Online Food Ordering Application - Project Report

1. Introduction

The Online Food Ordering Application is a web-based platform that allows users to browse food categories, add items to a cart, and place orders. The project involves **frontend development** using **HTML, CSS, JavaScript**, along with **local storage and session management** for user authentication and cart functionality.

2. Features Implemented

User Authentication:

- **Registration Page:** Users can create an account with a username and encrypted password.
- **Login Page:** Users can log in securely using SHA-256 password hashing.
- **Session Management:** Users remain logged in during a session until they log out.

Menu & Category Filtering:

- **Dynamic Menu Display:** Menu items load dynamically from menu.json.
- **Category Filtering:** Users can select different food categories to view relevant items.

Shopping Cart Functionality:

- **Add to Cart:** Users can add food items to the cart with quantity tracking.
- **Cart Page:** Displays added items with options to increase/decrease quantity or remove items.
- **Total Price Calculation:** Automatically updates the total price based on selected items.
- **Local Storage Persistence:** Ensures cart data is saved even after page refresh.

Deployment & Hosting:

- The project can be deployed using **GitHub Pages, Netlify, or a local server**.

---

3. Technologies Used

Technology	Purpose
HTML	Structuring the web pages
CSS & Bootstrap	Styling and responsive layout

Technology	Purpose
JavaScript	Interactivity and dynamic functionality
LocalStorage	Storing user and cart data
SessionStorage	Managing user sessions
SHA-256 (Crypto API)	Password encryption for security

---

4. File Structure

```
/online-food-ordering-app
   ├── index.html (Home Page)
   ├── register.html (User Registration Page)
   ├── login.html (User Login Page)
   ├── cart.html (Shopping Cart Page)
   ├── menu.json (Menu Data)
   ├── script.js (Main JavaScript Logic)
   ├── cart.js (Cart Page Logic)
   ├── style.css (Styling & Layout)
```

---

5. Code Implementation Overview

Fetching Menu Data (script.js)

```
async function fetchMenuData() {
  try {
    const response = await fetch("menu.json");
    if (!response.ok) throw new Error("Failed to load menu data");
    menuData = await response.json();
    loadMenu("Burgers");
  } catch (error) {
    console.error("Error fetching menu data:", error);
    }
}
```

Loading Menu Based on Category (script.js)

```
function loadMenu(category) {
    const menuContainer = document.querySelector("#menu-items .row");
    menuContainer.innerHTML = "";
    const filteredItems = menuData.filter(item => item.category.toLowerCase() === category.toLowerCase());
    filteredItems.forEach(item => {
        menuContainer.innerHTML += `
            <div class="col-md-4 mb-3">
                <div class="card">
                    <div class="card-body">
                        <h5 class="card-title">${item.name}</h5>
                        <p class="card-text">$${item.price.toFixed(2)}</p>
                        <button class="btn btn-primary add-to-cart" data-id="${item.id}">Add to Cart</button>
                    </div>
                </div>
            </div>`;
    });
    attachCartEventListeners();
}
```

Handling Add to Cart Functionality (script.js)

```
function addToCart(itemId) {
    let cart = JSON.parse(localStorage.getItem("cart")) || [];
    const item = menuData.find(item => item.id === itemId);
    if (!item) return;
    let existingItem = cart.find(cartItem => cartItem.id === itemId);
    if (existingItem) {
        existingItem.quantity += 1;
    } else {
        cart.push({ ...item, quantity: 1 });
    }
    localStorage.setItem("cart", JSON.stringify(cart));
    updateCartCount();
}
```

User Login with SHA-256 Encryption (script.js)

```
async function loginUser(event) {
    event.preventDefault();
    const username = document.getElementById("loginUsername").value;
    const password = document.getElementById("loginPassword").value;
    const storedHashedPassword = localStorage.getItem(username);
    if (!storedHashedPassword) return alert("User not found.");
    const hashedPassword = await hashPassword(password);
    if (hashedPassword === storedHashedPassword) {
        sessionStorage.setItem("loggedInUser", username);
        alert("Login successful!");
        window.location.href = "index.html";
    } else {
        alert("Invalid password.");
    }
}
```

---

6. Future Enhancements

- **Checkout & Payment Gateway Integration** (Stripe, PayPal, Razorpay, etc.)
- **Backend Integration** (Node.js, Express, MongoDB for persistent data storage)

- **User Profile Management** (Allow users to view past orders)
- **Email Verification & Password Reset Feature**

---

7. Conclusion

The **Online Food Ordering Application** is a fully functional, **frontend-based** food ordering system with essential **authentication, cart management, and menu filtering** features. It serves as a foundation for future enhancements such as **backend integration, payment processing, and user order tracking**.
