

A Project Report On
E-commerce Website (Online Dress Shopping)
For the course
Course : Software Development II

Project Partner(s):

MD.Lutfor Rahman(IT-21047)
Ashraful Alam Ridoy(IT-21048)
Rocktim Chakma(IT-21052)
Umma Kulsum Ashamoni (IT-21050)
Mukta Rani Barman(IT-21051)
Md.Rifat Ullah Khan(IT-21061)
Anas Sabbir (IT-21060)
Moshiur Rahman (IT-21049)

Supervised By,

Dr. Ziaur Rahman
Associate Professor
ICT ,MBSTU



MAWLANA BHASHANI SCIENCE AND TECHNOLOGY UNIVERSITY

E-Commerce Dress Shop Website Documentation

December 2, 2024

Contents

1	Introduction	2
2	Objectives	2
3	System Design	2
3.1	Architecture	2
3.2	Database Design	2
4	Implementation	2
4.1	Frontend	2
4.1.1	HTML	5
4.1.2	CSS	6
4.1.3	JavaScript (JS)	6
4.1.4	Bootstrap	7
4.1.5	Express.js	7
4.1.6	MongoDB	8
4.2	Backend	9
5	Testing	9
6	Conclusion	9

1 Introduction

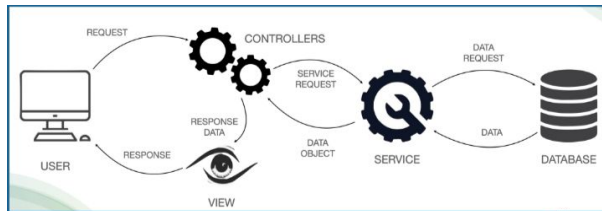
The e-commerce website is developed to offer an easy-to-use, responsive, and secure platform for online shopping. This document covers the project's objectives, methodology, and technologies used.

2 Objectives

The main objectives of the project are as follows:

- Develop a user-friendly and responsive website for online shopping.
- Implement a secure login and registration system.
- Create a product catalog with search and filter options.
- Integrate a shopping cart and checkout system.

3 System Design



3.1 Architecture

The website follows a Model-View-Controller (MVC) architecture. The backend is powered by a server-side language (e.g., PHP or Node.js), while the frontend uses HTML, CSS, and JavaScript.

3.2 Database Design

The database is designed to store user data, product information, orders, and transactions. Tables include `Users`, `Products`, `Orders`, and `OrderDetails`.

4 Implementation

4.1 Frontend

Front-end web development for an online dress shopping platform involves creating the part of the website that users interact with directly. It includes designing the user interface (UI), implementing responsive layouts, and ensuring smooth functionality. Here are key aspects and technologies involved:

Core Technologies

- **HTML (HyperText Markup Language):** Structures the content of the web pages.
- **CSS (Cascading Style Sheets):** Styles the appearance, including layout, colors, fonts, and animations.
- **JavaScript:** Adds interactivity such as search filters, product carousels, and dynamic updates.

Frameworks and Libraries

CSS Frameworks

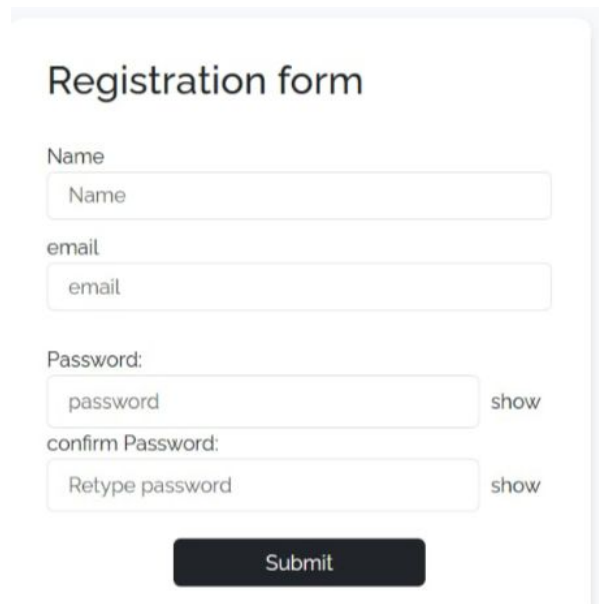
- **Bootstrap:** Simplifies responsive design and offers pre-built components like grids, buttons, and modals.
- **Tailwind CSS:** A utility-first framework for custom designs.

JavaScript Frameworks/Libraries

- React.js: Component-based library for building dynamic and reusable UI elements.
- Vue.js: A progressive framework that is easy to integrate with small and large projects.
- Angular: A comprehensive framework for building feature-rich web apps.

Registration Page

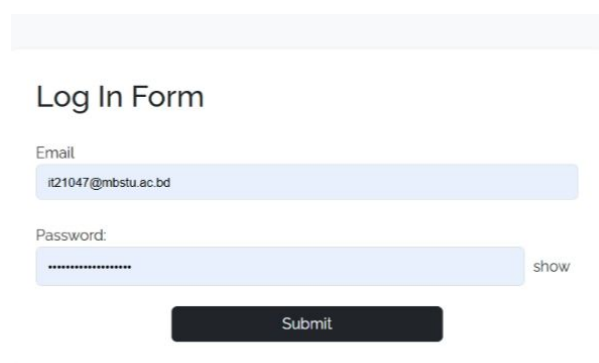
The Registration Page allows new users to create an account by entering their username, email, and password.



The registration form is titled "Registration form". It contains four input fields: "Name", "email", "Password:", and "confirm Password:". The "Password:" field has a "show" toggle, and the "confirm Password:" field has a "show" toggle. Below the fields is a "Submit" button.

Login Page - login.html

The Login Page allows users to sign in with their email and password. It includes a form and a link to the registration page for new users.



The login form is titled "Log In Form". It contains two input fields: "Email" and "Password:". The "Email" field has a placeholder value "it21047@mbstu.ac.bd". The "Password:" field has a "show" toggle. Below the fields is a "Submit" button.

Home Page - home.html

The Home Page provides an overview of the dress shop, featuring links to other sections such as Shop, Blog, Cart, and Login.

Contact Page - contact.html

The Contact Page provides a form for users to send inquiries or feedback to the shop.

Checkout Page - checkout.html

The Checkout Page gathers shipping and payment information from the user.

Payment Page - payment.html

The Payment Page confirms the payment process.

4.1.1 HTML

HTML (HyperText Markup Language) is the standard language used to create and design the structure of web pages. It defines the structure and content of a webpage by using a system of tags and attributes. The main purpose of HTML is to provide a way to structure and present content on the web.

Core Concepts:

Tags:

Enclosed in `<tag>`, tags define elements. Examples:

- `<h1>`: Heading 1
- `<p>`: Paragraph
- `<a>`: Anchor tag for links
- ``: Image
- Attributes: Provide additional information about elements.
- Example:

```
<a href="https://example.com" target="_blank">Click Here</a>
```

- Forms: Enable user interaction with fields and buttons. Example:

```
<form action="/submit" method="POST">
  <input type="text" name="username">
  <button type="submit">Submit</button>
</form>
```

- Document Structure:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Page Title</title>
  </head>
  <body>
    <h1>Welcome!</h1>
    <p>This is an example page.</p>
  </body>
</html>
```

4.1.2 CSS

CSS (Cascading Style Sheets)

CSS is used to style HTML elements and make web pages visually appealing.

Core Concepts:

- Selectors: Specify which elements to style.
- Universal: *
- Class: .className
- ID: idName
- Element: tag
- Properties: Control styling.

Example:

```
body {  
  background-color: #f0f0f0;  
  font-family: Arial, sans-serif;  
}
```

- Box Model: Controls spacing and layout.
- Components: margin, border, padding, content.
- Responsive Design:
Use media queries to adapt to screen sizes.

```
@media (max-width: 768px) {  
  body {  
    font-size: 14px;  
  }  
}
```

- CSS Animation:

```
@keyframes fadeIn {  
  from { opacity: 0; }  
  to { opacity: 1; }  
}  
  
.fade {  
  animation: fadeIn 2s ease-in-out;  
}
```

4.1.3 JavaScript (JS)

JavaScript enables dynamic behavior, interactivity, and logic in web applications.

Core Concepts:

Variables:

javascript

```
let name = "John"; // Changeable variable  
const age = 25; // Immutable variable
```

- Functions :

```
function greet() {
  console.log("Hello, World!");
}
greet();
```

- DOM Manipulation: Interact with and update HTML dynamically.

```
document.getElementById("title").innerText = "New Title";
```

- Event Handling:

```
document.querySelector("button").addEventListener("click", () => {
  alert("Button clicked!");
});
```

- Promises and Async/Await: Handle asynchronous operations.

```
fetch("https://api.example.com/data")
  .then(response => response.json())
  .then(data => console.log(data));
```

4.1.4 Bootstrap

A front-end framework for building responsive and mobile-friendly websites quickly.

Core Concepts:

- Grid System:
Layouts are based on a 12-column grid. Example:

```
<div class="container">
  <div class="row">
    <div class="col-6">Column 1</div>
    <div class="col-6">Column 2</div>
  </div>
</div>
```

- Components:
Ready-made UI elements.
Example:

```
<button class="btn btn-primary">Primary Button</button>
```

- Responsive Utilities:
Show/hide elements based on screen size.

```
<div class="d-none d-md-block">Visible on Desktop</div>
```

- Customization: Use Sass for advanced styling.

4.1.5 Express.js

A Node.js framework for building server-side applications and APIs.

Core Concepts:

- Basic Server Setup:


```
const express = require('express');
const app = express();

app.get('/', (req, res) => {
  res.send('Hello World!');
});

app.listen(3000, () => {
  console.log('Server is running on port 3000');
});
```

- Middleware : Functions that modify requests and responses.

```
app.use((req, res, next) => {
  console.log('Request received');
  next();
});
```

- Routing: Define endpoints.

```
app.post('/submit', (req, res) => {
  res.send('Form Submitted');
});
```

4.1.6 MongoDB

A NoSQL database that stores data in flexible, JSON-like documents.

Core Concepts:

- CRUD Operations:

```
const mongoose = require('mongoose');

const UserSchema = new mongoose.Schema({
  name: String,
  email: String,
});

const User = mongoose.model('User', UserSchema);

// Create
User.create({ name: 'John', email: 'john@example.com' });

// Read
User.find().then(users => console.log(users));

// Update
User.updateOne({ name: 'John' }, { email: 'johnny@example.com' });

// Delete
User.deleteOne({ name: 'John' });
```

- Collections and Documents:

Collection: Like a table in SQL (e.g., 'users').

Document: Like a row, but in JSON format. Example:

```
{
  "name": "Alice",
  "age": 30,
  "isAdmin": true
}
```

- Indexing: Speeds up query performance.

```
User.createIndex({ email: 1 });
```

- Aggregation: Perform advanced data analysis.

```
User.aggregate([
  { $match: { isAdmin: true } },
  { $group: { _id: null, total: { $sum: 1 } } }
]);
```

4.2 Backend

The backend handles requests and serves data to the frontend. RESTful APIs are used to manage data exchange between the frontend and backend.

1. Key Features

- User Management: Registration, login, profile updates.
- Product Catalog: Display products with filters, search, and details.
- Cart Checkout: Add/remove items, update quantities, apply discounts, and payment processing.
- Order Management: View, track, and manage orders.
- Admin Panel: Manage products, users, and orders.
- Notifications: Email or SMS updates for orders and shipping.

Technologies

- Backend Frameworks: Node.js (Express), Django, Laravel, Ruby on Rails.
- Database: MySQL/PostgreSQL (relational) or MongoDB (NoSQL).
- APIs: RESTful or GraphQL.
- Authentication: JWT or OAuth 2.0.
- Cloud Hosting: AWS, Google Cloud, Firebase.

5 Testing

Testing includes unit tests for backend functions, integration tests for the API, and user testing to ensure a smooth experience.

6 Conclusion

The e-commerce website project successfully implemented core features of an online shopping platform. Future improvements may include implementing a recommendation system and optimizing the website's performance.