (name, category, price, quantity)

// Step 1: Validate Input

IF (All mandatory fields are filled and data is in the correct format)

Proceed to Step 2

ELSE

PRINT "Invalid input. Please provide valid information."

RETURN

// Step 2: Generate Unique Identifier

uniqueIdentifier = generateUniqueIdentifier() // You need to implement this function.

```
// Step 3: Create Item Object
    newItem = {
        'id': uniqueIdentifier,
        'name': name,
        'category': category,
        'price': price,
        'quantity': quantity
    }

// Step 4: Store Item Information
    saveItemToDatabase(newItem)  // You need to implement this function.

// Step 5: Confirmation
    PRINT "Item has been successfully registered."
END
```

5.5 User Interface Design

The following interface design pictures describe the logical characteristics of some interfaces between the system and the users. So, the sample interfaces are shown as follows:

.

Figure 5. 7 User interface design for home page

CHAPTER SIX

6. IMPLEMETATION AND TESTING

6.1. Implementation of the Database

//Table structure for table ‘user account’

```
$createTableSql = "CREATE TABLE IF NOT EXISTS createdaccount (
    id INT(11) AUTO_INCREMENT PRIMARY KEY,
    full_name VARCHAR(255) NOT NULL,
    email VARCHAR(255) NOT NULL,
    user_type VARCHAR(255) NOT NULL,
    password VARCHAR(255) NOT NULL,
    created_at  TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    profile_picture VARCHAR(255)
)";
```

//Table structure for table ‘requestitem’

```
$sql = "CREATE TABLE requestitem (
  id INT AUTO_INCREMENT PRIMARY KEY,
  username VARCHAR(255) NOT NULL,
  itemname VARCHAR(255) NOT NULL,
  itemtype VARCHAR(255) NOT NULL,
  quantity INT NOT NULL,
  description TEXT NOT NULL,
  created_at  TIMESTAMP DEFAULT CURRENT_TIMESTAMP
```

```
)";
//item table table configuration
$tableSql = "CREATE TABLE IF NOT EXISTS itemtable(
   itemname VARCHAR(255) NOT NULL ,

   itemid INT  PRIMARY KEY ,
   itemtype VARCHAR(255) NOT NULL ,
   itemquantity INT NOT NULL,
   date DATE NOT NULL
   )";
if ($conn->query($tableSql) === TRUE) {
   echo "Table 'itemtable' created successfully";
} else {
   echo "Error creating table: " . $conn->error;
}
```

6.2 Implementation of Class Diagram

//user detail implementation

```
<?php class Database
{
  private $conn;
    public function __construct($dbConnection)
  {
    $this->conn = $dbConnection;
  }
```

```
    public function getAllUsers()
    {
        $sql = "SELECT * FROM createdaccount";
        $result = mysqli_query($this->conn, $sql);
        $users = [];
        while ($row = mysqli_fetch_assoc($result)) {
            $users[] = new User($row);
        }
        return $users;
    }
}
```

?>

//implementation of user list

```
<?php   // class to represent user data
class User
{
    public $fname;
    public $username;
    public $email;
    public $password;
    public $status;
    public $profile;
    public function __construct($userData)
    {
        $this->profile = $userData['profile_picture'];
```

```
        $this->fname = $userData['full_name'];
        $this->username = $userData['email'];
        $this->email = $userData['email'];
        $this->password = $userData['password'];
        $this->status = $userData['status'];
    }
    public function renderTableRow()
?>
```

6.3. Configuration of Application Server

XAMP server: is a server used to run the codes as a virtual machine.

PHPMyAdmin: front-end, to develop the PHP programming of the system translation.

MYSQL: to hold data base and to store data.

These are used for testing all the functional requirements are function able and access able in the environment of XAMP server.

Web browser: for internet connection.

The configuration of application server in folders like

Apache configuration file: \xampp\apache\conf\httpd.conf, \xampp\apache\conf\extra\httpd

xampp.conf

PHP configuration file: \xampp\php\php.ini

MySQL configuration file: \xampp\mysql\bin\my.ini

6.4. Configuration of Application Security

We use md5 encryption for password, look the below code part functions.

```php
<?php
require 'vendor/autoload.php';
use PHPMailer\PHPMailer\PHPMailer;
use PHPMailer\PHPMailer\Exception;
require 'vendor/PHPMailer/PHPMailer/src/Exception.php';
require 'vendor/PHPMailer/PHPMailer/src/PHPMailer.php';
require 'vendor/PHPMailer/PHPMailer/src/SMTP.php';
include('C:\xampp\htdocs\IMSP\DBconnection.php');
$createTableSql = "CREATE TABLE IF NOT EXISTS createdaccount (
    id INT(11) AUTO_INCREMENT PRIMARY KEY,
    full_name VARCHAR(255) NOT NULL,
    email VARCHAR(255) NOT NULL,
    password VARCHAR(255) NOT NULL,
    profile_picture VARCHAR(255)
)";
// Execute the table creation query
if ($conn->query($createTableSql) === TRUE) {
    echo "Table 'createdaccount' created successfully.";
} else {
    echo "Error creating table: " . $conn->error;
}
```

```
// Process the form submission
if ($_SERVER["REQUEST_METHOD"] == "POST") {
   $username = mysqli_real_escape_string($conn, $_POST["username"]);
$email = mysqli_real_escape_string($conn, $_POST["email"]);
$password = $_POST["password"];
$usertype = $_POST["usertype"];
$profile_picture = $_POST["profile_picture"]; // Assuming the name attribute of the input field is "profile-picture"

   // Check if the email already exists
   $checkEmailSql = "SELECT * FROM createdaccount WHERE email='$email'";
   $result = $conn->query($checkEmailSql);
   if ($result->num_rows > 0) {
      // Email already exists, display an error message
      echo '<script>alert("Error: Email already exists. Please use a different email.");</script>';
   } else {
      // Insert the form data into the "createdaccount" table
      $password_md5 = md5($password); // Hash the password
      $insertSql = "INSERT INTO createdaccount (full_name, email, password, profile_picture,user_type) VALUES ('$username', '$email', '$password_md5', '$profile_picture' , '$usertype')";
```

```
if ($conn->query($insertSql) === TRUE) {
    // Send email to the user with their username and password
    $mail = new PHPMailer(true);
    try {
        //Server settings
        $mail->isSMTP();
        $mail->Host = 'smtp.gmail.com';  // Specify main and backup SMTP servers
        $mail->SMTPAuth = true;                               // Enable SMTP authentication
        $mail->Username = 'daniel.bekele.garedew@gmail.com';           // SMTP
        username
        $mail->Password = 'xfnq vqnu skwg gfcw';                    // SMTP password
        $mail->SMTPSecure = 'tls';                            // Enable TLS encryption, ssl also
        accepted
        $mail->Port = 587;                                    // TCP port to connect to
      //Recipients
        $mail->setFrom('gold@gmail.com', 'Admin');
        $mail->addAddress($email, $username);     // Add a recipient
        $mail->addReplyTo('gold@gmail.com', 'Admin');
        //Content
        $mail->isHTML(true);                                  // Set email format to HTML
```

```
            $mail->Subject = 'Your Account Details';
            $mail->Body = "Hello,<br><br>Your account has been created.<br><br>Username: $email<br>Password: $password<br><br>Please keep this information secure.";
            $mail->send();
            echo '<script>alert("User account created successfully. Account details sent to user\'s email address.");</script>';
            echo '<script>window.location.href = "login.php";</script>'; // Redirect to login page
        } catch (Exception $e) {
            echo '<script>alert("Error sending email: ' . $mail->ErrorInfo . '");</script>';
        }

    } else {
        echo '<script>alert("Error creating account: ' . $conn->error . '");</script>';
    }
  }
}
$conn->close();
?>
<?php
include('C:\xampp\htdocs\IMSP\DBconnection.php');
$email = $_GET['updateid'];
```

```
$db = new Database($conn);
$user = $db->getUserByEmail($email);
$fname = $user['full_name'];
$username = $user['email'];
$password = $user['password'];
$status = $user['status'];
$profile = $user['profile_picture'];
$imgpath = "img/" . "" . $profile;
  // $imgpath =  "img/me.jpg";
$base64Image = base64_encode($profile);
if (isset($_POST['submitt'])) {
   $fname = $_POST['fname'];
   $username = $_POST['email'];
   $password = $_POST['password'];
   $status = $_POST['status'];
   $profile_picture = $_POST['profile_picture'];
   if ($db->updateUser($profile_picture, $fname, $email, $username, $password, $status)) {
      header('location:userlist.php');
   } else {
      die(mysqli_error($conn));  }
}
?>
```

6.5. Implementation of User Interface

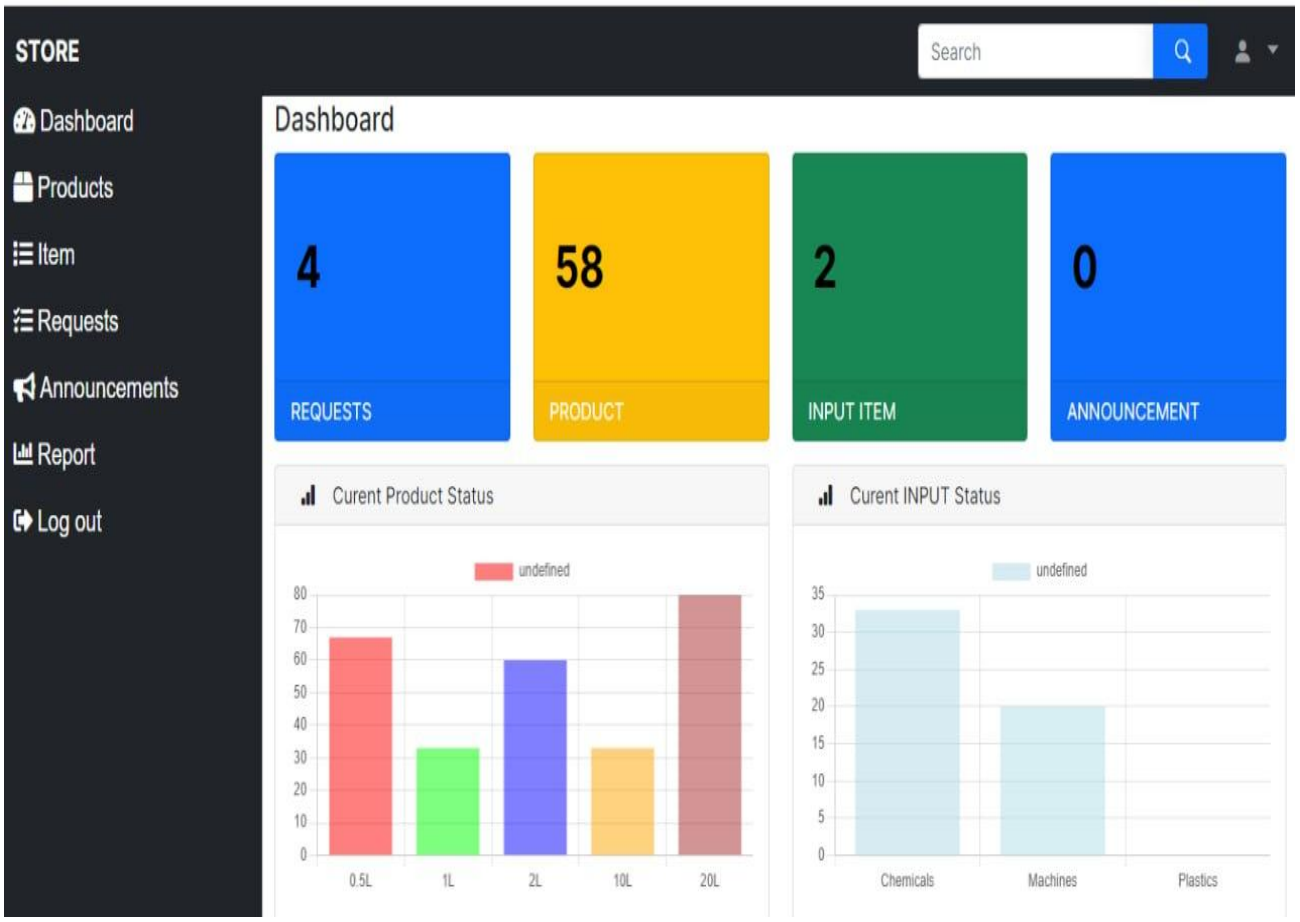


Figure 6. 1 User interface design for store manager page

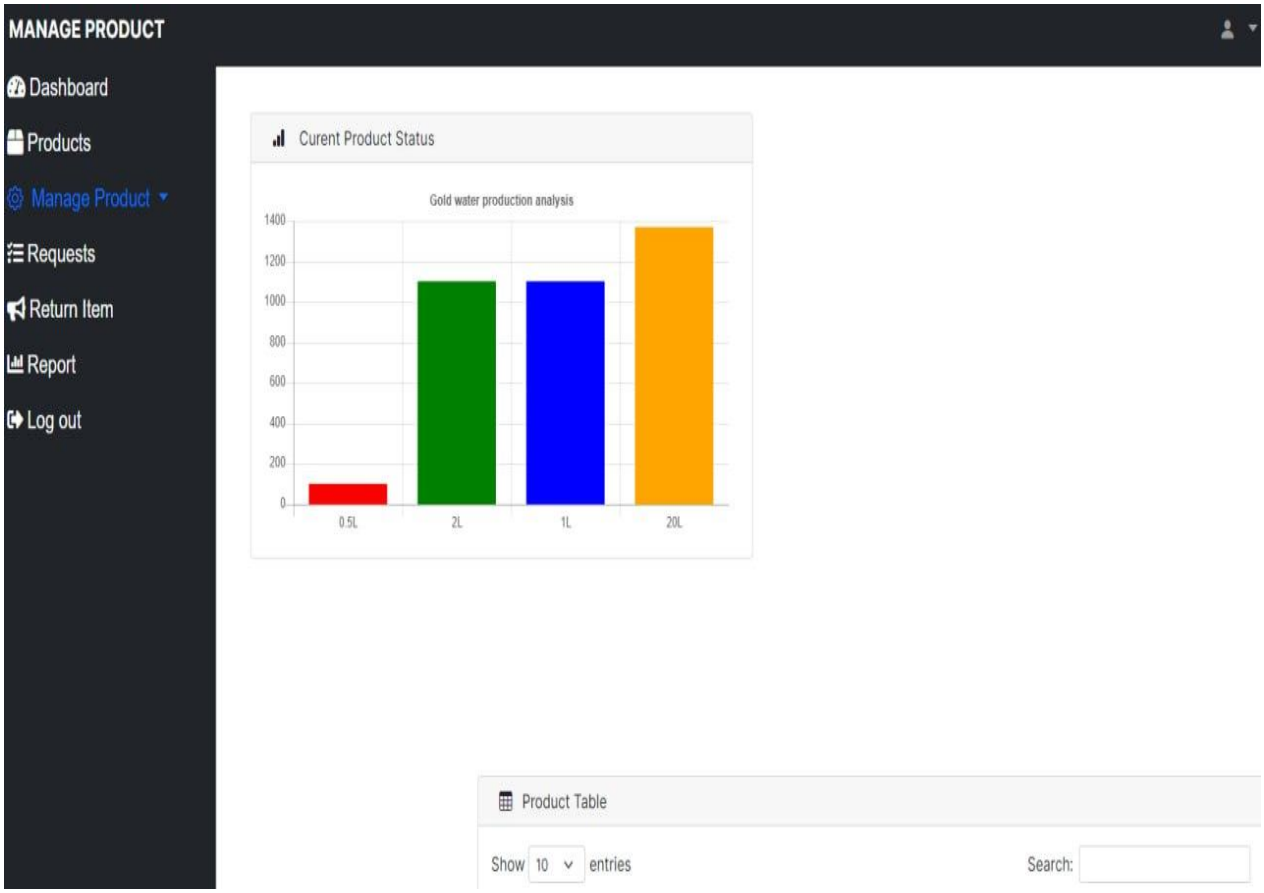


Figure 6.2 User interface design for Product Manager Page

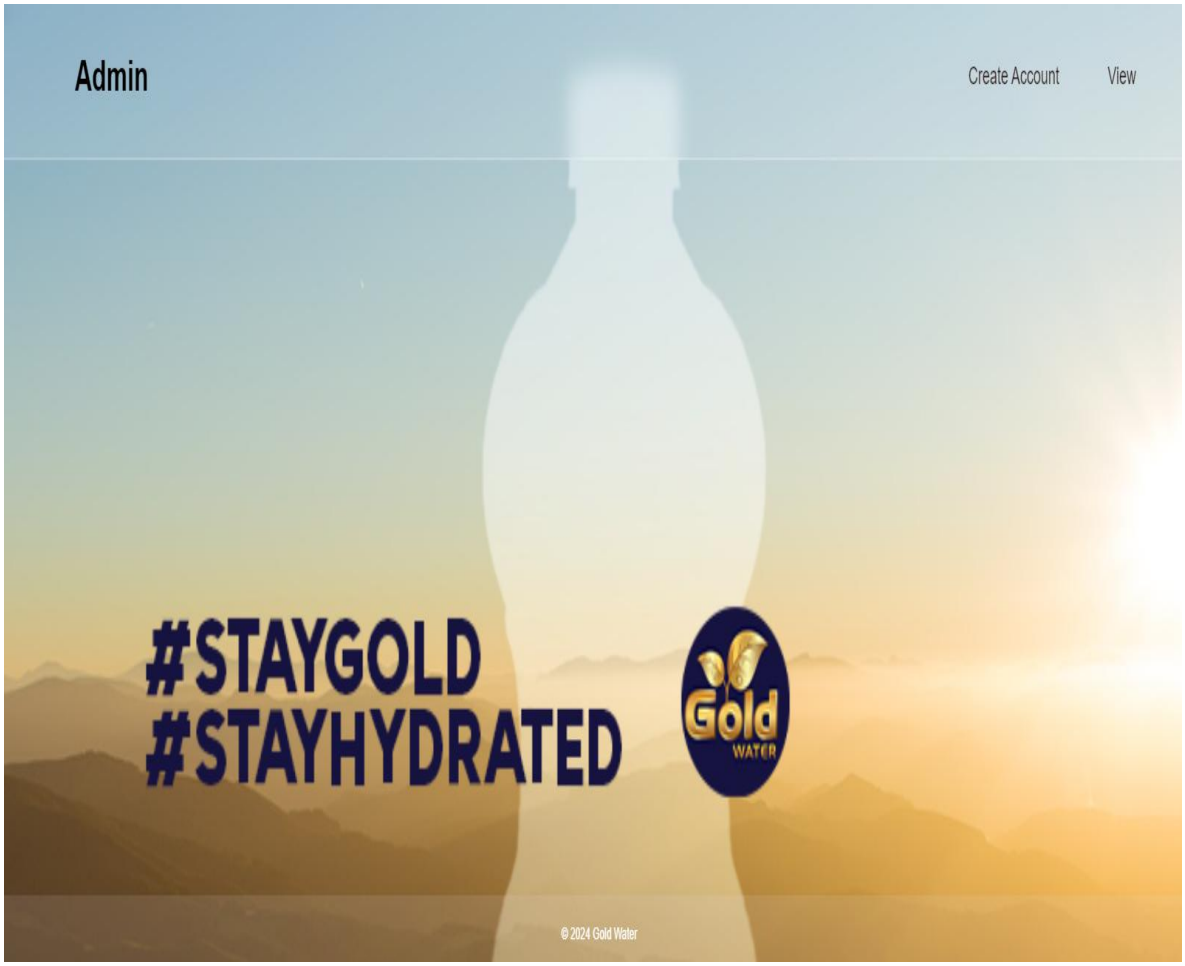


Figure 6.3 User interface design for Admin Page

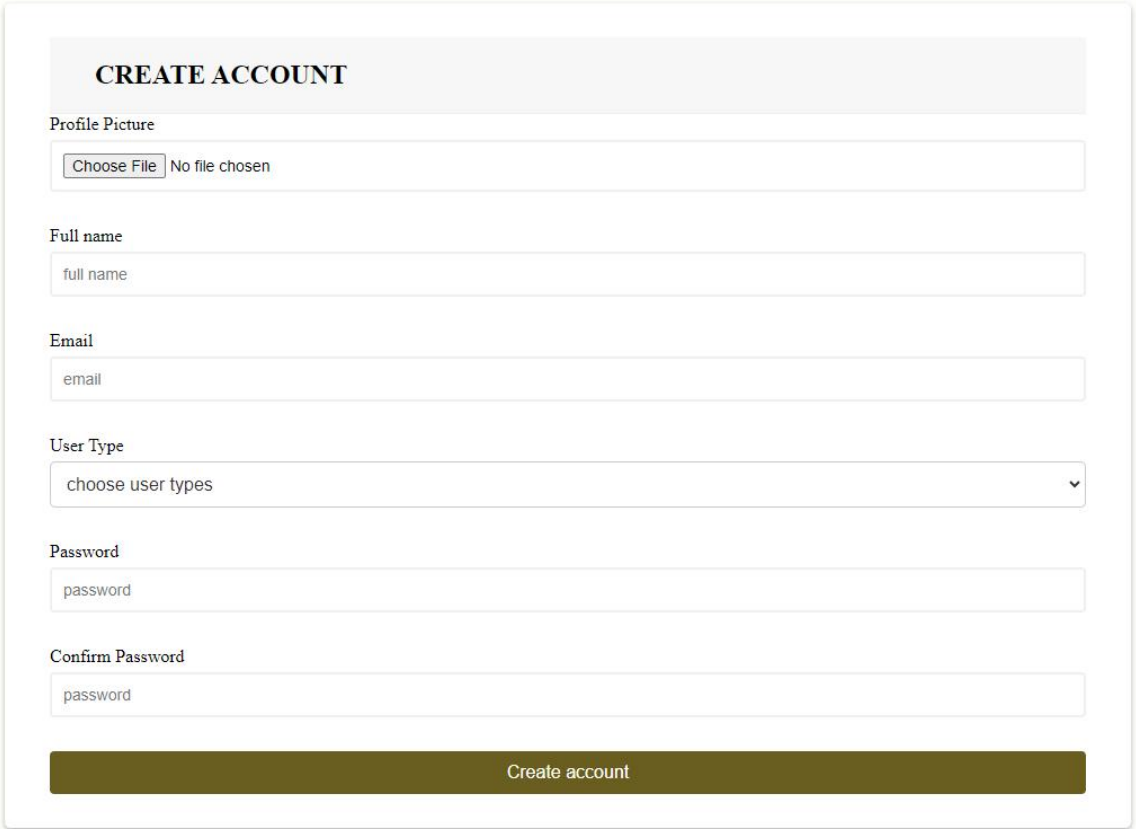


Figure 6.4 User interface design for Account Creation Page

6.6. Testing

6.6.1. Unit Testing

Every module of the system is separately tested. The team tests every module by applying some selection mechanism. Though this mechanism every modules gets tested. If an error occurs correction will be taken without affecting another module. Unit testing gives stress on modules independently of one another, to find errors. This helps the tester in detecting errors in coding and logic that are contained with that module alone. The errors resulting from the interaction between modules are initially avoided. Unit testing gives stress on modules independently of one another, to find errors. This helps the tester in detecting errors in coding and logic that are contained within that module alone. The errors resulting from the interaction between modules are initially avoided.

Example for item request

```
const userNameError=document.getElementById('req1');
```

```
const itemnameError=document.getElementById('req2');
const itemtypeError=document.getElementById('req3');
const quantityError=document.getElementById('req4');
const descriptionError=document.getElementById('req5');
const submitError=document.getElementById('req6');
function validateUserName(){
    var userName=document.getElementById('username').value;
    var reg = /^[a-zA-Z]+$/;
    if(userName.length==0){
        userNameError.innerHTML='user name is required';
        return false;
    }
    if(!userName.match(reg)){
        userNameError.innerHTML='user name only contians letter';
        return false;
    }
    return true;
}
function validateitemname(){
    var itemname=document.getElementById('itemname').value;
    var reg = /^[a-zA-Z]+$/;
```

```
    if(itemname.length==0){
        itemnameError.innerHTML='item name is required';
        return false;
    }
    if(!itemname.match(reg)){
        itemnameError.innerHTML='item name only contians letter';
        return false;
        }
 return true;
}
function validateitemtype(){
    var itemtype=document.getElementById('itemtype').value;
    if (itemtype === "Enter types of item") {
    if(!itemtype.match(reg)){
        itemtypeError.innerHTML='choose item type';
        itemtype.focus();
    return false;
    }
    return true;
}
}
```

```
function validateQuantity(){
    var quantity=document.getElementById('quantity').value;
    if(quantity==""  quantity==null){
        quantityError.innerHTML='please insert quantity';
    }
        if (isNaN(quantity) && !quantity==0) {
            quantityError.innerHTML='only digit possible';
            return false;
        }
        quantityError.innerHTML='';
        return true;
      }
function validateDescription(){
    var description=document.getElementById('description').value;
    if(description.length<10){
        descriptionError.innerHTML='description length should be greater than ten char';
        return false;
    }
    var lettersOnlyRegex = /^[a-zA-Z\s]*$/;
    if (!lettersOnlyRegex.test(description)) {
        descriptionError.innerHTML ='Description must contain only letters';
```

```
        return false;
    }
    descriptionError.innerHTML='';
    return true;
}
        function
validateForm(){          if( !validateUserName()  !validateitemname()  !validateitemtype() !
validateQuantity() ||validateDescription()){
              submitError.style.display='block';
              submitError.innerHTML='please fill the form correctly';
             setTimeout(function(){
              submitError.style.display='none';
           },3000)
              return false;
           }
           return true;
        }
```

Example for product transfer

```
const senderNameError=document.getElementById('tra1');
const itemIdError=document.getElementById('tra2');
const itemNameError=document.getElementById('tra3');
const itemQuantityError=document.getElementById('tra4');
```

```
const receiverIdError=document.getElementById('tra5');
const receiverNameError=document.getElementById('tra6');
const remarkError=document.getElementById('tra7');
const submitError=document.getElementById('tra8');
function validateSenderName(){
    var senderName=document.getElementById('sendername').value;
    var reg = /^[a-zA-Z]+$/;
    if(senderName.length==0){
        senderNameError.innerHTML='sender name is required';
        return false;
    }
    if(!senderName.match(reg)){
        senderNameError.innerHTML='sender name only contians letter';
        return false;
    }
    senderNameError.innerHTML='<i class="fa-solid fa-check"></i>';
    return true;
}
function validateItemId(){
    var itemid=document.getElementById('productid').value;
    if(itemid.length==0){
```

```
        itemIdError.innerHTML='please enter item id';
        return false;
    }
    if(!itemid.length>=1){
itemIdError.innerHTML='item id length should be greater than zero';
return false;
    }
    if(!itemid.match(/^[0-9]+$/)){
itemIdError.innerHTML='only digit possible';
return false;
    }
    itemIdError.innerHTML='';
    return true;
}
function validateItemName(){
    var itemName=document.getElementById('productname').value;
    var reg = /^[a-zA-Z]+$/;
    if(itemName.length==0){
        itemNameError.innerHTML='product name is required';
        return false;
    }
```

```
    if(!itemName.match(reg)){
        itemNameError.innerHTML='item name only contians letter';
        return false;
    }
    itemNameError.innerHTML='<i class="fa-solid fa-check"></i>';
    return true;
}
function validateQuantity(){
    var quantity=document.getElementById('productquantity').value;
    if(quantity==""  quantity==null){
        itemQuantityError.innerHTML='please insert quantity';
    }
 if (isNaN(quantity) && !quantity==0) {
                itemQuantityError.innerHTML='only digit possible';
                return false }
            itemQuantityError.innerHTML='';
            return true;
        }
        function validateReceiverId(){
            var receiverid=document.getElementById('receiverid').value;
            if(receiverid.length==0){
```

```
        receiverIdError.innerHTML='please enter receiver id';
        return false;
    }
    if(!receiverid.length>=1){
        receiverIdError.innerHTML='receiver id length should be greater than zero';
return false;
    }
    if(!receiverid.match(/^[0-9]+$/)){
        receiverIdError.innerHTML='only digit possible';
return false;
    }
    receiverIdError.innerHTML='';
    return true;
}
function validateReceiverName(){
    var receiverName=document.getElementById('receivername').value;
    var reg = /^[a-zA-Z]+$/;
    if(receiverName.length==0){
        receiverNameError.innerHTML='receiver name is required';
        return false;
    }
```

```
    if(!receiverName.match(reg)){
        receiverNameError.innerHTML='receiver name only contians letter';
        return false;
    }
    receiverNameError.innerHTML='<i class="fa-solid fa-check"></i>';
    return true;
}
function validateRemark(){
    var remark=document.getElementById('remark').value;
    var required=40;
    var left=required-remark.length;
    if(left>0){
        remarkError.innerHTML=left+'more char is left';
        return false;
    }
    remarkError.innerHTML='';
    return true;
}
function isValidDate(dateString) {
    var dateRegex = /^(\d{4})-(\d{2})-(\d{2})$/;
```

```
if (dateRegex.test(dateString)) {

    var selectedDate = new Date(dateString);
```

6.6.2. System Testing

System test insure that the entire integrated software system meets requirements. It tests a configuration to insure known and predictable results. System testing is based on process description and flows, emphasizing pre-driven process links and integration points. In essence system testing is not about checking the individual part of design, but about checking the system as a whole in effect it is one giant component.

6.6.3. Integration Testing

In this testing part, all the modules would be combined together and tested it for it fitness with each other and with the systems functionality. If error occurs in combining them, the module with problem will be identified and recombined. Performance: So that based on the above testing suits the system has high performances that the system works correctly since it detects many invalid inputs. Then we try with valid input then the system accepts and we entered invalid inputs. Then we try with valid input then the system accepts and we entered an invalid input that the system rejects. As an administrator we apply performance testing, because

when the manager enters an invalid input the system responds him/her to enter the correct input.

6.6.4. Acceptance Testing

The process by which the actual users test a completed information system. The end result of which is the user acceptance of the system. Acceptance testing checks the system delivers what was requested.

CHAPTER SEVEN

7. CONCLUSION AND RECOMMENDATION

7.1. Conclusion

Gold Water Company has been used a manual inventory management system for managing or controlling, keeping and storing the materials and production. This project enhances and speed ups the working process of the registration items and all activities that happened in the company by saving time, man power. As it tried to show all of the procedures of the entire project step by step the result of the project seems successful. We start the project by analyzing the problem and then we make a design for the system. The testing we use testing by requirement for the purpose of reliability and security. But the success of the software project is inspected in work environment after implementation. We performed requirements analysis to discover the needs of the new solution to proposed system. This phase consists of drawing out functional and nonfunctional requirements of the system. In analysis, we tried to model the new and proposed inventory management system using UML diagrams: use case, sequence, class, activity diagrams. In design phase, we extend our work in analysis with more models. The class type architecture, class diagrams in analysis were extended in design to step closer to implementation, while deployment and component diagrams were also drawn. We also used persistence modeling to have a feel of the database that would accompany the system. Our development team believes that we have at large accomplished what we set out for the beginning of this project. We set to develop a working system prototype that would solve the problems in the beginning of our project. But before winding up, we would like to give our final word on how to go from here by recommending further improvement that could be made in the future.

7.2. Recommendation

While doing this system the team members has faced different challenges. But by the cooperation of all the group members and the Advisor the team is now able to reach to the final result. I.e. all the group members strongly fight these challenge and take the turn to the front. So now all the group members strongly recommend the department that for

the coming students, it has to provide them with better service than the present in better hard ware, guaranteed software's, giving orientations how to proceed, offering guest to provide them with more experienced work, support morally, manually, forming good relation with students, giving students description of each phases and so on. So that it will get what it expects from its students and satisfy with them. Neatly, the team would recommend that further work should be done on the system in order to make the system perform better.

- System needs that manager should not notify the staff members to replace the item or pay the amount to the store keeper if they lost or damage the item.
- Our system does not do purchasing also we recommend that it needs online purchase material.
- Finally we recommend that who are voluntary to add the above listed features of the system to be fulfilled us will give full documentation with coding.

CHAPTER EIGHT

8. REFERENCES

REFERENCES

[1] G. Booch, "Object Oriented Design," in *Object Oriented System Analysis and Design 2nd edition*, California, USA, 1998, p. 36.

[2] P. Jalote, "Development Methdology," in *Concise Introduction ToSoftware Engineering*, USA, Library of Cataloguine, 2008, p. 19.

[3] B. Brugger, "Functional Requirement," in *Object Oriented Software Engineering Third Edition*, Germeny, USA, 1994, p. 14.

[4] B. Bruegge, "nonfunctional Requirment," in *Object Oriented Software Engineering Third Eidtion*, Germeny, USA, 1994, p. 14.

[5] W. Boggs, "Uscase Diagram," in *Mestering UML With Rational Rose*, Alameda, Richard Miils, 2002, p. 14.

[6] B. Brugge, "Object Model," in *Object Oriented Software Engineering Third Edition*,

Germeny, USA, 1994, p. 30.

[7] G. Booch, "Object Model," in *Object Oriented System Analysis and Design 2nd Edition*, California, USA, 1998, p. 15.

[8] B. Bruegge, "class diagram," in *Object Oriented Software Engineering Using UML Third Edition*, Germeny, 1994, p. 32.

[9] A. Sereberink, "package diagram," in *software specification and architecture*, p. 9.

[10] Final project documentation from 2010-2013

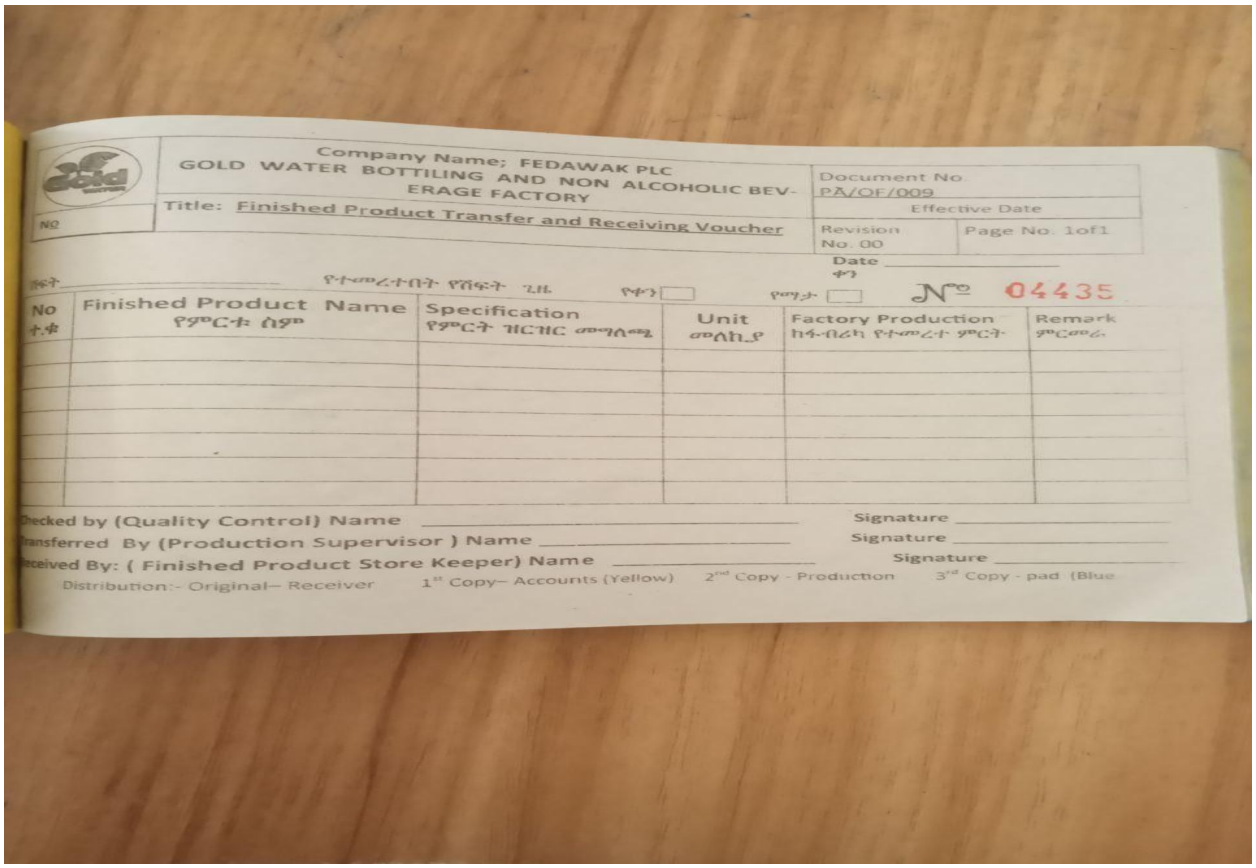
[11] https://www.w3schools.com

CHAPTER NINE

9. APPENDICES

9.1. Appendix A: Existing Forms and Reports

Appendix A



Company Name; FEDAWAK PLC
GOLD WATER BOTTILING AND NON ALCOHOLIC BEV-ERAGE FACTORY

Document No. PA/QF/009
Effective Date
Revision No. 00
Page No. 1of1

Title: Finished Product Transfer and Receiving Voucher

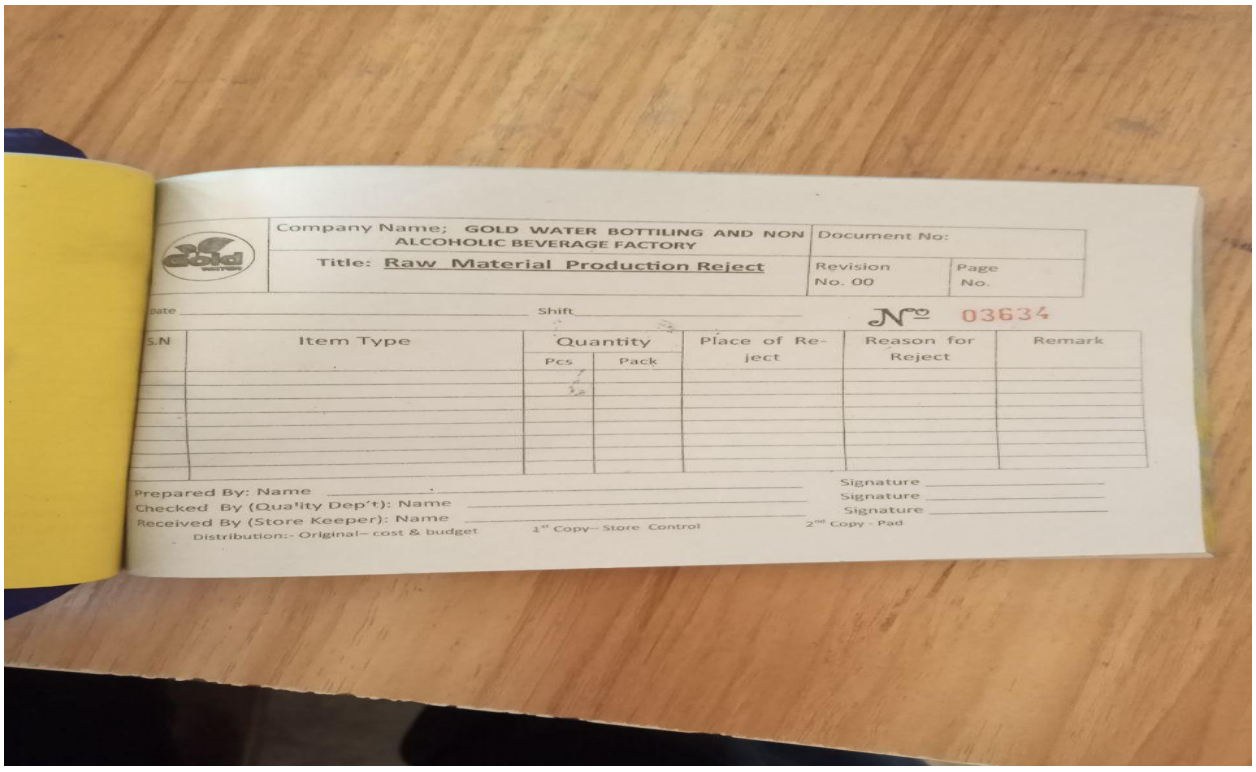
Date
ቀን

No 04435

No ተ.ቁ	Finished Product Name የምርቱ ስም	Specification የምርት ዝርዝር መግለጫ	Unit መለኪያ	Factory Production ከፋብሪካ የተመረተ ምርት	Remark ምርመራ

Checked by (Quality Control) Name _____ Signature _____
Transferred By (Production Supervisor) Name _____ Signature _____
Received By: (Finished Product Store Keeper) Name _____ Signature _____
Distribution:- Original– Receiver 1st Copy– Accounts (Yellow) 2nd Copy - Production 3rd Copy - pad (Blue

Appendix A 1. Item transferring form



Company Name; GOLD WATER BOTTILING AND NON ALCOHOLIC BEVERAGE FACTORY

Document No:

Title: Raw Material Production Reject

Revision No. 00

Page No.

Date _____ Shift _____ № 03634

S.N	Item Type	Quantity		Place of Re-ject	Reason for Reject	Remark
		Pcs	Pack			

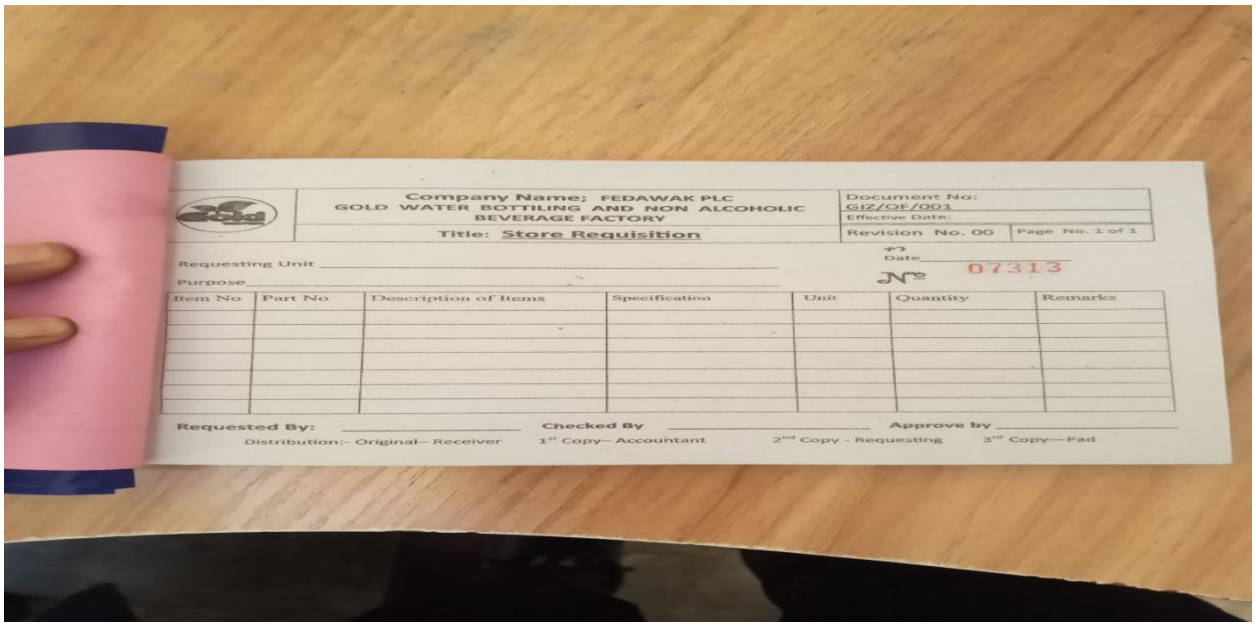
Prepared By: Name _____ Signature _____

Checked By (Qua'ity Dep't): Name _____ Signature _____

Received By (Store Keeper): Name _____ Signature _____

Distribution:- Original– cost & budget 1st Copy– Store Control 2nd Copy - Pad

Appendix A 2. Product Rejection Form



Company Name; FEDAWAK PLC GOLD WATER BOTTILING AND NON ALCOHOLIC BEVERAGE FACTORY

Document No: GIZ/OF/001

Effective Date:

Title: Store Requisition

Revision No. 00 Page No. 1 of 1

Requesting Unit _____ Date _____

Purpose _____ № 07313

Item No	Part No	Description of Items	Specification	Unit	Quantity	Remarks

Requested By: _____ Checked By _____ Approve by _____

Distribution:- Original– Receiver 1st Copy– Accountant 2nd Copy - Requesting 3rd Copy– Pad

Appendix A 3. Store Request Form