

Abstract

E-commerce is India's fastest growing and most exciting channel for commercial transactions. The Indian e-commerce market is expected to grow to US\$200 billion by 2026 from US\$ 48.5 billion as of 2018. This growth has been triggered by increasing internet and smartphone penetration.

The ongoing digital transformation in the country is expected to increase India's total internet user base to 829 million by 2021 from 560.01 million as of September 2018. India's Internet economy is expected to double from US\$125 billion as of April 2017 to US\$ 250 billion by 2020, majorly backed by e-commerce. India's E-commerce revenue is expected to jump from US\$ 39 billion in 2017 to US\$ 120 billion in 2020, growing at an annual rate of 51%, the highest in the world.

There is a growing appetite for international brands and better-quality foreign products amongst digitally connected Indian shoppers due to rising income levels and increased awareness. Several categories including lifestyle products, consumer electronics, clothing, footwear, jewelry and accessories, health and beauty, household goods, art and collectibles, event tickets and online music are doing well for online sales.

Technology enabled innovations such as digital payments, hyper-local logistics, analytics driven customer engagement and digital advertisements have enabled the eCommerce industry in India to grow at a much faster rate.

Government initiatives such as Digital India, Skill India, Startup India and Make in India are also contributing to the growth of the eCommerce industry.

Table Of Contents

Abstract	1
Table Of Contents	2
List of Figures	4
Introduction	5
Problem Statement	5
Objectives	5
Scope	5
Literature Survey	6
2.1 Traditional File System	6
2.2 Pros and Cons of the Traditional Approach	6
2.3 Downfall of Traditional Management System	7
2.4 Introduction to the database management system	8
2.5 Indicative areas for the use of DBMS	8
2.6 Advantages of DBMS	8
2.7 Components of DBMS	9
System Requirements	10
3.1 Hardware Requirements	10
3.2 Software Requirements	10
Design	11
4.1 Requirements and Constraints	11
4.2 Entities and Attributes	12
4.3 ER Diagram	16
4.4 Relational Schema	17

Implementation	18
5.1 Technologies/Frameworks used in building the project	18
5.2 Code Snippets	19
5.2.1 Create table commands	19
5.2.2 Config- To establish connection	23
5.2.3 The Login and Sign Up pages	23
5.2.4 Triggers	29
Testing and Result	30
6.1 Testing Process	30
6.2 Testing Objectives	30
6.3 Test Cases	30
6.4 Snapshots	31
Conclusion	36
References	37

List of Figures

Fig No.	Description	Page No.
Fig 6.1	Login Page	29
Fig 6.2a	Register Page for Buyer	29
Fig 6.2b	Register Page for Seller	30
Fig 6.3	Homepage	30
Fig 6.4	Seller Dashboard	31
Fig 6.5	Order page	32
Fig 6.6a	Admin Page for Buyer	33
Fig 6.6b	Admin Page for Seller	33
Fig 6.7	Product input form	34

Chapter 1

Introduction

1.1 Problem Statement

With the advent of the Internet and technological advancement, is it much quicker and easier to do everything done traditional on the internet. Commerce is no exception in this regard. Internet users are no longer limited to sitting at desktop computers in order to do research or send emails. People are now using their laptops, tablets and even mobile phones to do more things including making purchases and even selling products via the Internet.

1.2 Objectives

Aim is to create an e-commerce portal using the bleeding edge of available technology.

The following objectives are achieved:

- i. A seller, after registering with the portal can add any product that he/she intends to sell for public listing.
- ii. A buyer can buy any product of his choice from the store, which is sold by the sellers.
- iii. Any visitor to the site can view the items sold before deciding to register on the platform.
- iv. The admin can view all the users and take action on any users disrespecting the rules of the platform

1.3 Scope

The scope of this project is to ensure the best use of computer based management system to bring the best available products to the fingertips of all users. Products from all over the world can be sold on a single platform thereby encouraging competition and ensuring the product is of the best quality. The project also ensures that any individual with a product to sell can do so, thus providing an open marketplace.

Chapter 2

Literature Survey

2.1 Traditional File System

In the early days of computing data management and storage was a very new concept for organizations. The traditional approach to data handling offered a lot of the convenience of the manual approach to business processes (e.g. handwritten invoices and account statements, etc.) as well as the benefits of storing data electronically.

The traditional approach usually consisted of custom built a data processing and computer information systems tailored for a specific business functions. An accounting department would have their own information system tailored to their needs, where the sales department would have an entire separate system for their needs.

Initially, these separate systems were very simple to set up as they mostly mirrored the business process that departments had been doing for years but allows them to do things faster with less work. However, once our systems were in use for so long, that became very difficult for individual departments to manage and rely on their data because there was no reliable system in place to enforce the data standards for management.

Separate information systems for each business function also lead to conflicts of interest within the company. These separations of data also lead to unnecessary redundancy and high rate of unreliable and inconsistent data.

2.2 Pros and Cons of the Traditional Approach

Pros

i. Simple

- o Matched existing business processes and functions.
- o Companies were not as interested in funding complicated Information Systems.

ii. Initially low cost

- o Early computing was not viewed as beneficial for large funding.
- o Systems are designed to be cheap in order to save on cost.

Cons**i. Separated ownership**

- o Business functions had a high sense of data ownership.
- o Department unwilling to share data for fear of minimising their superiority.

ii. Unmanaged redundancy

- o Multiple instances of the same data appeared throughout various files, systems and databases.
- o Information updated in one place was not replicated to the other locations.
- o Disk space was very expensive and redundancy had a big impact on storage.

iii. Data inconsistency

- o Redundant data stored in various locations was usually never stored the same way.
- o Formatting was not centrally managed.

iv. High cost in the long run

- o Hiring data processes for each department was very expensive and each position of typically working on the same thing just for a different area.

2.3 Downfall of Traditional Management System

Conceived in a relatively centralized era when software was developed in static environment, legacy database architectures failed to support an increasingly mobile world when applications are accessed anytime, anywhere.

However , legacy database Technologies fall short in serving the needs of today's distributed and cloud environment for the following reasons:

- o Inadequate failure capabilities
- o Latency issues
- o Insufficient provisions during peak demands
- o Lack of high availability at all times
- o Increasing operational costs

For all of these reasons, traditional databases are unable to deliver results in a rapidly growing environment where the workload is geographically distributed across heterogeneous data centres. Upgrading two or more distributed data model is costly and complicated and your DBAs can't just sit back and give up on this situation. Hence due to these various reasons, the downfall of the traditional system was inevitable.

2.4 Introduction to the database management system

A Database Management System (DBMS) refers to the technology for creating and managing databases. Basically, a DBMS is a software tool to organise (create, retrieve, update and manage) data in the database.

The main aim of DBMS is to supply a way to store and retrieve database information that is both convenient and efficient. By data, we mean known facts that can be recorded and that have embedded meaning. Normally people use software such as DBASE IV or V, Microsoft ACCESS or EXCEL to store data in the form of database. A datum is a unit of data.

Database systems are meant to handle large collection of information. Management of data involves both defining structures for storage of Information and providing mechanism that can do the manipulation of the store information. Moreover, the database systems must ensure the safety of the information stored, despite system crashes or attempts at unauthorised access.

2.5 Indicative areas for the use of DBMS

- Airlines: reservations, schedules etc.
- Telecom: calls made, customer details, network usage etc.
- Universities: registration, results, grade etc.
- Sales: products, purchases, customers etc.
- Banking: all transactions etc.

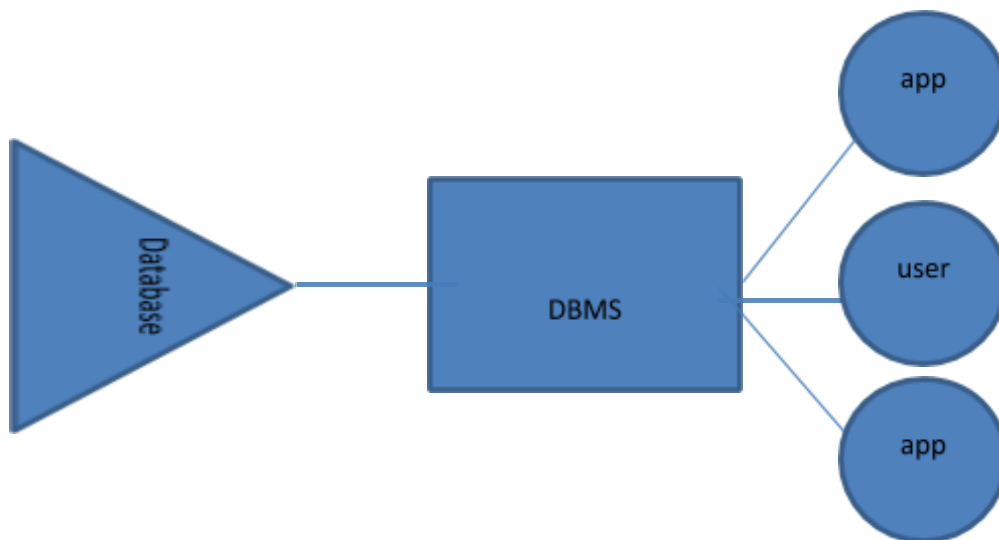
2.6 Advantages of DBMS

A Database Management System has many advantages over the traditional file system used in the earlier days, such as:

- **Data Independence:** Application programs should be as free or independent as possible from the details of data representation and storage device can supply and abstract of the date of insulating application code from such facts.

- **Inefficient Data Access:** DBMS utilizes the mixture of sophisticated concepts and techniques for storing and retrieving data competently and this feature becomes important in cases where the data is stored on external storage devices.
- **Data Administration:** When several users share the data, integrating the administration of data can offer major implements. Experienced professional understand the nature of data being managed and can be responsible for organizing the data representation to reduce redundancy and make the data to retrieve efficiently.

2.7 Components of DBMS



- **Users:** Users may be of any kind, such as database administrators, system developers or database users.
- **Database application:** Database application may be Departmental, Personal, Organizational and /or Internal
- **DBMS:** Software that allows users to create and manipulate database access.
- **Database:** A Collection of logical data as a single unit.

Chapter 3

System Requirements

3.1 Hardware Requirements

- i. Processor: Pentium 4 or above
- ii. Ram: 2GB or more
- iii. Hard Disk: 500MB or more

3.2 Software Requirements

Technologies used:

- i. Front end: HTML,CSS,REACT JS
- ii. Connection/Controller: EXPRESS JS
- iii. Back-end/Database: MySQL

Software:

- i. Text Editor: VS Code
- ii. Server: Apache(on XAMPP)
- iii. Operating System: Windows 10
- iv. Database Support: MySQL 5.7
- v. Back-end: Express JS

Chapter 4

Design

In order to design a web site, the relational database must be designed first. Conceptual design can be divided into two parts: The data model and the process model. The data model focuses on what data should be stored in the database while the process model deals with how the data is processed. To put this in the context of the relational database, the data model is used to design the relational tables. The process model is used to design the queries that will access and perform operations on those tables.

4.1 Requirements and Constraints

- i. Feature to create a new user account/log into existing account.
- ii. Feature to access information about products.
- iii. Feature to buy various category related products.
- iv. Feature to add products to orders.
- v. Feature to remove an already added item.
- vi. Feature to view details about the product according to the category selected.
- vii. Feature to add new products, new seller, and new buyer.
- viii. Update the products tables with new information.
- ix. Feature to remove products, sellers and buyers.

4.2 Entities and Attributes

Name	Description
Login	Contains login credentials of all users
Buyer	Contains the buyer credentials
Seller	Contains the seller credentials
Products	Contains the product information
Categories	Contains the list of categories available
Orders	Contains the orders history of all the buyers

Table 4.1 Table-names and their description

1. Login

Name	Description	Type
Lid (<i>Primary key</i>)	Contains the login id	int(11)
Password	Contains the hashed password	varchar(70)
Type	Contains the type of the user logged in. (buyer,seller,admin)	char(10)

Table 4.2 Login table

2. Buyer

Name	Description	Type
Bid (<i>Primary key</i>)	Contains the buyer id	int(11)
Name	Contains the name of the buyer	varchar(20)
Location	Contains the location of the buyer	varchar(50)

Table 4.3 Buyer table

3. Seller

Name	Description	Type
Sid (<i>Primary key</i>)	Contains the seller id	int(11)
Name	Contains the name of the seller	varchar(20)
Contact	Contains the contact details of the seller	int(11)

Table 4.4 Seller table

4. Products

Name	Description	Type
Pid (<i>Primary key</i>)	Contains the product id	int(11)
Sid (<i>Primary key</i>)	Contains the seller id	int(11)
Cid	Contains the category id	int(11)
Pname	Contains the product name	varchar(50)
Description	Contains brief description of the product	varchar(300)
Price	Contains the price of the product	int(11)
Quantity	Contains the available quantity available	int(11)

Table 4.5 Products table

5. Categories

Name	Description	Type
Cid	Contains the category id	int(11)
Category	Contains the category of the product	varchar(30)

Table 4.6 Categories table

6. Orders

Name	Description	Type
Oid (<i>Primary key</i>)	Contains the order id	int(20)
Bid	Contains the buyer id	int(11)
Sid	Contains the seller id	int(11)
Pid	Contains the product id	int(11)
Date	Contains the date at which the product was ordered	timestamp
Price	Contains the price of the product	int(11)

Table 4.7 Orders table

4.3 ER Diagram

An **entity relationship diagram (ERD)** shows the relationships of entity sets stored in a database. An entity in this context is an object, a component of data. An entity set is a collection of similar entities. These entities can have attributes that define its properties. The fig. 4.1 shows the ERD for the database.

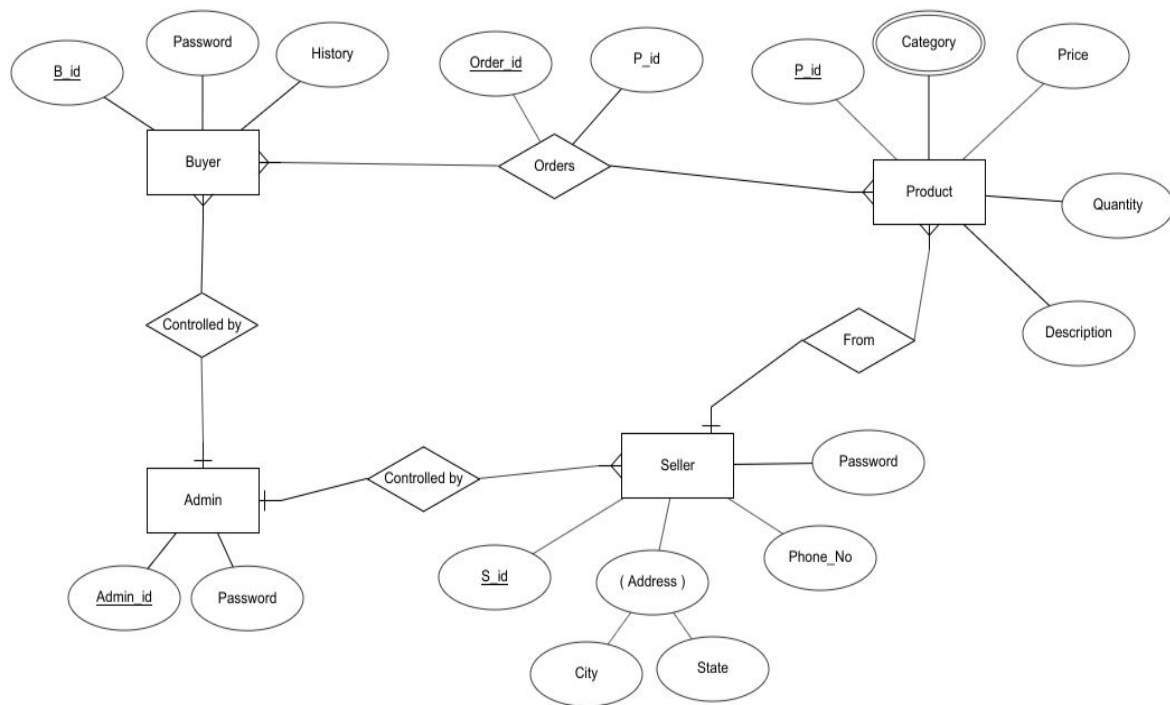


Fig 4.1 ER Diagram

4.4 Relational Schema

A relational schema shows references among fields in the database. When a primary key is referenced in another table in the database, it is called a foreign key. This is denoted by an arrow with the head pointing to the referenced key attribute. The fig. 4.2 shows the schema diagram for the database.

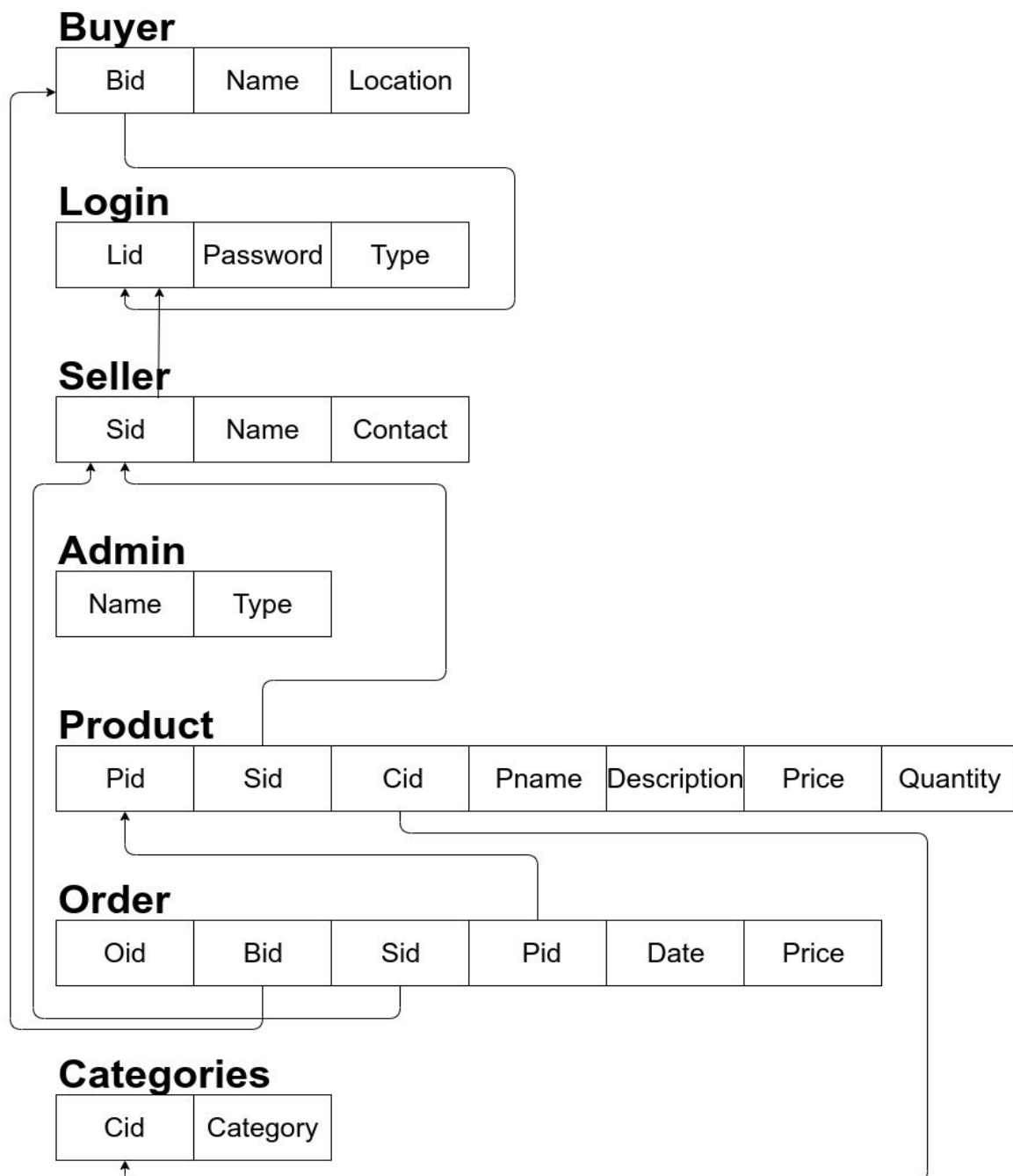


Fig 4.2 Schema Diagram

Chapter 5

Implementation

5.1 Technologies/Frameworks used in building the project

i. HTML5

HTML 5 is a mark-up language used for structuring and presenting content on the World Wide Web. It is the fifth and current major version of the HTML standard. HTML5 includes detailed processing models to encourage more interoperable implementations; it extends, improves and rationalises the mark-up available for documents and introduces mark-up and application programming interfaces (APIs) for Complex Web applications.

ii. React JS

React JS (also known as React.js or React) is a JavaScript library for building user interfaces. It is maintained by Facebook and a community of individual developers and companies.

React can be used as a base in the development of single-page or mobile applications, as it is optimal for fetching rapidly changing data that needs to be recorded. However, fetching data is only the beginning of what happens on a web page, which is why complex React applications usually require the use of additional libraries for state management, routing, and interaction with an API: Redux, React Router and axios are examples of such libraries.

iii. Express JS

Express.js, or simply Express, is a web application framework for Node.js, released as free and open-source software under the MIT License. It is designed for building web applications and APIs. It has been called the de facto standard server framework for Node.js.

The original author, TJ Holowaychuk, described it as a Sinatra-inspired server, meaning that it is relatively minimal with many features available as plugins. Express is the back-end component of the MEAN stack, together with the SQL database software and ExpressJS front-end framework.

iv. SQL (Structured Query Language)

MySQL Database Management System is used to store the user and product information which is stored in the secondary storage device and can be altered anytime. Normalized and efficient schema is used to avoid redundancy and inconsistency. The data is updated in real time.

v. CSS

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a mark-up language like HTML5.

5.2 Code Snippets

5.2.1 Create table commands

i. Login

```
CREATE TABLE `login` (  
  `Lid` int(11) NOT NULL,  
  `Password` varchar(70) NOT NULL,  
  `Type` char(10) NOT NULL  
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
ALTER TABLE `login`  
  ADD PRIMARY KEY (`Lid`);
```

ii. Buyer

```
CREATE TABLE `buyer` (  
  `Bid` int(11) NOT NULL,  
  `Name` varchar(20) NOT NULL,  
  `Location` varchar(50) NOT NULL  
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
ALTER TABLE `buyer`
```

```
ADD PRIMARY KEY (`Bid`);
```

```
ALTER TABLE `buyer`
```

```
ADD CONSTRAINT `buyer_ibfk_1` FOREIGN KEY (`Bid`) REFERENCES  
`login` (`Lid`) ON DELETE CASCADE ON UPDATE CASCADE;
```

iii. Seller

```
CREATE TABLE `seller` (  
  `Sid` int(11) NOT NULL,  
  `Name` varchar(20) NOT NULL,  
  `Contact` int(11) NOT NULL  
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
ALTER TABLE `seller`  
ADD PRIMARY KEY (`Sid`);
```

```
ALTER TABLE `seller`  
ADD CONSTRAINT `seller_ibfk_1` FOREIGN KEY (`Sid`) REFERENCES  
`login` (`Lid`) ON DELETE CASCADE ON UPDATE CASCADE;
```

iv. Product

```
CREATE TABLE `products` (  
  `Pid` int(11) NOT NULL,  
  `Sid` int(11) NOT NULL,  
  `Cid` int(11) NOT NULL,  
  `Pname` varchar(50) NOT NULL,  
  `Description` varchar(300) NOT NULL,  
  `Price` int(11) NOT NULL,  
  `Quantity` int(11) NOT NULL  
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
ALTER TABLE `products`  
ADD PRIMARY KEY (`Pid`,`Sid`) USING BTREE,  
ADD KEY `Sid` (`Sid`),  
ADD KEY `Cid` (`Cid`);
```

```
ALTER TABLE `products`
```

```
MODIFY `Pid` int(11) NOT NULL AUTO_INCREMENT,  
AUTO_INCREMENT=13;
```

```
ALTER TABLE `products`  
ADD CONSTRAINT `products_ibfk_1` FOREIGN KEY (`Sid`) REFERENCES  
`seller` (`Sid`) ON DELETE CASCADE ON UPDATE CASCADE,  
ADD CONSTRAINT `products_ibfk_2` FOREIGN KEY (`Cid`) REFERENCES  
`categories` (`Cid`) ON DELETE CASCADE ON UPDATE CASCADE;
```

v. Categories

```
CREATE TABLE `categories` (  
  `Cid` int(11) NOT NULL,  
  `Category` varchar(30) NOT NULL  
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
ALTER TABLE `categories`  
ADD PRIMARY KEY (`Cid`);
```

```
ALTER TABLE `categories`  
MODIFY `Cid` int(11) NOT NULL AUTO_INCREMENT,  
AUTO_INCREMENT=203;
```

vi. Orders

```
CREATE TABLE `orders` (  
  `Oid` int(20) NOT NULL,  
  `Bid` int(11) NOT NULL,  
  `Sid` int(11) NOT NULL,  
  `Pid` int(11) NOT NULL,  
  `Date` timestamp NOT NULL DEFAULT current_timestamp(),  
  `Price` int(11) NOT NULL  
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
ALTER TABLE `orders`  
ADD PRIMARY KEY (`Oid`),  
ADD KEY `Bid` (`Bid`),  
ADD KEY `Sid` (`Sid`),  
ADD KEY `Pid` (`Pid`);
```

```
ALTER TABLE `orders`  
    MODIFY `Oid` int(20) NOT NULL AUTO_INCREMENT,  
    AUTO_INCREMENT=8;
```

```
ALTER TABLE `orders`  
    ADD CONSTRAINT `orders_ibfk_1` FOREIGN KEY (`Bid`) REFERENCES  
    `buyer` (`Bid`) ON DELETE CASCADE ON UPDATE CASCADE,  
    ADD CONSTRAINT `orders_ibfk_2` FOREIGN KEY (`Sid`) REFERENCES  
    `seller` (`Sid`) ON DELETE NO ACTION ON UPDATE CASCADE,  
    ADD CONSTRAINT `orders_ibfk_3` FOREIGN KEY (`Pid`) REFERENCES  
    `products` (`Pid`) ON DELETE CASCADE ON UPDATE CASCADE;
```

5.2.2 Config- To establish connection

This is a code snippet to show how Express JS is used to connect to the local MySQL database using the localhost server.

- Start session by creating javascript variables to hold hostname,username and password
- Using “mysql.createPool” we establish connection with javascript variables as parameters

```
let pool = mysql.createPool({  
    connectionLimit: 10,  
    host: "localhost",  
    user: "aa",  
    password: "super",  
    database: "aa-dbms",  
    multipleStatements: true  
});
```

5.2.3 The Login and Sign Up pages

Login page validation snippet

- i. Login fields are filled and “Login” is clicked and data is redirected to /api/login.
- ii. jsonParser is executed to decode the login credentials sent from client.
- iii. The parsed data is decrypted using blow-fish decryption module from bcrypt
- iv. Login fields are checked with the one in database
- v. Corresponding result is set to the client

```
app.post("/api/login", jsonParser, (req, res) => {  
    let requestData = req.body;  
    pool.getConnection((err, connection) => {  
        if (err) {  
            console.log("Error in getting connection");  
        } else {  
            connection.query("SELECT * FROM login", (error, result, fields) => {  
                connection.release();  
                let i = 0;
```

```
        let valid = false;
        for (i = 0; i < result.length; i++) {
            if (result[i].Lid == requestData.userid) {
                valid = true;
                break;
            }
        }
        if (valid == true) {
            bcrypt.compare(requestData.password,result[i].Password)
                .then(passRes => {
                    valid = passRes;
                    let responseObj = {
                        valid: valid,
                        type: result[i].Type
                    };
                    res.json(responseObj);
                });
        } else {
            res.json(valid);
        }
    });
}
```


Sign Up page snippet (Insert into DB)

- i. The user fills the fields and the data is posted onto /api/register.
- ii. Using “INSERT INTO users VALUES(‘&username’, ‘&password’, ‘&type’)” the details is inserted into the database by the connection.query.
- iii. It redirects to back to login once the user account is created.

```
app.post("/api/register", jsonParser, (req, res) => {  
    let requestData = req.body;  
    var addToLogin = `insert into login values(?,?,?)`;   
    console.log(requestData);  
  
    pool.getConnection((err, connection) => {  
        if (err) {  
            console.log("Error in getting connection");  
        } else {  
            bcrypt.hash(requestData.password, saltRounds).then(hash => {  
                let loginValues = [requestData.userid, hash,  
                                    requestData.accType];  
                connection.query(  
                    addToLogin,  
                    loginValues,  
                    (error, result, fields) => {  
                        console.log(error);  
                    }  
                );  
  
                if (requestData.accType == "buyer") {  
                    let addToBuyer = `insert into buyer values(?,?,?)`;   
                    let buyerValues = [
```

```
        requestData.userid,
        requestData.name,
        requestData.city + ", " + requestData.state
    ];
    connection.query(
        addToBuyer,
        buyerValues,
        (error, result, fields) => {
            if (error == null) res.json(true);
            else res.json(false);
        }
    );
} else if (requestData.accType == "seller") {
    let addToSeller = `insert into seller values(?,?,?)`;
    let sellerValues = [
        requestData.userid,
        requestData.name,
        requestData.phone
    ];
    connection.query(
        addToSeller,
        sellerValues,
        (error, result, fields) => {
            if (error === null) res.json(true);
            else res.json(false);
        }
    );
}
```

```
        }  
    });  
}  
connection.release();  
});  
});
```

Buyer order snippet

```
app.post("/api/orders", jsonParser, (req, res) => {  
    let orders = req.body;  
    // console.log(orders);  
    pool.getConnection((error, connection) => {  
        if (error) {  
            console.log("Error in getting connection");  
        } else {  
            let ordervalues = [orders.bid, orders.sid, orders.pid, orders.price];  
            connection.query(  
                `insert into orders values(NULL,?,?,?,NULL,?)`,  
                ordervalues,  
                (error,result,fields) => {  
                    // console.log(error)  
                    console.log(result)  
                    res.send(result)  
                }  
            );  
        }  
    })  
    connection.release();  
})
```

```
});  
  
});
```

Admin delete functionality snippet

```
app.post("/api/admin/del",jsonParser,(req,res)=>{  
    let requestData = req.body;  
    pool.getConnection((err, connection) => {  
        if (err) {  
            console.log("Error in getting connection");  
        } else {  
            if(requestData.type == "buyer"){  
                let delQuery = `DELETE FROM login WHERE Lid=(SELECT  
Bid from buyer where name = ?)`;  
                connection.query(  
                    delQuery,  
                    [requestData.name],  
                    (error, result, fields) => {  
                        connection.release();  
                        console.log("Error = " + error);  
                        res.send(result);  
                        console.log(result)  
                        console.log(fields)  
                    }  
                );  
            }  
            else if(requestData.type == "seller"){  
                let delQuery = `DELETE FROM login WHERE Lid=(SELECT  
Sid from seller where name = ?)`;  
                connection.query(  
                    delQuery,  
                    [requestData.name],  
                    (error, result, fields) => {
```

```
        connection.release();
        console.log("Error = " + error);
        res.send(result);
        console.log(result)
        console.log(fields)
    }
    );
}
}
});
console.log(req.body)
});
```

5.2.4 Triggers

The “reduceQuant” is fetched when the buyer orders a product using the Buy Product form from the database and is stored in the logs table, using Triggers. “reduceQuant” decreases the quantity of the product when a user orders that product.

```
CREATE TRIGGER `reduceQuant` AFTER INSERT ON `orders`
FOR EACH ROW UPDATE products set Quantity=( SELECT quantity-1 from
products where pid=new.pid) where pid=new.pid
```

Chapter 6

Testing and Result

This chapter gives an outline of all testing methods that are carried out to get a bug free system. It provides a way to check the functionality of components sub-assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

6.1 Testing Process

Testing is an integral part of software development. Testing process certifies whether the product that is developed compiles with the standards that it was designed to. Testing process involves building of test cases against which the product has to be tested.

6.2 Testing Objectives

The main objectives of testing process are as follows.

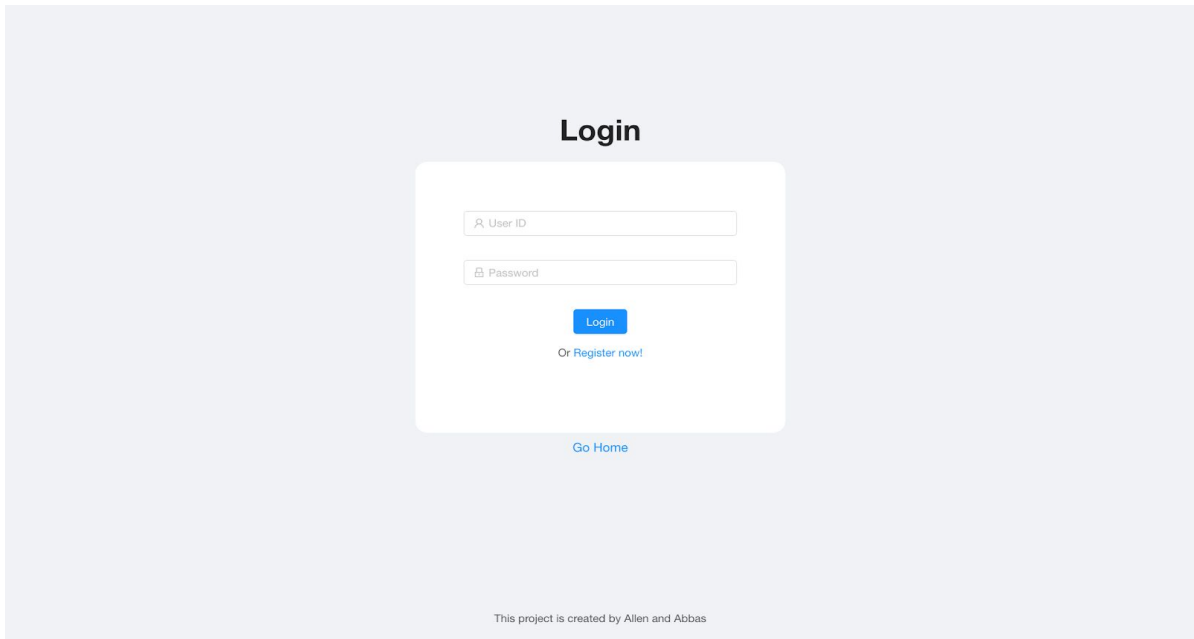
1. Testing is a process of executing a program with the intent of finding an error.
2. A good test case is one that has a high probability of finding undiscovered error.

6.3 Test Cases

SL NO	TEST RESULTS	EXPECTED RESULTS	OBSERVED RESULTS	REMARKS
1	Insert a record	New tuple should be inserted	Query OK 1 row affected or inserted	PASS
2	Search a record	Display the Record	Return the requested record	TRUE
3	Update a record	Update the particular record in the table	Update successful	PASS
4	Create trigger	Trigger Created	Query OK Trigger created	PASS
5	Delete a record	Tuple removed from database	Query OK record deleted	PASS

6.4 Snapshots

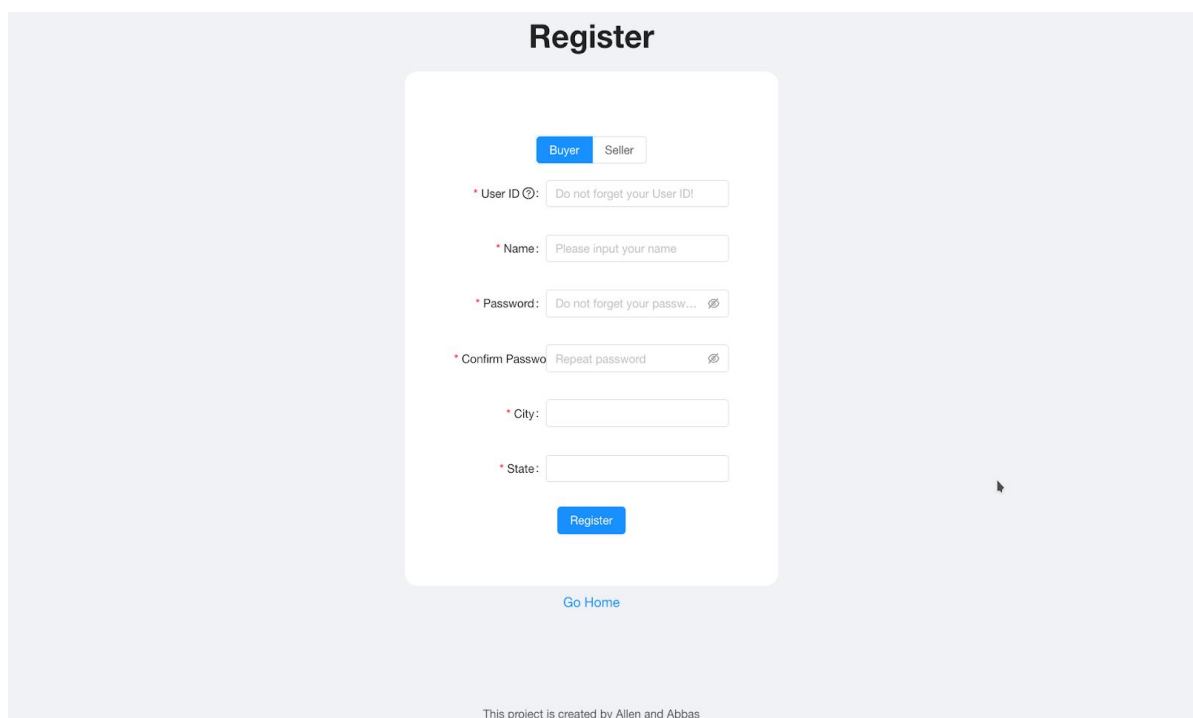
1. Login Page



The screenshot shows a login page with a central white card on a light blue background. The card is titled "Login" in bold black text. It contains two input fields: "User ID" with a magnifying glass icon and "Password" with a key icon. Below these fields is a blue "Login" button. Under the button is a link "Or Register now!". At the bottom of the card is a blue "Go Home" link. At the very bottom of the page, below the card, is the text "This project is created by Allen and Abbas".

Fig 6.1 Login Page

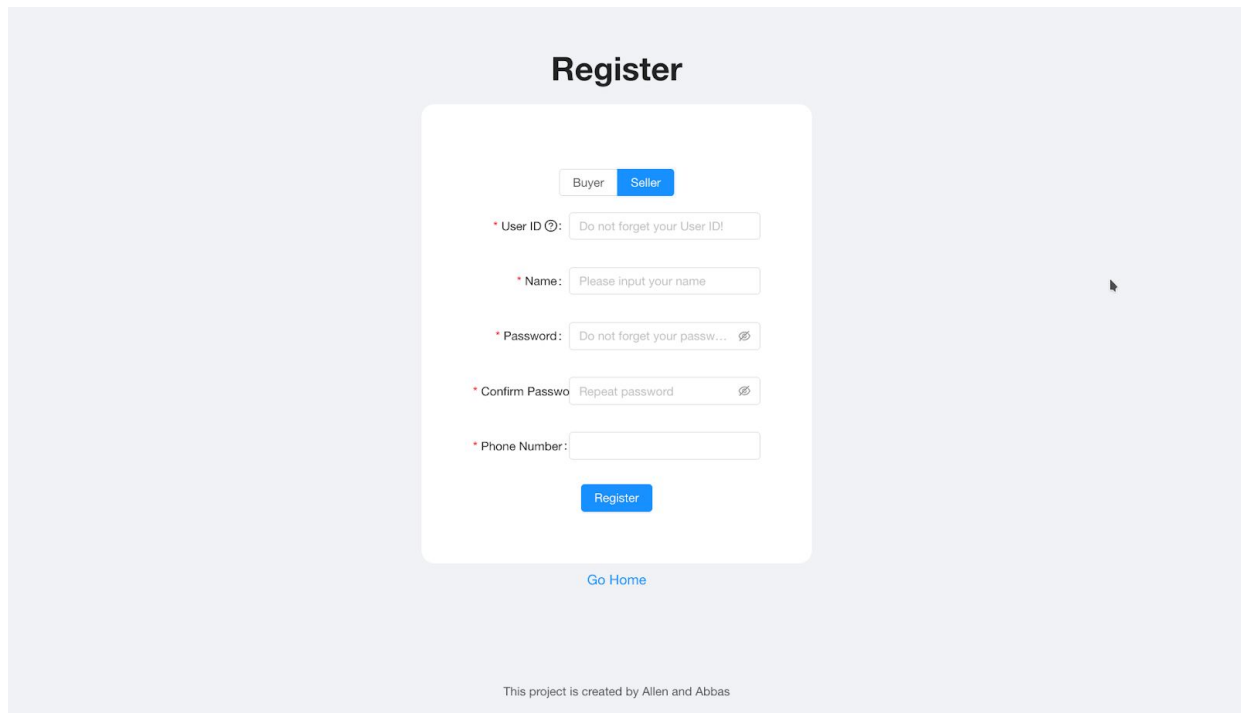
2. Register-Buyer



The screenshot shows a register page for a buyer. It features a central white card on a light blue background. The card is titled "Register" in bold black text. At the top of the card are two tabs: "Buyer" (selected) and "Seller". Below the tabs are several input fields, each with a red asterisk indicating a required field: "User ID" with a hint "Do not forget your User ID!", "Name" with a hint "Please input your name", "Password" with a hint "Do not forget your passw..." and a toggle icon, "Confirm Passwo" with a hint "Repeat password" and a toggle icon, "City", and "State". At the bottom of the card is a blue "Register" button. Below the card is a blue "Go Home" link. At the very bottom of the page is the text "This project is created by Allen and Abbas".

Fig 6.2a Register Page for Buyer

3. Register-Seller



The image shows a 'Register' form for a seller. At the top, there are two tabs: 'Buyer' and 'Seller', with 'Seller' being the active tab. The form contains the following fields:

- * User ID: A text input field with a hint 'Do not forget your User ID!'.
- * Name: A text input field with a hint 'Please input your name'.
- * Password: A text input field with a hint 'Do not forget your passw...' and a toggle icon for password visibility.
- * Confirm Password: A text input field with a hint 'Repeat password' and a toggle icon for password visibility.
- * Phone Number: A text input field.

Below the fields is a blue 'Register' button. At the bottom of the form area is a blue link 'Go Home'. At the very bottom of the page, there is a small text line: 'This project is created by Allen and Abbas'.

Fig 6.2b Register Page for Seller

4. Home Page

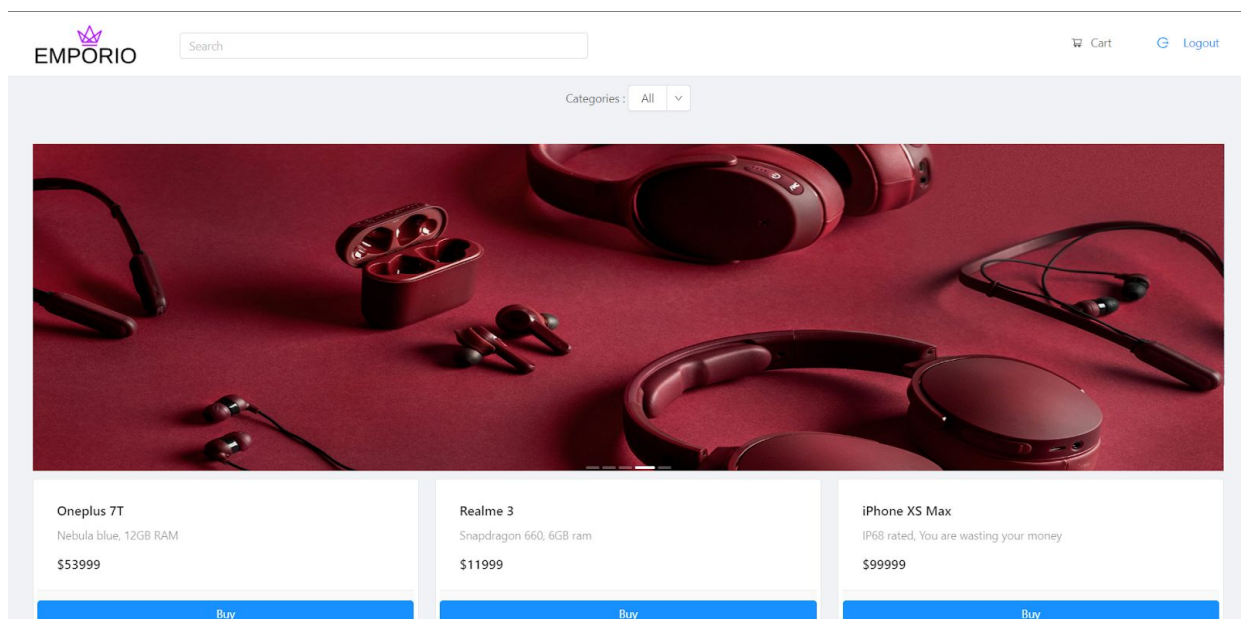
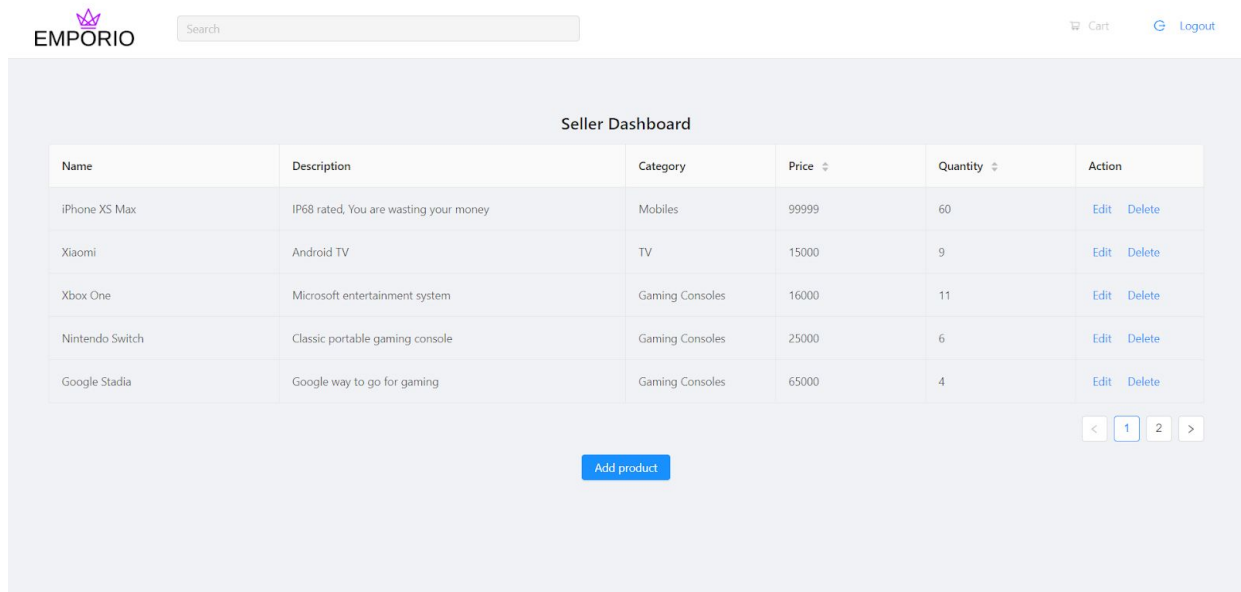


Fig 6.3 Home Page

5. Seller Dashboard



EMPORIO

Search

Cart Logout

Seller Dashboard

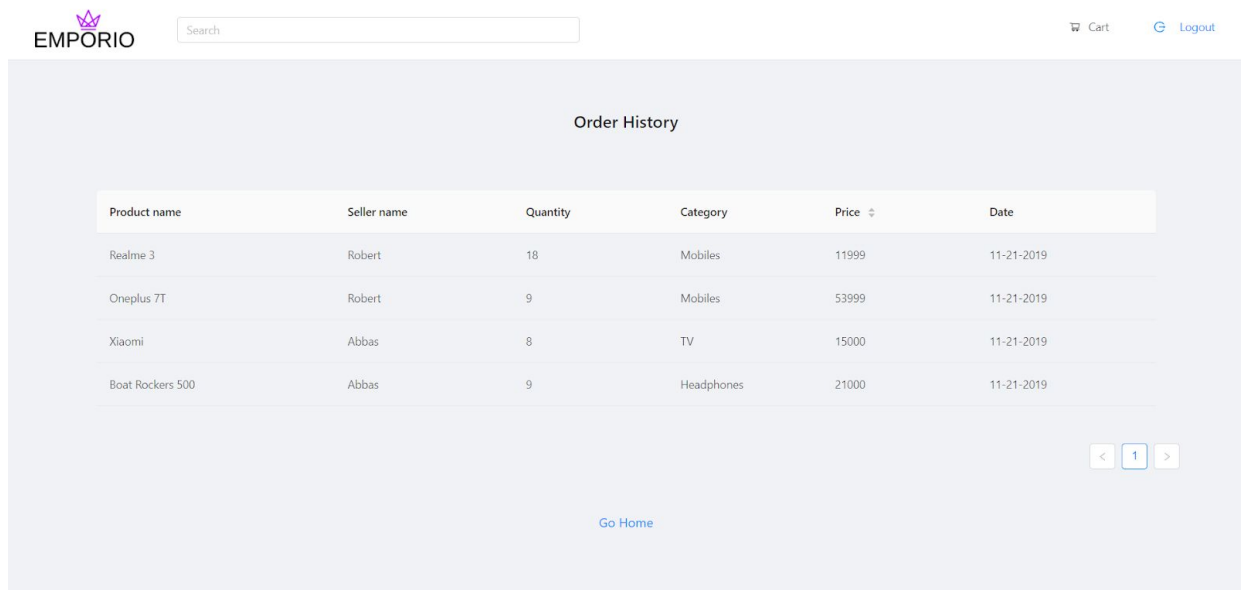
Name	Description	Category	Price	Quantity	Action
iPhone XS Max	IP68 rated, You are wasting your money	Mobiles	99999	60	Edit Delete
Xiaomi	Android TV	TV	15000	9	Edit Delete
Xbox One	Microsoft entertainment system	Gaming Consoles	16000	11	Edit Delete
Nintendo Switch	Classic portable gaming console	Gaming Consoles	25000	6	Edit Delete
Google Stadia	Google way to go for gaming	Gaming Consoles	65000	4	Edit Delete

< 1 2 >

[Add product](#)

Fig 6.4 Seller Dashboard

6. Order Page



EMPORIO

Search

Cart Logout

Order History

Product name	Seller name	Quantity	Category	Price	Date
Realme 3	Robert	18	Mobiles	11999	11-21-2019
Oneplus 7T	Robert	9	Mobiles	53999	11-21-2019
Xiaomi	Abbas	8	TV	15000	11-21-2019
Boat Rockers 500	Abbas	9	Headphones	21000	11-21-2019

< 1 >

[Go Home](#)

Fig 6.5 Order Page

7. Admin-Buyer Page

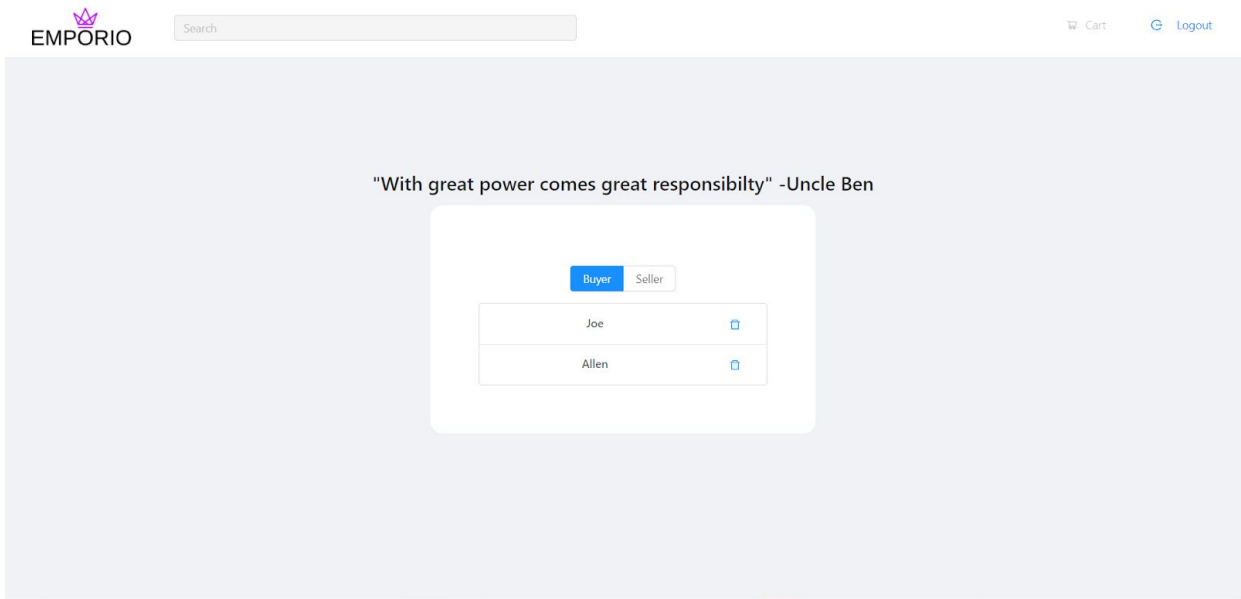


Fig 6.6a Admin Page for Buyer

8. Admin-Seller Page

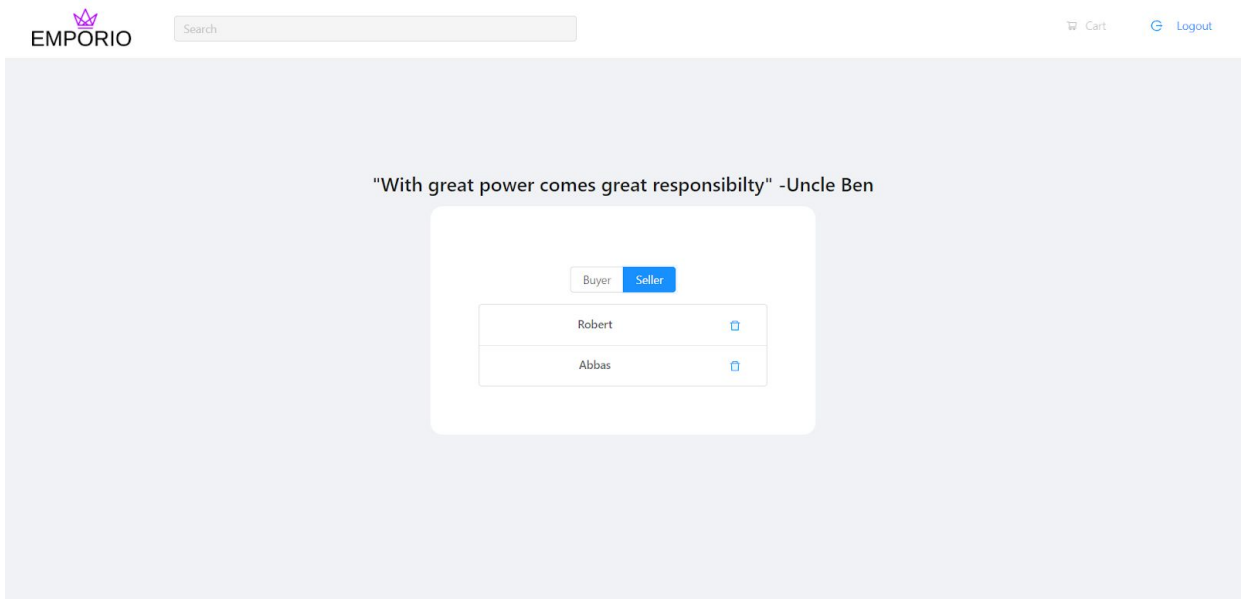


Fig 6.6b Admin Page for Seller

9. Product Input Form

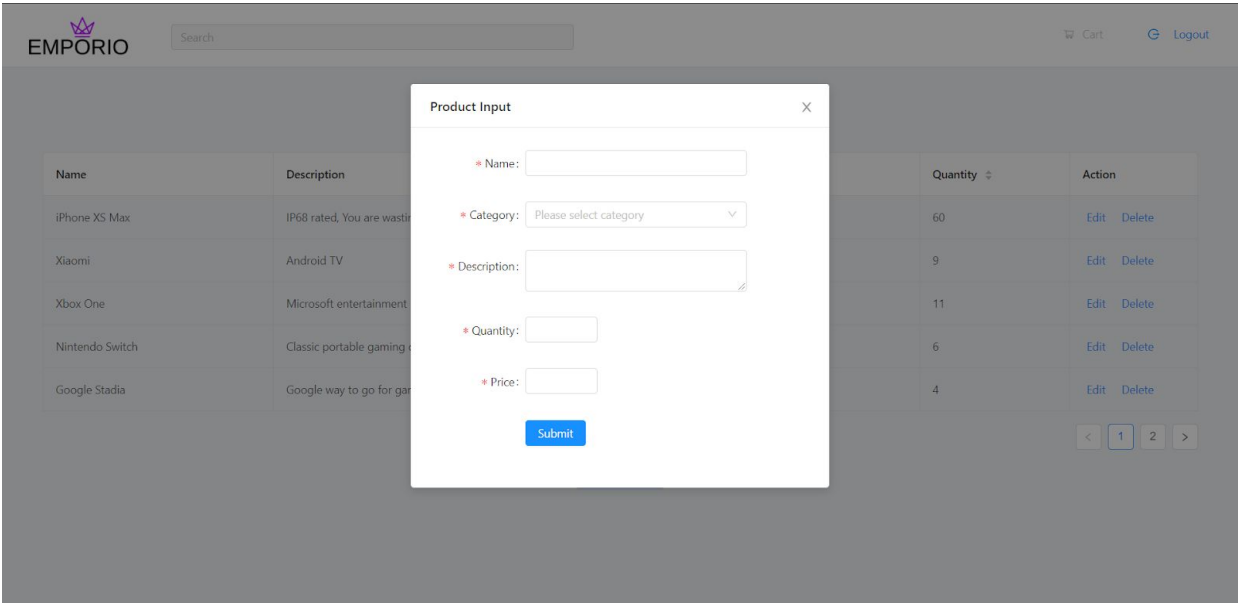


Fig 6.7 Product Input Form

Conclusion

The design of the E-commerce Database Management System is done by taking into consideration all the needs of the modern day market. Providing the normal citizen with the power to buy and sell from any point in the world the best of products in ever changing the landscape of the traditional marketplace. The design of the website is well-thought out and provides the best of speed and looks with easy access to all the features provided by the platform. The MySQL database paired with the powerful ExpressJS running on NodeJS provides unmatched speed while maintaining security and reliability. This application has been successfully tested with suitable test cases. It is user friendly and contains suitable options for users. The front-end for this project was created using a mixture of HTML, CSS, ReactJS. Middle-ware used is ExpressJS running on the NodeJS. MySQL is used to store and manage the database. The goals achieved by this project are:

- Global access
- User friendly environment
- Efficient management of records
- Simplification of operations

References

1. Fundamental of Database Systems by Ramez Elmasri 7th Edition.2017.
2. Database Systems by Ramakrishnan 3rd Edition,2003.
3. ReactJS Docs (<https://reactjs.org/>)
4. Ant Design (<https://ant.design/>)
5. ExpressJS Docs (<https://expressjs.com/>)
6. Maria DB (<https://mariadb.org/>)
7. Apache Friends (<https://www.apachefriends.org/index.html>)