KARNATAK UNIVERSITY DHARWAD

Janata Shikshana Samiti's

Shri Manjunatheshwara Institute of UG & PG Studies, Vidyagiri, Dharwad-580 004.

A PROJECT REPORT ON

"CO-OPERATIVE SOCIETY MANAGEMENT SYSTEM"

BACHELOR OF COMPUTER APPLICATIONS (BCA) OF KARNATAK UNIVERSITY, DHARWAD

PROJECT GUIDED BY:

Prof. Venus Mishra

Submitted By

K. N. Keerti

BCA VI SEMESTER

REG NO: U02BF22S0001

DEPARTMENT OF COMPUTER SCIENCE 2024-2025

JANATA SHIKSHANA SAMITI’S

SHRI MANJUNATHESHWARA INSTITUTE OF UG & PG STUDIES, VIDYAGIRI, DHARWAD - 580004

CERTIFICATE

This is to certify that Ms. K. N. Keerti has satisfactorily completed Project Work entitled “CO-OPERATIVE SOCIETY MANAGEMENT SYSTEM ” for the partial fulfillment of BCA prescribed by Karnatak University Dharwad during the academic year 2024-2025.

Prof. Venus Mishra	**Prof. Vivek M Laxmeshwar**	**Dr. Ajith Prasad**
Project Guide	[HOD] Computer Department	Principal

Examiners:

1)

2)

ACKNOWLEDGEMENT

The successful presentation of this project is acknowledgements of the immense support extended by JSS SMI UG & PG STUDIES, DHARWAD, which has provided us opportunity to fulfil the most cherished & desired way to reach our goal.

I would like to express my heartfelt thanks to our President Shri Shri Vishwaprasanna Theertha Swamiji of Sri Pejavarmath, Udupi, Parama Pujya Dr. D. Veerendra Heggade ji, the Chairman of Janata Shikshana Samiti & Dharmadhikari of Dharmasthala.

I would like to express my sincere and heartfelt gratitude to our beloved Principal Dr. Ajith Prasad, for his unwavering support and invaluable time. His constant encouragement and guidance have been a source of great motivation throughout our journey. I am truly grateful for his inspiring leadership and commitment to my success.

I would also like to express my gratitude towards our Head of the Computer Science Department, Prof. Vivek M. Laxmeshwar, who gave us the golden opportunity to do this wonderful project on the topic "Co-operative Society Management System," which helped us conduct extensive research and exposed us to a lot of new information that will surely benefit us in the near future.

I would also take this opportunity to offer my sincere gratitude to my Project Guide Mrs. Venus Mishra for her excellent support throughout the development of this project and providing the necessary information on our demand at all the times.

K. N. Keerti

DECLARATION

I, Ms. K.N.Keerti, Student of Sixth Semester BCA, Department of Computer Science, JSS SHRI MANJUNATHESHWARA INSTITUTE OF UG & PG STUDIES, VIDYAGIRI, DHARWAD affiliated to Karnatak University, Dharwad hereby declare that the Project entitled "Co-operative Society Management System " submitted in partial fulfilment of the course requirement for the Award of Degree in Bachelor of Computer Application, Karnatak University, Dharwad during the Academic Year 2024-2025. I have not submitted the matter embodied to any other University or Institution for the Award of any other Degree.

Date: **K. N. Keerti**

Place: Dharwad

CONTENTS

1.PROJECT SYNOPSIS

1.1 TITLE OF THE PROJECT

Menasi Seemeya Group Gramagala Seva Sahakari Sangha Niyamita vanalli Co-operative Society Management System

1.2 INTRODUCTION

A **Cooperative Society Management System (CSMS)** is a software solution designed to automate and streamline the administrative and operational processes of cooperative societies. It is a voluntary association of individuals who come together to achieve a common economic goal. The society provides various services to its members, including loans, shares, and other financial products.

In order to manage the day-to-day operations of the society efficiently, a Cooperative Society Management System is essential. This system provides a comprehensive solution for managing member information, share transactions, loan applications, and financial transactions. The CSMS aims to replace traditional manual methods with a centralized digital platform, improving efficiency, transparency, and accuracy.

1.3 OBJECTIVE OF THE PROJECT

- Automate society operations
- Improve member satisfaction
- Enhance financial management
- Increase efficiency and productivity
- Provide accurate reporting and analytics
- Data will be stored, maintained and access by a computer software

1.4 INPUT OF THE PROJECT

- **Admin Details :** Username, password, role
- **Members details :** Personal info, employment status, member ID, account info
- **Deposits and withdrawal :** Transaction date, amount, account number, transaction type
- **Balance requests :** Account number, member ID
- **Transaction extract request :** Account number, data range, transaction
- **Product Details :** Product ID, product name, description, price, quantity in stock

1.5 OUTPUT OF THE PROJECT

- **Admin details :** Admin dashboard, acess to manage employee/member data
- **Members records:** Members profile, account information, transaction history
- **Deposits & withdrawal confirmation :**Trasaction confirmation, updated account balance, transaction receipt
- **Balance request response :** Current account balance, account statement
- **Trasaction extract report:** Trasaction history report, detailed transaction list,
- **Product details output:** Product catalog with name, price, availability status, and description

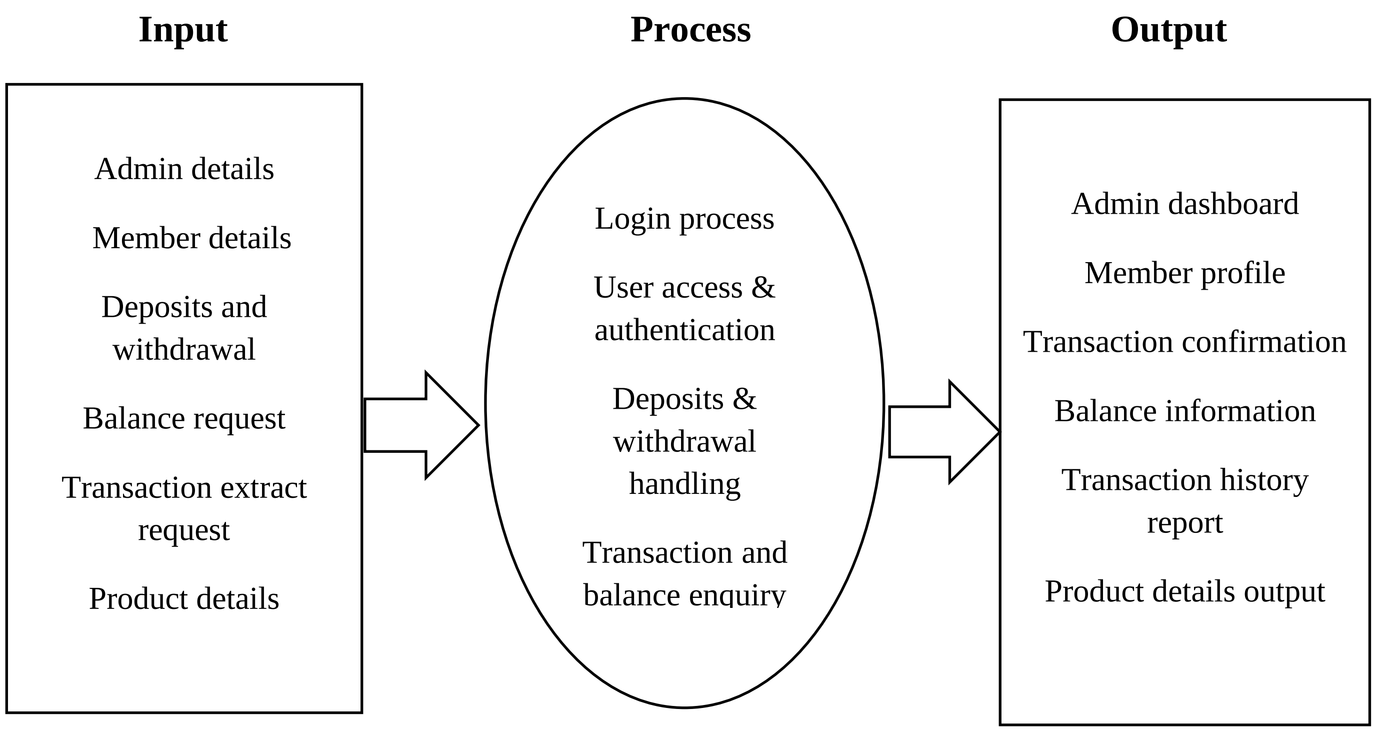
1.6 MODULES

1. **Admin Module**
   - Admin login
   - Manage member accounts
   - Manage product details
2. **Member Management Module**
   - Add/update member details
   - View member profiles
   - Assign membership IDs
3. **Transaction Management Module**

- Record deposit and withdrawal transactions
- View transaction history
- Generate and print transaction receipts

4. **Balance Inquiry Module**
   - Process balance check requests
   - Display current balance of member accounts
5. **Transaction Extract Module**
   - Generate transaction history for a given date range or member
6. **Product Management Module**
   - Add new products (ID, name, price, stock quantity, etc.)
   - Update and delete product details

1.7 PROCESS LOGIC



1.8 TOOLS / PLATFORM, LANGUAGES TO BE USED

HARDWARE REQUIREMENTS

➢ Hard disk	: 50GB or above
➢ Processor	: Dual core processor(e.g.,Intel core i5 &above , AMD ryzen3 &above)
➢ RAM	: 8GB or more

SOFTWARE REQUIREMENTS

➢ Operating Environment	: windows, Linux
➢ Front End	: HTML, CSS, JavaScript
➢ Backend	: MYSQL
➢ Middleware	: PHP
➢ Hosting	: XAMPP server

1.9 ARE YOU DOING THIS PROJECT FOR ANY INDUSTRY/CLIENT? IF YES, ACCEPTANCE OF THE PROJECT.

- Yes , Co-Operative society

1.10 DURATION OF THE PROJECT

- 2 Months

1.11 MEMBER OF THE PROJECT

- K. N. KEERTI [U02BF22S0001]

1.12 LIMITATIONS OF THE PROJECT

- Scalability issues
- Security risk
- No multi -branch support

1.13 SCOPE OR FUTURE APPLICATIONS

- The system reduces manual work.
- Enable online access for members to view their details from anywhere.
- Automate daily market updates via WhatsApp, SMS, and email notifications.
- Gather feedback from society staff to refine and optimize the system based on actual usage.
- Implement regular data backups for efficient issue recovery and data security.
- Online payment facility can be provided.

2. INTRODUCTION

In today's digital era,, manual management of financial and service-based institutions has become inefficient. Cooperative societies, especially in rural and semi-urban areas, play a vital role by offering banking services and distributing essential products to members. However, many still depend on paper-based systems, leading to errors, delays, and poor service.

The **Co-Operative Society Management System (CSMS)** is developed to overcome these challenges. It is a centralized, computerized platform that simplifies and automates core society functions such as member data management, deposits and withdrawals, balance inquiries, transaction reports, and product inventory tracking.

This system replaces manual tasks with a fast, accurate, and user-friendly solution. Designed for simplicity and scalability, it helps societies adopt digital practices with ease. CSMS enhances transparency, improves data accuracy, and enables quick responses to member needs.

By using this system, cooperative societies can modernize their operations, build trust with members, and move towards efficient, digital-first service delivery.

MAIN USERS OF THE SYSTEM

- Administrator: Has full control over the system. Responsible for managing members, transactions, and product details.
- Members: Individuals registered in the society. Through admin assistance, they can view their account details, request balance updates, and inquire about transactions and available products.

KEY FEATURES OF THE SYSTEM

- Add, update, and manage member records with unique IDs and account information.
- Record and track deposits and withdrawals with automated balance updates and transaction receipts.
- Respond to member requests to check their current account balance.
- Maintain a catalog of products with product ID, name, price, stock quantity, and availability.
- Provide a simple and user-friendly interface for smooth admin operation.

The system is built using a combination of modern web technologies:

- **PHP**
  Used as the server-side language to handle backend logic, manage user authentication, process transactions (deposits, withdrawals), and communicate with the database.

- **MYSQL**
  A relational database used to securely store member and employee records, account balances, transaction history and product details.

- **HTML&CSS**
  HTML structures the content while CSS styles the interface, ensuring a clean, user-friendly, and responsive design for all users.

- **JAVASCRIPT**
  Adds interactivity to the web application, such as form validation, dynamic content updates, and smooth user experiences.

- **BOOTSTRAP**
  A front-end framework that ensures mobile responsiveness and consistency across devices using pre-built components and grid systems.

The **Co-Operative Society Management System** simplifies and automates core activities of cooperative societies.It improves accuracy, saves time, and enhances service delivery to members.With an easy-to-use interface, it ensures smooth and efficient operations.This system helps societies move toward a more transparent and digital future.

PHP FRAMEWORK

XAMPP ARCHITECTURE

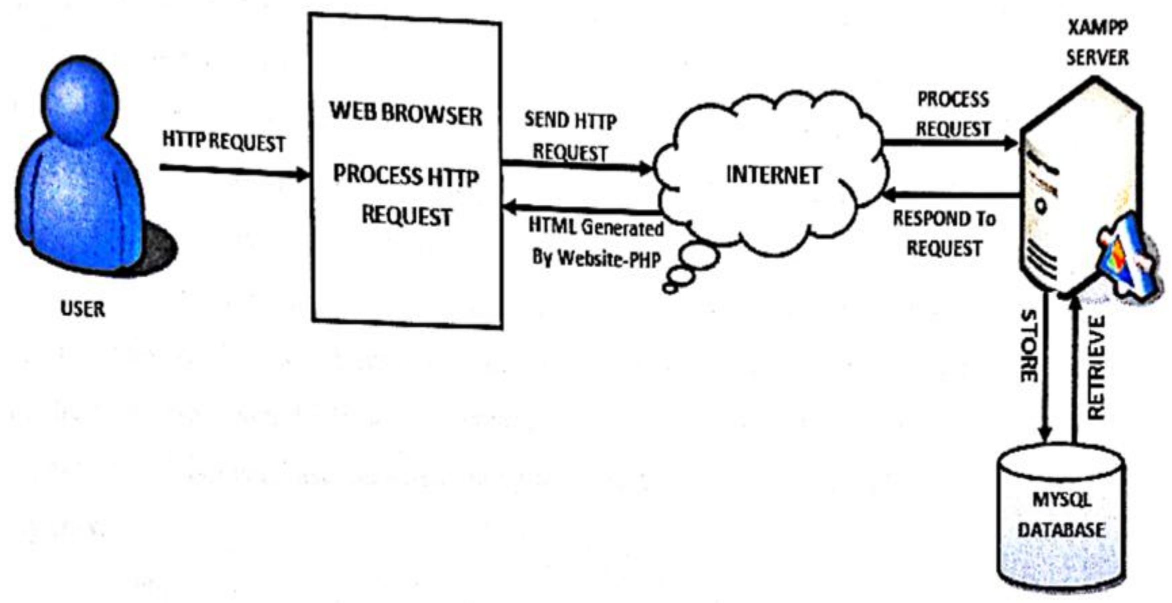


FIGURE 1: XAMPP ARCHITECTURE

XAMPP stands for Cross-Platform (X), Apache (A), MySQL (M), PHP (P) and Perl (P). It is a simple, lightweight Apache distribution that makes it extremely easy for developers to create a local web server for testing purposes. Everything you need to set up a web server server application (Apache), database (MySQL), and scripting language (PHP) is included in a simple extractable file. XAMPP is also cross-platform, which means it works equally well on Linux, Mac and Windows. Since most actual web server deployments use the same components as XAMPP, it makes transitioning from a local test server to a live server extremely easy.

What's included in XAMPP?

XAMPP has three primary components. These are:

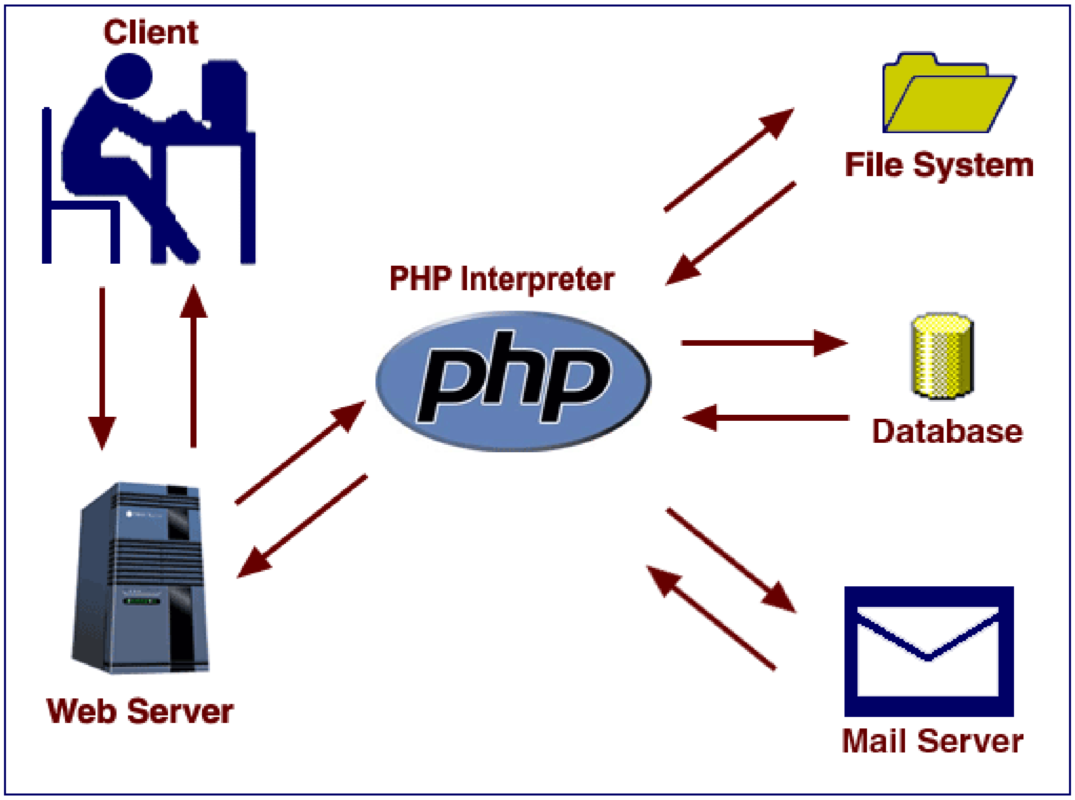
- **Apache:** Apache is the actual web server application that processes and delivers web content to a computer. Apache is the most popular web server online, powering nearly 54% of all websites.
- **MySQL:** Every web application, howsoever simple or complicated, requires a database for storing collected data. MySQL, which is open source, is the world's most popular database management system. It powers everything from hobbyist websites to professional platforms like WordPress etc.

Hypertext Pre-Processor (PHP)

PHP (Hypertext Preprocessor) is a widely-used, open-source server-side scripting language primarily designed for web development. It is highly effective in creating dynamic web pages and applications. PHP can be embedded directly into HTML code, making it a versatile tool for developing both simple and complex websites.

Originally developed by Rasmus Lerdorf in 1993, PHP has evolved into one of the most popular server-side programming languages due to its flexibility, ease of use, and integration with databases such as MySQL, PostgreSQL, and others.

PHP ARCHITECTURE



FEATURES OF PHP

1.Open-Source

PHP is free to use and distribute, with its source code available for modification. This makes it highly flexible and cost-effective.

2.Server-Side-Scripting

PHP runs on the server to process user requests and send back HTML content. It handles backend logic like form submission and database access.

3.Database-Integration

PHP supports databases like MySQL for storing and retrieving data. It enables easy management of records in dynamic web applications.

4.Cross-Platform-Compatibility

PHP works on various operating systems such as Windows, Linux, and macOS. It can be used with most popular web servers.

5.Embedded-in-HTML

PHP can be written within HTML code to create dynamic pages. This simplifies web development and speeds up coding.

6.Supports-Object-Oriented-Programming(OOP)

PHP supports OOP concepts like classes, objects, inheritance, and polymorphism. This helps organize complex code and improves code reusability.

WHAT IS MYSQL DATABASE?

MYSQL is a freely available open source Relational Database Management System

(RDBMS) that uses Structured Query Language (SQL). SQL is the most popular language for

adding, accessing and managing content in a database. It is most noted for its quick processing,

proven reliability, ease and flexibility of use.

FEATURES OF MYSQL

1.Relational Database System: Like almost all other database systems on the market, MySQL

is a relational database system.

2.Client/Server Architecture: MySQL is a client/server system. There is a database server

(MySQL) and arbitrarily many clients (application programs), which communicate with the

server; that is, they query data, save changes, etc. The clients can run on the same computer as the server or on another computer (communication via a local network or the Internet).

ADVANTAGES OF MYSQL

1. It is easy to use: While a basic knowledge of SQL is required—and most relational databases require the same knowledge—MySQL is very easy to use. With only a few simple SQL statements, you can build and interact with MySQL.

2. It is secure: MySQL includes solid data security layers that protect sensitive data from intruders. Rights can be set to allow some or all privileges to individuals. Passwords are encrypted.

3. It is inexpensive: MySQL is included for free with NetWare® 6.5 and available by free download from MySQL Web site.

4.It is fast: In the interest of speed, MySQL designers made the decision to offer fewer features than other major database competitors, such as Sybase and Oracle .However, despite having fewer features than the other commercial database products, MySQL still offers all of the features required by most database developers.

5.It's scalable: MySQL can handle almost any amount of data, up to as much as 50 million rows or more. The default file size limit is about 4 GB. However, you can increase this number to a theoretical limit of 8 TB of data.

6. It runs on many operating systems: MySQL runs on many operating systems, including Novell NetWare, Windows* Linux*, many varieties of UNIX* (such as Sun* Solaris*, AIX, and DEC* UNIX), OS/2, FreeBSD*, and others.

7. It manages memory very well: MySQL server has been thoroughly tested to prevent memory leaks.

HTML

HTML means Hypertext Markup Language. HTML is a method of describing the format of documents which allows them to be viewed on computer screens. HTML documents are displayed by web browsers, programs which can navigate across networks and display a wide variety of types of information. HTML pages can be developed to be simple text or to be complex multimedia extravaganzas containing sound, moving images, virtual reality, and Java applets.

The global publishing format of the Internet is HTML. It allows authors to use not only text but also format that text with headings, lists, and tables, and to include still images, video, and sound within text. Readers can access pages of information from anywhere in the world at the click of a mouse-button. Information can be downloaded to the reader‘s own PC or workstation. HTML pages can also be used for entering data and as the front-end for commercial transactions.

FEATURES OF HTML

- It is not a programming language.
- It is not a data description language.
- It is simple to understand and implement.
- HTML constructs a very easy to comprehend, and can be used effectively by anybody.
- The methodology used by HTML to mark up information is independent of its representation on a particular hardware or software architecture.
- HTML syntax is a worldwide standard.

CSS

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language like HTML. CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript**.**

CSS is designed to enable the separation of presentation and content, including layout, colours, and fonts. This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple web pages to share formatting by specifying the relevant CSS in a separate .css file, and reduce complexity and repetition in the structural content.

Separation of formatting and content also makes it feasible to present the same markup page in different styles for different rendering methods, such as on-screen, in print, by voice (via speech-based browser or screen reader), and on Braille-based tactile devices. CSS also has rules for alternate formatting if the content is accessed on a mobile device.
The name cascading comes from the specified priority scheme to determine which style rule applies if more than one rule matches a particular element. This cascading priority scheme is predictable.Cascading Style Sheets, fondly referred to as CSS, is a simply designed language intended to simplify the process of making web pages presentable. CSS allows you to apply styles to web pages. More importantly, CSS enables you to do this independent of the HTML that makes up each web page.

CSS SYNTAX:-

A CSS comprises of style rules that are interpreted by the browser and then applied to the corresponding elements in your document.
A style rule set consists of a selector and declaration block.

Selector => h1

Declaration => {color: blue; font size: 12px ;}

The selector points to the HTML element you want to style.

The declaration block contains one or more declarations separated by semicolons.

Each declaration includes a CSS property name and a value, separated by a colon.

JAVA SCRIPT

- ➢ JavaScript is a very powerful client-side scripting language.
- ➢ JavaScript is used mainly for enhancing the interaction of a user with the webpage.
- ➢ In other words, you can make your webpage more lively and interactive, with the help of JavaScript.
- ➢ The language was initially called Live Script and was later renamed JavaScript.
- ➢ The syntax of JavaScript is mostly influenced by the programming language C.
- ➢ JavaScript is a cross-platform, object-oriented scripting language used to make WebPages interactive (e.g. having complex animations, clickable buttons, popup menus, etc.).
- ➢ JavaScript contains a standard library of objects, such as Array, Date, and Math, and a core set of language elements such as operators, control structures, and statements.
- ➢ Client-side JavaScript extends the core language by supplying objects to control a browser and it‘s Document Object Model (DOM).
- ➢ For example, client-side extensions allow an application to place elements on an HTML form and respond to user events such as mouse clicks, form input, and page navigation.
- ➢ Server-side JavaScript extends the core language by supplying objects relevant to running JavaScript on a server.
- ➢ For example, server-side extensions allow an application to communicate with a database, provide continuity of information from one invocation to another of the

application, or perform file manipulations on a server.

- This means that in the browser, JavaScript can change the way the webpage (DOM) looks. And, likewise, Node.js JavaScript on the server can respond to custom requests from code written in the browser.

BOOTSTRAP

Bootstrap is a popular open-source front-end framework developed by Twitter for designing responsive and mobile-first web pages. It provides a large collection of predesigned HTML,CSS, and JavaScript components such as navigation bars, buttons, forms, modals, and carousels, which help developers build attractive and consistent user interfaces quickly. Bootstrap uses a grid system that makes it easy to create layouts that automatically adjust to different screen sizes, ensuring websites look great on desktops, tablets, and smartphones.

One of the main advantages of using Bootstrap is that it saves development time and reduces the need for writing custom CSS from scratch. Its built-in classes and responsive utilities allow developers to implement modern designs with minimal effort, while still offering the flexibility to customize styles as needed. Bootstrap also includes extensive documentation and active community support, making it a go-to tool for both beginners and experienced web developers.

3. SYSTEM ANALYSIS

INTRODUCTION

System analysis is a crucial phase in the development of any software system. It helps in understanding user requirements, data flow, and the key operations the system must support. For the **Cooperative Society Management System,** the analysis phase involved a thorough examination of the existing manual or semi-digital processes. This included identifying inefficiencies in managing member records, handling financial transactions, and tracking product inventory. The goal was to define clear requirements for an automated system that would improve accuracy, enhance accessibility, and streamline cooperative operations.

1.EXISTING SYSTEM

- Most cooperative societies use manual methods or basic tools like spreadsheets.
- Member and transaction records are maintained physically or in unstructured digital files.
- Calculations for deposits, withdrawals, and balances are done manually.
- Report generation and account statements are time-consuming and not automated.
- Data is scattered, making it hard to access and manage efficiently.

LIMITATIONS OF THE EXISTING SYSTEM

- Time-consuming and tedious data handling.
- High chances of human error in calculations and records.
- Difficult to manage, retrieve, and secure large volumes of member data.

- No centralized system for real-time transaction or balance tracking.
- Poor tracking of financial activities like loans and savings.
- Inadequate data security and backup mechanisms.

2. PROPOSED SYSTEM

The Cooperative Society Management System overcomes these challenges by providing a secure, web-based solution for managing all operations efficiently.

FEATURES OF THE PROPOSED SYSTEM

- Digital entry and storage of member, employee, and product details.
- Automated handling of deposits, withdrawals, and balance calculations.
- Real-time access to transaction history and account statements.
- Admin dashboard to monitor overall system activities and generate reports.
- Responsive interface accessible via desktops, tablets, and smartphones.
- Secure login system with role-based access and data protection.
- Easy retrieval of data and backups to prevent information loss.

FEASIBILITY STUDY

Preliminary investigations assess whether the proposed system will be useful, efficient, and cost-effective. The system aims to replace manual processes in cooperative societies with a centralized software solution. Three key types of feasibility were evaluated:

1. OPERATIONAL FEASIBILITY

- The proposed system is easy to use and understand.
- It requires no special training for the admin to operate.
- The system follows the current business rules and processes of cooperative societies.
- It has a user-friendly interface with clear navigation and responsive design.
- Members can easily access account details and transaction extracts through admin help.
- Reduces physical effort and paperwork for society staff and members.

2. TECHNICAL FEASIBILITY

- The system uses commonly available technology (PHP, MySQL, HTML, JavaScript).
- It does not require high-end hardware; a basic computer setup is sufficient.
- Can be deployed using WAMP or XAMPP servers on Windows or Linux OS.
- The technical infrastructure needed is minimal and affordable.
- Easily maintainable and scalable for future upgrades if needed.

3. FINANCIAL FEASIBILITY

- The development and deployment of the system are cost-effective.
- No need for expensive hardware or third-party licenses.
- Saves costs associated with manual record-keeping and errors.
- Reduces time spent on administrative tasks, improving overall efficiency.
- System maintenance is low-cost due to the use of open-source tools.
- Increases return on investment by improving speed, accuracy, and user satisfaction.

4. SYSTEM DESIGN

INTRODUCTION

System design is a crucial phase in the development of the **Co-Operative Society Management System**, as it lays out the blueprint for how the system will function. This stage involves planning how different components—such as member management, transaction handling, and product tracking—will interact to fulfill the requirements identified during system analysis.

The primary goal is to design a simple, effective, and reliable system that automates the daily operations of a cooperative society. The system will be managed by an admin, who performs all core functions such as adding members, recording deposits and withdrawals, checking balances, and managing product details.

During this phase, key design elements such as system architecture, database schema, user interface layout, and data flow are defined. These components ensure that the system is easy to use, minimizes errors, and improves the efficiency of administrative tasks.

Overall, the design ensures that the system is not only functional but also scalable and easy to maintain, with potential for future features like multi-branch support, member logins, and notification systems.

SOFTWARE DESIGN

The **Cooperative Society Management System** is designed with a modular and scalable approach to efficiently manage tasks such as member registration, financial transactions, product tracking, and reporting. It follows a three-tier architecture: the **presentation layer** (HTML, CSS, Bootstrap, JavaScript), the **business logic layer** (PHP), and the **data layer** (MySQL database). This architecture ensures that the system remains maintainable, easy to scale, and performs optimally even as the number of members and transactions increases.

The **database** is carefully structured with normalized tables and established relationships between entities, such as members, transactions, and products. Primary and foreign keys are used to maintain data integrity and ensure that the system operates smoothly. Security measures, such as **prepared SQL statements** and **password hashing**, are incorporated to protect sensitive user data and prevent common security vulnerabilities like SQL injection.

The user interface is built with **Bootstrap**, which ensures that the system is responsive and offers a user-friendly experience on both desktop and mobile devices. Additionally, **AJAX** is utilized to enhance system performance by dynamically loading data without requiring full-page refreshes, resulting in a faster and more interactive user experience.

Key modules include **role-based login** for secure access, **transaction management** for automated deposits, withdrawals, and loan requests, and **member and employee management** for easy profile and transaction tracking. The system also features **product management** for real-time inventory tracking and **financial reporting** for instant transaction summaries. The design is flexible, ensuring future scalability and security.

ARCHITECTURE DESIGN

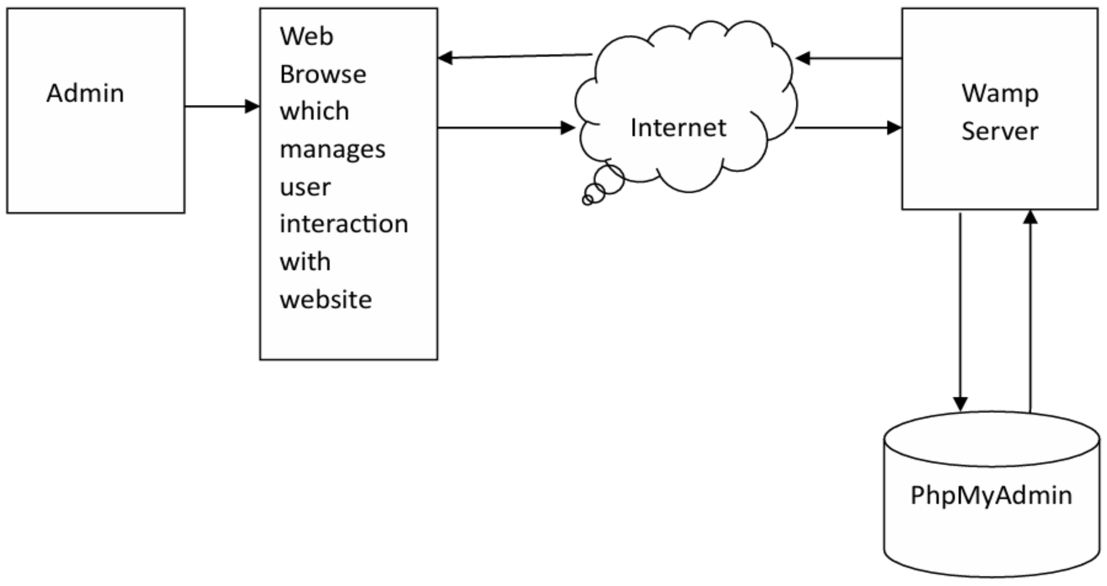
The **Cooperative Society Management System** is designed using a three-tier architecture, which separates the presentation, business logic, and data layers. This approach enhances the system's flexibility, scalability, and maintainability, ensuring it can handle growing data and future feature additions effectively.

1. **Presentation Layer**
   - **Technologies**: HTML, CSS, Bootstrap, JavaScript
   - **Description**: This is the front-end interface that interacts directly with the users, including admins and members. It features elements such as login screens, dashboards for managing accounts and transactions, member profiles, product

listings, and financial reports. **AJAX** is employed to enhance responsiveness by dynamically loading data without requiring page reloads.

2. **Business Logic Layer**
    - **Technologies**: PHP
    - **Description**: This layer handles the core operations of the system, including user authentication, role-based access management, transaction processing, and communication between the front-end and back-end. It manages operations such as deposit and withdrawal processing, account updates, and generating reports. The PHP layer ensures that all user actions are executed securely and efficiently.

3. **Data Layer**
    - **Technologies**: MySQL Database
    - **Description**: This layer is responsible for storing all crucial data related to members, employees, transactions, products, and account balances. It maintains data integrity through normalized tables and relationships between different entities. The database layer supports fast data retrieval, updates, and secure data storage, ensuring that all operations are processed quickly and accurately.



This architecture ensures that each layer is focused on its specific task, improving performance, security, and ease of maintenance.

LOGICAL DESIGN

The graphical representation of systems' data and how the process transforms the data is known as Data Flow Diagram. It shows the logical flow of the data.

The logical design describes the detailed specification for the system, describing its features, an effective communication and the user interface requirements.

The logical system of proposed system should include the following.

1. External system structure.

2. Relationship between all the activities.

3. The physical construction and all the activities.

4. Global data.

5. Control flow.

6. Derived program structure.

DESIGN PRINCIPLES

Basic design principles that enable the software engineer to navigate the design process are:

• The design process should not suffer from “Tunnel vision”.

• The design should be traceable to analysis model.

• The design should not reinvent the wheel.

• The design should minimize the intellectual distance between the software and the

problem, as it exists in the real world.

• The design should exhibit uniformity and integrity.

• The design should be structured to accommodate changes.

• The design not coding, the coding is not a design.

• The design should be assessed for the quality, as it is being created, not after the fast.

• The design should be reviewed to minimize the conceptual errors.

System Design – Programming Approach

How Systems Are Built (Programming Approaches)

Software systems can be built using different **programming paradigms**—which define the structure and logic of the code. The most common approaches include:

1. **Procedural Programming:**
    - Based on the concept of **procedures or functions**.
    - Code is written as a series of steps or instructions.
    - Example in PHP: mysqli_connect(), mysqli_query()
    - Simple, but less modular and harder to scale for large systems.
2. **Object-Oriented Programming (OOP):**
    - Based on the use of **classes and objects**.
    - Helps group related data and functions together.
    - Promotes **modularity, reusability, and scalability**.
    - Example in PHP: new mysqli(), $object->method()
    - Used in most modern frameworks like Laravel, Symfony, etc.
3. **Plain Old Data (POD):**
    - A term mainly from **C++**, representing simple data structures without behavior (just data).
    - Not typically applicable in PHP projects.

Programming Paradigm Used in This Project

This project, **Co-Operative Society Management System**, was developed using the **Object-Oriented Programming (OOP)** approach in **PHP** with a **MySQL** backend.

- Database connections and operations were handled using the mysqli class, allowing better abstraction and code organization.

- Core modules like **Member Management, Transaction Handling, Product Records, and Billing** were implemented using **classes** and **object methods**.
- OOP allowed the system to be more **modular, maintainable**, and ready for **future enhancements**.

Why OOP Was Chosen

- Easier to manage growing system complexity.
- Promotes code reuse and separation of concerns.
- Aligns well with **MVC**-like architecture (even without a full framework).
- Makes it easier to implement features like **session handling**, **form validation**, and **report generation** in an organized way.

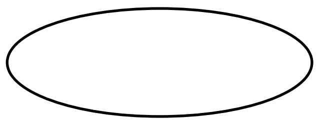
DATA FLOW DIAGRAM

Graphical description of system‘s data & how the processes transform the data is known as data flow diagram. It is also known as Bubble chart. DFD modules, system using external entities from which data flow to a process, which transforms the data, and create output data transforms which go to. Other processes or external entities like files. The main metric of DFD is that it can provide an overview of what data a system would process, what transformation of data are done, what files are used and where the result flows.

NOTATIONS USED IN DATA FLOW DIAGRAM

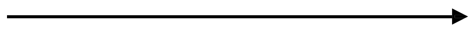
1.Functional Processing:

It is represented by an oval. The processing or main transactions are specified by this notation.



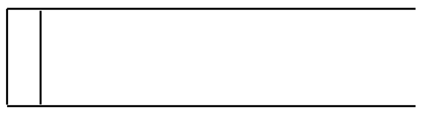
2.Data Flow:

It is represented by an arrow line and name of the data is specified by the side of the line as label. This is used for data movement.



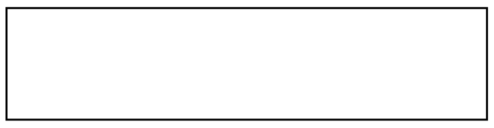
3.Data Store:

It is represented by one open-end rectangle. The databases used in the system are specified by this notation.



4.Source or sink:

It is represented by one open-end rectangle. It is used for specifying from where data comes and where it reaches.



STEPS REQUIRED IN DFD

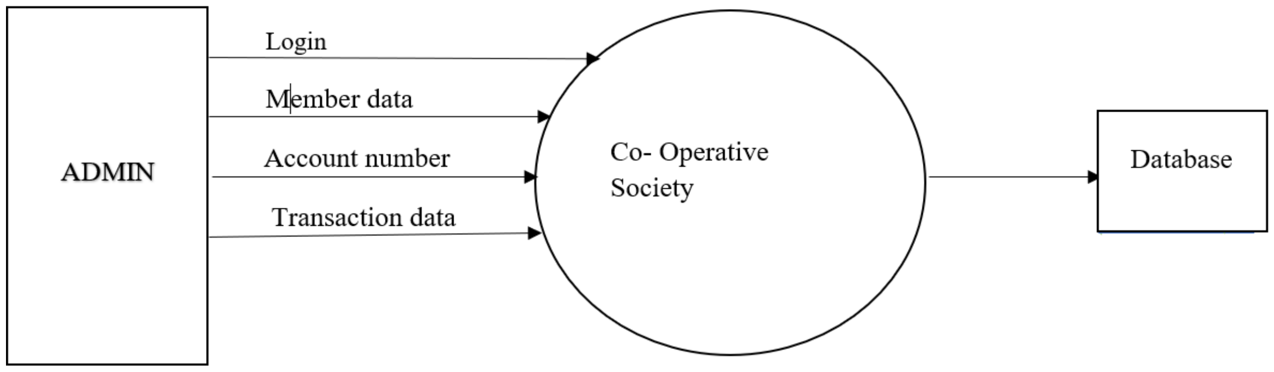
Several rules of thumb are used in drawing DFD'S:

- Process should be named and numbered for an easy reference. Each name should be representative of the process.
- The direction of flow is from top to bottom and from left to right. Data traditionally flow from source to the destination although they may flow back to the source. One way to indicate this is to draw long flow line back to a source. An alternative way is to repeat the source symbol as a destination. Since it is used more than once in the DFD it is marked with a short diagonal.
- When a process is exploded into lower level details, they are numbered.
- The names of data stores and destinations are written in capital letters. Process and dataflow names have the first letter of each work capitalized

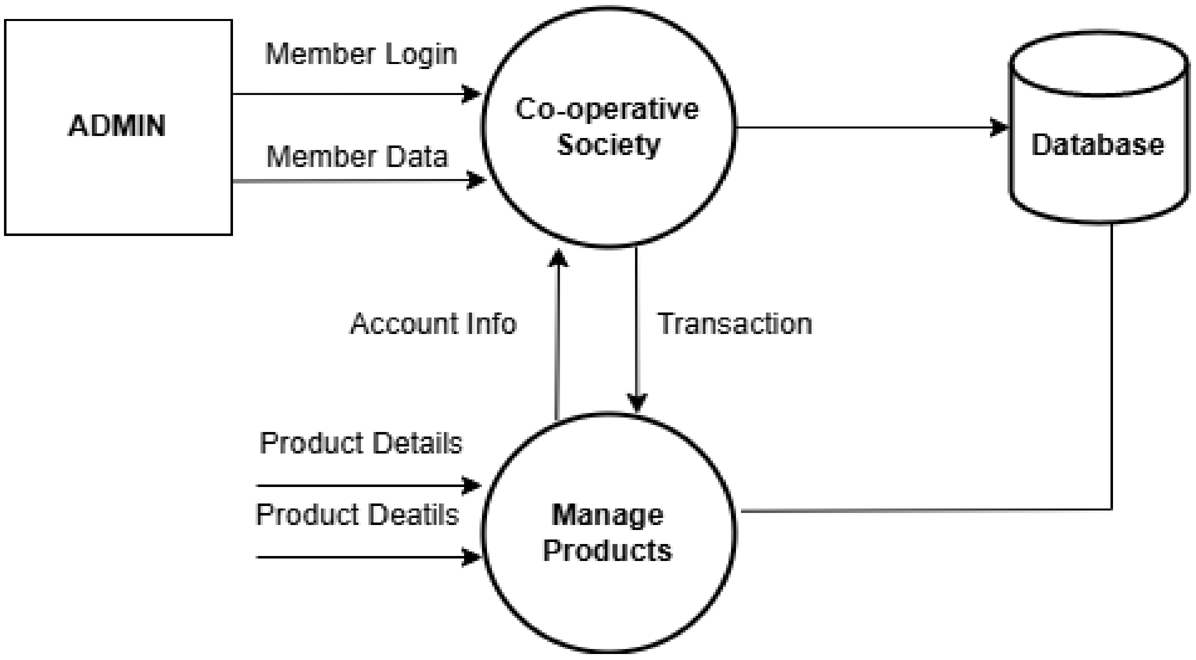
OBJECTIVES

- To graphically document boundaries of a system.
- To provide hierarchy breakdown of the system.
- To show movement of information between a system and its environment.
- To document information flows within the system.
- To aid communication between users and developers.

ZERO LEVEL DATA FLOW DIAGRAM



LEVEL ONE DATA FLOW DIAGRAM



CONTEXT MODEL

What is a Context Diagram?

A **Context Diagram** is a fundamental tool used in system analysis that provides a **high-level overview of a system**. It shows the system as a single process (or box) and identifies all the **external entities** that interact with it. These interactions are represented as **data flows** (inputs and outputs). It is also known as **Level 0 of a Data Flow Diagram (DFD)** because it does not break down the internal processes of the system but focuses on **external relationships and data exchange**.

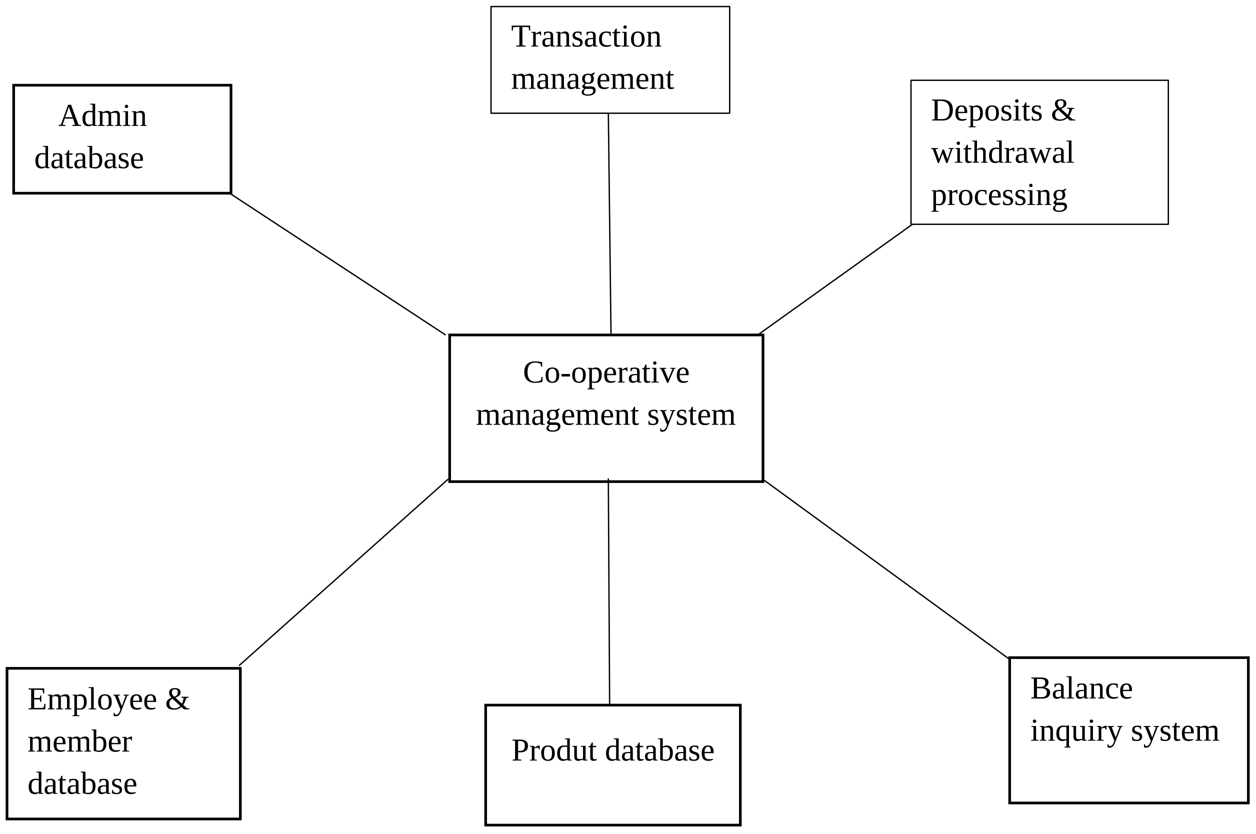
PURPOSE OF THE CONTEXT DIAGRAM

The primary purposes of a context diagram are to:

- Define the **boundary** of the system clearly.
- Identify all **external entities** that interact with the system.
- Show the **input and output data flows** between the system and external actors.
- Provide a **simple, visual representation** that is easy to understand for technical and non-technical stakeholders.
- Serve as the **starting point** for further system design and detailed data flow diagrams.

CONTEXT DIAGRAM OF CO-OPERATIVE SOCIETY MANAGEMENT SYSTEM

In the case of the **Co-Operative Society Management System**, the context diagram illustrates how the system interacts with the **key users** (Admin And Member) and the **database**. It describes the data that flows into and out of the system from each of these external entities.


Transaction management
Admin database
Deposits & withdrawal processing
Co-operative management system
Employee & member database
Produt database
Balance inquiry system

ENTITY - RELATIONSHIP MODEL

The entity-relationship model is based on a perception of the world as consisting of a collection of basic objects (entities) and relationships among these objects.An entity is a distinguishable object that exists.

- An entity is a distinguishable object that exists.
- Each entity has associated with it a set of attributes describing it.
- E.g. number and balance for an account entity.
- A relationship is an association among several entities.
- e.g. A customer account relationship associates a customer with each account he or she has.

- The set of all entities or relationships of the same type is called the entity set or relationship set.
- Another essential element of the E-R diagram is the mapping cardinalities, which express the number of entities to which another entity can be associated via a relationship set.

THE THREE MAIN COMPONENTS OF ER DIAGRAM ARE

The entity is a person, object, place or event for which data is collected. For example, if we consider the information system for a business, entities would include not only customers, but the customer's address, and orders as well. The entity is represented by a rectangle and labeled with a singular noun.

The relationship is the interaction between the entities. In the example above, the customer places an order, so the word "places" defines the relationship between that instance of a customer and the order or orders that they place. A relationship may be represented by a diamond shape, or more simply, by the line connecting the entities. In either case, verbs are used to label the relationships.

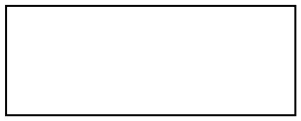
The cardinality defines the relationship between the entities in terms of numbers. An entity may be optional: for example, a sales representative could have no customers or could have one or many customers; or mandatory: for example, there must be at least one product listed in an order. There are several different types of cardinality notations; crow's foot notation, used here, is a common one. In crow's foot notation, a single bar indicates one, a double bar indicates one and only one (for example, a single instance of a product can only be stored in one warehouse), a circle indicates zero, and a crow's foot indicates many. The three main cardinal relationships are: one-to-one, expressed as 1:1; one-to-many, expressed as 1: M; and many-to-many, expressed as M: N.

ENTITY RELATIONSHIP DIAGRAM NOTATIONS

Peter Chen developed ER Diagram's in 1976. Since then Charles Bachman and James Martin have added some sleigh refinements to the basic ERD principle.

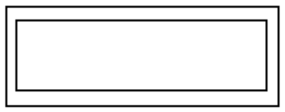
1.Entity:

An entity is an object or concept about which we want to store Information. An entity is a real-world item or concept that exists on its own. In an ER model, we diagram an entity type as a rectangle containing the type name.



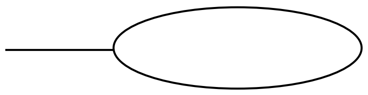
2.Weak Entity:

Alone a weak entity is an entity that must defined by a foreign key relationship with another entity as it cannot be uniquely identified by its own attributes.



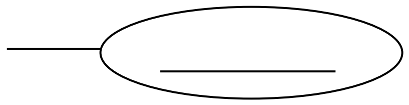
3.Attribute:

Each entity has attributes or particular properties that describe the entity. Most of the data in a database consists of values of attributes. The set of all possible values of an attribute is the attribute domain. In an ER model, an attribute name appears in an oval that has a line to the corresponding entity box.



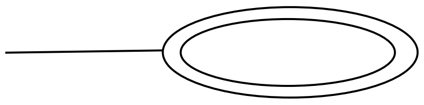
4.Key attributes:

A key attribute is the unique, distinguishing characteristic of the entity. An attribute or set of attributes that uniquely identifies a particular entity is a key. A key attribute in an ER Diagram is represented by an oval that has a line inside it and a line to the corresponding entity box. For example, an employee's social security number might be the employee's key attribute.



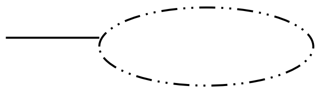
5.Multi-valued attribute:

A multi-valued attribute can have more than one value. We indicate this with a double oval. For example, an employee entity can have multiple skill values.



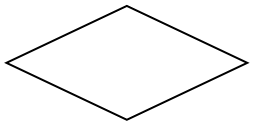
6. Derived attribute:

A derived attribute is based on another attribute. It is denoted by a oval and dotted line within it. For example, an employee's monthly salary is based on the employee's annual salary.



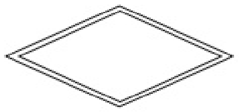
7.Relationships:

Relationships illustrate how two entities share information in the database structure. An association among entities is called a relationship. An attribute can also be a property of a relationship set. The association among the entities is described as one-to-one, one-to-many, many-to-many. A relationship is indicated by a rhombus.



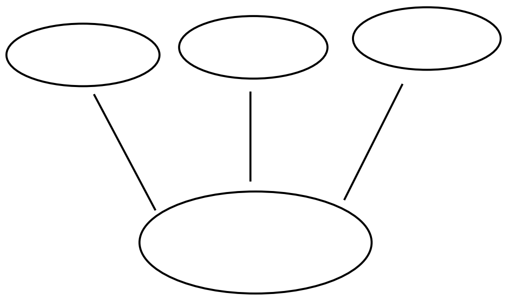
<u>IDENTIFYING RELATIONSHIP</u>

1.Identifying relationship is denoted by double rhombus.



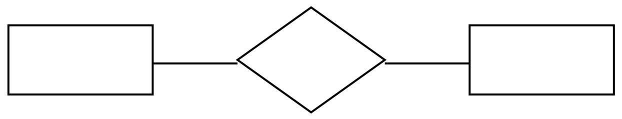
2.Composite attribute:

A composite attribute has multiple components and each component is atomic or composite. We illustrate this composite nature in the ER model branching of the component attributes.



3.Total Participation:

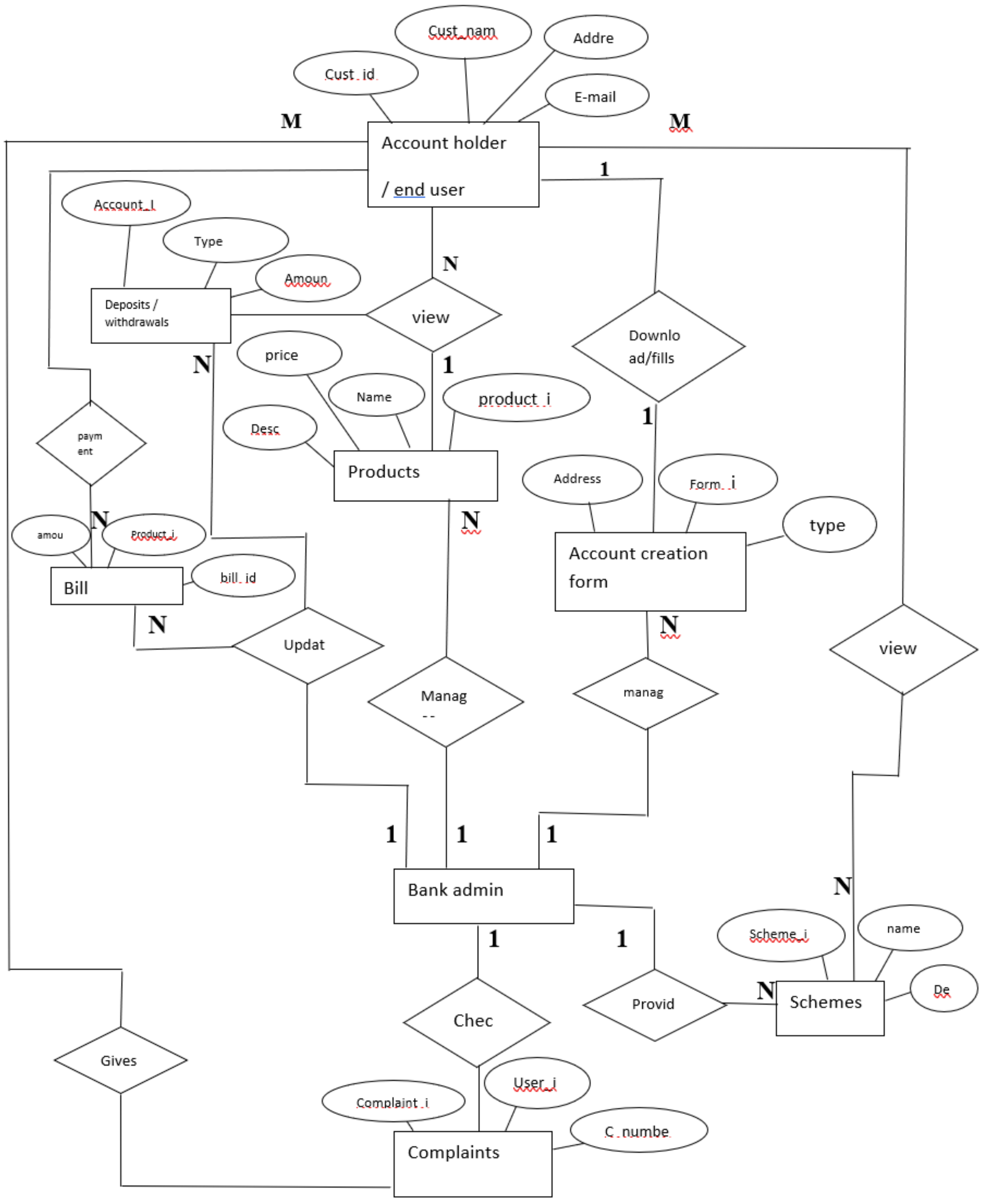
Total participation is represented by a double line.



4.Cardinality:

Cardinality specifies how many instances of an entity relate to one instance of another entity. Ordinarily is also closely linked to cardinality. While cardinality specifies the occurrences of a relationship, ordinarily describes the relationship as either mandatory or optional. In other words, cardinality specifies the maximum number of relationships and ordinarily specifies the absolute minimum number of relationships.

ENTITY RELATIONSHIP MODEL



USE CASES DESCRIPTION

What is a Use Case Diagram?

A Use Case Diagram is part of the **Unified Modeling Language (UML)** that visually represents the functional requirements of a system. It shows how different types of users (actors) interact with the system and what services or functionalities (use cases) the system offers.

Use case diagrams are simple but powerful tools that help developers, analysts, and stakeholders understand the scope of the system, the user interactions, and the main functionalities at a glance.

Purpose of Use Case Diagram in CSMS

For the **Co-Operative Society Management System**, the use case diagram:

- Identifies all the actors (Admin, Employee, Member).
- Shows the operations each actor can perform.
- Defines the system boundaries.
- Helps ensure all functional requirements are covered.

Actors in the System

1. **Admin**
    - Manages the entire system.
    - Has full access to employee, member, transaction, and product management.
2. **Member**
    - Requests deposits and withdrawals.

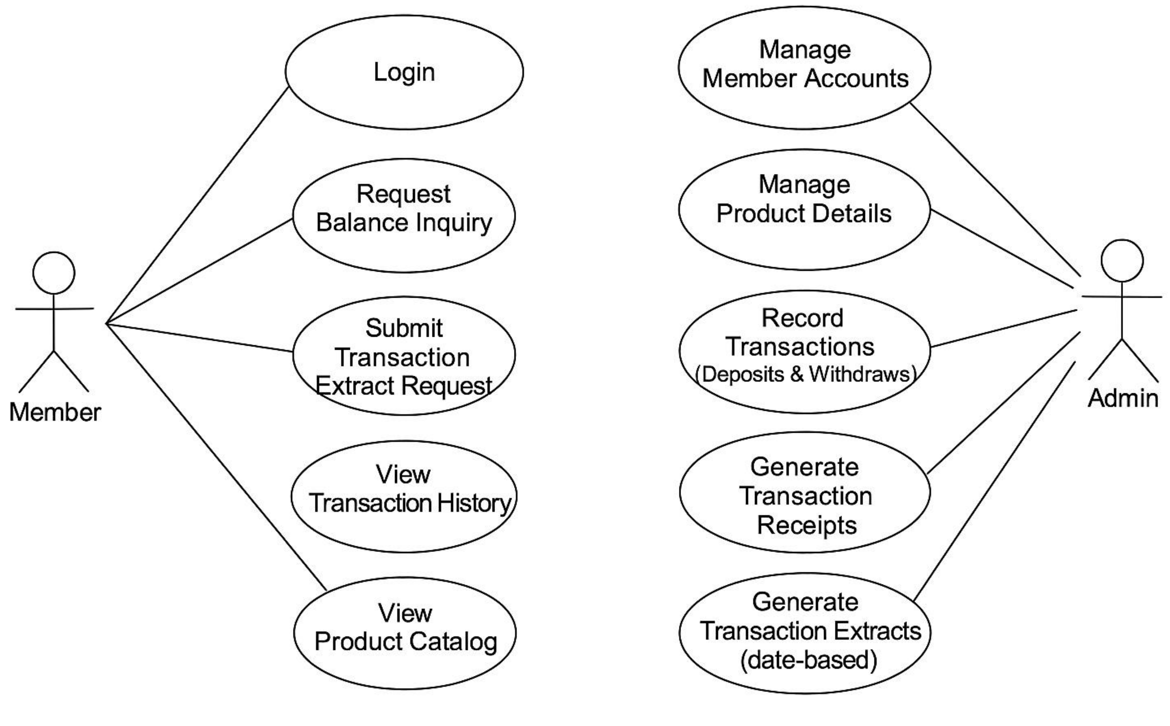
- Checks account balance.

- Requests transaction history or extract.

Importance of Use Case Diagram

- Provides a **clear picture** of system functionality.
- Helps in understanding the **user roles and permissions**.
- Serves as a **foundation for system design and testing**.
- Aids communication between technical and non-technical stakeholders.

USE CASE DIAGRAM


Member
Login
Request Balance Inquiry
Submit Transaction Extract Request
View Transaction History
View Product Catalog
Manage Member Accounts
Manage Product Details
Record Transactions (Deposits & Withdraws)
Generate Transaction Receipts
Generate Transaction Extracts (date-based)
Admin

Use Case Diagram

5.SYSTEM DEVELOPMENT

System development is the process of converting the system design into a functional software application. It involves coding, testing, and integrating various modules to ensure the system meets the specified requirements. The **Co-Operative Society Management System** was developed using a structured, modular approach for better clarity, maintenance, and scalability.

The development phase focuses on how, that is, during development a software engineer attempts to define how data are to be structured, how function is to be implemented within a software architecture, how procedural details are to be implemented, how interfaces are to be characterized and how testing will be performed.

1. Development Approach

The development followed the **Waterfall Model**, where each phase (Requirement Analysis, Design, Implementation, Testing, and Deployment) was completed in sequence. This model was ideal due to the well-defined and fixed set of requirements for the system.

2. Tools and Technologies Used

- **Frontend**: HTML, CSS, JavaScript (for forms and user interaction)
- **Backend**: PHP (to handle business logic and database operations)
- **Database**: MySQL (to store member records, transactions, product details)
- **Server**: WAMP/XAMPP (used for local development and testing)
- **Platform**: Windows OS (compatible with WAMP stack)

3. Development Steps

1. **Database Creation**
    - Tables for members, transactions, and products were created using MySQL.
    - Relationships were defined using primary and foreign keys.
2. **Frontend Design**
    - Web pages were designed with HTML and styled using CSS.
    - Forms were created for login, member entry, product entry, and transaction entry.
3. **Backend Development**
    - PHP scripts were written to handle logic such as user login, adding members, processing deposits/withdrawals, and generating reports.
    - Data validation was included to ensure input correctness.
4. **Module Integration**
    - Each functional module (e.g., Member Management, Product Management, Transactions) was tested and integrated into a unified system.
5. **Testing and Debugging**
    - Functional testing ensured each feature worked as intended.
    - Sample data was entered to verify transactions, balances, and product updates.
    - Bugs and errors were identified and corrected.

4. Outcome

The final system is a web-based application that allows the admin to:

- Manage member data

- Record and track transactions
- Handle balance requests and extract reports
- Manage product inventory

It offers a simple interface and operates smoothly on standard local server environments.

6.SYSTEM IMPLEMENTATION

System implementation is the process of developing, testing, and deploying the software based on design specifications. For the Co-Operative Society Management System, it involves setting up the environment, creating modules for member and transaction management, integrating the database, and ensuring the system runs smoothly for real-time use.

1. Objective of Implementation

To deploy the system in a real-time environment where administrators can perform operations such as managing members, processing deposits and withdrawals, generating transaction extracts, and handling product inventory effectively.

2. Implementation Activities

a. Setting Up the Environment:

- Installed **XAMPP** (Apache, MySQL, PHP) on the development system.
- Deployed the system on a **local server** (WAMP/XAMPP) supporting PHP and MySQL.
- Imported the **MySQL database** using **phpMyAdmin**.
- Configured database connection credentials in PHP files securely.

b. Database Integration:

- Created necessary tables: admin, members, transactions, products.
- Linked tables using **primary and foreign keys** to maintain data integrity.
- Inserted **sample records** for initial testing of members, transactions, and products.

c. Frontend Deployment:

- Built the interface using **HTML**, **CSS**, and **JavaScript**.
- Tested the UI for proper layout and responsiveness on desktop browsers.
- Connected forms (login, deposit, withdrawal, balance check, product entry) to backend logic via PHP.

d. Backend Configuration:

- Exported the database from the development environment and imported it into the server.
- Set up **PHP scripts** to manage core functions like login, member registration, transactions, and inventory.
- Applied **form validations**, input sanitization, and basic **security measures**.

3. User Role Setup

Admin:

- Can log in securely.
- Manage member records (add, update, delete).
- Record deposits and withdrawals.
- Check account balances and generate transaction extracts.
- Manage product inventory (add, update, remove).

4. User Training

Admin users were trained on:

- How to add and manage member profiles.
- Performing deposits and withdrawals accurately.
- Viewing account balances and generating reports.
- Managing product stock and updating inventory.
- A **user manual with screenshots** was created for easy reference.

5. Post-Implementation Testing

- Tested **login/logout**, deposit and withdrawal flows, member search, and transaction reporting.
- Ensured account balances updated correctly after each transaction.
- Verified product updates reflected accurately in the product catalog.
- Fixed minor layout and logic issues found during live usage.
- Included a **basic feedback form** to collect admin suggestions.

6. Security Measures Implemented

- Passwords were stored using **secure hashing** methods.
- Admin access was protected through **session handling**.
- Unauthorized access was restricted through session control mechanisms.

7. Testing in Live Environment

After deployment, the following were tested:

- Admin login/logout and session control
- Adding and updating member details
- Deposit and withdrawal processes
- Transaction extract generation
- Product inventory updates
- Data storage and retrieval accuracy

All identified issues were resolved before the final system handover.

8. Maintenance and Support

- Regular backups scheduled for the **MySQL database** and application files.
- System logs monitored for unusual activity or errors.
- Support provided through email or local access in case of technical issues.

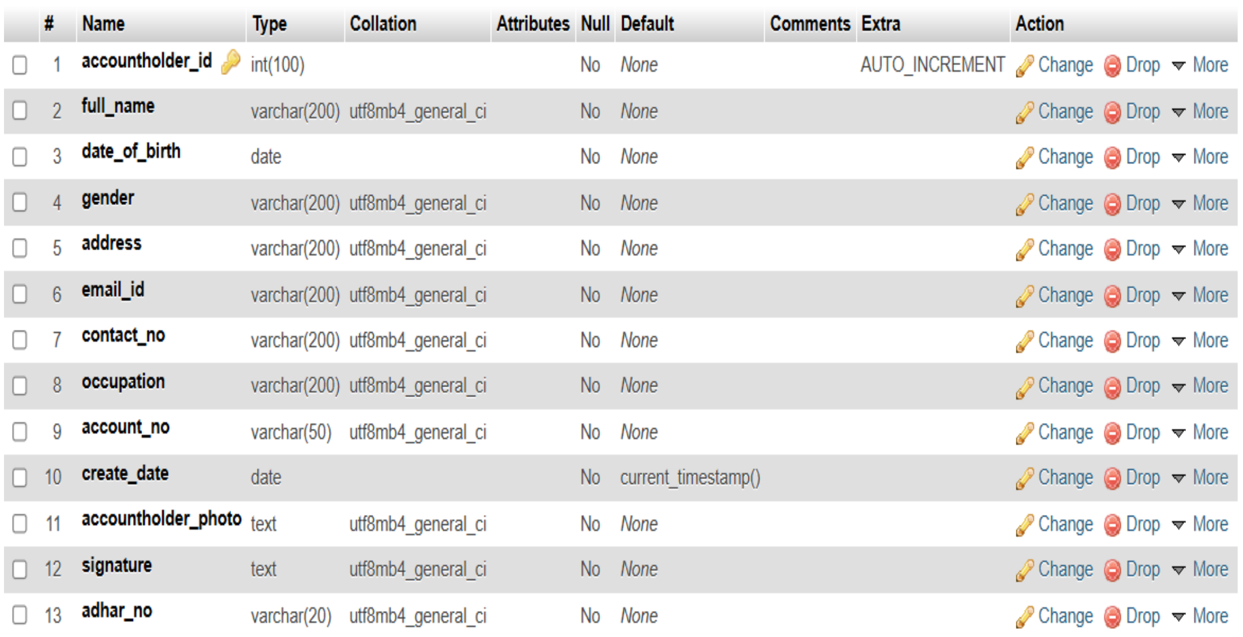
The implementation of the **Co-Operative Society Management System** was successful. It automated manual operations, improved member and transaction handling, and provided a more organized and efficient system for cooperative management.

The system is now fully functional and ready for future enhancements based on real-time usage feedback. With its deployment, the society has taken a significant step toward digital transformation, ensuring transparency, accuracy, and faster service delivery.

Continuous monitoring and user feedback will guide upcoming updates, making the system even more adaptive to the evolving needs of the cooperative and its members.

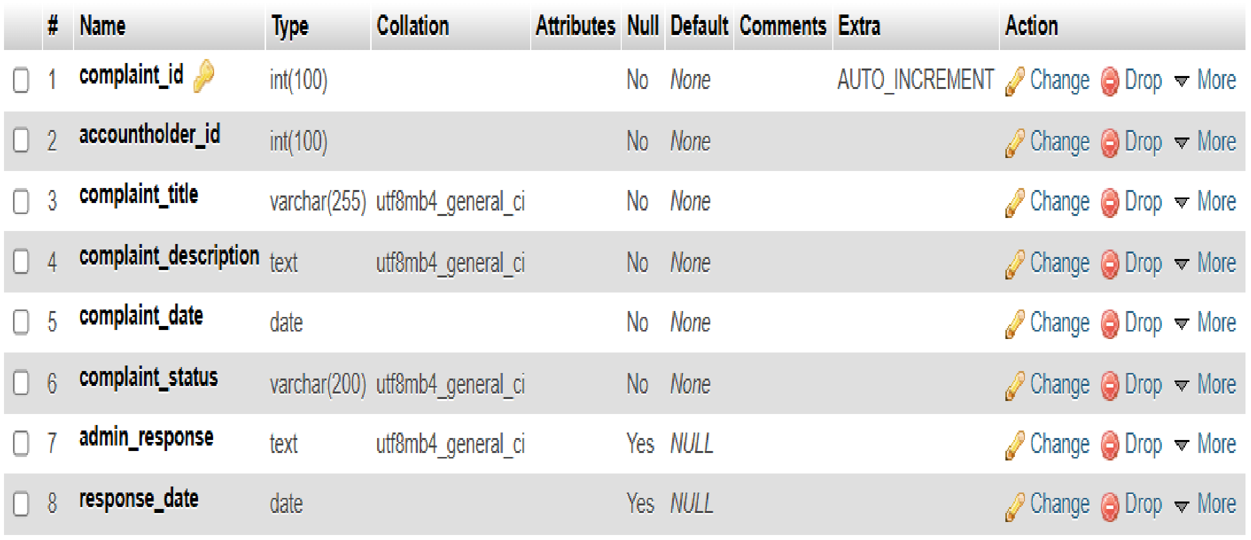
TABLES USED IN THE DATABASE

1. Account Holder Table



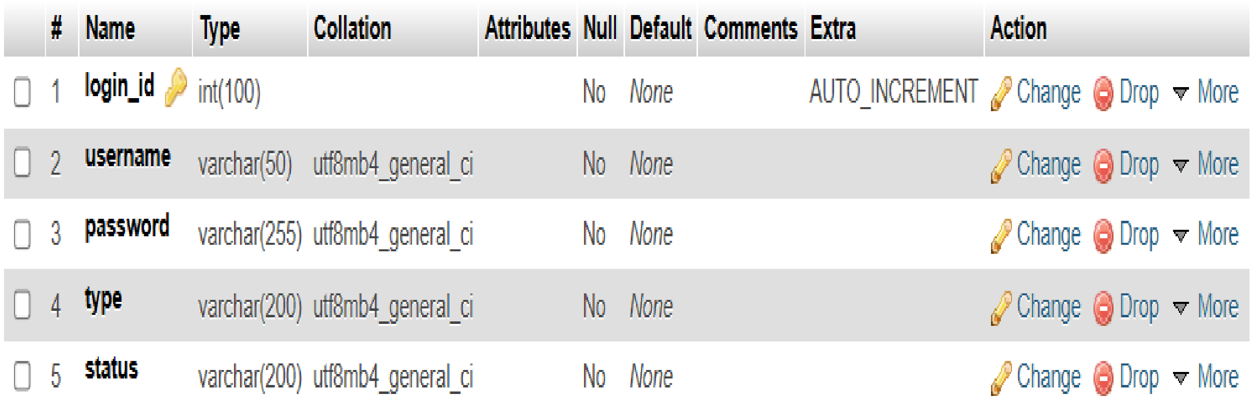
#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	accountholder_id	int(100)			No	*None*		AUTO_INCREMENT	Change Drop More
2	full_name	varchar(200)	utf8mb4_general_ci		No	*None*			Change Drop More
3	date_of_birth	date			No	*None*			Change Drop More
4	gender	varchar(200)	utf8mb4_general_ci		No	*None*			Change Drop More
5	address	varchar(200)	utf8mb4_general_ci		No	*None*			Change Drop More
6	email_id	varchar(200)	utf8mb4_general_ci		No	*None*			Change Drop More
7	contact_no	varchar(200)	utf8mb4_general_ci		No	*None*			Change Drop More
8	occupation	varchar(200)	utf8mb4_general_ci		No	*None*			Change Drop More
9	account_no	varchar(50)	utf8mb4_general_ci		No	*None*			Change Drop More
10	create_date	date			No	current_timestamp()			Change Drop More
11	accountholder_photo	text	utf8mb4_general_ci		No	*None*			Change Drop More
12	signature	text	utf8mb4_general_ci		No	*None*			Change Drop More
13	adhar_no	varchar(20)	utf8mb4_general_ci		No	*None*			Change Drop More

2. Complaints Table



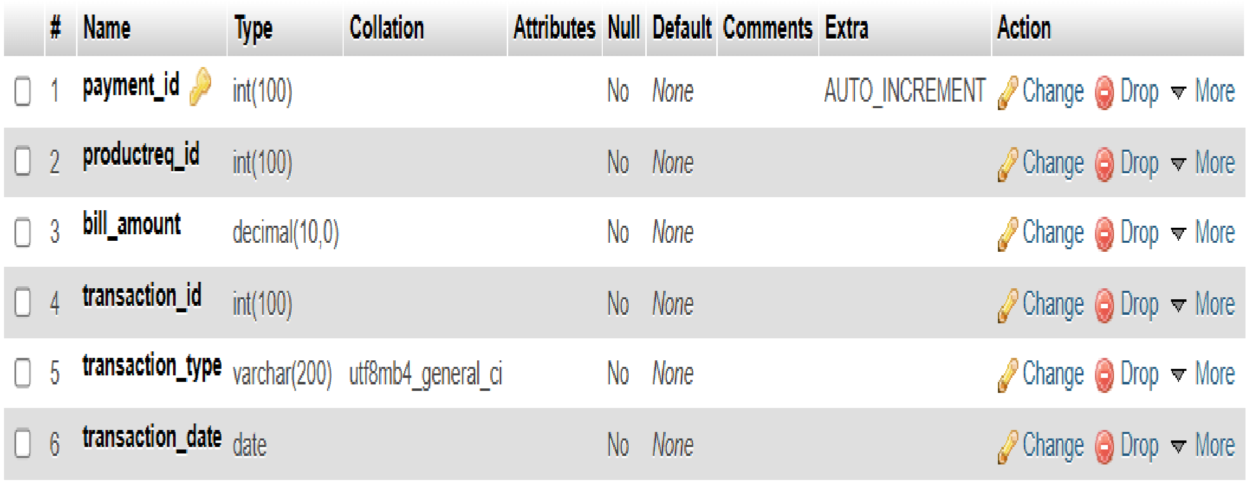
#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	complaint_id	int(100)			No	*None*		AUTO_INCREMENT	Change Drop More
2	accountholder_id	int(100)			No	*None*			Change Drop More
3	complaint_title	varchar(255)	utf8mb4_general_ci		No	*None*			Change Drop More
4	complaint_description	text	utf8mb4_general_ci		No	*None*			Change Drop More
5	complaint_date	date			No	*None*			Change Drop More
6	complaint_status	varchar(200)	utf8mb4_general_ci		No	*None*			Change Drop More
7	admin_response	text	utf8mb4_general_ci		Yes	*NULL*			Change Drop More
8	response_date	date			Yes	*NULL*			Change Drop More

3.Login Table



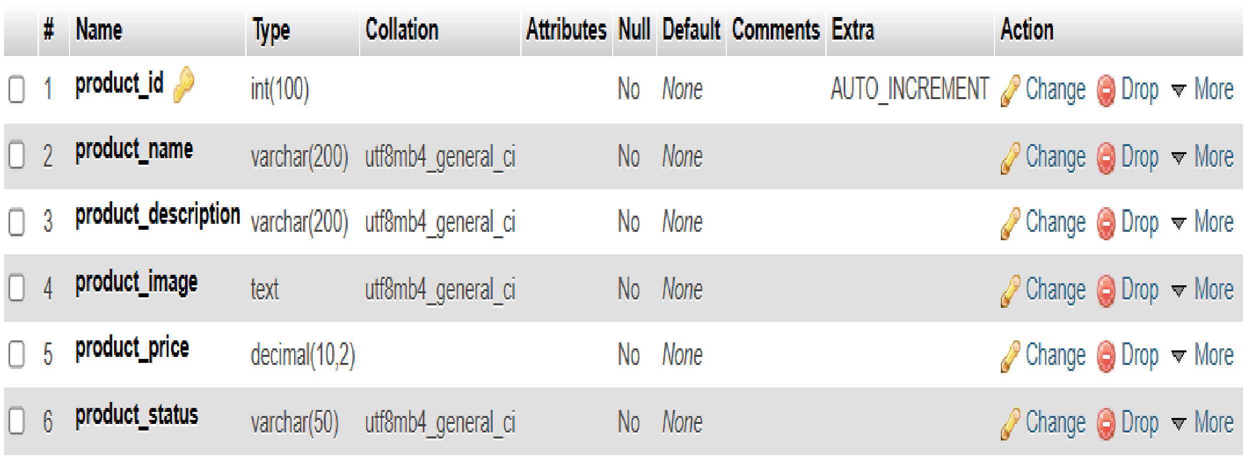
#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	login_id	int(100)			No	*None*		AUTO_INCREMENT	Change Drop More
2	username	varchar(50)	utf8mb4_general_ci		No	*None*			Change Drop More
3	password	varchar(255)	utf8mb4_general_ci		No	*None*			Change Drop More
4	type	varchar(200)	utf8mb4_general_ci		No	*None*			Change Drop More
5	status	varchar(200)	utf8mb4_general_ci		No	*None*			Change Drop More

4. Payments Table



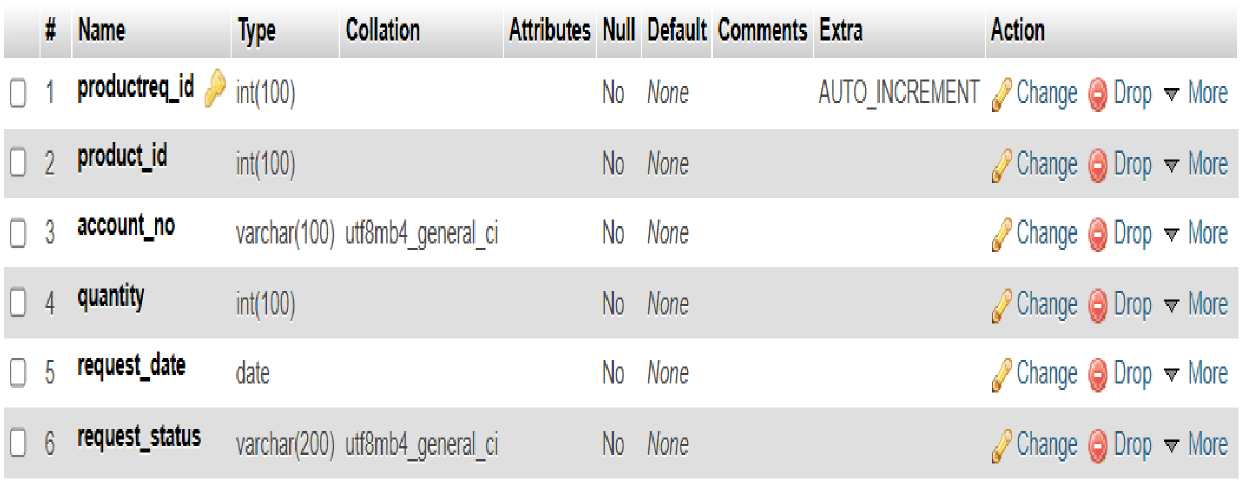
#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	payment_id	int(100)			No	*None*		AUTO_INCREMENT	Change Drop More
2	productreq_id	int(100)			No	*None*			Change Drop More
3	bill_amount	decimal(10,0)			No	*None*			Change Drop More
4	transaction_id	int(100)			No	*None*			Change Drop More
5	transaction_type	varchar(200)	utf8mb4_general_ci		No	*None*			Change Drop More
6	transaction_date	date			No	*None*			Change Drop More

5. Products Table



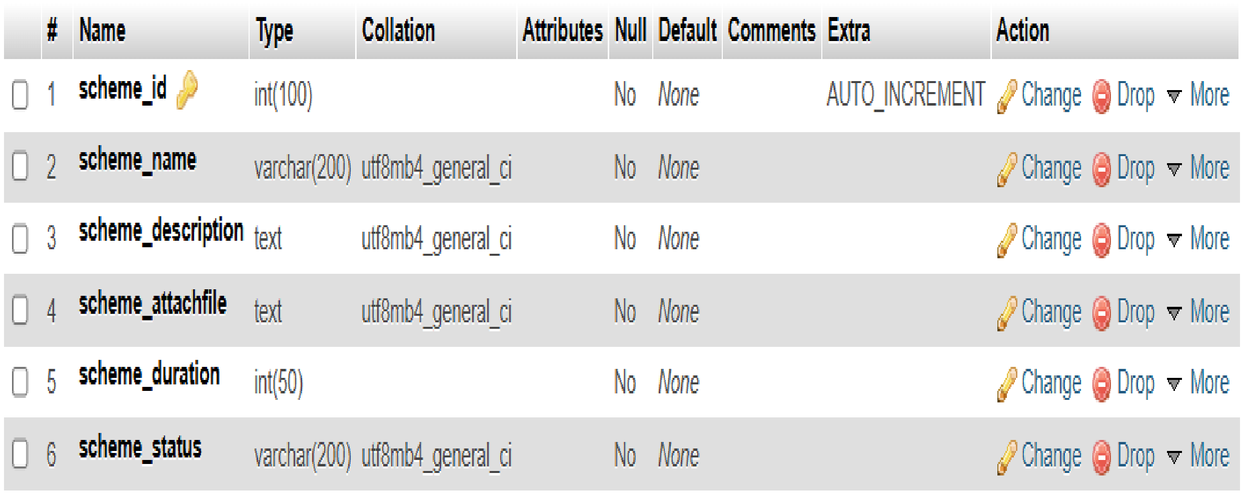
#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	product_id	int(100)			No	*None*		AUTO_INCREMENT	Change Drop More
2	product_name	varchar(200)	utf8mb4_general_ci		No	*None*			Change Drop More
3	product_description	varchar(200)	utf8mb4_general_ci		No	*None*			Change Drop More
4	product_image	text	utf8mb4_general_ci		No	*None*			Change Drop More
5	product_price	decimal(10,2)			No	*None*			Change Drop More
6	product_status	varchar(50)	utf8mb4_general_ci		No	*None*			Change Drop More

6. Product Request Table



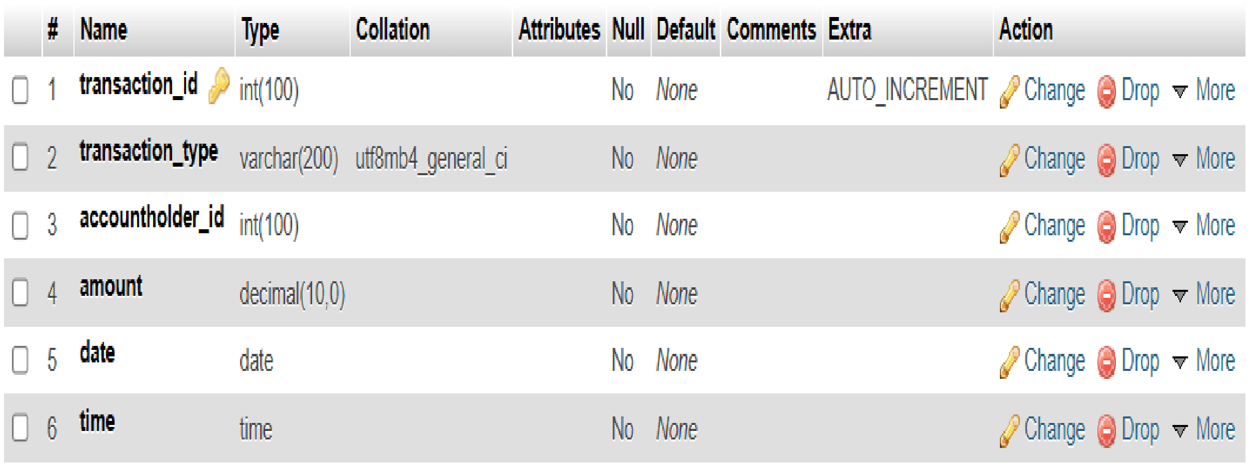
#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	productreq_id	int(100)			No	*None*		AUTO_INCREMENT	Change Drop More
2	product_id	int(100)			No	*None*			Change Drop More
3	account_no	varchar(100)	utf8mb4_general_ci		No	*None*			Change Drop More
4	quantity	int(100)			No	*None*			Change Drop More
5	request_date	date			No	*None*			Change Drop More
6	request_status	varchar(200)	utf8mb4_general_ci		No	*None*			Change Drop More

7. Schemes Table



	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
☐	1	scheme_id	int(100)			No	None		AUTO_INCREMENT	Change Drop More
☐	2	scheme_name	varchar(200)	utf8mb4_general_ci		No	None			Change Drop More
☐	3	scheme_description	text	utf8mb4_general_ci		No	None			Change Drop More
☐	4	scheme_attachfile	text	utf8mb4_general_ci		No	None			Change Drop More
☐	5	scheme_duration	int(50)			No	None			Change Drop More
☐	6	scheme_status	varchar(200)	utf8mb4_general_ci		No	None			Change Drop More

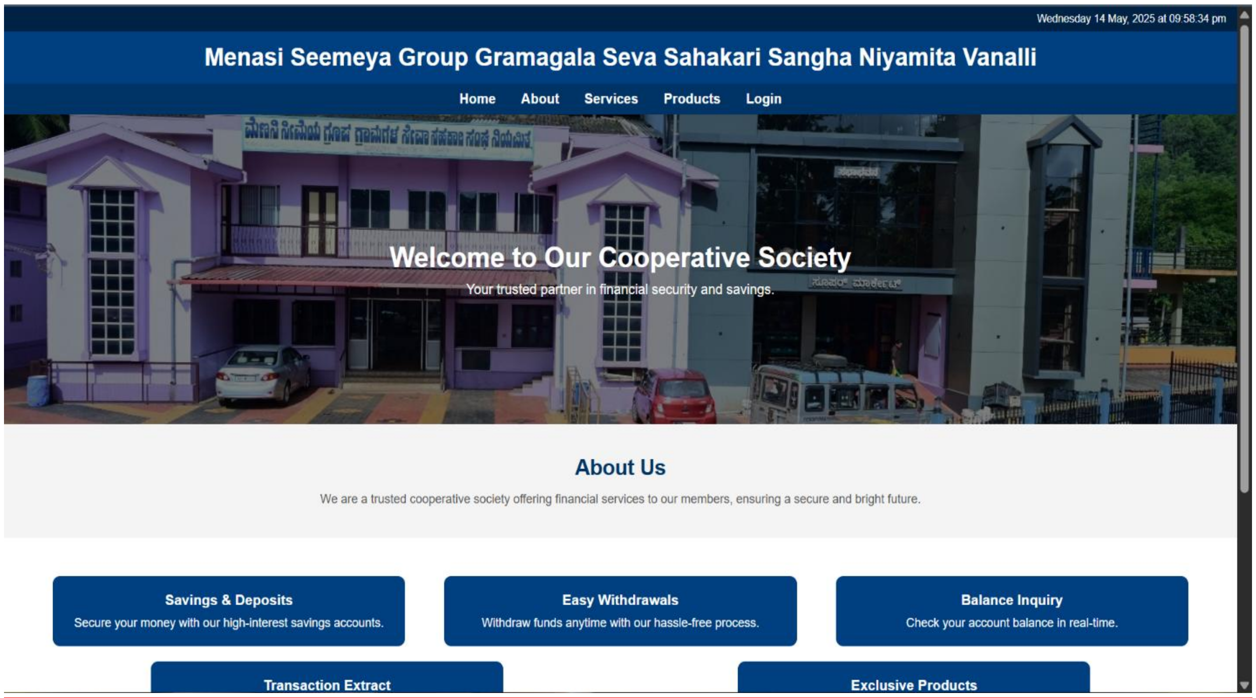
7. Transaction Table



	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
☐	1	transaction_id	int(100)			No	None		AUTO_INCREMENT	Change Drop More
☐	2	transaction_type	varchar(200)	utf8mb4_general_ci		No	None			Change Drop More
☐	3	accountholder_id	int(100)			No	None			Change Drop More
☐	4	amount	decimal(10,0)			No	None			Change Drop More
☐	5	date	date			No	None			Change Drop More
☐	6	time	time			No	None			Change Drop More

7.SCREEN SHOTS

HOME PAGE


Wednesday 14 May, 2025 at 09:58:34 pm

Menasi Seemeya Group Gramagala Seva Sahakari Sangha Niyamita Vanalli

Welcome to Our Cooperative Society

Your trusted partner in financial security and savings.

About Us

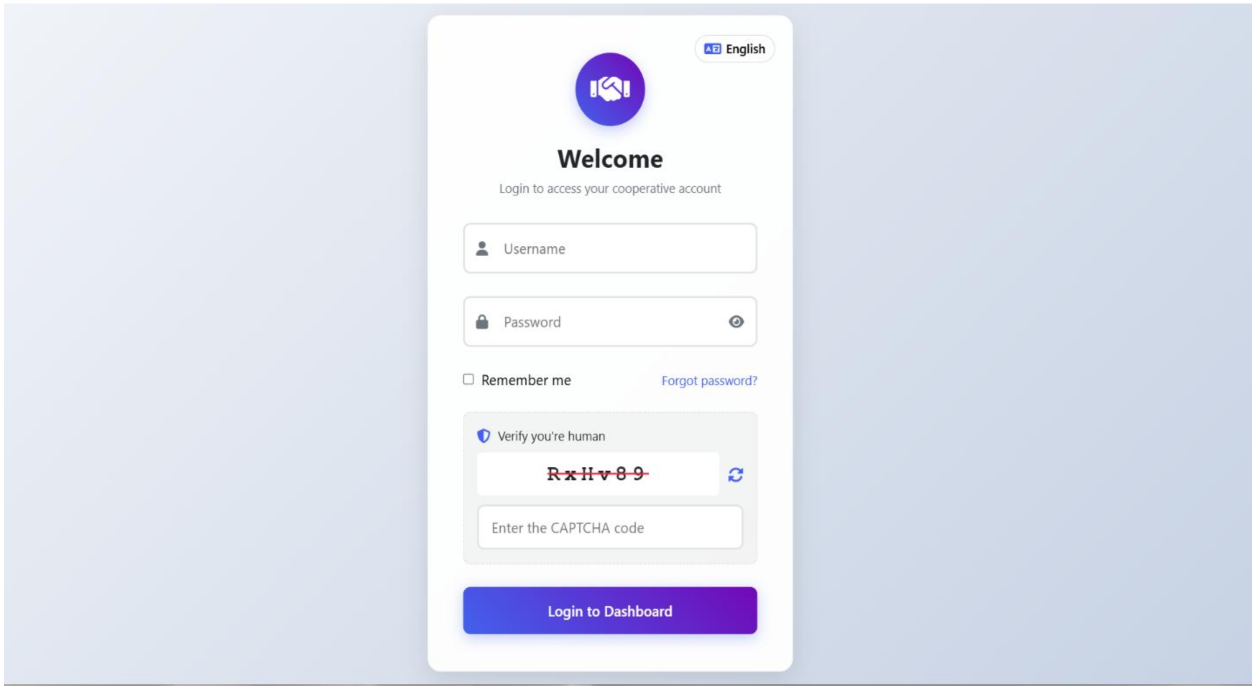
We are a trusted cooperative society offering financial services to our members, ensuring a secure and bright future.

Savings & Deposits
Secure your money with our high-interest savings accounts.

Easy Withdrawals
Withdraw funds anytime with our hassle-free process.

Balance Inquiry
Check your account balance in real-time.

LOGIN PAGE

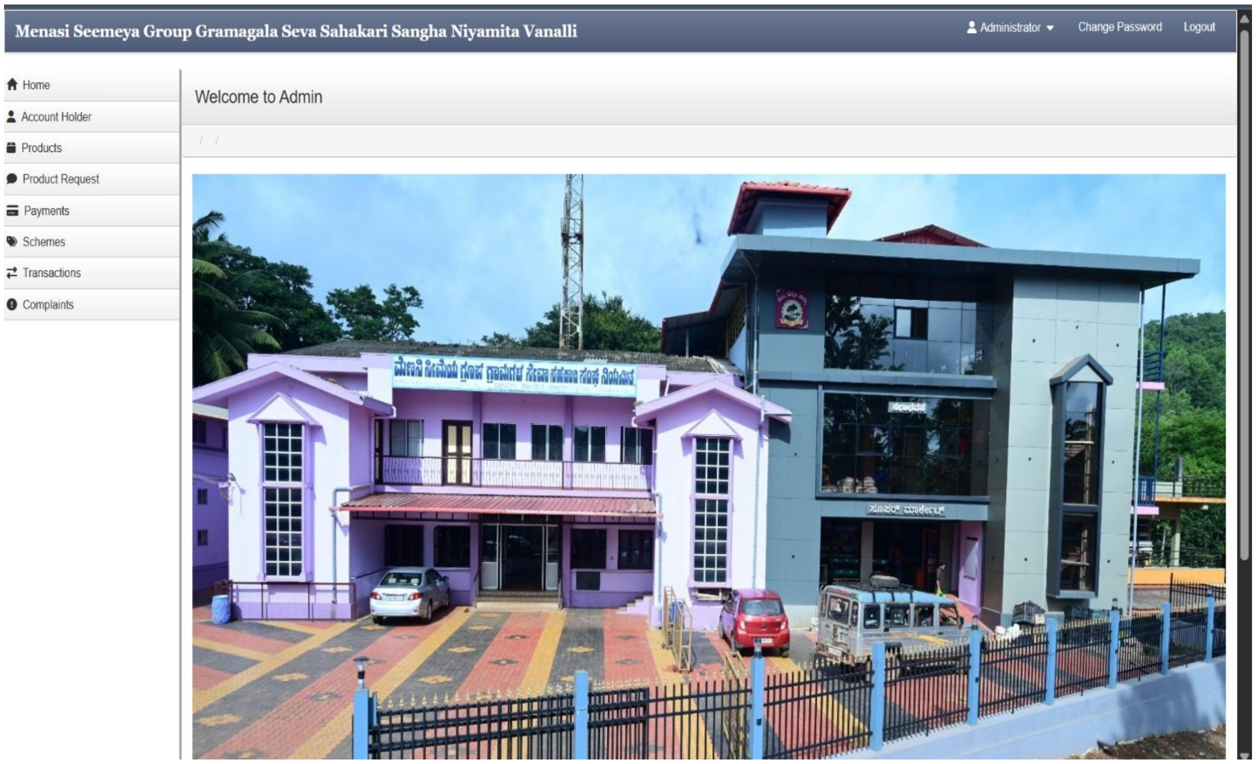


Welcome

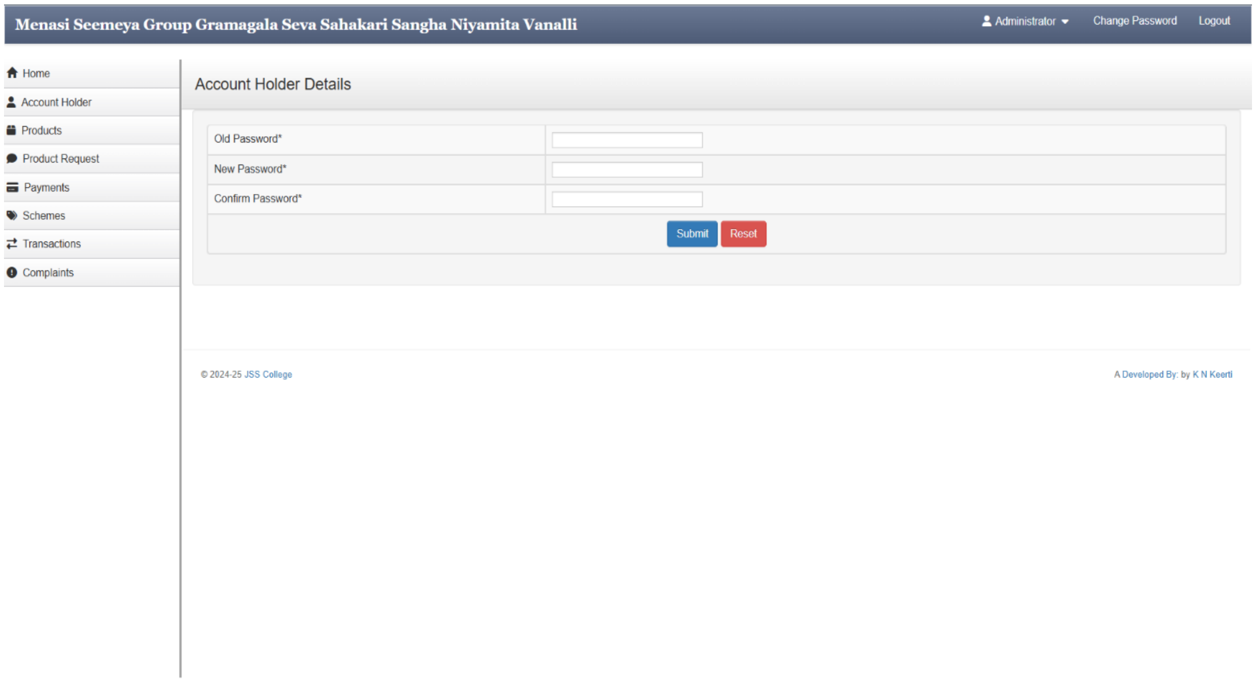
Login to access your cooperative account

Verify you're human

ADMIN DASHBOARD



CHANGE PASSWORD

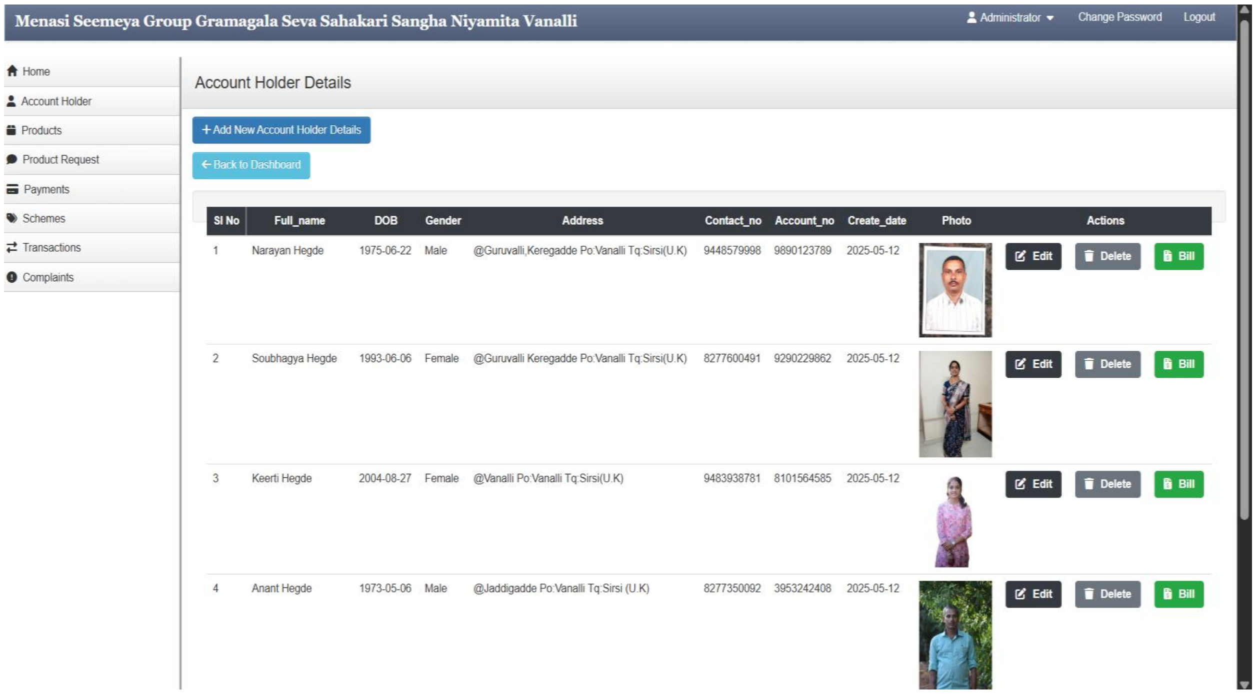


Old Password*	
New Password*	
Confirm Password*	

© 2024-25 JSS College

A Developed By: by K N Keerti

ACCOUNT HOLDER INFORMATION

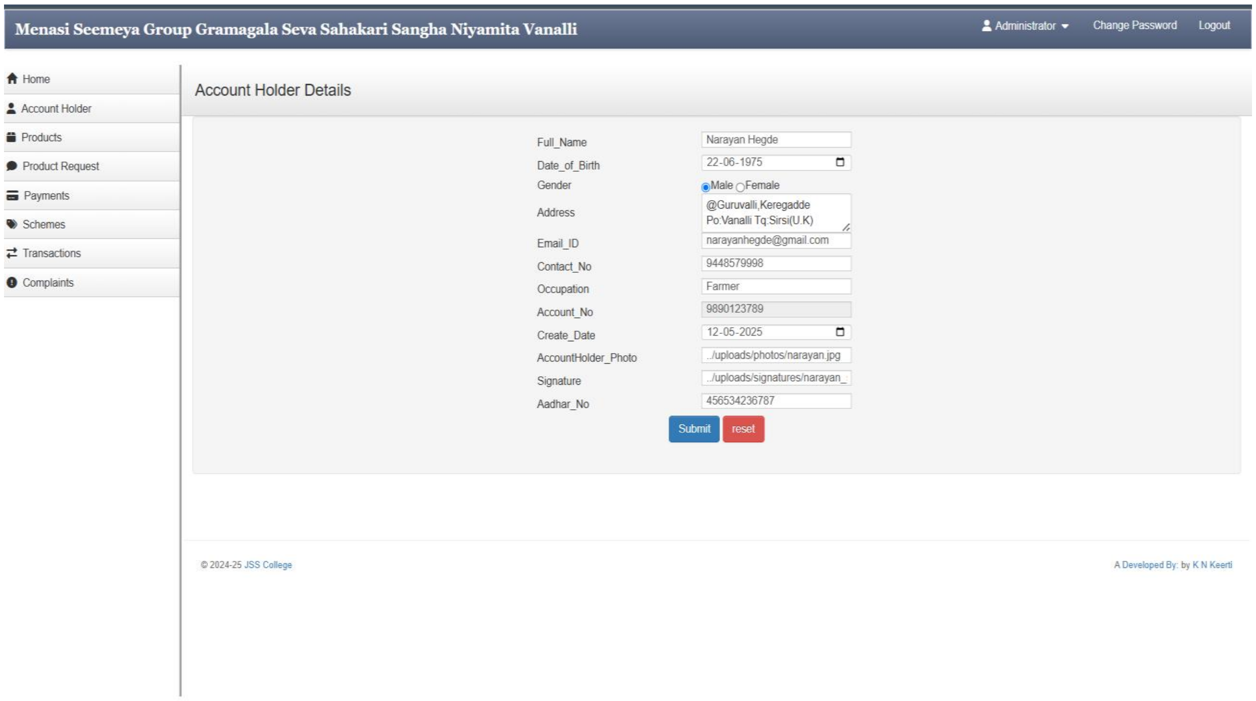


Menasi Seemeya Group Gramagala Seva Sahakari Sangha Niyamita Vanalli

Account Holder Details

Sl No	Full_name	DOB	Gender	Address	Contact_no	Account_no	Create_date	Photo	Actions
1	Narayan Hegde	1975-06-22	Male	@Guruvalli,Keregadde Po:Vanalli Tq:Sirsi(U.K)	9448579998	9890123789	2025-05-12		Edit Delete Bill
2	Soubhagya Hegde	1993-06-06	Female	@Guruvalli Keregadde Po:Vanalli Tq:Sirsi(U.K)	8277600491	9290229862	2025-05-12		Edit Delete Bill
3	Keerti Hegde	2004-08-27	Female	@Vanalli Po:Vanalli Tq:Sirsi(U.K)	9483938781	8101564585	2025-05-12		Edit Delete Bill
4	Anant Hegde	1973-05-06	Male	@Jaddigadde Po:Vanalli Tq:Sirsi (U.K)	8277350092	3953242408	2025-05-12		Edit Delete Bill

ACCOUNT HOLDER EDIT FORM



Menasi Seemeya Group Gramagala Seva Sahakari Sangha Niyamita Vanalli

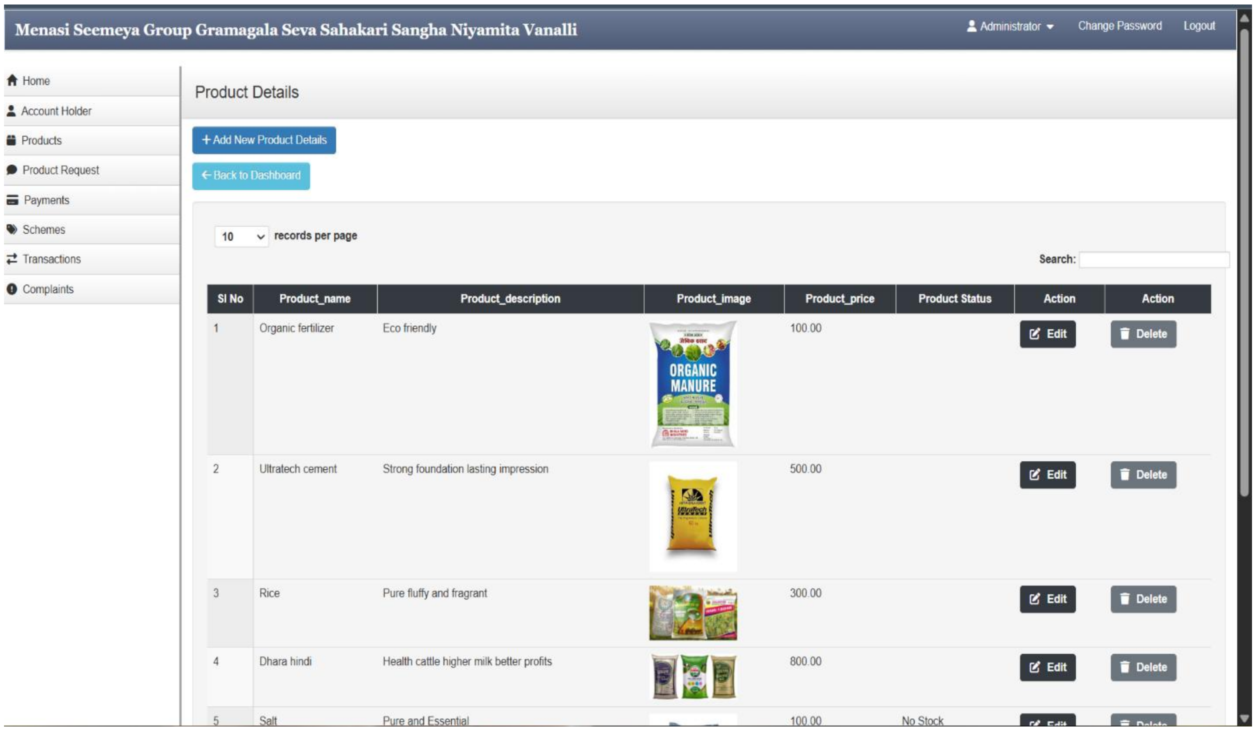
Account Holder Details

Full_Name	Narayan Hegde
Date_of_Birth	22-06-1975
Gender	[✓ Male] Female
Address	@Guruvalli,Keregadde Po:Vanalli Tq:Sirsi(U.K)
Email_ID	narayanhegde@gmail.com
Contact_No	9448579998
Occupation	Farmer
Account_No	9890123789
Create_Date	12-05-2025
AccountHolder_Photo	../uploads/photos/narayan.jpg
Signature	../uploads/signatures/narayan_
Aadhar_No	456534236787

© 2024-25 JSS College

A Developed By: by K N Keerti

PRODUCT INFORMATION

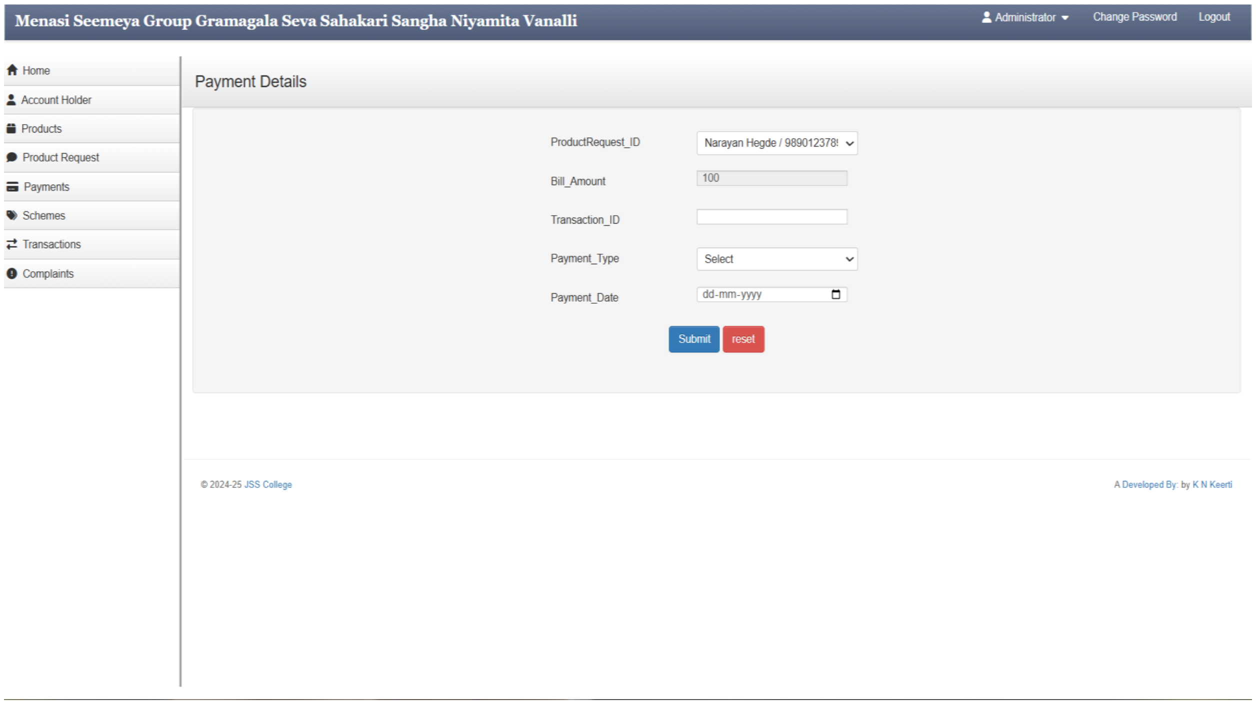


Menasi Seemeya Group Gramagala Seva Sahakari Sangha Niyamita Vanalli

Product Details

Sl No	Product_name	Product_description	Product_image	Product_price	Product Status	Action	Action
1	Organic fertilizer	Eco friendly		100.00		Edit	Delete
2	Ultratech cement	Strong foundation lasting impression		500.00		Edit	Delete
3	Rice	Pure fluffy and fragrant		300.00		Edit	Delete
4	Dhara hindi	Health cattle higher milk better profits		800.00		Edit	Delete
5	Salt	Pure and Essential		100.00	No Stock		

PAYMENT DETAILS FORM

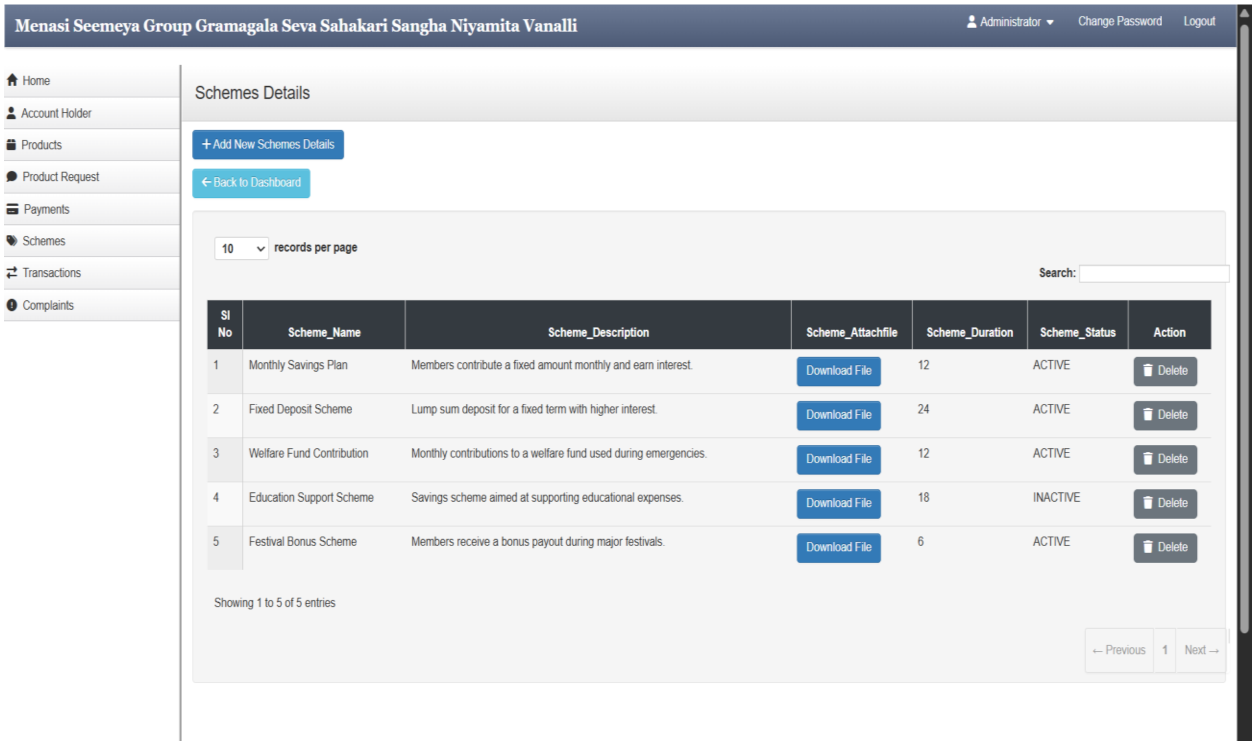


Menasi Seemeya Group Gramagala Seva Sahakari Sangha Niyamita Vanalli

Payment Details

ProductRequest_ID	Narayan Hegde / 989012378
Bill_Amount	100
Transaction_ID	
Payment_Type	Select
Payment_Date	dd-mm-yyyy

SCHEMES DETAILS



Menasi Seemeya Group Gramagala Seva Sahakari Sangha Niyamita Vanalli

Schemes Details

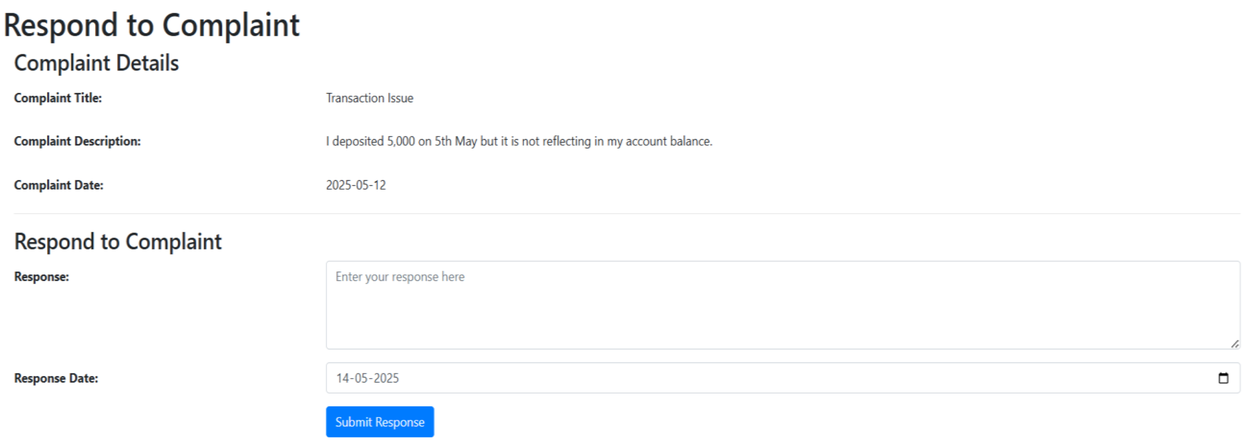
10 records per page

Search:

Sl No	Scheme_Name	Scheme_Description	Scheme_Attachfile	Scheme_Duration	Scheme_Status	Action
1	Monthly Savings Plan	Members contribute a fixed amount monthly and earn interest.	Download File	12	ACTIVE	Delete
2	Fixed Deposit Scheme	Lump sum deposit for a fixed term with higher interest.	Download File	24	ACTIVE	Delete
3	Welfare Fund Contribution	Monthly contributions to a welfare fund used during emergencies.	Download File	12	ACTIVE	Delete
4	Education Support Scheme	Savings scheme aimed at supporting educational expenses.	Download File	18	INACTIVE	Delete
5	Festival Bonus Scheme	Members receive a bonus payout during major festivals.	Download File	6	ACTIVE	Delete

Showing 1 to 5 of 5 entries

COMPLAINT RESPOND FORM



Respond to Complaint

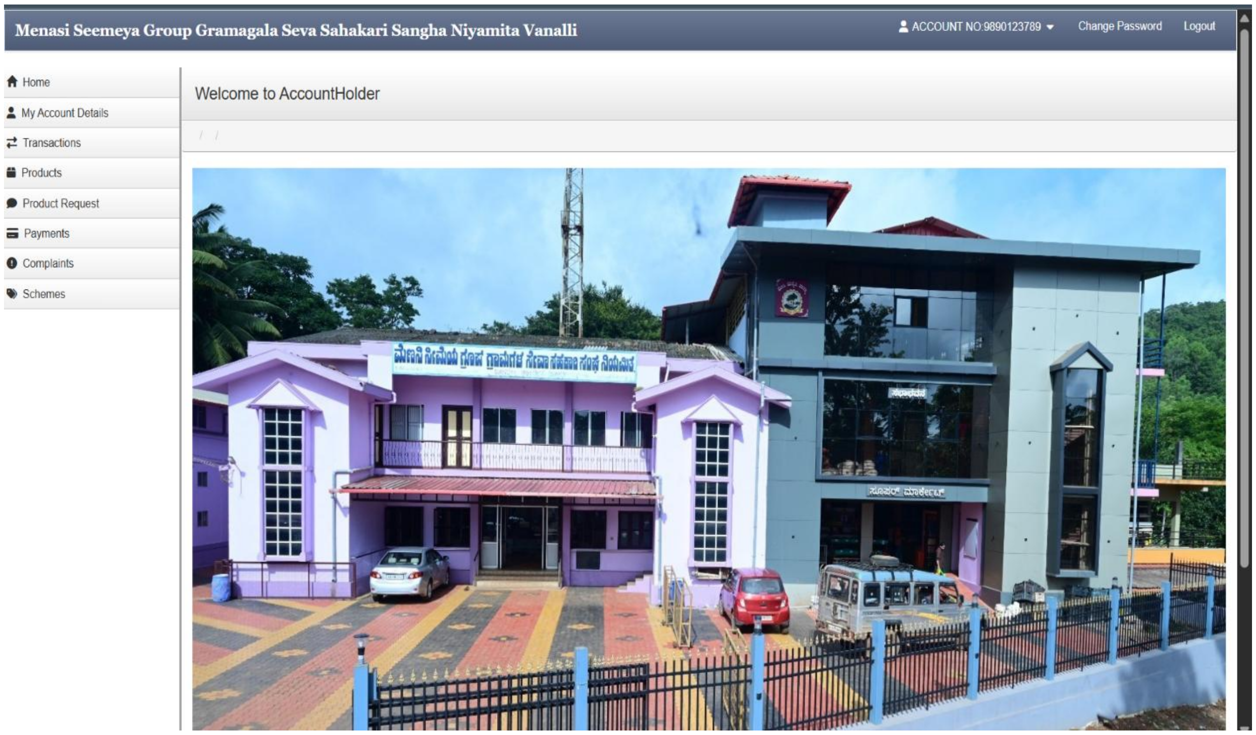
Complaint Details

Complaint Title:	Transaction Issue
Complaint Description:	I deposited 5,000 on 5th May but it is not reflecting in my account balance.
Complaint Date:	2025-05-12

Respond to Complaint

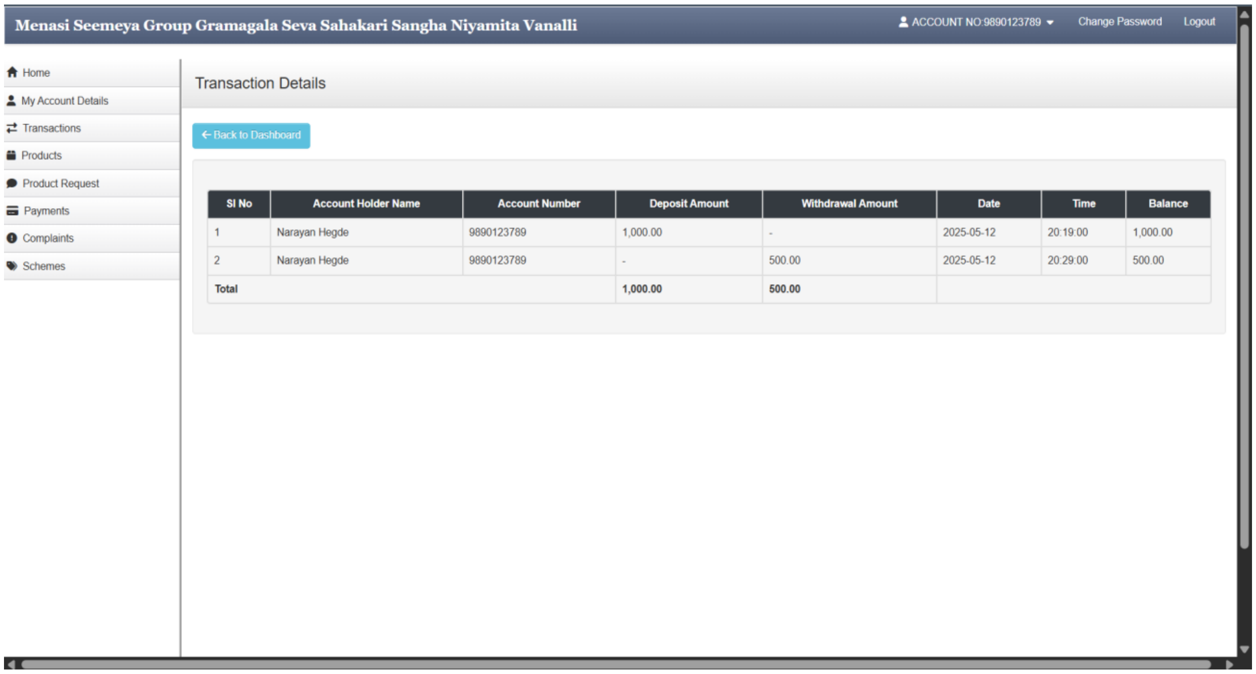
Response:	Enter your response here
Response Date:	14-05-2025

CUSTOMER DASHBOARD



Welcome to AccountHolder

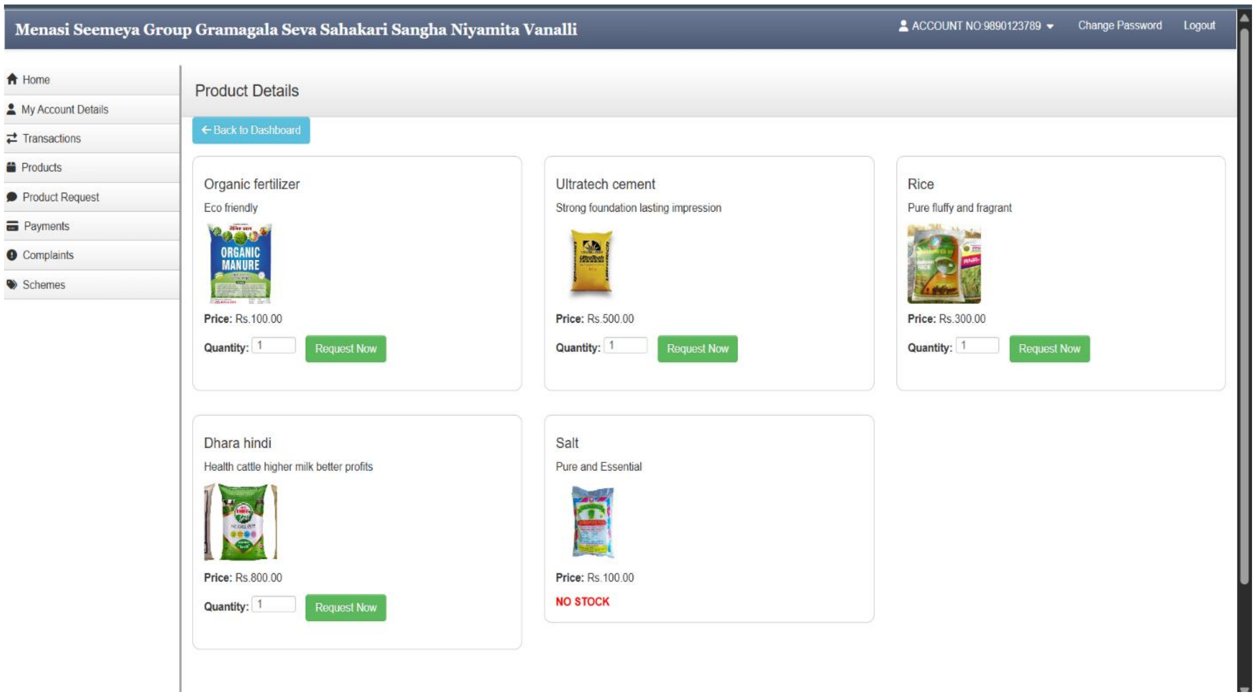
TRANSACTION DETAILS



Transaction Details

Sl No	Account Holder Name	Account Number	Deposit Amount	Withdrawal Amount	Date	Time	Balance
1	Narayan Hegde	9890123789	1,000.00	-	2025-05-12	20:19:00	1,000.00
2	Narayan Hegde	9890123789	-	500.00	2025-05-12	20:29:00	500.00
Total			**1,000.00**	**500.00**			

PRODUCTS VIEW IN CUSTOMER DASHBOARD



Menasi Seemeya Group Gramagala Seva Sahakari Sangha Niyamita Vanalli

ACCOUNT NO:9890123789 Change Password Logout

Product Details

Organic fertilizer
Eco friendly
Price: Rs.100.00
Quantity: 1 Request Now

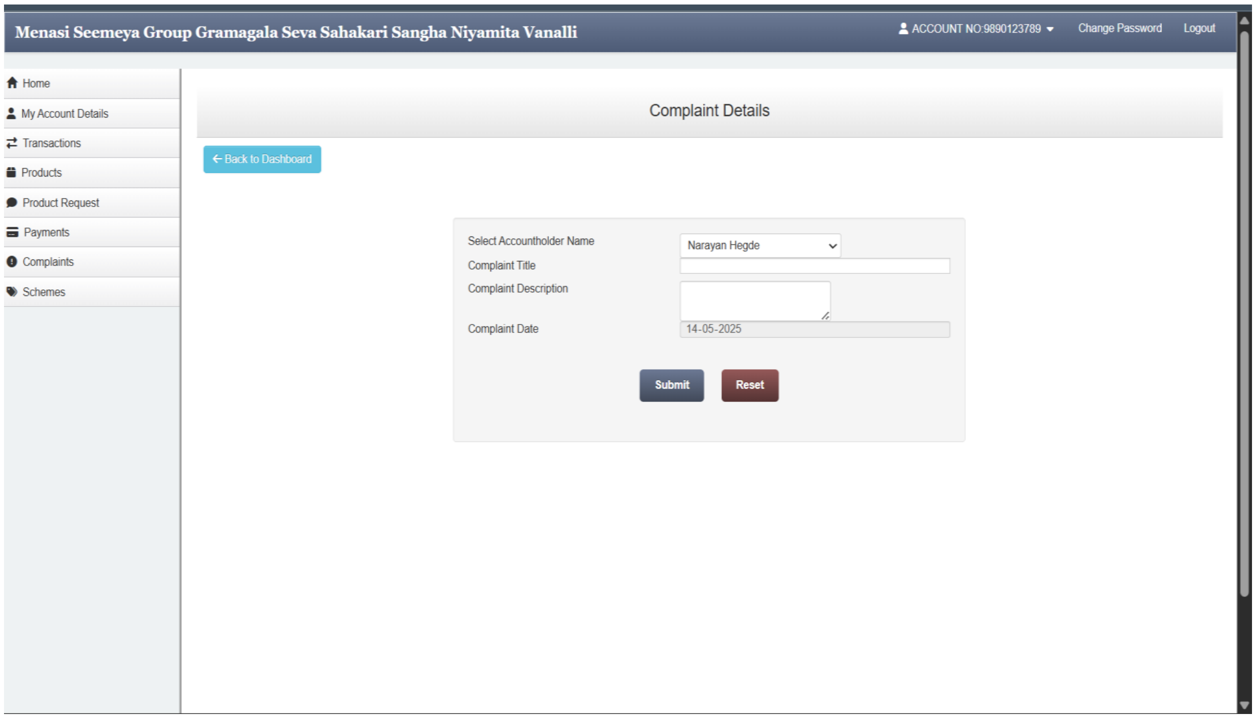
Ultratech cement
Strong foundation lasting impression
Price: Rs.500.00
Quantity: 1 Request Now

Rice
Pure fluffy and fragrant
Price: Rs.300.00
Quantity: 1 Request Now

Dhara hindi
Health cattle higher milk better profits
Price: Rs.800.00
Quantity: 1 Request Now

Salt
Pure and Essential
Price: Rs.100.00
NO STOCK

COMPLAINT DETAILS FORM



Menasi Seemeya Group Gramagala Seva Sahakari Sangha Niyamita Vanalli

ACCOUNT NO:9890123789 Change Password Logout

Complaint Details

Select Accountholder Name	Narayan Hegde
Complaint Title	
Complaint Description	
Complaint Date	14-05-2025

8.TESTING

INTRODUCTION :

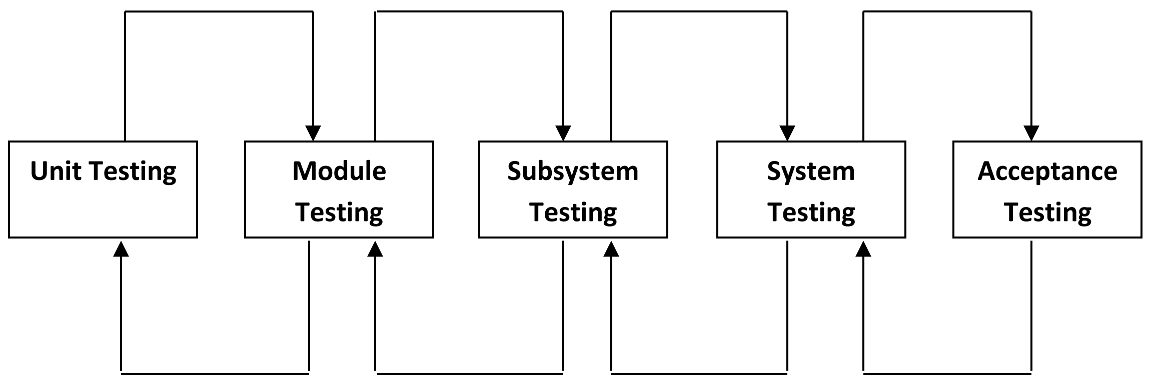
Software Testing is a process of executing program with an indent of finding error. Testing is vital to success of the system. Testing demonstrates that the software functions appear to be working according to the specifications and performance requirements appeared to have been met. If a test is conducted successfully, it will discover errors in the software.

Software Testing consists of all test life cycle activities like static and dynamic testing concerned with planning, preparation and evaluation of software products to determine that the software products satisfy customers requirements and are fit the customer use.

The various strategies that were used in testing this software are as follows:

- Unit Testing
- Integration Testing
- System Testing
    - Validation Testing
    - Black Box Testing
    - White Box Testing

ACCEPTANCE TESTING



UNIT TESTING

Unit testing is done on individual modules as they are completed and become executable. This system was tested with the set of proper test data for each module and results were checked with the expected output. Unit testing focuses on verification effort on the smallest unit of the software design module.

This is also known as MODULE TESTING. This testing is carried out during phases, each module is founded to be working satisfactory as regards to the expected result from the module.

Unit testing involves the design of the test cases that validate the internal program logic is functioning properly, and that program input produces valid output. All decision branches and internal code flow should be validated.

INTEGRATION TESTING

Integration testing ensures that software and subsystems work together as a whole. It tests the interface of all the modules to make sure that the modules behave properly when integrated together. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfied as shown by the Unit testing, the combination of the components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.Integration Testing takes as its input modules that have been unit tested, groups them in larger aggregates, applies tests to aggregates and delivers as its output. The Integration Testing verifies functional, performance, and reliablility requirements placed on a major design items.

FUNCTIONAL TESTING

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

Functional Testing is centered on the following items:

Valid Input	: Identified classes of valid input must be accepted.
Invalid Input	: Identified classes of invalid input must be accepted.
Functions	: Identified functions must be exercised.
Output	: Identified classes of application outputs must be exercised.
Systems/Procedures	: Interfacing systems or procedures must be invoked.

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

SYSTEM TESTING:

In this the entire software system is tested. The reference document for this process is the requirement document. Here the entire software is tested and the performance of the system was observed to see that it satisfies the requirement specification.

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

System testing involves in-house testing of the entire system before delivery to the user. Its aim is to satisfy the user. The system meets all requirements of the client's specifications.

The following are the types of system tests that were carried out for the system:

- **Validation Testing:**

The system has been tested and implemented successfully and thus ensured that all requirements as listed in the software requirements specification are completely fulfilled. In case of erroneous input corresponding error messages are displayed.

- **Black Box Testing:**

This method focuses on the functional requirements of the software. This testing enables to derive set input conditions that will fully exercise all functional requirements of the program. Black Box Testing attempts to find errors in the following category.

  - Incorrect or missing functions.
  - Interface errors.
  - Error in external database access.
  - Performance errors.
  - Initialization and Termination errors.

- **White Box Testing:**

  This is performed early in the testing process, while Black Box testing is applied during the last stage of testing. In this test cases are generated on the logic of each module by drawing flow graphs of that module and logical decisions are tested on all the cases.

It has been used to generate the test case in the following test cases:

- Guarantee that all independent paths have been executed.
- Execute all logical decisions from their True and False side.
- Execute all loops at their boundaries and within their operational bounds.
- Execute internal data structures to ensure their validity.
- Ensure whether all the possible validity checks and validity lookups have been provided to validate data entry.

PERFORMANCE TESTING

Performance Testing can serve different purpose. It can demonstrate that the system meets the performance criteria. It can compare two systems to find which performs better, or it can measure what parts of the system or workload cause the system to perform badly. In the diagnostic case, software engineers use tools such as profilers to measure what parts of a device or software contributes most to the poor performance.

It was a good idea to do our stress testing early on, because it gave us time to fix some of the unexpected deadlocks and stability problems that only occurred when components were exposed to very high transaction volumes.

ACCEPTANCE TESTING :

It is a pre-delivery testing in which entire system is tested at client's site on real world data to find errors. It deals with successful satisfaction of user needs. This project is approved and accepted by the clients. The process flow and execution is 99% working with respect to system testing procedure.

TEST OBJECTIVES

- The system is tested with variety of inputs. The System is tested for accuracy and correctness of the results obtained. Finally the system is tested for inter-operability.
- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

FEATURES TO BE TESTED

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

Testing Overview

During the development of the **Co-Operative Society Management System**, several levels of testing were conducted to ensure system stability and correctness. The following types of testing were implemented:

- **Unit Testing** was used to test individual PHP functions and modules.
- **Integration Testing** ensured that the system components such as deposit handling and account balance updates worked together as expected.

- **Functional Testing** was conducted on each feature including member registration, product management, and transaction processing.
- **Form Validation Testing** was done to ensure users enter valid input and that the system handles incorrect data gracefully.
- **Database Testing** verified data consistency, integrity, and proper CRUD operations in the MySQL database.
- **User Interface Testing** ensured that the web interface was intuitive, responsive, and free of UI bugs.
- **Acceptance Testing** was conducted at the final stage to confirm that all system requirements were fulfilled and the system was ready for deployment.

TEST CASES

EXAMPLE

INTAKE- LOGIN FORM

TEST NO	TESTCASE	EXPECTED RESULT	PASS
1.	Leave the Username and Password textbox blank and press Login	Message slating that provide Username and Password	Passed
2.	Enter the Username and Password	Homepage will display.	Passed
3.	Enter invalid Username and Password	Error message stating that the Invalid Username or Password	Passed
4.	Enter the invalid old Password	Message stating that old Password incorrect.	Passed
5.	Leave all textboxes blank	Required field sign is shown and login button will not work.	Passed

TEST CASE FOR INSERTING DATA

TEST NO	TESTCASE	EXPECTED RESULT	PASS
1.	Leave all the Textboxes empty and press Save Button.	Message should appear that ‘Please fill out this field’.	Passed

TEST CASE FOR DELETE DATA

TEST NO	TESTCASE	EXPECTED RESULT	PASS
1.	Press Delete button without selecting the details to Delete	Message should appear that ‘Please select a record to delete’.	Passed
2.	Press Delete button after selecting a member	Message should appear that ‘Do you want delete?’	Passed

TEST CASE FOR UPDATE DATA

TEST NO	TESTCASE	EXPECTED RESULT	PASS
1.	Press update button without selecting a member	Message should appear that ‘Please enter the data’.	Passed

9.LIMITATIONS OF THE PROJECT

- **Scalability issues :** Handling a large number of transactions and members may slow down performance.
- **Security risks :** Sensitive financial data is at risk without strong encryption and authentication measures.
- **No multi -branch support :** If a cooperative has multiple branches, integrating them might be challenging.

10.FUTURE SCOPE AND ENHANCEMENT

The Co-Operative Society Management System is designed to be scalable, allowing new features to be added as needed. In the future, the following enhancements can be implemented:

1. Online money transaction integration
2. Multi-language support
3. Member login portal
4. Multi-branch management
5. SMS/Email notifications
6. Mobile app version for easier access

These additions will improve usability, accessibility, and overall system efficiency.

11.CONCLUSION

The Co-Operative Society Management System has been successfully designed and implemented to automate the core operations of a cooperative society. The system provides an efficient, secure, and user-friendly platform for managing member information, processing deposit and withdrawal transactions, checking balances, and handling product inventory. By replacing traditional manual methods, the system minimizes human errors, reduces paperwork, and improves the overall speed and accuracy of daily tasks.

Through a centralized and structured interface, the admin can easily perform all necessary operations from a single dashboard. The use of open-source technologies like PHP and MySQL makes the system cost-effective and maintainable. The successful implementation of this system ensures better record management, real-time data access, and a scalable foundation for future enhancements such as multi-branch integration or member self-service modules.

Overall, the project fulfills its objective of providing a reliable and practical solution to streamline cooperative society operations.

12.SOURCE CODE

LOGIN PAGE

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Cooperative Society Login</title>
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.4.0/css/all.min.css">
<form id="loginForm" method="post" action="logcheck.php">
      <div class="input-group">
        <i class="fas fa-user input-icon"></i>
        <input type="text" class="input-field" id="usernameInput" name="username" placeholder="Username" required>
      </div>

      <div class="input-group">
        <i class="fas fa-lock input-icon"></i>
        <input type="password" id="password" name="password" class="input-field" placeholder="Password" required>
        <i class="fas fa-eye password-toggle" id="togglePassword"></i>
      </div>

      <div class="options-row">
        <label class="remember-me">
          <input type="checkbox" id="remember" name="remember">
          <span id="rememberText">Remember me</span>
        </label>
        <a href="forgot.php" class="forgot-password" id="forgotPassword">Forgot password?</a>
      </div>

      <div class="captcha-container">
        <div class="captcha-header">
          <i class="fas fa-shield-alt"></i>
          <span id="captchaPrompt">Verify you're human</span>
        </div>
        <div class="captcha-display">
          <div class="captcha-text" id="captchaText"></div>
          <button type="button" class="refresh-captcha" id="refreshCaptcha">
            <i class="fas fa-sync-alt"></i>
        </button>
      </div>
            <input type="text" class="captcha-input" id="captchaInput" name="captcha"
placeholder="Enter the CAPTCHA code" required>
      <div class="captcha-error" id="captchaError">Incorrect CAPTCHA. Please try again.</div>
     </div>

        <input type="submit" class="login-btn" name="submit" value="Login to Dashboard"
id="loginButton">
    </form>
  </div>

  <script>
    const togglePassword = document.querySelector('#togglePassword');
    const password = document.querySelector('#password');
    const loginForm = document.querySelector('#loginForm');
    const captchaText = document.querySelector('#captchaText');
    const captchaInput = document.querySelector('#captchaInput');
    const refreshCaptcha = document.querySelector('#refreshCaptcha');
    const captchaError = document.querySelector('#captchaError');
    const languageToggle = document.querySelector('#languageToggle');
    const languageText = document.querySelector('#languageText');

    // Language content
    const content = {
      en: {
        welcome: "Welcome",
        loginPrompt: "Login to access your cooperative account",
        username: "Username",
        password: "Password",
        remember: "Remember me",
        forgotPassword: "Forgot password?",
        captchaPrompt: "Verify you're human",
        captchaPlaceholder: "Enter the CAPTCHA code",
        captchaError: "Incorrect CAPTCHA. Please try again.",
        loginButton: "Login to Dashboard",
      },

    let currentLanguage = 'en';
    let captchaSolution;

    // Toggle language
    function toggleLanguage() {
      currentLanguage = currentLanguage === 'en' ? 'kn' : 'en';
  updateContent();
}

// Update all text content based on current language
function updateContent() {
  const lang = content[currentLanguage];

  // Update all text elements
  document.getElementById('welcomeText').textContent = lang.welcome;
  document.getElementById('loginPrompt').textContent = lang.loginPrompt;
  document.getElementById('usernameInput').placeholder = lang.username;
  document.getElementById('password').placeholder = lang.password;
  document.getElementById('rememberText').textContent = lang.remember;
  document.getElementById('forgotPassword').textContent = lang.forgotPassword;
  document.getElementById('captchaPrompt').textContent = lang.captchaPrompt;
  document.getElementById('captchaInput').placeholder = lang.captchaPlaceholder;
  document.getElementById('captchaError').textContent = lang.captchaError;
  document.getElementById('loginButton').value = lang.loginButton;
  languageText.textContent = lang.switchLanguage;
}

// Generate random alphanumeric CAPTCHA
function generateCaptcha() {
  const chars = 'ABCDEFGHJKLMNPQRSTUVWXYZabcdefghjkmnpqrstuvwxyz23456789';
  let result = '';
  const length = 6;

  for (let i = 0; i < length; i++) {
    result += chars.charAt(Math.floor(Math.random() * chars.length));
  }

  captchaSolution = result;
  captchaText.textContent = result;
  captchaInput.value = '';
  captchaError.style.display = 'none';
}

// Toggle password visibility
togglePassword.addEventListener('click', function() {
  const type = password.getAttribute('type') === 'password' ? 'text' : 'password';
  password.setAttribute('type', type);
  this.classList.toggle('fa-eye');
  this.classList.toggle('fa-eye-slash');
});
```

```
  // Refresh CAPTCHA
  refreshCaptcha.addEventListener('click', generateCaptcha);

  // Form submission with CAPTCHA validation
  loginForm.addEventListener('submit', function(e) {
    const userInput = captchaInput.value.trim();

    if(userInput === '' || userInput !== captchaSolution) {
      e.preventDefault();
      captchaError.style.display = 'block';
      generateCaptcha();
      return false;
    }

    // CAPTCHA verified - allow form submission
    // The form will now submit to logcheck.php with all form data
  });

  // Language toggle event
  languageToggle.addEventListener('click', toggleLanguage);

  // Initialize
  generateCaptcha();
 </script>
```

CHANGE PASSWORD

```
<?php include('meta_tags.php');?>
<?php include('menus.php');?>


  <?php include('side_menu.php');?>


  <div class="content">

<div class="header">

        <h1 class="page-title">Account Holder  Details</h1>
      </div>
```

```
        <div class="container-fluid">
      <div class="row-fluid">

<div class="well">

  <div id="myTabContent" class="tab-content">
    <div class="tab-pane active in" id="home">
<?php include('val.php');?>
<form name="form1" method="post" id="formID" action="changepassword.php" class="form-
horizontal">
 <table class="table table-striped table-bordered" id="example">

  <tr>
    <td width="163">Old Password*</td>
    <td width="328"><input name="old_password" type="password" id="old_pwd"
required="1" /></td>
  </tr>
  <tr>
    <td>New Password* </td>
    <td><input name="new_password" type="password" id="new_password" required="1"
/></td>
  </tr>
  <tr>
    <td>Confirm Password* </td>
    <td><input name="confirm_password" type="password" id="con_pwd" required="1"
/></td>
  </tr>
  <tr>
    <td height="29" colspan="2"><div align="center">
    <input type="submit" name="Submit" value="Submit" class="btn btn-primary">
      <input type="reset" name="Reset" value="Reset" class="btn btn-danger">
     </div></td>
  </tr>
 </table>
 <div align="center"></div>
</form>

</div>
 </div>

</div>

<div class="modal small hide fade" id="myModal" tabindex="-1" role="dialog" aria-
labelledby="myModalLabel" aria-hidden="true">
```

```
  <div class="modal-header">
    <button type="button" class="close" data-dismiss="modal" aria-hidden="true">□</button>
    <h3 id="myModalLabel">Delete Confirmation</h3>
  </div>
  <div class="modal-body">

    <p class="error-text"><i class="icon-warning-sign modal-icon"></i>Are you sure you want to
delete the user?</p>
  </div>
  <div class="modal-footer">
    <button class="btn" data-dismiss="modal" aria-hidden="true">Cancel</button>
    <button class="btn btn-danger" data-dismiss="modal">Delete</button>
  </div>
</div>


          <?php include('footer.php');  ?>
```

CREATION FORM CODE FOR PRODUCTS

```
<body>
<!-- Back to Admin Dashboard Button -->
<a href="productdet_view.php" class="btn btn-info" style="margin-bottom: 15px;">
  <i class="fas fa-arrow-left"></i> Back to Dashboard
</a>

<div class="header">
  <h1 class="page-title">Product Details</h1>
</div>

<form action="productdet_insert.php" method="post" enctype="multipart/form-data"
name="form1" id="formID">
  <table border="0">
    <tr>
      <td>Product Name</td>
      <td><input name="product_name" class="validate[required,custom[onlyLetter]]"
type="text" id="product_name"></td>
    </tr>
    <tr>
      <td>Product Description</td>
      <td><input name="product_desc" class="validate[required]" type="text"
id="product_desc"></td>
    </tr>
    <tr>
      <td>Product Image</td>
      <td><input name="product_img" type="file" id="product_img"></td>
    </tr>
    <tr>
      <td>Product Price</td>
      <td><input name="product_price" class="validate[required,custom[onlyNumber]]"
type="text" id="product_price"></td>
    </tr>
  </table>

  <div class="form-footer">
    <input type="submit" name="Submit" value="Submit">
    <input type="reset" name="reset" value="Reset">
  </div>
</form>

<?php include('footer.php'); ?>

<!-- JavaScript for Form Validation -->
<script>
document.getElementById('formID').addEventListener('submit', function(event) {
  // Get form field values
  let productName = document.getElementById('product_name').value;
  let productDesc = document.getElementById('product_desc').value;
  let productImage = document.getElementById('product_img').value;
  let productPrice = document.getElementById('product_price').value;

  // Check if any required field is empty
  if (!productName || !productDesc || !productImage || !productPrice) {
    event.preventDefault(); // Prevent form submission
    // Show pop-up error message
    alert('Please fill in all the required fields.');
    return false; // Prevent further execution
  }

  // Additional validation logic if needed (e.g., price as a number)
  if (isNaN(productPrice)) {
    event.preventDefault(); // Prevent form submission
    alert('Please enter a valid product price.');
    return false; // Prevent further execution
  }
});
</script>
```

13.BIBLIOGRAPHY

REFERENCES

1. *Software Engineering* – 9th Edition by Ian Sommerville
2. *Fundamentals of Database Systems* – B. Navathe
3. Ivan Bayross (2010). Web Enabled Commercial Application Development Using HTML, JavaScript, DHTML and PHP. BPB Publications.
4. Achyut S. Godbole & Atul Kahate (2008). Web Technologies: TCP/IP to Internet Application Architectures. McGraw-Hill Education.
5. HTML & CSS: The Complete Reference, Fifth Edition, Thomas A Powell.-2017
6. MASTERING HTML, CSS & Java Script Web Publishing, Laura Lemay

WEBSITES

1. https://www.w3schools.com – Web development tutorials and references
2. https://getbootstrap.com – Bootstrap framework for responsive design
3. https://www.youtube.com – For development tutorials and UI design help
4. https://chat.openai.com – ChatGPT AI assistance for coding and explanation
5. https://deepseek.com – AI-based development help and coding suggestions

AI Tools Used:

During the development and documentation of the LocalHandz project, the following AI tools and platforms were utilized to improve efficiency, assist with code generation, and enhance content writing.

1. **ChatGPT(OpenAI):**

Used extensively for code suggestions, system design drafting, technical documentation writing, testing strategy planning, and troubleshooting PHP-MySQL integration issues. It also assisted in refining report sections like system design, development, testing, and user manuals.

2. **GitHubCopilot:**

Integrated within the code editor (VS Code), GitHub Copilot provided real-time code completion and intelligent suggestions for HTML, CSS, JavaScript, and PHP. It significantly accelerated backend logic implementation and frontend validations.

3. **Canva AI (for diagrams):**

Assisted in creating clean and visually appealing architecture and system design diagrams using AI-powered templates and layouts.