**A Project Report**

**On**

**ServeU**

submitted for partial fulfillment of the requirements

for the award of the degree of

Bachelor of Technology

in

Computer Science

**Submitted by**

MD. Faizal (2000290120095)

Manish Kumar Mandavi (2000290120090)

Harsh Choudhary (2000290120071)

**Under supervision of**

Prof. Akash Goel



**KIET Group of Institutions, Ghaziabad**

**Dr. A.P.J. Abdul Kalam Technical University, Lucknow**

**May, 2024**

**DECLARATION**

We hereby declare that this submission is our own work and that, to the best of our knowledge and belief, it contains no material previously published or written by another person nor material which to a substantial extent has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgment has been made in the text.

Date:

Signature: Signature:

Name: Md Faizal Name: Manish Kumar

Roll No.: 2000290120095 Roll No.: 2000290120090

Signature:

Name: Harsh Choudhary

Roll No.: 2000290120071

**CERTIFICATE**

This is to certify that Project Report entitled “ServeU” which is submitted by Md Faizal, Manish Kumar Mandavi and Harsh Choudhary in partial fulfillment of the requirement for the award of degree B. Tech. in Department of Computer Science of Dr. A.P.J. Abdul Kalam Technical University, Lucknow is a record of the candidates own work carried out by them under my supervision. The matter embodied in this report is original and has not been submitted for the award of any other degree.

.

**Date: Supervisor**

Akash Goel

(Assistant Professor)

ACKNOWLEDGEMENT

It gives us a great sense of pleasure to present the report of the B. Tech Project undertaken during B. Tech. Final Year. We owe special debt of gratitude to Professor Akash Goel, Department of Computer Science, KIET, Ghaziabad, for his constant support and guidance throughout the course of our work. His sincerity, thoroughness and perseverance have been a constant source of inspiration for us. It is only his cognizant efforts that our endeavors have seen light of the day.

We also take the opportunity to acknowledge the contribution of Dr. Ajay Kumar Shrivastava, Head of the Department of Computer Science, KIET, Ghaziabad, for his full support and assistance during the development of the project. We also do not like to miss the opportunity to acknowledge the contribution of all the faculty members of the department for their kind assistance and cooperation during the development of our project.

Last but not the least, we acknowledge our friends for their contribution in the completion of the project.

Date :

Signature: Signature:

Name: Md Faizal Name: Manish Kumar

Roll No.: 2000290120095 Roll No.: 2000290120090

Signature:

Name: Harsh Choudhary

Roll No.: 2000290120071

**ABSTRACT**

In today’s fast-paced world, bachelors often struggle with efficiently managing essential services, such as maid, laundry, mess/tiffing services, and general daily services. The main challenges they face include finding reliable service providers, comparing prices, and coordinating bookings, leading to wasted time and subpar experiences. on the other side different types of daily service providers often find difficulty for getting users according to their skills due to communication gap between provider and receiver [1].

To address these issues, our research introduces user-friendly web application [4] designed to streamline the process of discovering, booking, and managing these services according to their schedule and convenience and also provides a communication channel between service provider and user. A feedback-based rating system can improve reliability of the service provider and improve the service provider skills.

|  |  |
| --- | --- |
| **TABLE OF CONTENTS** |  |
|  | **Page No.** |
| DECLARATION…………………………………………………………… | *i.* |
| CERTIFICATE……………………………………………………………... | *ii.* |
| ACKNOWLEDGEMENTS………………………………………………… | *iii.* |
| ABSTRACT………………………………………………………………… | *iv.* |
| LIST OF FIGURES ………………………………………………………… | *vii.* |
| LIST OF ABBREVIATIONS………………………………………………. | *viii.* |
|  | **Page No.** |
| CHAPTER 1 INTRODUCTION  1.1 INTRODUCTION TO PROJECT  1.2 PROJECT CATEGORY  1.3 OBJECTIVES  1.4 PROBLEM FORMULATION  1.5 PROPOSED SYSTEM  1.6 UNIQUE FEATURES OF THE SYSTEM  CHAPTER 2. REQUIREMENT ANALYSIS AND SYSTEM SPECIFICATION  2.1 FEASIBILITY STUDY  2.2 SOFTWARE REQUIREMENT SPECIFICATION  2.3 SDLC MODEL TO BE USED  CHAPTER 3. SYSTEM DESIGN  3.1 DETAIL DESIGN  3.2 SYSTEM DESIGN USING DFD LEVEL 0 AND LEVEL 1  3.3 USE CASE DIAGRAM  3.4 DATABASE DESIGN  3.4.1 ER DIAGRAMS  3.5 CLASS DIAGRAM  3.6 ARCHITECTURAL DIAGRAM  3.7 ACTIVITY DIAGRAM  CHAPTER 4. IMPLEMENTATION, TESTING, AND MAINTENANCE  4.1 INTRODUCTION TO LANGUAGES, TOOLS AND TECHNOLOGIES USED FOR IMPLEMENTATION  4.2 TESTING TECHNIQUES AND TEST CASES USED  CHAPTER 5. RESULTS AND DISCUSSIONS  5.1 USER INTERFACE REPRESENTATION  5.1.1 BRIEF DESCRIPTION OF VARIOUS MODULES OF THE SYSTEM  5.2 SNAPSHOTS OF SYSTEM  5.3 BACK-ENDS REPRESENTATION  5.3.1 SNAPSHOTS OF DATABASE  CHAPTER 6. CONCLUSION AND FUTURE SCOPE  REFERENCES | 9  9  10  11  12  12  13  14  14  15  24  25  25  28  29  30  35  36  37  **Page No.**  41  42  42  43  44  44  46  47  48  51  52  53 |
|  |  |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **Figure No.** | **Description** | **Page No.** |
| 1 | Model Connection | 25 |
| 2 | Flow Diagram | 26 |
| 3 | 0-Level DFD | 28 |
| 4 | 1-Level DFD | 28 |
| 5 | Use Case Diagram | 29 |
| 6 | Database Tables | 30 |
| 7 | Entity Relation Diagram | 38 |
| 8 | Class Diagram | 39 |
| 9 | Architecture Diagram | 40 |
| 10 | Activity Diagram | 40 |
| 11 | Landing Page | 43 |
| 12 | Service Category Page | 43 |
| 13 | Service Request Page | 44 |
| 14 | Service Listing Page | 44 |
| 15 | Chat Component | 44 |
| 16 | Landing Page | 46 |
| 17 | Service Listing page | 48 |
| 18 | Service Provider profile | 48 |
| 19 | Service Provider Registration | 49 |
| 20 | Service listing setting | 49 |
| 21 | User Service Requests | 50 |
| 21 | Chat Interface | 50 |
|  |  |  |

**LIST OF ABBREVIATIONS**

|  |  |  |
| --- | --- | --- |
| **No.** | **ABBRE.** | **DEFINATION** |
| 1 | SRS | Software Requirements Specification |
| 2 | UI | User Interface |
| 3 | QoS | Quality of Service |
| 4 | VPN | Virtual Private Network |
| 5 | SSL/TLS | Secure Sockets Layer/Transport Layer Security |
| 6 | HTTP | Hypertext Transfer Protocol |
| 7 | ER | Entity Relation |
| 8 | DFD | Data Flow Diagram |
| 9 | CRUD | Create, Read, Update, Delete |
| 10 | API | Application Programming Interface |

**CHAPTER 1**

**INTRODUCTION**

**1.1 INTRODUCTION**

Being in the 21st century with the rapid technological advancements and ever-increasing busy schedule of individuals, especially bachelors, often find themselves struggling to efficiently find and manage essential services to daily life. Tasks like maid, cook/chef, grocery shopping, and cleaning, once mundane, have become time-consuming challenges, often resulting in wasted hours, frustration, and subpar experiences[8]. The difficulties lie in the timeless scroll on google to find the right service provider[2]. process of finding reliable service providers, comparing prices, and coordinating bookings, leading to a pressing need for a solution that simplifies and streamlines this experience.

ServeU platform is presenting a comprehensive and user-friendly solution to the daily challenges faced by bachelors. By leveraging cutting-edge technologies, this research aims to develop a robust responsive web application that revolutionizes the way individuals access and manage essential services the most popular techniques is the responsive Web design. This application does not merely serve as a bridge for connecting users with different service providers as shown in Figure 2. it is a transformative tool designed to enhance efficiency, reliability, and convenience in the daily lives of bachelors.

From its foundational technologies to its user-friendly interface, the research will provide a detailed analysis of the application’s architecture, design principles, and implementation strategies. We will also examine the challenges faced by bachelors in accessing essential services and how the proposed application addresses these pain points. Moreover, this paper will shed light on the project’s impact on user experience, and Quality of service[9].

**1.2 PROJECT CATEGORY**

This project falls under the category of "Web Application Development". Specifically, it focuses on creating a user-friendly web platform to streamline essential services for busy bachelors.

1. **Service Marketplace Platform**: ServeU aims to create a platform where users can discover, book, and manage various essential services, such as maid, laundry, and cooking. It functions as a marketplace connecting service providers with users, which falls under the broader category of service marketplace platforms.
2. **Consumer Service Technology**: The project revolves around leveraging technology to enhance the consumer experience in accessing essential services. It involves the use of web development technologies, geolocation services, real-time messaging, and online payments to streamline service delivery, thereby contributing to advancements in consumer service technology.
3. **E-commerce and Online Transactions**: Given that ServeU facilitates online transactions for booking services, it aligns with the category of e-commerce platforms. It enables users to make payments securely through the platform, enhancing convenience and accessibility.
4. **Local Business Support**: The project contributes to supporting local businesses by providing them with a platform to offer their services to a wider audience. By connecting users with nearby service providers, ServeU promotes local entrepreneurship and economic growth.
5. **User Experience Enhancement**: A key aspect of the project is improving the user experience in accessing essential services. Through features such as service listings, price comparison, user reviews, and real-time chat, ServeU aims to enhance user satisfaction and convenience, contributing to advancements in user experience design.

**1.3 OBJECTIVES**

The project objectives for ServeU can be summarized as follows:

1. **Comprehensive Service Management Portal**: Develop a user-friendly web portal that allows users to efficiently discover, book, and manage all their daily service needs in one centralized platform.
2. **Establish Communication Bridge**: Implement a communication bridge between users and service providers to facilitate seamless interaction. Service providers receive request messages from users promptly, ensuring efficient booking and management of daily services.
3. **Skill and Locality Matching**: Ensure that service providers are matched with the right users based on their skills and locality, enhancing the relevance and effectiveness of service delivery.
4. **Service Listing**: Create a comprehensive catalog showcasing nearby essential service providers, including laundry, grocery shopping, meal services, and maid services. Provide detailed descriptions of pricing, ratings, user feedback, and service provider information to aid users in decision-making.
5. **Location-based Services**: Utilize geolocation services, such as Google Geolocation API, to help users find nearby service providers, enhancing accessibility and convenience.
6. **Price Comparison**: Enable users to compare prices from different service providers, empowering them to make informed decisions based on cost-effectiveness and service quality.
7. **Booking and Scheduling**: Implement a booking and scheduling system that allows users to book services at their preferred date and time. Manage appointments, such as laundry pickup and drop, efficiently to accommodate user schedules.
8. **User Review and Rating**: Enable users to leave reviews and ratings for service providers, facilitating transparency and accountability. User feedback helps other users make informed decisions and improves the overall quality of service provision.
9. **Chat and Messaging**: Implement a messaging system using Socket.io module to facilitate real-time communication between users and service providers. Establish a two-way communication channel for seamless interaction and quick resolution of queries or concerns.

By achieving these objectives, ServeU aims to revolutionize the way users access and manage essential services, providing convenience, efficiency, and satisfaction in their daily lives.

**1.4 PROBLEM FORMULATION**

In today’s fast-paced world, bachelors often struggle with efficiently managing essential services, such as maid, laundry, mess/tiffing services, and general daily services. The main challenges they face include finding reliable service providers, comparing prices, and coordinating bookings, leading to wasted time and subpar experiences. on the other side different types of daily service providers often find difficulty for getting users according to their skills due to communication gap between provider and receiver[1].

**1.5 PROPOSED SYSTEM**

ServeU is an innovative web application designed to simplify the process of discovering, booking, and managing essential services for busy individuals, particularly bachelors. At its core, ServeU aims to provide a comprehensive and user-friendly web portal where users can efficiently access a variety of daily services in one centralized platform. The primary objective of ServeU is to establish a seamless communication bridge between users and service providers, ensuring prompt response to user requests and efficient management of services. Service providers receive request messages promptly upon user initiation, and they are matched with the right users based on their skills and locality, enhancing the relevance and effectiveness of service delivery.

The platform includes features such as real-time chat and messaging, allowing users and service providers to communicate directly and resolve queries or concerns efficiently. Additionally, ServeU offers a comprehensive service listing with detailed information on pricing, ratings, and user feedback, enabling users to make informed decisions and compare service providers effectively. A feedback-based rating system further enhances the reliability of service providers and helps improve their skills over time.

User authentication is a crucial aspect of ServeU, ensuring the security and integrity of the platform for both users and service providers. Users can create accounts and manage their service requests through a personalized dashboard, while service providers have access to tools for managing their service listings, availability, and customer inquiries. Overall, ServeU aims to revolutionize the way users access and manage essential services, providing convenience, reliability, and satisfaction in their daily lives.

**1.6 UNIQUE FEATURES OF THE SYSTEM**

One unique feature of ServeU is its emphasis on facilitating direct communication between users and service providers through real-time chat and messaging functionality. This feature allows for seamless interaction, enabling users to communicate their needs, preferences, and inquiries directly to service providers. Likewise, service providers can promptly respond to user messages, address any concerns, and provide timely assistance. By prioritizing this direct communication channel, ServeU enhances the user experience, fosters transparency, and ensures efficient service delivery, setting it apart from other platforms in the market.

service provider location in the ServeU project lies in the utilization of geolocation technology to connect users with nearby service providers efficiently. This feature allows users to find service providers based on their current location, ensuring proximity and convenience. By integrating geolocation APIs, ServeU optimizes the user experience by offering services from providers in the same locality, enhancing accessibility and reducing response times. Additionally, this approach promotes local businesses and fosters community engagement by prioritizing service providers within the user's vicinity.

**CHAPTER 2**

**REQUIREMENT ANALYSIS AND SYSTEM SPECIFICATION**

**2.1 FEASIBILITY STUDY**

1. **Technical Feasibility:**
   * Software Development: ServeU requires the development of a robust web application using modern technologies such as React for frontend, Express for backend, and MongoDB Atlas for the database. The technical expertise to develop and maintain these technologies is readily available.
   * Real-time Communication: Implementing real-time chat and messaging functionality using Socket.io may require specialized technical skills, but it is feasible with the available libraries and frameworks.
   * Geolocation Integration: Utilizing geolocation services, such as Google Geolocation API, is technically feasible and can enhance the user experience by providing location-based services.
2. **Economic Feasibility:**
   * Cost of Development: While the initial development cost of ServeU may be significant due to the need for specialized technical expertise and infrastructure, the long-term operational costs can be relatively low, primarily consisting of maintenance and server hosting expenses.
   * Revenue Generation: ServeU can generate revenue through various monetization strategies, such as subscription models for service providers, transaction fees for bookings, and premium features for users. The potential revenue streams make the project economically viable.
3. **Operational Feasibility:**
   * User Adoption: ServeU aims to streamline the process of accessing essential services, addressing a genuine need in the market. With its user-friendly interface and convenient features, it is likely to attract a significant user base, enhancing its operational feasibility.
   * Service Provider Engagement: Engaging service providers to join the platform may require marketing efforts and incentives initially. However, the potential for increased visibility, customer reach, and revenue generation may incentivize service providers to participate actively.
   * Scalability: ServeU is designed to be scalable, allowing for future expansion and adaptation to evolving user needs and market trends. This scalability ensures its long-term operational viability and relevance in the dynamic service industry.

In conclusion, ServeU demonstrates strong technical feasibility, supported by the availability of modern development tools and expertise. Economically, the project is viable, with potential revenue streams and manageable operational costs. Operationally, ServeU is well-positioned to attract users and engage service providers, with scalability to accommodate growth and changes in the service landscape. Overall, the feasibility study indicates that ServeU has the potential to be a successful and sustainable venture in the realm of essential service management.

**2.2 SOFTWARE REQUIREMENT SPECIFICATION:**

**1. Introduction**

**1.1 Purpose**

The purpose of this document is to outline the requirements for ServeU, a user-friendly web application designed to streamline the process of discovering, booking, and managing essential services such as maid, laundry, mess/tiffin services, and general daily services. It aims to address the challenges faced by bachelors in efficiently accessing and managing these services, as well as bridging the communication gap between service providers and users.

**1.2 Scope**

ServeU will provide a comprehensive platform for users to:

* Discover and book essential services.
* Communicate with service providers in real-time.
* Manage service requests and feedback.
* Facilitate online transactions for services.

**1.3 Definitions, Acronyms, and Abbreviations**

SRS: Software Requirements Specification

UI: User Interface

**1.4 References**

ServeU Website:

ServeU GitHub Repository:

**2. Overall Description**

**2.1 Product Perspective**

ServeU operates as a standalone web application accessible through modern web browsers. It serves as a platform connecting users with service providers, facilitating communication and transactions.

**2.2 Product Functions**

* User Registration and Authentication
* Service Discovery and Booking
* Real-time Communication
* Service Management
* Online Transactions
* Feedback and Rating System

**2.3 User Classes and Characteristics**

* Users: Individuals seeking essential services, primarily bachelors.
* Service Providers: Individuals or businesses offering essential services.

**2.4 Operating Environment**

ServeU is a web-based application accessible on various devices with internet connectivity. It utilizes responsive design principles to ensure compatibility with desktop and mobile browsers.

**2.5 Design and Implementation Constraints**

ServeU is developed using React for the frontend, Express for the backend, and MongoDB Atlas for the database. It follows a modular design approach for scalability and maintainability.

**3. External Interface Requirements**

**3.1 User Interfaces**

**1 Landing Page**

* **Description**: The landing page serves as the initial interface users encounter when accessing ServeU. Its design aims to capture users' attention and encourage them to explore the platform further.
* **Components**:
  + **Header**: Includes the ServeU logo and navigation menu, providing links to essential sections like service listings, user profiles, and support.
  + **Hero Section**: Features captivating visuals and brief descriptions highlighting the platform's key features and benefits, such as convenience, reliability, and user-friendliness.
  + **Call-to-Action Buttons**: Prominently placed buttons encourage users to sign up for an account or log in to access the platform's services.
  + **Service Categories**: Offers a visually appealing display of popular service categories (e.g., maid, laundry, cooking), allowing users to quickly navigate to their desired section.

**2 Service Listing Page**

* **Description**: The service listing page presents users with a comprehensive list of available service providers based on their search criteria and preferences.
* **Components**:
  + **Search Bar**: Allows users to search for specific services or service providers by entering keywords or selecting predefined categories.
  + **Filter Options**: Provides advanced filtering capabilities, enabling users to refine search results based on parameters such as service type, location, price range, and ratings.
  + **Service Provider Cards**: Each card represents a service provider and displays essential information such as name, location, service offerings, ratings, and reviews.
  + **Booking Button**: Directs users to initiate the booking process for a selected service provider, prompting them to provide necessary details such as date, time, and specific requirements.

**3 Service Provider Profile Page**

* **Description**: The service provider profile page offers users detailed insights into individual service providers, helping them make informed decisions.
* **Components**:
  + **Profile Picture**: Displays an image representing the service provider, fostering familiarity and trust.
  + **Contact Information**: Provides essential contact details such as phone number, email address, and physical location, facilitating direct communication.
  + **Service Offerings**: Presents a comprehensive list of services offered by the provider, accompanied by detailed descriptions, pricing information, and any special offers.
  + **Ratings and Reviews**: Showcases aggregated ratings and reviews from previous users, offering valuable feedback on the provider's performance, reliability, and customer satisfaction.
  + **Chat Interface**: Enables users to engage in real-time communication with the service provider, facilitating inquiries, bookings, and clarifications.

**4 User Dashboard**

* **Description**: The user dashboard serves as a centralized hub for users to manage their account settings, bookings, and interactions with service providers.
* **Components**:
  + **Account Settings**: Allows users to view and update their profile information, including name, contact details, and password.
  + **Booking History**: Displays a chronological record of past and upcoming service bookings, along with relevant details such as service provider, date, time, and status.
  + **Feedback and Ratings**: Provides users with the opportunity to submit feedback and ratings for services they have received, contributing to the platform's reputation and service quality.
  + **Chat Interface**: Facilitates seamless communication with service providers, displaying ongoing conversations and notifications for new messages.

**5 Service Provider Dashboard**

* **Description**: The service provider dashboard equips service providers with tools and features to manage their services, bookings, and interactions with users effectively.
* **Components**:
  + **Service Listings**: Enables providers to view, add, update, or remove their service offerings, ensuring accurate and up-to-date information for users.
  + **Booking Requests**: Presents incoming service requests from users, allowing providers to accept or reject requests based on availability and preferences.
  + **Chat Interface**: Provides a dedicated platform for real-time communication with users, facilitating prompt responses to inquiries, booking confirmations, and service-related discussions.
  + **Feedback and Ratings**: Offers insights into user feedback and ratings, enabling providers to evaluate their performance, address any concerns, and maintain a positive reputation on the platform.

**3.2 Hardware Interfaces**

1. **Server:**
   * The server infrastructure should be robust and scalable to handle the incoming requests from clients.
   * It should be equipped with sufficient processing power, memory, and storage capacity to ensure smooth operation even during peak usage hours.
   * Redundancy measures such as load balancing and failover systems should be in place to ensure high availability and reliability.
   * Security mechanisms like firewalls and intrusion detection systems should be implemented to protect the server from unauthorized access and cyber threats.
2. **Storage:**
   * The storage solution should be capable of handling large volumes of data generated by the application.
   * It should provide high-speed access to data to minimize latency and ensure optimal performance.
   * Data redundancy and backup mechanisms should be implemented to prevent data loss in case of hardware failures or other disasters.
   * Security measures such as encryption and access controls should be in place to protect sensitive data stored on the storage devices.
3. **Network:**
   * The network infrastructure should provide high-speed and reliable connectivity between clients and the server.
   * It should be scalable to accommodate increasing traffic as the user base grows.
   * Quality of Service (QoS) mechanisms should be implemented to prioritize critical traffic and ensure low latency for real-time communication.
   * Security protocols such as VPNs and SSL/TLS encryption should be used to secure data transmitted over the network and protect against eavesdropping and unauthorized access.

**3.3 Software Interfaces**

ServeU integrates with external payment gateways for processing transactions and utilizes HTTP protocols for communication between clients and servers.

**3.4 Communications Interfaces**

ServeU enables real-time communication between users and service providers through a chat interface.

1. **User-Provider Interaction:**
   * Facilitates seamless communication between users and service providers.
   * Allows users to send queries, requests, and messages to service providers.
   * Enables service providers to respond to user queries, accept or reject service requests, and provide assistance.
2. **Real-Time Messaging:**
   * Supports real-time messaging for immediate interaction.
   * Users can instantly communicate their requirements, preferences, and feedback to service providers.
   * Service providers can promptly respond to user inquiries, address concerns, and confirm service requests.
3. **Documentation and History:**
   * Maintains a record of communication history between users and service providers.
   * Allows users and providers to access past conversations, service requests, and interactions for reference and documentation purposes.
4. **Feedback Mechanism:**
   * Integrates a feedback mechanism to gather user ratings, reviews, and opinions on service quality and provider performance.
   * Enables users to provide feedback after service completion, helping to improve service standards and provider reliability.

**4. System Features**

**4.1 User Registration and Authentication**

* **Description:** Users can register for an account by providing necessary information such as name, email, and password. Authentication mechanisms ensure secure access to user accounts.
* **Inputs:** User-provided registration details (name, email, password).
* **Outputs:** Confirmation of successful registration, authentication tokens for logged-in users.
* **Dependencies:** Integration with backend server for user account management.

**4.2 Service Discovery and Booking**

* **Description:** Users can browse and search for available services based on their preferences and location. They can view service provider profiles, compare prices, and book services.
* **Inputs:** User search queries, location data, service preferences.
* **Outputs:** List of available services, service provider profiles, booking confirmations.
* **Dependencies:** Integration with backend server for retrieving service data and handling bookings.

**4.3 Real-time Communication**

* **Description:** Users can communicate with service providers in real-time through a chat interface. This enables quick inquiries, scheduling adjustments, and clarifications.
* **Inputs:** User messages, service provider responses.
* **Outputs**: Real-time chat messages, notifications for new messages.
* **Dependencies**: Integration with backend server for handling chat messages and notifications.

**4.4 Service Management**

* **Description:** Service providers can manage their service offerings, including adding, updating, or deleting services. They can also view and respond to service requests from users.
* **Inputs:** Service provider updates to service listings, responses to service requests.
* **Outputs:** Updated service listings, notifications for new service requests.
* **Dependencies:** Integration with backend server for managing service data and requests.

**4.5 Online Transactions**

* **Description:** Users can make online payments for booked services through secure payment gateways. Upon successful payment, users receive confirmation of the transaction.
* **Inputs:** User payment details, service booking confirmations.
* **Outputs:** Payment confirmation, service booking details.
* **Dependencies:** Integration with external payment gateways for processing transactions securely.

**4.6 Feedback and Rating System**

* **Description:** Users can provide feedback and ratings for services they have received. This helps improve service quality and reliability by providing valuable insights to both users and service providers.
* **Inputs:** User feedback, ratings for service providers.
* **Outputs:** Aggregated service ratings, feedback summaries.
* **Dependencies:** Integration with backend server for collecting and displaying feedback data.

**5. Other Nonfunctional Requirements**

**5.1 Performance Requirements**

* **Response Time:** The system should respond to user actions within a reasonable timeframe (e.g., less than 3 seconds).
* **Scalability:** The system should be able to handle increasing numbers of users and service providers without significant degradation in performance.

**5.2 Safety Requirements**

**Data Encryption:** User data and payment information should be encrypted to ensure confidentiality and prevent unauthorized access.

**Transaction Security:** Online transactions should be conducted securely using industry-standard encryption protocols.

**5.3 Security Requirements**

**User Authentication:** Strong authentication mechanisms should be implemented to verify the identity of users and prevent unauthorized access.

**Data Protection:** Measures should be in place to protect user data from unauthorized access, manipulation, or theft.

**5.4 Software Quality Attributes**

**Reliability:** The system should be reliable and available for use at all times, with minimal downtime for maintenance or updates.

**Usability:** The user interface should be intuitive and easy to navigate, catering to users with varying levels of technical proficiency.

**5.5 Business Rules**

**Service Availability:** Service providers should specify their availability for bookings, and users should be able to view available time slots when booking services.

**Cancellation Policy:** Clear policies should be in place regarding service cancellations, refunds, and rescheduling.

**6. Other Requirements**

**6.1 Integration with External Systems**

**Payment Gateways:** Integration with external payment gateways for processing online transactions securely.

**Location Services:** Integration with location services to provide accurate service recommendations based on user location.

**6.2 Accessibility**

Accessibility Standards: The system should comply with accessibility standards to ensure that users with disabilities can access and use the platform effectively.

**6.3 Internationalization**

Multilingual Support: The system should support multiple languages to cater to users from diverse linguistic backgrounds.

**6.4 User Training and Support**

User Guides: Provide user guides and tutorials to help users navigate the platform and utilize its features effectively.

Customer Support: Offer customer support channels for addressing user queries, issues, and feedback.

**6.5 Continuous Improvement**

Feedback Mechanisms: Implement mechanisms for collecting user feedback and suggestions for improving the platform.

Regular Updates: Release regular updates and enhancements to address user needs and improve system performance.

**2.3 SDLC MODEL TO BE USED**

Agile SDLC Model Given the dynamic nature of the project and the need for frequent feedback and adaptation, Agile is well-suited to ensure the success of ServeU. It enables the development team to respond effectively to changing requirements, deliver valuable features iteratively, and ultimately, meet the needs of users and service providers efficiently.

**CHAPTER 3**

**SYSTEM DESIGN**

**3.1 DETAIL DESIGN**

**3.1.1 SYSTEM MODELS**

A User Version

Basically, user who want avail the service needs to create the account by entering name, email, and password which will store in the database for the future. After the login process, he redirects to the user dashboard where he can search new service and manage his old services and feedback. If the user wants to search new service, he has to search the service name and then choose the service provider from the list where he can raise the query or message to the service provider. After the query raise system will create a request on the service provider dashboard where he can accept or reject the request according to his availability. User can also give feedback to the service provider.

A Service Provider Version

In the service provider dashboard, he gets lots of functionality like he can see status of all his user request where he can accept or reject his requests. He can reply to the user query from the chat functionality. In addition to this provider has the feature to maintain his service list like he can add, update, or delete his services. Service provider get the message whenever any new customer raises the service request and can read the feedback of user. he can update his user profile whenever he wants.

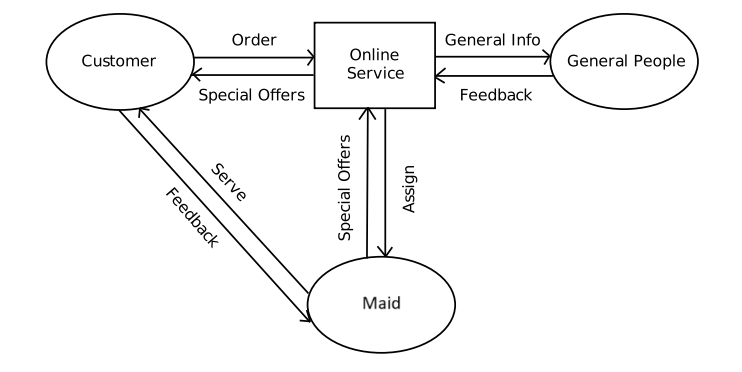


Figure 1: Model Connection

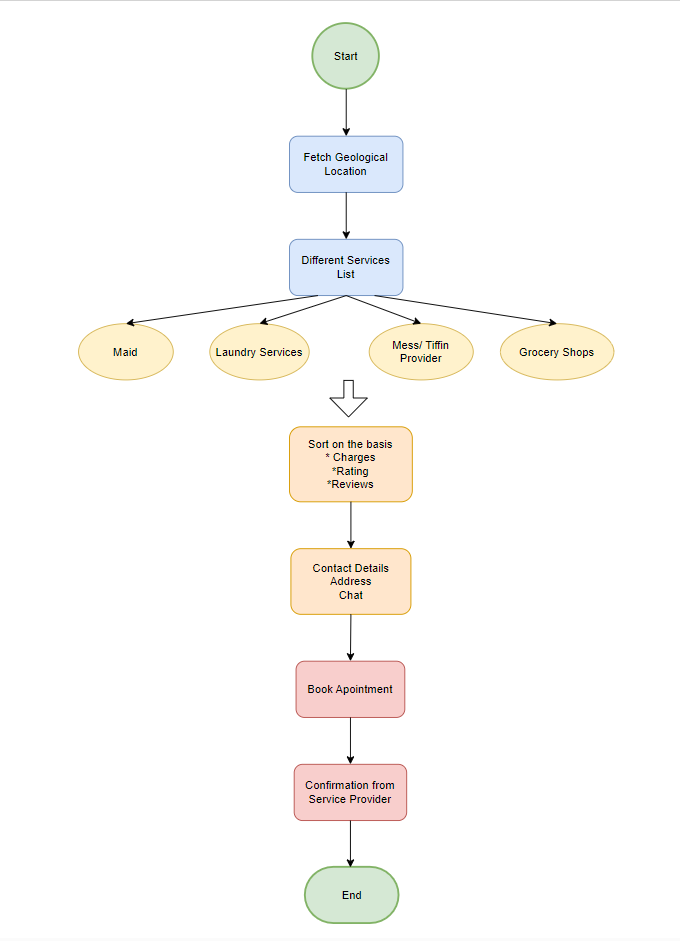
****

Figure 2: Flow Diagram

**3.1.2 SYSTEM TOOLS**

A. React (Frontend)

In this project we have used the react library for the frontend work. It helps us to divide our project in small modules which is easy to manage. Along with this we have used tailwind CSS for styling[7]. This helps us to build the responsive[6] user interfaces like authentication pages, dashboard pages, service listing pages etc. We have integrated these pages with the main app and make a single page web application.

B. Express (Backend)

It helps us to create a express server which will handle all the backend tasks like storing all the authentication data to database. It connects the right user with the right service provider. And also handle the chat feature.

C. MongoDB Atlas (Database)

In atlas we have added all the user data, provider data, service data, and chat data.

**3.1.3. SYSTEM MODULES**

A. Authentication Module

This will handle all the authentication tasks like signup and login. If the new user comes it will create new user and after taking the name, phone no, email, and password. And if existing user comes it will login the user to his dashboard.

B. User Module

Here user can manage all his service request and his profile. He can also give feedback to his previous service provider. From his dashboard he can chat will the service provider directly or raise the service request whenever he require[5].

C. Service Provider Module

Service Provider can see status of all his user request where he can accept or reject his requests. He can reply to the user query from the chat functionality. In addition to this provider has the feature to maintain his service list like he can add, update, or delete his services. Service provider get the message whenever any new customer raises the service request and can read the feedback of user. he can update his user profile whenever he wants.

D. Service Listing Module

This will list down all the nearby service provider according to the user need. Where user can compare prices, read user reviews and chat with the provider.

E. Chat Module

This enables us to establish two-way communication channel between the user and service provider.

F. Payment Module

Through this module customer can pay for the services opted. It is done through an external payment gateway which guarantees a secure and safe transaction. Once the payment is done, a confirmation acknowledgement is forwarded to the user about all the details of services opted and also an onsite confirmation is displayed on the website.

**3.2 SYSTEM DESIGN USING DFD LEVEL 0 AND LEVEL 1**

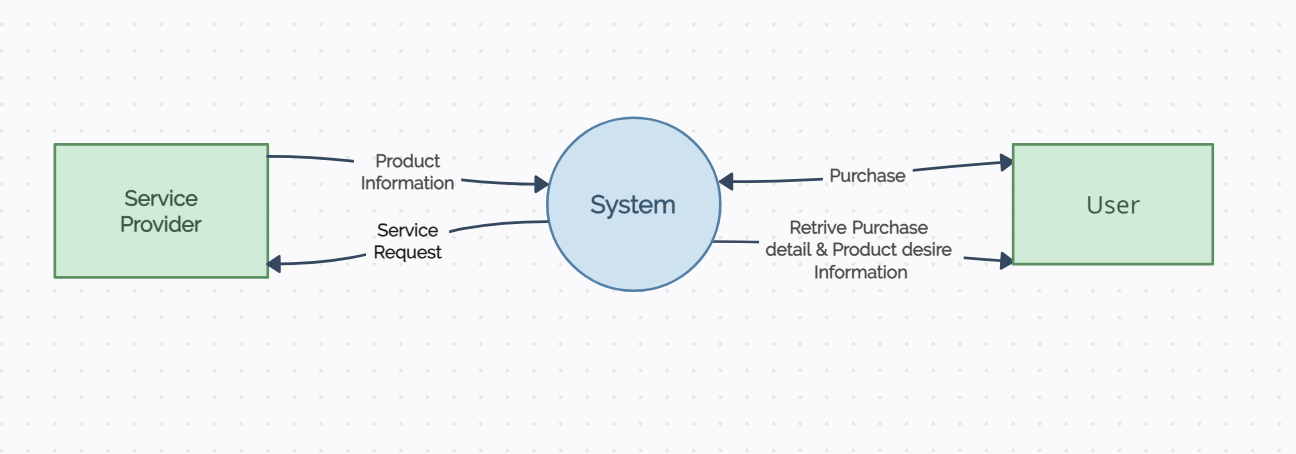
****

Figure 3: 0-Level DFD

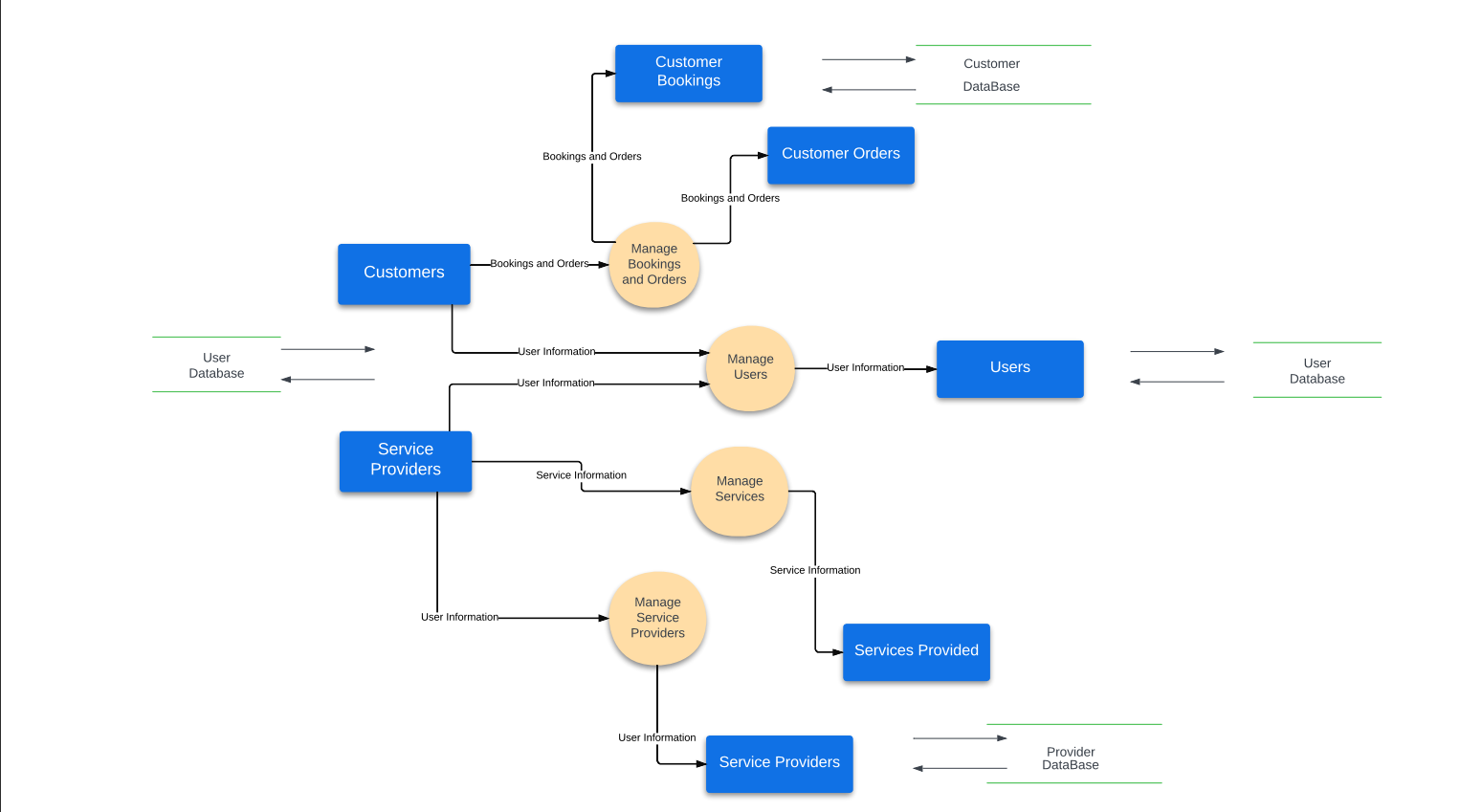
****

Figure 4: 1-level DFD

**3.3 USE CASE DIAGRAM**

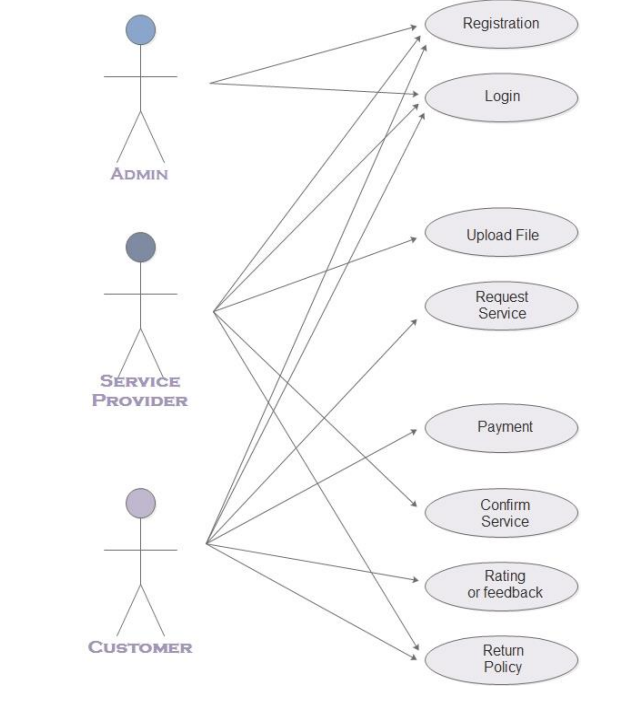


Figure 5: Use Case Diagram

**3.4 DATABASE DESIGN**

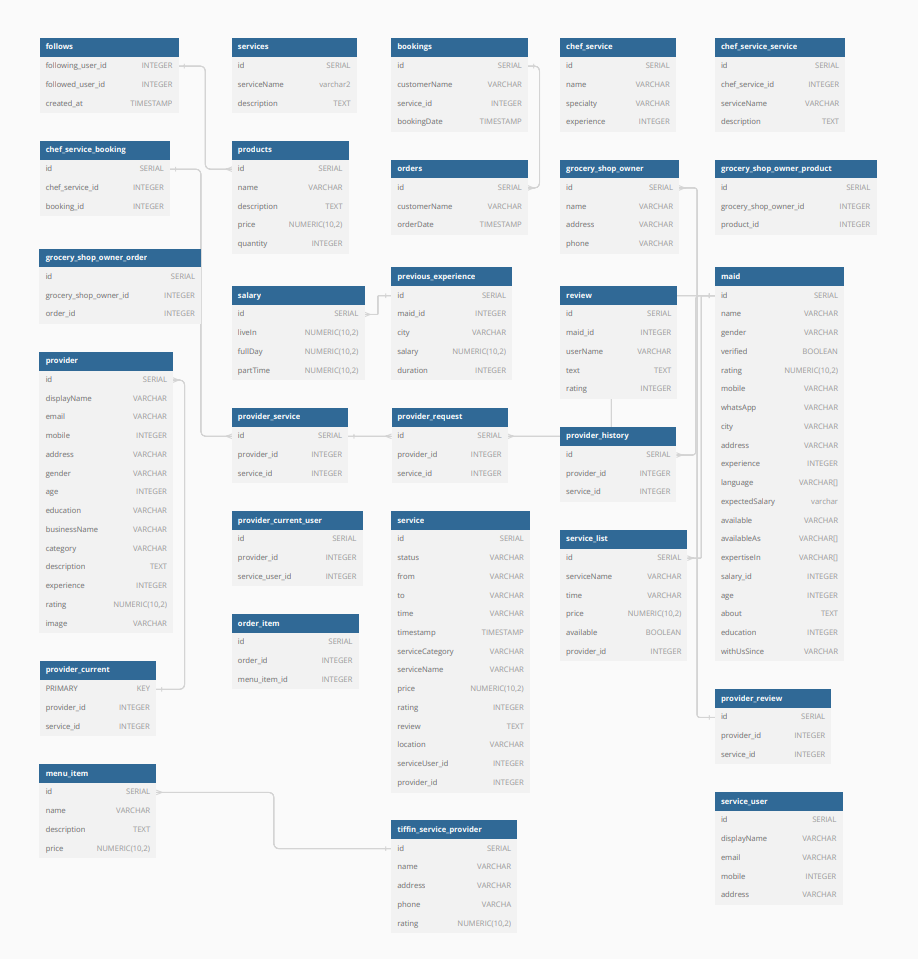
****

Figure 6: Database Table

// Schema for Services Provided

const ServiceSchema = new Schema({

    serviceName: { type: String, required: true },

    description: { type: String, required: true },

});

// Schema for Customer Bookings

const BookingSchema = new Schema({

    customerName: { type: String, required: true },

    service: { type: Schema.Types.ObjectId, ref: 'Service' },

    bookingDate: { type: Date, default: Date.now },

});

// Schema for Cook/Chef Service Provider

const ChefServiceSchema = new Schema({

    name: { type: String, required: true },

    specialty: { type: String, required: true },

    experience: { type: Number, required: true },

    services: [ServiceSchema], // Embedded Service Schema

    bookings: [BookingSchema], // Embedded Booking Schema

});

// Schema for Products

const ProductSchema = new Schema({

    name: { type: String, required: true },

    description: { type: String, required: true },

    price: { type: Number, required: true },

    quantity: { type: Number, required: true },

});

// Schema for Customer Orders

const OrderSchema = new Schema({

    customerName: { type: String, required: true },

    items: [{ type: Schema.Types.ObjectId, ref: 'Product' }],

    orderDate: { type: Date, default: Date.now },

});

// Schema for Grocery Shop Owner

const GroceryShopOwnerSchema = new Schema({

    name: { type: String, required: true },

    address: { type: String, required: true },

    phone: { type: String, required: true },

    products: [ProductSchema], // Embedded Product Schema

    orders: [OrderSchema], // Embedded Order Schema

});

const salarySchema = new mongoose.Schema({

  liveIn: [Number],

  fullDay: [Number],

  partTime: [Number]

})

const previousExperienceSchema = new mongoose.Schema({

  city: String,

  salary: Number,

  duration : [Number]

})

const reviewSchema = new mongoose.Schema({

  userName: String,

  text: String,

  rating: Number

})

const maidSchema = new mongoose.Schema({

    name: String,

    gender: String,

    verified: Boolean,

    rating: Number,

    mobile: String,

    whatsApp: String,

    city: String,

    address: String,

    experience: Number,

    language: [String],

    expectedSalary: [Number],

    available: String,

    availableAs: [String],

    expertiseIn: [String],

    salary: salarySchema,

    age: Number,

    about: String,

    education: Number,

    withUsSince: String,

    previousExperience: [previousExperienceSchema],

    reviews: [reviewSchema]

})

const providerSchema = new mongoose.Schema({

    displayName: String,

    email: String,

    mobile: Number,

    address: String,

    gender: String,

    age: Number,

    education: String,

    businessName: String,

    category: String,

    description: String,

    experience: Number,

    rating: Number,

    image:String,

    provides : [{ type: mongoose.Schema.Types.ObjectId, ref: 'ServiceList' }],

    requests: [{ type: mongoose.Schema.Types.ObjectId, ref: 'Service' }], // Reference to Service

    history: [{ type: mongoose.Schema.Types.ObjectId, ref: 'Service' }], // Reference to Service

    reviews: [{ type: mongoose.Schema.Types.ObjectId, ref: 'Service' }], // Reference to Service

    current: [{ type: mongoose.Schema.Types.ObjectId, ref: 'Service' }], // Reference to Service

    currentUser: [{ type: mongoose.Schema.Types.ObjectId, ref: 'ServiceUser' }] // Reference to ServiceUser

});

const serviceSchema = new mongoose.Schema({

    status: String,

    from: String,

    to: String,

    time: String,

    timestamp : Date,

    serviceCategory: String,

    serviceName: String,

    price: Number,

    rating: Number,

    review: String,

    location: String,

    serviceUser: { type: mongoose.Schema.Types.ObjectId, ref: 'ServiceUser' }, // Reference to ServiceUser

    provider: { type: mongoose.Schema.Types.ObjectId, ref: 'Provider' } // Reference to Provider

});

const serviceListSchema = new mongoose.Schema({

    serviceName : String,

    time : String,

    price : Number,

    available : Boolean,

    provider :  {

        type: mongoose.Schema.Types.ObjectId,

        ref: 'Provider'

      }

})

const serviceUserSchema = new mongoose.Schema({

    displayName: String,

    email: String,

    mobile: Number,

    address: String,

    applied: [{ type: mongoose.Schema.Types.ObjectId, ref: 'Service' }], // Reference to Service

    current: [{ type: mongoose.Schema.Types.ObjectId, ref: 'Service' }], // Reference to Service

    history: [{ type: mongoose.Schema.Types.ObjectId, ref: 'Service' }] // Reference to Service

});

const MenuItemSchema = new Schema({

    name: { type: String, required: true },

    description: { type: String, required: true },

    price: { type: Number, required: true },

// Schema for Customer Orders

const OrderSchema = new Schema({

    customerName: { type: String, required: true },

    items: [{ type: Schema.Types.ObjectId, ref: 'MenuItem' }],

    orderDate: { type: Date, default: Date.now },

});

// Schema for Tiffin Service Provider

const TiffinServiceProviderSchema = new Schema({

    name: { type: String, required: true },

    address: { type: String, required: true },

    phone: { type: String, required: true },

    menu: [MenuItemSchema], // Embedded Menu Schema

    orders: [OrderSchema], // Embedded Order Schema

    rating: Number

});

const userSchema = new mongoose.Schema({

  passwordHash: String,

  type: String,

  googleId:String,

  displayName:String,

  email:String,

  image:String

},{timestamps: true});

**3.4.1 ER DIAGRAM**

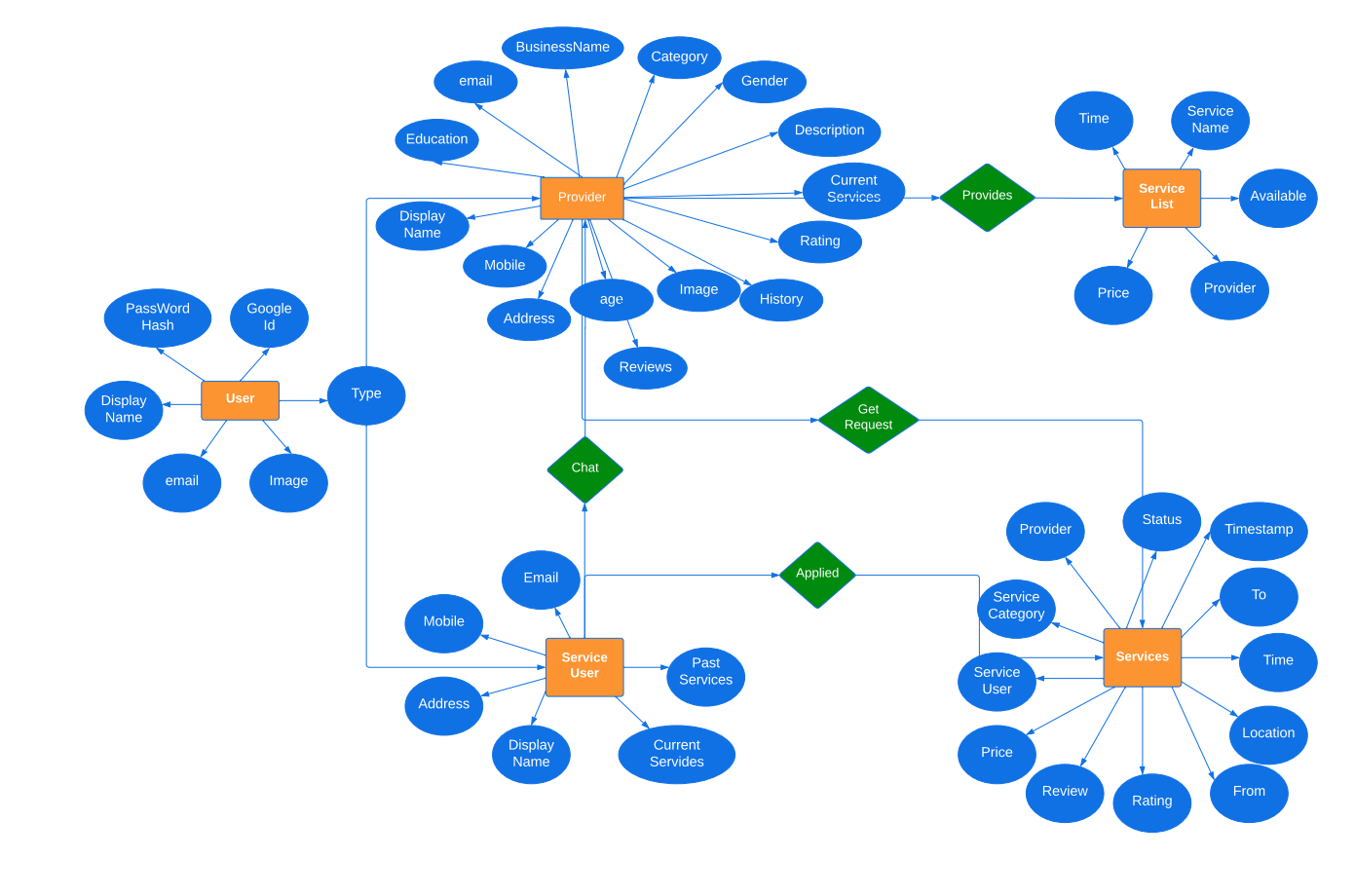
****

Figure 7: Entity Relation Diagram

**3.5 Class Diagram**

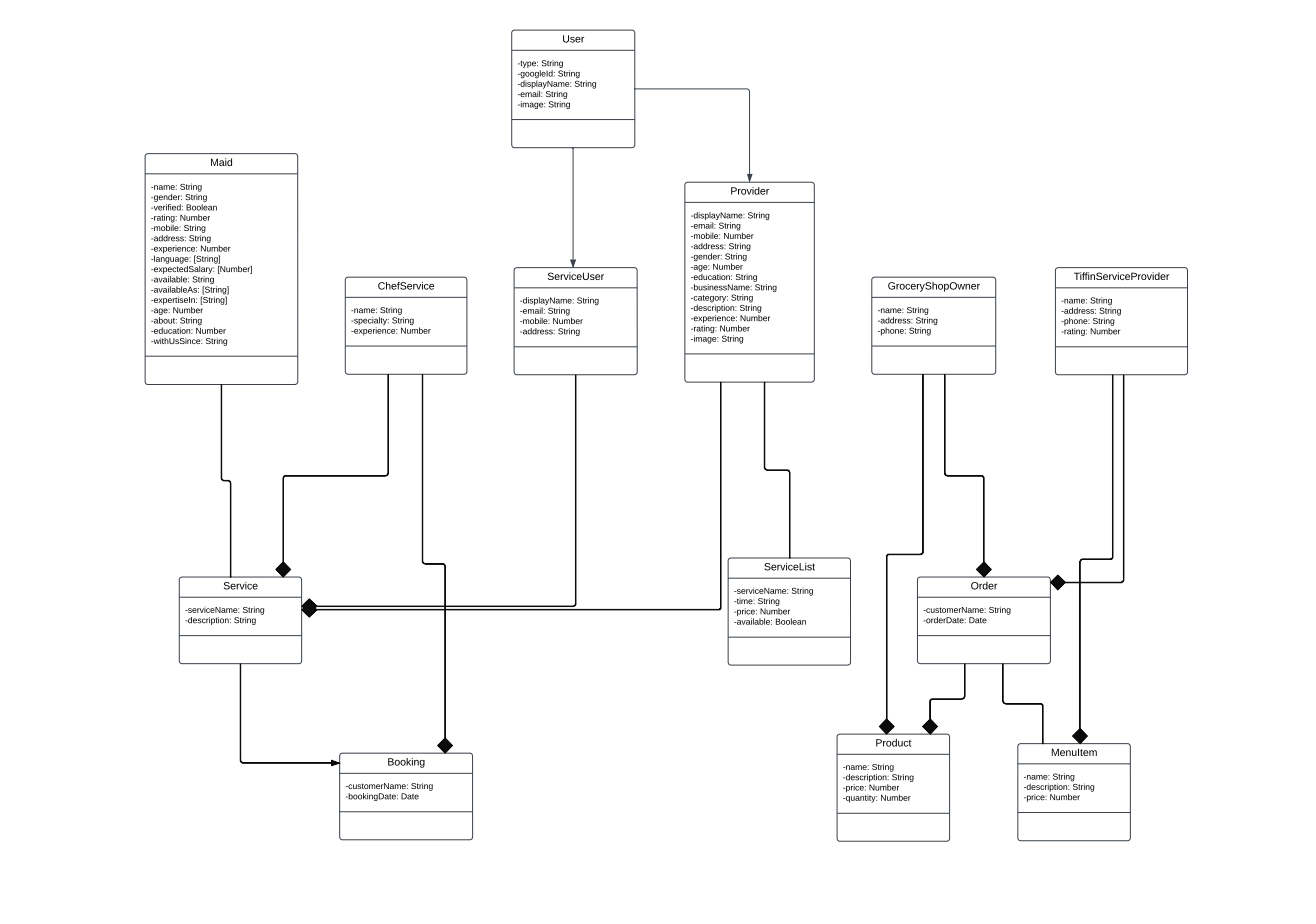
****

Figure 8: Class Diagram

**3.6 Architectural Diagram**

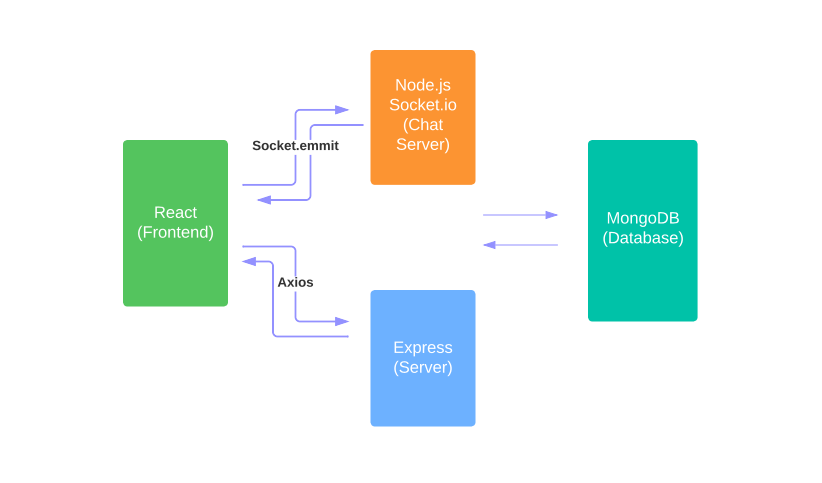
****

Figure 9: Architecture Diagram

**3.7 Activity Diagram:**

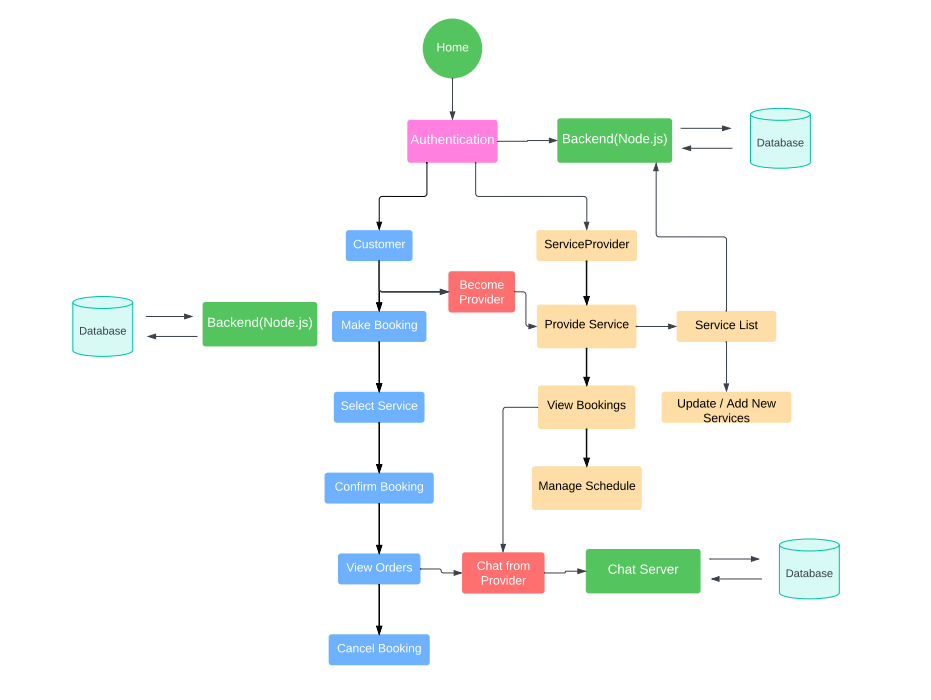
****

Figure 10: Activity Diagram

**CHAPTER 4**

**IMPLEMENTATION, TESTING, AND MAINTENANCE**

**4.1 INTRODUCTION TO LANGUAGES, TOOLS AND TECHNOLOGIES USED FOR IMPLEMENTATION**

4.1.1 React (Frontend)

In this project we have used the react library for the frontend work. It helps us to divide our project in small modules which is easy to manage. Along with this we have used tailwind CSS for styling[7]. This helps us to build the responsive[6] user interfaces like authentication pages, dashboard pages, service listing pages etc. We have integrated these pages with the main app and make a single page web application.

4.1.2 Express (Backend)

It helps us to create a express server which will handle all the backend tasks like storing all the authentication data to database. It connects the right user with the right service provider. And also handle the chat feature.

4.1.3 MongoDB Atlas (Database)

In atlas we have added all the user data, provider data, service data, and chat data.

MongoDB Atlas is a fully managed cloud database service provided by MongoDB, offering a secure and scalable platform for storing, querying, and analyzing data. It enables users to deploy, manage, and scale MongoDB databases with ease, providing features such as automated backups, monitoring, and fine-grained access control.

4.1.4 NodeJS (Backend Running Environment)

Node.js is an open-source, cross-platform JavaScript runtime environment that executes JavaScript code outside of a web browser. It’s designed to build scalable network applications and is known for its event-driven, non-blocking I/O model, which makes it lightweight and efficient. Node.js runs on the V8 JavaScript engine, allowing developers to use JavaScript for server-side scripting and to write command line tools.

**4.2 TESTING TECHNIQUES AND TEST CASES USED**

1. Unit Testing:
   * Jest: A JavaScript testing framework with a focus on simplicity.
2. Integration Testing:
   * Postman: An API testing tool for testing web services. It allows for automated testing of API endpoints and data validation.
3. End-to-End Testing:
   * Cypress: A modern JavaScript testing framework for end-to-end testing of web applications. It provides an easy-to-use API and supports both unit and integration testing.
4. Browser Testing:
   * BrowserStack: A cloud-based platform for testing web applications across different browsers and devices. It allows for cross-browser compatibility testing without the need for setting up multiple virtual machines.

**CHAPTER 5**

**RESULTS AND DISCUSSIONS**

**5.1 USER INTERFACE REPRESENTATION**



Figure 11: Landing Page

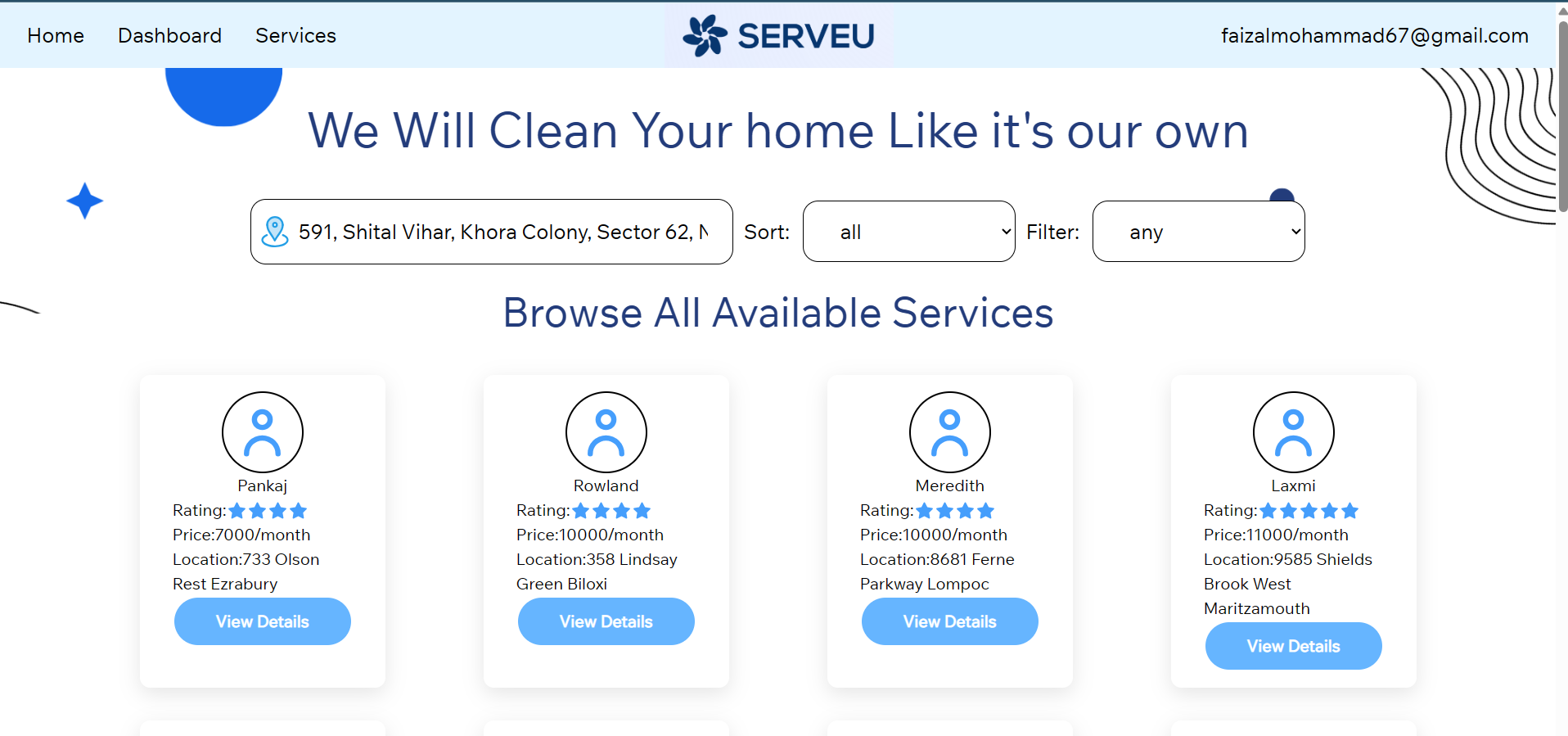


Figure 12: Service Category Page



Figure 13: Service Request Page

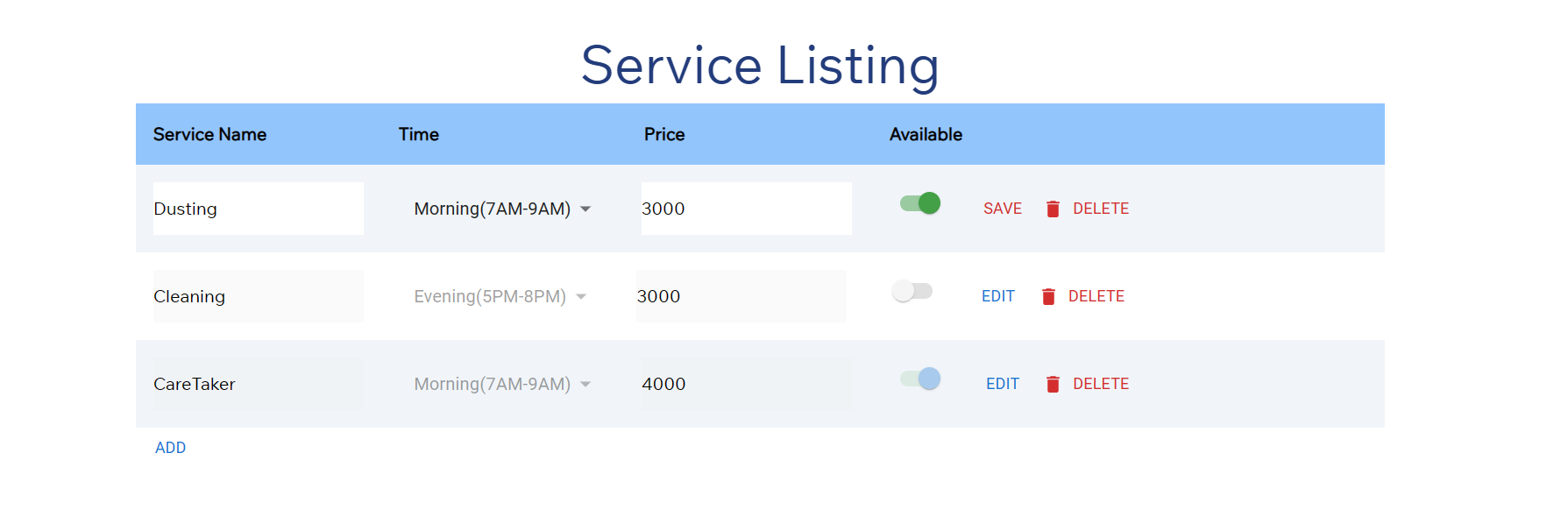


Figure 14: Service Listing Page

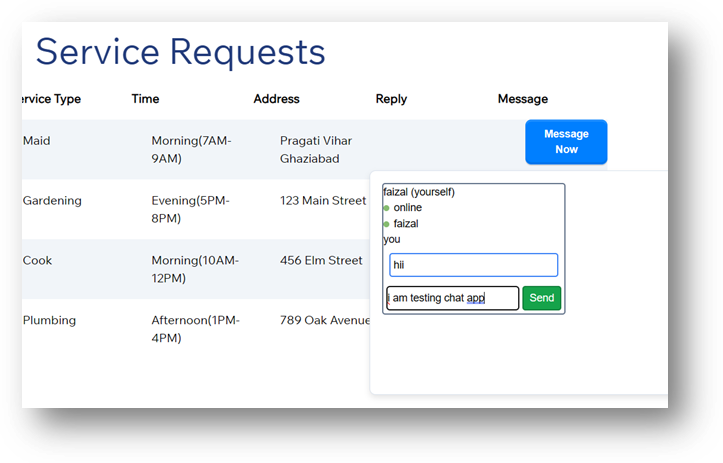


Figure 15: Chat Component

**5.1.1 BRIEF DESCRIPTION OF VARIOUS MODULES OF THE SYSTEM**

A. Authentication Module

This will handle all the authentication tasks like signup and login. If the new user comes it will create new user and after taking the name, phone no, email, and password. And if existing user comes it will login the user to his dashboard.

B. User Module

Here user can manage all his service request and his profile. He can also give feedback to his previous service provider. From his dashboard he can chat will the service provider directly or raise the service request whenever he require[5].

C. Service Provider Module

Service Provider can see status of all his user request where he can accept or reject his requests. He can reply to the user query from the chat functionality. In addition to this provider has the feature to maintain his service list like he can add, update, or delete his services. Service provider get the message whenever any new customer raises the service request and can read the feedback of user. he can update his user profile whenever he wants.

D. Service Listing Module

This will list down all the nearby service provider according to the user need. Where user can compare prices, read user reviews and chat with the provider.

E. Chat Module

This enables us to establish two-way communication channel between the user and service provider.

F. Payment Module

Through this module customer can pay for the services opted. It is done through an external payment gateway which guarantees a secure and safe transaction. Once the payment is done, a confirmation acknowledgement is forwarded to the user about all the details of services opted and also an onsite confirmation is displayed on the website.

**5.2 SNAPSHOTS OF SYSTEM WITH BRIEF DETAIL OF EACH**

As a part of our study, we have implemented this application by running on local machine. Most of the original design is based on intuition. we here implemented our primary design followed by the user’s primary household problems. Our primary locations are Mohan Nagar and Murad Nagar and primary service orders are maid, mess and laundry etc.

A. User

When user has created account and login to the website. Figure shows the how landing page shows.

When user search any service, he will get list of all the service provider in his nearby location. Figure shows how the listing page will be shown. if user want to see the provider profile he can see as shown in Figure .



Figure 16: Landing Page

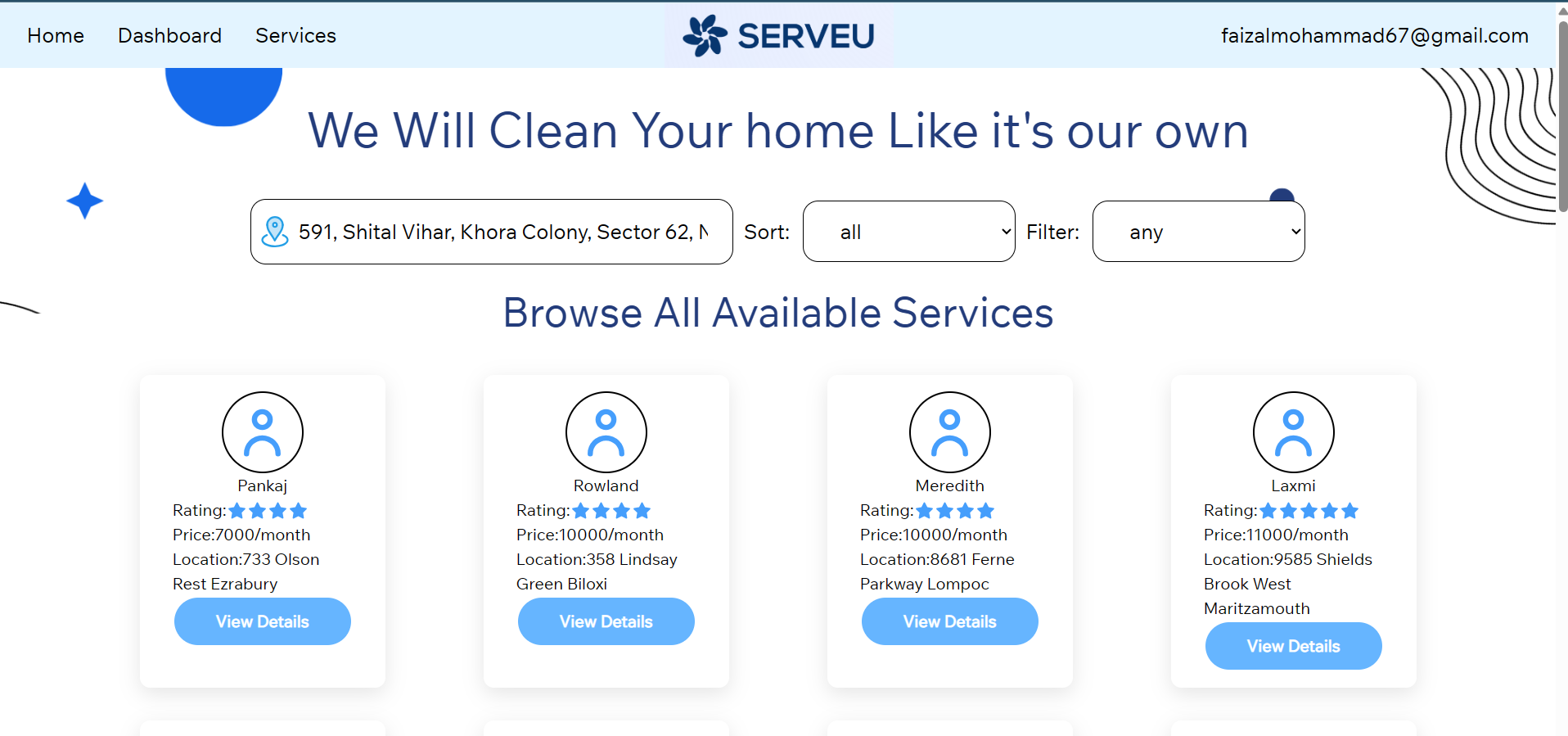


Figure 17: Service Listing page

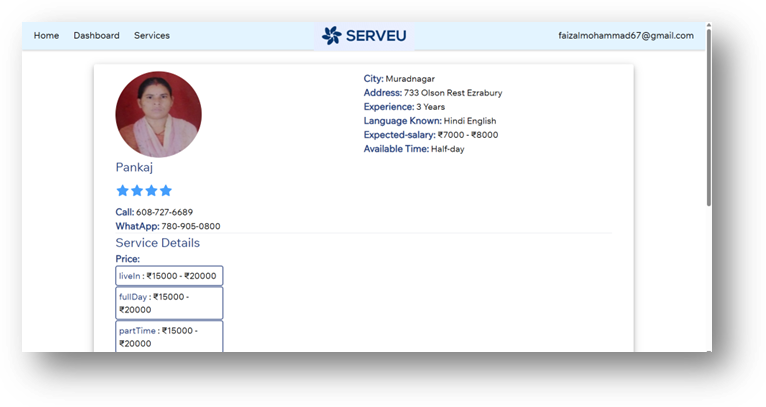


Figure 18: Service Provider profile

B. Service Provider

If the user want to become a service provider he can become after the service registration form as shown in Figure 9 after that he can add different services as Shown in Figure 10 whenever user send request to service provider new request adds into the service requests table as shown in Figure 12 where he can chat with the user with the help of chat popup as shown in Figure 13.

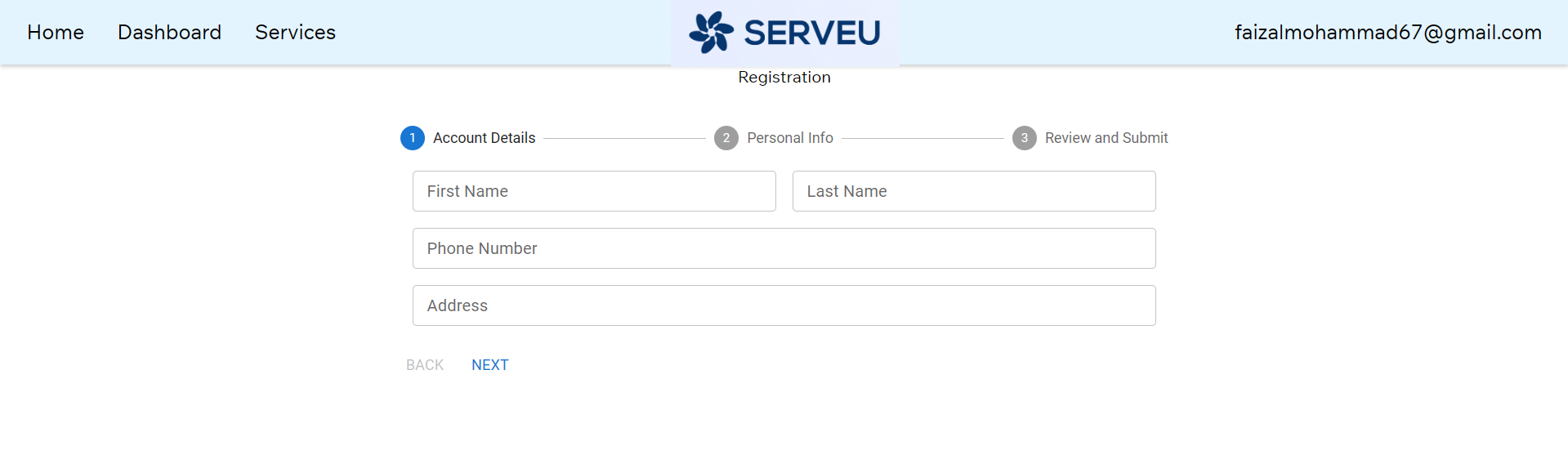


Figure 19: Service Provider Registration

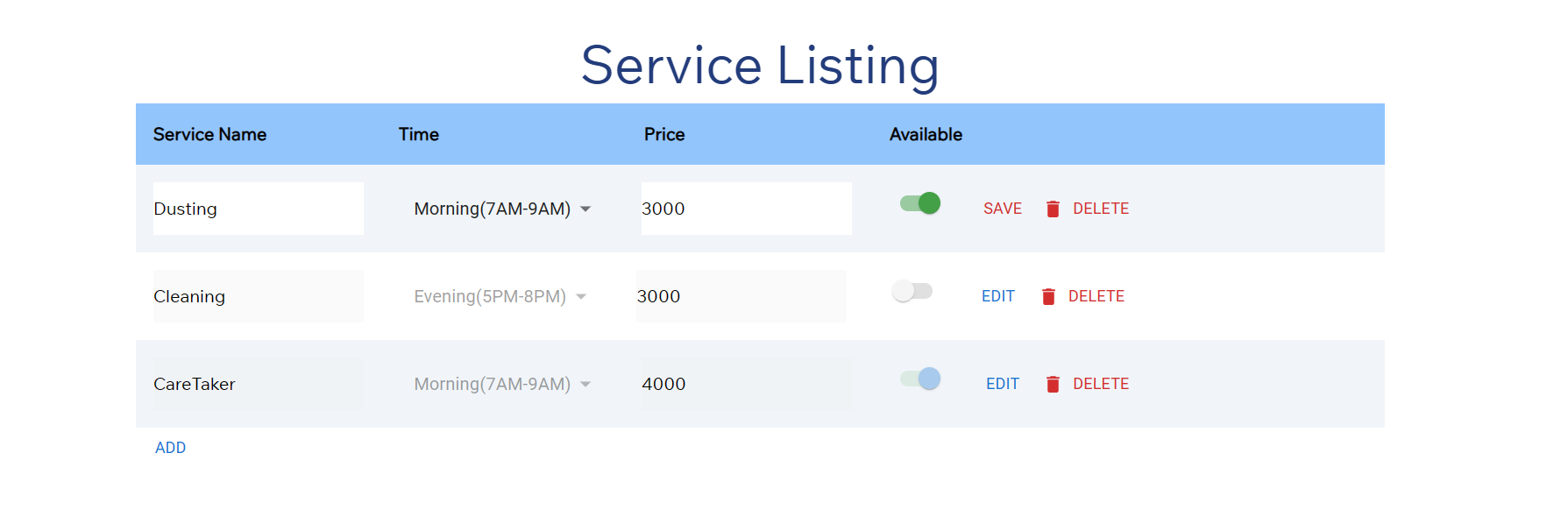


Figure 20: Service listing setting



Figure 21: User Service Requests

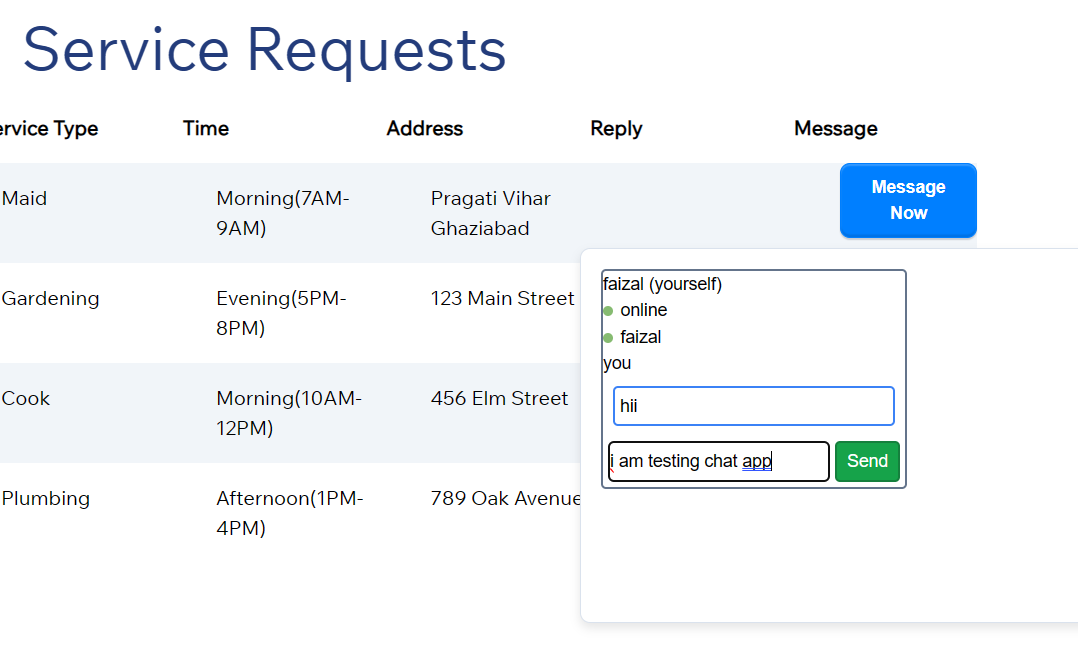


Figure 22: Chat Interface

**5.3 BACK-ENDS REPRESENTATION**

For the backend representation of ServeU, MongoDB Atlas will be used as the database. MongoDB is a NoSQL database that offers flexibility, scalability, and performance, making it suitable for managing the data requirements of ServeU. Here's how the database will be structured:

Collections:

Users: This collection will store user information, including names, email addresses, passwords (hashed for security), and other relevant details.

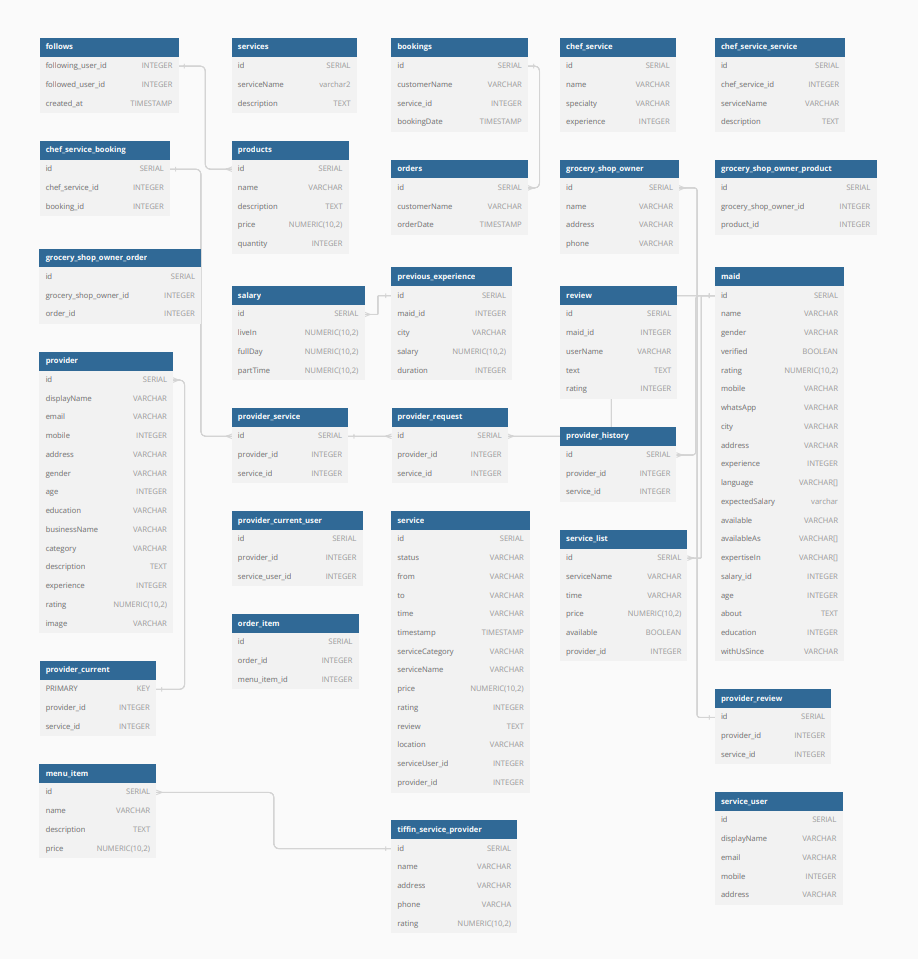
Service Providers: Service provider information will be stored in this collection, including contact details, service offerings, availability, and other pertinent data.

Services: Details of the services offered by service providers will be stored here, such as service names, descriptions, prices, and any associated metadata.

Requests: This collection will track user requests for services, including the service requested, the user making the request, the status of the request, and any additional details.

Chats: All chat messages between users and service providers will be stored in this collection, including sender, receiver, timestamp, and message content.

Document Structure:

****

**CHAPTER 6.**

**CONCLUSION AND FUTURE SCOPE**

**6.1 CONCLUSION**

In conclusion, the successful implementation of our user-friendly web application represents a significant milestone in addressing the challenges faced by bachelors in accessing and managing daily essential services. By offering a centralized platform[3] that seamlessly connects users with reliable service providers, we have provided individuals with convenience, efficiency, and peace of mind. The intuitive user interface and smart algorithms ensure swift and hassle-free service discovery and booking, saving valuable time and effort. Moreover, our web app robust review system and stringent verification processes guarantee the reliability of service providers, fostering trust and satisfaction among users. This reliability, coupled with the removal of complexities associated with essential chores, allows bachelors to focus on their personal and professional pursuits, thereby enhancing their overall quality of life and well-being. Additionally, the app has a positive economic impact by supporting local businesses and service providers, thus stimulating economic growth in communities. Our commitment to continuous improvement, through regular updates, user feedback integration, and innovative feature additions, ensures that our application remains relevant and cutting-edge in meeting the evolving needs of users.

**6.2 FUTURE SCOPE**

Some of the most popular home services are available through this application. When new requirements arise, the system as a whole can be built to accommodate them by expanding its capacity, for which the application offers a suitable offshore service. By simply adding the necessary services and extra payment methods, this application may be extended even further. The present system offers several services, including maid, house cleaning, cook and laundry, and mess food. It can also be expanded to meet the specific needs of the customer. The system can be extended by including services like electrician and mechanic repair, fitter, and catering.

**REFERENCES**

[1] Bhattacharjee, Ananya & Tasnim, Tahrina & Sohel, Md & Chakraborty, Amit & Maruf, Md. Abdullah & Nitun, Shantanu & Ishmam, Alvi & Toha, Tarik & Islam, A. B. M. Alim Al. (2017). Service-Hub: A Better Approach for Developing the System of Online Marketing for Daily Services. 1-5. 10.1109/NSYSS2.2017.8267801.

[2] N. M. Indravasan, Adarsh G, Shruthi C, Shanthi K , Dadapeer, 2018, An Online System for Household Services, INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH & TECHNOLOGY (IJERT) NCESC – 2018 (Volume 6 – Issue 13),

[3] Dhiman, Karan & Phansikar, Mayuresh & Kazi, Mohamed Zain & Thakur, Rutuja. (2022). A WEB-BASED SERVICE MARKETPLACE.

[4] Sarhan, Qusay & Gawdan, Idrees. (2018). Web Applications and Web Services: A Comparative Study. Science Journal of University of Zakho. 6. 10.25271/2018.6.1.375.

[5] Arora, Karishma & Vaishnavi, & Nagpal, Jai. (2023). Implementation of MERN : A Stack of Technologies to Design Effective Web Based Freelancing Applications. International Journal of Scientific Research in Computer Science, Engineering and Information Technology. 23-32. 10.32628/CSEIT23902104.

[6] Almeida, Fernando & Monteiro, J.. (2017). The role of responsive design in web development. Webology. 14. 48-65.

[7] Gurjeet Singh, Madiha Javed, Dr. Balwinder Kaur Dhaliwal (2022). Full Stack Web Development: Vision, Challenges and Future Scope.

[8] Gautam, Indu & Jain, Sameeksha. (2018). A STUDY OF WORK-LIFE BALANCE: CHALLENGES AND SOLUTIONS. Conference: Emerging Role of Leadership, Values and Ethics in Organisational DevelopmentAt: KIET, Ghaziabad.

[9] Gupta, Varun & Saxena, Nitin & Kanungo, Abhas & Salim, & Singh, Gavendra. (2024). An Efficient FrWT and IPCA Tools for an Automated Healthcare CAD System. Wireless Personal Communications. 10.1007/s11277-024-10877-y. Varun Gupta Krishna Institute Of Engineering And Technology | KIET · Department of Electrical and Electronics Engineering.