EASYWORK – Service Provider Web Application

A Major Project Report Submitted in Partial Fulfillment for the Award of the Degree of Bachelor of Technology in Computer Science and Engineering

***Submitted to***



### Dr. A.P.J. ABDUL KALAM TECHNICAL UNIVERSITY, LUCKNOW

***Submitted by:***

**QURESHI SOHEL SHAFIQUE (2003420100064)**

**OBAIDUR RAHMAN (2003420100059)**

**MUKESH PRAJAPATI (2003420100055)**

**MO SAHIL HUSSAIN (2003420100054)**

**UNDER THE SUPERVISION OF**

Mr. Shivendra Singh (Assistant Professor)



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING UNITED COLLEGE OF ENGINEERING AND RESEARCH, PRAYAGRAJ**

**MAY 2024**

# CANDIDATE’S DECLARATION

We, hereby certify that the project entitled “ EasyWork ”submitted by us in partial fulfillment of the requirement for the award of degree of the B. Tech. (Computer Science & Engineering) submitted to Dr. A.P.J. Abdul Kalam Technical University, Lucknow at United College of Engineering and Research, Prayagraj is an authentic record of our own work carried out duringa period from June, 2023 to May, 2024 under the guidance of Mr.Shivendra Singh, Assistant Professor, Department of Computer Science & Engineering. The matter presented in this project has not formed the basis for the award of anyother degree, diploma, fellowship or any other similar titles.

#### Signature of the Student

(Qureshi Sohel Shafique 2003420100064)

#### Signature of the Student

(Obaidur Rahman 2003420100059)

#### Signature of the Student

(Mukesh Prajapati 2003420100055)

#### Signature of the Student

(Mo Sahil Hussain 2003420100054)

#### Place:

#### Date:

i

# CERTIFICATE

This is to certify that the project titled “EasyWork” is the bona fide work carried out by

Qureshi Sohel Shafique (2003420100064), Obaidur Rahman (2003420100059), Mukesh Prajapati (2003420100055), Mo Sahil Hussain (2003420100054) in partial fulfillment of the requirement for the award of degree of the B. Tech. (Computer Science &

Engineering) submitted to Dr. A.P.J Abdul Kalam Technical University, Lucknow at United College of Engineering and Research, Prayagraj is an authentic record of their own work carried out during a period from June, 2023 to May, 2024 under the guidance of Mr.Shivendra Singh, Assistant Professor , Department of Computer Science & Engineering. The Major Project Viva-Voce Examination has been held on

.

#### Signature of the Guide

Mr. Shivendra Singh

#### Signature of Project Coordinator

Mr. Shyam Bahadur Verma

#### Signature of the Head of Department

Dr. Vijay Kumar Dwivedi

# ABSTRACT

The "EasyWork-Service Provider Web Application" aims to revolutionize the utility service booking experience by providing a centralized platform for users to access services such as beauty, electrical maintenance, home cleaning, and pest control. The application offers a seamless user experience, allowing users to select services, book appointments, pay charges, and provide feedback all in one place.

Key features of EasyWork include a user-friendly interface for service selection, a calendar integration for easy appointment booking, and a secure payment gateway for hassle-free transactions. The application leverages Angular for the frontend, GoLang for the backend, and MySQL with the GORM library for the database, ensuring a robust and scalable solution.

EasyWork's primary goal is to provide users with a convenient and reliable platform for accessing utility services. By offering easy booking and cancellation options without additional charges, the application aims to simplify the process of booking utility services and improve overall user satisfaction.

In conclusion, EasyWork strives to be the go-to platform for all utility service needs.

Offering a Comprehensive and user-friendly solutions for users.

# ACKNOWLEDGEMENT

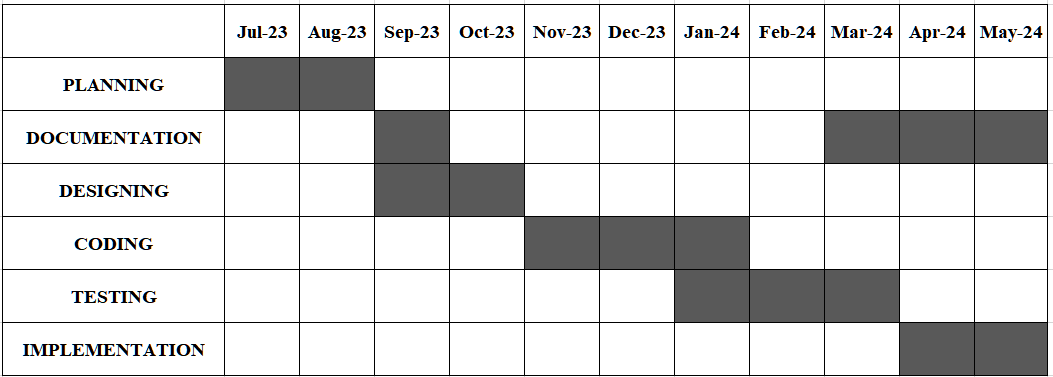
We express our sincere gratitude to the Dr. A.P.J Abdul Kalam Technical University, Lucknow for giving us the opportunity to work on the Major Project during our final year of B.Tech. (CSE) is an important aspect in the field of engineering.

We would like to thank Prof. H.P. Shukla, Principal and Dr. Vijay Kumar Dwivedi, Head of Department, CSE at United College of Engineering and Research, Prayagraj for their kind support.

We also owe our sincerest gratitude towards Mr.Shivendra Singh, Assistant Professor, Department of Computer Science & Engineering for his valuable advice and healthy criticism throughout our project which helped us immensely to complete our work successfully.

We would also like to thank everyone who has knowingly and unknowingly helped us throughout our work. Last but not the least, a word of thanks for the authors of all those books and papers whichwe have consulted during our project work as well as for preparing the report.

**PROGRESS CHART**



**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **FIG No.** | **FIGURE** | **PAGE** |
| Fig 3.4.1 | Flowchart for EasyWork Working | 15 |
| Fig 3.4.2 | Entity Realationship Diagram | 16 |
| Fig 4.1 | HomePage | 31 |
| Fig 4.2 | Service Page | 32 |
| Fig 4.3 | Schedule Page | 33 |
| Fig 4.4 | Booking Page | 34 |
| Fig 4.5 | Adaptive Components | 35 |
| Fig 4.6 | Sign In Page | 36 |
| Fig 4.7 | Sign Up Page | 37 |
| Fig 7 | Plagrism Report | 42 |

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **S No.** | **TITLE** | **PAGE** |
| I | **CANIDATE’S DECLARATION** | i |
| II | **CERTIFICATE** | ii |
| III | **ABSTRACT** | iii |
| IV | **ACKNOWLEDGEMENT** | iv |
| V | **PROGRESS CHART** | v |
| VI | **LSIT OF FIGURES** | vi |
| 1. | **INTRODUCTION** | 1-7 |
|  | 1.1 Overview of the "EasyWork" Web Application | 1- 2 |
|  | 1.2 Problem Statement | 3 |
|  | 1.3 Hardware Specification | 4-5 |
|  | 1.4 Software Specification | 6-7 |
| 2. | **LITRATURE SURVEY** | 8-11 |
|  | 2.1 Existing System | 8 |
|  | 2.2 Proposed System | 9-10 |
|  | 2.3 Feasibility Study | 11 |
|  | 2.3.1 Technical Feasibility | 11 |
|  | 2.3.2 Operational Feasibility | 11 |
|  | 2.3.3 Economic Feasibility | 11 |
| 3. | **SYSTEM ANALYSIS AND DESIGN** | 21-35 |
|  | 3.1 Functional Requirements | 12-13 |
|  | 3.2 Non-Functional Requirements | 13-14 |
|  | 3.3 Flow Charts | 15 |
|  | 3.4 DESIGN AND TEST STEPS / CRITERIA | 16-18 |
|  | 3.4.1 Design Steps | 32 |
|  | 3.4.2 Test Steps | 33 |
|  | 3.4.3 Criteria for Evaluation | 34-35 |
|  | 3.5 API Doocumentation | 19-26 |
|  | 3.6 Algorithms | 27-28 |
|  | 3.7 Testing Process | 29-30 |
|  | **RESULTS AND OUTPUT** | 31-38 |
|  | **CONCLUSION AND FUTURE SCOPE** | 39-40 |
|  | **REFERENCES** | 41 |
|  | **PLAGARISM REPORT** | 42-43 |

**CHAPTER 1: INTRODUCTION**

## OVERVIEW OF THE "EASYWORK"

In today's fast-paced world, accessing reliable services efficiently can be a challenge. Traditional methods of service requests often involve cumbersome processes, leading to delays and frustrations for users. Whether it's finding a plumber for a leaking faucet, a mechanic for a car repair, or a tutor for academic assistance, navigating through the labyrinth of service providers can be daunting and time-consuming. Moreover, the lack of transparency and accountability in the service industry exacerbates these challenges, leaving users uncertain about the quality and reliability of the services they receive.

The problem statement for our project, EasyWork, is rooted in these pervasive pain points faced by users in their quest for reliable services. Recognizing the need for a modern and streamlined solution, EasyWork aims to revolutionize the way users engage with service providers. By leveraging innovative technologies and user-centric design principles, EasyWork seeks to address the shortcomings of traditional service request methods and enhance user satisfaction.

At the heart of the problem lies the inefficiency of existing service request processes. From phone calls and emails to word-of-mouth referrals, the methods available for accessing services are often fragmented and laborious. Users must navigate through a maze of directories, listings, and recommendations, hoping to find a suitable service provider within a reasonable timeframe. This fragmented approach not only consumes valuable time and effort but also increases the likelihood of users encountering unreliable or unqualified service providers.

Furthermore, the lack of transparency and communication exacerbates the user experience. Once a service request is initiated, users are often left in the dark regarding the status of their request, leading to anxiety and frustration. Without real-time updates or clear communication channels, users are forced to rely on sporadic updates from service providers or third-party intermediaries, further complicating the process.

EasyWork's mission is to address these pain points by streamlining the service request

process and enhancing user satisfaction. By providing a centralized platform where users can easily browse, select, and request services, EasyWork aims to simplify the entire service procurement journey. Through intuitive interfaces, seamless communication channels, and transparent workflows, EasyWork will empower users to make informed decisions and access high-quality services with confidence.

By embracing innovative technologies such as the Angular framework for the frontend, GoLang for the backend, and MySQL with the GORM library for the database, EasyWork seeks to create a modern and scalable platform that meets the evolving needs of users in today's digital age. Through continuous iteration and improvement, EasyWork endeavors to revolutionize the way users engage with service providers, making the process seamless,

Transperent and Intuitive.

## PROBLEM STATEMENT

In today's fast-paced world, accessing reliable services efficiently can be a significant challenge. Traditional methods of service requests, such as phone calls, emails, and word-of-mouth referrals, often lead to delays, frustrations, and uncertainty about the quality of services. Users face difficulties in finding suitable service providers within a reasonable timeframe, navigating through directories, listings, and recommendations. This fragmented approach consumes valuable time and effort, increasing the likelihood of encountering unreliable or unqualified service providers.

Moreover, the lack of transparency and communication further complicates the user experience. Once a service request is initiated, users are often left uninformed about the status of their request, leading to anxiety and frustration. Without real-time updates or clear communication channels, users must rely on sporadic updates from service providers or third-party intermediaries.

EasyWork aims to address these challenges by revolutionizing the way users engage with service providers. The platform recognizes the need for a modern, streamlined solution that simplifies the service procurement journey and enhances user satisfaction. By providing a centralized platform where users can easily browse, select, and request services, EasyWork aims to simplify the entire process. Through intuitive interfaces, seamless communication channels, and transparent workflows, EasyWork will empower users to make informed decisions and access high-quality services with confidence.

EasyWork's mission is to streamline the service request process and enhance user satisfaction. By leveraging innovative technologies such as the Angular framework for the frontend, GoLang for the backend, and MySQL with the GORM library for the database, EasyWork seeks to create a modern and scalable platform that meets the evolving needs of users in today's digital age. Through continuous iteration and improvement, EasyWork endeavors to revolutionize the way users engage with service providers, making the process seamless, transparent, and intuitive.

The key challenges addressed by EasyWork include: Fragmented and laborious service request processes Lack of transparency and communication in the service industry

Difficulty in finding reliable and qualified service providers Time-consuming and inefficient methods of accessing services EasyWork's innovative approach aims to transform the service request platform, setting new standards for efficiency, reliability, and convenience in

the digital age.

## HARDWARE SPECIFICATION

EasyWork's hardware specifications are tailored to support the platform's robust performance and scalability, ensuring an optimal user experience across various devices. The hardware components encompass both server-side infrastructure and client-side devices utilized by users accessing the platform.

Server Infrastructure:

EasyWork's server infrastructure is designed to handle the computational and storage requirements of the platform, ensuring seamless operation and responsiveness. Key hardware specifications include:

**High-performance Servers:** EasyWork will be hosted on high-performance servers equipped with multi-core processors and ample RAM to handle concurrent user requests efficiently.

**Scalable Storage:** The server infrastructure will feature scalable storage solutions, such as solid-state drives (SSDs) or cloud-based storage, to accommodate growing data volumes and ensure rapid access to information.

**Redundancy and Failover:** Redundant hardware configurations and failover mechanisms will be implemented to minimize downtime and ensure uninterrupted service availability in case of hardware failures or maintenance activities.

**Network Infrastructure:** The network infrastructure supporting EasyWork's servers will be equipped with high-speed internet connectivity and robust security protocols to facilitate seamless data transfer and protect against potential security threats.

**Client Devices:** EasyWork is designed to be accessible from a wide range of client devices, including desktop computers, laptops, tablets, and smartphones. While the hardware specifications of client devices may vary, EasyWork is optimized to deliver a consistent user experience across different platforms.

Compatibility: EasyWork's web-based platform is compatible with popular web browsers such as Google Chrome, Mozilla Firefox, Safari, and Microsoft Edge, ensuring compatibility with a diverse range of desktop and mobile devices.

**Responsive Design:** The user interface of EasyWork is built using responsive design principles, allowing it to adapt seamlessly to various screen sizes and resolutions. Whether accessed from a desktop computer or a smartphone, users can expect a consistent and intuitive user experience.

**Performance Considerations:** While EasyWork is designed to be lightweight and optimized for performance, users may benefit from accessing the platform using devices with sufficient processing power, memory, and network connectivity to ensure smooth operation and responsiveness.

Overall, EasyWork's hardware specifications are designed to support the platform's performance, scalability, and accessibility, enabling users to access reliable services efficiently from a wide range of devices while

Ensuring seamless operation and responsiveness.

## SOFTWARE SPECIFICATION

This project utilizes a powerful and versatile technology stack to deliver a robust, scalable, and user-friendly platform. The key technologies employed include:

**1.4.1 Angular:**

Angular is a comprehensive front-end framework developed and maintained by Google. It provides a structured approach to building dynamic and interactive web applications. Angular's features include two-way data binding, dependency injection, and a component-based architecture, which enable developers to create modular and maintainable code.

**1.4.2 GoLang:**

GoLang, also known as Golang, is a statically typed, compiled programming language designed for building efficient and reliable software. Developed by Google, GoLang offers built-in support for concurrency and is known for its simplicity, performance, and scalability. In this project, GoLang is used to develop the backend services, handling requests, processing data, and interacting with the database.

**1.4.3 MySQL with GORM Library:**

MySQL is a popular open-source relational database management system used for storing and managing data. The GORM library is a powerful Object-Relational Mapping (ORM) tool for GoLang, which provides a convenient and efficient way to interact with MySQL databases. GORM simplifies database operations such as querying, inserting, updating, and deleting records, making database management easier and more efficient.

**1.4.4 Git:**

Git is a distributed version control system used for tracking changes in the project codebase. It allows multiple developers to collaborate on the same project, managing changes and resolving conflicts efficiently. Git provides features such as branching, merging, and tagging, which are essential for managing complex software development workflows.

**1.4.5 Visual Studio Code:**

Visual Studio Code (VS Code) is a lightweight, open-source code editor developed by Microsoft. It provides a range of features for writing, debugging, and deploying code, including syntax highlighting, code completion, and integrated terminal. VS Code is highly customizable and supports a wide range of programming languages and extensions, making it a popular choice among developers for building web applications.

By leveraging this technology stack, the project delivers a modern, scalable, and efficient platform that meets the demands of modern web development. Angular provides a robust front-end framework for building responsive and interactive user interfaces, while GoLang ensures high performance and reliability on the backend. MySQL with the GORM library offers a reliable database solution, and Git enables efficient version control. Visual Studio Code provides a user-friendly environment for developing and maintaining the project codebase, ensuring a seamless development experience.

**CHAPTER 2: LITRATURE SURVEY**

Conducting a comprehensive literature survey is essential for understanding the current state-of-the-art, identifying existing solutions, and gathering insights that can inform the development of EasyWork. This section explores the existing literature and research relevant to service request platforms, user experience design, and technology frameworks.

## EXISTING SYSTEM

The existing service request platforms, such as Urban Company, TaskRabbit, and Thumbtack, represent a diverse and thriving ecosystem catering to a wide range of user needs. These platforms offer a plethora of services, spanning from home maintenance and repairs to professional services and personal assistance. They leverage technology to connect users with service providers, streamlining the service request process and enhancing user satisfaction.

Urban Company, for example, offers a comprehensive range of services, including beauty, wellness, home repairs, and maintenance. Users can easily browse through the available services, select their preferred service providers, and schedule appointments at their convenience. The platform also provides features for online payment and feedback, ensuring a seamless and user-friendly experience.

TaskRabbit focuses on providing on-demand services, allowing users to find skilled individuals for various tasks, such as furniture assembly, home cleaning, and handyman services. The platform enables users to post their tasks, receive offers from taskers, and choose the most suitable candidate based on reviews and ratings.

Thumbtack, on the other hand, caters to a broader range of services, including event planning, tutoring, and professional services. The platform emphasizes the importance of trust and reliability, enabling users to find and hire service providers with confidence.

Overall, these existing platforms demonstrate the effectiveness of leveraging technology to streamline service requests, connect users with service providers, and enhance user satisfaction. Their success serves as a valuable benchmark for EasyWork, highlighting the importance of user-centric design, seamless user experiences, and efficient service delivery mechanisms.

## PROPOSED SYSTEM

EasyWork aims to revolutionize the service request platform landscape by combining innovative features and cutting-edge technologies to deliver a seamless and user-centric experience. The platform builds upon the strengths of existing platforms while introducing new elements to enhance user satisfaction and operational efficiency.

One of the key aspects of EasyWork is its use of Angular for frontend development. Angular provides a robust and structured framework for building dynamic and interactive user interfaces. By leveraging Angular's features such as two-way data binding and dependency injection, EasyWork aims to deliver a responsive and intuitive user experience.

On the backend, EasyWork utilizes GoLang, a statically typed, compiled programming language known for its efficiency and scalability. GoLang's built-in support for concurrency and its simple yet powerful syntax make it an ideal choice for building the backend services of EasyWork. Additionally, GoLang's performance and reliability ensure that EasyWork can handle a large number of concurrent users and requests.

For data storage, EasyWork uses MySQL with the GORM library. MySQL is a popular relational database management system known for its reliability and scalability. The GORM library provides a convenient and efficient way to interact with MySQL databases, simplifying database operations and ensuring data integrity.

In terms of version control, EasyWork relies on Git, a distributed version control system. Git enables multiple developers to collaborate on the same project, tracking changes and resolving conflicts efficiently. Git's branching and merging capabilities are particularly useful for managing complex software development workflows.

To streamline the development process, EasyWork utilizes Visual Studio Code as its code editor. Visual Studio Code provides a range of features for writing, debugging, and deploying code, making it an ideal choice for developing the EasyWork platform.

Overall, EasyWork's proposed system aims to deliver a modern, scalable, and user-friendly service request platform that meets the needs of today's users. By leveraging Angular, GoLang, MySQL with the GORM library, Git, and Visual Studio Code, EasyWork seeks to set new standards for efficiency, reliability, and user satisfaction in the service request platform market.users to contribute their own resources, share study materials, and collaborate on creating new content, fostering a culture of knowledge sharing and collaboration among students.

## FEASIBILITY STUDY

The feasibility study for EasyWork assesses the technical, operational, and economic feasibility of implementing the platform.

**2.3.1 Technical Feasibility**

EasyWork's technical feasibility is high, as it leverages well-established technologies such as Angular, GoLang, MySQL, Git, and Visual Studio Code. These technologies are widely used and supported by a large community, ensuring compatibility, scalability, and reliability. Additionally, the use of these technologies enables EasyWork to meet the technical requirements of a modern service request platform, including responsive user interfaces, efficient backend services, and secure data storage.

**2.3.2 Operational Feasibility**

EasyWork's operational feasibility is also high, as the platform is designed to streamline service request processes and enhance user satisfaction. By providing a centralized platform for accessing a wide range of services, EasyWork aims to simplify the service procurement journey for users. Additionally, the platform's intuitive interfaces and transparent workflows ensure that users can easily navigate the platform and make informed decisions.

**2.3.4 Economic Feasibility**

From an economic standpoint, EasyWork's feasibility is supported by its potential to generate revenue through service bookings and partnerships with service providers. The platform's cost-effectiveness is further enhanced by its use of open-source technologies and cloud-based infrastructure, which reduce development and maintenance costs. Additionally, the platform's scalability and adaptability to changing market conditions ensure its long-term viability and profitability.

In conclusion, EasyWork's feasibility study demonstrates the platform's strong potential to deliver a modern, scalable, and economically viable service request platform. By leveraging well-established technologies and focusing on user-centric design principles, EasyWork aims to revolutionize the way users engage with service providers, setting new standards for efficiency, reliability, and user satisfaction in the digital age.

**CHAPTER 3: SYSTEM ANALYSIS AND DESIGN**

System analysis and design are pivotal stages in the development of Easywork, ensuring that the platform is meticulously crafted to meet user needs, optimize performance, and ensure scalability.

Requirement Specification: The requirement specification phase identifies and documents the functional and non-functional requirements of Easywork. These requirements serve as the guiding principles for subsequent design and development phases.

System Architecture: Easywork adopts a robust and scalable system architecture, following a microservices approach. A backend server, built using GoLang and Express.js, handles core functionalities like user authentication and service requests. MySQL, with the GORM library, serves as the primary database, while the frontend, developed with Angular, provides a responsive user interface.

User Interface Design: The user interface design prioritizes usability, accessibility, and aesthetics. Wireframes and mockups are created to visualize the layout and flow of the application. Chakra UI, a library of UI components, ensures consistency and coherence across the application, while responsive design principles ensure optimal usability across different devices and screen sizes.

Testing and Quality Assurance: Testing methodologies, including unit testing, integration testing, system testing, and user acceptance testing, are employed to validate the functionality and performance of Easywork. Continuous integration and deployment pipelines automate testing and deployment processes, ensuring rapid and reliable delivery of updates.

By following a systematic approach to system analysis and design, Easywork aims to deliver a robust, scalable, and user-friendly platform that meets the needs of users and service providers while ensuring optimal performance and reliability.

## FUNCTIONAL REQUIREMENTS

The requirement specification phase of EasyWork involves meticulously defining the functional and non-functional requirements to guide the development process. These requirements serve as the foundation for designing and implementing features that meet user needs and ensure the platform's success.

**3.1.1Functional Requirements:**

**User Authentication:**

Users must be able to create accounts securely, log in, and access personalized features. The authentication process should be smooth and intuitive, ensuring user privacy and security.

**Service Listing and Browsing:**

The platform should display a catalog of available services, categorized for easy navigation, allowing users to browse and explore options. Each service listing should include detailed information such as service description, pricing, and provider details.

**Service Requests:**

Users should be able to submit service requests, providing relevant details such as service type, location, and preferred time. The platform should support multiple service request formats, including one-time requests and recurring bookings.

Booking Management:

Users should be able to view and manage their bookings, including editing or canceling existing bookings. The platform should provide clear and user-friendly interfaces for managing bookings.

Service Provider Management:

Service providers should have profiles where they can manage their availability, services offered, and other relevant information. The platform should provide tools for service providers to update their profiles and manage their bookings.

**3.2 Non-functional Requirements:**

Performance:

The platform should be responsive, with fast loading times and minimal latency, to ensure a smooth user experience. Performance metrics should be monitored regularly, and optimizations should be implemented to improve performance as needed.

Scalability:

EasyWork should be designed to handle increasing user traffic and data volumes without compromising performance or reliability. The platform should be scalable both horizontally and vertically, allowing for seamless expansion as the user base grows.

Security:

The platform should implement robust security measures to protect user data and prevent unauthorized access. This includes encryption of sensitive information, secure authentication mechanisms, and regular security audits and updates.

**Accessibility:**

EasyWork should be accessible to users with disabilities, following accessibility guidelines and standards. This includes providing alternative text for images, keyboard navigation options, and other features that enhance accessibility for all users.

**Reliability:**

The platform should be reliable, with minimal downtime and disruptions. This requires implementing backup and recovery mechanisms, as well as monitoring tools to detect and address issues promptly.

**Usability:**

The platform should be user-friendly, with an intuitive interface and clear navigation paths. Usability testing should be conducted to gather feedback from users and make improvements to the user experience.

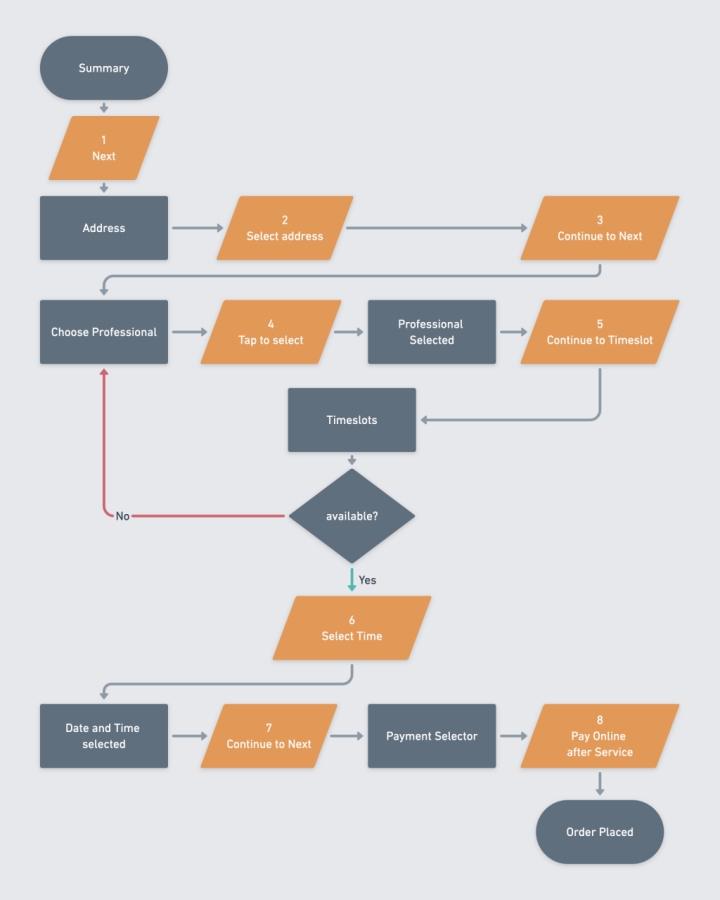
**Compatibility:**

EasyWork should be compatible with a wide range of devices and browsers, ensuring a consistent user experience across different platforms. Compatibility testing should be conducted to identify and address any compatibility issues.

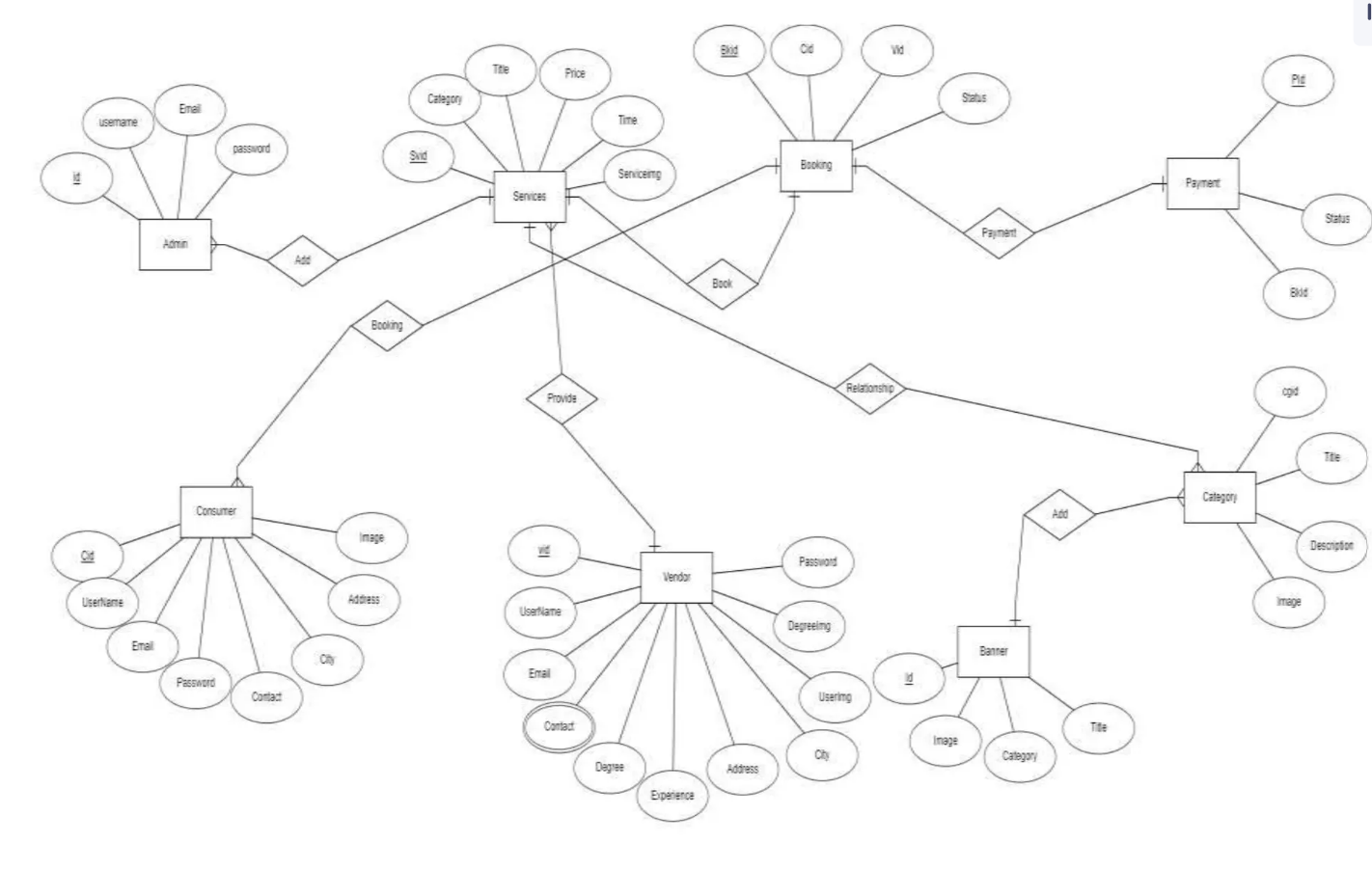
These requirements are essential for the successful development and deployment of EasyWork, ensuring that the platform meets user expectations and delivers a seamless

Service experience.

## FLOWCHART



**3.4.1** **Flow Chart**



**3.4.2** **Entity Relationship Diagram**

## DESIGN AND TEST STEPS / CRITERIA

Designing and testing EasyWork is a multifaceted process that requires careful planning, execution, and evaluation to ensure the platform meets its objectives effectively. This section outlines the comprehensive approach taken in designing the system components, defining test strategies, and establishing evaluation criteria to deliver a robust and user-friendly platform.

**34.1 Design Steps:**

**User Interface Design:**

**Objective:** Design intuitive and visually appealing user interfaces to enhance user experience.

**Approach:** Utilize wireframes, mockups, and prototypes to visualize interface layouts and interactions. Focus on consistency, clarity, and accessibility in design elements.

**Tools**: Use design tools like Figma, Sketch, or Adobe XD to create prototypes and iterate on design concepts.

**System Architecture Design:**

**Objective**: Define a scalable and maintainable system architecture that supports EasyWork's functionality and performance requirements.

**Approach:** Identify backend and frontend components, database structure, and integration points. Follow microservices architecture principles for modularity and scalability.

**Tools:** Utilize architectural design tools like Lucidchart or draw.io to create system architecture diagrams and document component interactions.

**Database Design:**

**Objective:** Design a database schema that ensures data integrity, efficiency, and scalability.

**Approach:** Use Entity-Relationship Diagrams (ERDs) to model entities, attributes, and relationships. Normalize the database structure to minimize redundancy and optimize query performance.

**Tools:** ERD design tools like dbdiagram.io or MySQL Workbench can be used to design and visualize database schemas.

**Component Design:**

**Objective:** Design individual software components and modules with a focus on reusability, modularity, and testability.

**Approach:** Break down system functionalities into smaller components. Use design patterns and coding best practices to implement clean and maintainable code.

**Tools:** IDEs (Integrated Development Environments) like Visual Studio Code or IntelliJ IDEA facilitate component design and development with features like code refactoring and debugging.

**3.4.2 Test Steps:**

**Unit Testing:**

**Objective:** Verify the functionality of individual components and functions in isolation.

**Approach:** Write unit tests for each component or function to validate its behavior. Utilize testing frameworks like Jest (for JavaScript) or JUnit (for Java) to automate test execution.

**Tools:** Testing frameworks and libraries like Jest, Mocha, or NUnit can be used for writing and executing unit tests.

Integration Testing:

**Objective:** Validate the interaction and integration between different system components.

Approach: Test the communication and data flow between frontend and backend components. Verify API interactions, data transformations, and error handling mechanisms.

**Tools:** Tools like Postman or Insomnia can be used for manual API testing, while frameworks like Supertest (for Node.js) or RestAssured (for Java) facilitate automated API testing.

**System Testing:**

**Objective:** Evaluate the overall system functionality and behavior under different usage scenarios.

**Approach:** Conduct end-to-end testing of user workflows and system processes. Test boundary cases, error conditions, and performance under load.

**Tools:** Testing frameworks like Selenium (for web applications) or Appium (for mobile applications) can be used for automated end-to-end testing.

**User Acceptance Testing (UAT):**

**Objective:** Involve users or stakeholders in testing the platform to validate its usability and suitability for their needs.

**Approach:** Provide users with access to the platform and gather feedback on their experiences. Incorporate user feedback to make improvements and enhancements.

**Tools:** Collaboration and feedback tools like UserTesting or BugHerd can be used to collect user feedback and track issues.

**3.4.3 Criteria for Evaluation:**

**Functionality:**

Ensure that all features and functionalities specified in the requirement specification are implemented correctly and perform as expected.

**Usability:**

Evaluate the user interface for ease of use, intuitiveness, and accessibility. Ensure that users can navigate the platform easily and complete tasks efficiently.

**Performance:**

Measure system performance in terms of response times, throughput, and resource utilization. Ensure that the platform can handle expected user loads and provide a responsive user experience.

**Reliability:**

Verify system reliability by testing error handling, recovery mechanisms, and fault tolerance. Ensure that the platform remains available and functional under different conditions.

**Security:**

Assess the platform's security posture by testing for vulnerabilities, implementing security best practices, and ensuring data protection.

By following these comprehensive design and test steps, along with the specified evaluation criteria, EasyWork can be developed, tested, and deployed with confidence, ensuring a high quality platform that meets user needs and expectations.

## ALGORITHM

**User Authentication Algorithm:**

**Objective:** Securely authenticate users and manage user sessions.

**Algorithm Steps:**

User enters login credentials (username/email and password).

Backend server verifies credentials against the database.

If credentials are valid, a session token is generated and stored in the user's browser.

User is authenticated and granted access to personalized features.

**Service Listing and Browsing Algorithm:**

**Objective:** Display a catalog of available services for users to browse and explore.

**Algorithm Steps:**

Retrieve service listings from the database.

Display service listings in a categorized format for easy navigation.

Allow users to filter and search for specific services based on criteria such as location, price, and ratings.

**Service Request Submission Algorithm:**

**Objective:** Enable users to submit service requests with relevant details.

**Algorithm Steps:**

User selects the desired service and provides necessary details (e.g., location, preferred time).

User submits the service request.

Backend server processes the request and notifies the relevant service providers.

**CRUD Operations Algorithm:**

Objective: Enable users and administrators to perform CRUD operations on relevant entities.

Algorithm Steps:

Identify the entity to be created, read, updated, or deleted.

Validate user permissions to perform the operation.

Execute the operation on the database and update the user interface accordingly.

**Testing and Quality Assurance Algorithm:**

**Objective**: Implement testing methodologies to validate the functionality and performance of EasyWork.

**Algorithm Steps:**

Write unit tests for individual components and functions.

Conduct integration tests to verify interactions between components.

Perform system tests to evaluate overall system functionality and behavior.

Engage users in user acceptance testing to gather feedback and improve the platform.

By following these algorithmic approaches, EasyWork can be developed with a structured and efficient design, ensuring that the platform meets user needs and performs optimally.

## TESTING PROCESS

**Unit Testing:**

**Objective**: Verify the functionality of individual components and functions in isolation.

**Approach**: Write unit tests for each component or function to validate its behavior. Utilize testing frameworks like Jest (for JavaScript) or JUnit (for Java) to automate test execution.

**Tools**: Jest, Mocha, Jasmine.

**Integration Testing:**

**Objective**: Validate the interaction and integration between different system components.

**Approach**: Test the communication and data flow between frontend and backend components. Verify API interactions, data transformations, and error handling mechanisms.

**Tools**: Supertest, Postman.

**System Testing:**

**Objective**: Evaluate the overall system functionality and behavior under different usage scenarios.

**Approach**: Conduct end-to-end testing of user workflows and system processes. Test boundary cases, error conditions, and performance under load.

**Tools**: Selenium, Cypress.

**User Acceptance Testing (UAT):**

**Objective**: Involve users or stakeholders in testing the platform to validate its usability and suitability for their needs.

**Approach**: Provide users with access to the platform and gather feedback on their experiences. Incorporate user feedback to make improvements and enhancements.

**Tools**: UserTesting, BugHerd.

**Test Cases:**

**User Authentication:**

**Test case:** Verify that users can log in with correct credentials.

**Test case:** Verify that users cannot log in with incorrect credentials.

**Test case:** Verify that users can reset their password.

**Service Listing and Browsing**:

**Test case:** Verify that services are displayed correctly in the catalog.

**Test case:** Verify that users can search for services by name.

**Test case:** Verify that users can filter services by category.

**Service Requests:**

**Test case:** Verify that users can submit service requests with valid details.

**Test case:** Verify that users cannot submit requests with invalid details.

**Test case:** Verify that users receive confirmation of their request.

**CRUD Operations:**

**Test case:** Verify that users can create new service listings.

**Test case:** Verify that users can read existing service listings.

**Test case:** Verify that users can update service listings.

**Test case:** Verify that users can delete service listings.

**Search and Filter Functionality:**

**Test case:** Verify that users can search for services by keyword.

**Test case:** Verify that users can filter services by category.

**Test case:** Verify that users can sort services by price or rating.

**Performance Testing:**

**Test case:** Verify that the system responds quickly to user actions.

**Test case:** Verify that the system can handle a large number of simultaneous users.

**Test case:** Verify that the system performs well under heavy load.

**Security Testing:**

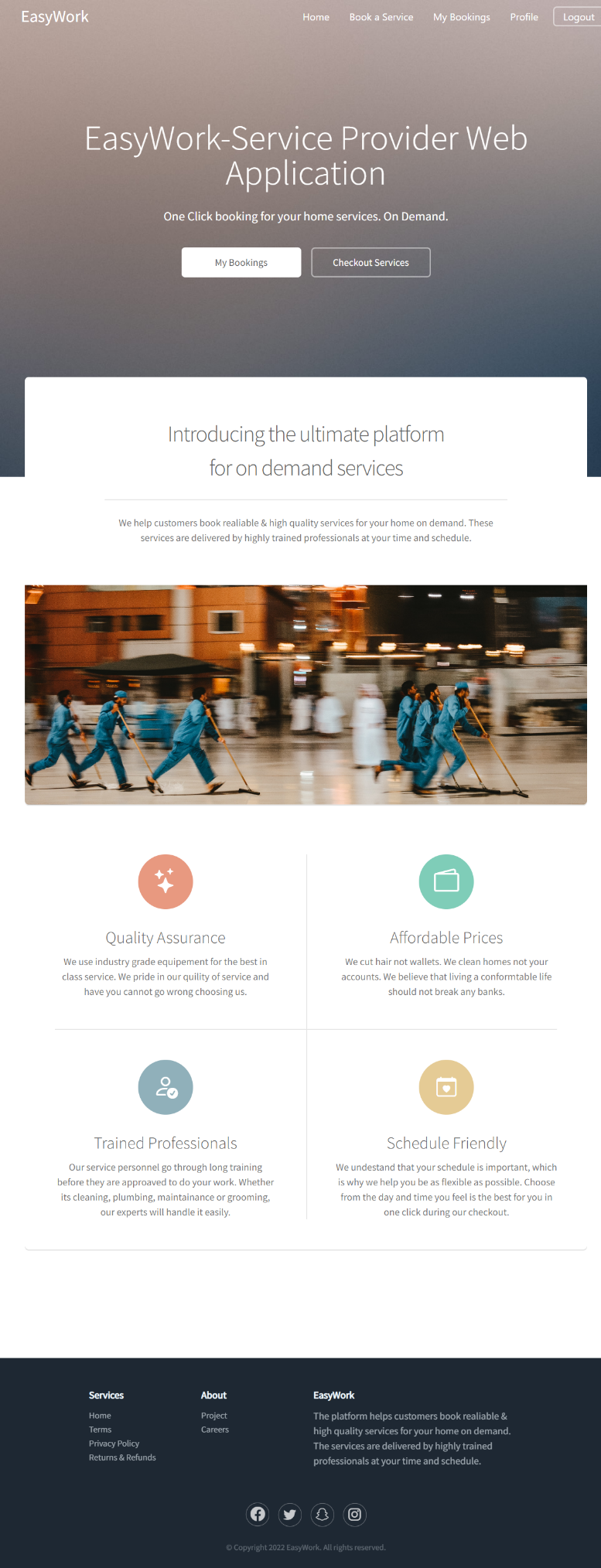
**Test case:** Verify that user data is stored securely.

**Test case:** Verify that the system is protected against common security threats such as SQL injection and cross-site scripting (XSS).

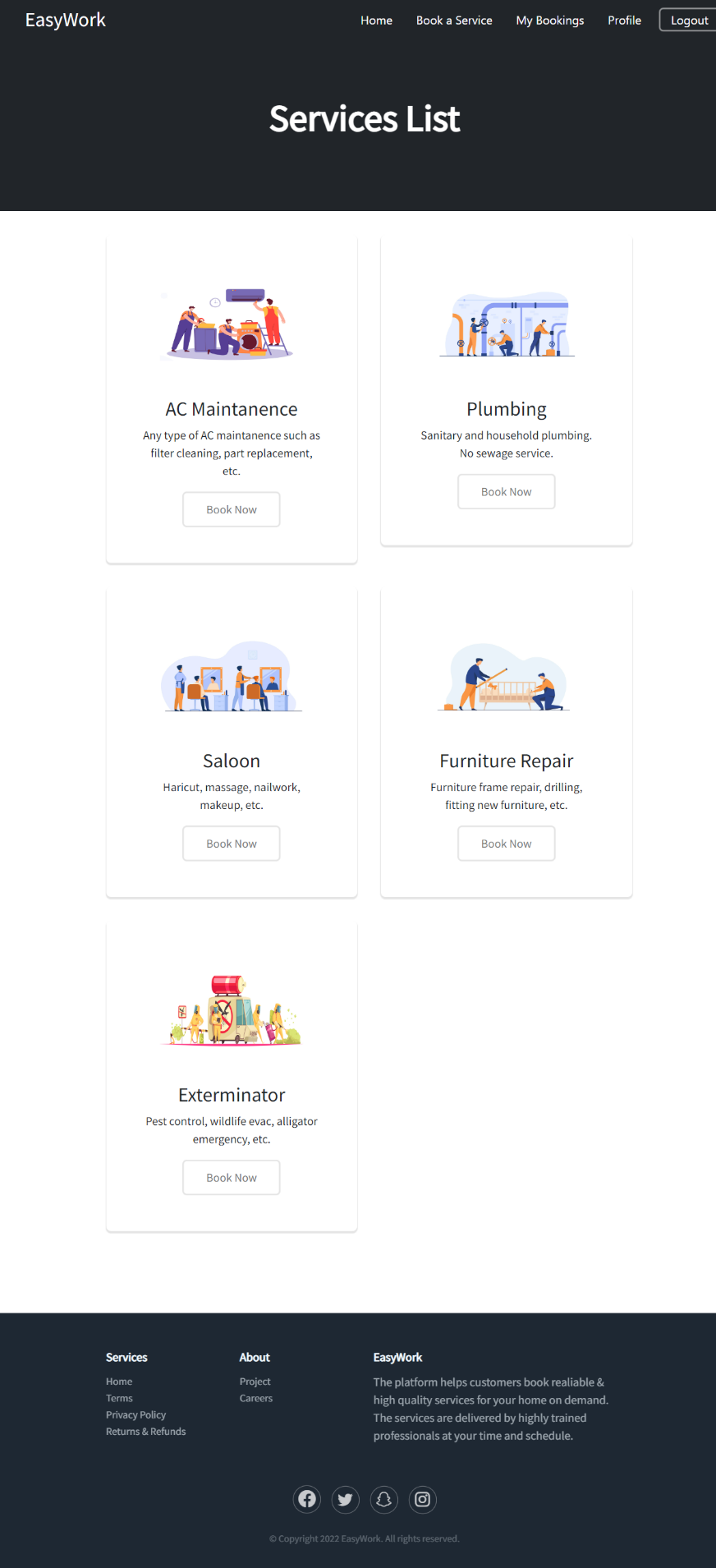
By following these test cases and testing processes, Easywork can be thoroughly tested to ensure its

functionality, reliability, and performance before deployment.

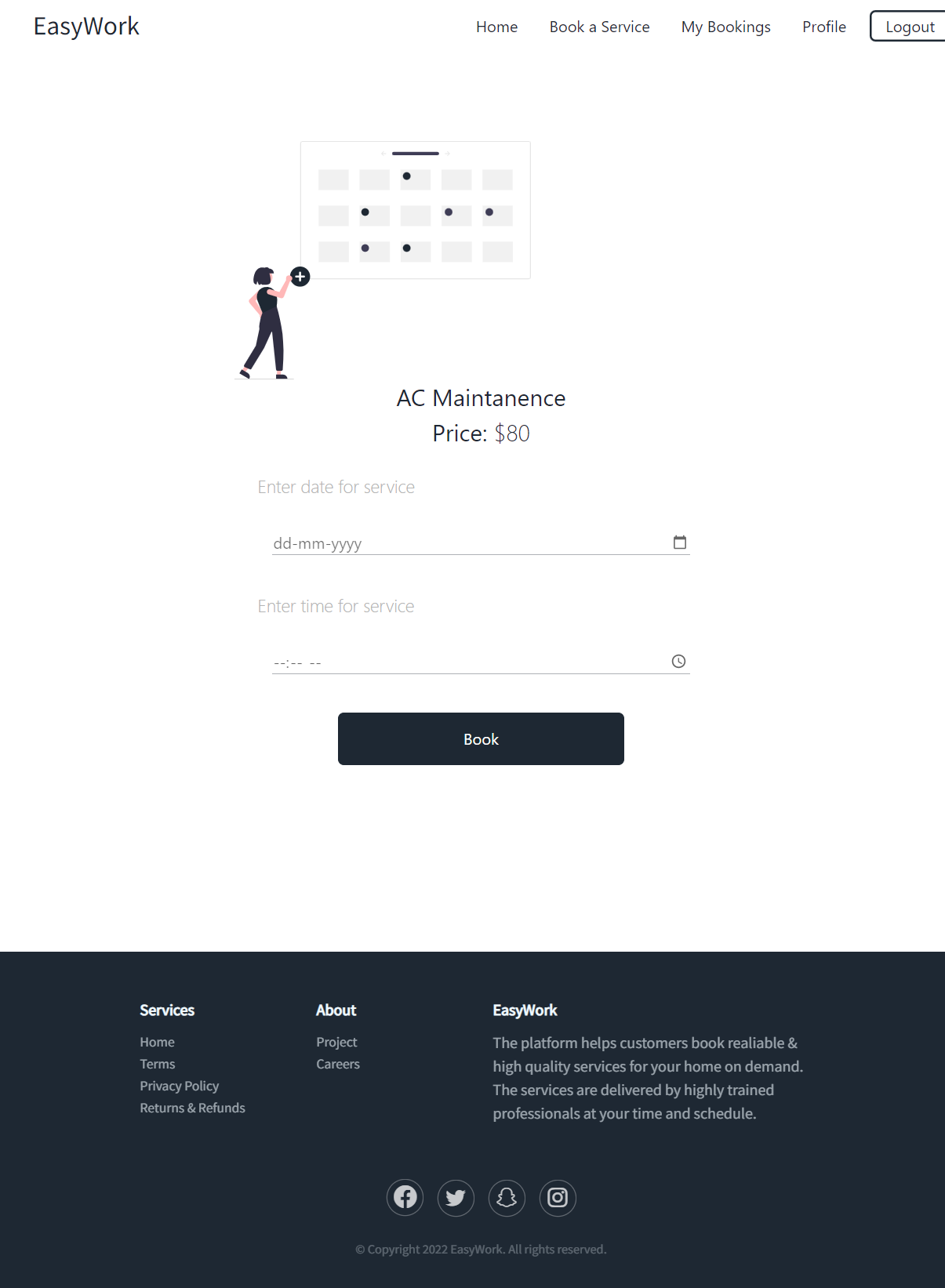
# CHAPTER 4: RESULTS / OUTPUTS



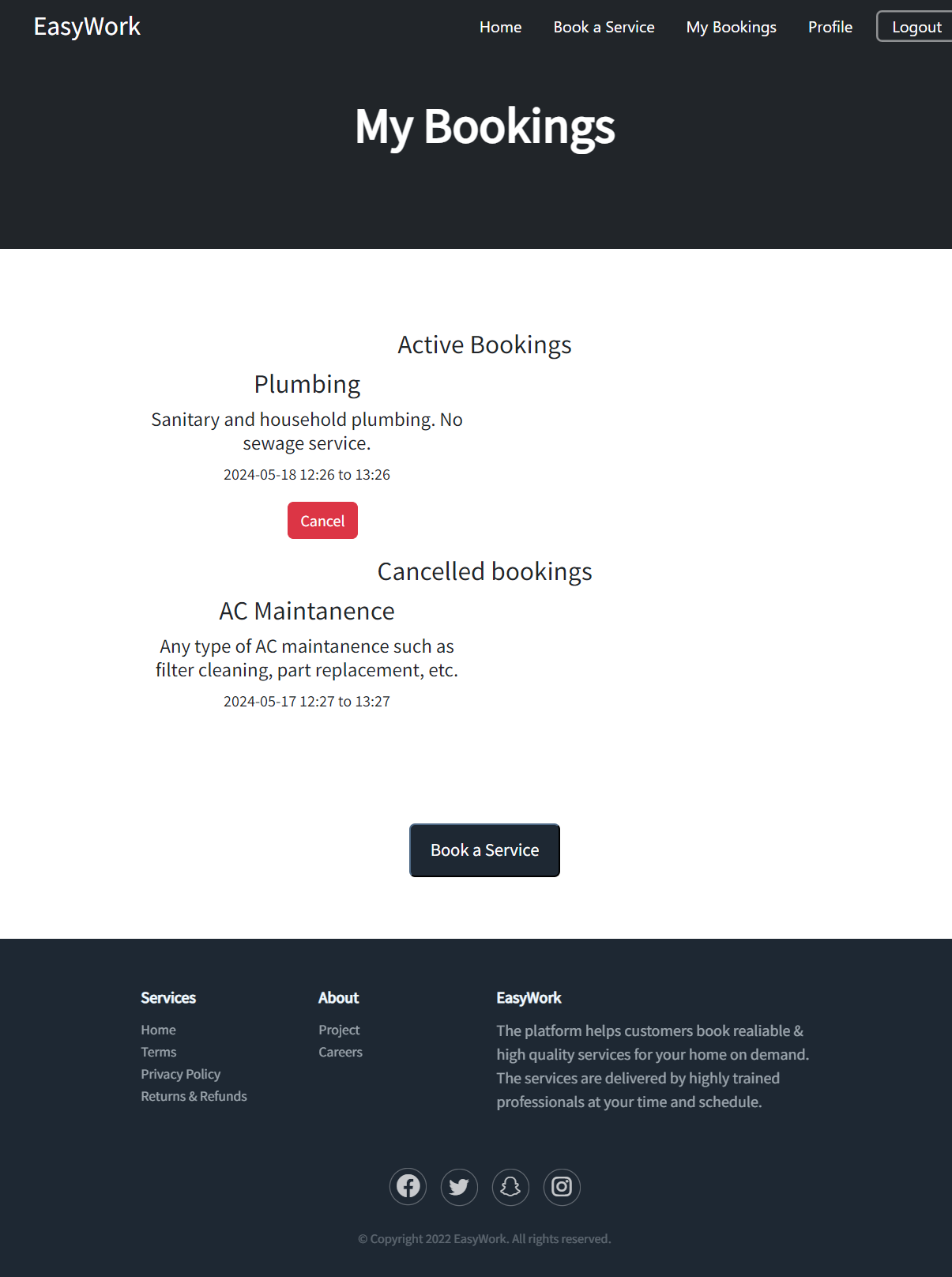
**Figure – 4.1 (Homepage 1)**



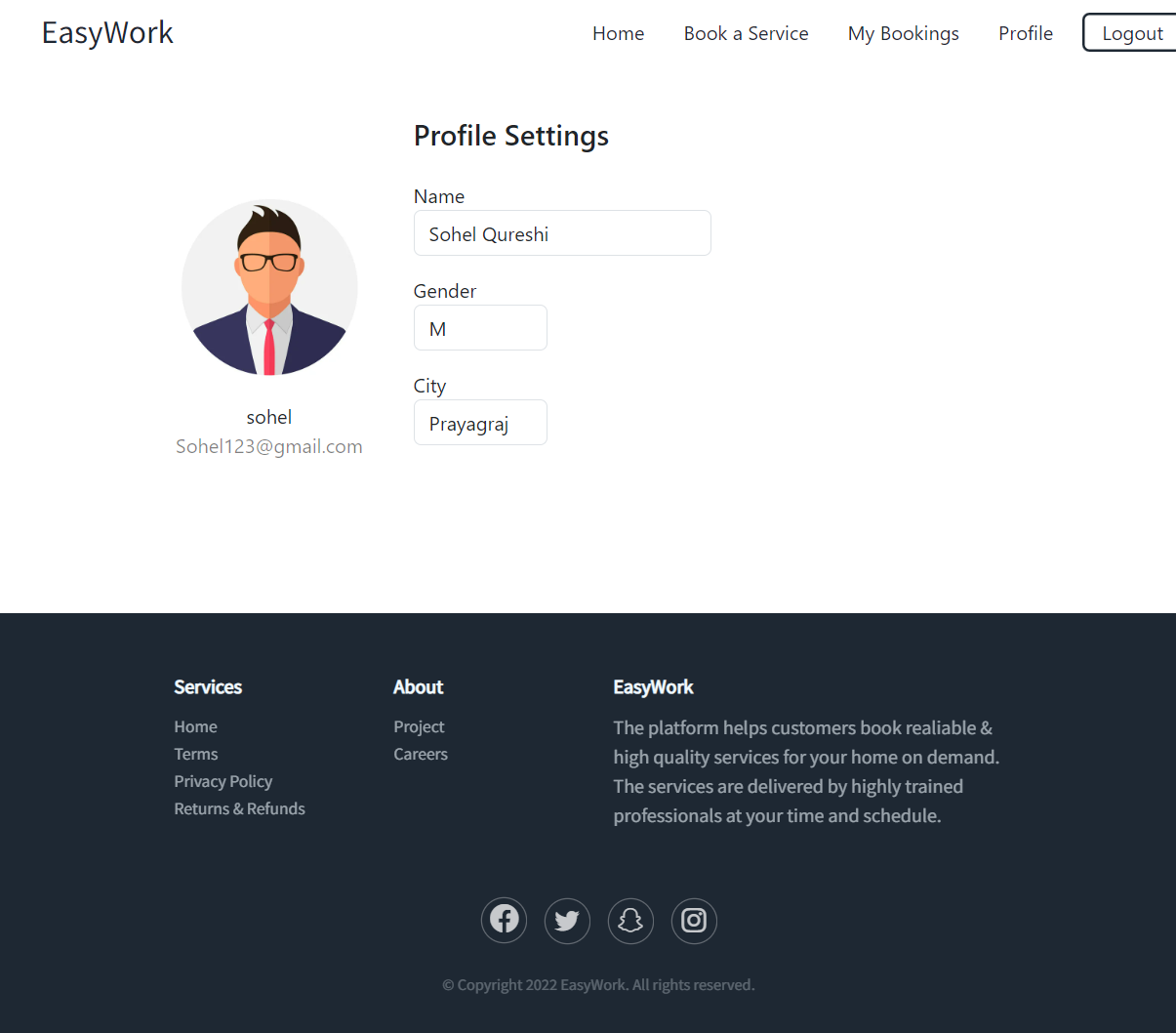
**Figure – 4.2 ( Services )**



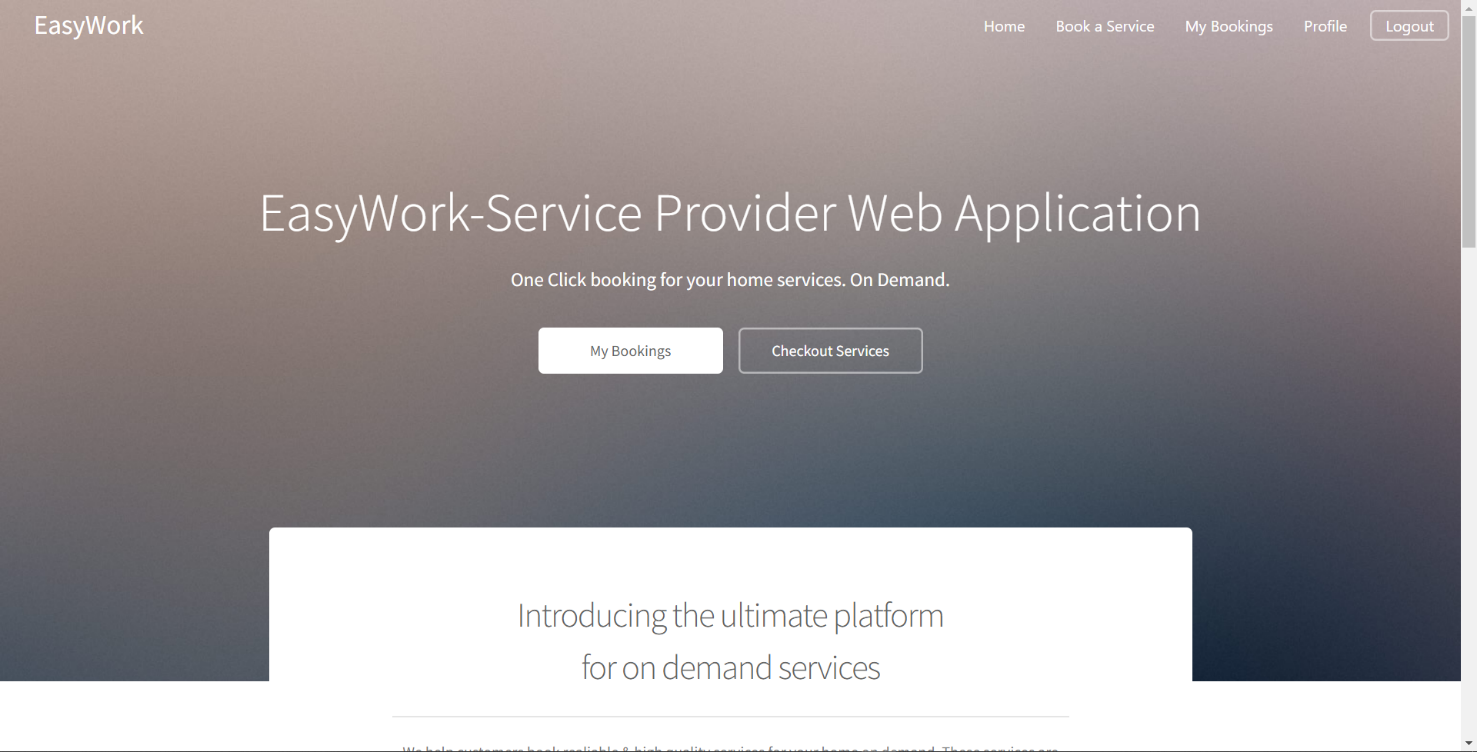
**Figure – 4.3 ( Schedule)**



**Figure – 4.4 ( Bookings )**



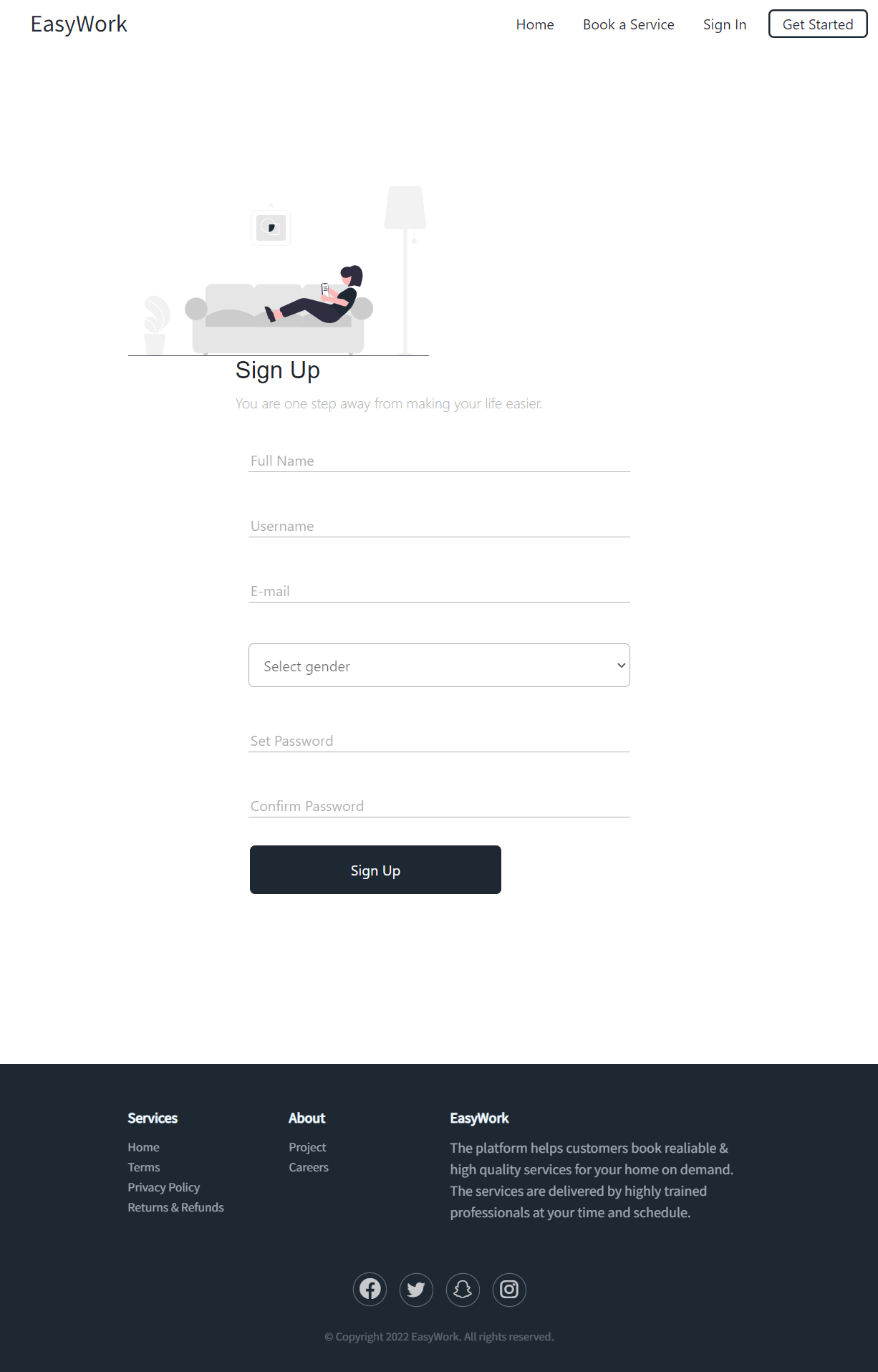
**Figure – 4.5 ( Profile )**



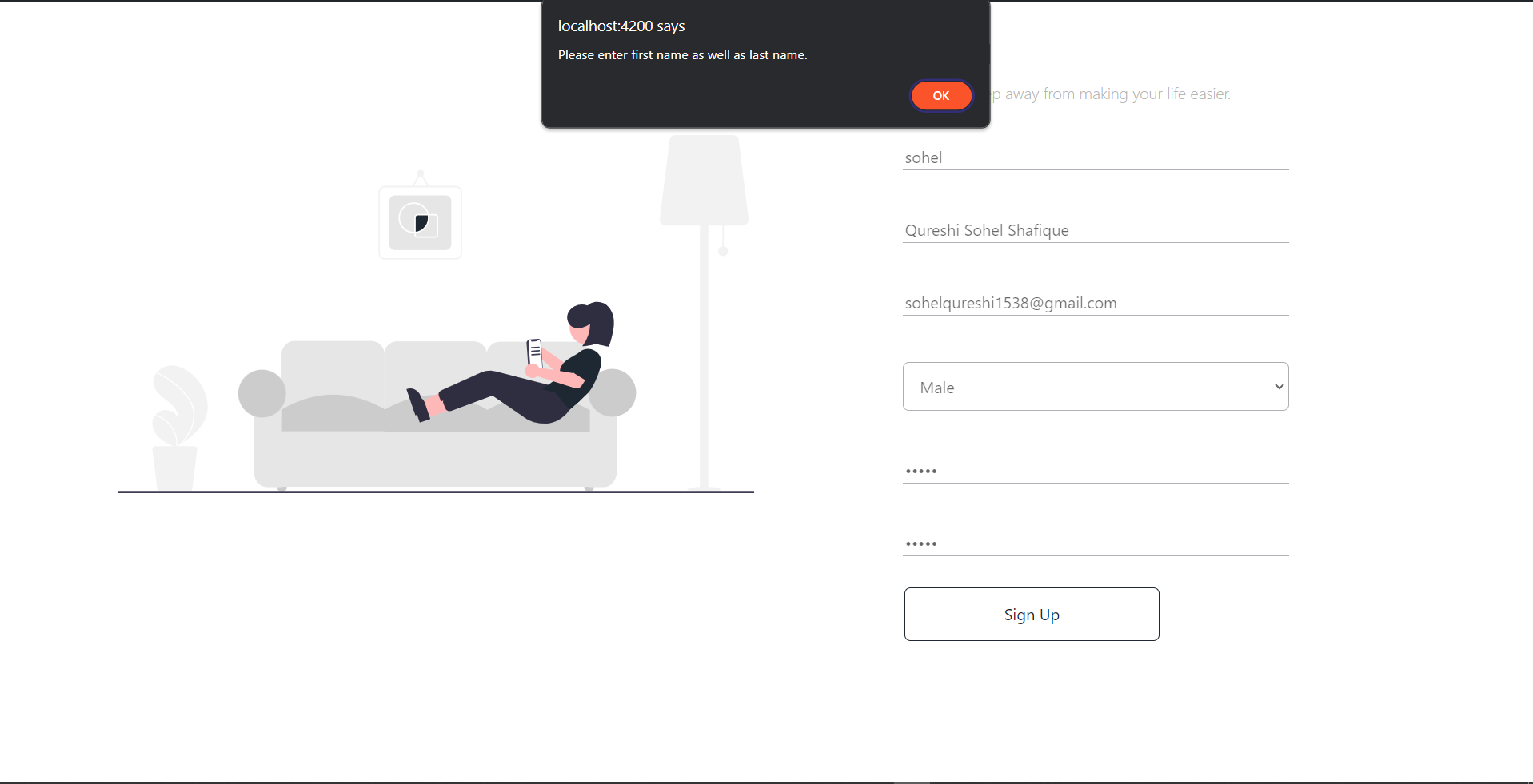
**Figure – 4.5 (Adaptive Components)**

# 

**Figure – 4.6 ( Sign In )**



**Figure – 4.7 (Sign Up)**



**Figure – 4.8 ( Validation )**

# CONCLUSIONS

The development of Easywork, a modern service request platform, has been a journey marked by innovation, collaboration, and a relentless pursuit of excellence. Through the integration of cutting-edge technologies and user-centric design principles, Easywork aims to redefine the way users engage with service providers, setting new standards for efficiency, reliability, and convenience in the digital age.

One of the key strengths of Easywork lies in its technology stack, which includes Angular for the frontend, GoLang for the backend, and MySQL with the GORM library for the database. This combination of technologies provides a solid foundation for a scalable and responsive platform that can meet the evolving needs of users and service providers.

The project began with a comprehensive analysis of existing service request platforms, which helped identify key challenges and opportunities in the market. This analysis informed the design and development of Easywork, ensuring that the platform would address these challenges while providing a seamless and intuitive user experience.

The system analysis and design phase of Easywork focused on defining clear requirements, designing a scalable architecture, and creating an intuitive user interface. This phase was crucial in laying the groundwork for the development of Easywork, ensuring that the platform would meet the needs of its users and service providers.

The testing process played a vital role in ensuring the quality and reliability of Easywork. Through rigorous testing methodologies, including unit testing, integration testing, system testing, and user acceptance testing, Easywork was able to identify and resolve issues early in the development process, ensuring a smooth and seamless user experience.

Looking ahead, Easywork is poised to revolutionize the way users engage with service providers. With its innovative features, user-friendly interface, and robust performance, Easywork is set to become the go-to platform for all utility needs. As Easywork continues to evolve, it will remain committed to its core principles of simplicity, efficiency, and technological advancement, ensuring that users can access reliable services efficiently and conveniently.

In conclusion, Easywork represents a paradigm shift in the realm of service request platforms. By leveraging innovative technologies and user-centric design principles, Easywork aims to redefine user experiences and set new standards for service request platforms. As Easywork continues to grow and evolve, it will remain committed to delivering a platform that meets the needs of its users and service

providers, ensuring a seamless and efficient service request process for all.

# REFERENCES

Urban Company. (n.d.). Retrieved from https://www.urbancompany.com/

TaskRabbit. (n.d.). Retrieved from <https://www.taskrabbit.com/>

Angular. (n.d.). Retrieved from https://angular.io/

Supertest. (n.d.). Retrieved from https://github.com/visionmedia/supertest

Postman. (n.d.). Retrieved from https://www.postman.com/

Cypress. (n.d.). Retrieved from https://www.cypress.io/

UserTesting. (n.d.). Retrieved from https://www.usertesting.com/

BugHerd. (n.d.). Retrieved from <https://www.bugherd.com/>

MySQL. (n.d.). Retrieved from https://www.mysql.com/

GORM. (n.d.). Retrieved from <https://gorm.io/>

Visual Studio Code. (n.d.). Retrieved from https://code.visualstudio.com/

Figma. (n.d.). Retrieved from https://www.figma.com/

These references provide valuable insights and resources that have contributed to the development and testing of Easywork.

**PLAGRISM REPORT**



