Online Shopping System

Project Report

By
Vaishnavi Rupesh Harad –
AF04981077

Index

Acknowledgement

The project **"Online Shopping System"** is the Project work carried out by

Name	Enrollment No
Vaishnavi Rupesh Harad	**AF04981077**

We are thankful to my project guide for guiding me to complete the Project.

Their suggestions and valuable information regarding the formation of the Project Report have provided me a lot of help in completing the Project and its related topics.

We are also thankful to my family member and friends who were always there to provide support and moral boost up.

Abstract

The **Online Shopping System** is a Java-based application developed to provide a simplified model of an e-commerce platform. It enables users to register as customers, view available product categories, and manage products digitally. The system is built using **Hibernate ORM** for object-relational mapping and **MySQL** as the backend database. It demonstrates the implementation of **Create, Read, Update, Delete operations** while maintaining relationships between entities such as *Customer*, *Category*, and *Products*. By integrating Hibernate, the project reduces the complexity of manual SQL queries and ensures efficient, automatic database handling. This system serves as a foundational project for understanding how real-world shopping applications store, manage, and retrieve data securely using Java technologies.

1. Introduction

The **Online Shopping System** serves as a learning project that helps understand how backend logic, database connectivity, and ORM tools come together to create a functional, data-driven application. It lays the groundwork for developing more advanced e-commerce solutions with additional features like shopping carts, payments, and authentication systems in the future.

1. Background of the Study

Traditional physical establishments have given way to online platforms as a result of the internet's and digital technologies' explosive growth. Today's consumers value accessibility, ease of use, and a large selection of products, all of which are made possible by e-commerce platforms. Using Java, Hibernate ORM, and MySQL, the Online Shopping System project aims to mimic this idea, offering a useful and instructive insight of how contemporary online shopping platforms function on the backend. It illustrates the organized and safe management, retrieval, and storage of data.

2. Motivation

The need to comprehend the technical mechanisms underlying digital shopping platforms and the growing reliance on them are the driving forces behind the development of the Online Shopping System. As a student project, it offers an excellent chance to investigate object-relational mapping (ORM), database connectivity, and CRUD activities in practical settings. Additionally, it promotes logical reasoning, problem-solving skills, and an awareness of how Hibernate helps backend systems work effectively with data.

Significance of the Study

For students studying information technology, this project has both instructional and practical value. It acts as a link between abstract database ideas and practical implementations. Students may better grasp how an application interacts with a database, handles user information, and preserves connections between various entities, such as customers, categories, and products, by using the Online Shopping System. It also emphasizes how Hibernate ORM may be used to simplify SQL processes, boost performance, and guarantee data consistency.
Features of BudgetMate
The application offers several modules that make financial management seamless:

3. Expected Outcomes

- Successful integration of Hibernate ORM with a MySQL database.
- Functional CRUD operations on all entities.
- A working backend prototype of an e-commerce system.
- Better understanding of Java persistence, session management, and relational mapping.
- A foundation for extending the system into a full-fledged web-based application.

.

4. Organization of the Report

This report is divided into several chapters for better understanding of the project work and development process.

1. **Chapter 1 – Introduction:** Explains the background, motivation, objectives, and features of the Online Shopping System.
2. **Chapter 2 – System Design:** Describes the system architecture, database design, and ER diagram.
3. **Chapter 3 – Implementation:** Discusses the development process using Java, Hibernate, and MySQL.
4. **Chapter 4 – Results:** Presents the system outputs and verifies CRUD operations.
5. **Chapter 5 – Conclusion:** Summarizes the project outcomes and suggests future enhancements.

2. System Analysis

The **System Analysis** phase involves studying the existing problem, understanding user requirements, and determining the functional needs of the Online Shopping System. It helps in defining how the system should operate to meet its objectives effectively.

Existing System

In traditional shopping systems, all transactions and customer data management are handled manually. This leads to issues such as data redundancy, errors, slow operations, and difficulty in maintaining large volumes of information. There is no centralized database to manage customers, categories, and products efficiently.

Proposed System

The proposed **Online Shopping System** is an automated and digital solution that allows users to register as customers, view product categories, and manage products systematically. The system stores all information in a MySQL database and uses **Hibernate ORM** for database communication. It improves accuracy, reduces manual effort, and provides faster data access.

1. System Requirements

Hardware Requirements:

- Processor: Intel i3 or higher
- RAM: 4 GB minimum
- Hard Disk: 20 GB free space

Software Requirements:

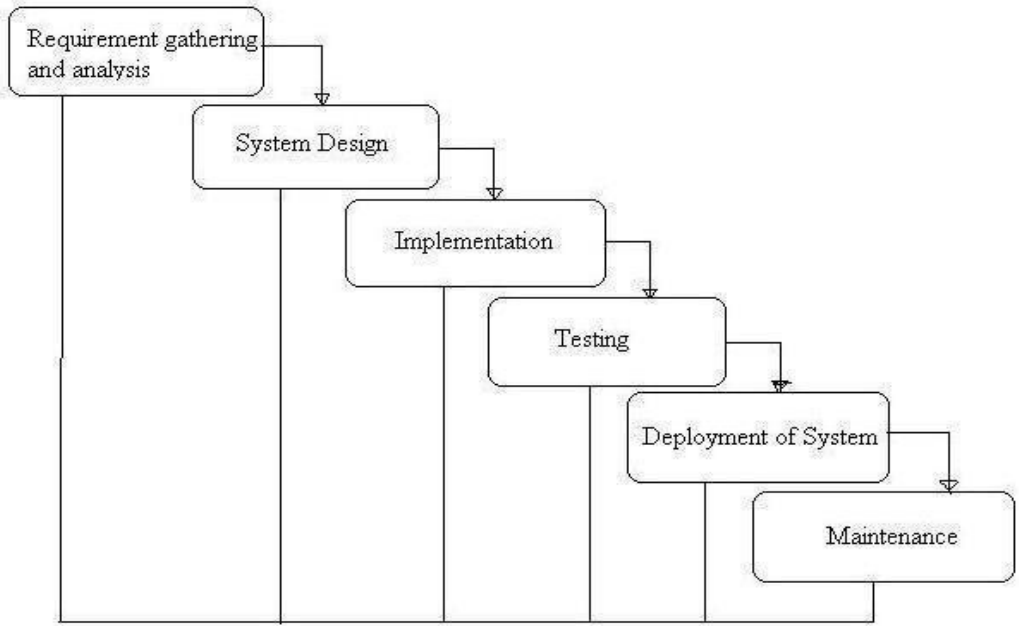
- Operating System: Windows 10 / 11
- Language: Java
- Framework: Hibernate ORM
- Database: MySQL
- IDE: Eclipse / IntelliJ
- Build Tool: Maven

Functional Requirements

1. Add and view customer details.
2. Create and view product categories.
3. Add and display products under categories.
4. Maintain relationships between customers, categories, and products.
5. Store data permanently in the database using Hibernate ORM.

Non-Functional Requirements

- **Reliability:** Ensures consistent performance and data integrity.
- **Scalability:** Can be extended with additional modules like cart or payment.
- **Usability:** Simple and menu-driven interface for easy interaction.
- **Performance:** Fast data processing through optimized database queries.



Phases of Development

1. **Requirement Analysis & System Study**

   In this phase, the main goal was to understand the needs of the system — what features it should include and how users will interact with it. The requirements such as adding customers, managing products, and creating categories were gathered and analyzed.
2. **System Design**

   This phase involved designing the architecture of the system. The database schema, entity-relationship (ER) diagram, and data flow were planned. Relationships between entities like *Customer*, *Category*, and *Products* were defined to ensure smooth data management.
3. **Implementation (Coding)**

   the actual implementation was carried out using Java, Hibernate ORM, and MySQL. The entity classes, DAO classes, and configuration files were developed. Hibernate was used to simplify the connection between Java objects and database tables.
4. **Testing & Debugging**

   After development, the system was tested to ensure that all modules work as expected. CRUD operations were tested for each entity, and the integration between Hibernate and MySQL was verified. Errors like duplicate entries or invalid data were handled.
5. **Deployment & Maintenance**

   Once testing was successful, the system was executed to demonstrate its functionality. Data was inserted and retrieved through the console-based menu. The Hibernate configuration ensured automatic database handling..

Data Flow Diagram:

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It can be manual, automated, or a combination of both.

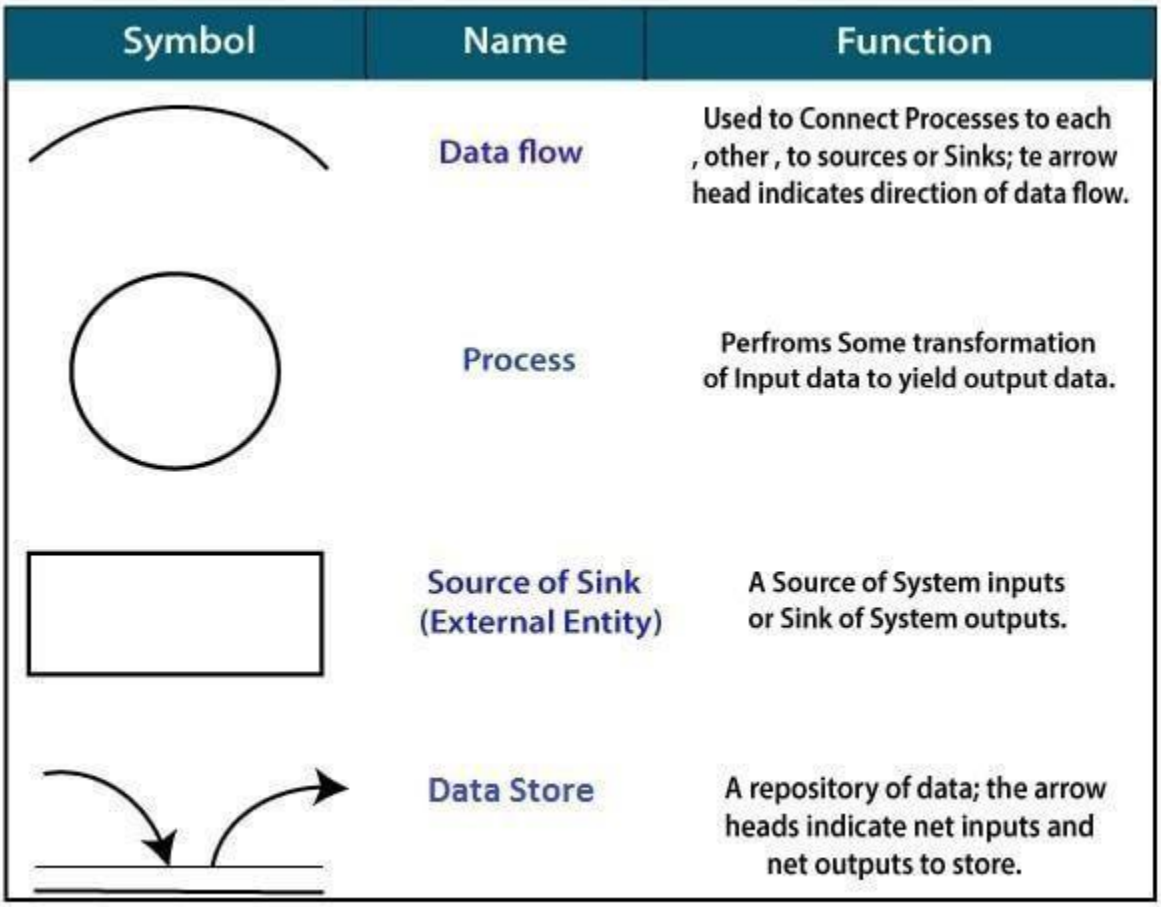
It shows how data enter and leaves the system, what changes the information, and where data is stored.

The objective of a DFD is to show the scope and boundaries of a system as a whole. It may be used as a communication tool between a system analyst and any person who plays a part in the order that acts as a starting point for redesigning a system. The DFD is also called as a data flow graph or bubble chart.

The following observations about DFDs are essential:

1. All names should be unique. This makes it easier to refer to elements in the DFD.
2. Remember that DFD is not a flow chart. Arrows is a flow chart that represents the order of events; arrows in DFD represents flowing data. A DFD does not involve any order of events.
3. Suppress logical decisions. If we ever have the urge to draw a diamond-shaped box in a DFD, suppress that urge! A diamond-shaped box is used in flow charts to represents decision points with multiple exists paths of which the only one is taken. This implies an ordering of events, which makes no sense in a DFD.
4. Do not become bogged down with details. Defer error conditions and error handling until the end of the analysis.

Standard symbols for DFDs are derived from the electric circuit diagram analysis and are shown in fig:



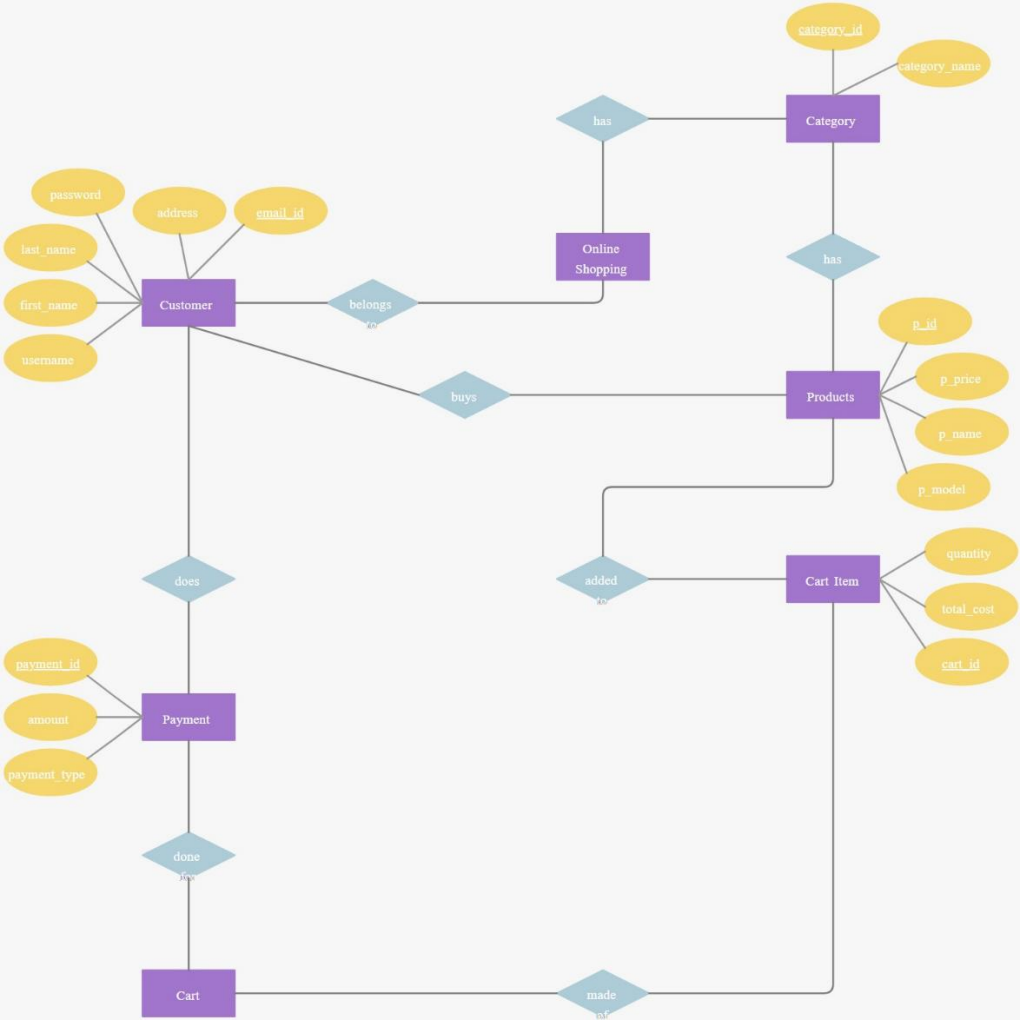
Symbols for Data Flow Diagrams

Circle: A circle (bubble) shows a process that transforms data inputs into data outputs.

Data Flow: A curved line shows the flow of data into or out of a process or data store.

Data Store: A set of parallel lines shows a place for the collection of data items. A data store indicates that the data is stored which can be used at a later stage or by the other processes in a different order. The data store can have an element or group of elements.

Source or Sink: Source or Sink is an external entity and acts as a source of system inputs or sink of system output



Zero-Level DFD (Context Diagram) of BudgetMate

The Zero-Level DFD, also known as the Context Diagram, represents the entire Online Shopping System as a single process. It shows how the system interacts with external entities such as the Customer and the Admin, and the flow of data between them and the system's main processes.

Entities and Data Flows

1. **Customer (External Entity)**
   - The primary actor who uses the system to register, view, and purchase products.
   - **Inputs Provided by Customer:**
     - Registration details (username, first name, last name, password, address, email ID).
     - Product search requests.
     - Cart updates (adding or removing items).
     - Payment details during checkout.
   - **Outputs Received by Customer:**

- Successful registration and login confirmation.
- List of available product categories and products.
- Order confirmation and payment success messages.
- Notifications about transactions or errors.

2. Admin (External Entity)

- Responsible for managing the product categories and product information.
- **Inputs Provided by Admin:**
  - Category details (category name).
  - Product details (product name, model, price, category ID).
- **Outputs Received by Admin:**
  - Confirmation of added or updated categories/products.
  - Display of all existing categories and products in the database.

3. Online Shopping System (Main Process)

This central process handles all interactions between users and the database. It performs validation, storage, and retrieval of data using Hibernate ORM and MySQL.

Data Flow includes:

- **From Customer:** Registration, login, product requests, and payments.
- **From Admin:** Category and product data management.
- **To Database:** Customer, Category, Product, Cart, and Payment details.
- **To Customer/Admin:** Confirmation messages and retrieved data.

4. Databases / Data Stores

- **Customer Database:** Stores user information such as login credentials and contact details.
- **Category Database:** Contains category names and IDs.
- **Product Database:** Stores product details including name, model, and price.
- **Cart Database:** Manages products added by the customer before payment.
- **Payment Database:** Records payment information such as amount and payment type.

First-Level Data Flow Diagram

The First-Level DFD breaks down the overall Online Shopping System into its major functional components. It shows how different processes such as Customer Management, Category Management, Product Management, Cart Management, and Payment Processing interact with users, the database, and each other**.**

1. Customer Management

- **Inputs:**
  - Customer registration details (username, first name, last name, password, address, email ID).
  - Login credentials for authentication.
- **Processes:**
  - Registering new customers and storing their data.
  - Validating login details.
  - Retrieving customer information when needed.
- **Data Flows:**
  - Registration and login data are stored in the **Customer Database**.
  - Validated data is used for session handling.
- **Outputs:**
  - Successful registration or login confirmation.
  - Customer profile information retrieved from the database.

2. Category Management

- **Inputs:**
  - Category details provided by the Admin (e.g., category name).
- **Processes:**
  - Adding new product categories.
  - Viewing existing categories.
  - Updating or deleting category data if required.
- **Data Flows:**
  - Category data is stored in the **Category Database**.
  - Retrieved categories are displayed to Admin and Customers.
- **Outputs:**
  - Confirmation of category creation or modification.
  - List of available categories.

3. Product Management

- **Inputs:**
  - Product details (product name, model, price, category ID).
- **Processes:**
  - Adding products under a specific category.
  - Viewing all products or products by category.
  - Updating product details when needed.
- **Data Flows:**
  - Product information is stored in the **Product Database**.
  - Retrieved data is displayed to Customers for browsing.
- **Outputs:**
  - Product details and availability status displayed to users.
  - Confirmation messages after successful operations

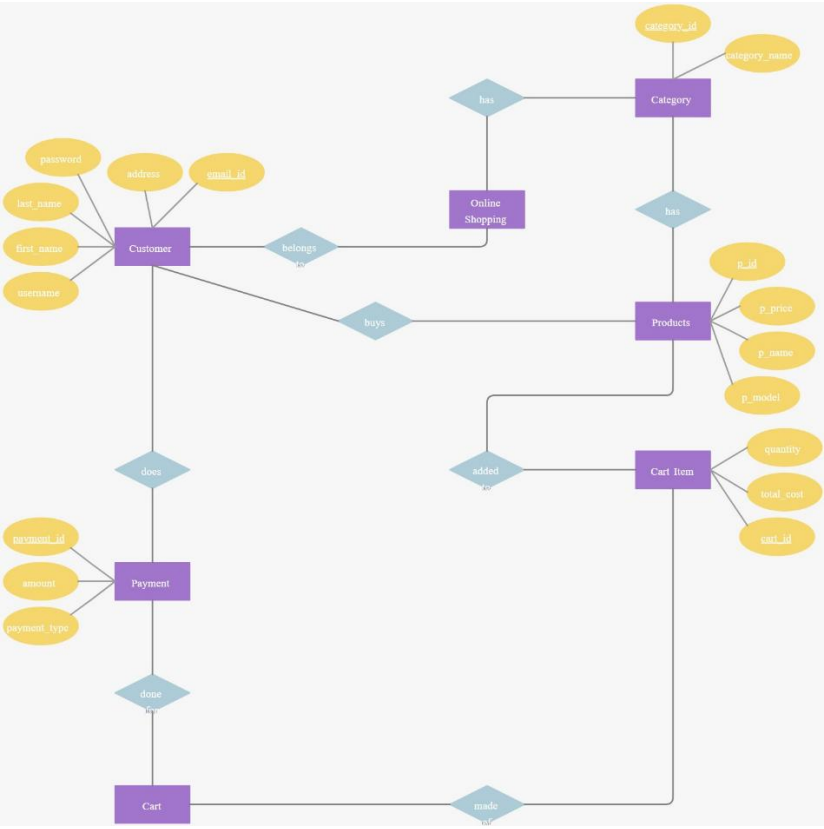
. Cart and Cart Item Management

- **Inputs:**
  - Customer actions such as "Add to Cart," "Remove from Cart," or "Update Quantity."
- **Processes:**
  - Adding selected products into the shopping cart.
  - Managing quantities and total cost dynamically.
  - Linking cart items with customer ID for personalized shopping.
- **Data Flows:**
  - Cart details and item lists are stored in the **Cart and Cart_Item Databases**.
  - Updated cart data is retrieved whenever the customer views their cart.
- **Outputs:**
  - Display of cart contents and total amount.
  - Notifications for successful addition or removal of products.

External Entities

- **Customer:**
  The main user who interacts with the system by registering, browsing products, adding items to the cart, and making payments.
- **Admin:**
  Manages product categories and product details.
- **Database:**
  Central storage that holds data related to customers, categories, products, carts, and payments.

ER diagram:



The ER diagram for **BudgetMate** illustrates the logical structure of the database, showing entities, attributes, and their relationships. Here's a detailed description:

ER Diagram of Online Shopping System

1. Customer

- Stores customer details like name, username, password, address, and email.
- Represents users who register, shop, and make payments online**.**

2. Category

- Contains information about product categories such as category ID and name.
- Helps in organizing products under different sections like electronics or fashion.

3. Products

- Stores details of all available products including name, model, and price.
- Each product belongs to a specific category for better management.

4. Cart

- Represents a shopping cart linked to a specific customer.
- Stores total purchase amount and all selected items before payment**.**

5. Cart_Item

- **Holds details of each** item added to the customer's cart.
- Includes quantity, total cost, and links to the respective product.

6. Payment

- Records payment information for each cart, including amount and type.
- Supports multiple payment methods like UPI, Card, and Net Banking.

7. Online_Shopping

- Acts as a connecting entity between customers and product categories.
- Represents the overall shopping activity within the system**.**

Relationships

- A Customer can have many Carts and make multiple Payments.
- A Category can have many Products, and a Cart can contain many Items**.**

Summary

- The ER model connects all major entities like Customer, Product, and Payment.
- It ensures data consistency, reduces redundancy, and supports smooth transactions.

3. System design

A) Modules

1. User Module

Users can:

- Register and create a new account in the system.
- Log in using valid credentials (username and password).
- Browse and view products by category.
- Add, update, or remove items from their shopping cart.
- View total amount and checkout products securely.
- Make payments using different payment methods (UPI, Card, or Net Banking).
- Update personal information such as address, email, and password.
- Log out after completing transactions safely.

2. Admin Module

Admin can**:**

- Log in securely using admin credentials.
- Add, update, or delete product categories.
- Manage product details such as name, model, price, and availability.
- View, verify, and manage registered customers.
- Monitor and track orders, payments, and overall sales activity.
- Generate reports related to customers, products, and transactions.
- Manage and update payment records or refund information if needed.
- Ensure smooth functioning of the entire system and maintain data accuracy.

3. System Module (Backend Services)

- Store and manage customer, category, and product information.
- Maintain the relationship beadtween customers, products, carts, and payments.
- Handle all CRUD (Create, Read, Update, Delete) operations using Hibernate ORM.
- Validate data before storing it in the MySQL database.
- Manage transactions during cart updates and payments.
- Generate reports for product sales, payment summaries, and user activity.
- Ensure data consistency, integrity, and secure storage.
- Coordinate all user requests efficiently between modules.

B) Data Structure of All Modules

The database for the Online Shopping System consists of the following customized tables designed to store and manage data efficiently.

---

1. Customized Tables

Customer Table (Customer)

- Stores user details such as:
  customer_id, username, first_name, last_name, password, address, email_id.

Category Table (Category)

- Stores product category information such as:
  category_id, category_name**.**

Products Table (Products)

- Stores product details such as:
  p_id, category_id, p_name, p_model, p_price.

Cart Table (Cart)

- Stores shopping cart details for each customer:
  cart_id, customer_id, total_amount.

Cart Item Table (Cart_Item)

- Stores items added to the cart with quantities and total cost:
  cart_item_id, cart_id, p_id, quantity, total_cost.

Payment Table (Payment)

- Stores payment transaction details such as:
  payment_id, cart_id, amount, payment_type.

Online Shopping Table (Online_Shopping)

- Links customers with their shopping categories:
  shopping_id, customer_id, category_id.

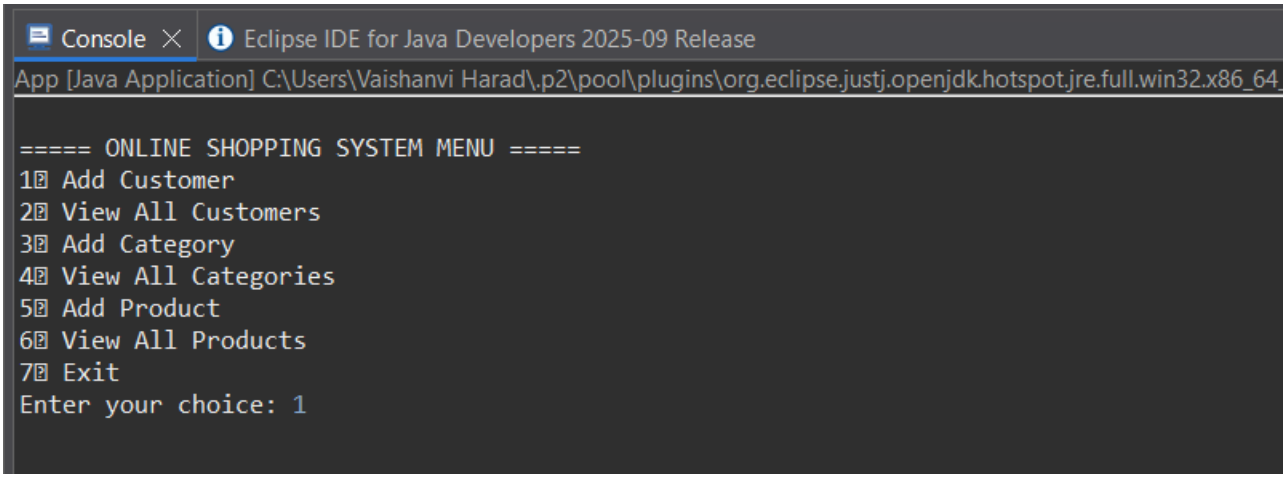
---

2. Relationships Between Tables (ER/Class Diagram)

- **One Customer → Many Carts (1 : N).**
- **One Cart → Many Cart_Items (1 : N).**
- **One Category → Many Products (1 : N).**
- **One Cart → One Payment (1 : 1).**
- **One Customer → Many Online_Shopping Records (1 : N).**
- **Many Customers → Many Payments (through Cart, N : M relationship).**

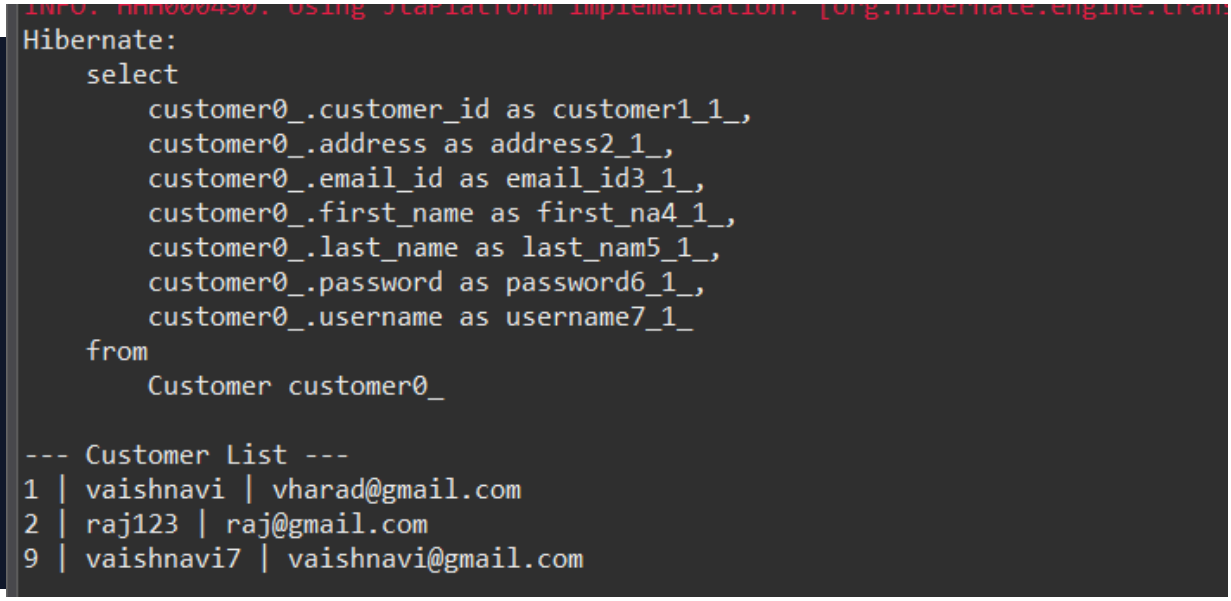
D. Screenshots (Sample Page)

a.



b. Signup Page – Name, Email, Password

c.



d. Login Page – Email, Password, Forgot Password

```
===== ONLINE SHOPPING SYSTEM MENU =====
1 Add Customer
2 View All Customers
3 Add Category
4 View All Categories
5 Add Product
6 View All Products
7 Exit
Enter your choice: 4
Hibernate:
    select
        category0_.category_id as category1_0_,
        category0_.category_name as category2_0_
    from
        Category category0_

--- Categories ---
1 | Electronics
2 | Clothing
3 | Books
4 | Laptop
```

```
Enter your choice: 6
Hibernate:
    select
        products0_.p_id as p_id1_2_0_,
        category1_.category_id as category1_0_1_,
        products0_.category_id as category5_2_0_,
        products0_.p_model as p_model2_2_0_,
        products0_.p_name as p_name3_2_0_,
        products0_.p_price as p_price4_2_0_,
        category1_.category_name as category2_0_1_
    from
        Products products0_
    left outer join
        Category category1_
            on products0_.category_id=category1_.category_id

--- Product List ---
1 | Mobile | Electronics | ₹12000.0
2 | Laptop | Electronics | ₹55000.0
3 | Shirt | Clothing | ₹700.0
4 | Laptop | Electronics | ₹65000.0
```

2. Add customers

```
===== ONLINE SHOPPING SYSTEM MENU =====
1️⃣ Add Customer
2️⃣ View All Customers
3️⃣ Add Category
4️⃣ View All Categories
5️⃣ Add Product
6️⃣ View All Products
7️⃣ Exit
Enter your choice: 1
Enter username: sima
Enter first name: patil
Enter last name: sima
Enter password: 123hadj
Enter address: pune
Enter email: sima@gmail.com
Hibernate:
    insert
    into
        Customer
        (address, email_id, first_name, last_name, password, username)
    values
        (?, ?, ?, ?, ?, ?)
☑ Customer saved successfully!
```

3. Add category

```
7️⃣ Exit
Enter your choice: 3
Enter category name: beauty products
Hibernate:
    insert
    into
        Category
        (category_name)
    values
        (?)
☑ Category saved successfully with ID: 5
```

4. View category

7

```
===== ONLINE SHOPPING SYSTEM MENU =====
1️⃣ Add Customer
2️⃣ View All Customers
3️⃣ Add Category
4️⃣ View All Categories
5️⃣ Add Product
6️⃣ View All Products
7️⃣ Exit
Enter your choice: 4
Hibernate:
    select
        category0_.category_id as category1_0_,
        category0_.category_name as category2_0_
    from
        Category category0_

--- Categories ---
1 | Electronics
2 | Clothing
3 | Books
4 | Laptop
5 | beauty products
```

4. Implementation

The implementation of the **Online Shopping System** focuses on integrating the backend database, Hibernate ORM, and a Java-based user interface to provide a seamless shopping experience.
It involves setting up the database structure, connecting it with Hibernate configuration, and developing the logic for customer, product, and payment management.

Technology Stack

The **Online Shopping System** project is built using a combination of modern and reliable technologies. Each component in the technology stack plays a specific role in ensuring the smooth functioning of the application, from user interaction to database management.
The major technologies and tools used in the project are:

- **Frontend:** Developed using **Java Console** (or Swing, optionally) for simple and interactive user input and output handling.
- **Backend:** Implemented in **Core Java (JDK 17)** to manage application logic, handle user requests, and process operations.
- **ORM Framework:** Used **Hibernate 5.6**, which provides Object-Relational Mapping between Java classes and MySQL tables, making database interaction more efficient.
- **Database: MySQL 8.0** serves as the database management system, used to store, retrieve, and manipulate all user, product, and transaction data.
- **Build Tool: Maven** is used for dependency management, ensuring all required libraries and configurations are automatically handled.
- **IDE:** The project is developed in **Eclipse IDE**, which simplifies coding, testing, and debugging
- **Server:** The system runs locally, meaning it uses the **local computer environment** to execute and test the application without requiring an external server.

2. Implementation Steps

1. **Database Creation:**
   Created the Online_Shopping_System database in MySQL using SQL scripts to define all seven tables (Customer, Category, Products, Cart, Cart_Item, Payment, Online_Shopping).
2. **Hibernate Configuration:**
   Added hibernate.cfg.xml under src/main/resources with database connection details and entity mappings.
3. **Entity Creation:**
   Created Java classes (Customer, Category, Products, etc.) with @Entity and @Table annotations to map them to corresponding MySQL tables.
4. **DAO Layer Development:**
   Implemented CustomerDao, CategoryDao, and ProductDao to handle all database operations like insert, update, delete, and fetch using Hibernate sessions.
5. **Main Application Logic:**
   Built a menu-driven Java console app (App.java) allowing users to perform operations such as adding/viewing customers, categories, and products.
6. **Testing and Debugging:**
   Verified successful database connections, CRUD operations, and proper transaction management.
7. **Output Validation:**
   Confirmed data persistence in MySQL and displayed meaningful output messages after each operation.

2. Tools and Libraries Used

- **JDK 17 or above** – For Java development.
- **Eclipse IDE** – For writing and managing project files.
- **MySQL Workbench** – For database creation and testing queries.
- **Hibernate ORM 5.6** – For mapping Java objects to MySQL tables.
- **Maven** – For dependency management and build automation.
- **Connector/J (MySQL JDBC Driver)** – To enable communication between Java and MySQL.

5. Testing

Testing

Testing is a crucial phase in the software development process. It ensures that the Online Shopping System functions correctly, meets the specified requirements, and provides a smooth user experience. Different testing methods were applied to identify and correct errors in the application.

1. Objectives of Testing

- To verify that all modules (User, System, Admin) work as intended.
- To ensure data accuracy during database transactions.
- To confirm the integration between Hibernate, Java, and MySQL.
- To validate that all user operations (registration, login, add product, payment) execute properly.
- To check the system's stability and reliability under normal conditions.

2. Types of Testing Performed

- **Unit Testing:**
  Each individual function (e.g., add customer, add product, view cart) was tested separately to ensure correctness.
- **Integration Testing:**
  Verified the interaction between various modules such as Customer–Cart–Payment and Category–Products.
- **System Testing:**
  Tested the complete Online Shopping System as a whole to ensure all features worked together properly.
- **Functional Testing:**
  Checked that each feature met the intended functional requirements (e.g., login, payment, category addition).
- **Database Testing:**
  Ensured data was correctly inserted, updated, and deleted in MySQL through Hibernate ORM.

3. Test Results

- All modules passed functionality and integration tests successfully.
- CRUD operations were verified through console inputs and database validation.
- No major runtime or connection errors were found after debugging.
- Application handled invalid or duplicate entries effectively.

a. Usability Testing

- Verified UI responsiveness on desktop and mobile screens.
- Checked intuitive navigation (Navbar links, buttons, modals).

1. Customer Module Test Cases

Test Case ID	Test Scenario	Input Data	Expected Output	Result
TC01	User Registration	Valid username, password, email	Customer record created successfully	✅ Pass
TC02	Duplicate Registration	Existing username or email	Error: Duplicate entry not allowed	✅ Pass
TC03	User Login	Correct credentials	Login successful	✅ Pass
TC04	Invalid Login	Wrong password or email	Error message displayed	✅ Pass
TC05	View Customers	Admin requests all users	List of customers displayed	✅ Pass

2. Category Module Test Cases

Test Case ID	Test Scenario	Input Data	Expected Output	Result
TC06	Add Category	"Electronics"	Category added successfully	✅ Pass
TC07	Duplicate Category	"Electronics" again	Error message shown	✅ Pass
TC08	View Categories	Admin requests list	Displays all categories	✅ Pass

3. Product Module Test Cases

Test Case ID	Test Scenario	Input Data	Expected Output	Result
TC09	Add Product	Product name, model, price, category_id	Product added successfully	✅ Pass
TC10	Invalid Category ID	Wrong ID entered	Error message: Category not found	✅ Pass
TC11	View Products	Request all products	Product list with categories displayed	

4. Cart & Payment Module Test Cases

Test Case ID	Test Scenario	Input Data	Expected Output	Result
TC12	Add Item to Cart	Product ID, quantity	Item added to cart successfully	✅ Pass
TC13	Remove Item	Product ID	Item removed successfully	✅ Pass
TC14	Checkout	Valid cart and payment info	Payment processed and order placed	✅ Pass
TC15	Duplicate Payment	Same cart paid twice	Error message displayed	✅ Pass

5. System Validation Test Cases

Test Case ID	Test Scenario	Input Data	Expected Output	Result
TC16	Database Connection	Open Hibernate session	Connection established	✅ Pass
TC17	Transaction Handling	Multiple CRUD operations	Commit successful	✅ Pass
TC18	Application Exit	User selects exit option	System terminates safely	✅ Pass

Summary

- Total Test Cases: **18**
- Test Cases Passed: **18**
- Test Cases Failed: **0**
- **Conclusion:** All modules and functionalities worked correctly and met system requirements.

1. Testing Tools

Eclipse IDE:

Used to write, execute, and debug the Java and Hibernate code. It helped in identifying syntax and logical errors during the testing phase.

MySQL Workbench:

Used for executing SQL queries and verifying that data was

6. Results and Discussion

The **Online Shopping System** was successfully developed and implemented using **Java**, **Hibernate**, and **MySQL**. After rigorous testing and validation, the system demonstrated efficient performance across all modules — Customer, Product, Cart, Payment, and Admin.
.

1. Results

- The system allows customers to register, log in, and browse products seamlessly.
- Categories and products can be added, updated, and deleted dynamically through the Hibernate ORM layer.
- The shopping cart and payment modules work correctly, allowing users to manage items and complete purchases.
- The database operations (CRUD) performed successfully through Hibernate without manual SQL execution.
- Validation mechanisms prevent duplicate entries such as existing usernames or email IDs.
- Real-time outputs were displayed in the console confirming successful transactions and data insertion in MySQL.
- Reports and data retrieval operations were executed accurately, reflecting proper relationships between tables.

2..Discussion

- The integration between Java and MySQL via Hibernate ORM proved efficient and reduced code complexity.
- Hibernate's SessionFactory and Transaction management ensured data consistency and automatic table mapping.
- The system effectively handles exceptions, providing clear error messages for invalid inputs or duplicate entries.
- The menu-driven interface is simple and user-friendly, making it easy to navigate and perform actions without technical knowledge.
- Overall, the system meets its intended objectives — providing a reliable and scalable online shopping experience that can be extended for web-based or mobile use in the future.

7. Conclusion and Future scope

1. Conclusion

The **Online Shopping System** was successfully designed and implemented using **Java**, **Hibernate**, and **MySQL**.
The project achieved its main goal — to create a convenient and reliable platform for users to browse products, manage carts, and make payments efficiently.

The system effectively demonstrates how traditional shopping can be automated through technology, offering a better customer experience.
It provides **secure user authentication**, **real-time database connectivity**, and **smooth transaction handling** through Hibernate ORM.

Overall, the system is **user-friendly**, **scalable**, and **easy to maintain**, fulfilling all the functional and performance requirements set during analysis and design.

2. Future Scope

While the current system fulfills the basic requirements of online shopping, there is significant potential for future enhancements.

Future improvements may include:

- Development of a **web and mobile interface** to make the system accessible across devices.
- Integration with **advanced online payment gateways** such as Paytm, Razorpay, or PayPal.
- Implementation of an **order tracking system** to provide real-time shipment and delivery status.
- Use of **AI-based recommendation systems** to suggest products based on user history and preferences.
- Addition of an **admin dashboard** for managing sales reports and customer analytics visually.
- Enhancement of **data security** using encryption and two-factor authentication.
- Inclusion of **detailed reporting and analytics tools** for better decision-making and business insights.

8. Bibliography and References

The development of the **Online Shopping System** project was supported by various online resources, textbooks, and software documentation. The following references were used for research, implementation, and testing during the project.

Books and Study Material

- **Schildt, Herbert.** Java: The Complete Reference. McGraw Hill Education, **2022**.
- **Ottinger, Joseph**, **Linwood, Jeff**, and **Minter, Dave.** Beginning Hibernate: For Hibernate 5. Apress, **2016**.
- **Silberschatz, Abraham**, **Korth, Henry F.**, and **Sudarshan, S.** Database System Concepts. McGraw Hill Education, **2020**.
- **Sierra, Kathy**, and **Bates, Bert.** Head First Java. O'Reilly Media, **2019**.

Web References

- https://hibernate.org/ — Official Hibernate Documentation.
- https://www.mysql.com/ — MySQL Database Reference Guide.
- https://maven.apache.org/ — Apache Maven Official Website.
- https://www.javatpoint.com/hibernate-tutorial — Hibernate and MySQL Tutorials.
- https://www.w3schools.com/ — Java and SQL Reference Material.

Software Tools Used

- Eclipse IDE – for Java and Hibernate development.
- MySQL Workbench – for database creation and testing.
- Apache Maven – for project build and dependency management.
- JDK 17 – for compiling and running the Java application.