A PROJECT ON

Household Service Management System

SUBMITTED IN

PARTIAL FULFILLMENT OF THE REQUIREMENT

FOR THE COURSE OF DIPLOMA IN ADVANCED COMPUTING FROM CDAC

SUNBEAM INSTITUTE OF INFORMATION TECHNOLOGY

Hinjawadi

SUBMITTED BY:

Rohit Sudam Gorde

Amol Rajendra Morankar

Shailendra Singh

Abhijeet Pandurang Jambkar

UNDER THE GUIDENCE OF:

Snehal Jadhav Sir

Faculty Member

Sunbeam Institute of Information Technology, Pune

CERTIFICATE

This is to certify that the project work under the title ‘Household Services Management System’ is done by Rohit Gorde in partial fulfillment of the requirement for award of Diploma in Advanced Computing Course.

Mr. Yogesh Kolhe

Project Guide Snehal Jadhav **Course Co-Coordinator**

Date:16/08/2024

ACKNOWLEDGEMENT

A project usually falls short of its expectation unless aided and guided by the right persons at the right time. We avail this opportunity to express our deep sense of gratitude towards Mr. Nitin Kudale (Center Coordinator, SIIT, Pune) and Mr. Yogesh Kolhe (Course Coordinator, SIIT ,Pune) .

We are deeply indebted and grateful to them for their guidance, encouragement and deep concern for our project. Without their critical evaluation and suggestions at every stage of the project, this project could never have reached its present form.

Last but not the least we thank the entire faculty and the staff members of Sunbeam Institute of Information Technology, Pune for their support.

Rohit Gorde

Amol Morankar

Shailendra Singh

Abhijeet Jambkar

0324 PG-DAC

SIIT Pune

1. Introduction

- **1.1 Project Overview**:
    - The Household Services Management System is designed to connect customers with service providers for various household tasks such as plumbing, electrical work, cleaning, and other maintenance services. The system allows customers to book services, track service history, and make payments online.

- **1.2 Objectives**:
    - To create a user-friendly platform that simplifies the process of finding and booking household services.
    - To provide a reliable and efficient system for managing service providers and customer requests.
    - To ensure secure transactions and data management.
    - Simplify Household Services: Our primary objective is to streamline the household services experience by offering a user-friendly platform where residents can effortlessly schedule and avail services such as appliance maintenance and repair- AC, Refrigerator, washing machine, geyser, purifier. Home cleaning – full home cleaning, room cleaning, bathroom cleaning, kitchen cleaning, other services like pest control and gardening, plumbing, carpentry, fan and light fixture repairs.
    - Enhance Accessibility: By providing an online platform, we aim to make household services more accessible to urban residents, eliminating the need for time-consuming searches or multiple service provider interactions. Users can easily browse available services, select desired options, and schedule appointments at their convenience.
    - Ensure Quality and Reliability: We are committed to maintaining high standards of service quality and reliability. Through rigorous vetting processes and partnerships with trusted service providers, we ensure that users receive professional and dependable assistance for their household needs.
    - Promote Convenience: Our platform offers a seamless and hassle-free experience from booking to service delivery. Users can track service requests, receive notifications, and provide ratings, all within the same application interface. Additionally, our scheduling system allows for flexible appointment timings to accommodate varying user preferences.
    - Technological Approach: By combining innovative technology with a customer-centric approach, the Household Services Management System aims to revolutionize the way city households' access and manage essential services, ultimately enhancing their quality of life.

- **1.3 Scope**:
    - Scope: 1. Service Offerings: The web application will facilitate the booking of various household services including AC repair, home cleaning, bathroom cleaning, kitchen cleaning, pest control, plumbing, carpentry, fan and light fixture repairs, and purifier servicing.
    - 2. User Management: Users will be able to register, create profiles, and manage their accounts securely. Profile management functionalities will include viewing booking history, updating personal information.
    - 3. Service Booking: Users can browse available services, select desired options, and schedule appointments based on their preferred date and time. The booking process will include real-time availability checks and confirmation notifications to users
    - 4. Service Provider Management: Administrators will have the capability to onboard and manage service providers. Service providers will have access to their schedules, service requests, and customer details through a dedicated interface.
    - 5. Scheduling and Notifications: The system will feature a scheduling mechanism to manage service appointments efficiently, avoiding conflicts and ensuring timely service delivery. Users and service providers will receive automated emails regarding booking confirmations, reminders, and updates.
    - 7. Admin Dashboard: An admin dashboard will be provided for administrators to oversee the entire system, manage users, service providers, bookings.
    - 8. Cross-Platform Compatibility: The web application will be designed to be responsive and compatible with various devices and screen sizes, ensuring a consistent user experience across desktops, tablets, and mobile devices.

2. System Requirements

- **2.1 Functional Requirements**:
    - **User Registration and Authentication**: Users (customers and service providers) can register, log in, and manage their profiles.
    - **Service Booking**: Customers can browse available services, select a service provider, and book a service for a specified date and time.
    - **Order Management**: Customers can view their order history, track ongoing services, and provide feedback.
    - **Service Provider Management**: Service providers can manage their services, view booked orders, and update order status.
    - **Admin Panel**: Admins can manage users, services, transactions, and view system analytics.
- **2.2 Non-Functional Requirements**:
    - **Performance**: The system should handle multiple users simultaneously without performance degradation.
    - **Security**: Data should be encrypted, and secure authentication mechanisms should be in place.
    - **Scalability**: The system should be scalable to accommodate future growth in users and services.
    - **Usability**: The user interface should be intuitive and easy to navigate.

3. System Design

- **3.1 System Architecture**:
    - **Client-Side**: React.js for the front-end to create an interactive user interface.
    - **Server-Side**: SpringBoot with RESTful APIs for the backend to handle API requests and business logic.
    - **Database**: MySQL for storing user data, service details, orders, and transactions.
    - **Tables**

3. DESIGN

3.1 Database Design

The following table structures depict the database design.

Table1: Customer

Key Type/ Constraint	Column Name	Data Type	Length	Allow Null (1=Yes;0=No)
primary	customer_id	varchar	5	0
0	created_on	date	6	1
unique	email	varchar	30	0
0	first_name	varchar	30	0
0	last_name	varchar	30	1
unique	mobile_no	varchar	15	0
0	password	varchar	100	0
0	updated_on	datetime	6	1
0	image_path	varchar	255	1

Table2: Customer_Address

Key Type/ Constraint	Column Name	Data Type	Length	Allow Null (1=Yes;0=No)
primary	id	bigint	64	0
0	created_on	date	6	1
0	updated_on	datetime	6	1
0	add_line_one	varchar	100	1

0	add_line_two	varchar	100	1
0	city	varchar	20	0
0	country	varchar	20	0
0	landmark	varchar	50	1
0	state	varchar	20	0
0	zip_code	varchar	20	0
multiple	customer_id	varchar	5	0

Table3: Partner

Key Type/ Constraint	Column Name	Data Type	Length	Allow Null (1=Yes;0=No)
primary	partner_id	varchar	5	0
0	created_on	date	6	1
unique	email	varchar	30	0
0	first_name	varchar	30	0
0	last_name	varchar	30	1
unique	mobile_no	varchar	15	0
0	password	varchar	100	0
0	updated_on	datetime	6	1
unique	card_no	varchar	20	0
0	card_type	varchar	20	0
0	id_image_path	varchar	255	1
0	is_approved	tinyint	1	1
0	is_deleted	tinyint	1	1
0	partner_image_path	varchar	255	1
0	service_title	varchar	50	1
multiple	address_id	bigint	64	1

Table4: Partner_Address

Key Type/ Constraint	Column Name	Data Type	Length	Allow Null (1=Yes;0=No)
primary	id	bigint	64	0
0	created_on	date	6	1
0	updated_on	datetime	6	1
0	add_line_one	varchar	100	1
0	add_line_two	varchar	100	1
0	city	varchar	20	0
0	country	varchar	20	0
0	landmark	varchar	50	1
0	state	varchar	20	0
0	zip_code	varchar	20	0

Table5: Admin

Key Type/ Constraint	Column Name	Data Type	Length	Allow Null (1=Yes;0=No)
primary	emp_id	varchar	5	0
0	created_on	date	6	1
unique	email	varchar	30	0
0	first_name	varchar	30	0
0	last_name	varchar	30	1
unique	mobile_no	varchar	15	0
0	password	varchar	100	0
0	updated_on	datetime	6	1

Table6: Orders

Key Type/ Constraint	Column Name	Data Type	Length	Allow Null (1=Yes;0=No)
primary	order_id	varchar	9	0
0	order_date	date	6	0
unique	order_amount	double	8	1
0	updated_on	datetime	6	1
multiple	address_id	bigint	64	0
multiple	customer_id	varchar	5	0

Table7: Order_Details

Key Type/ Constraint	Column Name	Data Type	Length	Allow Null (1=Yes;0=No)
primary	id	bigint	64	0
0	created_on	date	6	1
0	updated_on	datetime	6	1
0	order_status	varchar	10	1
0	package_qty	int	32	0
0	service_date	date	6	0
multiple	order_id	varchar	9	0
multiple	partner_id	varchar	5	0
multiple	rating_id	bigint	64	1
multiple	package_id	bigint	64	0

Table8: Service

Key Type/ Constraint	Column Name	Data Type	Length	Allow Null (1=Yes;0=No)
primary	id	bigint	64	0
0	created_on	date	6	1
0	updated_on	datetime	6	1
0	service_description	varchar	255	1
unique	service_title	varchar	50	0
multiple	category_id	bigint	64	0

Table9: Service_Category

Key Type/ Constraint	Column Name	Data Type	Length	Allow Null (1=Yes;0=No)
primary	id	bigint	64	0
0	created_on	date	6	1
0	updated_on	datetime	6	1
0	service_description	varchar	255	1
unique	category_title	varchar	50	0

Table10: Service_Package

Key Type/ Constraint	Column Name	Data Type	Length	Allow Null (1=Yes;0=No)
primary	id	bigint	64	0
0	created_on	date	6	1
0	updated_on	datetime	6	1
0	package_price	double	8	1

unique	package_title	varchar	50	0
multiple	service_id	bigint	64	0

Table11: Cart

Key Type/ Constraint	Column Name	Data Type	Length	Allow Null (1=Yes;0=No)
primary	id	bigint	64	0
0	created_on	date	6	1
0	updated_on	datetime	6	1
multiple	customer_id	varchar	5	0

Table12: Cart_Item

Key Type/ Constraint	Column Name	Data Type	Length	Allow Null (1=Yes;0=No)
primary	id	bigint	64	0
0	created_on	date	6	1
0	updated_on	datetime	6	1
0	quantity	int	32	0
multiple	cart_id	bigint	64	0
multiple	package_id	bigint	64	0

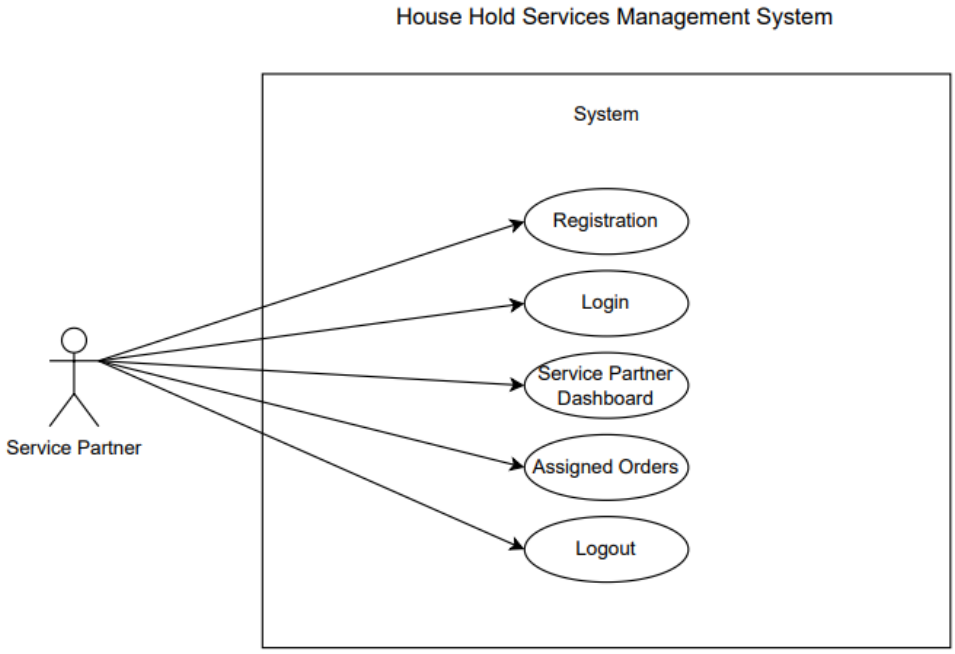
Table13: Rating

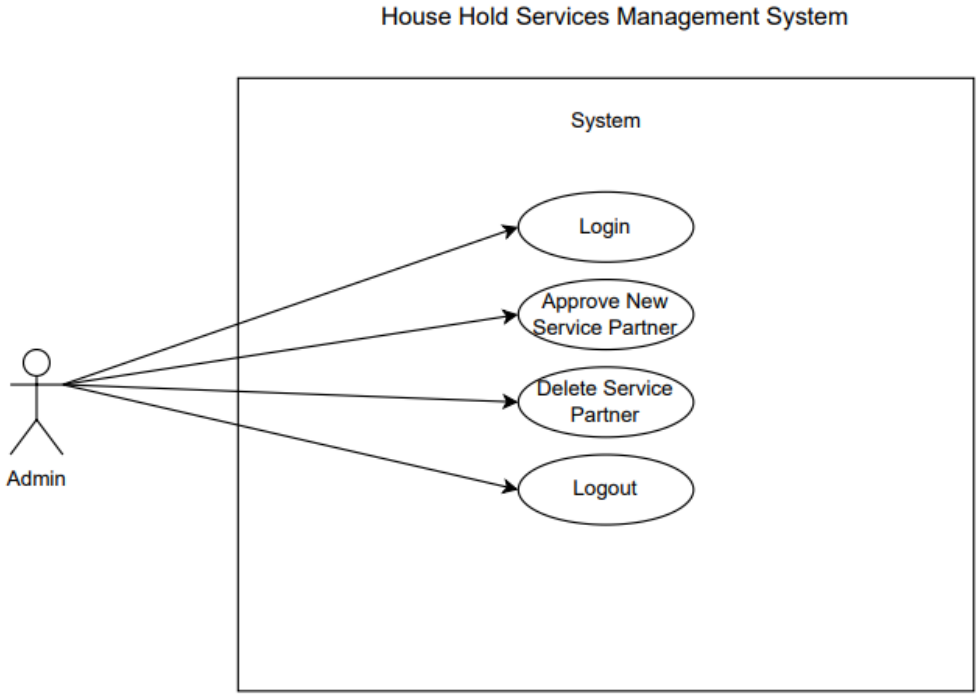
Key Type/ Constraint	Column Name	Data Type	Length	Allow Null (1=Yes;0=No)
primary	id	bigint	64	0
0	created_on	date	6	1
0	updated_on	datetime	6	1
0	rating	int	32	1
unique	order_details_id	bigint	64	0
multiple	package_id	bigint	64	0

-

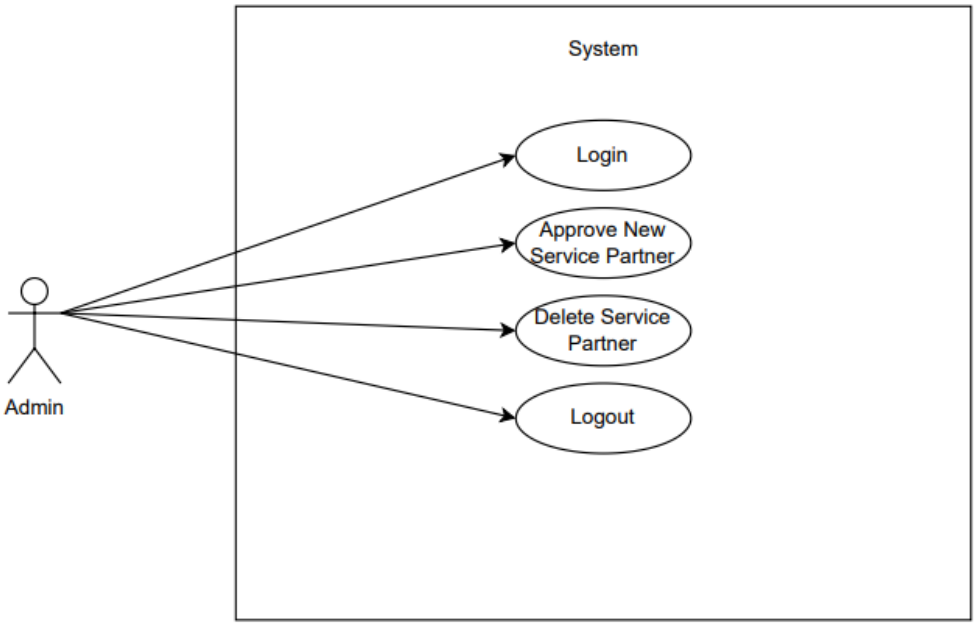
3.2 Use Case Diagrams:

- Diagrams representing the interaction between customers, service providers, and the system.



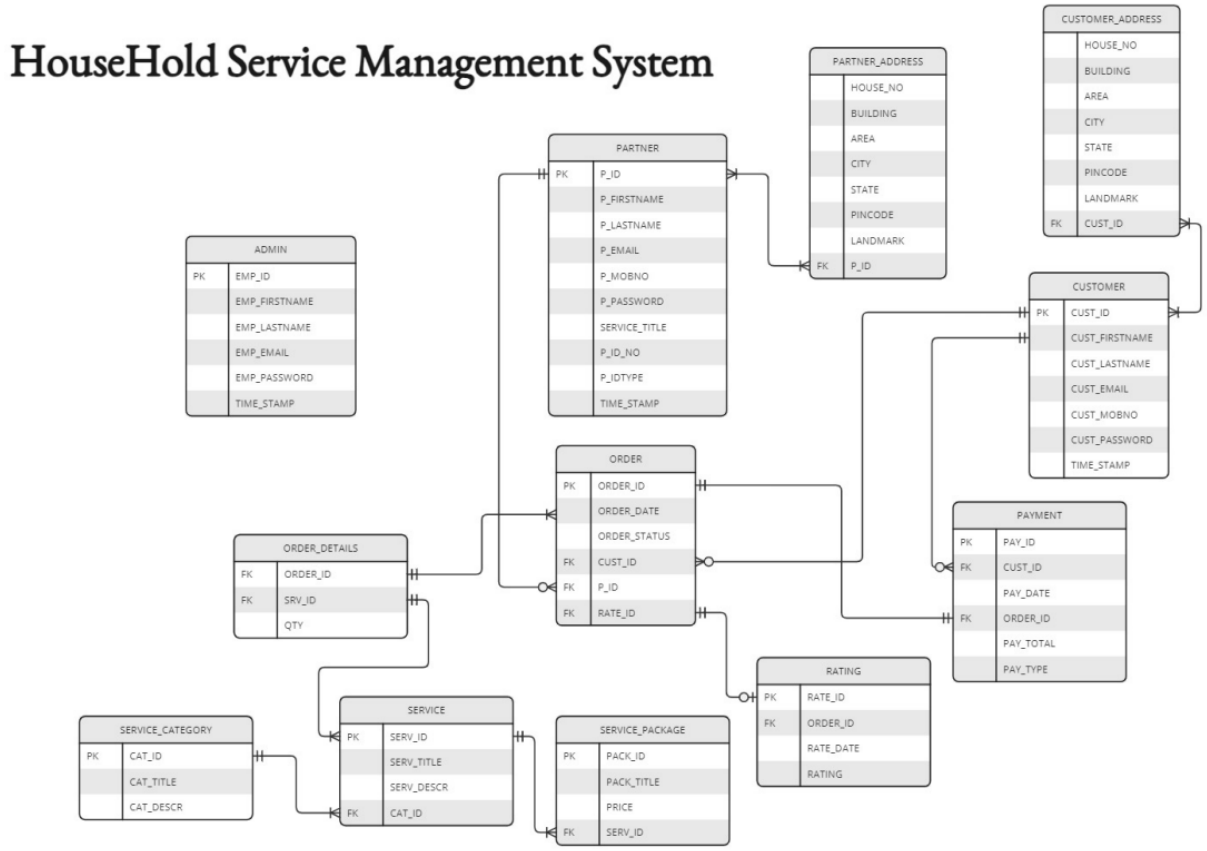


House Hold Services Management System



3.3 Entity-Relationship Diagram (ERD) with Class Diagram:

- Diagram illustrating the database schema, including tables for users, services, orders, payments, etc.



4. Implementation

- **4.1 Frontend Development**:
    - **UI Design**: Design of the user interface using HTML, CSS, Bootstrap, and React.js.
    - **Component Structure**: Breakdown of React components for modularity and reuse.
    - **State Management**: Use of Redux for managing global state across the application.
- **4.2 Backend Development**:
    - **API Development**: Development of RESTful APIs using SpringBoot for CRUD operations.
    - **Database Integration**: Implementation of database models and CRUD operations using MySQL.
    - **Authentication**: Implementation of JWT-based Spring Security authentication for secure login and session management.

5. Testing

- **5.1 Unit Testing**:
    - Testing of individual components and functions using testing frameworks like Jest for React and Mocha/Chai for Node.js.
- **5.2 Integration Testing**:
    - Testing the interaction between different modules to ensure they work together as expected.

6. Security

- **6.1 Data Encryption**:
    - Use of SHA256 Algorithm for encrypting data in transit.
    - Encryption of sensitive data stored in the database.
- **6.2 Authentication and Authorization**:
    - Implementation of role-based access control (RBAC) to manage user permissions.

7. CODING STANDARDS IMPLEMENTED

Naming and Capitalization

Below summarizes the naming recommendations for identifiers in Pascal casing is used mainly (i.e. capitalize first letter of each word) with camel casing (capitalize each word except for the first one) being used in certain circumstances.

Identifier	**Case**	**Examples**	**Additional Notes**
Class	Pascal	Person, BankVault, SMSMessage, Dept	Class names should be based on "objects" or "real things" and should generally be nouns. No '_' signs allowed. Do not use type prefixes like 'C' for class.
Method	Camel	getDetails, updateStore	Methods should use verbs or verb phrases.
Parameter	Camel	personName, bankCode	Use descriptive parameter names. Parameter names should be descriptive enough that the name of the parameter and its type can be used to determine its meaning in most scenarios.
Property	Pascal	ForeColor, BackColor	Use a noun or noun phrase to name properties.
Associated private member variable	_camelCase	_foreColor, _backColor	Use underscore camel casing for the private member variables
Exception Class	Pascal with "Exception" suffix	WebException,	

Comments

- Comment each type, each non-public type member, and each region declaration.
- Use end-line comments only on variable declaration lines. End-line comments are comments that follow code on a single line.
- Separate comments from comment delimiters (apostrophe) or // with one space.
- Begin the comment text with an uppercase letter.
- End the comment with a period.
- Explain the code; do not repeat it.

8. TEST REPORT

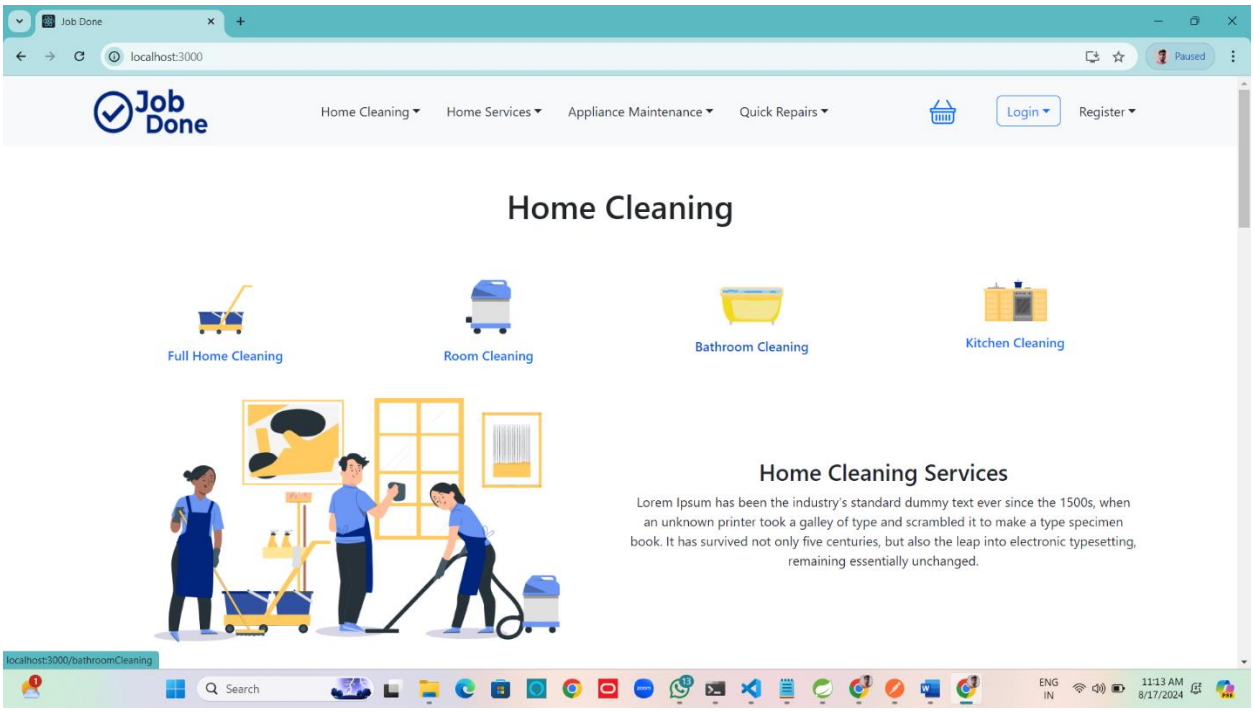
Another group called Linux did the testing and the report of the testing is given hereunder.

GENERAL TESTING:

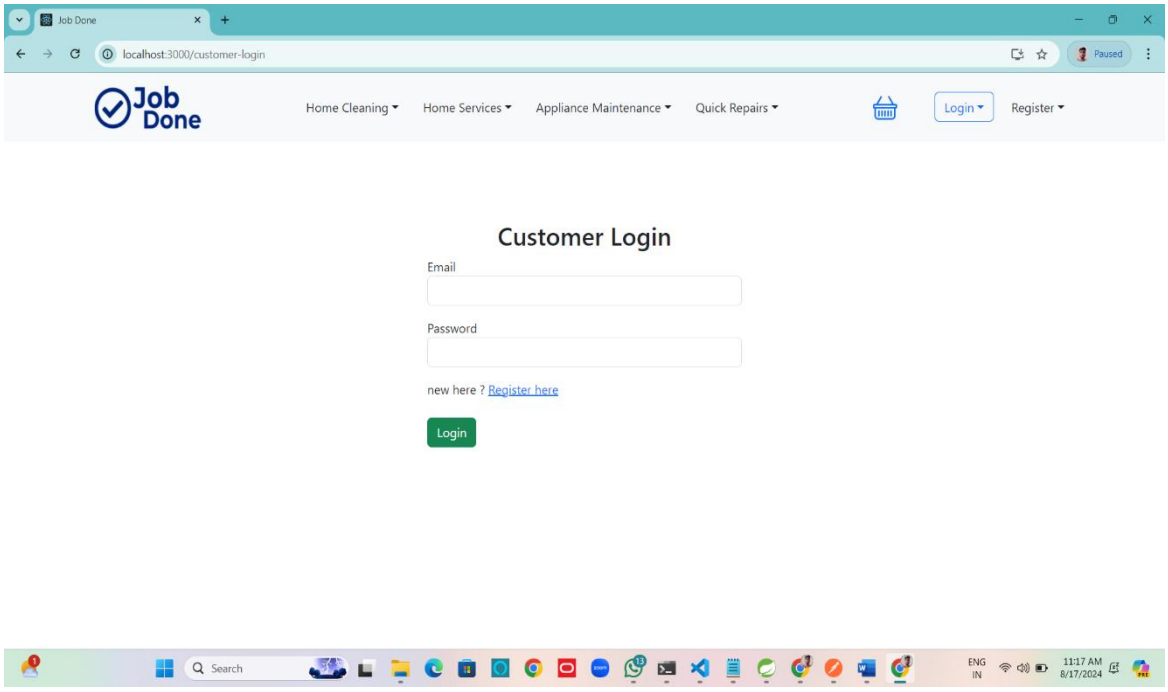
SR-NO	TEST CASE	EXPECTED RESULT	ACTUAL RESULT	ERROR MESSAGE
1	Register Page	Redirected to Next page	OK	Nothing
2	Login Page	Pop-up will come	Ok	Please enter username and password again .
3	Reset login	Only users password will be reseted	Ok	Nothing
4	Add Multiple addresses	Gives all addresses details	Ok	Nothing
5	Add Profile Image/verification Document	Gives profile image/ document image	Ok	Nothing
6	Checking login or not	User is logged in or not	Ok	Nothing
7	Approval of service partner	Successfully verifying/approving service partner	Ok	Nothing
8	Add to cart	Successfully adding items in cart	Ok	Nothing
9	Goto cart page	Save all cart items.	Ok	Nothing
10	Select date and address	selecting date and address.	Ok	Nothing
11	View order is placed.	It shows you order placed and service partner will get assigned.	Ok	Nothing
12	Logout	It will logout from user profile.	Ok	Nothing

9.Appendix

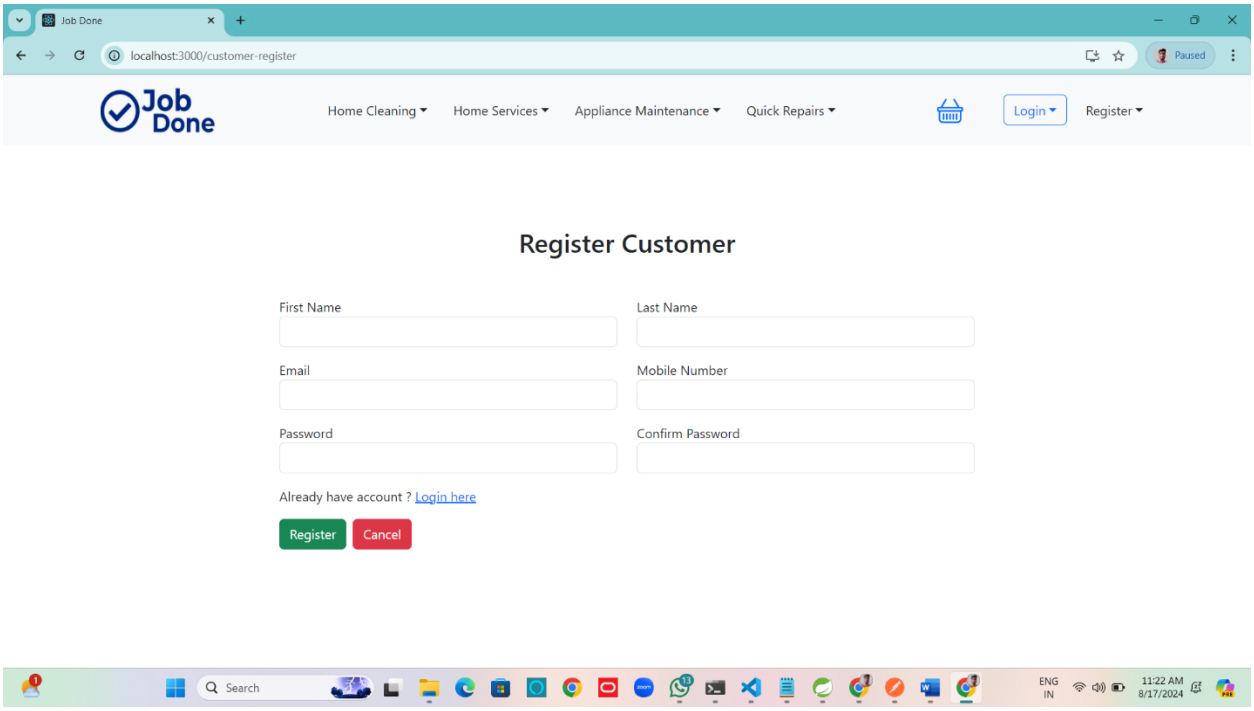
Home Page:



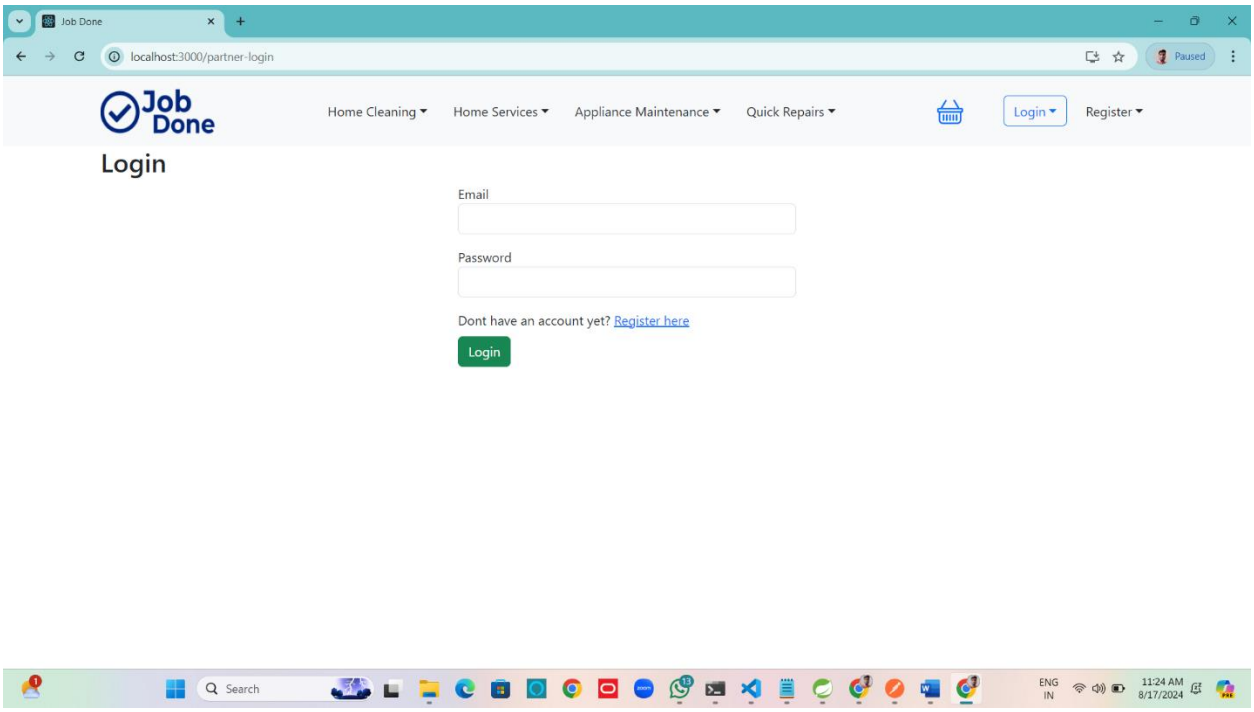
Customer Login:



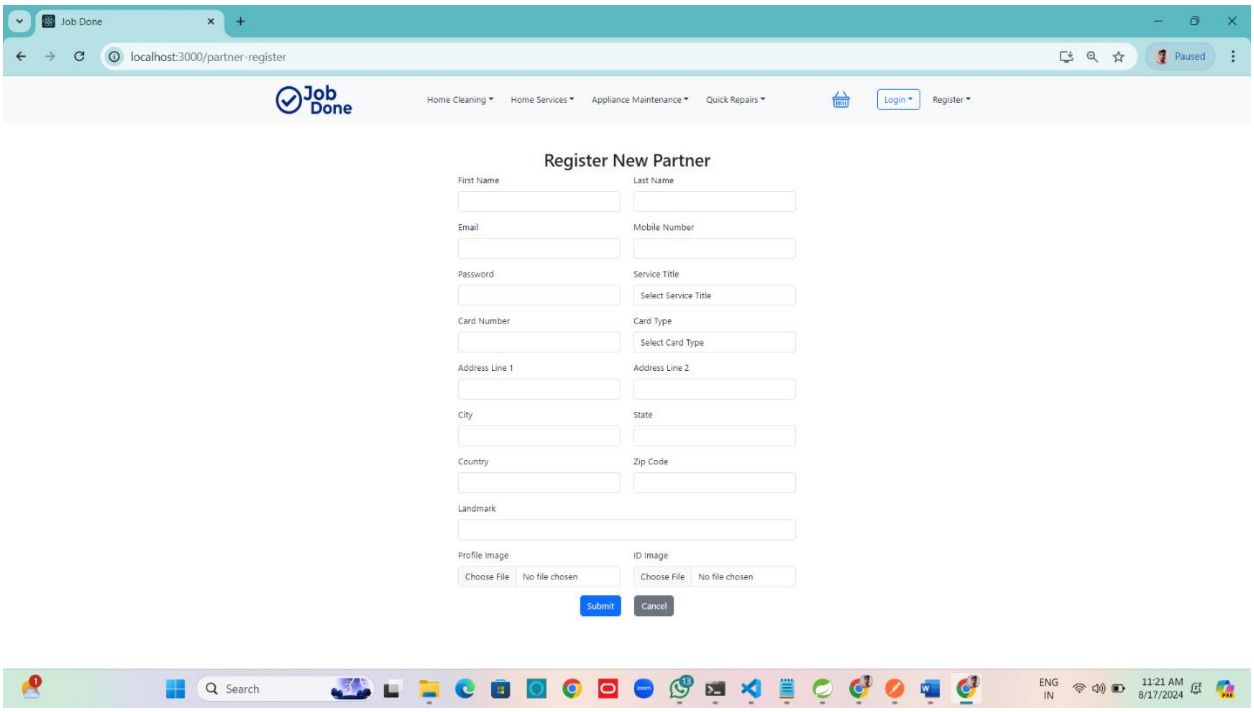
Customer Register:



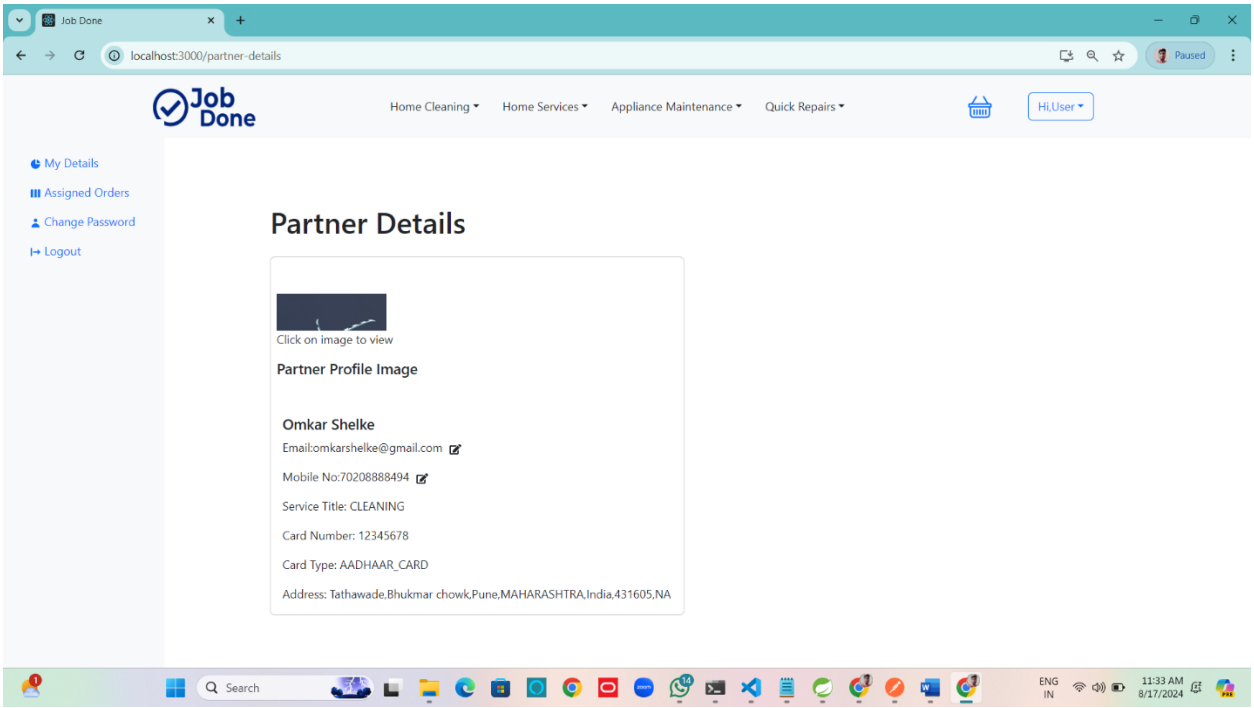
Partner login:



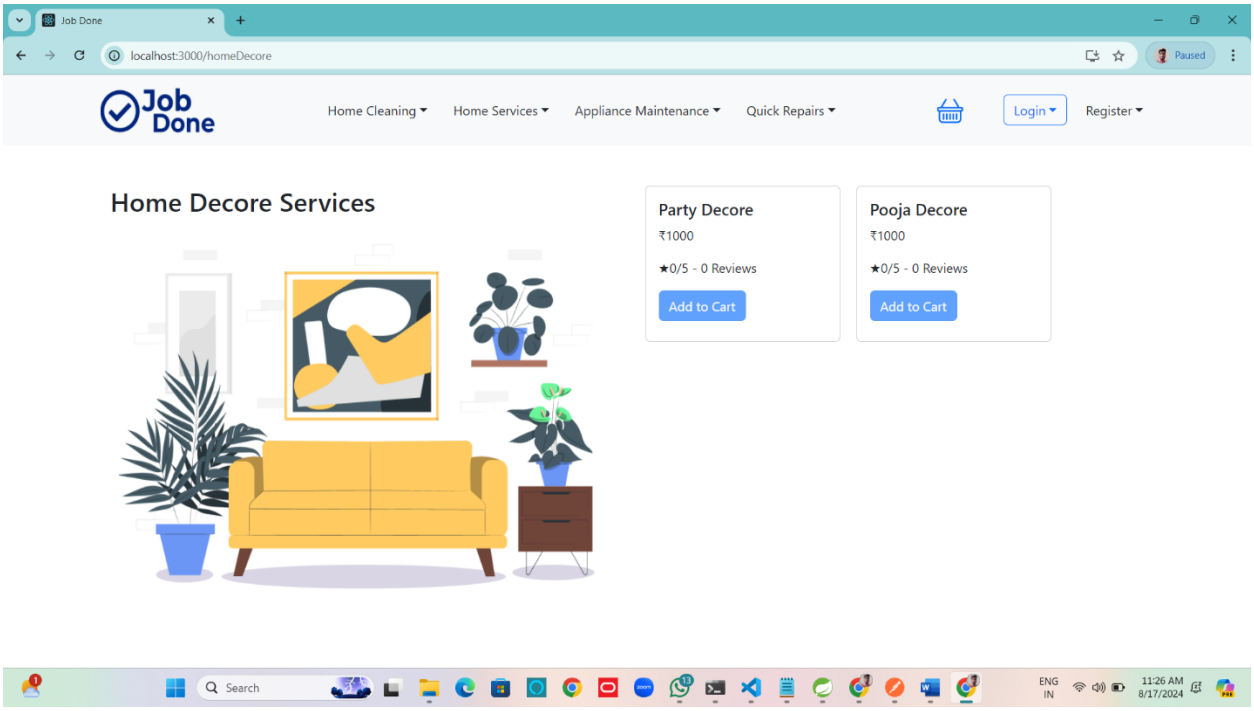
Partner Register:



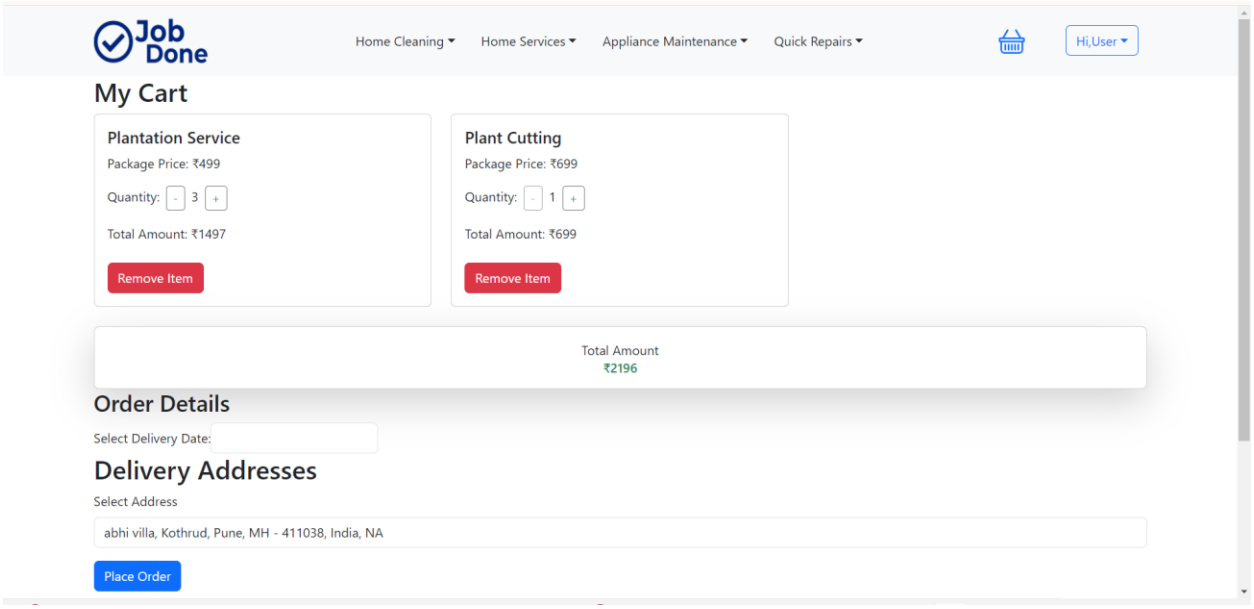
Partner Details(Image Fuctionality):



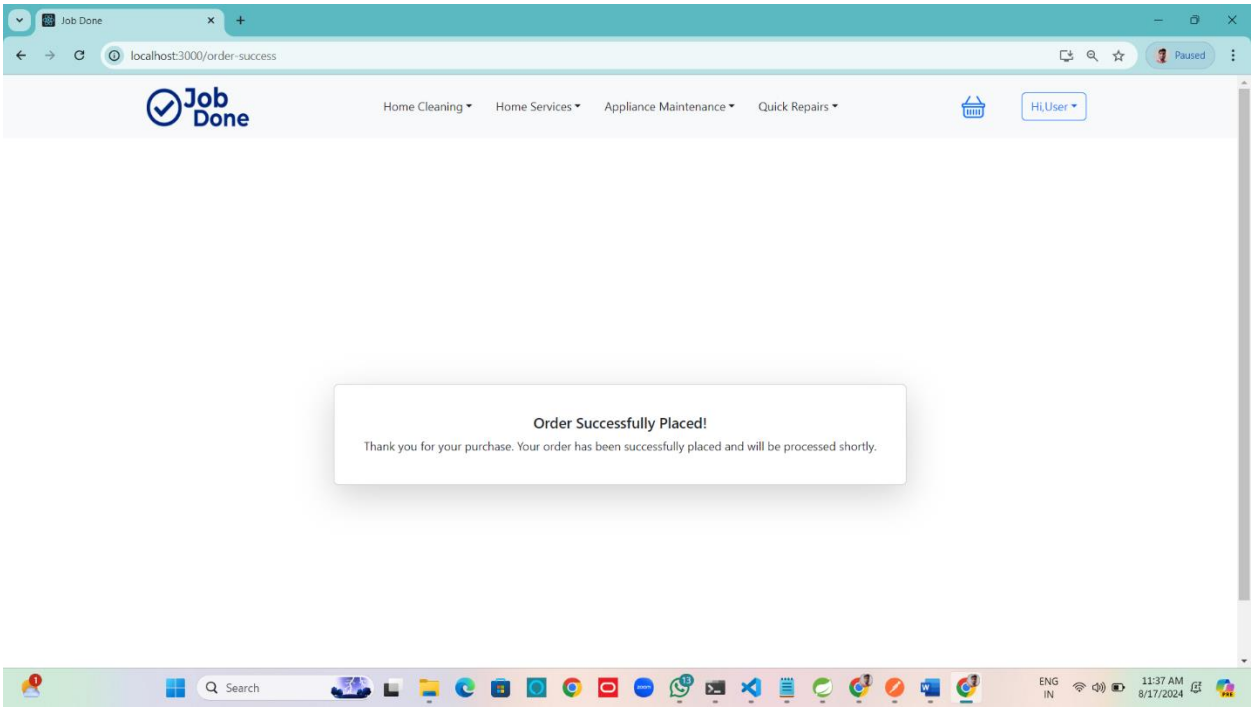
Add Service to cart:



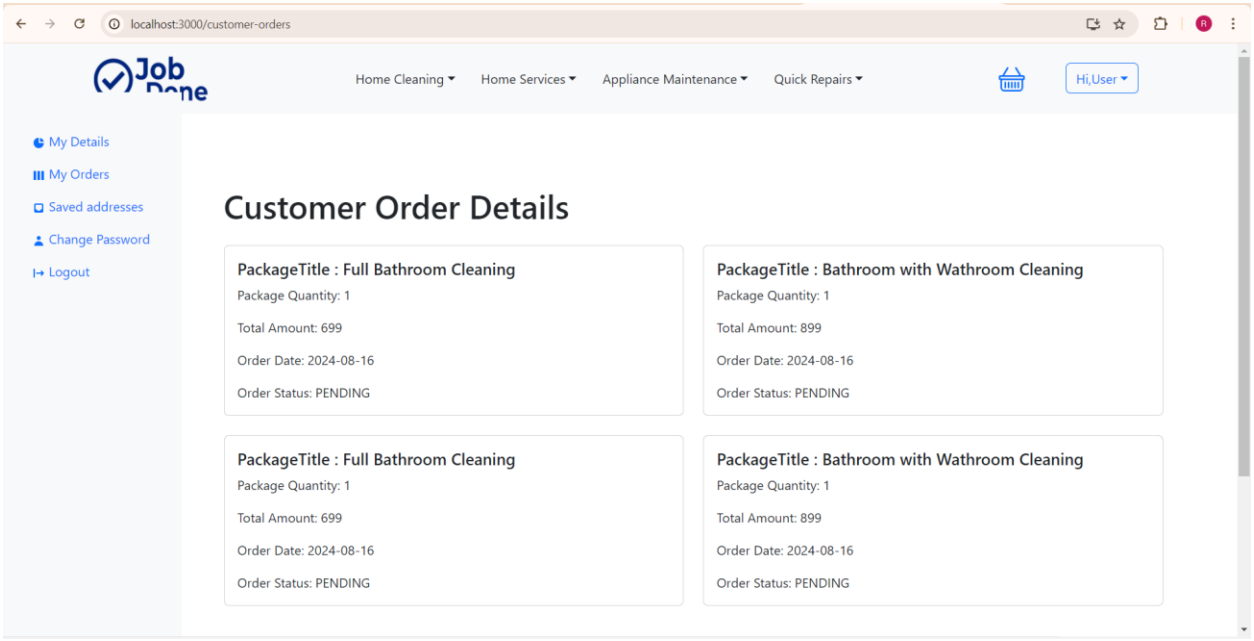
Cart Page:



Order Placed:



Customer Order Details:



10. Conclusion

- **10.1 Project Summary**:
    - The Household Services Management System is a comprehensive platform designed to bridge the gap between customers and service providers for a wide range of household tasks, such as plumbing, electrical work, cleaning, and general maintenance. The project aimed to create an easy-to-use web application where customers can browse and book services, and track the progress of their orders..
- **10.2 Project Outcomes**:
    - User-Friendly Interface: The project has successfully created an intuitive and responsive user interface that allows users to easily navigate through different services, book appointments, and manage their orders.
    - Efficient Service Management: The system has streamlined the process of managing service providers and customer bookings, making it easier for both parties to interact and fulfill service requests
- **10.3 Future Scope**:
    - Mobile Application Development: To reach a broader audience, the next step could involve developing a mobile application for Android and iOS platforms, offering users the convenience of accessing services on the go.
    - AI-Based Recommendations: Implementing machine learning algorithms to analyze user behavior and service history could provide personalized service recommendations, enhancing user experience.

9. References

http://www.google.com

https://www.urbancompany.com

http://www.webdevelopersjournal.com

http://www.w3.org

http://www.wikipedia.org

https://jwt.io/