Service Management System

CS-165 Software Engineering

Session: 2021-2025

Project Supervisor

Sir Laeeq Khan Niazi

Submitted By

Saleem Malik 2021-CS-32

Department of Computer Science
University of Engineering and Technology, Lahore
Pakistan

Contents

List of Figures

1 Introduction

1.1 Background

This project aims to create a user-friendly website called the Professional Services Management System (PSMS). It's like an online marketplace where you can easily hire professionals like electricians, plumbers, beauticians, and more, all from the comfort of your home. The website is designed to be easy to use for everyone. You can find various services, from fixing things in your home to personal care services, all in one place. The process of booking a service is simple you pick the service you need, choose a date and time that works for you, and the website takes care of the rest. One special thing about PSMS is that it makes sure the people offering services are skilled and trustworthy. The website checks and verifies their credentials, so you can be confident in your choices. As your service is being done, the website keeps you updated in real-time. You can see when the professional is expected to arrive and track the progress of the service. And when it comes to paying for the service, the website ensures a secure and safe transaction.

1.2 Objectives

The goal of this project is to make a website called the Professional Services Management System (PSMS). This website will let people easily hire professionals like electricians, plumbers, and beauticians from their homes. The main aim is to make it simple and convenient for users to find and book reliable services online.

1.3 Scope

In this Service Management System User The main purpose is to provide the user different kind of service like user can hire the electricians ,plumbers and many more for its home work and any user can make it service on it like the fiver it a service management platform

1.4 Significance

The Service Provider Management System addresses a crucial need in the market by streamlining the process of hiring professionals for various services. In an era where convenience and efficiency are paramount, this web application offers users the ability to effortlessly connect with verified service providers, schedule appointments, and manage transactions, all from the convenience of their homes. With its user-friendly interface, transparent feedback system, and secure payment options, the system caters to the growing demand for reliable, accessible, and technologically advanced solutions in the service industry. This project not only meets the current market demands for seamless service provision but also sets a new standard for user experience and trust in professional service interactions.

1.5 Functional Features

1. **User Authentication:**
   Clients can securely register on the platform to access professional services.
2. **Service Provider Registration:**
   Service providers can easily register, providing essential contact information and details about offered services.

3. **Service Provider Profiles Maintenance:**
   Service providers can manage and update their profiles, showcasing skills, qualifications, and work history.
4. **Service Provider Matching:**
   The system intelligently matches client needs with suitable service providers for a seamless hiring process.
5. **Billing and Payments:**
   Clients can review transparent billing details, generate invoices, and securely complete transactions online.
6. **Time Scheduling:**
   Clients have the flexibility to schedule service appointments based on their preferred dates and times.
7. **Feedback and Rating:**
   Clients can provide feedback and ratings, fostering transparency and aiding other users in making informed decisions.
8. **Communication:**
   A built-in communication feature allows clients to interact with service providers within the platform.
9. **Category-wise Searching:**
   Clients can easily search for services based on predefined categories, enhancing user experience.
10. **Advanced Searching:**
    Further search refinement options enable clients to specify their requirements in detail.
11. **Previous History:**
    Clients can view their history of previously availed services for reference.
12. **Reward System:**
    Clients can earn reward points for future benefits, including discounts or free services.
13. **Password Privacy:**
    Users can securely change their passwords with a verification process.
14. **Recommendations:**
    Clients can view highly-rated and recommended service providers.
15. **Sorting:**
    Clients can sort service providers based on ratings and prices.
16. **Reports Generation:**
    Service providers can generate performance reports with various details.
17. **Delete Account:**
    Users can request account deletion with a prompt acknowledging permanent data deletion.
18. **Notifying:**
    Users receive email notifications confirming their orders.
19. **Integrating Maps and Navigation:**
    Clients can track the real-time location of service providers.

1.6 Non-Functional Features

1. **Performance:**
   The system is designed to handle multiple users simultaneously without compromising response times.

2. **Scalability:**
   The application is scalable, capable of managing an increasing number of users, service providers, and data without performance decline.

3. **Security:**
   User information, including personal data, is securely stored to ensure privacy and data protection.

4. **Usability:**
   The user interface is intuitive, easy to understand, and user-friendly.

5. **Error Handling:**
   The system is equipped to handle and manage various possible errors effectively.

2 Diagrams with Explanation

2.1 Use Case Diagram

The use case diagram illustrates the various interactions between users and the system. It provides a high-level view of the system's functionalities and the actors involved. Refer to Figure 1 for the visual representation.

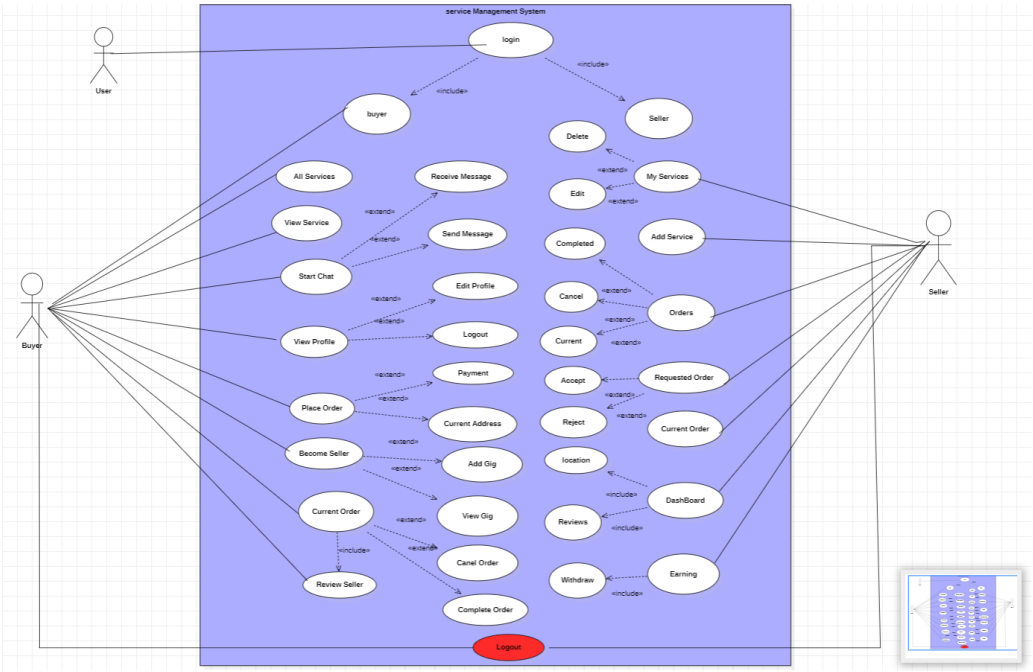

Figure 1: Use Case Diagram

2.2 Class Diagram

The class diagram models the structure of the system by depicting the classes, their attributes, and the relationships between them. Figure 2 visually represents the class diagram.

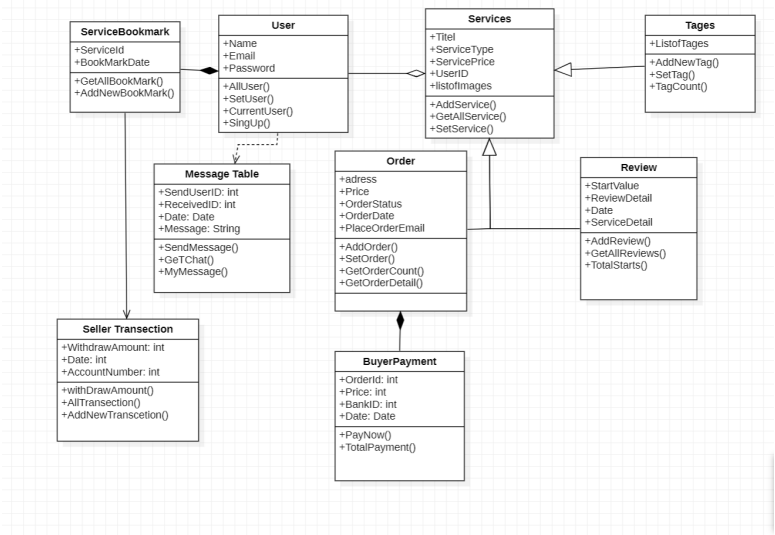

Figure 2: Class Diagram

2.3 Sequence Diagram

The sequence diagram captures the dynamic behavior of the system by illustrating the sequence of interactions between objects over time. See Figure 3 for a visual representation.

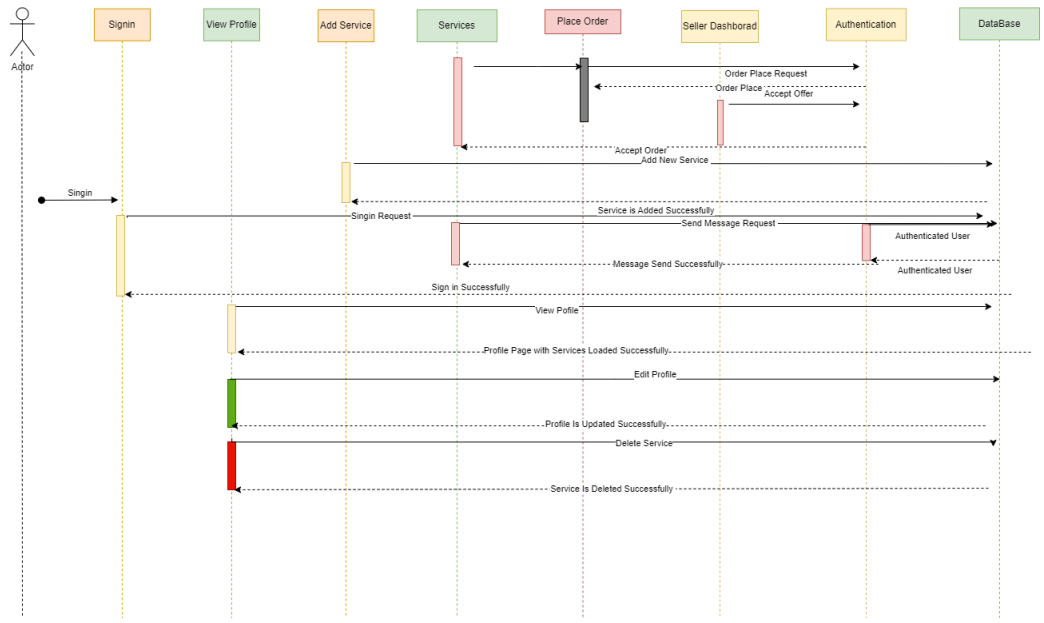

Figure 3: Sequence Diagram

2.4 DFD Level 0

The Level 0 Data Flow Diagram provides a holistic view of the system, showcasing the major processes and data flows visually represents this high-level overview.

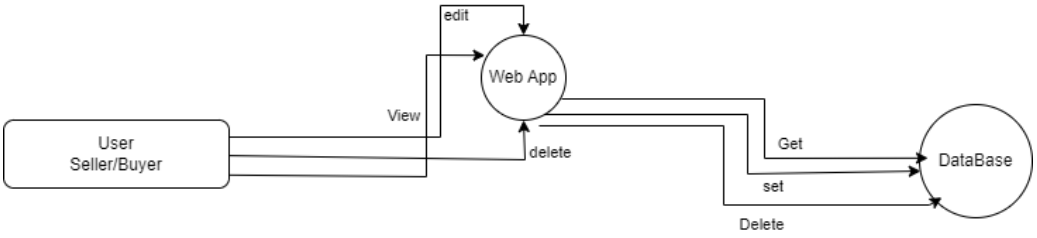

Figure 4: DFD Level 0

2.5 DFD Level 1

The Level 0 Data Flow Diagram provides a holistic view of the system, showcasing the major processes and data flows visually represents this high-level overview.

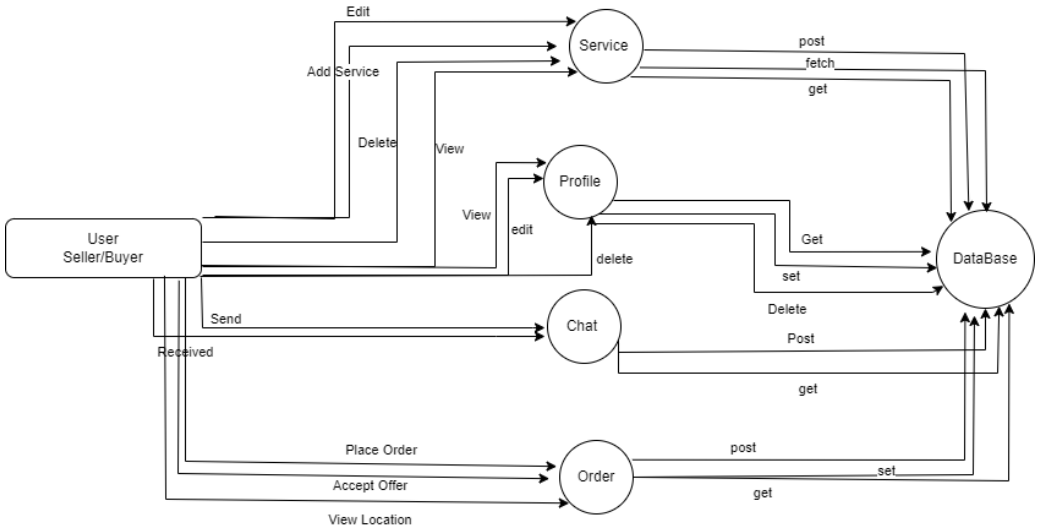

Figure 5: DFD Level 1

2.6 Architecture Diagram

The architecture diagram provides an overview of the system's structure and how its components interact. Figure 6 visually represents the architecture of the Service Management System

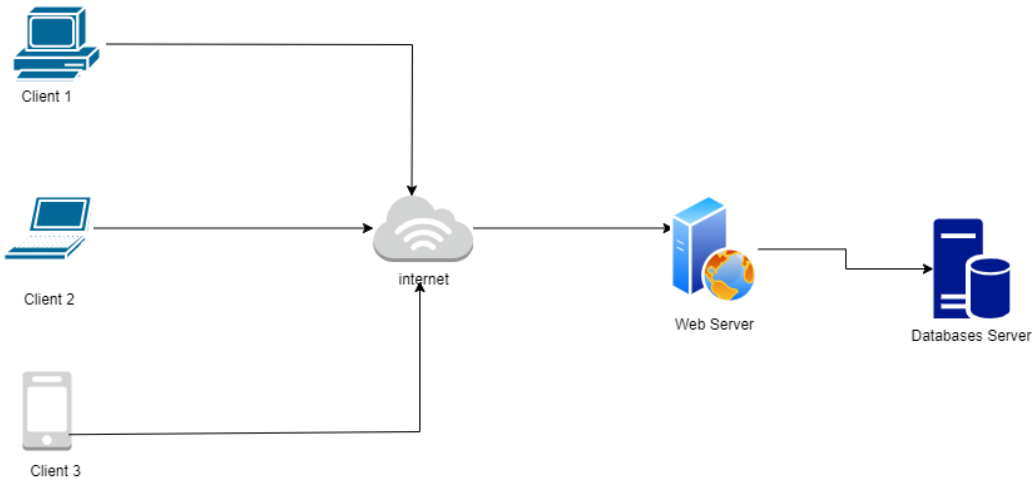

Figure 6: Architecture Diagram

3 Project Implementation Details

3.1 Best Practices Adopted

I have adopted the CamelCase convention for my code. In this convention, I start every first letter of a variable with a capital letter. For instance, if I have a variable to store a person's age, I name it `personAge` instead of `personage` or `PersonAge`. This approach enhances code readability and consistency, making it easier for both myself and other developers to understand and maintain the codebase.

3.1.1 Coding Standards

The code in the Project I follows the rules CamelCase convention for writing good code. It uses consistent spacing, clear names for things, and a neat structure. This makes it easy for developers to understand and work on the code.

3.1.2 Security Measures

The code is careful about security. It handles things like user tokens (a way to prove someone is logged in) in a safe manner. It also uses local storage properly for storing temporary information, which is a good security practice.

3.1.3 Performance Optimization

The code is designed to work well and not slow down. It shows a loading spinner while things are happening in the background, so users know something is going on. This helps in making the app feel responsive and smooth.

3.2 Communication Tools

3.2.1 Collaboration Platforms

To talk and share ideas easily, we use a tool called Slack. It's like a smart chatroom where we can send messages, share files, and stay updated in real-time. This helps us work together smoothly and keeps everyone in the loop.

3.2.2 Meeting Schedule

We've set up a plan to meet regularly, twice with the client. Meetings are like team huddles where we discuss how the project is going, what the client needs, and any changes we might need to make. This way, we make sure everyone is on the same page and can give feedback to make things better.

3.2.3 Communication Protocols

Along with tools like Slack, we have some rules in place to make sure we talk and understand each other well. These rules include how we use our communication tools, how quickly we respond to messages, and how we share important project info. Having these rules helps us avoid confusion and makes our communication smoother, so we can get things done faster and better.

3.3 Version Control

3.3.1 Git Repository Setup

To manage different versions of our project, we use a tool called Git, and our project is hosted on GitHub. Think of Git as a time machine that helps us keep track of changes in our code. GitHub is like a safe place where our project is stored online. This way, we can work on different parts of the project without getting in each other's way.

3.3.2 Branching Strategy

In Git, we use something called branches. Branches are like different copies of our project where we can try out new things without affecting the main version. It's like having separate workspaces. This helps us work on new features or fix issues without messing up what's already working.

3.3.3 Commit Guidelines

Here is the commit Style i follow for My Complete Project

```
git commit -m "Add sign-in functionality"
```

Fixing Bug in User Profile Display:

```
git commit -m "Fix bug in user profile display"
```

Updating Homepage Layout:

```
git commit -m "Update homepage layout"
```

Implementing Password Reset Feature:

```
git commit -m "Implement password reset functionality"
```

Refactoring Code for Better Readability:

```
git commit -m "Refactor code for improved readability"
```

Resolving Merge Conflict in Login Component:

```
git commit -m "Resolve merge conflict in login component"
```

Adding Unit Tests for User Registration:

```
git commit -m "Add unit tests for user registration"
```

Fixing Responsive Design Issue in Navigation Bar:

```
git commit -m "Fix responsive design issue in navigation bar"
```

3.4 Project Management

3.4.1 Task Breakdown

We adopt a systematic approach to task breakdown, breaking down major project goals into smaller, manageable tasks. This ensures clarity and a step-by-step progression throughout the development process.

3.4.2 Milestone Planning

Our project plan incorporates milestone planning to mark significant achievements. This helps us monitor progress, set realistic goals, and ensure that the project is advancing according to the established timeline.

3.4.3 Issue Tracking

To streamline issue resolution, we implement issue tracking. This involves logging and monitoring challenges or bugs that arise during the project. It enables us to address issues promptly, maintain project quality, and enhance overall productivity.

4 Application Working

4.1 User Registration

In Sigh-Up i take the User Name,Email And Password and save it in database every user can make only one account with one email Check this Figure 7

4.1.1 Screenshot

Here is the UI for singin Page 7

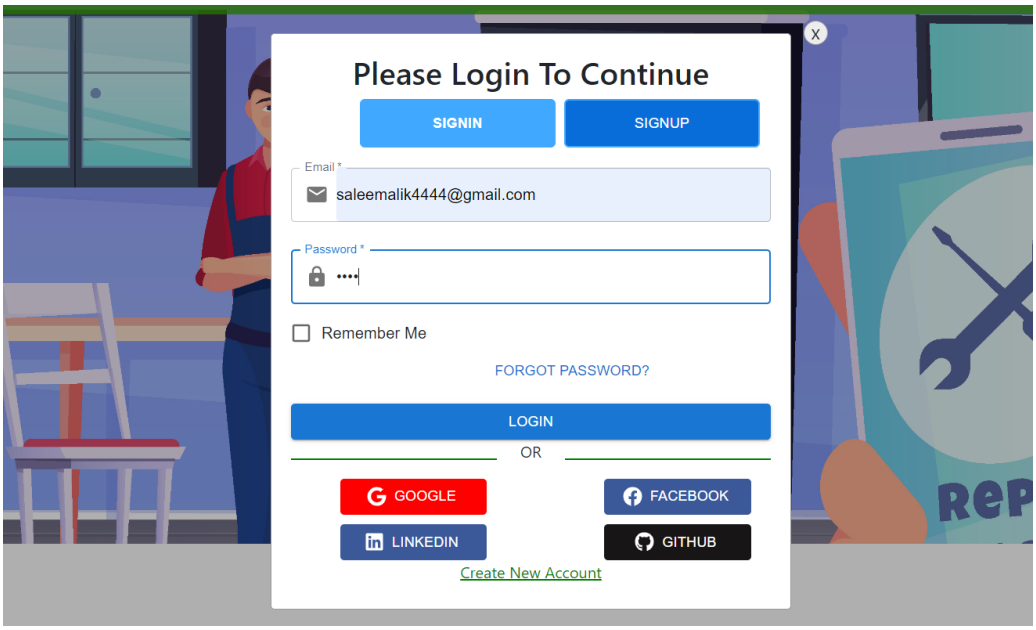

Figure 7: User Sign-in

4.1.2 Code Snippet

here is the code Snippet for the SingIn Page

```jsx
import LockIcon from '@mui/icons-material/Lock';
import Checkbox from '@mui/material/Checkbox';
import FormControlLabel from '@mui/material/FormControlLabel';
import { useState } from 'react';
import { FaFacebook, FaGithub, FaGoogle, FaLinkedin } from 'react-icons/fa';
import { AppContext } from '../../App';
import { useNavigate } from 'react-router-dom';
import Snackbar from '@mui/material/Snackbar';
import MuiAlert from '@mui/material/Alert';
import Loading from "react-loading";
const SignIn = () => {
    const { HomeHeader, SetHomeHeader } = React.useContext(AppContext);
    const navigate = useNavigate();
    const ValidUser = () => {
        SetHomeHeader(true);
    }
    const [formData, setFormData] = useState({
        Email: '',
        Password: '',
    });
    const [isMessageSent, setMessageSent] = useState(false);
    const [ErrorContent, setErrorContent] = useState('');
    const [severity, setSeverity] = useState('Success');
    const [isLoading, setLoading] = useState(false);
    const handleCloseSnackbar = () => {
        setMessageSent(false);
    }
    const handleChange = (e) => {
        const { name, value } = e.target;
        setFormData({
            ...formData,
            [name]: value,
        });
    };
    const handleSubmit = async (e) => {···
    };

    return (···
    );
};
```

Figure 8: User Sign-in

4.2 View Services Top Rated Sellers

In View Services Page User Can view All the Services That are added in our system and User can Place Order to Any Service here are some working Screenshot

4.2.1 Screenshot

Here is the working Screen-short of the Top-Rated-Sellers

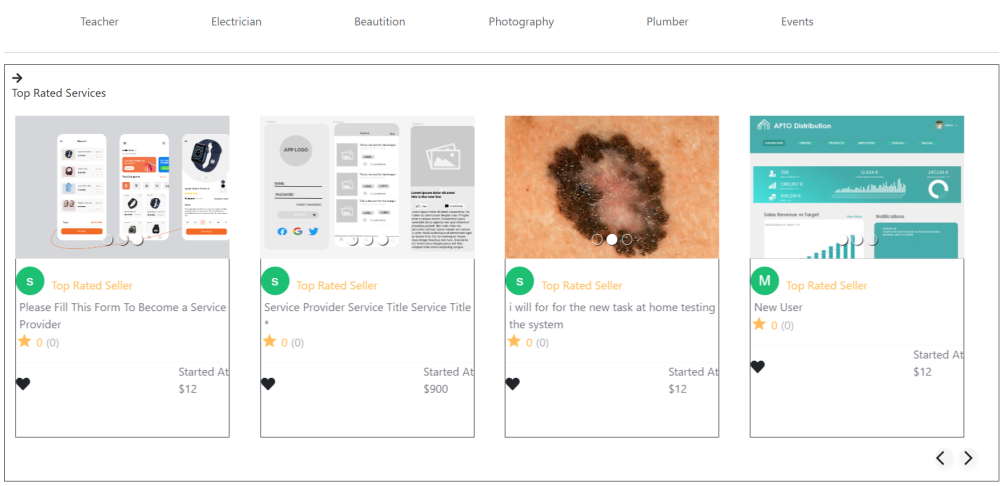

Figure 9: Top Rated Sellers

4.2.2 Code Snippet

Here is the Code Snippet for the Top-Rated-Sellers

```jsx
import React, { useState, useEffect } from 'react';
import Sellar_Cart from '../Saller/Sellar_Cart';
import Carousel from 'react-multi-carousel';
import 'react-multi-carousel/lib/styles.css';
import { FaArrowRight } from 'react-icons/fa';
import ArrowBackIosIcon from '@mui/icons-material/ArrowBackIos';
import ArrowForwardIosIcon from '@mui/icons-material/ArrowForwardIos';


const Top_Rated_slider = () => {
    const [AllServices, setAllServices] = useState([]);
    const [loading, setLoading] = useState(false);

    useEffect(() => {
        const GetUserData = async () => {
            setLoading(true);
            setAllServices([]);
            try {
                const apiUrl = `http://localhost:5000/api/GetAllServicesForallUser`;
                const response = await fetch(apiUrl, {
                    method: 'GET',
                    headers: {
                        'Content-Type': 'application/json',
                    },
                });
                const data = await response.json();
                if (response.ok) {
                    console.log(data);
                    setAllServices(data);
                    setLoading(false);
                } else {
                    console.log('Error');
                }
            } catch (error) {
                console.log(error);
            }
        };
        GetUserData();

    }, []);
```

Figure 10: Top Rated Sellers Code

4.3 Service Details

In Service Details Page we show the complete Gallery of the Service With The complete Description of the Service

4.3.1 Screenshot

Here is the Some Working Screen Shorts of the Page

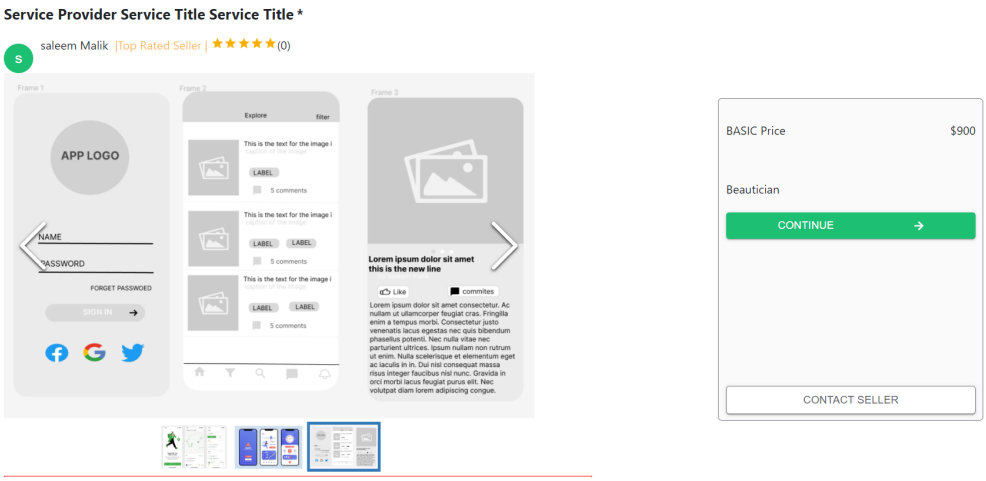

Figure 11: Service Details

4.3.2 Code Snippet

Here is the Code Snippet for the Service Details Page

```
import Gallery from 'react-image-gallery';
import 'react-image-gallery/styles/css/image-gallery.css';
import { Button, Grid } from '@mui/material';
import { FaArrowRight } from 'react-icons/fa';
import Avatar from '@mui/material/Avatar';
import Rating from '@mui/material/Rating';
import { LinearProgress, Typography } from '@mui/material';
import ReviewComponent from './ReviewComponet.js';
import { useEffect, useState } from 'react';
import { AppContext } from '../../App.js';
import * as React from 'react';
import PlaceOrderShow from './PlaceOrder.js';
import Chat from '../Saller/textingCard.js'
const Gig_View = () => {
  const { Current_Service, Set_Current_Service } = React.useContext(AppContext);
  const [PlaceOrder, SetPlaceOrder] = useState(false);
  const [PlaceOrderData, SetPlaceOrderData] = useState(false);
  const [one, setOne] = useState(Current_Service);
  const [ShowGig, SetShowGig] = useState(true);
  const [CheckChat, SetShowChat] = useState(false);
  const [SenderID, SetSenderID] = useState();
  const [ReceiverID, SetReceiverID] = useState();
  // Create a state variable for 'one'
  const ShowChatComponent = async () => {···
  }
  useEffect(() => {···
  }, []);
  //  image gallary is done
  const images = one.Gallary.map((url) => ({···
  }));
  const ShowDilog = () => {···
  }
  const HideDilog = () => {···
  }
  return (···
  );
};
export default Gig_View;
```

Figure 12: Service Details Code

4.4 Seller Chat

In Seller Chat Seller Can Send Message to the Buyer and Receive the Message from the buyer

4.4.1 Screenshot

Here is the working Screenshot of the Project

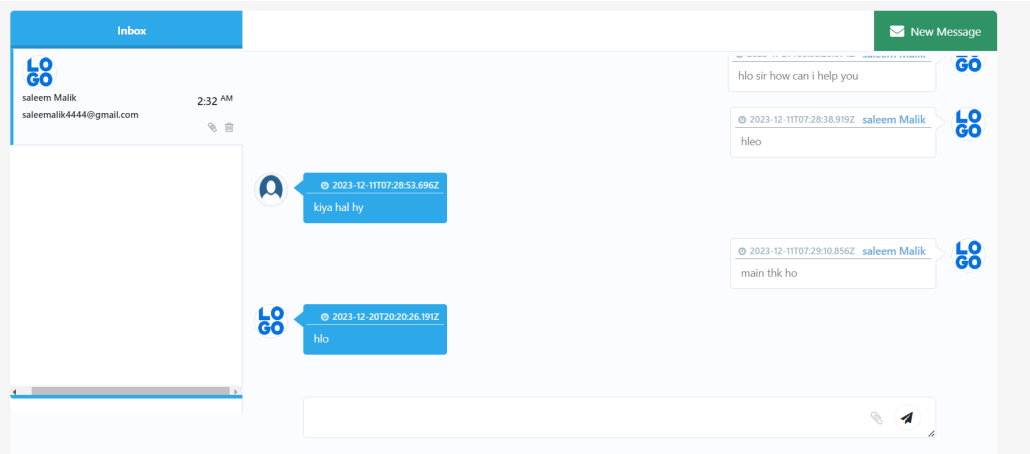

Figure 13: Chat Page

4.4.2 Code Snippet

Here is Some Code Snippet for Chat Page

```
import React, { useState, useEffect } from 'react';
import 'bootstrap/dist/css/bootstrap.min.css';
import './Testing.css';


function Chat({ Sender, Receiver }) {
    const [active, setActive] = React.useState(1); // Loading state
    const [LoadChat, setLoadChat] = React.useState(1); // Loading state
    const [IDUserWhoSendMessage] = useState(Sender); // Replace with your actual sender ID
    const [IDUserWhoGetMessage, SetIDUserWhoGetMessage] = useState(Receiver); // Replace with your actual receiver ID
    const [Profile, SetProfiles] = useState([]);
    const [newMessage, setNewMessage] = useState('');
    const [messages, setMessagesData] = useState([]);
    const [SelectedUserSendMessages, SetSelectedUserSendMessages] = useState([]);

    const handleChatClick = (userId) => {···
    };
    useEffect(() => {···
    }, []);

    const AllUsersData = async () => {···
    }

    const fetchMessagesNew = async () => {···
    };

    const sendMessage = async () => {···
    };
    return (···
    );
}
export default Chat;
```

Figure 14: Chat Page code

4.5 Seller dashboard

In Seller dashboard page seller Can View the Number of the request Order and its number of order he completed

4.5.1 Screenshot

This is the Dashboard Working Page

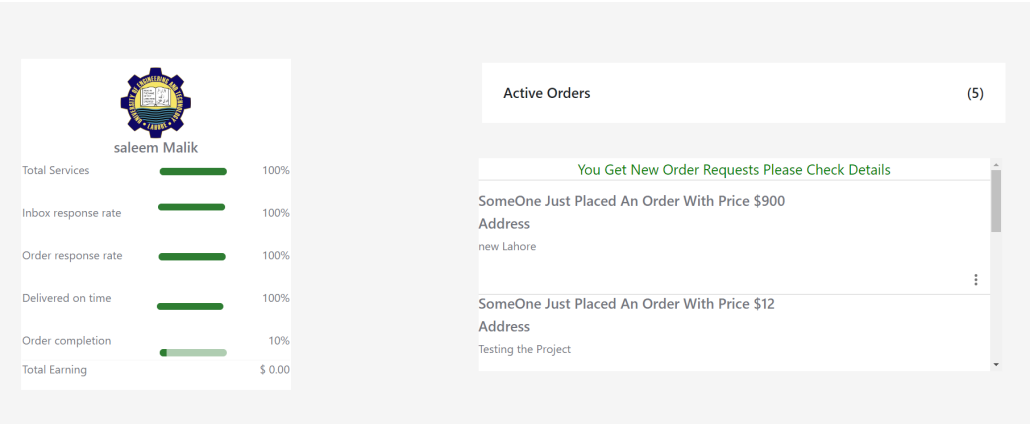

Figure 15: Dashboard UI

4.5.2 Code Snippet

Here is the code of the Seller Dashboard

```jsx
import React from 'react';
import Grid from '@mui/material/Grid';
import { ProfileLeftside } from './ProfileLeftside';
import ViewServices from './ViewServices';
import OrderSelecter from './OrderSelecter';
import Chat from './textingCard';
import {UserProfile} from './ViewSallerProfile';



const App = () => {
    const Receiver = localStorage.getItem('Sender');
    const Sender = localStorage.getItem('Receiver');
    return (

        <div style={{ backgroundColor: '#f5f5f5', width: '100%', height: '100%', minHeight: '100vh', marginTop: '-15px' }}>
            <Grid container>
                <Grid item xs={6} sm={6} md={6} lg={6} width={2} marginTop={10}>
                    <UserProfile UserID = {Sender} />
                </Grid>
                <Grid item xs={12} sm={6} md={6} lg={6} marginX={-5} marginTop={10}>
                    <OrderSelecter />
                </Grid>

                <Grid item xs={12} sm={12} md={12} lg={12} marginX={-5} marginTop={10}>
                    <Chat Sender={Sender} Receiver={Receiver} />
                </Grid>

            </Grid>
        </div>
    );
};

export default App;
```

Figure 16: Dashboard Code

4.6 Seller Profile

In Seller Profile Section Seller Can View its Total Number of services it can also update or delete any of them

4.6.1 Screenshot

Here is the working Screenshot of the Seller Profile

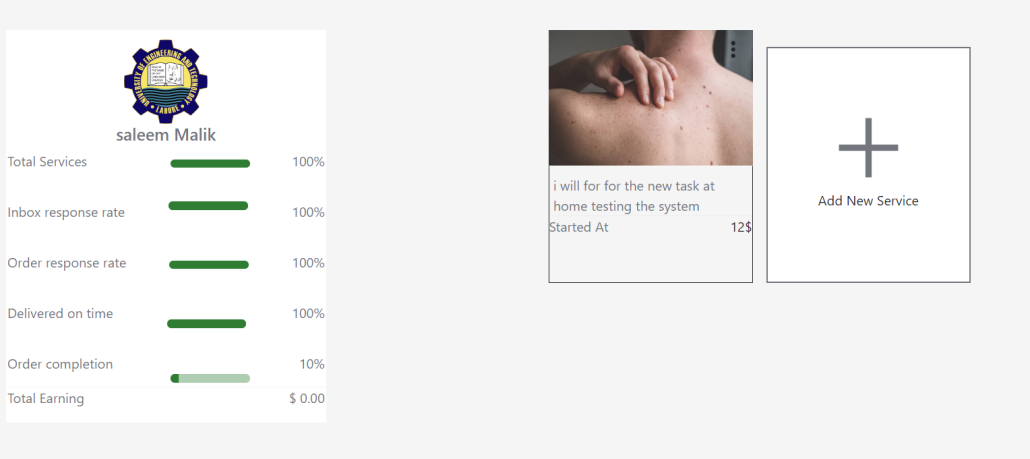

Figure 17: Seller Profile View

4.6.2 Code Snippet

Here is the code of the View Profile

```
import React from 'react'
import { Grid } from '@mui/material';
import ViewServices from './ViewServices';
import { ProfileLeftside } from './ProfileLeftside';




export const ViewProfile = () => {


    return (
        <>
            <Grid container className='viewProfile-main-div'
            >
                <Grid item xs={6} sm={6} md={6} lg={6} width={2}  marginTop={10}>
                    <ProfileLeftside />
                </Grid>
                <Grid item xs={12} sm={6} md={6} lg={6} marginX={-5} marginTop={10}>
                    <ViewServices />
                </Grid>
            </Grid>
        </>
    );

}
```

Figure 18: View Profile Code

5 Conclusion

5.1 Summary of Achievements

In summary, the project has achieved significant milestones, including the successful implementation of key features such as the Add Seller Service ,Placing Order, Chat implementation thought which Seller can Send or received the message from the buyer ,there is the Google Map Implantation thought which seller can view the Current Location of the Buyer and Order Location. These accomplishments mark progress toward the project's goals and contribute to an enhanced user experience.

5.2 Challenges Faced

Throughout the project, we encountered challenges, such as resolving merge conflicts and addressing bugs in the In the Implantation of the Complex feature Such as Google Map Implementation , Chat system Implementation , Add New Service and Many more. However, these challenges served as learning opportunities and were overcome through collaborative problem-solving and effective communication within the team.

5.3 Lessons Learned

The project provided valuable lessons in project management, version control, and coding standards. Establishing clear communication protocols, utilizing Git for version control, and adhering to coding standards like CamelCase improved overall project organization and code quality.

5.4 Future Enhancements

Looking ahead, there is room for future enhancements. This includes the potential addition of advanced features, optimization of existing functionalities, and further refinement of the user interface. Future development efforts will build upon the project's current foundation, incorporating user feedback and emerging technologies for continuous improvement.