

# ***Department of Animation***

# ***Gujarat University***

***E-Waiter Web Application***

Table of Contents

* Abstract
* Introduction
* Overview of E-Waiter Web Application
* Technologies Used
* Application Architecture
* Features and Functionality
* User Interface
* Data Management
* Conclusion

1. ***Abstract:*** *E-Waiter is a web application designed to optimize order-taking and billing processes in restaurants. Developed using HTML, CSS, PHP, and JavaScript, it streamlines the tasks of waiters, enabling efficient management of customer orders and generating bills. This report provides an in-depth analysis of E-Waiter, covering its features, technologies, architecture, user interface, data management, security measures, deployment, and maintenance.*
2. ***Introduction:*** *In today's fast-paced restaurant environment, efficiency is crucial for delivering exceptional customer service. E-Waiter is a web-based solution aimed at enhancing the operational efficiency of restaurants by digitizing the order-taking and billing processes. By leveraging modern web technologies, E-Waiter empowers restaurant staff to serve customers more effectively, leading to improved overall dining experiences.*
3. ***Overview of E-Waiter Web Application:*** *E-Waiter comprises several interconnected modules, including home.php, innermenu.php, billing.php, and datainsert.php, supported by a SQL database. These components work together seamlessly to facilitate order management and billing tasks. The application provides a user-friendly interface for waiters to take orders, manage menus, generate bills, and interact with the database.*
4. ***Technologies Used:*** *E-Waiter is developed using the following technologies:*

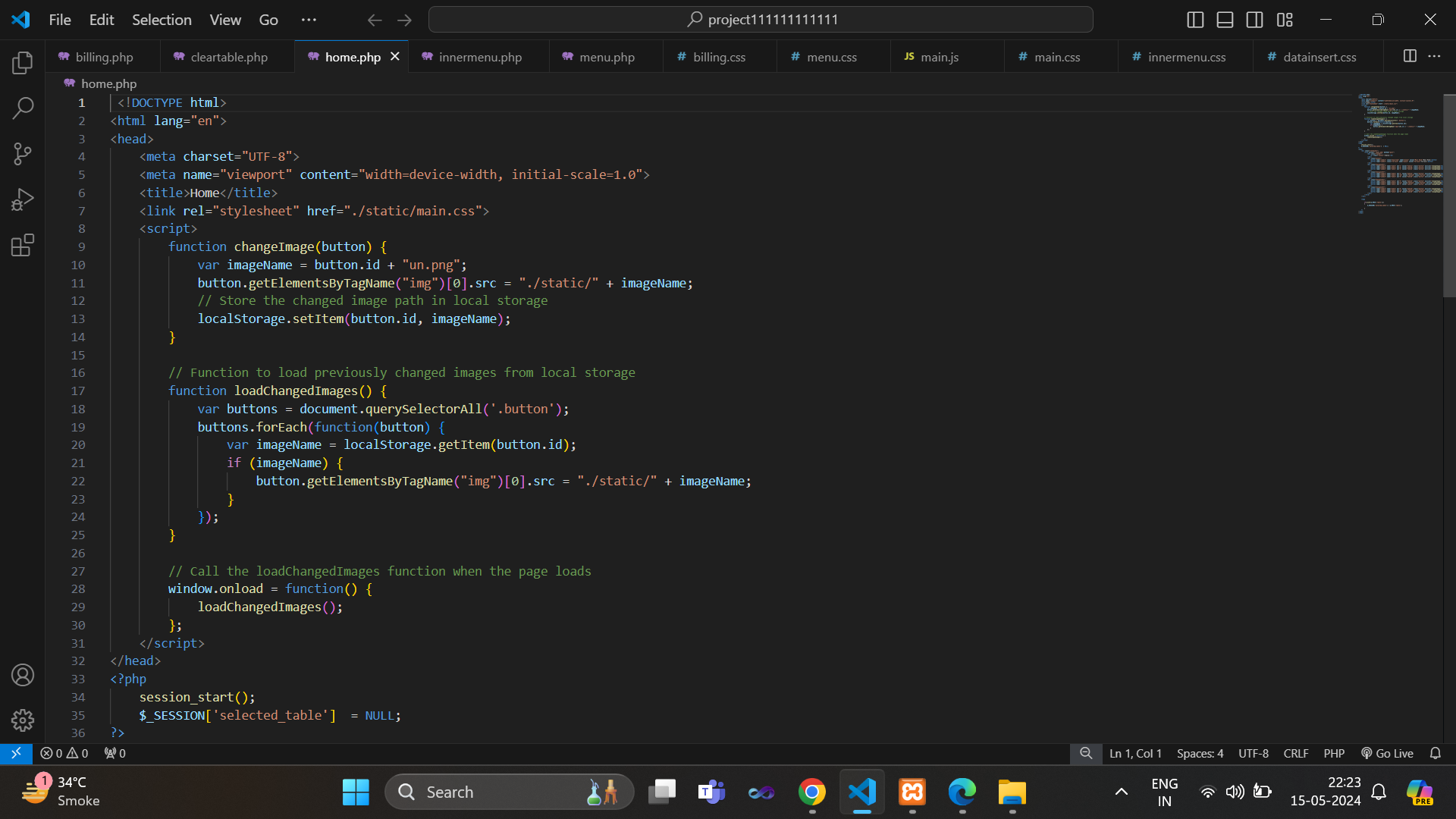
* *HTML: Provides the structure for web pages.*
* *CSS: Styles the appearance of web elements.*
* *PHP: Handles server-side scripting for dynamic content generation.*
* *JavaScript: Enhances interactivity and user experience.*
* *SQL: Manages the relational database for storing order and menu data.*

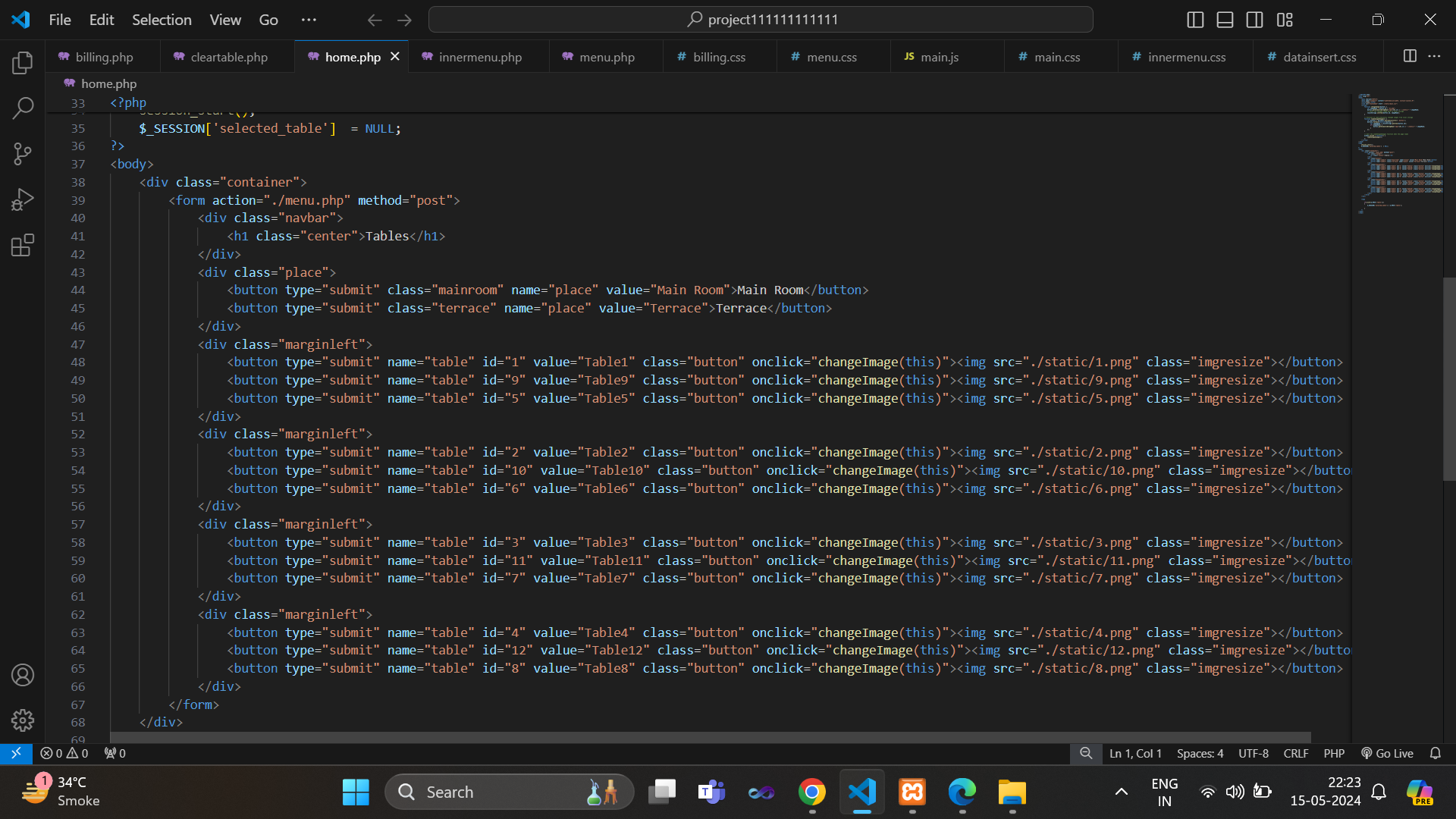
1. ***Application Architecture:*** *The architecture of E-Waiter follows a client-server model, where the client-side comprises web browsers accessing the application interface, while the server-side consists of PHP scripts executing on the web server. The application interacts with the SQL database to retrieve and manipulate data as required.*
2. ***Features and Functionality:*** *E-Waiter offers the following key features:*

* *Order Management: Allows waiters to input customer orders efficiently.*
* *Menu Management: Enables easy customization and updating of restaurant menus.*
* *Billing Generation: Automatically calculates and generates bills based on order details.*
* *Data Insertion: Facilitates the insertion of new order and menu data into the database.*
* *Real-time Updates: Provides instant updates on order status and billing information.*

1. ***User Interface:*** *The user interface of E-Waiter is designed to be intuitive and user-friendly. It features clear navigation menus, interactive elements for order selection, and visually appealing layouts for displaying menus and bills.*
2. ***Data Management:*** *E-Waiter utilizes a SQL database to manage restaurant data, including menus, orders, and billing information. The database schema is designed for efficient storage and retrieval of relevant data, ensuring optimal performance even during peak hours of operation. Data integrity and security are maintained through proper access controls and validation mechanisms.*
3. ***Conclusion:*** *E-Waiter represents a significant advancement in restaurant management technology, offering a comprehensive solution for optimizing order-taking and billing processes. By leveraging the power of web technologies, E-Waiter enhances operational efficiency, improves customer satisfaction, and empowers restaurant staff to deliver exceptional dining experiences. With its user-friendly interface, robust features, and strong security measures, E-Waiter is poised to revolutionize the way restaurants manage their operations in the digital age.*

***Home page :***

******



***Explanation of Home.php:***

1. *Head Section:*

* *Script Section: JavaScript functions are defined here. Two functions are defined: changeImage() and loadChangedImages(). These functions handle image changes and local storage manipulation respectively.*

1. *Body Section:*

* *Marginleft Divs: Contains sets of buttons representing tables. Each button has an associated image.*
* *PHP Section: Starts with <?php and ends with ?>. This section contains PHP code that interacts with the server. If the form is submitted (check isset($\_POST['table'])), it stores the selected table in a session variable $\_SESSION['selected\_table'].*

1. *JavaScript Functions:*

* *changeImage(button): This function changes the image displayed on a button when it's clicked. It also stores the changed image path in the browser's local storage.*
* *loadChangedImages(): This function loads previously changed images from the local storage when the page is loaded.*

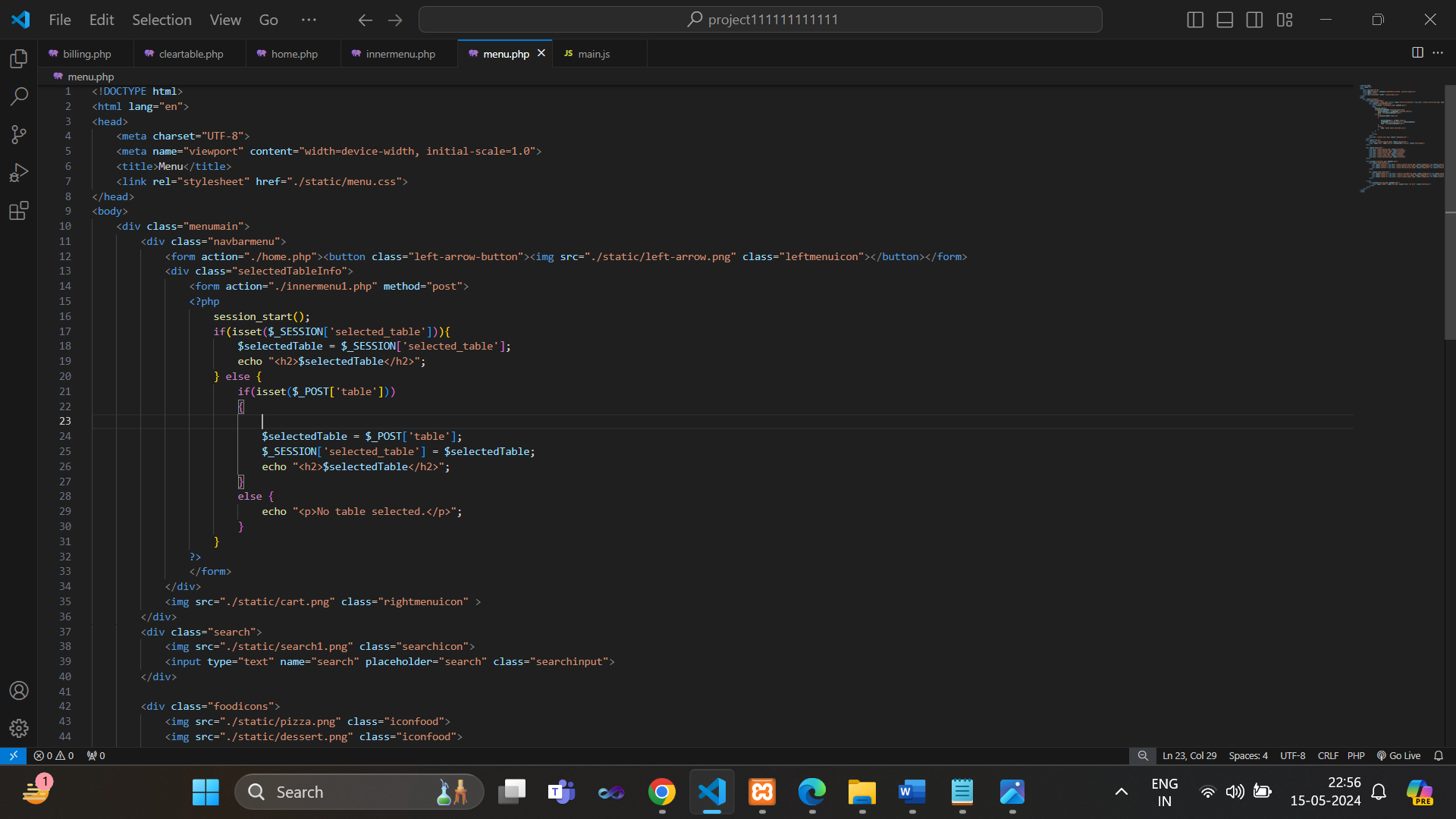
1. *PHP Section:*

* *Session Handling: Starts a session using session\_start(). When a table button is clicked, it stores the selected table value in the session variable $\_SESSION['selected\_table'].*

***Output:***



***Menu page:***



***Explanation of Menu.php:***

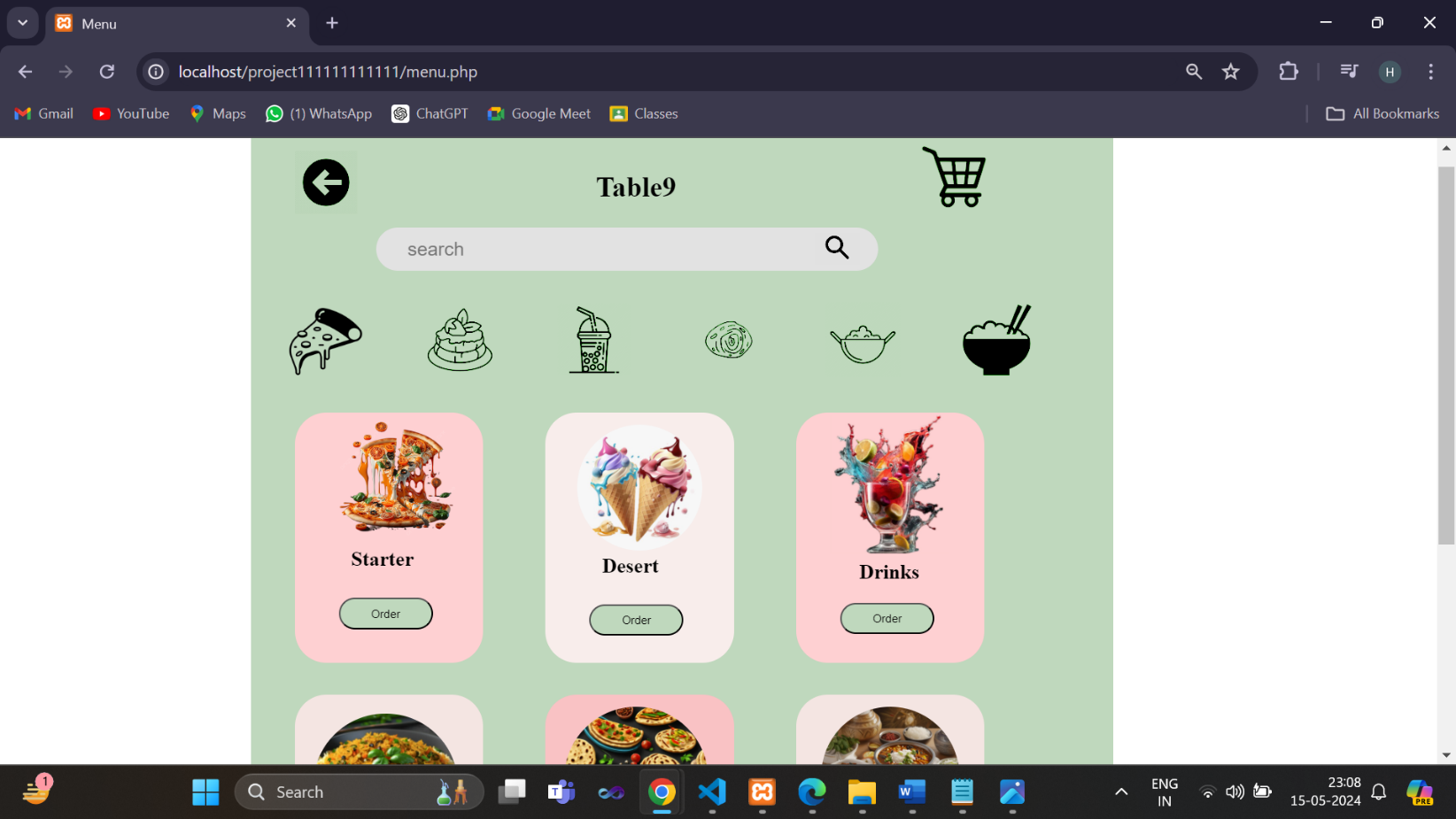
1. *Body Section:*

* *Selected Table Info: Displays the selected table information retrieved from the session variable $\_SESSION['selected\_table']. If no table is selected, it displays a message.*
* *Search Div: Contains a search input field.*
* *Food Icons: Icons representing different food categories.*
* *Food Item Divs: Divided into two sections. Each section contains food items with images and buttons to order them.*
* *Order for Bill Button: Submits the order for billing.*

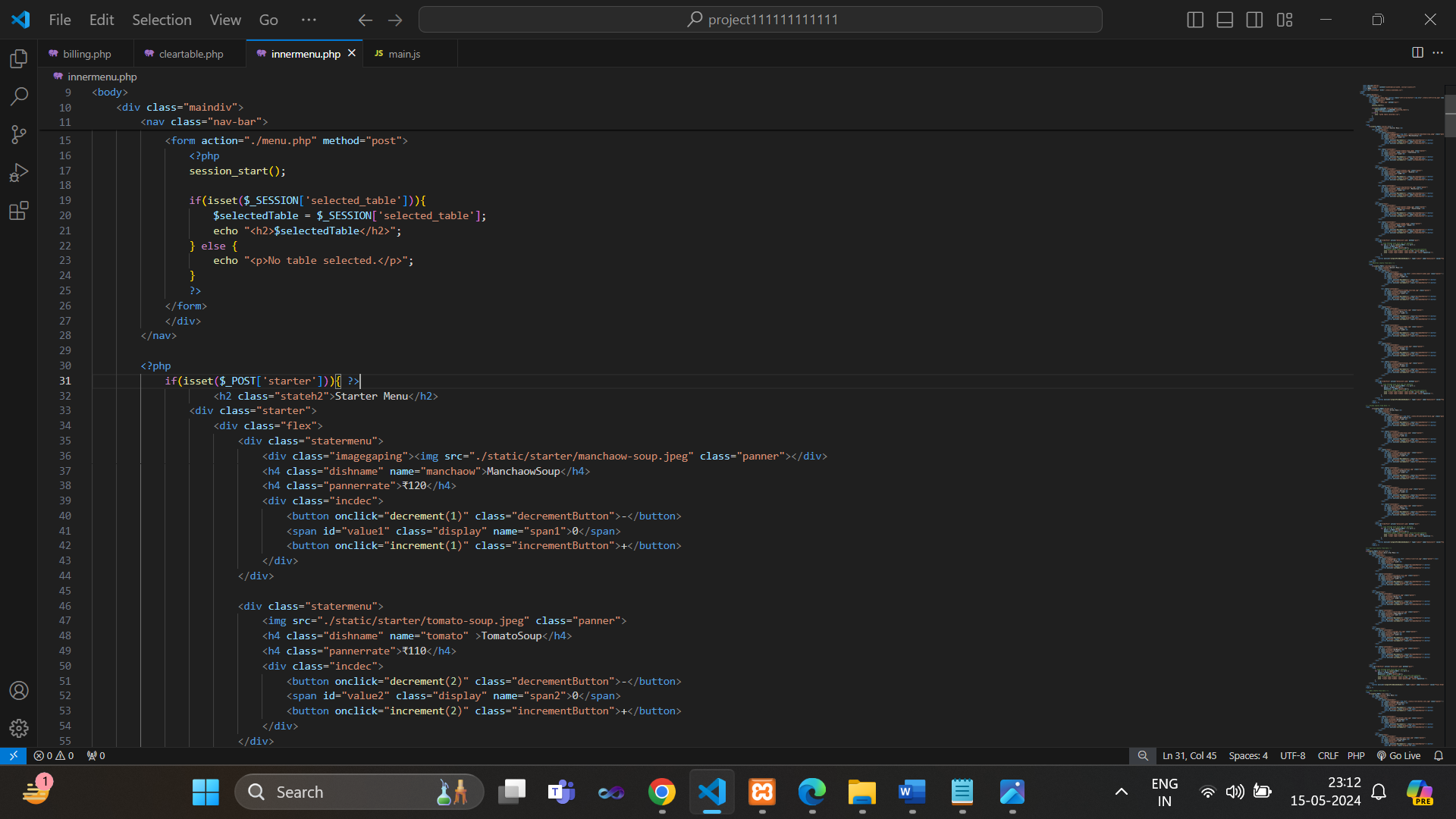
1. *PHP Section:*

* *Session Handling: Starts a session using session\_start(). Retrieves and displays the selected table information stored in the session variable $\_SESSION['selected\_table']. If no table is selected, it checks if a table is selected through $\_POST['table'] and sets the session variable accordingly.*

***Output:***

******

**I*nnermenu:***

******

***Explanation of Innermenu.php:***

1. *Body Section:*

* *Inside the body, there's a <div> element with the class "maindiv", which is likely used for styling or layout purposes.*

1. *PHP Section:*

* *It starts a session using session\_start().*
* *It checks if a table is selected ($\_SESSION['selected\_table']). If selected, it displays the table number; otherwise, it shows a message indicating that no table is selected.*
* *If a form with the name starter is submitted (checked by isset($\_POST['starter'])), it displays a list of starter menu items with images, names, prices, and buttons to increment/decrement quantities.*

1. *JavaScript Section:*

* *increment(index): Increments the quantity of the menu item corresponding to the given index.*
* *decrement(index): Decrements the quantity of the menu item corresponding to the given index.*
* *prepareFormDataAndSubmit(): Prepares form data based on the selected menu items and quantities and submits the form to datainsert.php.*

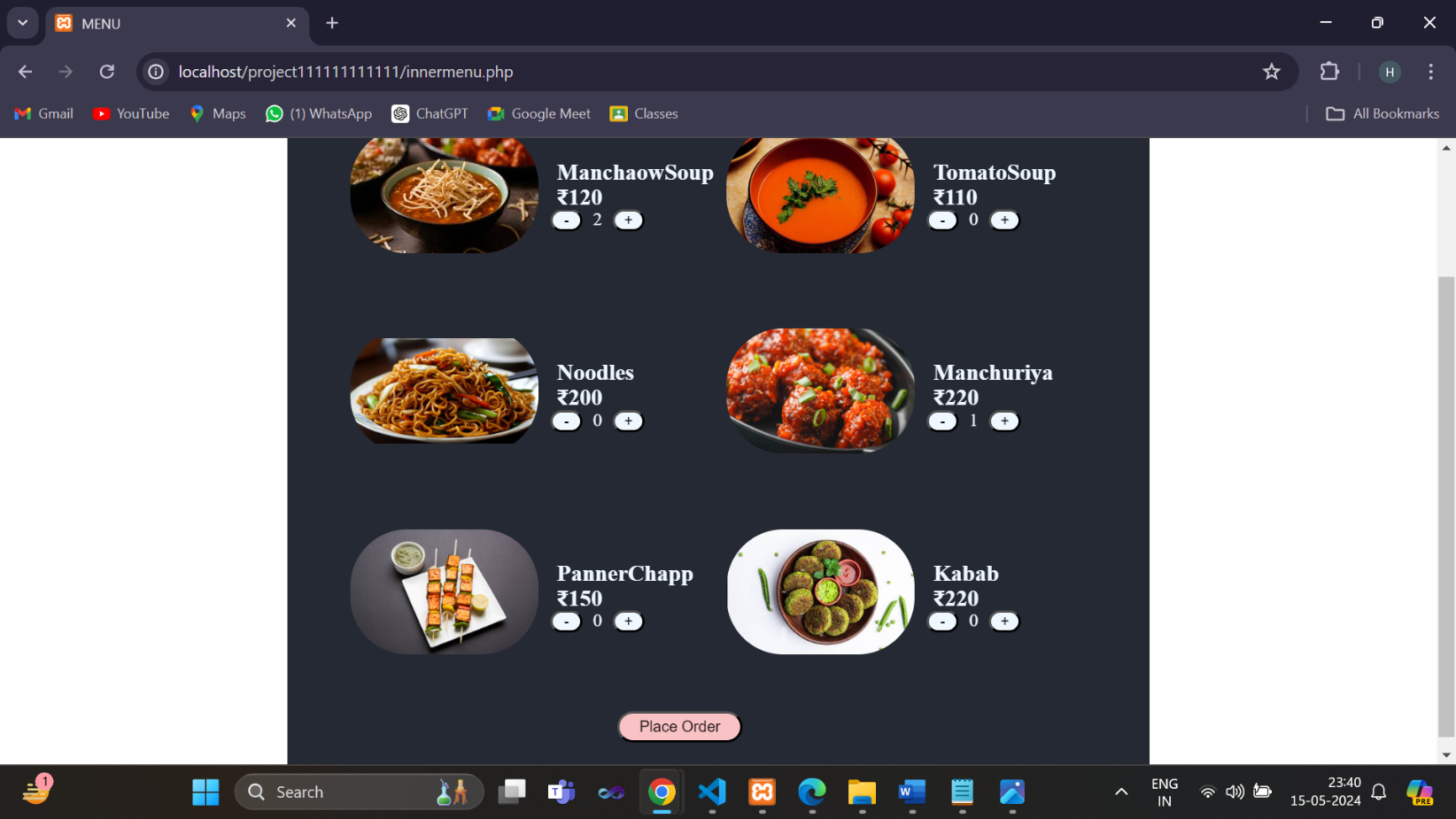
1. *Form Submission:*

* *Inside the prepareFormDataAndSubmit() function, it collects data for each menu item (name and quantity) and creates hidden input fields for each item in a form named orderForm.*
* *It then submits the form to datainsert.php when the "Place Order" button is clicked.*

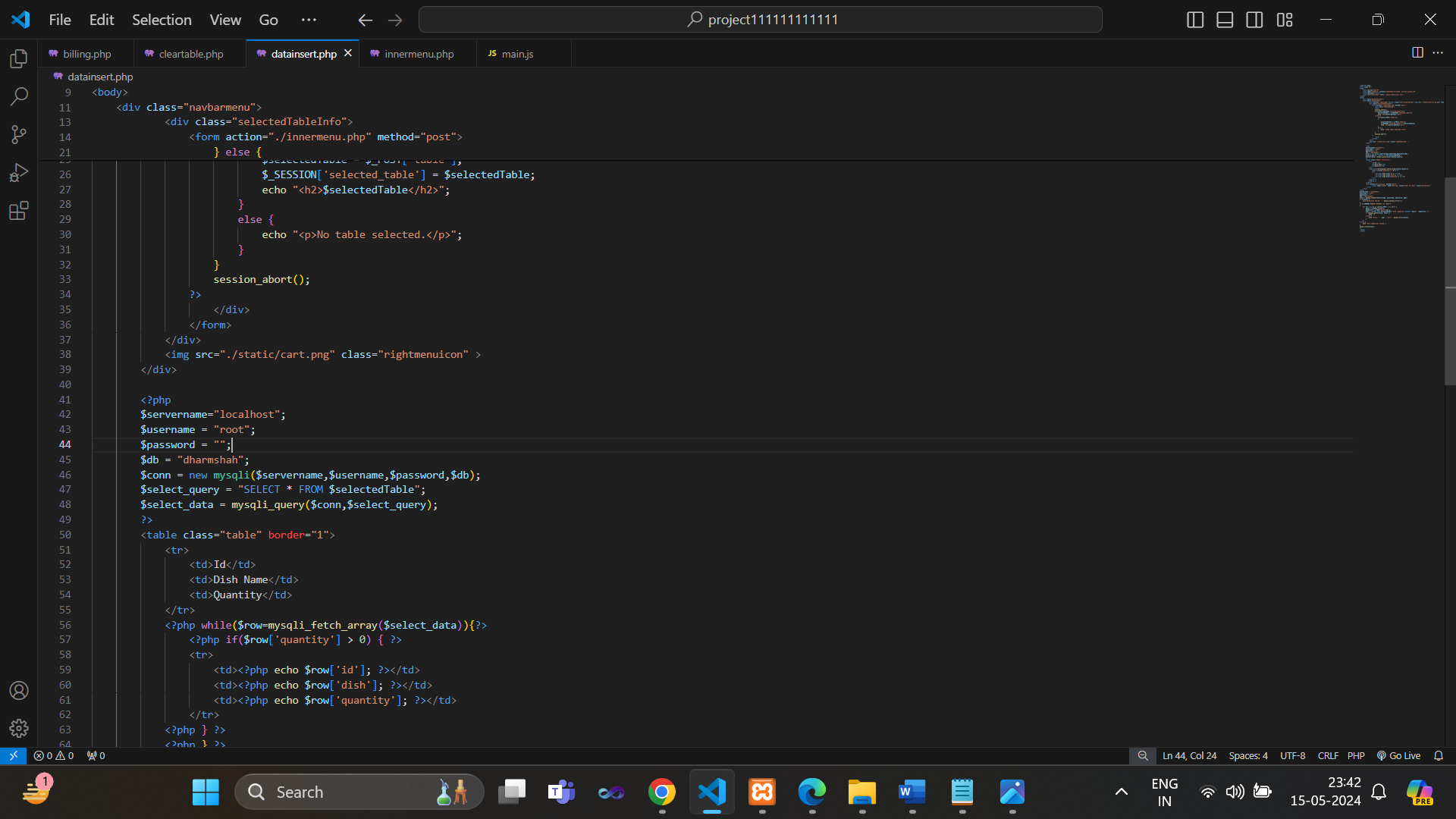
1. *Dynamic Content Generation:*

* *Menu items are dynamically generated using PHP within the HTML structure.*
* *Hidden input fields for dish names and quantities are dynamically generated based on the selected items when placing an order.*

***Output:***

******

***Data insert:***

******

***Explanation of Datainset.php:***

1. *Navigation Bar:*

* *A left arrow button (inside a form) for navigation back to the menu page (menu.php).*
* *A section for displaying information about the selected table, or a message if no table is selected. This section is wrapped in a form, suggesting it might handle table selection.*

1. *PHP Section:*

* *Checks if a table is selected. If selected, it displays the table number. If not selected, it checks if a table selection form is submitted ($\_POST['table']), then sets and displays the selected table.*

1. *Database Query and Display:*

* *PHP code connects to a MySQL database and retrieves data from a table based on the selected table.*
* *The table displays the "Id", "Dish Name", and "Quantity" of each item retrieved from the database. Only items with a quantity greater than 0 are displayed.*

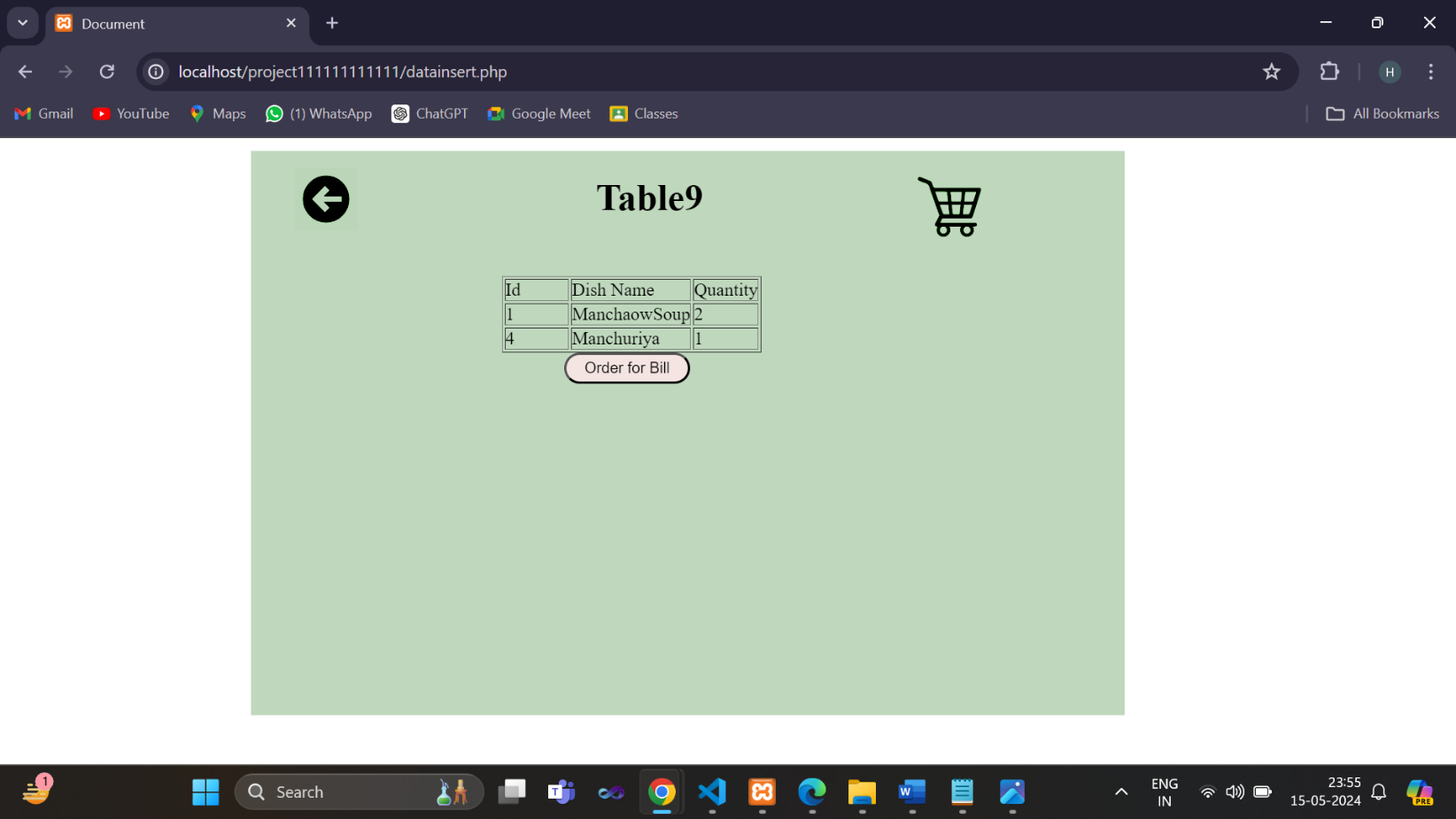
1. *Another PHP section handles form submission:*

* *It connects to the database and iterates over the submitted form data (presumably from the menu selection page).*
* *It constructs and executes SQL INSERT queries to insert the selected dishes and their quantities into the selected table in the database.*
* *If the form submission fails, it displays an error message.*

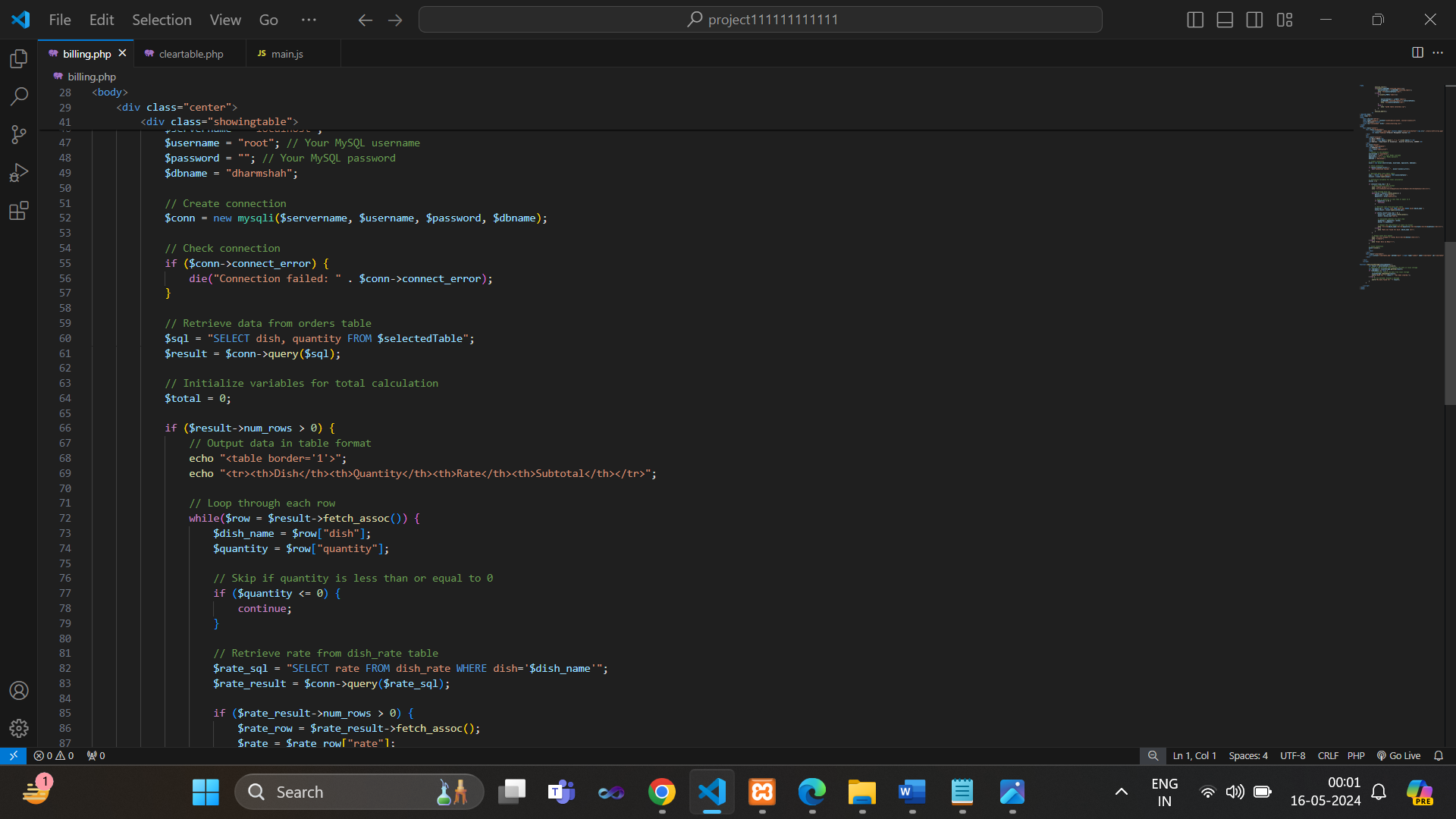
1. *Order for Billing:*

* *At the end of the page, there's a form with a single input button labeled "Order for Bill". This form likely submits the selected items for billing.*

***Output:***

******

***Billing page:***

******

***Explanation of Billing.php:***

1. *Invoice Section:*

* *It displays an invoice with details such as date and address.*
* *PHP is used to dynamically display the current date.*

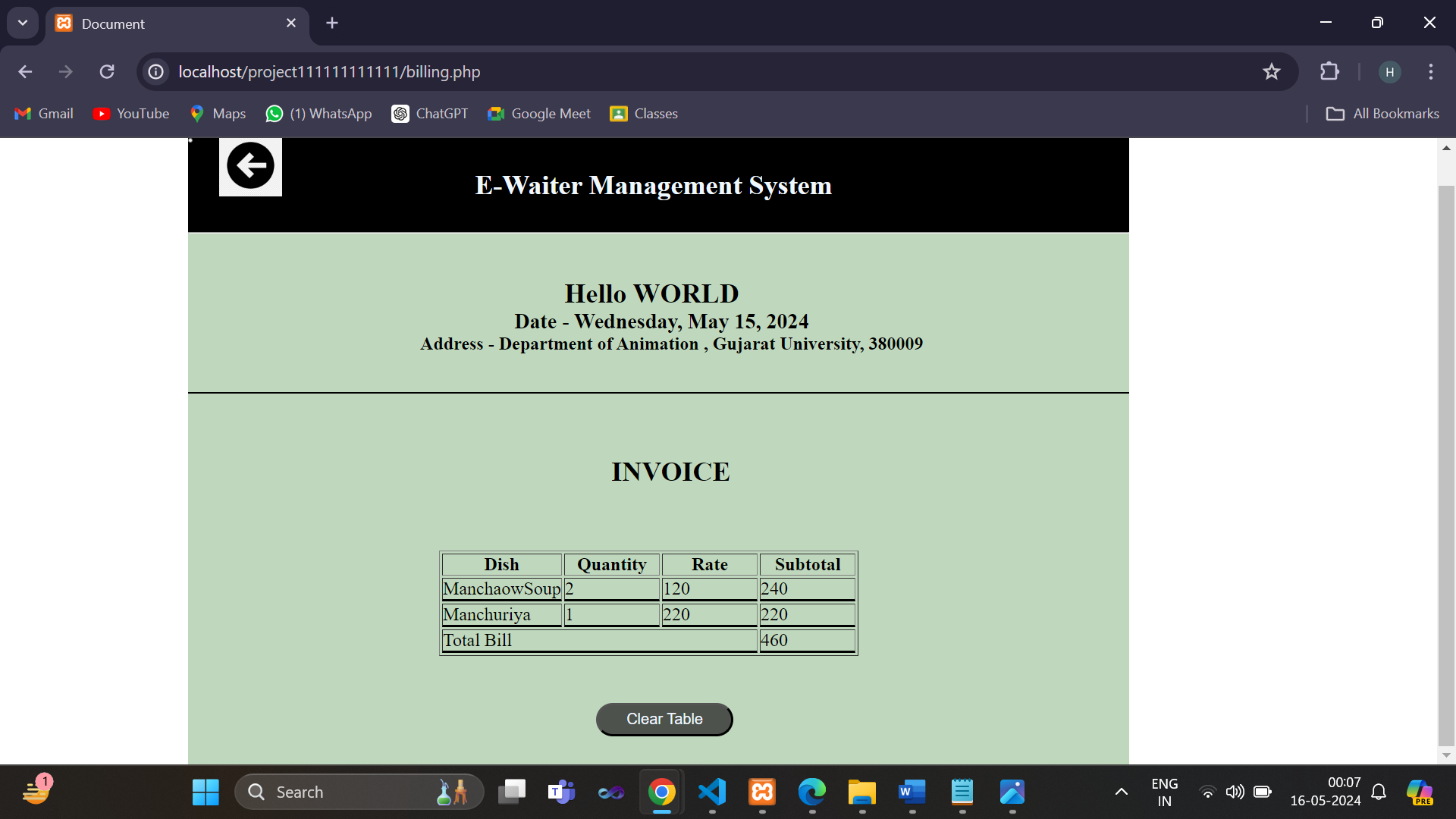
1. *Displaying Invoice Items:*

* *PHP code connects to the database and retrieves data from the selected table.*
* *It constructs an HTML table to display the retrieved data, including dish names, quantities, rates, and subtotals.*
* *It calculates the total bill amount.*
* *If no items are found in the selected table, it displays a message prompting to place an order.*

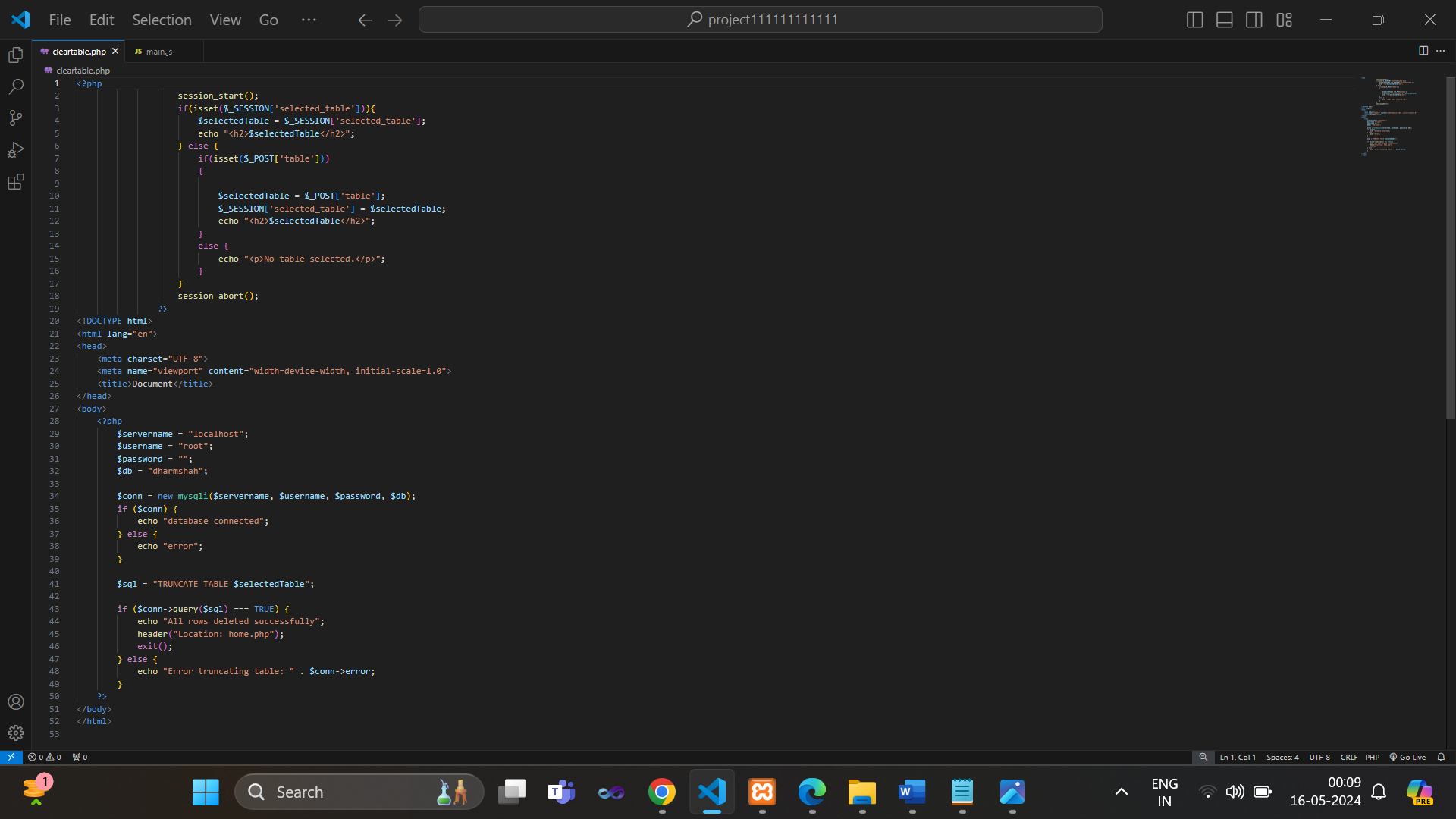
1. *Clear Table Button:*

* *It provides a button to clear the table, which triggers the clearLocalStorage() JavaScript function.*
* *The JavaScript function clears the data for the selected table from the local storage and provides an alert message confirming the action.*

***Output:***

******

***Clear Table:***

******

***Explanation of ClearTable.php:***

1. *PHP Section:*

* *It checks if the session variable $\_SESSION['selected\_table'] is set. If it is, it retrieves the selected table from the session and displays it.*
* *If the session variable is not set, it checks if a table selection form has been submitted ($\_POST['table']). If it has, it sets the selected table from the form data and saves it in the session variable.*
* *If neither the session variable nor the form submission is detected, it displays a message indicating that no table is selected.*
* *session\_abort() is called to terminate the session.*

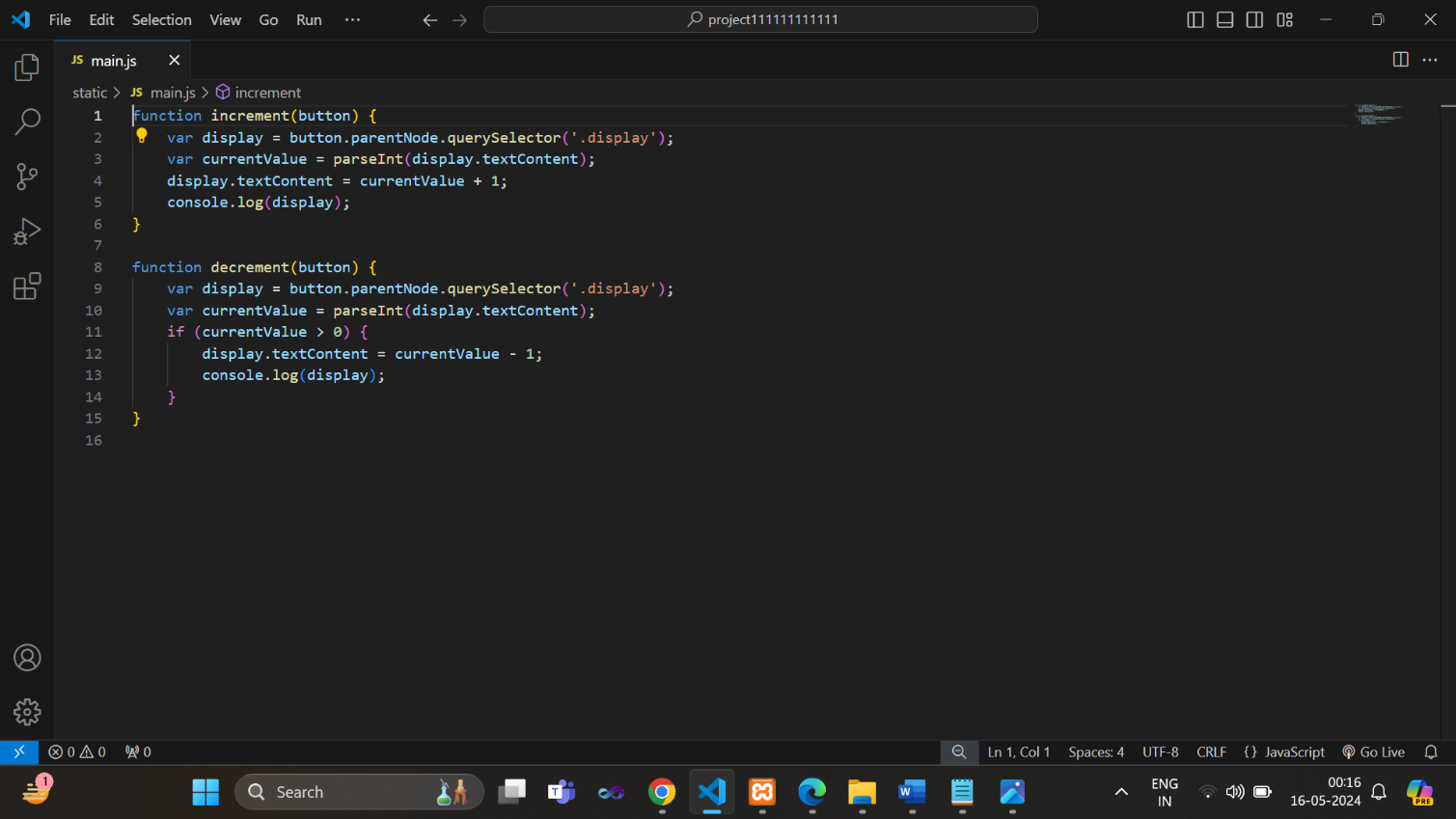
1. *Database Connection:*

* *PHP code establishes a connection to the MySQL database using the provided credentials ($servername, $username, $password, $db).*
* *It checks if the connection is successful and echoes either "database connected" or "error" accordingly.*

1. *Table Truncation:*

* *A SQL query is constructed to truncate the selected table (TRUNCATE TABLE $selectedTable).*
* *The query is executed using $conn->query($sql).*
* *If the query is successful, it echoes "All rows deleted successfully", redirects the user to "home.php" using header("Location: home.php"), and exits the script execution using exit().*
* *If there's an error executing the query, it echoes "Error truncating table" along with the specific error message from $conn->error.*

***Java Scripte page:***

******

***Explanation of Main.js:***

1. *increment() function:*

* *This function takes a parameter button, which represents the button element that triggered the function.*
* *It starts by finding the display element associated with the button using button.parentNode.querySelector('.display').*
* *The querySelector() method is used to find the first element within the parent node (button.parentNode) that matches the specified CSS selector (in this case, .display).*
* *Once the display element is found, it retrieves the current value from its text content using parseInt(display.textContent).*
* *It then increments the current value by 1 and updates the text content of the display element to reflect the new value using display.textContent = currentValue + 1.*
* *Finally, it logs the display element to the console for debugging purposes using console.log(display).*

1. *decrement() function:*

* *Similar to the increment() function, this function also takes a parameter button, representing the button element that triggered the function.*
* *It starts by finding the display element associated with the button using button.parentNode.querySelector('.display').*
* *It retrieves the current value from the display element's text content using parseInt(display.textContent).*
* *If the current value is greater than 0, it decrements the current value by 1 and updates the text content of the display element to reflect the new value using display.textContent = currentValue - 1.*
* *Additionally, it logs the display element to the console for debugging purposes using console.log(display).*

***Future Scope:***

1. *User Authentication and Profiles:*

* *Implement user authentication to allow waiters and staff to log in with unique credentials.*
* *Create user profiles to manage permissions and roles, such as waiter, manager, and admin.*

1. *Table Management:*

* *Implement table status tracking to indicate whether a table is occupied, ready for cleaning, or available for new customers.*

1. *Order Customization:*

* *Allow customers to customize their orders by adding special instructions or dietary preferences.*
* *Enable waiters to modify existing orders easily based on customer requests.*

1. *Real-time Order Updates:*

* *Implement real-time updates for orders, so both customers and staff can track the status of their orders.*
* *Provide notifications or alerts for important order events, such as order confirmation, preparation, and delivery.*

1. *Mobile App Support:*

* *Develop a mobile application companion for customers to browse the menu, place orders, and make reservations from their smartphones.*

1. *Integration with Payment Gateways:*

* *Integrate payment gateways to enable secure online payments for orders placed through the web application.*
* *Support various payment methods, including credit cards, mobile wallets, and digital currencies.*

1. *Feedback and Reviews:*

* *Allow customers to provide feedback and ratings for their dining experience.*
* *Enable restaurant management to respond to feedback and address customer concerns promptly.*

1. *Integration with Reservation System:*

* *Integrate a reservation system to allow customers to book tables in advance, reducing wait times and optimizing table turnover.*

1. *Social Media Integration:*

* *Enable customers to share their dining experiences on social media platforms directly from the web application.*
* *Integrate social media login options for user authentication and engagement.*

***Advantages:***

1. *Efficient Order Management:*

* *Streamlines the process of taking orders, reducing errors and improving order accuracy.*
* *Waiters can input orders directly into the system, eliminating the need for handwritten notes or order pads.*

1. *Improved Communication:*

* *Facilitates communication between waiters, kitchen staff, and management, ensuring smooth coordination and faster service.*
* *Enables staff to communicate special requests or modifications to orders more effectively.*

1. *Accurate Billing and Payment Processing:*

* *Automatically calculates the total bill based on the items ordered, reducing errors in billing.*
* *Integrates with payment gateways to enable secure and convenient payment options for customers.*

1. *Table Management:*

* *Helps staff manage table assignments and track table availability, optimizing seating arrangements and minimizing wait times.*
* *Improves table turnover by efficiently managing reservations and walk-in customers.*

1. *Data Analysis and Insights:*

* *Generates reports and analytics on sales trends, customer preferences, and peak hours, enabling informed decision-making for restaurant management.*
* *Provides valuable insights into customer behavior and preferences, allowing for targeted marketing efforts and menu optimizations.*

1. *Scalability and Adaptability:*

* *Can be scaled to accommodate the needs of small independent restaurants as well as large restaurant chains.*
* *Adaptable to changes in menu offerings, seasonal variations, and evolving customer preferences.*

1. *User-Friendly Interface:*

* *Intuitive Design: Simple and clean interface for easy navigation.*
* *Multi-Language Support: Available in multiple languages to cater to diverse clientele.*

***Disadvantages:***

1. *Technical Issues:*

* *Reliance on technology can lead to technical glitches, such as server downtime, software bugs, or connectivity issues, disrupting service and causing delays in order processing.*

1. *Learning Curve:*

* *Staff members, particularly older or less tech-savvy individuals, may require training to effectively use the application, leading to initial resistance and a learning curve that could affect productivity.*

1. *Resistance to Change:*

* *Some customers may prefer traditional methods of ordering and payment, such as speaking to a waiter or paying with cash. Introducing a digital ordering system may face resistance from certain customer segments, particularly older demographics or those who are less comfortable with technology.*

1. *Compatibility Issues:*

* *Ensuring compatibility with various devices, operating systems, and browsers can be challenging. Compatibility issues may arise, particularly when using the application across­ different types of hardware or software platforms, leading to inconsistencies in performance and user experience.*