

Fashion Store - E-Commerce Web Application

Adarsh Popat Patil

Computer Science & Engineering

S.E.A COLLEGE OF ENGINEERING & TECHNOLOGY

Niranjan S M

Computer Science & Engineering

S.E.A COLLEGE OF ENGINEERING & TECHNOLOGY

Sunil M Gudi

Computer Science & Engineering

S.E.A COLLEGE OF ENGINEERING & TECHNOLOGY

Chaitra Naik

Computer Science & Engineering

S.E.A COLLEGE OF ENGINEERING & TECHNOLOGY

Abhilash C S

Computer Science & Engineering

S.E.A COLLEGE OF ENGINEERING & TECHNOLOGY

Harisha C M

Computer Science & Engineering

S.E.A COLLEGE OF ENGINEERING & TECHNOLOGY

Kanayya V M

Computer Science & Engineering

S.E.A COLLEGE OF ENGINEERING & TECHNOLOGY

CHAPTER 1

COMPANY PROFILE

1.1 Introduction

1.2 Mission

The mission of the FashionStore – E-Commerce Web Application project is to develop a user- friendly and efficient e-commerce platform that transforms theoretical knowl-edge into practical web development skills. The project aims to provide hands-on experi-ence in designing and implementing a real-world online shopping system using modern web technologies.

The project achieves this through structured development, database integration, user interface design, and implementation of core functionalities such as authentication, product management, and order processing. It emphasizes building a complete system that reflects real-time industry applications.

1.3 Vision

The vision of the FashionStore – E-Commerce Web Application project is to create a scalable and responsive e-commerce platform that meets modern user expectations in online shopping.

By integrating frontend and backend technologies, the system prepares developers to build applications that align with current industry demands in e-commerce and web appli-cation development.

1.4 Services

The Fashion Stroe Web Application provides a wide range of functionalities in the e- commerce domain:

- User registration with name, email, and password fields
- Secure login authentication using email and hashed-comparable credentials stored
- Dynamic product catalogue — browse all available fashion products from the database
- Add-to-Cart functionality — add items to a session-scoped shopping cart
- Quantity update — increase or decrease the quantity of any item in the cart
- Remove-from-Cart — delete individual items from the shopping cart
- Monitoring user activities and orders
- Checkout and order completion with session cart clearance on purchase

1.4.1 Executive Summary

The FashionStore project demonstrates the end-to-end development of a production-ready e- commerce system by integrating frontend and backend technologies in a unified Java EE architecture. The application simulates a real-world online fashion store where registered users can explore products, manage a shopping cart, and complete purchases.

The project followed a structured MVC development approach covering database schema design, JDBC connectivity, Servlet-based controller logic, JSP view rendering, and session lifecycle management. It reflects professional development practices by focusing on usability, data integrity, and separation of concerns. The system is packaged as a Maven WAR project and deployed on Apache Tomcat, making it directly comparable to industry-standard Java web application deployment.

1.5 Conclusion

The Smart Placement Training and Assessment System Web Application plays a significant role in understanding real-world web development and e-commerce systems. It provides practical exposure to technologies used in building dynamic and interactive applications.

The project strengthens technical skills such as frontend design, backend development, database management, and user authentication. It also enhances problem-solving abilities and prepares students for careers in web development and software engineering. Overall, the project serves as a strong foundation for developing scalable and user-centric applications in the e-commerce domain.

CHAPTER 2

ABOUT COMPANY DEPARTMENT

1.1 Overview of Department

Tap Academy is a well-established technical training institute that focuses on providing industry-oriented education in software development. The organization is structured into multiple departments, each responsible for specific aspects of training, project development, evaluation, and student career growth. These departments work collaboratively to ensure that students gain both theoretical knowledge and practical experience required for the IT industry. The internship program at Tap Academy is designed to simulate real-world software development environments. It emphasizes hands-on learning, project-based development, and

continuous evaluation to prepare students for professional roles.

1.1.1 Full Stack Development Department

The Full Stack Development Department plays a crucial role in delivering technical training to students. This department focuses on both frontend and backend technologies, ensuring that students understand the complete lifecycle of web application development.

It conducts regular coding sessions, live demonstrations, and hands-on workshops to strengthen programming skills. Students are trained in technologies such as HTML, CSS, Bootstrap, JavaScript, Java, Servlets, JSP, and MySQL.

The department also guides students in developing real-time projects like the Smart Placement Training and Assessment System web application. Through continuous mentorship and practical exercises, it ensures that students gain confidence in building scalable and dynamic web applications.

1.1.2 Curriculum & Research Department

The Curriculum and Research Department is responsible for designing and updating the training content based on current industry trends and requirements. It continuously evaluates emerging technologies and integrates them into the curriculum.

This department ensures that students are trained in relevant tools and frameworks that are widely used in the software industry. It also prepares structured learning paths that include theoretical concepts, coding exercises, and project work.

By aligning the curriculum with industry standards, the department helps students stay competitive and job-ready in a rapidly evolving technological environment.

1.1.3 Project Development Department

The Project Development Department focuses on assigning and guiding students through real-time projects during their internship. These projects are designed to simulate real-world applications and provide practical exposure to software development.

Students are encouraged to work on projects such as e-commerce systems, including the Smart Placement Training and Assessment System application. The department provides step-by-step guidance on implementing features like user authentication, product management, shopping cart functionality, and database integration.

Mentors regularly monitor progress, provide feedback, and help students overcome technical challenges during development.

1.1.4 Evaluation & Quality Department

The Evaluation and Quality Department is responsible for assessing student performance throughout the internship. It ensures that the projects meet industry standards in terms of functionality, usability, and code quality.

The evaluation process is structured and comprehensive, focusing on multiple aspects of software development:

1.1.4.1 **Functionality:** Ensuring that all features of the application work correctly
1.1.4.2 **UI/UX Design:** Evaluating the visual design and user experience of the application
1.1.4.3 **Code Quality:** Checking code structure, readability, and best practices
1.1.4.4 **Documentation:** Assessing the quality of project reports and technical

documentation Regular feedback is provided to students to help them improve their skills and deliver high-quality projects.

1.1.5 Placement & Career Support Department

The Placement and Career Support Department plays a vital role in preparing students for job opportunities in the IT industry. It provides various training sessions focused on improving employability skills.

This department conducts mock interviews, resume-building workshops, and aptitude training sessions. It also organizes placement drives and connects students with potential employers.

By focusing on both technical and soft skills, the department ensures that students are well-prepared to succeed in recruitment processes.

1.1.6 Support & Administration

The Support and Administration Department manages all operational aspects of the internship program. It handles student registration, onboarding, attendance tracking, and documentation processes.

This department ensures smooth coordination between trainers, mentors, and students.

It also maintains records of student progress and internship activities.

Effective administrative support contributes to the overall success of the training and internship program.

1.2 Organizational Structure

The organizational structure of Tap Academy follows a hierarchical model that ensures efficient management and coordination of all activities.

- Founder / Director – Responsible for overall vision and strategic decisions
- Training Department Head – Oversees all technical training programs

- Technical Trainers – Deliver training in various technologies
- Project Mentors – Guide students in project development
- Evaluation and Quality Team – Assesses student performance and project quality
- Placement and Career Support Team – Handles job preparation and placements
- Administrative Staff – Manages operations and coordination
- Interns / Students – Participate in training and project development

All these roles work together to provide a structured learning environment and ensure that students gain industry-relevant skills.

CHAPTER 3

TASKS PERFORMED

During the internship at Tap Academy, several tasks were performed as part of the development of the Smart Placement Training and Assessment System Web Application. The tasks involved both frontend and backend development, along with database integration and testing.

- Understanding the project requirements and analyzing system functionality
- Designing the overall architecture of the Smart Placement Training and Assessment System application
- Creating database schema and tables using MySQL
- Writing SQL queries for data insertion, retrieval, and updates
- Developing user interface using HTML, CSS, and Bootstrap
- Developing backend logic using Java Servlets and JSP
- Implementing user registration and login authentication system
- Creating session management for secure user access
- Implementing add-to-cart functionality for selected products
- Deploying the application on Apache Tomcat Server
- Performing final testing and validation of all modules

3.1 Work Schedule

During the internship period, a structured learning and development schedule was followed. The internship focused on full-stack web development, including Core Java, Object-Oriented Programming (OOP), Exception Handling, MySQL database management, and frontend technologies such as HTML, CSS, and Bootstrap.

The entire internship was dedicated to learning fundamental programming concepts and applying them to develop the **Smart Placement Training and Assessment System Web Application**, an e-commerce platform for online shopping.

Table 3.1 Work Schedule

Month	Week	Task Performed
Month 1	Week 1	Orientation, onboarding process, introduction to internship program, and setup of development environment including Java and Eclipse IDE.
Month 1	Week 2	Learned Core Java fundamentals including variables, data types, operators, control statements, loops, and basic programming concepts.
Month 1	Week 3	Studied Object-Oriented Programming concepts such as classes, objects, inheritance, polymorphism, abstraction, and encapsulation with practical examples.
Month 1	Week 4	Learned Exception Handling concepts including try-catch blocks, finally block, throw and throws keywords, and custom exceptions.
Month 2	Week 5	Introduction to MySQL database, installation, and basic SQL queries such as CREATE, INSERT, SELECT, UPDATE, and DELETE operations.
Month 2	Week 6	Designed database schema for Smart Placement Training and Assessment System project including tables for users, products, orders, and cart management.
Month 2	Week 7	Practiced advanced SQL queries including joins, constraints, and normalization techniques for efficient database design.
Month 2	Week 8	Introduction to frontend development using HTML and CSS, including webpage structure, styling, and layout design.

Month	Week	Task Performed
Month 3	Week 9	Learned Bootstrap framework for responsive web design and created UI components such as navigation bar, product cards, and forms.
Month 3	Week 10	Designed Smart Placement Training and Assessment System homepage and product listing pages with proper layout and styling.
Month 3	Week 11	Developed user interface for login, registration, and product display pages.
Month 3	Week 12	Integrated frontend pages with backend concepts and prepared project structure for implementation.
Month 4	Week 13	Developed backend logic using Java Servlets and JSP for handling user requests and responses.
Month 4	Week 14	Implemented user authentication system and connected application with MySQL database.
Month 4	Week 15	Developed core functionalities such as add-to-cart, product management, and order processing.
Month 4	Week 16	Performed testing, debugging, and final project implementation. Prepared documentation and internship report.

Table 3.1 summarizes the complete internship work schedule followed during the training period.

The internship was divided into four structured months focusing on programming fundamentals, database management, frontend design, and full-stack web application development.

The initial phase concentrated on Core Java concepts, Object-Oriented Programming principles, and Exception Handling to build a strong programming foundation.

The intermediate phase focused on MySQL database design, SQL queries, and frontend development using HTML, CSS, and Bootstrap for creating responsive user interfaces.

The final phase involved integrating frontend and backend technologies, implementing business logic using Servlets and JSP, and developing the complete Smart Placement Training and Assessment System web application.

This planned schedule ensured continuous learning, practical exposure to real-world technologies, and successful completion of the Smart Placement Training and Assessment System project.

3.2 Introduction

The FashionStore E-Commerce Web Application was developed as a full-stack Java web project to simulate a real-world online fashion retail platform. The primary goal was to build a complete, functional system covering all major components of an e-commerce backend — user authentication, product management, cart operations, and order processing — using industry-standard Java EE technologies.

Traditional e-commerce training projects often remain theoretical. This project addressed that gap by implementing every feature end-to-end: from a relational MySQL database with properly designed tables, through a JDBC-based DAO layer, Servlet-based business logic controllers, all the way to JSP-rendered views styled with Bootstrap.

The application is architected following the MVC (Model-View-Controller) pattern. Java POJOs represent the domain model (Product, CartItem), Servlets act as controllers handling HTTP requests and responses, JSP pages render the view, and ProductDAO encapsulates all database interaction. Session management via HttpSession is used to maintain the shopping cart and user identity between page requests.

The system follows the MVC (Model-View-Controller) architectural pattern throughout. Java POJO classes — Product and CartItem — form the Model layer. Servlet classes in the com.fashionstore.servlet package act as Controllers, handling all business logic and session operations. JSP pages serve as the View layer, rendering HTML responses that are sent back to the browser.

3.3 System Requirements

The Smart Placement Training and Assessment System web application requires specific features, performance standards, and software tools for successful operation. These requirements are categorized into Functional Requirements, Non-Functional Requirements, and Software Requirements.

3.3.1 Functional Requirements

Functionality	Description
User Registration	New users can register by providing name, email, and password.
User Login	Registered users authenticate with email + password.
Product Catalogue	The homepage dynamically fetches all products from the products table using ProductDAO.
Product Detail View	Individual product details (name, description, price, image) are retrieved by ProductDAO.
Add to Cart	AddToCartServlet retrieves the product by ID, checks if it is already in the session cart
Remove from Cart	RemoveFromCartServlet uses List.removeIf() to delete the matching CartItem from the session
Update Cart Quantity	UpdateQuantityServlet accepts productId and new quantity via POST, locates the CartItem i

Table 3.2 Functional Requirements

Functionality	Description
Checkout / Order	CheckoutServlet removes the cart attribute from the session (simulating order placement) .
Complete Order	CompleteOrderServlet provides an additional cart-clearance path and redirects the user to homepage.
Logout	LogoutServlet calls session.invalidate() to destroy the entire session (cart + user data) .
Data Management	The system shall store and manage user and content data using a SQL database.
Session Management	The system shall maintain user sessions for secure and continuous interaction.

Table 3.2 presents the core functional requirements of the FashionStore E-Commerce Web Application System application.

These requirements define the essential operations of the system. User registration and login ensure secure access to the application. The tutorials, technical questions, and company questions modules help students prepare effectively for placements.

The internship and resume sections provide career guidance and opportunities. Together, these functional requirements ensure that the system provides a complete, interactive, and user- friendly platform for placement preparation.

3.3.2 Non-Functional Requirements

Requirement	Description
Performance	The application shall load pages and modules within 2–3 seconds under normal internet conditions.
Reliability	The system shall ensure that user data and content stored in the SQL database is consistent and secure.
Availability	The web application shall be accessible anytime with minimal downtime.
Scalability	The system shall support multiple users accessing training content simultaneously.
Security	User authentication and data storage shall be secured to prevent unauthorized access.

Maintainability	The system shall follow a modular design to allow easy updates and improvements.
Portability	The application shall work across multiple browsers such as Chrome, Edge, and Firefox.
Usability	The user interface shall be simple, intuitive, and easy to navigate.
Responsiveness	The system shall provide a responsive design for desktops, tablets, and mobile devices.

Table 3.3 Non-Functional Requirements

Table 3.3 outlines the non-functional requirements of the Online Placement Training and Assessment System.

These requirements define the quality aspects of the system. Performance ensures smooth interaction, while reliability ensures proper data storage in the SQL database.

Security protects user data, and scalability allows the system to handle multiple users. Usability and responsiveness ensure that the system is easy to use across different devices. Together, these requirements ensure that the system is efficient, secure, and user-friendly.

3.3.3 Software Requirements

Component	Specification
Programming Language	Java 21
IDE	Eclipse IDE
Frontend Technologies	HTML5, CSS3, JavaScript, Bootstrap 5
Backend Technology	Jakarta Servlet API 6.0, JSP (JavaServer Pages)
Database	MySQL — Database: fashionstore_db
JDBC Driver	mysql-connector-j 8.3.0 (via Maven dependency)
Application Server	Apache Tomcat

Table 3.4 Software Requirements

Table 3.4 shows the software tools and technologies used for developing the Online Placement Training and Assessment System web application.

3.4 System Design

The Fashion Store application was designed following the classic **MVC**

(Model-View-Controller) architectural pattern to ensure a clean separation of concerns, ease of maintenance, and scalability. The system is structured into three distinct layers:

Model Layer — Java POJOs (Product, CartItem) represent the domain objects. The ProductDAO class encapsulates all SQL queries using PreparedStatement, isolating database access from business logic. The DBConnection utility class manages JDBC connections to the MySQL fashionstore_db database.

Controller Layer — Java Servlets (in the com.fashionstore.servlet package) handle all incoming HTTP requests. Each Servlet is mapped to a specific URL pattern via the @WebServlet annotation and is responsible for one logical operation (login, register, add to cart, etc.).

View Layer — JSP pages render the HTML responses, using EL expressions and JSTL to display data forwarded by the Servlets. Bootstrap 5 provides a responsive, mobile-friendly layout across all pages.

3.4.1 System Architecture

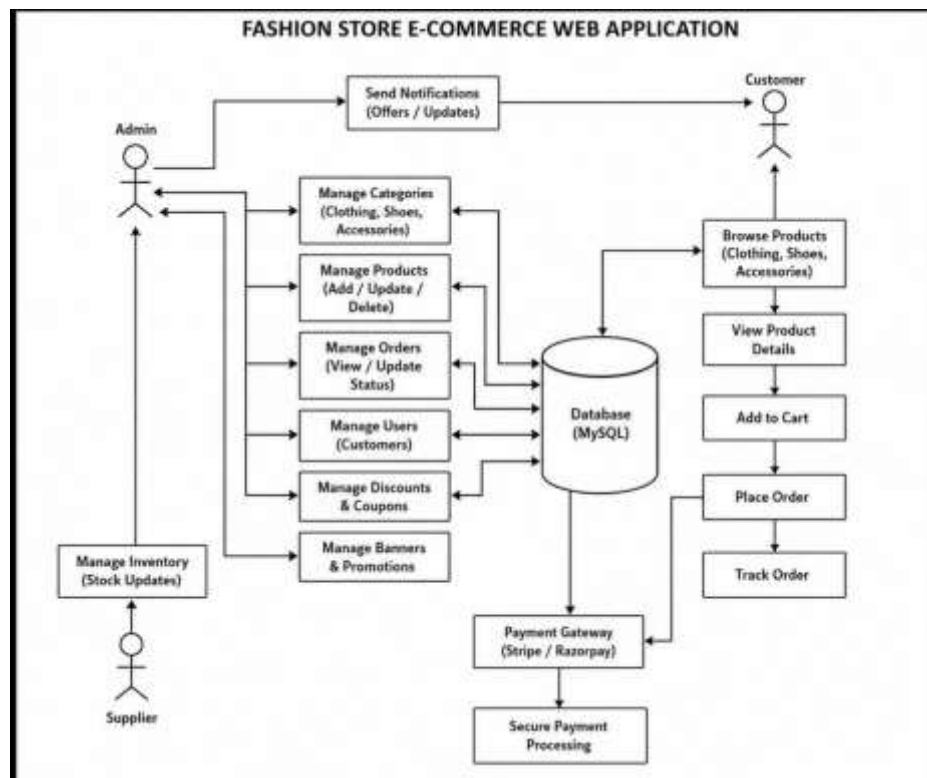


Figure 3.1 E - Commerce Web Application Architecture

As illustrated in Figure 3.1, the FashionStore system follows a layered web architecture. The client browser sends HTTP requests to **Apache Tomcat**, which routes them to the appropriate

Servlet controller based on the URL mapping defined by the @WebServlet annotation. The Servlet processes the request — validating input, calling the appropriate **DAO method**, or manipulating the **HttpSession** — and either forwards to a JSP for rendering or performs a redirect.

The DB Connection class provides JDBC connections to the **MySQL** fashionstore_db database. ProductDAO executes parameterised SQL queries using PreparedStatement to prevent SQL injection, maps ResultSet rows to Product objects, and returns them to the calling Servlet.

Application Server Layer: The client browser sends HTTP requests to **Apache Tomcat 11**, which acts as the Jakarta EE-compatible web application server. Tomcat routes each incoming request to the appropriate **Servlet controller** based on the URL pattern defined by the @WebServlet annotation on each Servlet class. For example, a POST request to /LoginServlet is routed to LoginServlet.java, and a GET request to /AddToCartServlet is

routed to AddToCartServlet.java.

The Servlet processes the request — validating input, calling the appropriate DAO method, or manipulating the HttpSession — and either forwards to a JSP page for view rendering or performs a redirect to another URL.

Controller Layer: Eight dedicated Servlet classes in the com.fashionstore.servlet package form the Controller layer. Each Servlet handles exactly one logical operation, keeping the controllers focused and maintainable.

Database Layer: MySQL 8.x stores all persistent data in two tables within the fashionstore_db database. The users table stores registered user credentials (id, name, email, password).

The overall data flow of the system is : the **Browser** initiates an HTTP request → **Tomcat** receives and routes it → the appropriate **Servlet** processes the business logic → **ProductDAO** interacts with **MySQL** for data retrieval or storage → the Servlet updates or reads the **HttpSession** for cart management → the result is forwarded to a **JSP page** which renders the HTML → the HTTP response is returned to the **Browser**.

3.4.2 Design Objectives

The Fashion Store Web Application was de-signed with the following objectives:

Implement the MVC pattern to separate presentation, business logic, and data access layers cleanly

- Use PreparedStatement for all SQL queries to prevent SQL injection vulnerabilities.
- Maintain cart state using server-side HttpSession — no client-side cookie manipulation required.
- Design a reusable DBConnection utility that centralises JDBC configuration.
- Implement a ProductDAO that abstracts all database operations behind clean Java calls.
- Provide full CRUD-style cart management: add, update quantity, and remove items.
- Ensure proper session lifecycle management — creation on login, invalidation on logout/checkout.
- Follow Java EE naming conventions and package structure (model, dao, db, servlet)

3.4.3 Major System Modules

3.4.3.1 Database Connection Module (DBConnection.java)

Provides a static getConnection() factory method that loads the com.mysql.cj.jdbc.Driver class and returns a JDBC Connection to the fashionstore_db MySQL database via DriverManager. Connection parameters (URL, user, password) are centralised here, making it the single point of change for any database reconfiguration. A main() test method is included to verify connectivity independently of the web server.

3.4.3.2 Domain Model Module (Product.java & CartItem.java)

Product is a plain Java object (POJO) with five fields — id, name, description, price, and imageUrl — populated from database rows. CartItem composes a Product with an integer quantity, representing a single line in the shopping cart. Both classes expose only getters and setters, keeping the model layer free of business logic.

3.4.3.3 Data Access Layer (ProductDAO.java)

Implements two methods: getAllProducts() executes SELECT * FROM products and returns a List<Product>; getProductById(int id) executes a parameterised SELECT ... WHERE id = ? and returns a single Product.

3.2.3.4 *Authentication Module (LoginServlet & RegisterServlet)*

RegisterServlet reads name, email, and password from a POST form and executes INSERT INTO users via PreparedStatement. On success it redirects to the login page. LoginServlet queries the users table with the submitted credentials; on a match it creates an HttpSession and stores userName and userId as session attributes, then redirects to index.jsp.

3.2.3.5 *Cart Management Module*

The cart is stored as a List<CartItem> in the HttpSession. AddToCartServlet fetches the product by ID, checks the session list for a matching CartItem, and either increments its quantity or adds a new entry.

3.2.3.6 *Order & Checkout Module*

CheckoutServlet removes the cart attribute from the session and forwards to

order-confirmation.jsp, simulating a successful purchase. CompleteOrderServlet provides an additional completion path — it similarly clears the cart and redirects the user back to index.jsp.

3.2.3.5 *Session & Logout Module*

LogoutServlet retrieves the current session with getSession(false) to avoid creating a new session if none exists, then calls session.invalidate() to completely destroy it — removing both the user identity and the cart in one operation. The user is then redirected to the homepage

3.2.3.6 *Session & Logout Module*

This module handles storage and retrieval of data using a SQL database. It manages user information, training content, internship details, and assessment data efficiently.

It ensures data consistency, integrity, and fast access to required information.

3.2.4 Working Flow of System

1. User opens the FashionStore homepage (index.jsp). The page calls ProductDAO.getAllProducts() to fetch products from MySQL and renders them as Bootstraps.
2. New user clicks **Register** and submits the registration form. RegisterServlet inserts the record into the users table and redirects to the login page.
3. User logs in via the login form. LoginServlet validates credentials against the users table. On success, an HttpSession is created with userName and userId.
4. User browses the product catalogue.
5. User clicks **Add to Cart**. AddToCartServlet fetches the product, checks the session cart (List<CartItem>), and adds or increments the item.
6. On the cart page, the user adjusts item quantities. UpdateQuantityServlet receives the new value via POST and updates the CartItem.quantity in the session..
7. User removes an item. RemoveFromCartServlet filters it out of the session list using removeIf().

8. User proceeds to checkout. CheckoutServlet clears the cart from the session and shows the order-confirmation page.

3.3 Implementation

The FashionStore application was implemented using a **Maven-based Eclipse project** structured under the package root com.fashionstore. Development followed a bottom-up approach: database schema first, then the model and DAO layer, then Servlet controllers, and finally JSP views. Each module was tested incrementally before integration. The full source is organised into four sub-packages: db, model, dao, and servlet.

3.3.3 Database Schema

Two tables were created in the fashionstore_db MySQL database:

users id INT AUTO_INCREMENT PK Primary key	users id INT AUTO_INCREMENT PK Primary key	users id INT AUTO_INCREMENT PK Primary key	users id INT AUTO_INCREMENT PK Primary key
users name VARCHAR(100) Full name	users name VARCHAR(100) Full name	users name VARCHAR(100) Full name	users name VARCHAR(100) Full name
users email VARCHAR(100) UNIQUE Login identifier	users email VARCHAR(100) UNIQUE Login identifier	users email VARCHAR(100) UNIQUE Login identifier	users email VARCHAR(100) UNIQUE Login identifier
users password VARCHAR(255) Stored password	users password VARCHAR(255) Stored password	users password VARCHAR(255) Stored password	users password VARCHAR(255) Stored password
products id INT AUTO_INCREMENT PK Primary key	products id INT AUTO_INCREMENT PK Primary key	products id INT AUTO_INCREMENT PK Primary key	products id INT AUTO_INCREMENT PK Primary key
products name VARCHAR(150) Product title	products name VARCHAR(150) Product title	products name VARCHAR(150) Product title	products name VARCHAR(150) Product title
products description TEXT Product details	products description TEXT Product details	products description TEXT Product details	products description TEXT Product details
products price DECIMAL(10,2) Product price	products price DECIMAL(10,2) Product price	products price DECIMAL(10,2) Product price	products price DECIMAL(10,2) Product price
products image_url VARCHAR(255) Path/URL to image	products image_url VARCHAR(255) Path/URL to image	products image_url VARCHAR(255) Path/URL to image	products image_url VARCHAR(255) Path/URL to image

3.3.4 Authentication Implementation

ProductDAO contains two public methods. `getAllProducts()` opens a JDBC connection via `DBConnection.getConnection()`, prepares and executes `SELECT * FROM products`, then iterates the `ResultSet` using a while-loop — constructing a new `Product` POJO for each row by reading `id`, `name`, `description`, `price`, and `image_url` columns — and accumulates them into an `ArrayList` returned to the caller. `getProductById(int id)` executes a parameterised `SELECT * FROM products WHERE id = ?` query, sets the integer parameter, and returns the first matching `Product` object or null if not found.

3.3.5 Product Listing and DAO Implementation

The shopping cart is implemented as a server-side `List<CartItem>` stored in the `HttpSession`. **AddToCartServlet** reads the `id` request parameter, retrieves the `Product` from `ProductDAO`, then iterates the session cart. If a `CartItem` with a matching product ID is found, its quantity is incremented by 1; otherwise a new `CartItem(product, 1)` is appended. **UpdateQuantityServlet** receives a POST request with `productId` and quantity parameters, iterates the cart to locate the item, and calls `item.setQuantity(newQuantity)`. Both Servlets redirect to `cart.jsp` on completion.

3.3.6 Cart Management Implementation

CheckoutServlet simulates a successful purchase by calling `session.removeAttribute("cart")` to clear the cart from memory and then redirecting the browser to `order-confirmation.jsp`. **CompleteOrderServlet** performs the same cart removal with a null-check guard before calling `removeAttribute`, then redirects to `index.jsp`, returning the user to the homepage after order completion. In a production system this module would additionally execute an SQL INSERT to persist order line items to an orders table before clearing the cart.

3.3.7 Checkout and Order Completion Implementation

The data management module was implemented using a SQL database to efficiently store and manage application data.

Features include:

- Storage of user details such as registration information and login credentials
- Management of training content, technical questions, and company-specific questions
- Storage of internship details and related information
- Retrieval of data for dynamic display across different modules

3.3.8 System Testing

The application was tested across the following scenarios:

- Valid login, invalid credentials, and server error paths
- Homepage product listing — verified all DB products rendered correctly in JSP
- Add-to-cart for new items and existing items (quantity increment)
- Cart removal — item disappears from `cart.jsp` after `RemoveFromCart`
- Checkout flow — cart is empty after checkout, confirmation page displayed

3.3.9 Project Work flow

Browser → HTTP Request → **Tomcat ® Servlet (Controller)** → **ProductDAO** / **HttpSession (Model)** → **MySQL DB** → Result returned to Servlet → **JSP View** renders HTML → HTTP Response → **Browser**

3.4 Platform

The E-Commerce Web Application was developed using modern web development technologies, programming languages, and tools. These platforms were used for designing the user interface, implementing system functionality, managing the database, and testing the application.

3.4.3 Programming Languages

The following programming languages were used during the development of the On-line Placement Training and Assessment System application:

- **Java 21** — Core application language. Used for all Servlet logic, DAO classes, model POJOs, and database connectivity. Compiled with Maven using maven-war-plugin.
- **CSS** – Used for styling web pages, improving visual appearance, and creating responsive layouts for better user experience.
- **JavaScript** – Used to add interactivity to web pages such as form validation, navigation handling, chatbot interaction, and dynamic content updates.
- **SQL** – Used for database operations such as storing user data, managing training content, retrieving information, and maintaining data integrity.

3.4.4 Libraries and Tools

The following libraries and tools were used for the implementation of the Online Placement Training and Assessment System application:

Bootstrap – Used for designing responsive and mobile-friendly user interfaces with components such as navigation bars, forms, and layout grids.

MySQL – Used as the relational database for storing user details, training content, internship data, and assessment-related information.

VS Code – Used as the development environment for writing, editing, and managing the project code.

Web Browser – Used for testing and running the application (Google Chrome, Microsoft Edge, etc.).

Git – Used for version control and tracking changes in the project (optional).

CHAPTER 4

RESULTS

The FashionStore E-Commerce Web Application was successfully developed, deployed on Apache Tomcat 11, and tested as a complete Java EE full-stack system. All major modules — user authentication, product catalogue, cart management, checkout, and session lifecycle — were implemented and verified through functional testing across multiple browsers and screen sizes.

4.1 Home Page Result

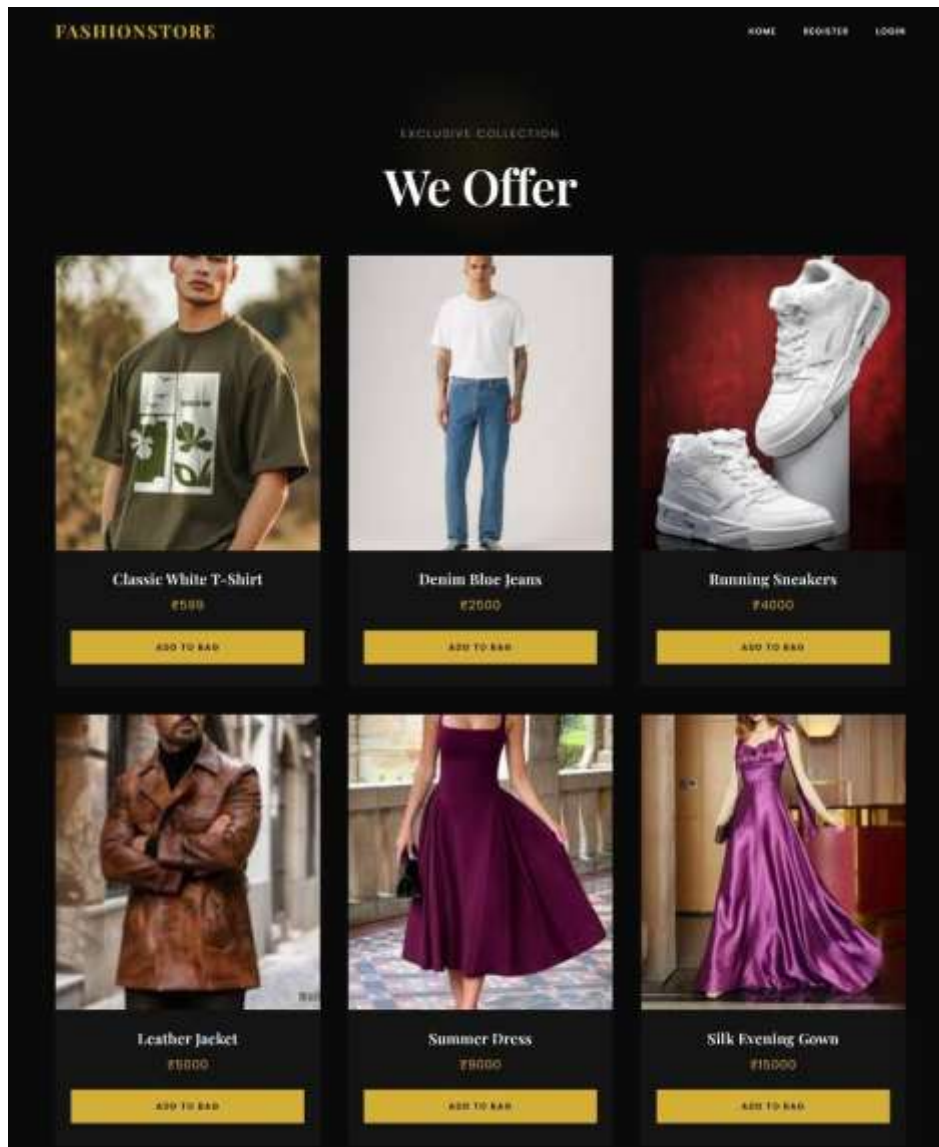


Figure 4.1 Fashion Store - E Commerce Web Application Home Page

The homepage (index.jsp) dynamically fetches all products from the MySQL products table via ProductDAO.getAllProducts() and renders them as Bootstrap card components. The navigation bar displays context-sensitive links — showing Login/Register for guests and the username + Logout option for authenticated users, driven by a check on the userName session attribute.

4.2 Login Page

The login page presents a form that submits email and password to LoginServlet via HTTP POST. On valid credentials, a session is created and the user is redirected to the homepage. On failure, the page reloads with an error message rendered from the error query parameter.

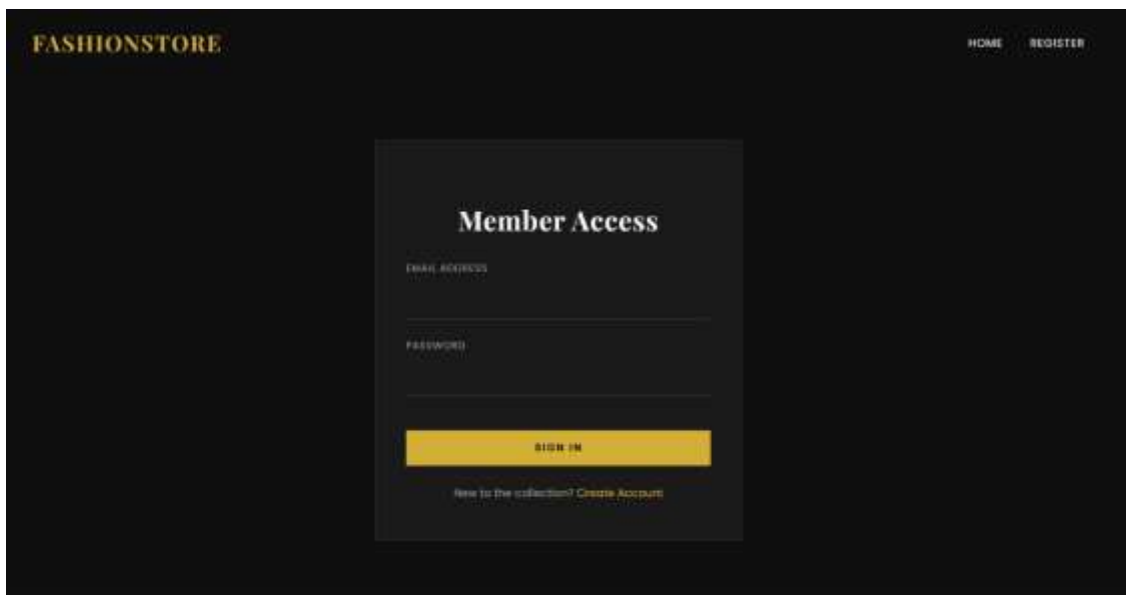


Figure 4.2 Login Page

4.3 Registration Page

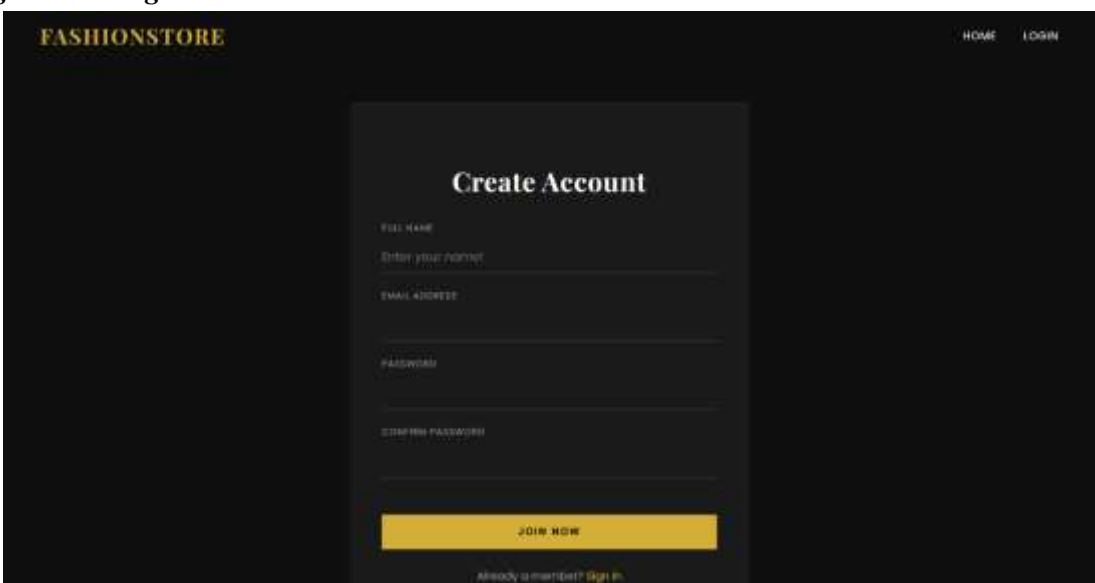


Figure 4.3 Tutorial Page

The tutorials page shown in **Fig. 4.3** provides structured learning content for place-ment preparation. It includes various topics such as aptitude, technical concepts, and interview prepa-ration strategies. The content is organized in a user-friendly manner to help students easily understand and learn.

4.4 Product Detail / Cart Page

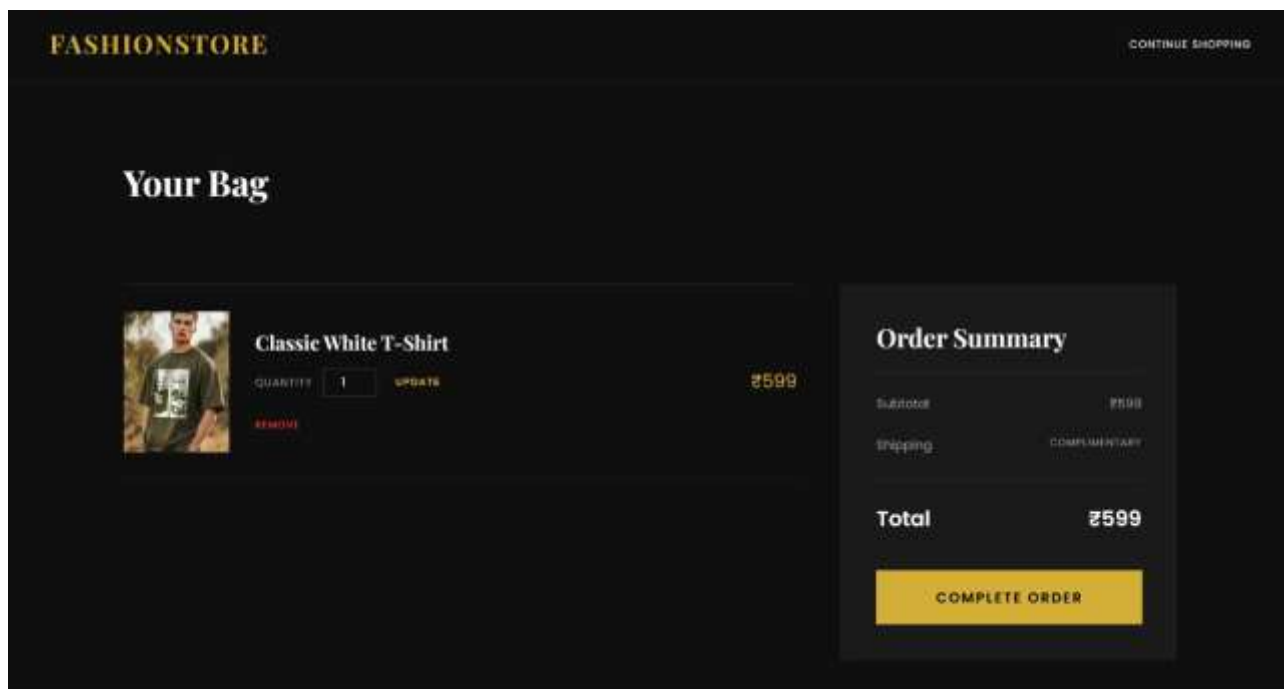


Figure 4.4 Product Detail / Cart Page

Clicking 'Add to Cart' triggers a GET request to AddToCartServlet with the product ID as a parameter. The servlet adds the item (or increments its quantity) in the session List<CartItem>.

The cart page (cart.jsp) reads the session list and renders each item with its quantity, unit price, and line total, along with Remove and Update Quantity controls.

4.5 Order Confirmation Page

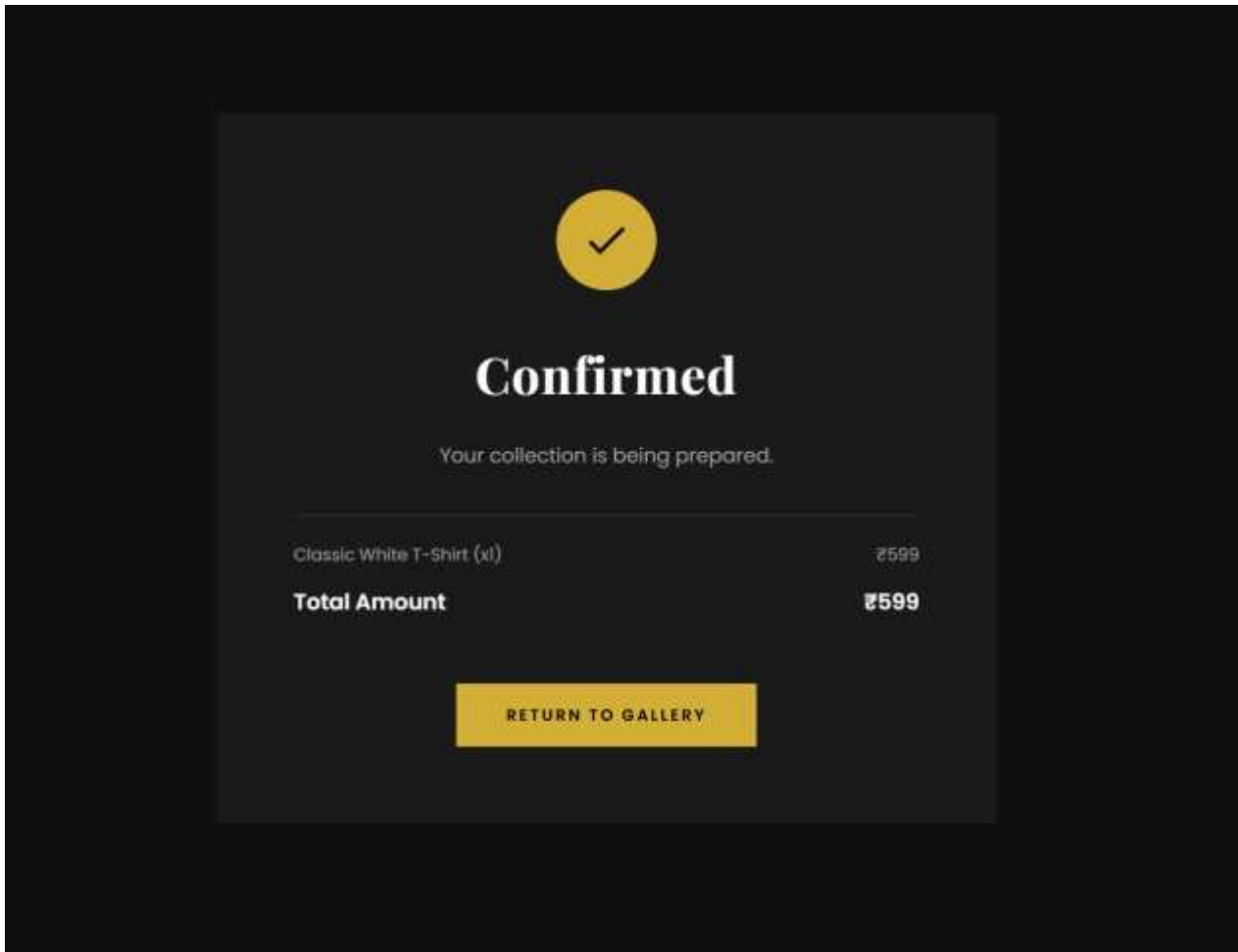


Figure 4.5 Order Confirmation Page

Clicking Checkout triggers CheckoutServlet, which removes the cart from the session and redirects to order-confirmation.jsp. The page thanks the user and provides a link back to the homepage. The cart is empty on return, confirming the session attribute was successfully cleared..

4.6 Advantages of the System

The developed Online Placement Training and Assessment System web application provides the following advantages:

- Clean MVC architecture ensures code is modular, readable, and easy to extend.
- DAO pattern isolates all SQL from business logic — switching databases only in ProductDAO and DBConnection.
- Server-side session management is more secure than client-side cart storage (cookies)
- The WAR packaging allows direct deployment to any Jakarta EE-compatible applicationserver.

4.7 Limitations

Some limitations observed during development:

- Passwords are stored in plain text — a production system must use bcrypt or PBKDF2 hashing.
- No orders table in the database — order details are not persisted after checkout; only the in-memory cart is cleared.

- No admin panel for product management — adding, updating, or deleting products
- No pagination on the product listing page — all products are loaded in a single query.

4.8 Future Enhancements

The system can be improved further with:

- Add an orders and order_items table to persist purchase history and enable order tracking
- Develop an admin panel (CRUD interface) for managing products, categories, and orders.
- Integrate a payment gateway (e.g., Razorpay or Stripe) for real transaction processing.
- Add product search and category-based filtering to the catalogue page.
- Implement pagination or infinite scroll for large product catalogues.
- Add product rating and review functionality for user engagement.

CHAPTER 5

CONCLUSION

The FashionStore E-Commerce Web Application project successfully demonstrated the full software development lifecycle of a real-world Java EE application. Starting from database schema design through JDBC connectivity, domain modelling, DAO implementation, Servlet-based controller logic, and JSP view rendering, the project provided end-to-end practical exposure to every tier of a modern web application.

The project reinforced critical software engineering concepts including the MVC design pattern, separation of concerns, parameterised SQL queries for security, server-side session management, and Maven-based build and deployment workflows. All eight Servlet modules and the DAO layer were individually developed, integrated, and tested — resulting in a fully functional e-commerce platform deployable on Apache Tomcat 11.

The internship at TAP Academy provided a structured learning environment that bridged academic knowledge and industry practice. The hands-on experience of building a complete Java web application — from blank Eclipse project to a WAR file running on a local Tomcat server — has established a solid technical foundation for professional software development. Future enhancements such as password hashing, order persistence, and admin panel development will further evolve this project into a production-ready system.

CHAPTER 6

REFERENCES

1. <https://tai.thetapacademy.com/> Used for coding practice, online recorded video sessions, assignments, technical learning, and skill development during the internship training program.
2. W3Schools Official Website Used for learning and understanding HTML, CSS, and JavaScript concepts for webpage design and development.
3. MDN Web Docs Used as a reference for understanding advanced web development concepts, syntax, and implementation.
4. Oracle Java Documentation Used for understanding Core Java, Advanced Java, Servlets, JDBC, and object-oriented programming concepts.

5. Apache Tomcat Official Website Used for configuring and deploying the Java web application on the Apache Tomcat Server.
6. Maven Official Website Used as a reference for Maven project configuration, dependency management, and build automation.
7. Eclipse IDE Official Website Used as the primary Integrated Development Environment (IDE) for coding, debugging, and project development.
8. Font Awesome Icons Library Used for implementing user interface icons, cart icons, navigation icons, and social media icons in the NovaMart E-Commerce project.
9. https://youtu.be/Ke_XgMR8r7k?si=8tt7A9E4tGGWY73