# Chapter 1

# INTRODUCTION

Inventory management is a critical aspect of any business that deals with physical goods. It refers to the process of overseeing and controlling the amount of stock a business holds. This includes tracking inventory levels, orders, sales, and deliveries. The goal of inventory management is to ensure that a business has the right amount of inventory at the right time, in order to meet customer demand and maintain profitability.

Effective inventory management helps a business avoid stockouts, which can lead to lost sales and disappointed customers. It also helps a business avoid overstocking, which can lead to wasted resources such as storage space and capital tied up in excess inventory. Additionally, effective inventory management allows a business to make better decisions about ordering, pricing, and product mix.

In this report, we will delve deeper into the concept of inventory management and its importance for businesses. We will discuss the various inventory management techniques and strategies that can be employed, as well as the tools and technologies available to assist with inventory management. We will also look at some of the common challenges faced by businesses when managing inventory, and provide recommendations for overcoming these challenges. Ultimately, the goal of this report is to provide a comprehensive understanding of inventory management and its role in the success of a business.



## 1.1 Overview of Database Management Systems

A Database Management System (DBMS) is a software system that is designed to manage and organize large amounts of data. It allows users to create, modify, and query databases, as well as control access to the data stored within them. The DBMS is the foundation of many business and organizational systems, providing a way to efficiently store and retrieve data.

There are several different types of DBMSs, including:

- Relational Database Management Systems (RDBMS): The most widely used type of DBMS, RDBMSs store data in tables with rows and columns. Each row represents a single record, and each column represents a specific field within that record. RDBMSs use SQL (Structured Query Language) to interact with the data stored in the tables.

- Hierarchical Database Management Systems (HDBMS): HDBMSs are organized in a tree-like structure, with each record having one parent and multiple children. These types of databases are used primarily in legacy systems.
- Network Database Management Systems (NDBMS): NDBMSs are similar to HDBMSs, but allow for multiple parent-child relationships between records.
- Object-Oriented Database Management Systems (OODBMS): OODBMSs store data as objects, which can be grouped into classes and subclasses. This type of DBMS is used primarily in object-oriented programming languages.
- Document-Oriented Database Management Systems (DDBMS): DDBMSs store data as documents, such as XML or JSON, rather than as rows and columns. This type of DBMS is commonly used in NoSQL databases.

DBMSs are critical for maintaining the integrity, security, and performance of data. They also provide users with advanced features such as data backup, recovery, and reporting.

## 1.2 Problem Statement

An inventory management system provides an overview of a company's inventory levels, including information on the quantity of goods on hand. A problem statement for an inventory management report might include issues such as:

- Difficulty in accurately tracking inventory levels: Without an efficient system for tracking inventory, it can be difficult to know how much of a particular product is on hand and when to order more.
- Inefficient use of resources: Without accurate inventory data, it may be difficult to determine which products are selling well and which are not, leading to wasted resources on slow-moving items.
- Stockouts and overstocking: Without an accurate inventory management system, it is easy for a business to run out of stock or overstock items, leading to lost sales and wasted resources respectively.
- Lack of visibility into inventory movement: Without detailed inventory tracking and reporting, it can be difficult to understand how products are moving through the supply chain, making it difficult to identify and address bottlenecks or other issues that may be affecting inventory levels.
- Difficulty in forecasting demand: Without accurate inventory data, it can be difficult to forecast demand for products, making it hard to plan production and purchasing.

- Difficulty in identifying opportunities for cost savings: Without detailed inventory data, it can be difficult to identify opportunities for cost savings, such as reducing the number of suppliers or negotiating better prices with existing suppliers.
- Difficulty in identifying and managing inventory risks such as expiration, obsolescence, and spoilage: A lack of inventory management can lead to unexpected inventory risks that can result in significant financial loss for a business.

Overall, the problem statement of an inventory management report can be summarized as the need for an efficient, accurate and detailed system for tracking inventory levels and movement, to improve the visibility, forecasting and management of the inventory, and to help identify and manage risks and cost savings opportunities.

## 1.3 OBJECTIVES

- Wide range of collection based on the requirement
- Simple User-friendly website
- Focus on the management of stocks in the store

## 1.4 DATASET DESCRIPTIONS

Inventory Management System contains:

- User dataset for obtaining the user credentials like username, password, email, contact, and address.
- Login dataset to keep track of login time of each user when they login.
- Categories dataset to have the list of categories and Stock dataset to keep track of the stock of each item in the inventory.
- Suppliers dataset has the list of suppliers and the items they supply to the inventory along with the quantity.

# Chapter 2

# SYSTEM REQUIREMENTS

## 2.1 SOFTWARE

- Back end: JavaScript (JS), Hypertext PreProcessor (PHP)
- Frontend: MySQL, Cascading Style Sheets (CSS), Hypertext Markup Language (HTML)

### MySQL

MySQL is a popular open-source relational database management system (RDBMS) that is widely used in web development and other applications. It is known for its reliability, ease of use, and performance. MySQL supports a wide variety of data types and can be used with many programming languages, including PHP, Python, and Java. It is often used in conjunction with the LAMP stack (Linux, Apache, MySQL, and PHP) for web development. MySQL is supported by Oracle Corporation and is available for free under the GNU General Public License.

### Hypertext PreProcessor

PHP (Hypertext PreProcessor) is an open-source, server-side scripting language that is widely used for web development. It is particularly well-suited for creating dynamic web pages and can be embedded directly into HTML code. PHP can be used to perform a wide range of tasks, including connecting to and querying databases, handling form data, and creating sessions. It also has built-in support for working with cookies and sessions and can be used in conjunction with various web frameworks such as Laravel, CodeIgniter, and Yii. PHP can be run on most web servers and is compatible with a wide variety of operating systems, including Windows, Linux, and macOS.

### JavaScript

JavaScript is a programming language that is widely used to create interactive and dynamic web pages. It is a client-side scripting language, which means that it runs on the user's web browser rather than on a web server. JavaScript can be used to add interactivity to web

pages, such as form validation, image sliders, and pop-up windows. It can also be used to create dynamic effects, such as animation, and to make web pages more responsive to user input. JavaScript can be run on all major web browsers and is supported by all major web development platforms. It is also widely used for creating web and mobile apps, using frameworks such as React, Angular and Vue.js. JavaScript is also commonly used on the server side, using Node.js.

## Cascading Style Sheets

CSS (Cascading Style Sheets) is a stylesheet language used for describing the presentation of a document written in a markup language. It is most used to style web pages written in HTML and XHTML, but can also be applied to any kind of XML document, including plain XML, SVG and XUL. CSS allows developers to separate the presentation of a document from its content, making it easier to maintain and update. It provides a wide range of styling options, including colours, fonts, layouts, and animations. CSS can be written in separate files or included within the HTML document itself. CSS has various levels such as CSS1, CSS2, CSS3 and the latest CSS4, each level added new features and capabilities. With the latest CSS3, it brings new layout modules such as Flexbox and Grid, animations, and responsive design capabilities to create a better web page layout and design. Overall, CSS plays a vital role in web development and design, as it allows developers to create visually appealing and well-structured web pages that are easy to navigate and use.

## HyperText Markup Language

HTML (Hypertext Markup Language) is the standard markup language used to create web pages. It consists of a series of tags and attributes that are used to define the structure and content of a web page. HTML tags are used to create the basic structure of a web page, such as headings, paragraphs, lists, and links, while attributes are used to provide additional information about the elements on a web page, such as id, class, and style. HTML documents are typically viewed in web browsers, and can be created and edited using text editors or specialized software such as Adobe Dreamweaver.

# Chapter 3

## SYSTEM DESIGN

### 3.1 ER DIAGRAM

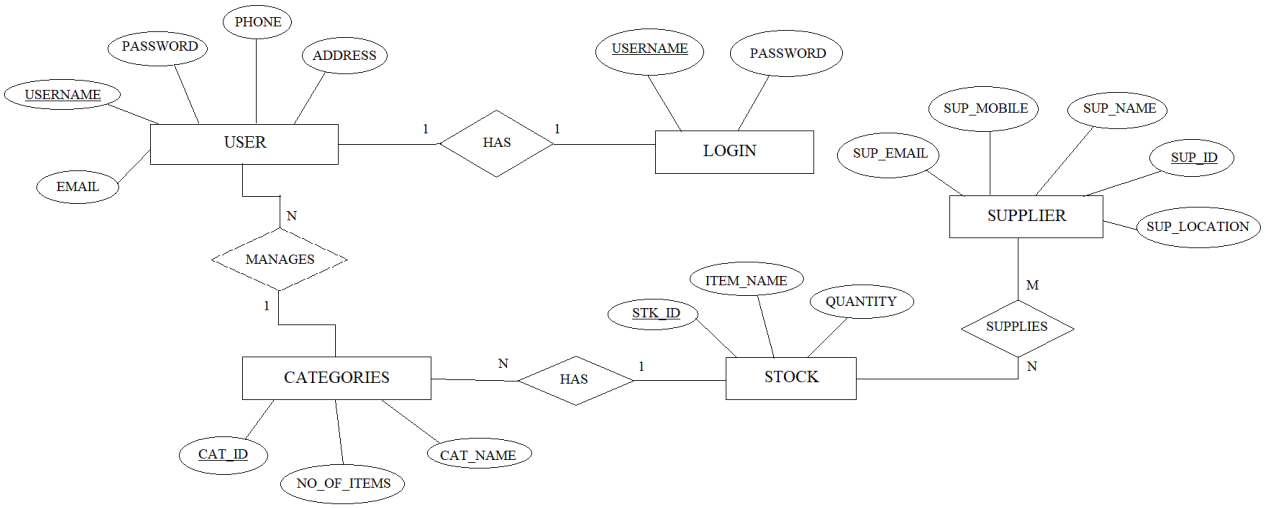



Figure 3.1: ER Diagram

Platform used for designing the ER diagram: Paint

### 3.2 ER TO RELATIONAL MAPPING

Step 1: For each regular entity type E in the ER schema, create a relation R that includes all the simple attributes of E. Here the strong entities are User, Categories, Supplier and Stock.

Step 2: For weak entity type LOGIN in the ER schema with owner entity type USER, create a relation R, and include all simple attributes of LOGIN as attributes. In addition, include as foreign key attributes of R the primary key attribute of the relation that correspond to the owner entity type.

Step 3: For binary 1:1 relationship type R in the ER schema, identify the relations USER and LOGIN that correspond to the entity types participating in R. Choose the relation LOGIN, and include the primary key of USER as a foreign key in LOGIN.

Step 4: For regular binary 1:N relationship type R identify the relation (N) relation S. the primary key of T as a foreign key of S. Simple attributes of R map to attributes of S.

Step 5: For each binary M:N relationship type supplies, create a relation SUPPLIES. Include the primary keys of SUPPLIER and STOCK relations as foreign keys in SUPPLIES. Their

combination will be the primary key for SUPPLIES. Simple attributes of supplies become attributes of SUPPLIES.

Step 6: Not applicable for the current ER diagram

Step 7: Not applicable for the current ER diagram

## 3.3 SCHEMA DIAGRAM

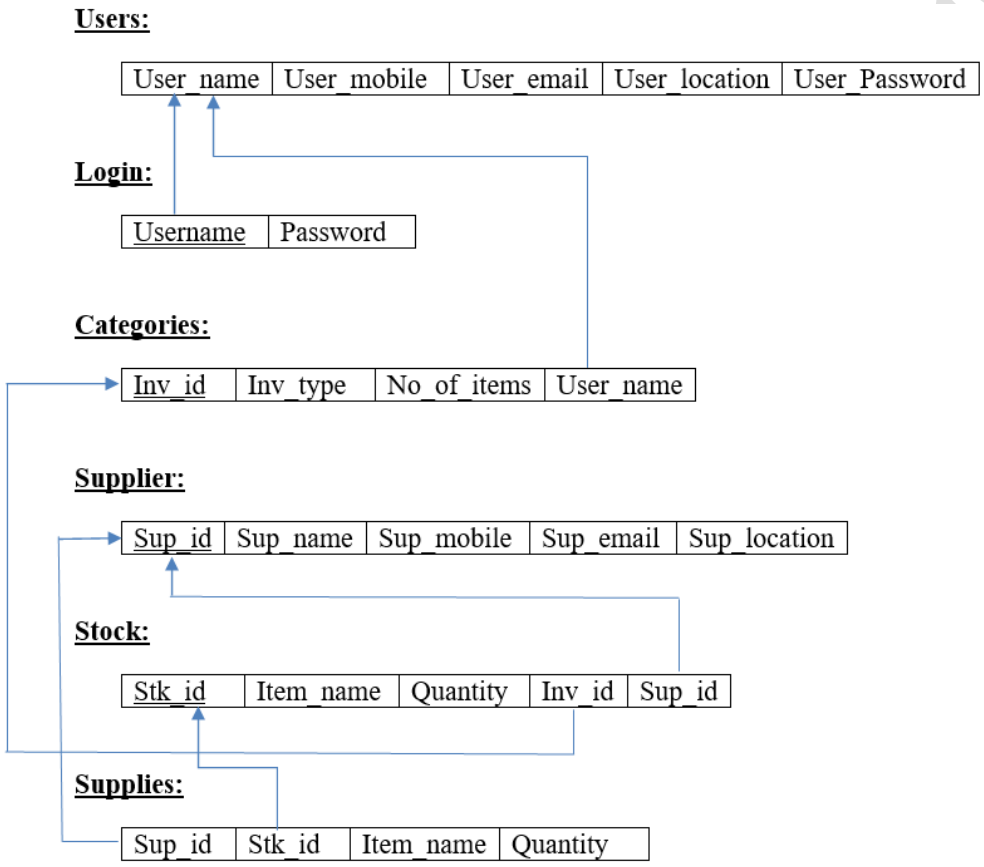



Figure 3.2: Schema

Here the primary key attribute of entity ‘USERS’, ’CATEGORIES’, ‘STOCK’, ‘SUPPLIERS’, are ‘username’, ‘inv_id’, ‘stk_id’, ‘sup_id’ respectively.

Platform used for designing schema: MS Word

## 3.4 OVERVIEW OF GUI

GUI is a program interface that takes advantage of the computer's graphics capabilities to make the program easier to use. A graphical user interface (GUI is an interface through

which a user interacts with electronic devices such as computers and smartphones using icons, menus, and other visual indicators or representations (graphics).

GUIs graphically display information and related user controls, unlike text-based interfaces, where data and commands are strictly in text. GUI representations are manipulated by a pointing device such as a mouse, trackball, or stylus, or by a finger on a touch screen. Well-designed graphical user interfaces can free the user from learning complex command languages. On the other hand, many users find that they work more effectively with a command-driven interface, especially if they already know the command language.

phpMyAdmin is a free and open-source web-based tool for managing MySQL and MariaDB databases. It is written in PHP and allows users to perform various database management tasks, such as creating and modifying tables, managing users and permissions, and executing SQL statements. One of the main advantages of phpMyAdmin is its ease of use. It has a user-friendly interface that allows users to perform complex database management tasks without having to write any SQL code.

It also supports various features such as browsing and editing records, managing database structure, and importing and exporting data. phpMyAdmin also supports a wide range of operating systems and web servers, making it a versatile choice for managing MySQL databases. It can be easily installed on a web server and configured to manage multiple databases. Additionally, it also supports several languages, so it is easy to use for non-English speakers.

Overall, phpMyAdmin is a powerful and widely used tool for managing MySQL databases. It is well-suited for both beginners and advanced users, and it is a good choice for managing databases on a web serve.

## 3.5 NORMALIZATION

Normalization is a process of analysing the given relation schema based on their functional dependencies and primary key to achieve desirable properties of minimizing redundancy and minimizing insert, delete, and update anomaly. The normalization process takes a relation schema through a series of tests to certify whether it satisfies a certain normal form. The normal form of a relation refers to the highest normal form condition that it

meets, and hence the degree to which it has been normalized. Normalization rules are divided into the following normal form.

- First Normal Form
- Second Normal Form
- Third Normal Form

## First Normal Form

The first normal form states that the domain of an attribute must include only atomic (simple, individual) values and that the value of any attribute in a tuple must be a single value from the domain of the attribute. Consider the relations of the Inventory Management System where all the relations are in INF as they have neither any multivalued attributes nor composite attributes.

Hence the relations are said to be in INF.

## Second Normal Form

The second normal form is based on the concept of full functional dependency. A functional dependency X → Y is a full functional dependency if the removal of any attribute A from X means that the dependency does not hold anymore. A relation schema R is in 2NF if every nonprime attribute A in R is fully functionally dependent on the primary key of R.

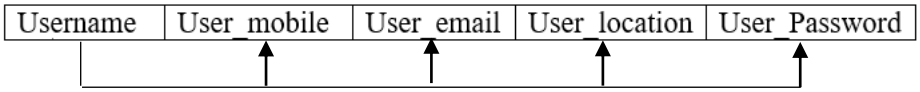
## Third Normal Form

The third normal form is based on the concept of transitive dependency. A relation schema R is in 3NF if it satisfies 2NF and no nonprime attribute of R is transitively dependent on the primary key. A relation schema R is in 3NF if every nonprime attribute of R meets both of the following conditions:

• It is fully functionally dependent on every key of R.

• It is non-transitively dependent on every key of R. The relations used in this database are fully functionally dependent on its key attribute and do not hold any transitive dependencies. Hence all the relations are in 3NF.

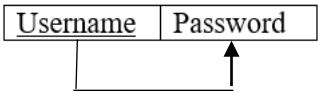
**USER**




It is in INF as it does not have any composite or multivalued attributes.

- It is in 2NF as it is fully functionally dependent.
- (Username) → (User_mobile, User_email, User_location, User_password)
- As the relation does not contain a non-key attribute functionally determining other non-key attributes, it is in 3NF (there is no transitivity in attributes).

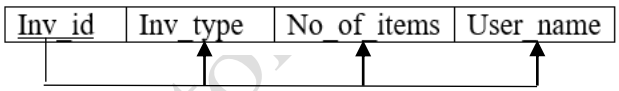
**LOGIN**




It is in INF as it does not have any composite or multivalued attributes.

- It is in 2NF as it is fully functionally dependent.
- (Username) → (Password)
- As the relation does not contain a non-key attribute functionally determining other non-key attributes, it is in 3NF (there is no transitivity in attributes).

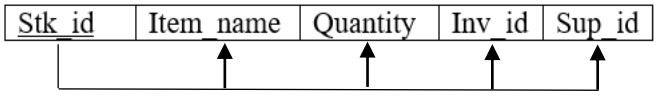
**CATEGORIES**




It is in INF as it does not have any composite or multivalued attributes.

- It is in 2NF as it is fully functionally dependent.
- (Inv_id) → (Inv_type, no_of_items, user_name)
- As the relation does not contain a non-key attribute functionally determining other non-key attributes, it is in 3NF (there is no transitivity in attributes).

**STOCK**



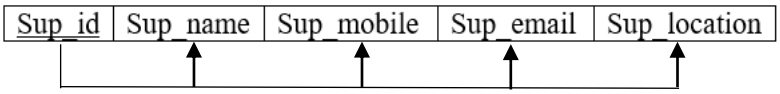
It is in INF as it does not have any composite or multivalued attributes.

• It is in 2NF as it is fully functionally dependent.

• (Stk_id) → (item_name, quantity, inv_id, sup_id)

• As the relation does not contain a non-key attribute functionally determining other non-key attributes, it is in 3NF (there is no transitivity in attributes).

**SUPPLIER**



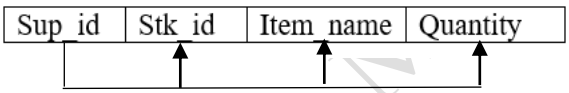

It is in INF as it does not have any composite or multivalued attributes.

• It is in 2NF as it is fully functionally dependent.

• (Sup_id) → (Sup_name, Sup_mobile, Sup_email, Sup_location)

• As the relation does not contain a non-key attribute functionally determining other non-key attributes, it is in 3NF (there is no transitivity in attributes).

**SUPPLIES**




It is in INF as it does not have any composite or multivalued attributes.

• It is in 2NF as it is fully functionally dependent.

• (Sup_id, Stk_id) → (Item_name, Quantity)

• As the relation does not contain a non-key attribute functionally determining other non-key attributes, it is in 3NF (there is no transitivity in attributes).

# Chapter 4

# IMPLEMENTATION

## 4.1 TABLE CREATION

```
-- Table structure for table `users`
--

CREATE TABLE `users` (
  `USER_NAME` varchar(25) NOT NULL,
  `USER_MOBILE` decimal(10,0) DEFAULT NULL,
  `USER_EMAIL` varchar(25) DEFAULT NULL,
  `USER_LOCATION` varchar(50) DEFAULT NULL,
  `USER_PASSWORD` varchar(10) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

Figure 4.1: Creation of the table 'USERS'

```
-- Table structure for table `login`
--

CREATE TABLE `login` (
  `USER_NAME` varchar(20) DEFAULT NULL,
  `USER_PASSWORD` varchar(16) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

Figure 4.2: Creation of the table 'LOGIN'

```
-- Table structure for table `categories`
--

CREATE TABLE `categories` (
  `INV_ID` int(4) NOT NULL,
  `INV_NAME` varchar(20) DEFAULT NULL,
  `NO_OF_ITEMS` decimal(3,0) DEFAULT NULL,
  `USER_NAME` varchar(25) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

Figure 4.3: Creation of 'CATEGORIES'

```sql
-- Table structure for table `stock`
--

CREATE TABLE `stock` (
  `STK_ID` int(4) NOT NULL,
  `ITEM_NAME` varchar(15) DEFAULT NULL,
  `QUANTITY` int(3) DEFAULT NULL,
  `SUP_ID` int(4) DEFAULT NULL,
  `INV_ID` int(4) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

Figure 4.4: Creation of the table 'STOCK'

```sql
-- Table structure for table `supplier`
--

CREATE TABLE `supplier` (
  `SUP_ID` int(4) NOT NULL,
  `SUP_NAME` varchar(25) DEFAULT NULL,
  `SUP_MOBILE` decimal(10,0) DEFAULT NULL,
  `SUP_EMAIL` varchar(25) DEFAULT NULL,
  `SUP_LOCATION` varchar(50) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

Figure 4.5: Creation of 'SUPPLIER'

```sql
-- Table structure for table `supplies`
--

CREATE TABLE `supplies` (
  `SUP_ID` int(4) DEFAULT NULL,
  `STK_ID` int(4) DEFAULT NULL,
  `ITEM_NAME` varchar(15) DEFAULT NULL,
  `QUANTITY` int(3) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

Figure 4.6: Creation of the table 'SUPPLIES'

## 4.2 DESCRIPTION OF TABLES

```
desc users;
```

[ Edit inline ] [ Edit ] [ Create PHP code ]

Extra options

| Field | Type | Null | Key | Default | Extra |
|---|---|---|---|---|---|
| USER_NAME | varchar(25) | NO | PRI | *NULL* | |
| USER_MOBILE | decimal(10,0) | YES | | *NULL* | |
| USER_EMAIL | varchar(25) | YES | | *NULL* | |
| USER_LOCATION | varchar(50) | YES | | *NULL* | |
| USER_PASSWORD | varchar(10) | YES | | *NULL* | |

Figure 4.7: Description of 'USERS'

```
desc login;
```

[ Edit inline ] [ Edit ] [ Create PHP code ]

Extra options

| Field | Type | Null | Key | Default | Extra |
|---|---|---|---|---|---|
| USER_NAME | varchar(20) | YES | MUL | *NULL* | |
| LOGIN_TIME | timestamp | YES | | *NULL* | |

Figure 4.8: Description of 'LOGIN'

```
desc categories;
```

[ Edit inline ] [ Edit ] [ Create PHP code ]

Extra options

| Field | Type | Null | Key | Default | Extra |
|---|---|---|---|---|---|
| INV_ID | int(4) | NO | PRI | *NULL* | |
| INV_NAME | varchar(20) | YES | | *NULL* | |
| NO_OF_ITEMS | decimal(3,0) | YES | | *NULL* | |
| USER_NAME | varchar(25) | YES | MUL | *NULL* | |

Figure 4.9: Description of 'CATEGORIES'

```
desc stock;
```

[ Edit inline ] [ Edit ] [ Create PHP code ]

Extra options

| Field | Type | Null | Key | Default | Extra |
|---|---|---|---|---|---|
| STK_ID | int(4) | NO | PRI | *NULL* | |
| ITEM_NAME | varchar(15) | YES | | *NULL* | |
| QUANTITY | int(3) | YES | | *NULL* | |
| SUP_ID | int(4) | YES | MUL | *NULL* | |
| INV_ID | int(4) | YES | MUL | *NULL* | |

Figure 4.10: Description of 'STOCK'

```
desc supplier;
```

[ Edit inline ] [ Edit ] [ Create PHP code ]

Extra options

| Field | Type | Null | Key | Default | Extra |
|---|---|---|---|---|---|
| SUP_ID | int(4) | NO | PRI | *NULL* | |
| SUP_NAME | varchar(25) | YES | | *NULL* | |
| SUP_MOBILE | decimal(10,0) | YES | | *NULL* | |
| SUP_EMAIL | varchar(25) | YES | | *NULL* | |
| SUP_LOCATION | varchar(50) | YES | | *NULL* | |

Figure 4.11: Description of 'SUPPLIER'

```
desc supplies;
```

[ Edit inline ] [ Edit ] [ Create PHP code ]

Extra options

| Field | Type | Null | Key | Default | Extra |
|---|---|---|---|---|---|
| SUP_ID | int(4) | YES | MUL | *NULL* | |
| STK_ID | int(4) | YES | MUL | *NULL* | |
| ITEM_NAME | varchar(15) | YES | | *NULL* | |
| QUANTITY | int(3) | YES | | *NULL* | |

Figure 4.12: Description of 'SUPPLIES'

## 4.3 POPULATED TABLES

```
SELECT * FROM `users`
```

Profiling [ Edit inline ] [ Edit ] [ Explain SQL ] [ Create PHP code ] [ Refresh ]

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

Extra options

| | USER_NAME | USER_MOBILE | USER_EMAIL | USER_LOCATION | USER_PASSWORD |
|---|---|---|---|---|---|
| Edit Copy Delete | mohan | 7654328765 | mohan@gmail.com | bangalore | 76543 |
| Edit Copy Delete | prisha | 2345676533 | prisha@gmail.com | Bangalore | 12345 |
| Edit Copy Delete | shreya | 7654328765 | shreya@gmail.com | bangalore | 5678 |
| Edit Copy Delete | tejal | 8765432876 | tejal@gmail.com | bangalore | 5678 |

Figure 4.13: 'USERS' table with values

```
SELECT * FROM `login`
```

Profiling [ Edit inline ] [ Edit ] [ Explain SQL ] [ Cr

Show all | Number of rows: 25

Extra options

| USER_NAME | LOGIN_TIME |
|---|---|
| prisha | 2023-01-26 20:24:15 |
| tejal | 2023-01-24 09:26:22 |

Figure 4.14: 'LOGIN' table with values

```
SELECT * FROM `categories`
```

Profiling [ Edit inline ] [ Edit ] [ Explain SQL ] [ Create PHP code ] [ Refresh ]

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

Extra options

| | INV_ID | INV_NAME | NO_OF_ITEMS | USER_NAME |
|---|---|---|---|---|
| Edit Copy Delete | 2345 | Grains & Pulses | 7 | prisha |
| Edit Copy Delete | 7645 | Edible oils | 5 | prisha |

Figure 4.15: 'CATEGORIES' table with values

```
SELECT * FROM `stock`
```

Profiling [ Edit inline ] [ Edit ] [ Explain SQL ] [ Create PHP code ] [ Refresh ]

Show all | Number of rows: 25 Filter rows: Search this table Sort

Extra options

| | STK_ID | ITEM_NAME | QUANTITY | SUP_ID | INV_ID |
|---|---|---|---|---|---|
| Edit Copy Delete | 345 | dal | 28 | 657 | 2345 |
| Edit Copy Delete | 396 | rice | 6 | 123 | 2345 |
| Edit Copy Delete | 835 | wheat | 28 | 657 | 2345 |
| Edit Copy Delete | 987 | moong dal | 30 | 456 | 2345 |

Figure 4.16: 'STOCK' table with values

```
SELECT * FROM `supplier`
```

Profiling [ Edit inline ] [ Edit ] [ Explain SQL ] [ Create PHP code ] [ Refresh ]

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

Extra options

| | SUP_ID | SUP_NAME | SUP_MOBILE | SUP_EMAIL | SUP_LOCATION |
|---|---|---|---|---|---|
| Edit Copy Delete | 123 | Rajesh | 9876765432 | rajesh@gmail.com | bangalore |
| Edit Copy Delete | 456 | suresh | 7654328763 | suresh@gmail.com | chennai |
| Edit Copy Delete | 657 | mahesh | 9873879545 | mahesh@gmail.com | chennai |

Figure 4.17: 'SUPPLIER' table with values

```
SELECT * FROM `supplies`
```

Profiling [ Edit inline ] [ Edit ] [ Explain SQL ] [ Create PHP code ] [ Refresh ]

Show all | Number of rows: 25 Filter rows: Search this table

Extra options

| | SUP_ID | STK_ID | ITEM_NAME | QUANTITY |
|---|---|---|---|---|
| Edit Copy Delete | 123 | 396 | rice | 20 |
| Edit Copy Delete | 657 | 345 | Dal | 6 |

Figure 4.18: 'SUPPLIES' table with values

## 4.4 SQL TRIGGERS AND STORED PROCEDURE

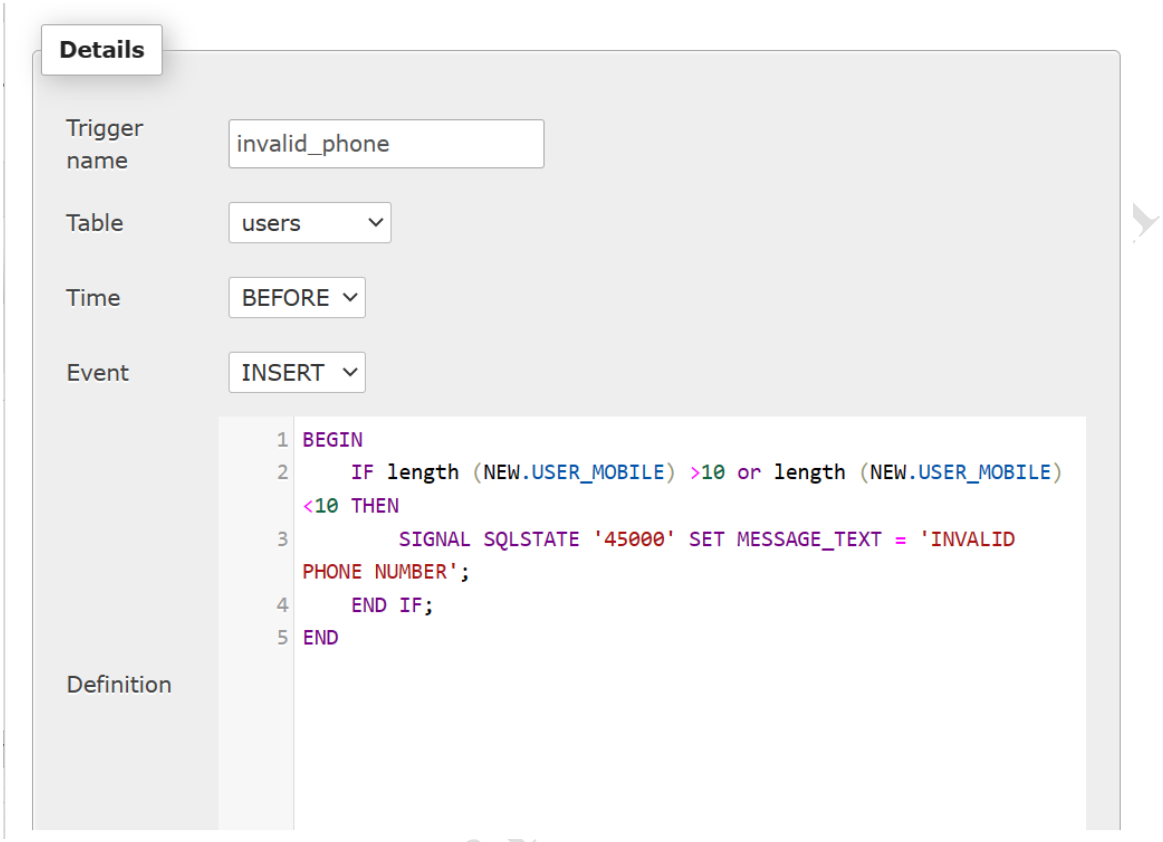



Figure 4.19: Trigger for 'USERS' table


Figure 4.20: Stored Procedure

## 4.5 DATABASE CONNECTIVITY

connection > connection.php

```php
<?php
    $host = 'localhost';
    $user = 'root';
    $password = '';
    $dbname = 'inventory_db';
    $conn = new mysqli($host, $user, $password, $dbname);

    if(!$conn){
        die("Error: Failed to connect to database");
    }
?>
```

Figure 4.21: Connection to database

PHP is a popular programming language that can be used to connect to a variety of databases, including MySQL, Oracle, and Microsoft SQL Server. To connect to a database using PHP, you can use the built-in function mysqli_connect() or PDO. The mysqli_connect() function is used to connect to a MySQL database and the PDO class is used to connect to other databases.

Once a connection has been established, you can use various functions to perform operations on the database such as querying, inserting, and updating data.

## 4.6 MODULES

In a project, the database user module is used to manage and accessing the data stored in a database. The user module provides additional functionality for managing the database itself, such as creating and modifying tables, and monitoring database performance. The modules typically require authentication and authorization to ensure that only authorized users can access and make changes to the data like update, insert and delete.

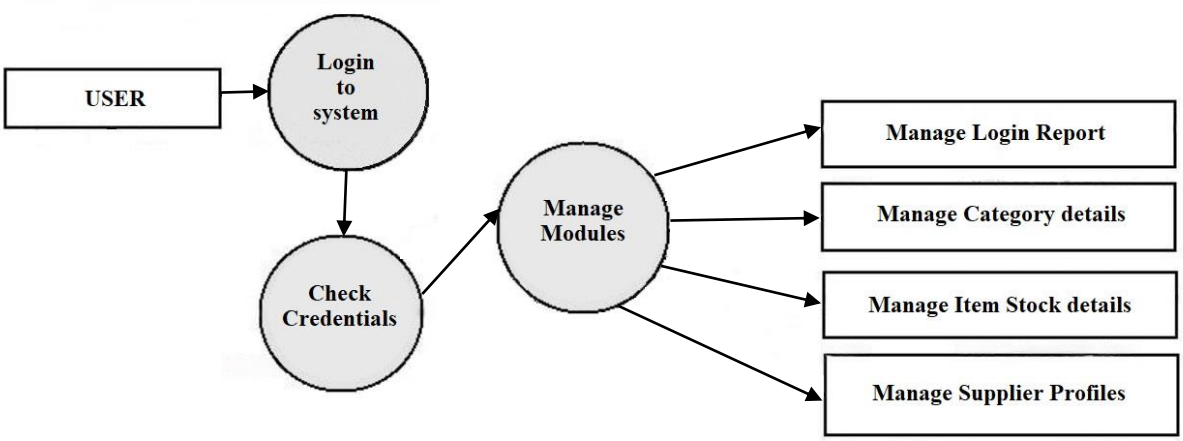



Figure 4.22: Dataflow diagram

# Chapter 5

## RESULTS

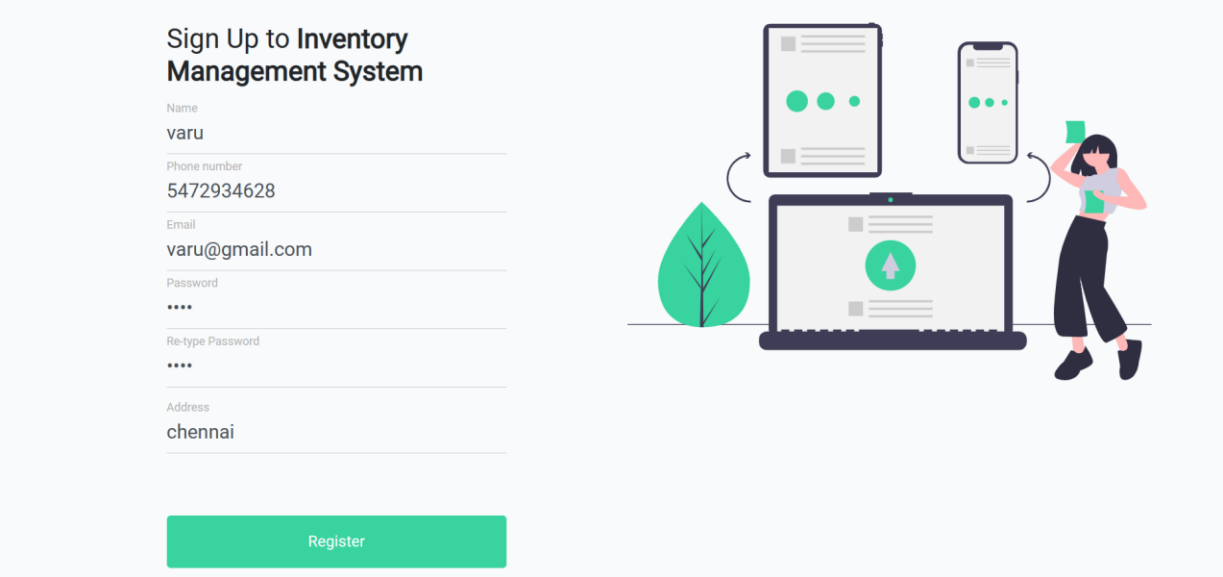



Figure 5.1: Signup page

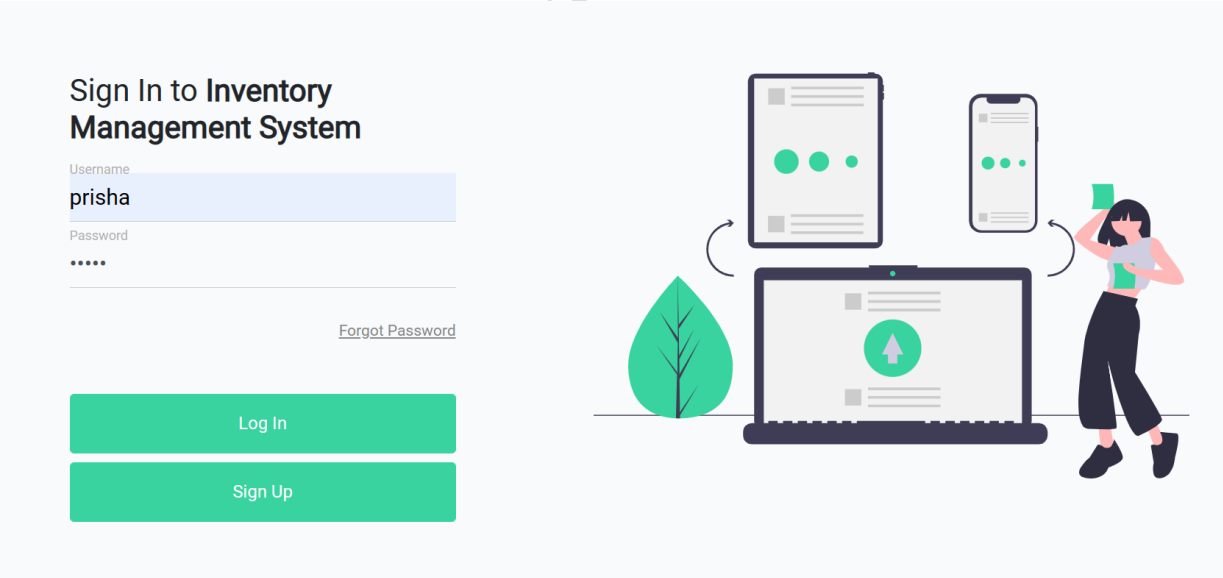



Figure 5.2: Login page

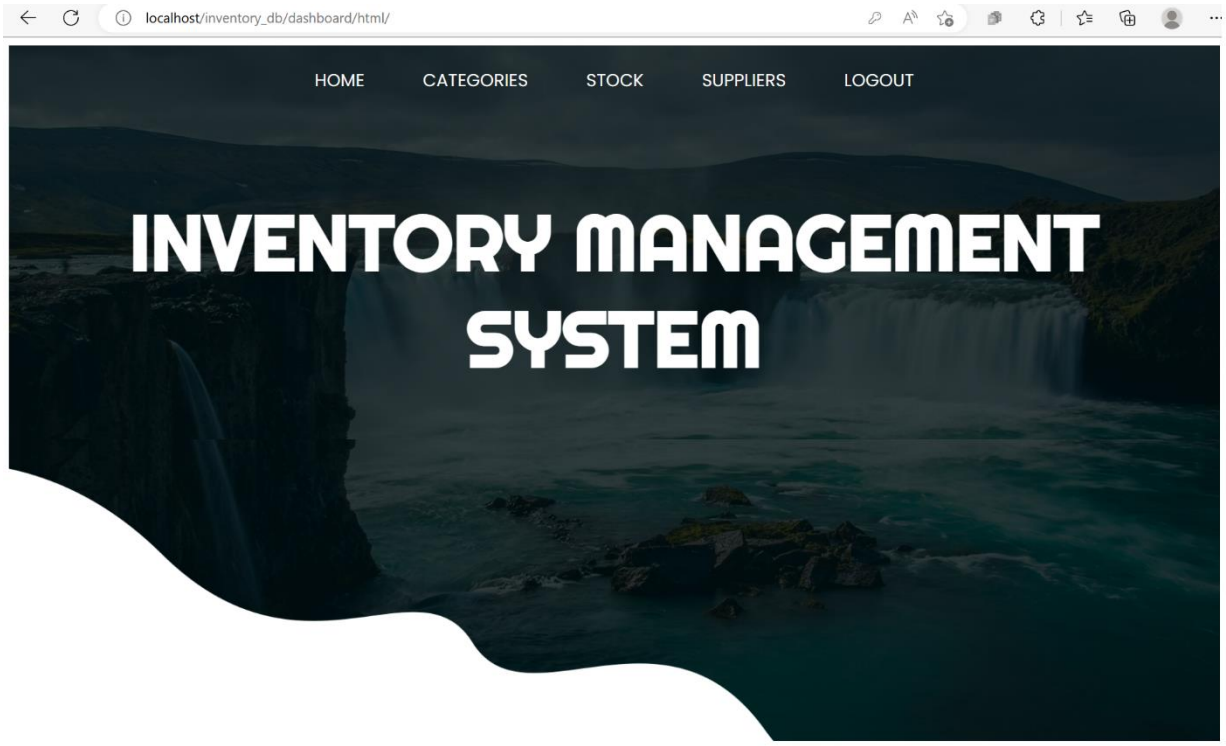



Figure 5.3: Home page

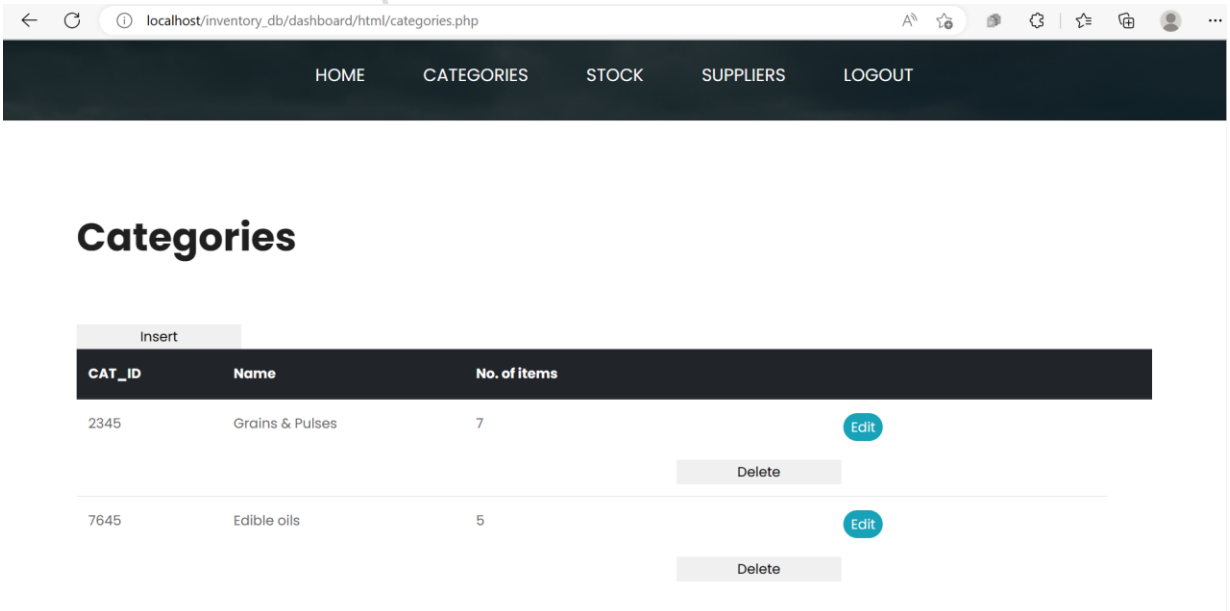



Figure 5.4: ‘CATEGORIES’ page

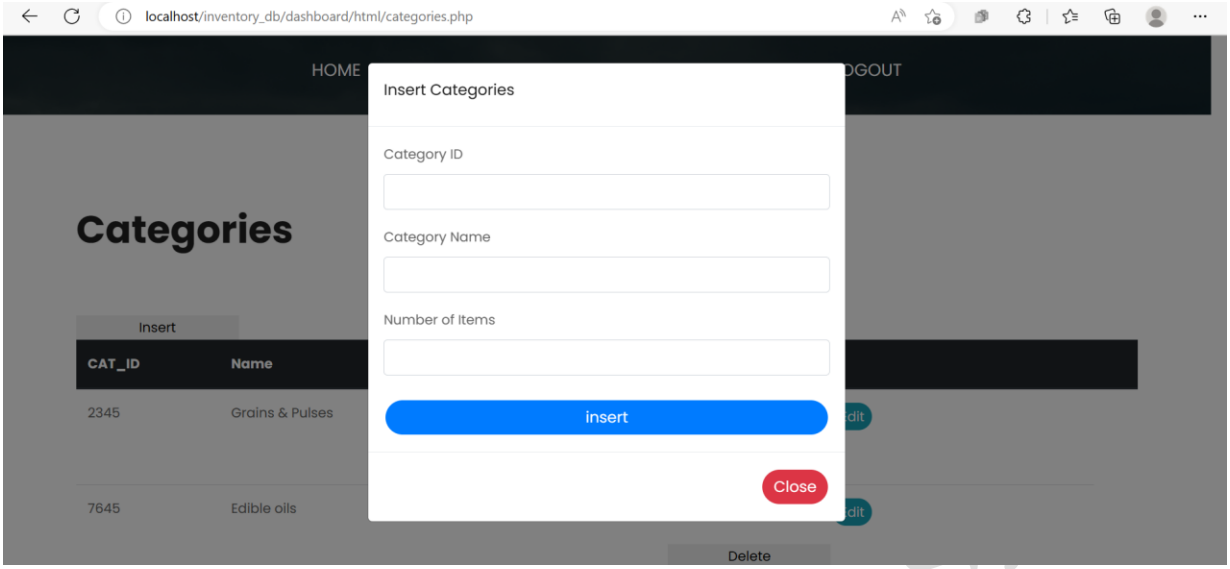



Figure 5.5: Insertion into 'CATEGORIES'

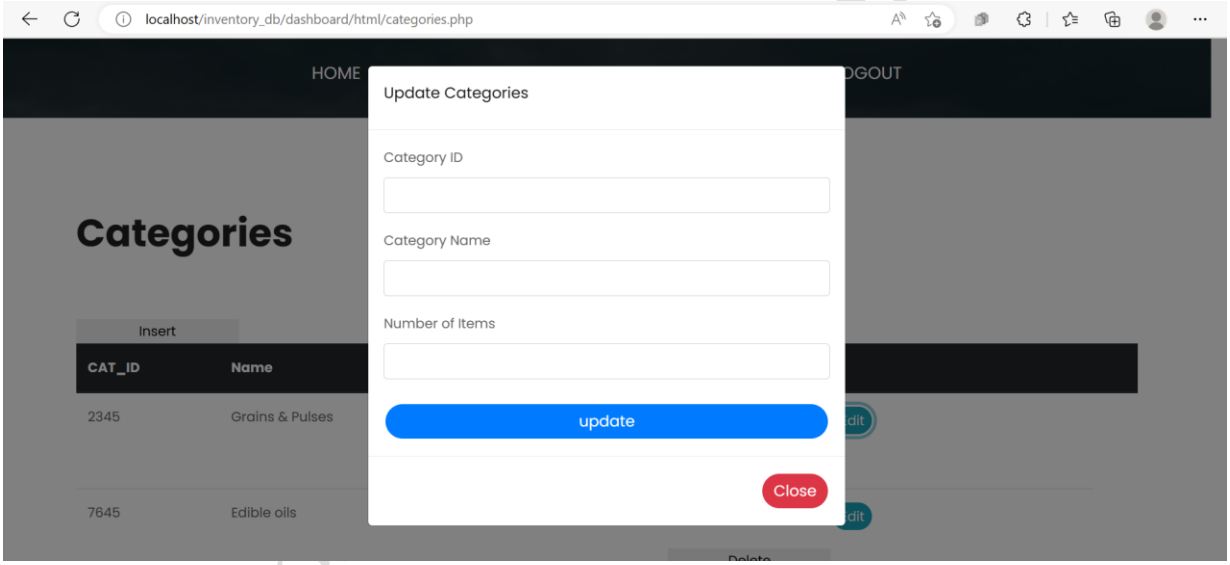



Figure 5.6: Update 'CATEGORIES' table

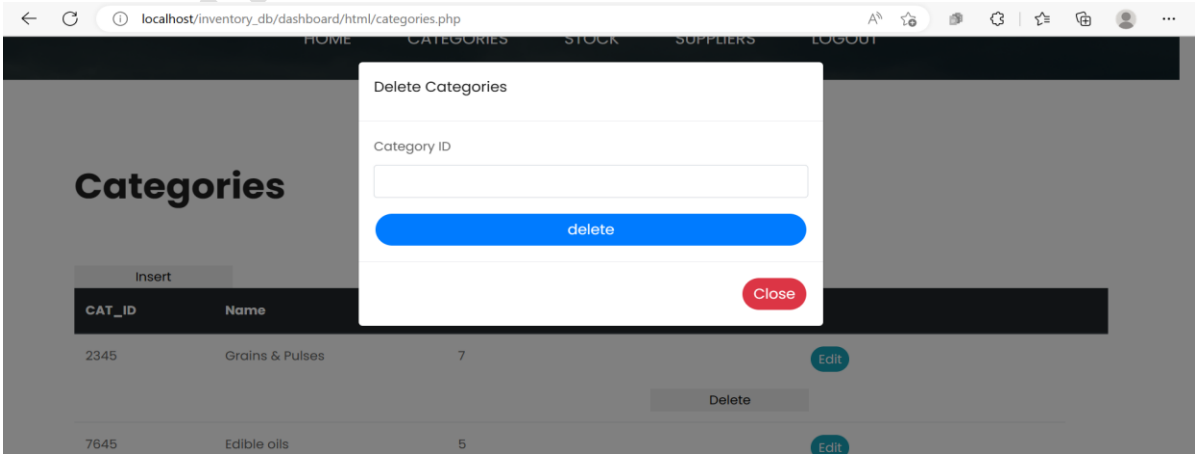



Figure 5.7: Delete from 'CATEGORIES'

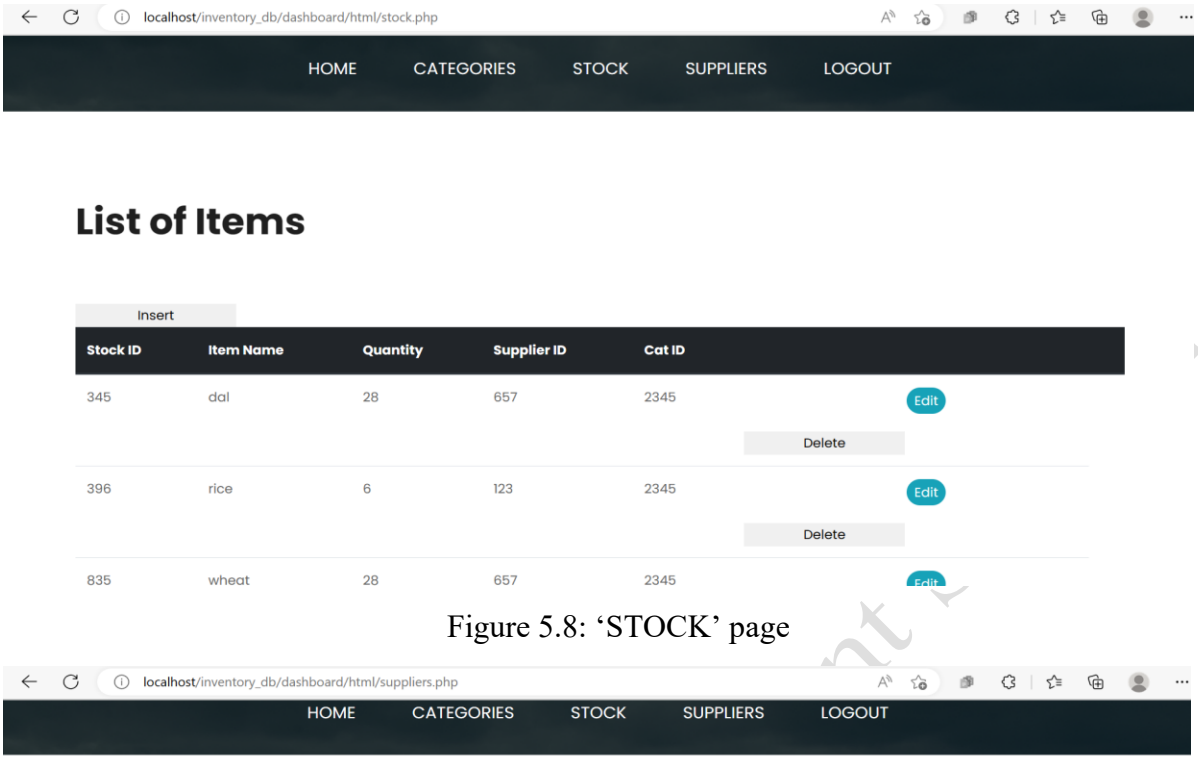



Figure 5.8: 'STOCK' page

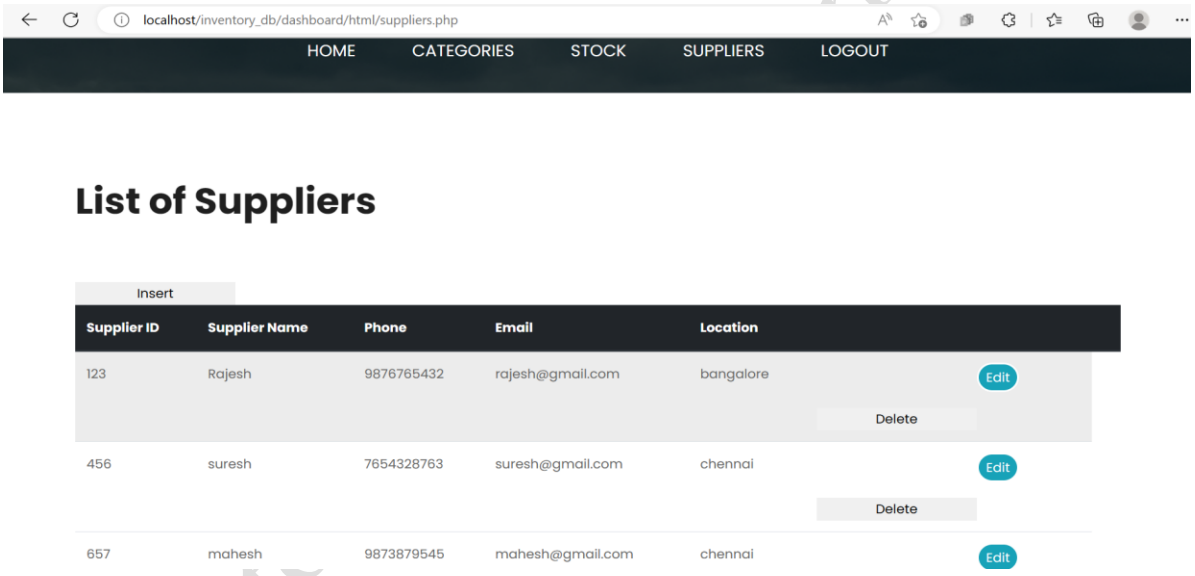



Figure 5.9: 'SUPPLIERS' pages

# Chapter 6

## CONCLUSION AND FUTURE ENHANCEMENTS

In conclusion, an inventory management system is a valuable tool for businesses of all sizes and industries. It provides detailed information on inventory levels, movement, and value. This can help businesses identify and address issues such as stockouts and overstocking, improve forecasting and planning, and identify opportunities for cost savings. However, an inventory management system is not only a tool for data collection and analysis but also a critical process that must be continuously improved.

There are several potential enhancements that could be made to an inventory management system to improve its efficiency and effectiveness. Some of these include:

1. Automation: Automating certain aspects of the inventory management process, such as tracking inventory levels and generating reports, can save time and reduce the risk of errors.
2. Real-time tracking: Implementing real-time tracking of inventory levels can help businesses quickly identify and address any issues that may arise, such as stockouts or overstocking.
3. Predictive analytics: Using predictive analytics to forecast demand for products can help businesses plan production and purchasing more effectively.
4. Integrating with other systems: Integrating the inventory management system with other systems, such as accounting or point-of-sale, can provide a more complete view of the business and help identify trends or patterns that may be affecting inventory levels.
5. Mobile and cloud-based solutions: Using mobile and cloud-based solutions can make it easier for multiple users and locations to access and update inventory information in real-time.
6. Machine Learning: Implementing machine learning algorithms, such as forecasting, optimization, and anomaly detection, can help in forecasting demand, optimizing the ordering process, and identifying potential risks and opportunities.
7. RFID and barcode scanning: RFID and barcode scanning can be used to automatically track inventory levels and movements, reducing the risk of errors and improving efficiency.

8. Automated reorder point and replenishment: Automated reorder point and replenishment systems can be used to automatically trigger purchase orders when inventory levels reach a certain threshold, reducing the risk of stockouts.

Overall, the future enhancements for inventory management could be focused on automating the process, increasing the accuracy and speed of data collection and analysis, and integrating the inventory management system with other systems to provide a more complete view of the business. By implementing these enhancements, businesses can improve the efficiency and effectiveness of their inventory management processes, ultimately leading to better decision-making, cost savings, and increased profitability.