



**Islington college**  
(इसलिंग्टन कलेज)

**Module Code & Module Title**  
**CS5054NI Advanced Programming & Technologies**

**Assessment Type**  
**50% Group Coursework**

**Semester**  
**2024 Spring**

**Group Members**

<b>London Met ID</b>	<b>Student Name</b>
Shreyash Basnet	22067847
Karish Khadka	22068101
Prabesh Shrestha	22067503
Kirtan Maharjan	22068180

**Project Title: Ecommerce**

**Assignment Due Date:** Friday, May 10, 2024

**Assignment Submission Date:** Friday, May 10, 2024

**Submitted to:** Mr. Prithivi Maharjan

**Word Count:** 6552

*I confirm that I understand my coursework needs to be submitted online via Google Classroom under the relevant module page before the deadline in order for my assignment to be accepted and marked. I am fully aware that late submissions will be treated as non-submission and a mark of zero will be awarded.*

## Table of Contents

1. Introduction .....	1
1.1 Java.....	1
1.1.1 Java Introduction.....	1
1.1.2 Java Use Case.....	1
1.2 Aim .....	3
1.3 Objectives .....	3
2. User Interface Design.....	4
2.1 Wireframe.....	4
2.1.1 Login Page .....	5
2.1.2 Signup Page.....	5
2.1.3 Home Page .....	6
2.1.4 Products Page.....	6
2.1.5 User Profile .....	7
2.1.6. Cart Section .....	7
2.1.6 Admin Login .....	8
2.1.7. Admin Panel.....	8
2.1.8 Admin ( Manage Users ).....	9
2.1.9 Admin (Add Product ) .....	9
2.1.10 Admin (Edit Product).....	10
2.2 Web Design .....	11
2.2.1 Home page.....	11
2.2.2 Products Design.....	11
2.2.3 Sort By Options .....	12

2.2.4 Login Page .....	12
2.2.5 Profile without Logged In .....	13
2.2.6 Edit Profile Page .....	14
2.2.7 Admin Panel.....	14
3. Class Diagram .....	17
3.1 Class Diagram for Controller Package.....	18
3.2 Class Diagram for Model Package.....	22
4. Method Description .....	24
4.1 Model Classes .....	24
4.1.1 DbConnection Model Class .....	24
4.1.2 AESEncryption Model Class.....	27
4.1.3 Order Model Class .....	27
4.1.4 Product Model Class.....	28
getProductName() .....	28
4.1.5 User Model Class.....	28
4.2 Controller.....	29
4.2.1 AddProducts Controller Class .....	29
4.2.2 AdminFilter Controller Class.....	29
4.2.3 AdminLogin Controller Class .....	30
4.2.4 AdminPage Controller Class .....	30
4.2.5 Category Controller Class .....	31
4.2.6 DeleteProducts Controller Class .....	31
Method.....	31
4.2.7 Delete User Controller Class .....	32
4.2.8 Edit Products Controller Class.....	33

4.2.9 EditProfile Controller Class.....	33
4.2.10 ErrorHandle Controller Class.....	33
5. Testing .....	35
5.1 Test 1: Sign Up Validation Testing.....	35
5.2 Test 2: Sign Up Testing .....	37
5.3 Test 3: Adding Product to the Cart.....	38
5.4 Test 4: Searching Item by name .....	40
5.5 Test 5: Add Product Through Admin.....	42
Objective .....	42
To verify the process of adding a product to the inventory from the admin panel.....	42
Action Performed .....	42
An admin entered details for a new product and submitted the form to add it to the inventory.....	42
Expected Outcome .....	42
The system should successfully add the product and display a confirmation message indicating the product has been added. ....	42
Actual Result.....	42
The product was successfully added to the inventory, and a confirmation message was displayed. ....	42
Conclusion .....	42
The functionality for adding products from the admin panel is operating correctly. ...	42
6. Tools and Library Used .....	44
6.1 Tools Used.....	44
6.1.1 Eclipse IDE.....	44
6.1.2 XAMPP .....	45
6.1.3 Figma.....	45

6.2 Library Used.....	45
6.2.1 MySQL .....	45
6.2.2 MySQL Jar Connector .....	45
7. Development Process .....	47
8. Critical Analysis .....	51
9. Conclusion .....	53
10. References .....	54

## Figure of Figures

Figure 1: Login Page Wireframe .....	5
Figure 2: Signup Page Wireframe 1.....	5
Figure 3:Home Page Wireframe 1 .....	6
Figure 4: Products Page Wireframe 1 .....	6
Figure 5: Login Page .....	8
Figure 6: Home Screen .....	11
Figure 7: Products View Screen .....	11
Figure 8: Sorting Option .....	12
Figure 9: Login Page .....	12
Figure 10: Registration Screen .....	13
Figure 11: Edit Profile Screen .....	14
Figure 12: Dashboard.....	14
Figure 13: Manage Users.....	15
Figure 14: Add Product .....	15
Figure 15: Edit Product.....	16
Figure 16 Class Diagram for Update Profile .....	18
Figure 17 Class Diagram for View .....	18
Figure 18 Class Diagram of Admin Login .....	18
Figure 19 Class Diagram for Admin Page Filter.....	19
Figure 20 Class Diagram of Edit Profile.....	19
Figure 21 Class Diagram for Logout.....	19
Figure 22 Class Diagram for Order.....	19
Figure 23 Class Diagram for Register User.....	20
Figure 24 Class Diagram of Add Produc .....	20
Figure 25 Class Diagram for Search .....	20
Figure 26 Class Diagram of Edit Product Filter .....	20
Figure 27 Class Diagram of Add Product Filter .....	21
Figure 28 Class Diagram of Login .....	21
Figure 29 Class Diagram for Product .....	22
Figure 30 Class Diagram for User .....	22

Figure 31 Class Diagram for AES Encryption .....	23
Figure 32 Class Diagram for DbConnection.....	23
Figure 33: Table of Test 1 .....	35
Figure 34: Ecommerce home screen.....	35
Figure 35: Login as Admin .....	36
Figure 36: Admin Dashboard .....	36
Figure 37: Register Member Screen.....	37
Figure 38: Add cart button pressed.....	38
Figure 39: Added to Cart.....	39
Figure 40: Searching H in search bar .....	40
Figure 41: Searched Products .....	41
Figure 42: Add Products.....	43
Figure 43: Added Product .....	43
Figure 44: Added Product Gets Displayed .....	43
Figure 45: ECommerce workspace in eclipse. ....	47
Figure 46: Model View and Controllers.....	48
Figure 47: User Model.....	48
Figure 48: Controller.....	49
Figure 49: View.....	49

## List of Tables

Table 1: DbConnection Model Class Method Description .....	26
Table 2: AESEncryption Model Class Method Description .....	27
Table 3: User Model Class Method Description .....	27
Table 4: Product Model Class Method Description .....	28
Table 5: User Model Class Method Description .....	29
Table 6: AddProducts Controller Class Method Description .....	29
Table 7: AdminFilter Controller Class Method Description .....	29
Table 8: AdminLogin Controller Class Method Description .....	30
Table 9: AdminPage Controller Class Method Description .....	30
Table 10: Category Controller Class Method Description .....	31
Table 11: DeleteProducts Controller Class Method Description .....	31
Table 12: DeleteUser Controller Class Method Description .....	32
Table 13: EditProducts Controller Class Method Description .....	33
Table 14: EditProfile Controller Class .....	33
Table 15: ErrorHandler Controller Class Method Description .....	34
Table 16 Table for Test 2 .....	37
Table 17 Table for Test 3 .....	38
Table 18 Table for Test 4 .....	40
Table 19 Table for Test 5 .....	42



# **1. Introduction**

## **1.1 Java**

### **1.1.1 Java Introduction**

Java, created in 1995, stands out as an incredibly versatile programming language. It ranks among the most popular programming languages globally, experiencing rapid growth in usage. Oracle currently develops and maintains Java, supported by a large and active community of developers. The platform-independent nature of Java allows its code to run across various systems with minimal adjustments, making it ideal for a wide range of applications including web and application servers, mobile applications, and desktop software. (w3schools, 2023).

### **1.1.2 Java Use Case**

Java is a versatile and widely used programming language with applications across various domains. According to W3Schools (2023), Java is prominently utilized in mobile app development, especially for Android devices, due to its portability and compatibility across different platforms. It's also favored for developing desktop applications, ranging from productivity tools to entertainment software, including well-known games like Minecraft.

Java's robustness extends to web application development, where it powers dynamic content on web and application servers, ensuring scalable, secure, and reliable web services. In the gaming industry, Java's capabilities are leveraged to create immersive experiences on desktops, consoles, and mobile platforms.

In database management, Java facilitates connectivity with various database systems such as MySQL, Oracle, and Microsoft SQL Server, making it invaluable for handling and analyzing extensive data sets.

The group coursework focuses on creating an e-commerce website for selling clothing using Java, adhering to the Model-View-Controller (MVC) architecture. This architecture splits the project into three main packages: model, view, and controller. The controller

manages user requests through servlets, the model handles database interactions, and the view is responsible for displaying user interfaces via JSP files.

Key features of the website include:

- A secure login system where user credentials are encrypted for safety.
- An administrative portal accessible only by admins to manage products and view orders.
- A user-friendly homepage showcasing products with options for searching and filtering without the need to log in.
- Enhanced user functionalities post-login, allowing users to modify their profile details and place orders.
- Comprehensive validation and exception handling to maintain error-free operations.
- The site employs standard programming practices such as commenting and consistent naming conventions for better maintainability and scalability.

## 1.2 Aim

The aim of the project is to design and develop an e-commerce website for electronics and gadgets that is both user-friendly and efficient, utilizing the Model View Controller (MVC) architectural pattern to ensure the site is well-structured, scalable, and maintainable.

## 1.3 Objectives

- Implement the MVC Architecture: - Organize the project into three distinct packages: model, view, and controller, each serving specific roles in the application structure.
- Develop a Secure Login System: - Create a login module that connects with a database for user authentication, supports session management, and handles different user roles (normal users and administrators) with appropriate redirection and security measures.
- Admin Panel Functionality: - Allow administrators to manage product listings through the capability to add, update, and delete products, and view orders with their statuses (pending, delivered).
- User Interface and Accessibility:
  - Design an engaging and accessible user interface that includes a homepage with search functionality, product information display, and user interactions like add to cart and product purchase.
  - Develop a profile management page where users can view and edit their personal information, accessible only to logged-in users.
- About Us Page: - Provide essential contact information and allow users to reach out via phone or email, enhancing user engagement and support.
- Validation and Exception Handling: - Incorporate robust validation and exception handling across the site to manage errors effectively and ensure data integrity.
- Compliance and Security: - Ensure that all parts of the application adhere to security best practices to protect user data and prevent common security threats.

## **2. User Interface Design**

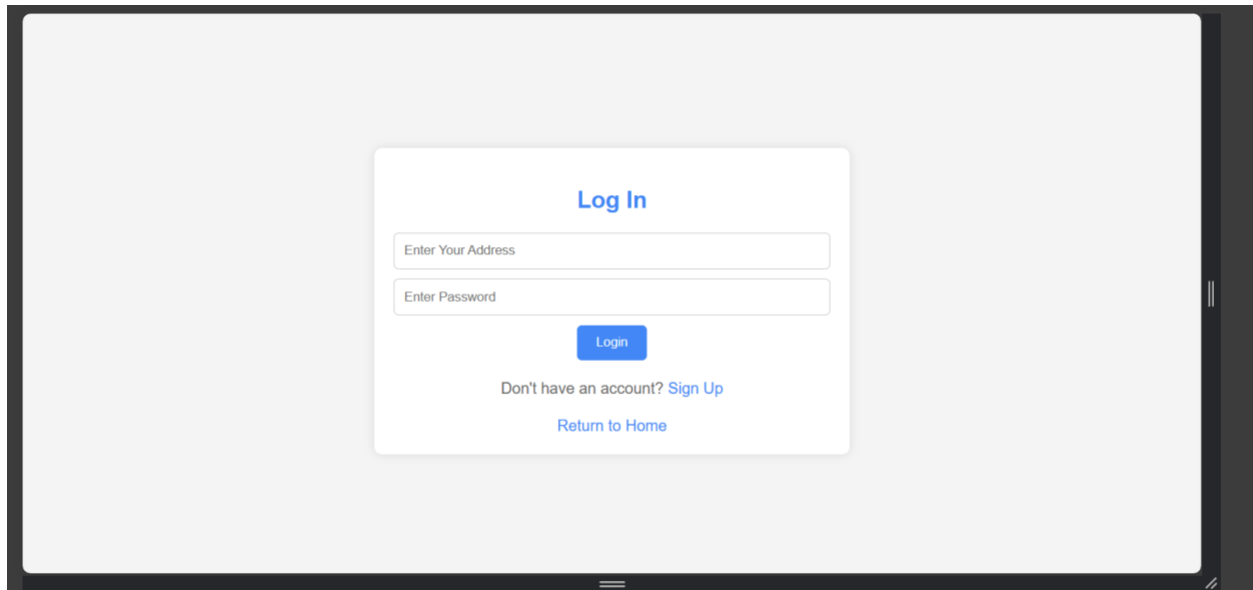
The technologies that have been used to implement user interface design are Html, CSS, and Vanilla JavaScript. Html is used to structure the content. CSS is used to style the content whereas JavaScript is used to add interactivity and add dynamic functionality. Html provided us with the structure of our pages, CSS was used to control the appearance and layout and JavaScript was used to add dynamic behavior to our pages.

### **2.1 Wireframe**

The wireframe was made with Figma. Figma is a tool mainly intended for users who want to design user interfaces. The program's initial release was in September 2016. More than four million users use Figma as of the writing in January 2023. Figma is now a part of the Adobe ecosystem after being acquired by Adobe in September 2022 for \$20 billion. The free plan for Figma offers three files for Figma and FigJam as well as unlimited collaborators. If you require more access, you can select from paid subscriptions (Mairoca, 2023).

Figma was used to explore ideas with our teammates at first then we brought those ideas to life by implementing designs and even adding some flows (UX). After implementing those designs, we started doing the prototype section which gave us a clear view of how our web app will function in the future.

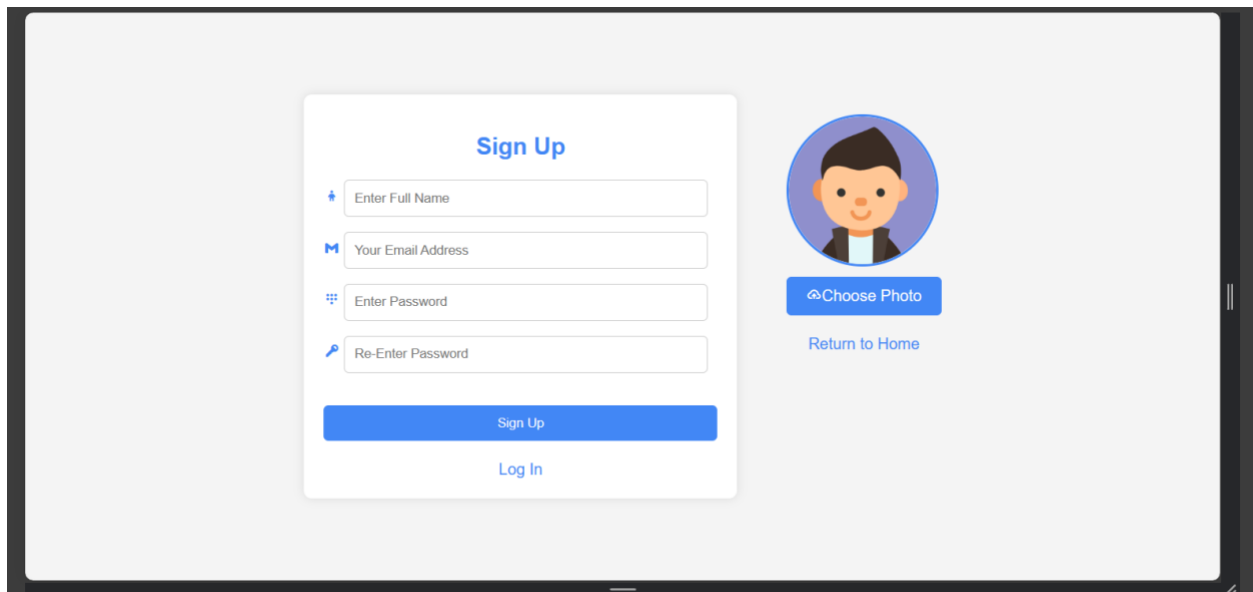
### 2.1.1 Login Page



A wireframe of a login page. It features a central white card on a light gray background. The card has a blue title "Log In". Below the title are two input fields: "Enter Your Address" and "Enter Password". A blue "Login" button is positioned below the password field. At the bottom of the card, there is a link "Don't have an account? Sign Up" and a link "Return to Home". The entire page is framed by a dark gray border with a mobile navigation bar at the bottom.

Figure 1: Login Page Wireframe

### 2.1.2 Signup Page



A wireframe of a signup page. It features a central white card on a light gray background. The card has a blue title "Sign Up". Below the title are four input fields: "Enter Full Name", "Your Email Address", "Enter Password", and "Re-Enter Password". A blue "Sign Up" button is positioned below the password fields. Below the button is a link "Log In". To the right of the card, there is a circular profile picture placeholder with a cartoon character. Below the placeholder is a blue "Choose Photo" button and a link "Return to Home". The entire page is framed by a dark gray border with a mobile navigation bar at the bottom.

Figure 2: Signup Page Wireframe 1

### 2.1.3 Home Page

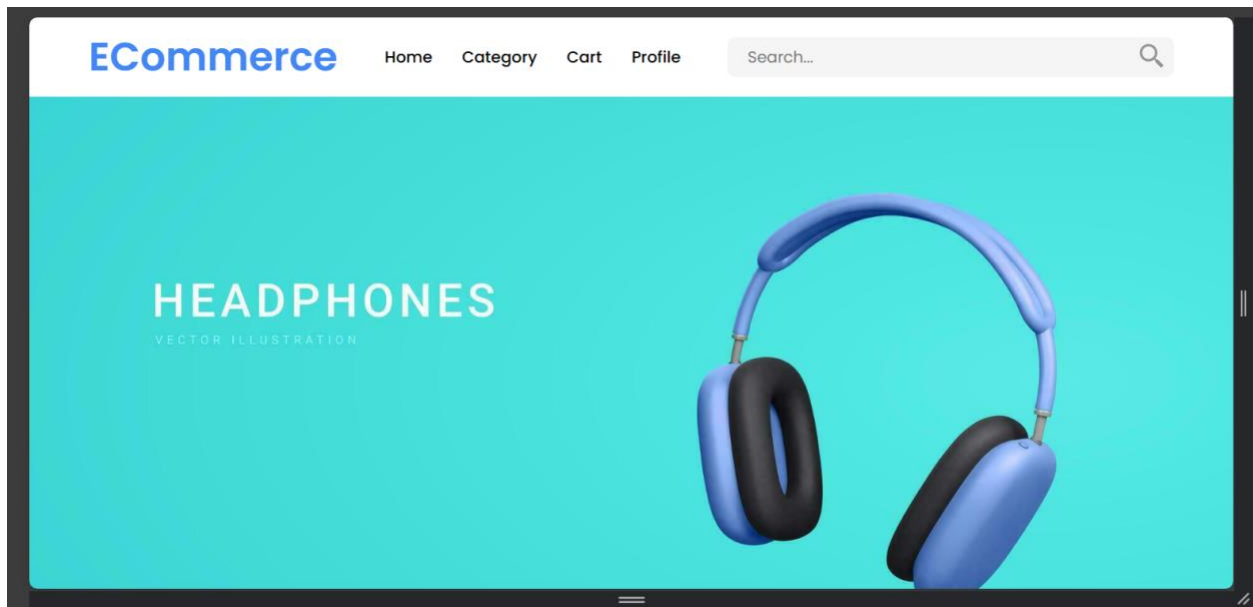


Figure 3:Home Page Wireframe 1

### 2.1.4 Products Page

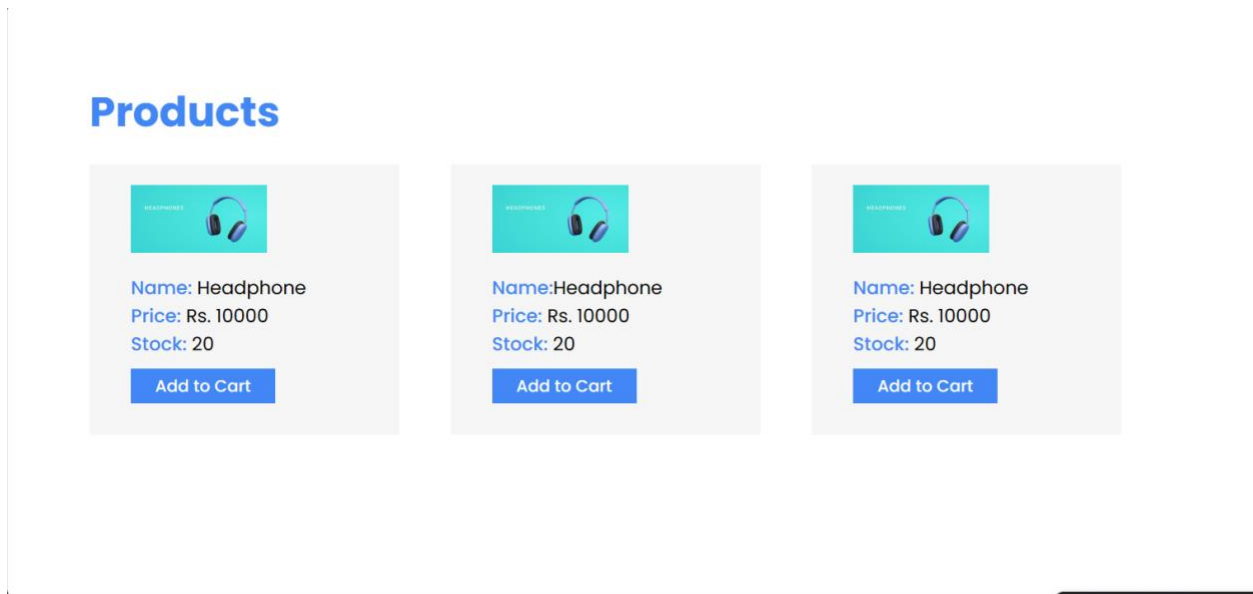




Figure 4: Products Page Wireframe 1

### 2.1.5 User Profile


## Edit Profile

 Test Useradada


Name:

 test@gmail.com

Email:


 password

Password:

 password

Confirm:

Update



Choose Photo

[Return to Home](#)

### 2.1.6. Cart Section


## ECommerce

[Home](#) [Category](#) [Cart](#) [Profile](#)


Search...

# HEADPHONES


VECTOR ILLUSTRATION



### Cart Items



**Headphone**  
Price: Rs. 10000



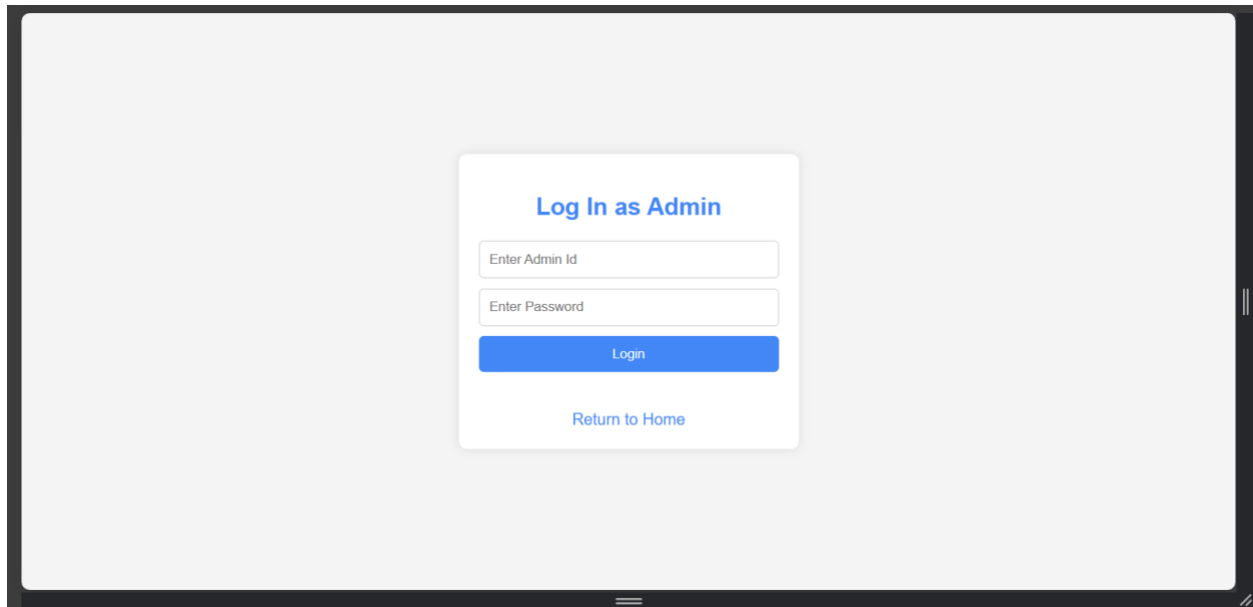
**Camera**  
Price: Rs. 30000

Total Price: 40000

Order Now!

7 | Page

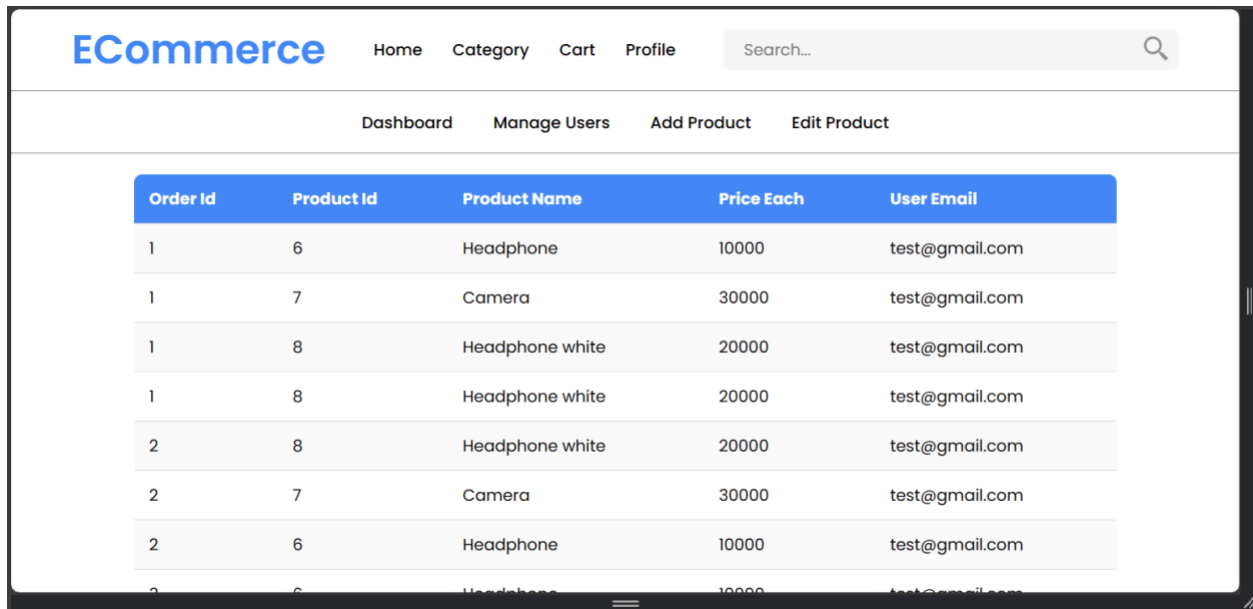
## 2.1.6 Admin Login



The image shows a login page with a central white card on a light gray background. The card has the title "Log In as Admin" in blue. Below the title are two input fields: "Enter Admin Id" and "Enter Password". A blue "Login" button is positioned below the password field. At the bottom of the card is a blue link that says "Return to Home".

Figure 5: Login Page

## 2.1.7. Admin Panel

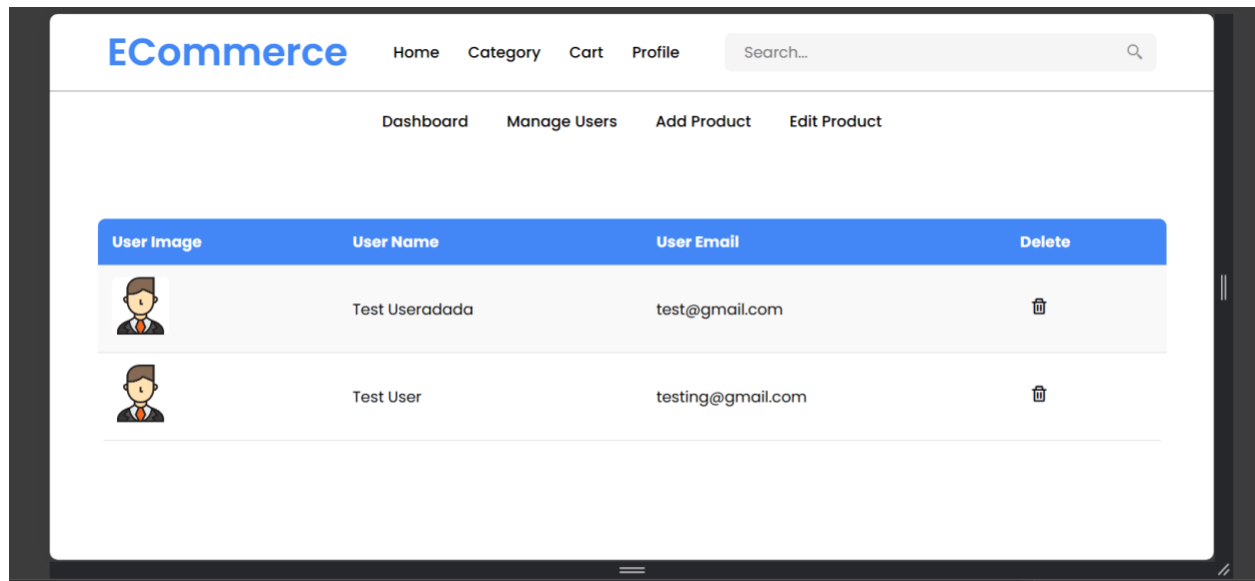


The image is a screenshot of an "ECommerce" admin panel. The top navigation bar includes links for "Home", "Category", "Cart", and "Profile", along with a search bar. Below this is a sub-navigation bar with "Dashboard", "Manage Users", "Add Product", and "Edit Product". The main content area displays a table with the following data:

Order Id	Product Id	Product Name	Price Each	User Email
1	6	Headphone	10000	test@gmail.com
1	7	Camera	30000	test@gmail.com
1	8	Headphone white	20000	test@gmail.com
1	8	Headphone white	20000	test@gmail.com
2	8	Headphone white	20000	test@gmail.com
2	7	Camera	30000	test@gmail.com
2	6	Headphone	10000	test@gmail.com



## 2.1.8 Admin ( Manage Users )




## 2.1.9 Admin (Add Product )

Dashboard   Manage Users   **Add Product**   Edit Product

**Product Name:**

**Product Price:**

**Product Quantity:**



**Upload**

No file chosen





## 2.1.10 Admin (Edit Product)

**ECommerce**

HomeCategoryCartProfile

Search...

DashboardManage UsersAdd ProductEdit Product

Product Image	Product Name	Total Quantity	Product Price	Action
	Headphone	20	10000	<a href="#">✎</a> <a href="#">🗑</a>
	Camera	10	30000	<a href="#">✎</a> <a href="#">🗑</a>
	Headphone white	15	20000	<a href="#">✎</a> <a href="#">🗑</a>
	Apple	10	10	<a href="#">✎</a> <a href="#">🗑</a>

## 2.2 Web Design

### 2.2.1 Home page

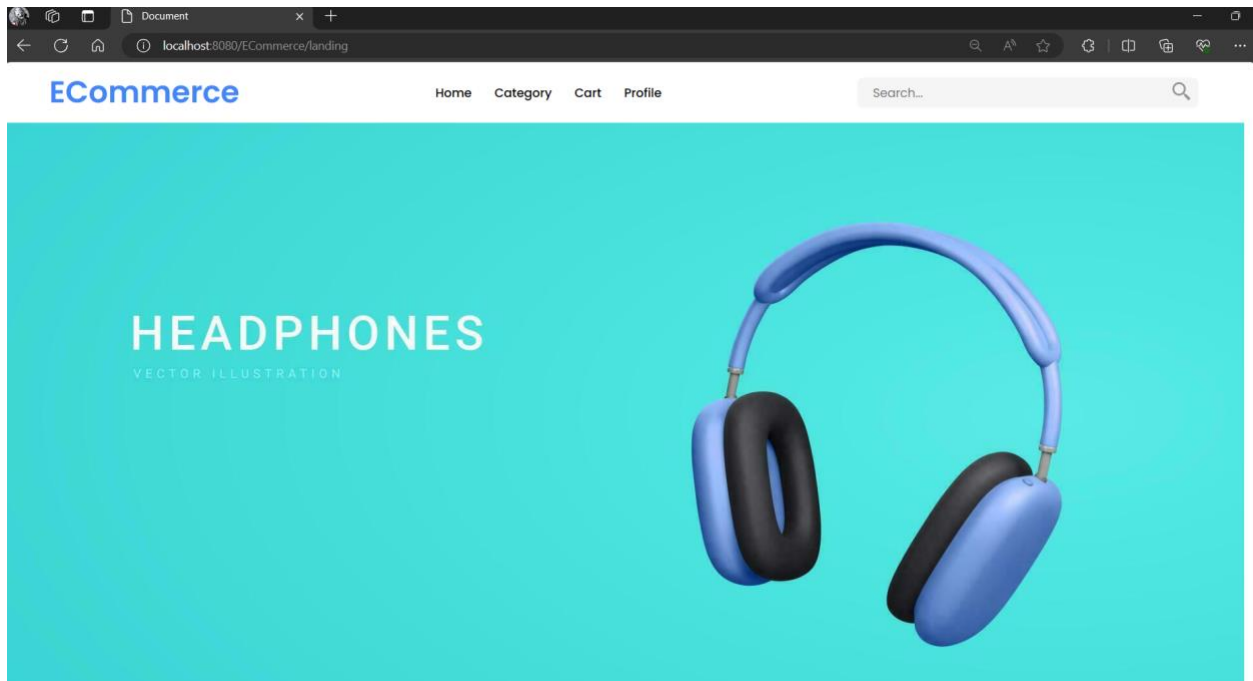


Figure 6: Home Screen

### 2.2.2 Products Design

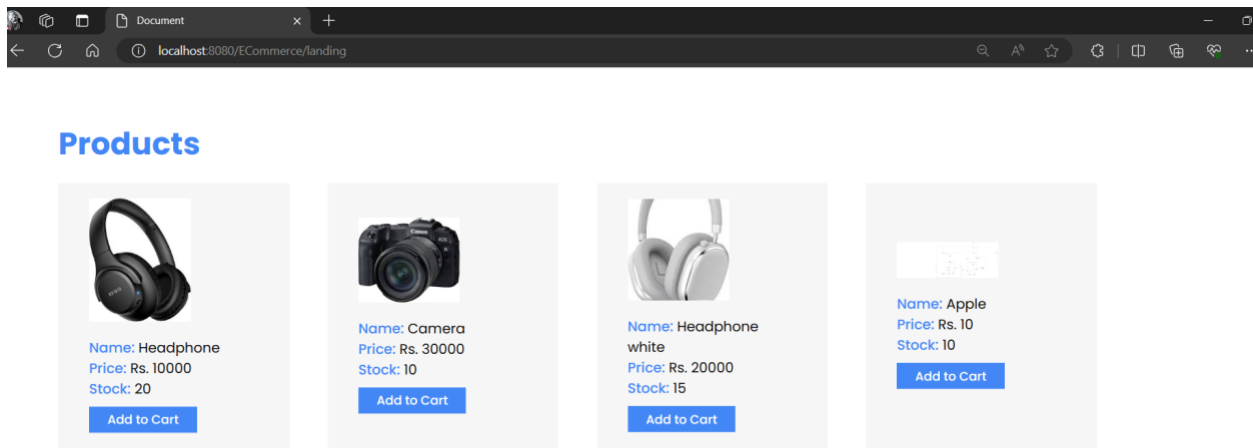


Figure 7: Products View Screen

### 2.2.3 Sort By Options

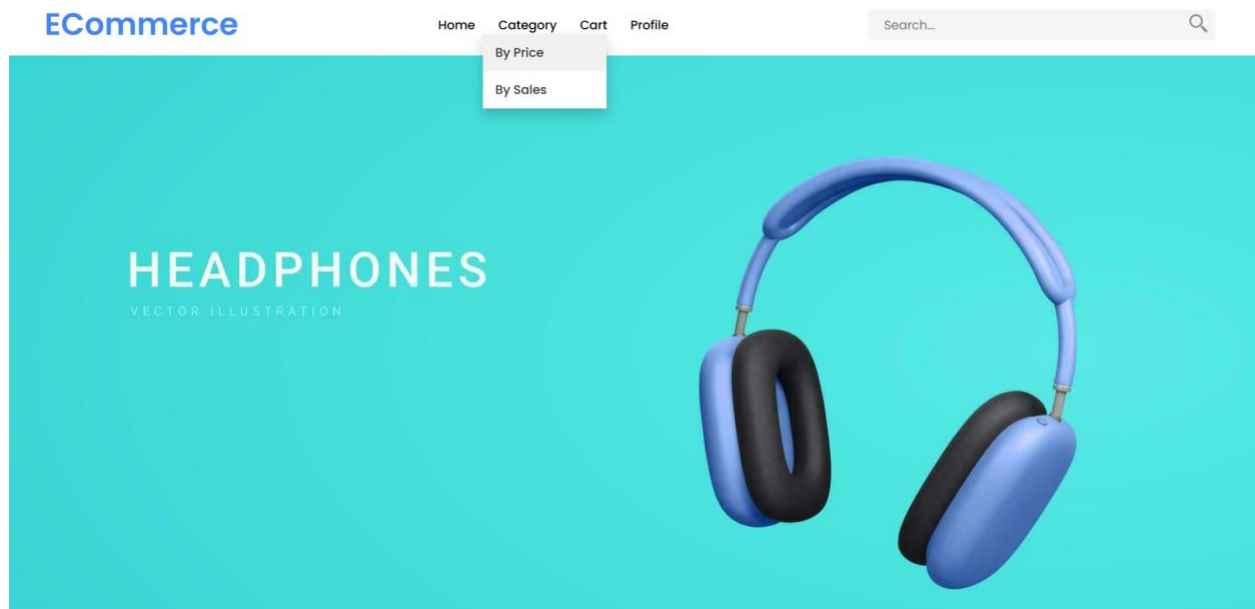


Figure 8: Sorting Option

### 2.2.4 Login Page

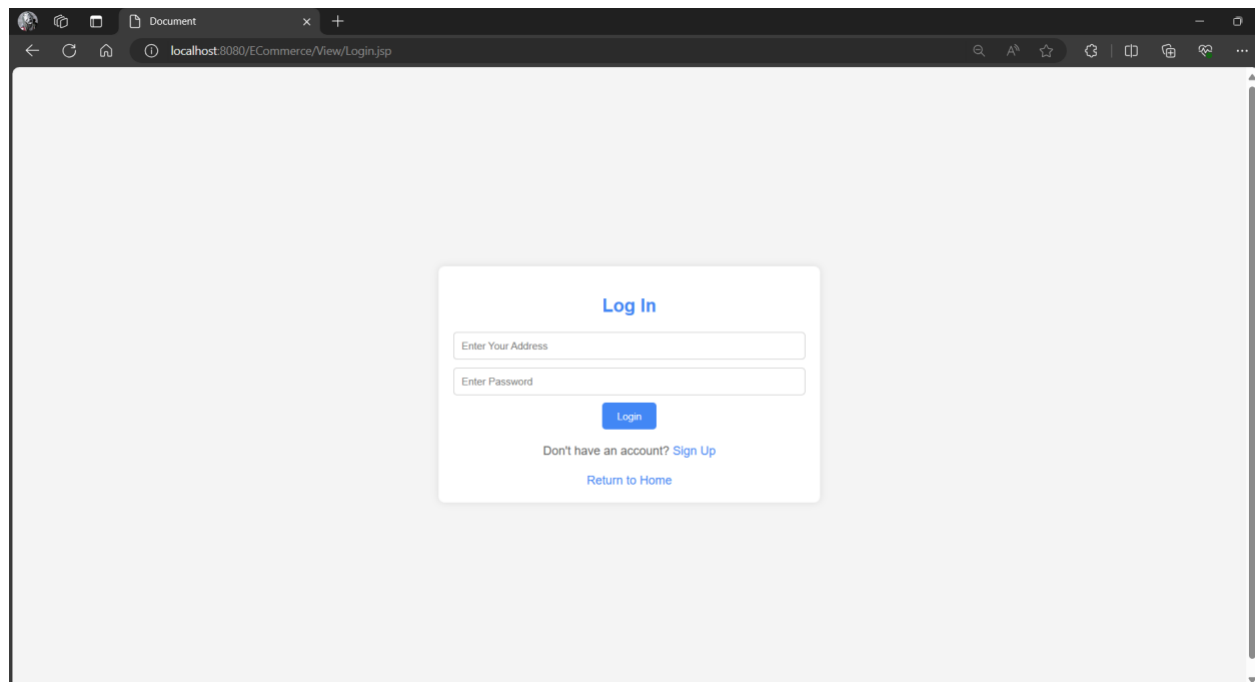


Figure 9: Login Page

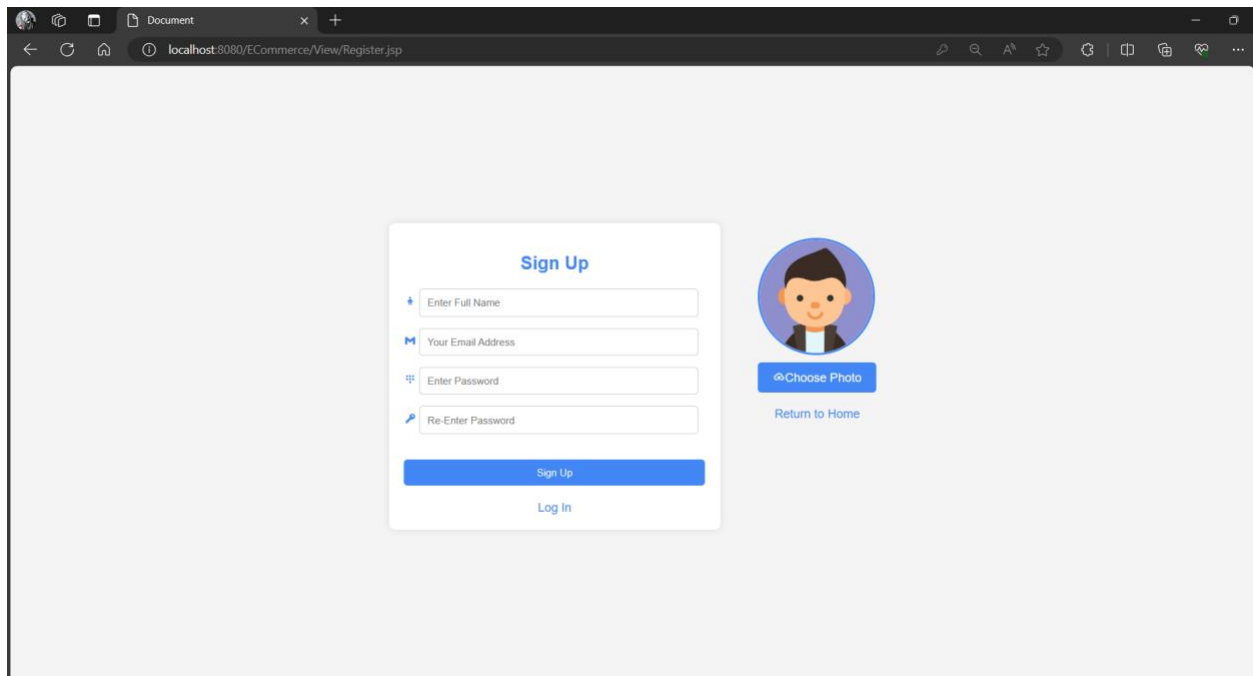
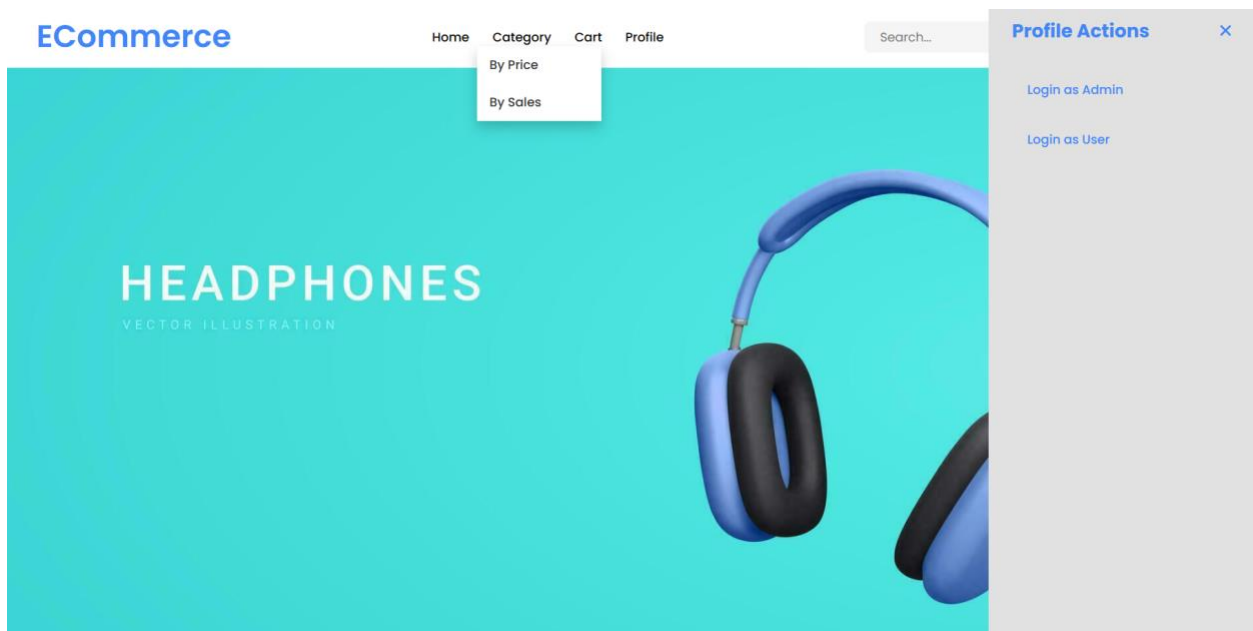


Figure 10: Registration Screen

## 2.2.5 Profile without Logged In



## 2.2.6 Edit Profile Page

The screenshot shows a web browser window with the address bar displaying 'localhost:8080/ECommerce/editProfile?email=test@gmail.com'. The page content is a light gray background with a white 'Edit Profile' form in the center. The form has a title 'Edit Profile' in blue. It contains the following fields: a text input for 'Name' with the value 'Test Useradada', an email input for 'Email' with the value 'test@gmail.com', a password input for 'Password' with the value 'password', and a confirm password input for 'Confirm' with the value 'password'. Below these fields is a blue 'Update' button. To the right of the form is a circular profile picture placeholder with a blue outline. Below the placeholder is a blue 'Choose Photo' button and a blue 'Return to Home' link.

Figure 11: Edit Profile Screen

## 2.2.7 Admin Panel

### 2.2.7.1 Dashboard

The screenshot shows the ECommerce Admin Panel Dashboard. The page has a navigation bar with links for Home, Category, Cart, and Profile, and a search bar. Below the navigation bar are tabs for Dashboard, Manage Users, Add Product, and Edit Product. The main content area displays a table with columns: Order Id, Product Id, Product Name, Price Each, and User Email. The table contains 10 rows of data.

Order Id	Product Id	Product Name	Price Each	User Email
1	6	Headphone	10000	test@gmail.com
1	7	Camera	30000	test@gmail.com
1	8	Headphone white	20000	test@gmail.com
1	8	Headphone white	20000	test@gmail.com
2	8	Headphone white	20000	test@gmail.com
2	7	Camera	30000	test@gmail.com
2	6	Headphone	10000	test@gmail.com
3	6	Headphone	10000	test@gmail.com
3	7	Camera	30000	test@gmail.com

Figure 12: Dashboard

### 2.2.7.2 Manage Users

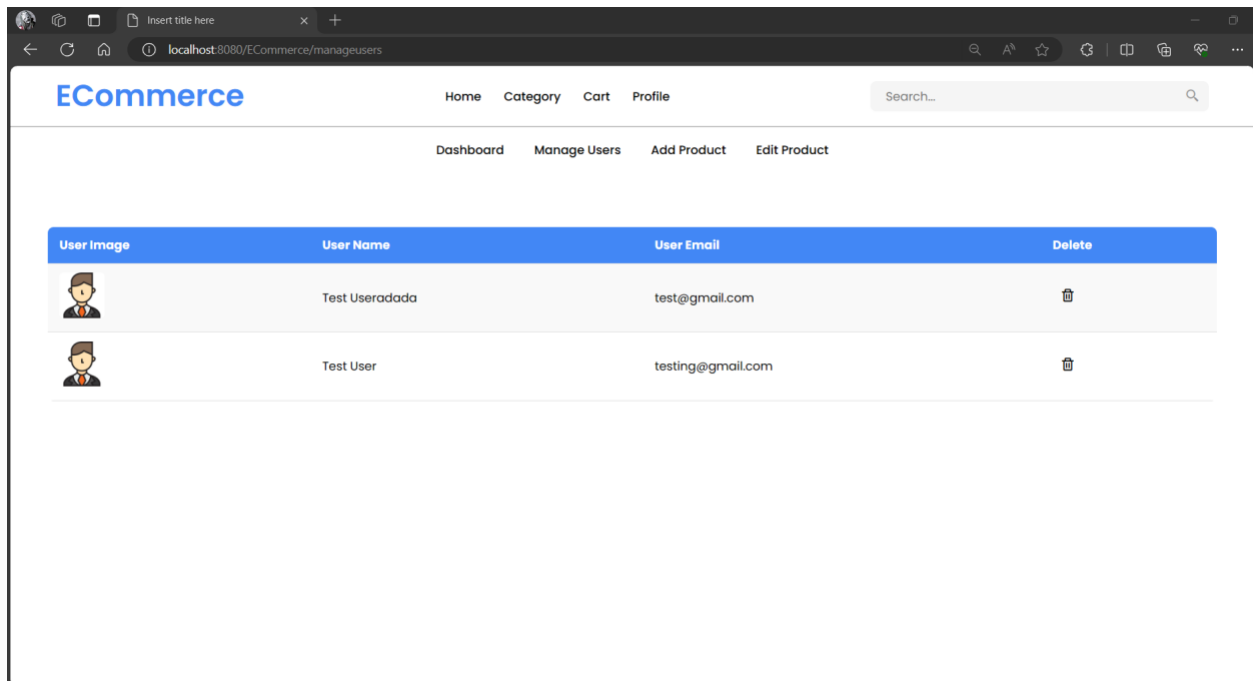


Figure 13: Manage Users

### 2.2.7.3 Add Product

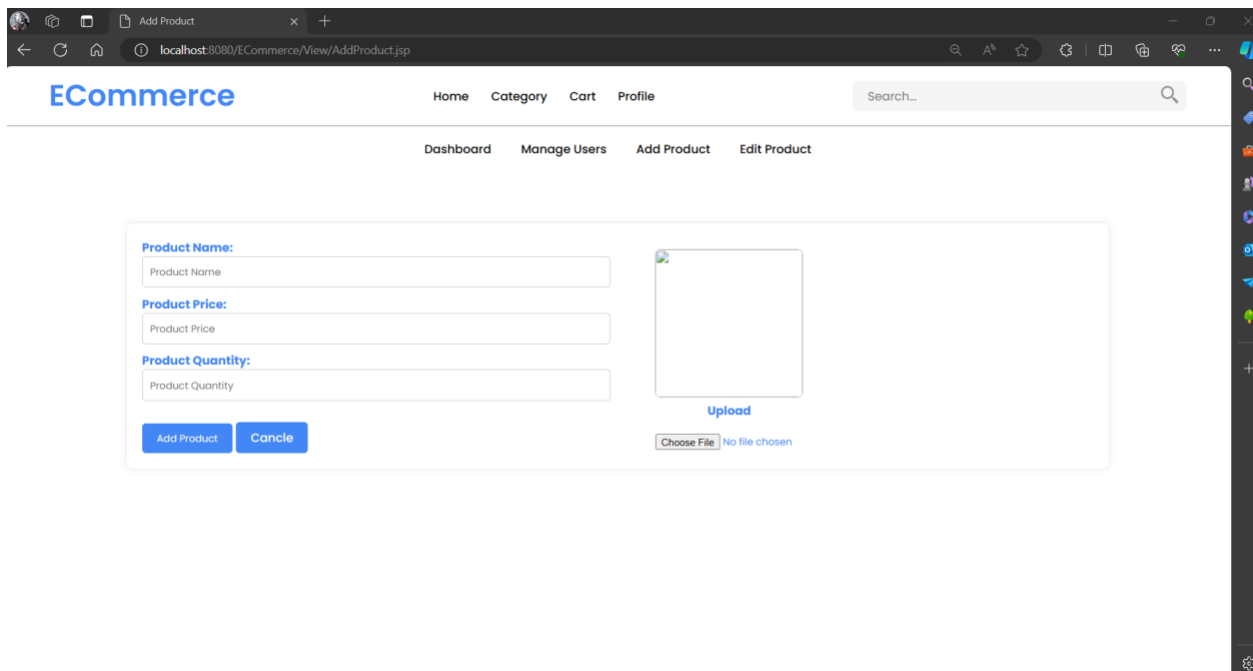


Figure 14: Add Product

#### 2.2.7.4 Edit Product

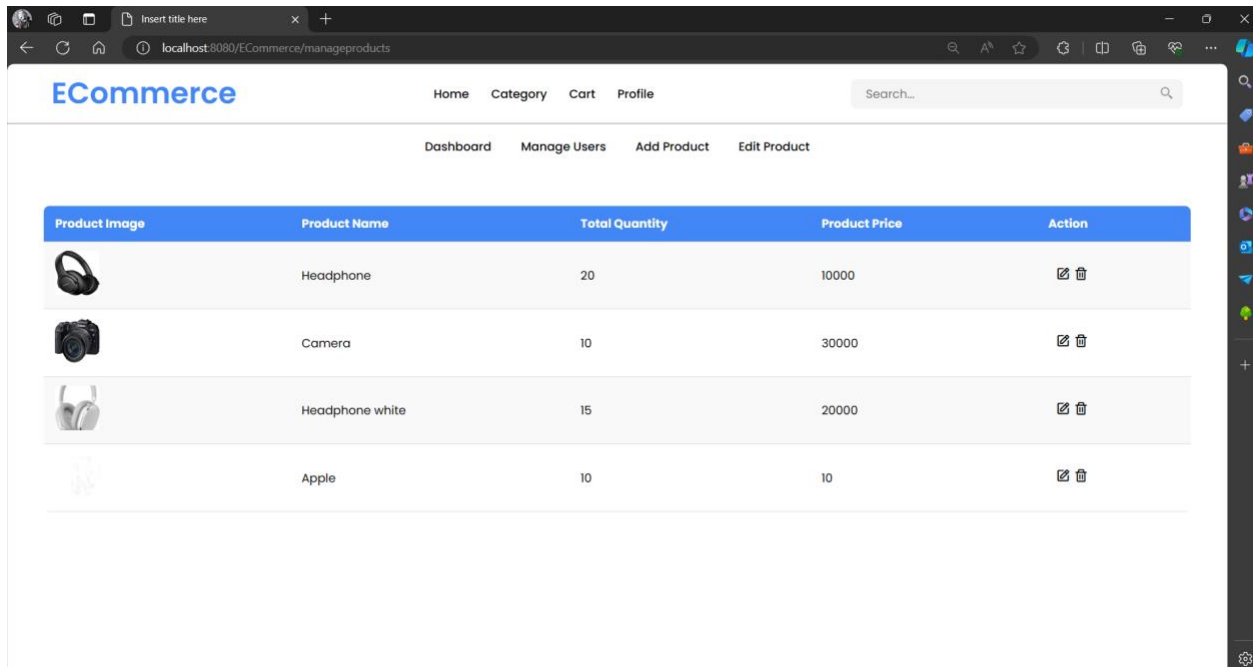


Figure 15: Edit Product



### **3. Class Diagram**

A class diagram is a particular kind of diagram that displays an application's structure. Understanding the many components of the program, such as the classes, interfaces, and connections between them, is helpful. Class diagrams are beneficial because they may be used to write application-specific code. They demonstrate the components and capabilities of each class. Programming that uses objects frequently makes use of them. Class diagrams assist programmers in designing and constructing complicated systems (tutorialspoint, 2023).

#### **Benefits of Class Diagram**

- It helps to simplify the complicated architecture programs and slows for the implementation of OOPs, a generally accepted idea.
- It expressed the code which is written in the thousand of words into a few of illustrative pictures, which might make it easier to understand.
- It helps to improve the clarity and authenticity of communication. • This concept helps to illustrate the full system's view which might be helpful.
- In the OOPs concept, once the software programmer has a thorough understanding of the issue, it becomes quite simple for them to implement the actual needs.

#### **Drawbacks of Class Diagram**

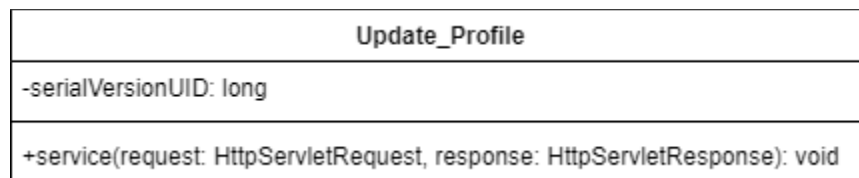
- One of the major drawbacks of UML is the significant amount of time required to handle the diagrams.
- UML consists of too much design which have take time to implement and get overweight of work for some companies.
- Due to its complexity, many developers or programmers hold an opinion that they may benefit from avoiding it, which may be reason in such a way programmer could get discourage from acquiring or utilizing it.

## Purpose of Class Diagram

- Demonstrates the system's classifier's static structure.
- Diagram offers a fundamental notation for addition UML-recommended structure diagrams.
- Beneficial or advantage creation for programmer or developers as well as other team members.
- Class diagram can be used by the business analysts to model systems from a business standpoint.

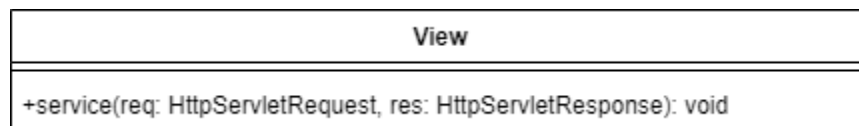
### 3.1 Class Diagram for Controller Package

#### Update Profile



*Figure 16 Class Diagram for Update Profile*

#### View



*Figure 17 Class Diagram for View*

#### Admin Login



*Figure 18 Class Diagram of Admin Login*

## Admin Page Filter



Figure 19 Class Diagram for Admin Page Filter

## Edit Profile

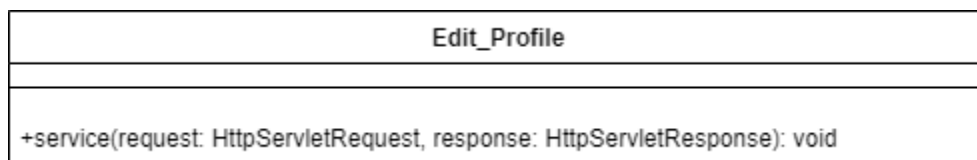


Figure 20 Class Diagram of Edit Profile

## Logout

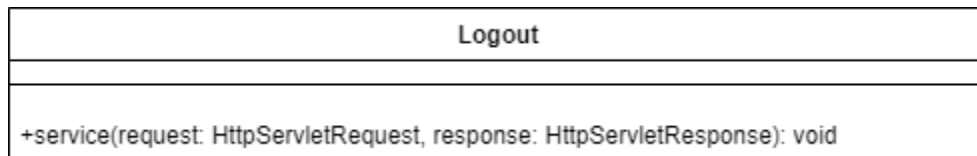


Figure 21 Class Diagram for Logout

## Order

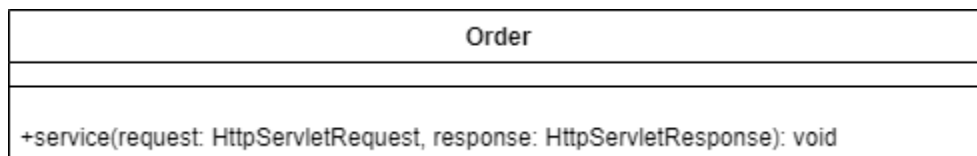


Figure 22 Class Diagram for Order

## Register User

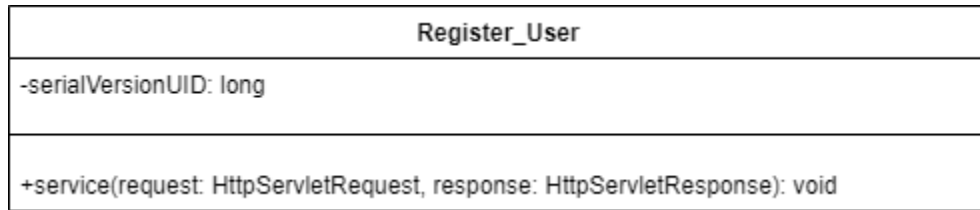


Figure 23 Class Diagram for Register User

## AddProduct.java

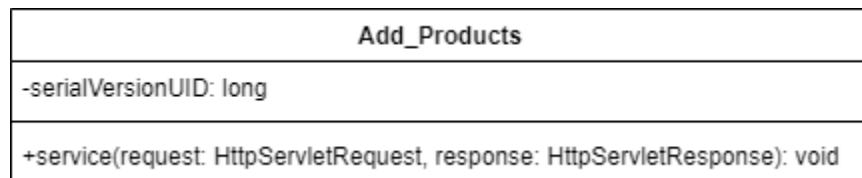


Figure 24 Class Diagram of Add Product

## Search

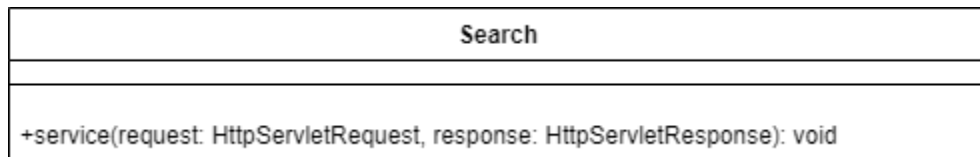


Figure 25 Class Diagram for Search

## Edit Product Filter



Figure 26 Class Diagram of Edit Product Filter

## AddProductFilter.java



Figure 27 Class Diagram of Add Product Filter

## Login

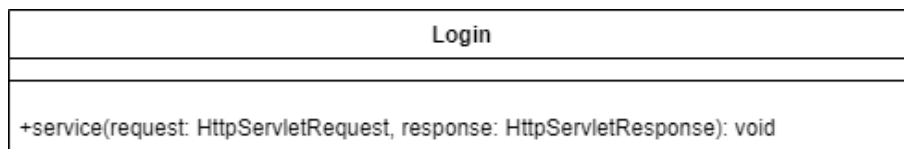


Figure 28 Class Diagram of Login

## 3.2 Class Diagram for Model Package

### Product

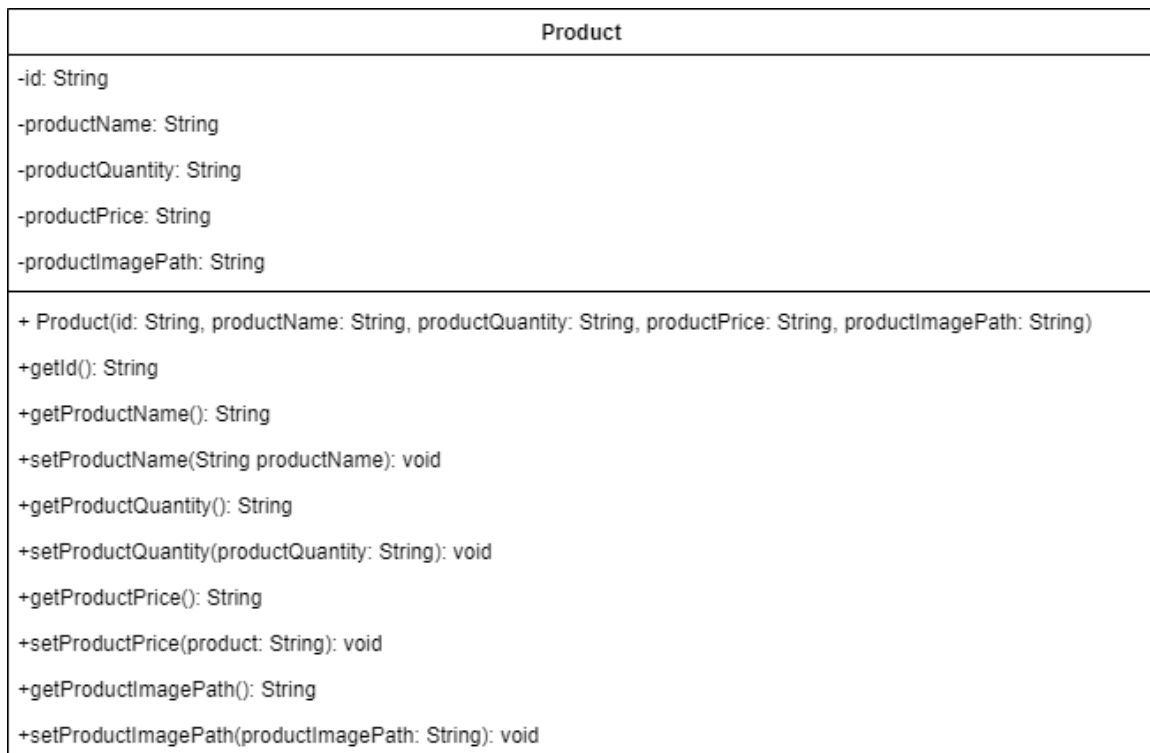


Figure 29 Class Diagram for Product

### User

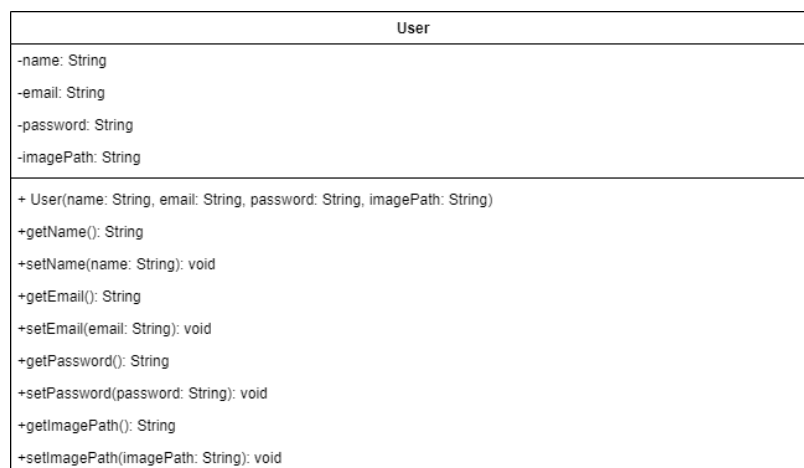


Figure 30 Class Diagram for User

## AES Encryption

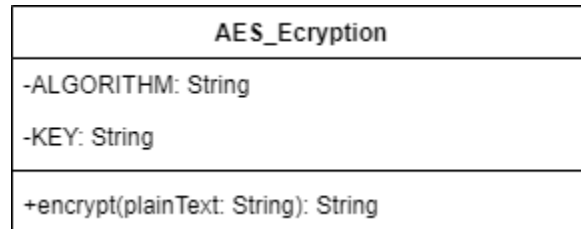


Figure 31 Class Diagram for AES Encryption

## DbConnection

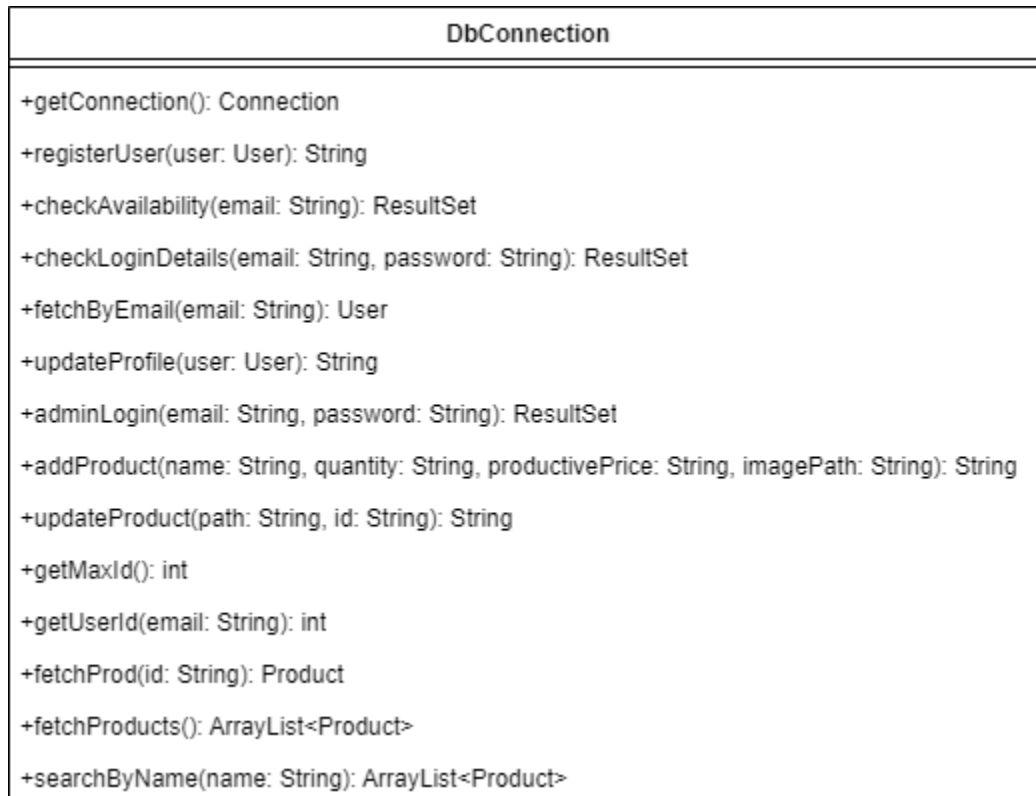


Figure 32 Class Diagram for DbConnection

## 4. Method Description

### 4.1 Model Classes

#### 4.1.1 DbConnection Model Class

Method	Description
getConnection()	This method provides a Connection object for a MySQL database named "ECommerce" on the local machine. It may raise ClassNotFoundException and SQLException if the necessary driver class is absent or if there's an issue with the connection establishment.
registerUser(User user)	This method accepts a User object, inserting the user's information into the "users" database table. It uses the getConnection() method to establish a database connection, prepares an SQL query for inserting the data, sets the parameters from the User object, executes the query, checks the affected rows, closes the connection, and returns a success or failure message.
checkAvailability(String email)	This method checks for the existence of a user with the provided email in the database. It returns a ResultSet with the user data if found, or null if not.
checkLoginDetails(String email, String password)	This method verifies user login details against the database. It retrieves and decrypts the stored password using AES encryption to compare with the provided password. If they match, it returns the user details as a ResultSet; otherwise, it returns null. Errors result in printing the stack trace and returning null.
fetchByEmail(String email)	This method fetches user details based on an email address, returning a User object with decrypted password if the user exists, or null if not.
updateProfile(User user)	This method updates a user profile in the database, changing fields like name, password, and image path based on the



	user's email. It returns a success message if the update is successful, otherwise, it reports an error.
adminLogin(String email, String password)	This method checks admin login credentials against the "admin" table in the database. If successful, it returns a ResultSet containing the admin's details; otherwise, it returns null.
addProduct(String name,String quantity,String productPrice,String imagePath)	This method adds a new product to the database and retrieves the product id. If successful, it returns the product id; otherwise, an error message.
updateProdotlImagePath(String path, String id)	This method updates the image path for a specific product in the database and confirms success with a message.
getMaxId()	This method retrieves the highest "Order_Id" from the "orders" table and calculates the next id by adding 1.
getUserId(String email)	This method returns the user ID associated with a given email from the "users" table, or returns 0 if no such user exists.
fetchProd(String id)	This method retrieves a Product object based on the product ID from the database or returns null if not found.
fetchProducts()	This method collects all products from the database into an ArrayList of Product objects.
searchByName(String name)	This method searches for products by name, returning an ArrayList of matching Product objects or null if an error occurs.
sortBy(String sortBy)	This method sorts products by specified criteria ("price" or "sale") and returns a sorted ArrayList of Product objects.
fetchOrderList()	This method retrieves and returns an ArrayList of Order objects, gathering order information by joining relevant database tables.
fetchOneRow(String id)	This method fetches a single product row based on the given ID, returning a Product object if successful or null otherwise.

updateProductsData(Product product)	This method updates product details in the database and returns true if the update affects at least one row, indicating success.
deleteParameter(String id)	This method deletes a product entry from the database based on the product ID, returning a success message or null if unsuccessful.
fetchUserData()	This method gathers all user data from the database into an ArrayList of User objects.
deleteUser(String email)	This method removes a user from the database based on their email and returns a success or error message depending on the outcome.
sortBy(String sortBy)	This function organizes a collection of products based on a specific criterion, either 'price' or 'sale', using the 'sortBy' parameter to determine the sorting logic. If 'sortBy' is set to 'price', the method fetches all products from the database and arranges them in ascending price order. If 'sortBy' is set to 'sale', it organizes the products by sales volume. It returns an ArrayList of sorted Product objects.

*Table 1: DbConnection Model Class Method Description*

#### 4.1.2 AESEncryption Model Class

Method	Description
encrypt(String plainText)	This function encrypts a plaintext string using a secret key and a specified encryption algorithm, returning the encrypted string in Base64 format. It also logs any errors that occur during the encryption process.
decrypt(String ciphertext)	This function decrypts an encrypted string using a designated secret key and algorithm, returning the plaintext string.

*Table 2: AESEncryption Model Class Method Description*

#### 4.1.3 Order Model Class

Method	Description
Order()	Initializes an Order object with details such as orderId, productId, productName, userName, and priceEach.
getOrderId()	Retrieves the orderId associated with this Order.
setOrderId(String orderId)	Assigns a new orderId to this Order.
getProductId()	Fetches the productId tied to this Order.
setProductId(String productId)	Updates the productId for this Order.
getProductName()	Obtains the productName linked to this Order.
setProductName(String productName)	Sets a new productName for this Order.
getUserName()	Gets the userName related to this Order.
setUserName(String userName)	Updates the userName for this Order.
getPriceEach()	Retrieves the price per unit for this Order.
setPriceEach(String priceEach)	Sets the price per unit of this Order to the specified value.

*Table 3: User Model Class Method Description*

#### 4.1.4 Product Model Class

Method	Description
Product()	Constructor that initializes a new Product object with attributes such as id, productName, productQuantity, productPrice, and productImagePath.
getProductName()	Retrieves the name of the product.
setProductName(String productName)	Updates the product's name to the specified value.
getProductQuantity()	Returns the quantity of the product available.
setProductQuantity(String productQuantity)	Sets the product's quantity to the given value.
getProductPrice()	Fetches the current price of the product.
setProductPrice(String productPrice)	Adjusts the product's price to the specified amount.
getProductImagePath()	Obtains the image path associated with the product.
setProductImagePath()	Updates the image path for the product.

Table 4: Product Model Class Method Description

#### 4.1.5 User Model Class

Method	Description
User ()	Constructor that sets up a new User object by initializing its name, email, password, and image path.
getName()	Retrieves the user's name.
setName(String name)	Updates the user's name to the specified value.
getEmail()	Fetches the user's email address.
setEmail(String email)	Changes the user's email to the provided address
getPassword()	Gets the user's password.
setPassword(String password)	Sets the user's password to the new value provided.
getImagePath()	Obtains the file path for the user's profile image.

setImagePath(String imagePath)	Updates the image path for the user's profile picture.
--------------------------------	--

*Table 5: User Model Class Method Description*

## 4.2 Controller

### 4.2.1 AddProducts Controller Class

Method	Description
service(HttpServletRequest request, HttpServletResponse response)	This method processes an HTTP request by extracting product details like name, price, and quantity from the request parameters. It initiates a new database connection through the DbConnection class and utilizes the addProduct method to insert the product into the database. Subsequently, it generates a local path for the product's image and saves the image file at this location. Following this, it updates the product's image path in the database with the updateProductImagePath method. After setting up a new Product object with the retrieved product details, it logs these details and redirects the user to the AdminPage.jsp using the response object's sendRedirect method.

*Table 6: AddProducts Controller Class Method Description*

### 4.2.2 AdminFilter Controller Class

Method	Description
doFilter(ServletRequest request, ServletResponse response, FilterChain chain)	This method belongs to a filter class that implements the Filter interface. It's designed to verify the presence of an "adminEmail" attribute in the user session. If this attribute is missing, the user is redirected to the AdminLogin.jsp page accompanied by an error message stating "Do login First!". If the attribute is present, the request is allowed to continue to the next link in the filter chain by invoking the chain's doFilter() method. This security measure helps restrict access to the administrative parts of the application, ensuring that only authorized users can reach these areas.

*Table 7: AdminFilter Controller Clas Method Description*

### 4.2.3 AdminLogin Controller Class

Method	Description
service(HttpServletRequest request, HttpServletResponse response)	This method is responsible for administering the login process for administrators. It extracts the email and password from the HttpServletRequest. Using the DbConnection object, it checks these credentials by invoking the adminLogin method, which returns a ResultSet. If this ResultSet is not empty, it indicates that the login details are correct. The method then initiates a new HttpSession, assigning the loggedInId and adminEmail based on the authenticated admin's email. The session is configured to expire after five minutes of inactivity. Subsequently, the user is redirected to the administrative page. If the ResultSet is empty, indicating incorrect credentials, the method redirects the user back to the login page.

Table 8: AdminLogin Controller Class Method Description

### 4.2.4 AdminPage Controller Class

Method	Description
service(HttpServletRequest request, HttpServletResponse response)	This service method is tasked with retrieving and displaying a list of orders. It achieves this by using a DbConnection instance to fetch order data from the database. The method iterates over this data, noting each order's specific attributes. It then stores the order list as an attribute within the request object. Using a RequestDispatcher, the method forwards this enriched request to the AdminPage.jsp for viewing. A try-catch block is employed throughout to manage any ServletExceptions or IOExceptions, ensuring the request is properly forwarded to the admin page while handling potential exceptions.

Table 9: AdminPage Controller Class Method Description

#### 4.2.5 Category Controller Class

Method	Description
service(HttpServletRequest req, HttpServletResponse res)	This method processes an HTTP request that contains a "by" parameter specifying the desired sorting order for a list of products. It first establishes a connection to the database using the DbConnection class and invokes the sortBy method to obtain an ArrayList of Product objects, organized according to the specified sorting criteria. Subsequently, it assigns this sorted list of products as an attribute of the request. The method then uses a RequestDispatcher to forward the request to the CategoryPage.jsp, where the sorted products are displayed in accordance with the chosen order.

Table 10: Category Controller Class Method Description

#### 4.2.6 DeleteProducts Controller Class

Method	
service(HttpServletRequest request, HttpServletResponse response)	This method manages HTTP requests and responses within a protected environment. It retrieves a 'id' parameter from the incoming request and uses it to invoke the 'deleteParameter' function from the DbConnection class. Depending on whether the message ('msg') returned from the method is null or not, the method logs either a generic message or the specific 'msg' to the console. Following this, it fetches an updated list of all products from the database using DbConnection's 'fetchProducts' method and assigns this list as an attribute named 'productList' to the request. The method then sets up a RequestDispatcher aimed at "View/ManageProduct.jsp" and forwards both the request and response to this destination, facilitating the management and display of product information.

Table 11: DeleteProducts Controller Class Method Description

#### 4.2.7 Delete User Controller Class

Method	Description
service(HttpServletRequest request, HttpServletResponse response)	This method processes a request aimed at deleting a user's account. It begins by extracting the user's email from the request parameters and then employs a method from the DbConnection class to remove the user's details from the database. After executing the deletion, the method verifies whether the returned message from the database operation is null. If the message is indeed null, it logs a statement indicating this null status. If not, it logs the actual message returned by the database. The method appears to contain a coding oversight, as it references a variable 'res' towards the end without performing any operations on it, potentially indicating an unfinished or erroneous segment in the code.

*Table 12: DeleteUser Controller Class Method Description*



#### 4.2.8 Edit Products Controller Class

Method	Description
doGet(HttpServletRequest request, HttpServletResponse response)	This doGet method manages HTTP GET requests specifically designed to fetch data from the server. It starts by extracting the 'id' parameter from the request. Using this id, it establishes a DbConnection instance to obtain the detailed information of the product associated with this id. Once the product details are retrieved, they are assigned to the 'productDetails' attribute within the request object. Finally, the request is forwarded to the EditProducts.jsp page where the product details are displayed for further editing or review.

Table 13: EditProducts Controller Class Method Description

#### 4.2.9 EditProfile Controller Class

Method	Description
Service (HttpServletRequest, HttpServletResponse response)	This Java servlet method manages requests aimed at editing a user's profile. Initially, it extracts the 'email' parameter from the request object. Subsequently, it leverages this email to retrieve the user data from the database utilizing the DbConnection class. The retrieved user data is then encapsulated within a User object, which is subsequently added as an attribute to the request object. Finally, the method forwards the request to the "EditProfile.jsp" view, where the user's data is presented in a form for editing.

Table 14: EditProfile Controller Class

#### 4.2.10 ErrorHandle Controller Class

Method	Description
service(HttpServletRequest request, HttpServletResponse response)	This service method within a servlet is responsible for directing users to an error page whenever an error occurs. It requires two parameters: an HttpServletRequest and an HttpServletResponse. Within this method, the sendRedirect() function of the HttpServletResponse is utilized to reroute the

	<p>user to the <code>ErrorPage.jsp</code>, which is located within the <code>View</code> folder. This redirection is triggered whenever an error is detected in the application, ensuring that users are informed of the issue through a designated error notification page.</p>
--	--

*Table 15: ErrorHandler Controller Class Method Description*

## 5. Testing

### 5.1 Test 1: Sign Up Validation Testing

<b>Objective</b>	To verify the functionality of the admin login process.
<b>Action Performed</b>	Attempted to log in using valid admin credentials.
<b>Expected Result</b>	The system should grant access and redirect to the admin dashboard.
<b>Actual Result</b>	The login was successful, and the page redirected to the admin dashboard.
<b>Conclusion</b>	The admin login functionality is operating correctly.

Figure 33: Table of Test 1.

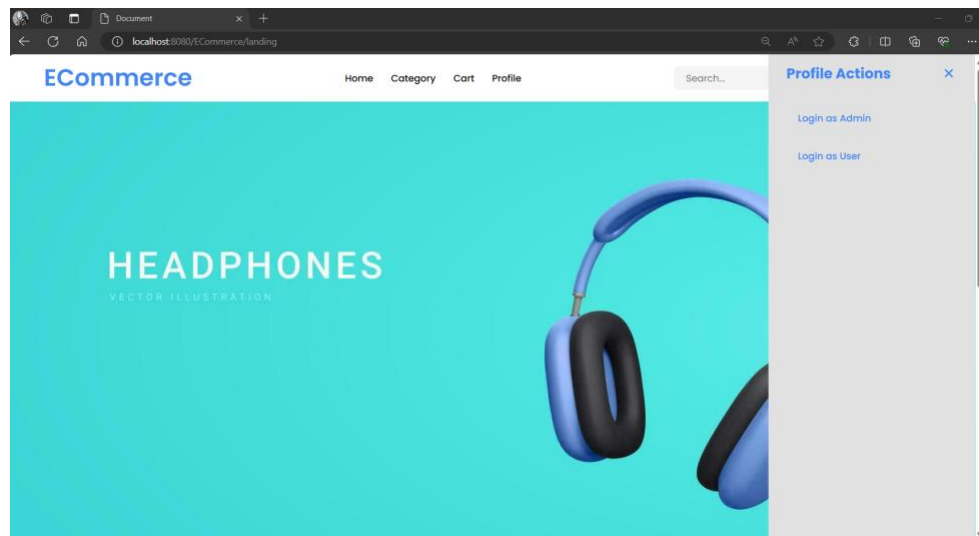


Figure 34: Ecommerce home screen.

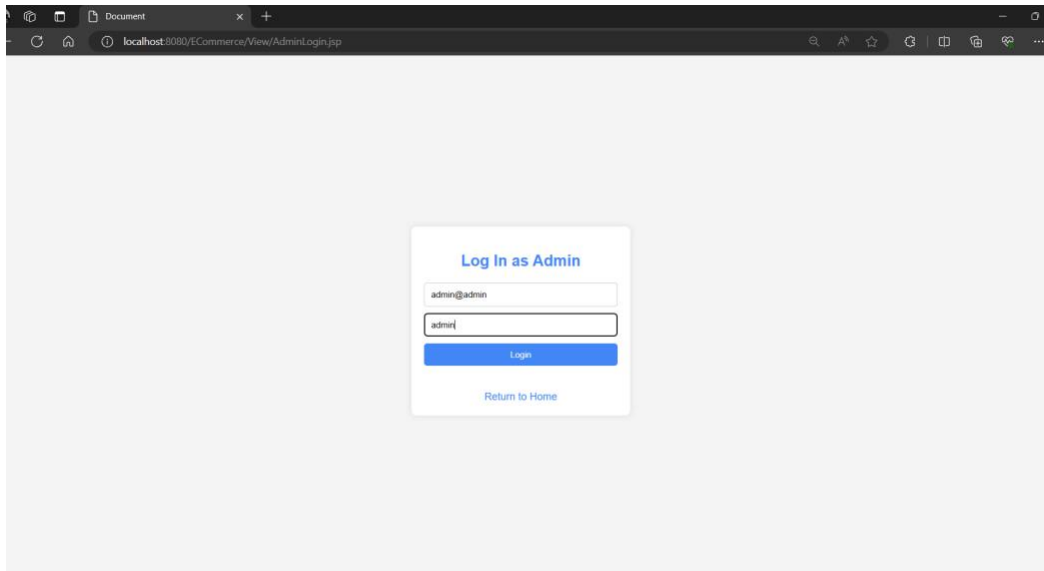


Figure 35: Login as Admin

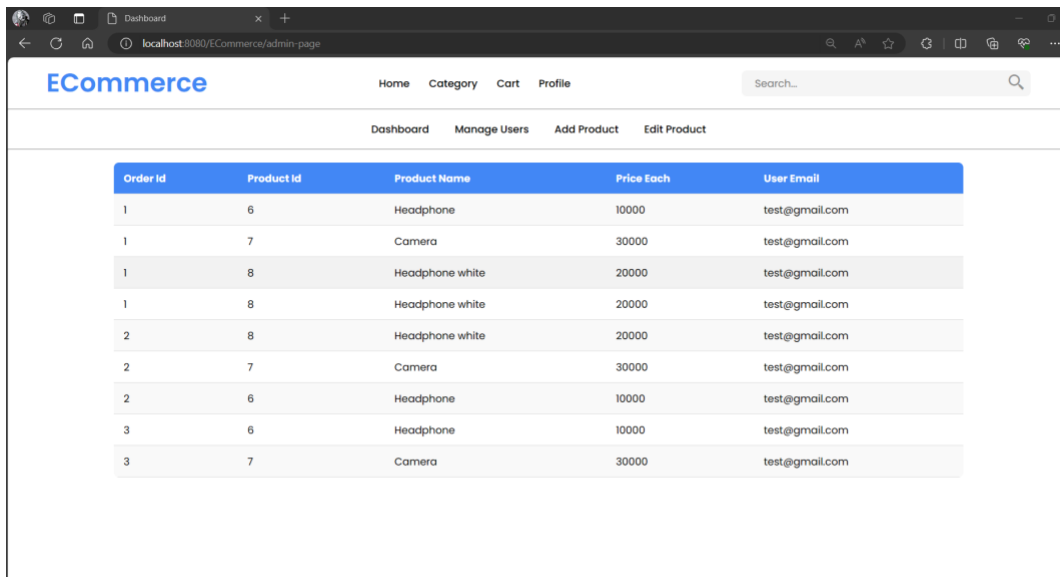


Figure 36: Admin Dashboard

## 5.2 Test 2: Sign Up Testing

<b>Objectives</b>	To verify the user registration process in the system.
<b>Action Performed</b>	A new user was registered using unique credentials.
<b>Expected Result</b>	The system should successfully register the user and display a confirmation message.
<b>Actual Result</b>	The user was successfully registered and a confirmation message was displayed.
<b>Conclusion</b>	The user registration functionality is working as expected.

Table 16 Table for Test 2

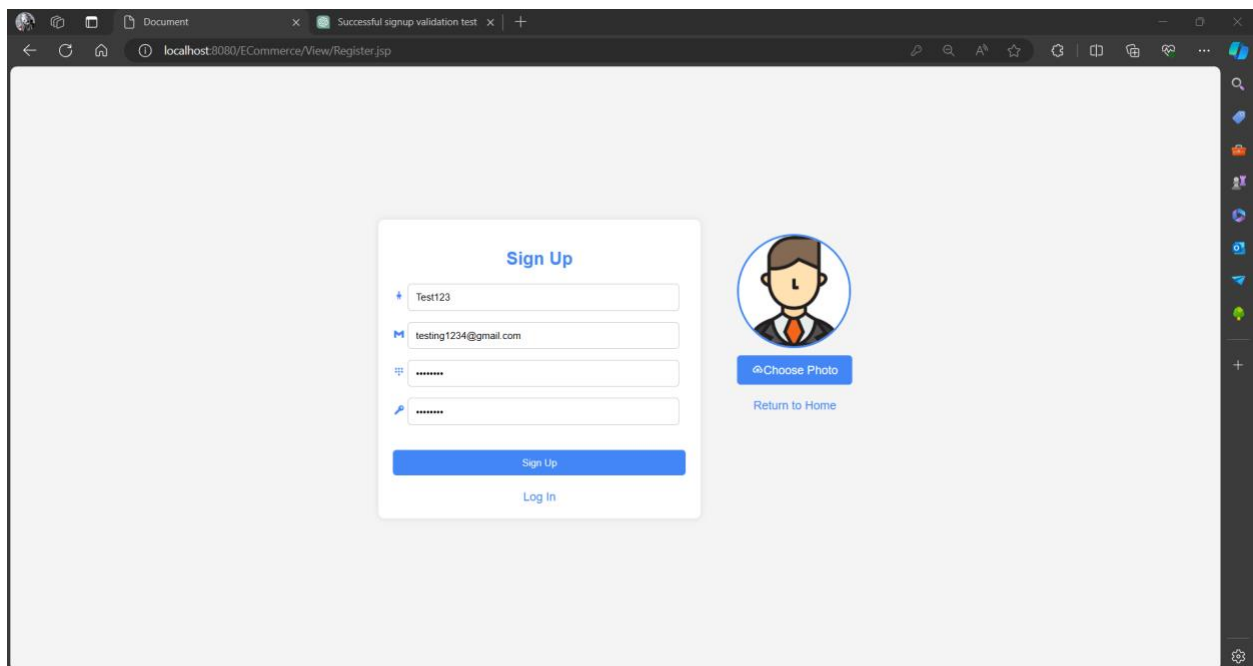











Figure 37: Register Member Screen

Extra options

			Id	Name	Email	Password	Image
<input type="checkbox"/>	 Edit	 Copy	 Delete	2	Test Useradada	test@gmail.com	+T89eI9cZn5HPiEGkNtp2g== userImage/test@gmail.com.png
<input type="checkbox"/>	 Edit	 Copy	 Delete	4	Test User	testing@gmail.com	+T89eI9cZn5HPiEGkNtp2g== UserImage/testing@gmail.com.png
<input type="checkbox"/>	 Edit	 Copy	 Delete	5	Test123	testing1234@gmail.com	+T89eI9cZn5HPiEGkNtp2g== UserImage/testing1234@gmail.com.png

### 5.3 Test 3: Adding Product to the Cart

<b>Objectives</b>	To test the functionality of adding products to the shopping cart.
<b>Action Performed</b>	Selected a product and added it to the cart.
<b>Expected Result</b>	The product should be successfully added to the cart, and the cart should update to reflect the addition.
<b>Actual Result</b>	The product was successfully added to the cart, and the cart was updated accordingly.
<b>Conclusion</b>	The functionality of adding products to the shopping cart is operating correctly.

Table 17 Table for Test 3

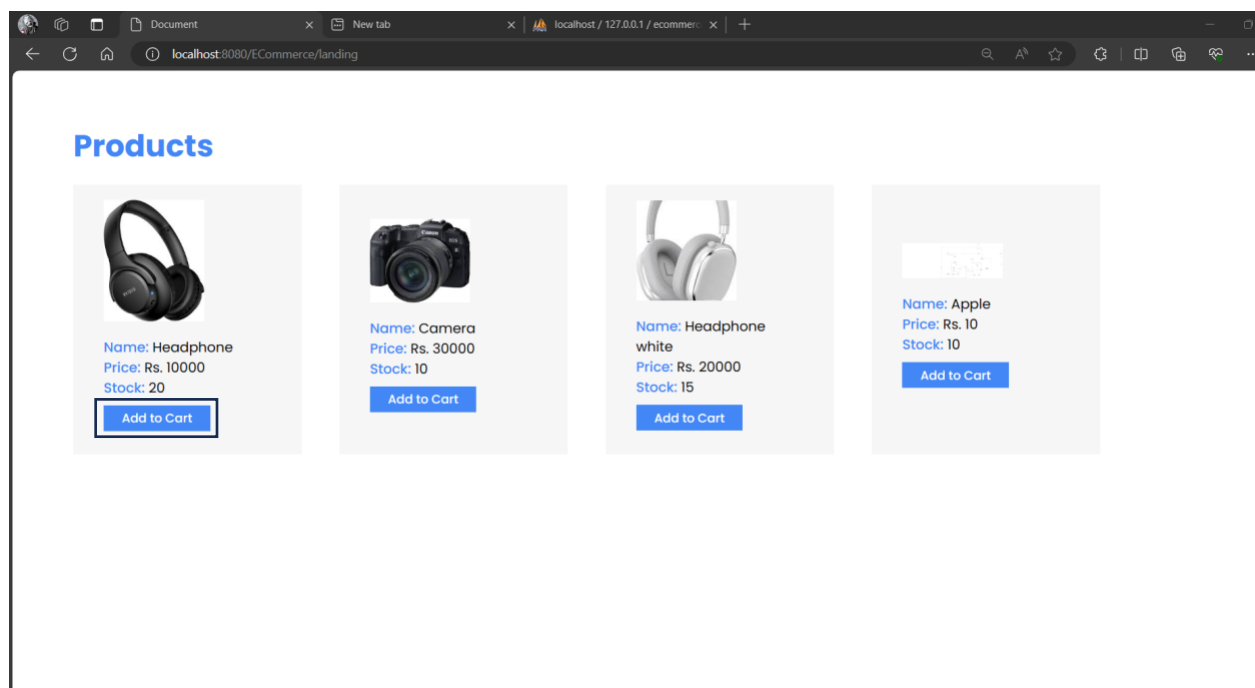


Figure 38: Add cart button pressed.

## Products



Name: Headphone  
Price: Rs. 10000  
Stock: 20

Add to Cart



Name: Camera  
Price: Rs. 30000  
Stock: 10

Add to Cart



Name: Headphone  
white  
Price: Rs. 20000  
Stock: 15


Add to Cart



Name: Apple  
Price: Rs. 10  
Stock: 10

Add to Cart

**Cart Items** ×

**Headphone**  
Price: Rs. 10000

Total Price: **10000**

**Order Now!**

Figure 39: Added to Cart

## 5.4 Test 4: Searching Item by name

<b>Objectives</b>	To test the search functionality by item name on the platform.
<b>Action Performed</b>	Entered a specific item name into the search bar and executed the search.
<b>Expected Result</b>	The system should display results relevant to the searched item name.
<b>Actual Result</b>	The search returned the correct items matching the input item name.
<b>Conclusion</b>	The search functionality by item name is working effectively.

Table 18 Table for Test 4

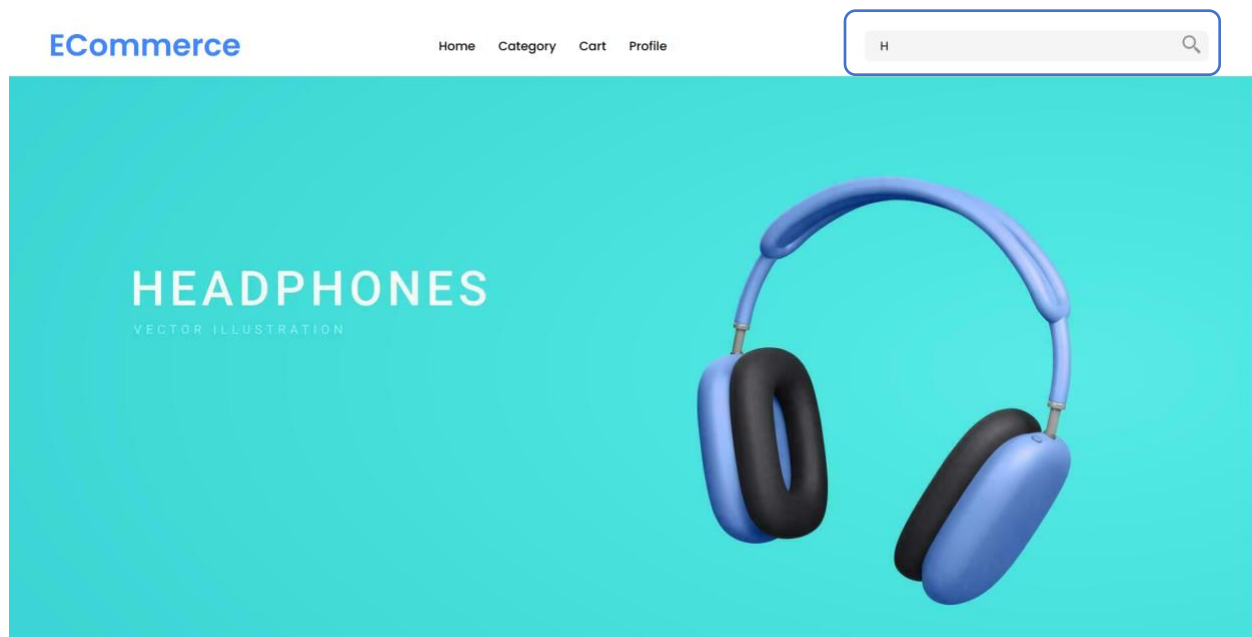


Figure 40: Searching H in search bar



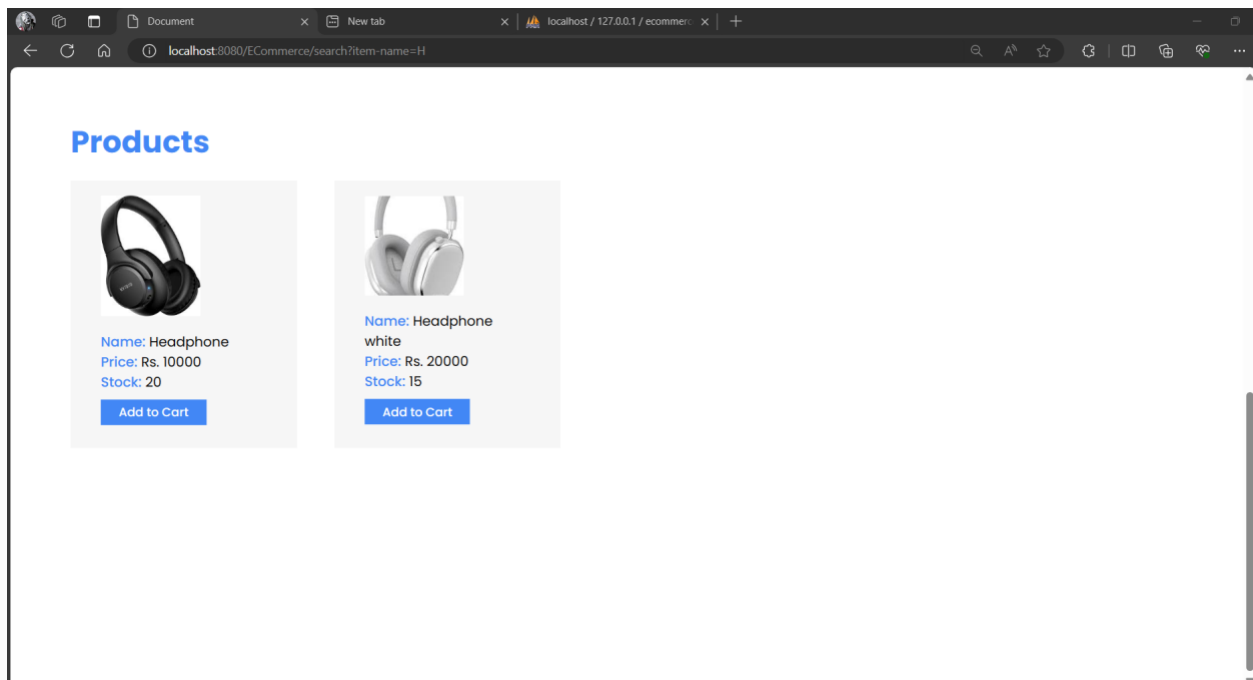


Figure 41: Searched Products

## 5.5 Test 5: Add Product Through Admin

<b>Objective</b>	To verify the process of adding a product to the inventory from the admin panel.
<b>Action Performed</b>	An admin entered details for a new product and submitted the form to add it to the inventory.
<b>Expected Outcome</b>	The system should successfully add the product and display a confirmation message indicating the product has been added.
<b>Actual Result</b>	The product was successfully added to the inventory, and a confirmation message was displayed.
<b>Conclusion</b>	The functionality for adding products from the admin panel is operating correctly.


Table 19 Table for Test 5

The screenshot displays a web browser window with the URL `localhost:8080/ECommerce/View/AddProduct.jsp`. The page features a navigation bar with the 'ECommerce' logo and links for Home, Category, Cart, and Profile. Below this, a secondary navigation bar includes Dashboard, Manage Users, Add Product (the active page), and Edit Product. The main content area contains a form for adding a new product. The form has three input fields: 'Product Name', 'Product Price', and 'Product Quantity'. To the right of these fields is a file upload section with a placeholder image, an 'Upload' button, and a 'Choose File' button. At the bottom of the form are 'Add Product' and 'Cancel' buttons. The browser's address bar shows the local host address and the page name.

**Product Name:**

**Product Price:**

**Product Quantity:**



Upload

Choose File Untitled Diagram (2).png


Figure 42: Add Products

				Product_Id	Product_Name	Product_Price	Product_Quantity	Product_Image
<input type="checkbox"/>				6	Headphone	10000	20	productImage/6.png
<input type="checkbox"/>				7	Camera	30000	10	productImage/7.png
<input type="checkbox"/>				8	Headphone white	20000	15	productImage/8.png
<input type="checkbox"/>				9	Apple	10	10	productImage/9.png
<input type="checkbox"/>				10	Product	1000	10	productImage/10.png


☐ Check all
 With selected:
 Edit
 Copy
 Delete
 Export

Figure 43: Added Product


## Products




Name: Headphone  
 Price: Rs. 10000  
 Stock: 20




Name: Camera  
 Price: Rs. 30000  
 Stock: 10



Name: Headphone white  
 Price: Rs. 20000  
 Stock: 15



Name: Apple  
 Price: Rs. 10  
 Stock: 10



Name: Product  
 Price: Rs. 1000  
 Stock: 10

Figure 44: Added Product Gets Displayed

## **6. Tools and Library Used**

In large-scale industries and during project development, access to specialized applications or software, commonly referred to as "tools," can be extremely beneficial. These tools can vary widely and include those for programming, project management, version control, and facilitating teamwork.

The term "libraries" refers to collections of pre-written code that can be incorporated into an existing software project to aid in its development. Libraries can be broadly classified into general-purpose or specific-purpose categories. General-purpose libraries cover a range of functions such as data manipulation, user interface design, and networking, among others.

There are various types of tools and libraries which have assisted in completing these projects which are:

### **6.1 Tools Used**

#### **6.1.1 Eclipse IDE**

Developers working with languages like Java, C++, Python, and PHP often choose Eclipse as their preferred IDE (Integrated Development Environment) because it is an open-source platform that enhances application development. Eclipse provides a wide array of tools, editors, and plugins, making software development faster and simpler. Its user-friendly UI aids in coding, compiling, and debugging, and features like code completion, syntax highlighting, auto formatting, and integration with version control systems save significant time. In our project, Eclipse IDE is primarily used for coding, employing libraries like JSTL and JSP. (eclipse, n.d.)

### **6.1.2 XAMPP**

XAMPP is a free, open-source web server solution that operates on multiple platforms. Its name represents Cross-Platform, Apache, MySQL, PHP, and Perl. This globally recognized server simplifies local web server development and testing. Originating as a complement to Apache, XAMPP's source code is publicly accessible for review and modification. It supports languages such as PHP and Perl, along with the MariaDB database, enabling developers to easily establish WAMP or LAMP stacks on their operating systems. XAMPP also supports popular applications like WordPress and Joomla, facilitating web development. (educba, n.d.)

### **6.1.3 Figma**

Figma is a cloud-based design tool that's become a standard in the industry due to its user-friendly interface and real-time collaboration capabilities. It allows designers to work together seamlessly on high-fidelity mockups, wireframes, and prototypes. One of Figma's key strengths is its facilitation of collaboration across different locations, enabling every team member to view and edit designs simultaneously, which increases efficiency and productivity.

## **6.2 Library Used**

### **6.2.1 MySQL**

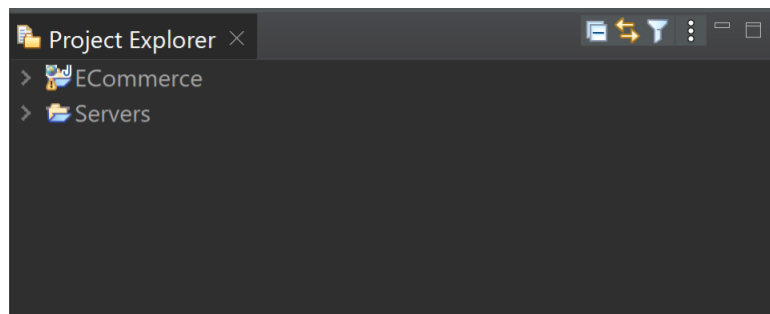
MySQL is a popular open-source database management system (DBMS) known for its performance, reliability, and user-friendliness. It utilizes Structured Query Language (SQL), the industry standard for database communication, allowing users to create, modify, and query databases. MySQL also includes features to manage access permissions and data security.

### **6.2.2 MySQL Jar Connector**

To access MySQL databases in Java, you can use the JDBC (Java Database Connectivity) driver found in the MySQL-connector-java JAR file. This connector is a library that facilitates a uniform interface for Java programs to connect to and interact with MySQL databases. The primary purpose of including the MySQL-connector-java JAR in a Java project's classpath is to enable its use within that project. This can be

achieved by either adding the JAR file as a dependency in the project's build configuration file or by placing it directly in the project's lib directory.

## 7. Development Process



*Figure 45: ECommerce workspace in eclipse.*

After establishing the development phases and setting milestones for the project, we decided on the frameworks that would be used throughout the project's development. We set up a version control system and created a repository to manage the code efficiently. The next step involved initiating the database and design processes. Initially, we defined the purpose and scope of the database, identifying key entities, characteristics, and relationships to store. This led to the creation of a conceptual design, offering a high-level view of the intended database structure, implemented using MySQL as part of a relational database management system (RDMS).

Once the database was set up and its structure finalized, we moved on to designing the website's user interface, creating wireframes and mockups. Actual development commenced with programming using HTML, CSS, and JavaScript. For dynamic content, we utilized JSP and servlets to render data from the backend. The project then progressed to the critical stages of testing and integration. We examined the various components for interdependencies and potential conflicts, employing integration strategies such as top-down, bottom-up, or a combination of both. Following successful integration, we conducted extensive tests for functionality, regression, performance, and security, addressing any bugs found to prepare the system for production.

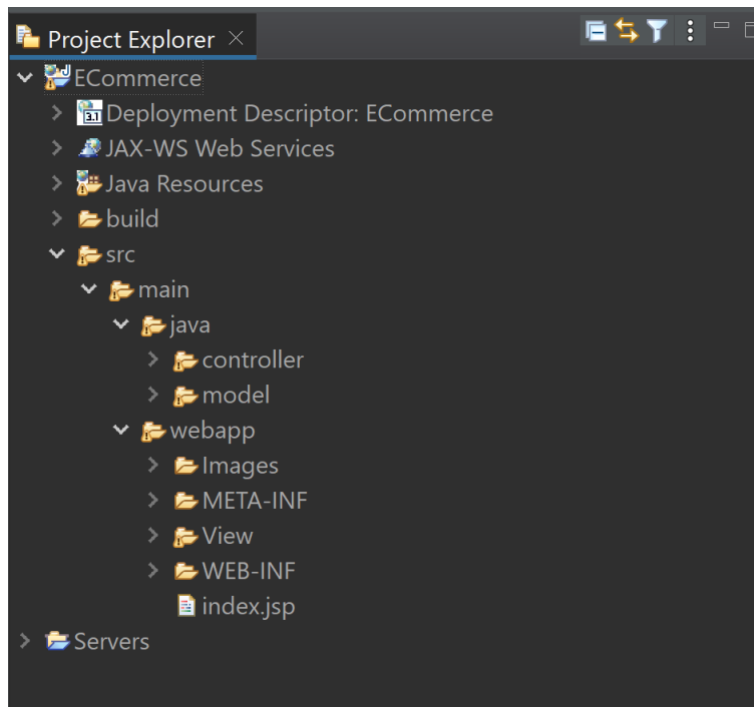


Figure 46: Model View and Controllers.

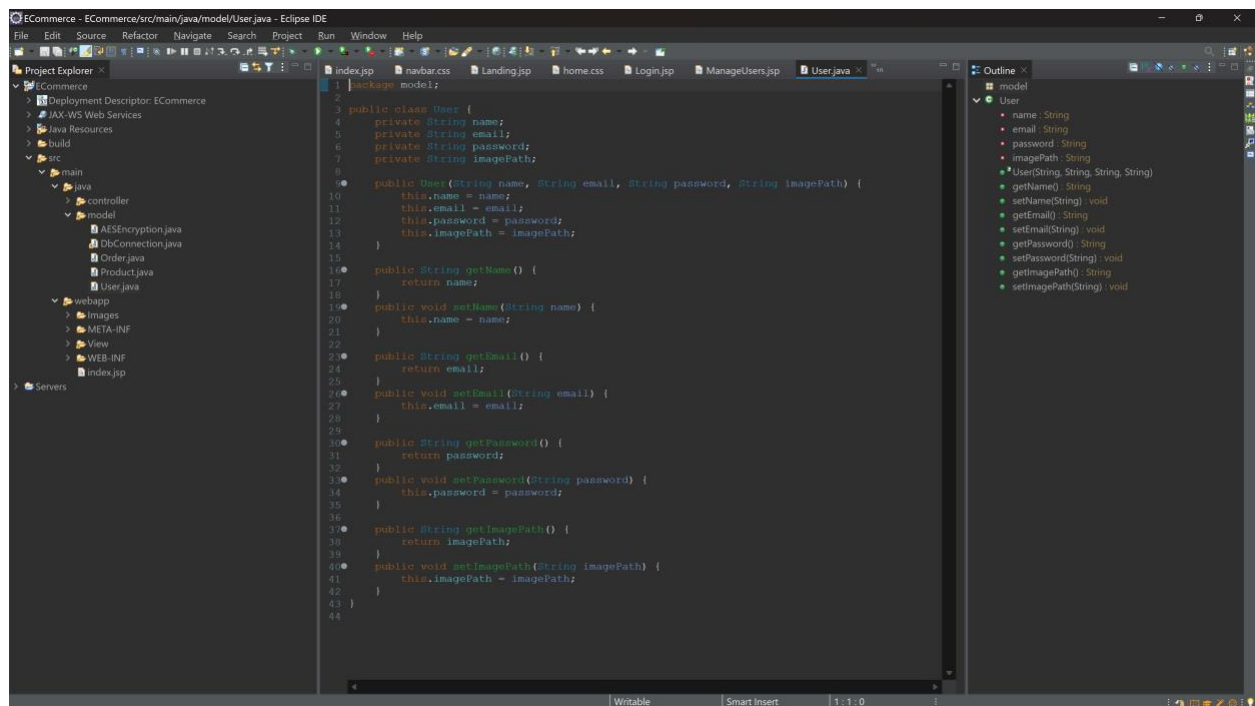


Figure 47: User Model.





The deployment process involved launching the software into a production environment, ensuring it was stable, reliable, and performed as expected. This phase required careful planning and execution, including building infrastructure, installing software packages, configuring servers, and setting up hardware. Throughout the project, we faced several challenges such as mastering a new IDE like Eclipse, using version control effectively, adhering to the MVC pattern, managing database schemas with complex relationships, and implementing advanced features like a search bar and filters for price and ratings. These hurdles were significant but provided valuable learning experiences.

## 8. Critical Analysis

The objective of this collaborative project is to develop a dependable and secure online clothing store using the Model-View-Controller (MVC) framework. The MVC approach structures the code to enhance understandability, scalability, and manageability. Key features of the website include a user login system, an admin page, and advanced product search and filtering options, which enhance the shopping experience.

The project goals are clearly outlined, focusing on building a robust e-commerce platform with MVC architecture and robust coding practices. We aim to provide a secure login system with encrypted password storage and features such as product displays with high-resolution images, prices, and stock information. The website will allow users to manage their shopping carts, update account settings, and reset passwords and profile pictures. Moreover, proper validation and exception handling measures are planned to ensure the website's security and functionality.

For the user interface, the team has employed HTML, CSS, and vanilla JavaScript. HTML provides the structural foundation, CSS manages the presentation, and JavaScript enables dynamic interactions. Using Figma, a UI design tool, the team collaborated on wireframes and prototypes to refine the design and user experience flows.

The backbone of this project is Java EE, which offers a solid and scalable foundation for developing enterprise-level applications. Various Java EE libraries and technologies are used to bring the site's features to life. The JavaServer Pages Standard Tag Library (JSTL) plays a vital role in standardizing common functionalities like iterations and conditional processing, which are used extensively to display dynamic data like product details and user profiles.

Servlets, part of the Java EE framework, handle HTTP requests and responses, forming the core of the project's controller package that interacts with the model package. This API is crucial for implementing functionalities such as session management and URL mapping.

The MySQL relational database management system is used for data storage, providing a secure and scalable solution. The project integrates the Java Database Connectivity (JDBC) API for consistent database connection, using prepared statements and parameterized queries to safeguard against SQL injection attacks.

Comprehensive validation and exception handling are implemented to respond appropriately to unforeseen circumstances and ensure data integrity. The project employs try-catch blocks to handle exceptions and input validation to sanitize user inputs.

Overall, this project leverages Java EE technologies and libraries to create a highly scalable and maintainable e-commerce platform. The systematic use of these technologies simplifies development and reduces the likelihood of errors. The team has devoted considerable effort to define and execute the project goals, emphasizing the MVC pattern and adherence to best coding practices. Careful attention to UI design and functional execution ensures the resulting web application is both visually appealing and robust.

## 9. Conclusion

In conclusion, the development of our E-Commerce website for selling electronic devices has been both challenging and rewarding, transcending the substantial time investment. The implementation of features like the login system, admin panel, and product search and filter options has resulted in a fully functional website that exemplifies our commitment and hard work.

Throughout the project, we adhered to the Model-View-Controller (MVC) pattern and maintained high programming standards to ensure the website was well-organized and easy to maintain. We incorporated robust validation and exception handling mechanisms to ensure errors were managed effectively.

Overall, this project has been a significant success for us, providing invaluable team experience in designing and developing a website using Java Enterprise. It has also honed our programming skills and taught us the importance of working efficiently and collaboratively.

## 10. References

eclipse, n.d. *Download Eclipse Technology.* [Online]  
Available at: <https://www.eclipse.org/downloads/>  
[Accessed 08 05 2023].

educba, n.d. *What is XAMPP?.* [Online]  
Available at: <https://www.educba.com/what-is-xampp/>  
[Accessed 08 05 2023].

Mairoca, D., 2023. *Make Use Of.* [Online]  
Available at: <https://www.makeuseof.com/what-is-figma-used-for/>  
[Accessed 25 March 2023].

tutorialspoint, 2023. *tutorialspoint.* [Online]  
Available at: [https://www.tutorialspoint.com/uml/uml\\_class\\_diagram.htm](https://www.tutorialspoint.com/uml/uml_class_diagram.htm)  
[Accessed 8 May 2023].

w3schools, 2023. *w3schools.* [Online]  
Available at: [https://www.w3schools.com/java/java\\_intro.asp](https://www.w3schools.com/java/java_intro.asp)  
[Accessed 8 May 2023].

