**SEVAK**

Submitted by

Priyesh Pachauri

O23MCA110246

For the award of the degree of

MCA

**Session: 2024-2025**

**Declaration**

We hereby declare that the work which is being presented in the MCA. Project **“SEVAK”**, in partial fulfillment of the requirements for the award of the **Masters of Computer Applications** and submitted to the Department of Computer Engineering and Applications of chandigarh university, is an authentic record of our own work carried under the supervision of **Department of Computer Engineering.**

The contents of this project report, in full or in parts, have not been submitted to any other institute or university for the award of any degree.

**Certificate**

#### This is to certify that the following students have carried out the project titled “Sevak” from 1st January 2025 to April 2025 for the award of Master of Computer Application from chandigarh university under my supervision. The report embodies result of original work and studies carried out by the students themselves and the contents of the report do not form the basis for the award of any other degree to the candidates or to anybody else.

Priyesh pachauri

O23MCA110246

Date:

**Acknowledgment**

The satisfaction which accompanies the successful completion of the project is incomplete without the mention of a few names. We take this opportunity to acknowledge the efforts of the many individuals who helped us to make this project possible.

Firstly, we would like to express our heartfelt appreciation and gratitude to our project guide **Department of Computer Engineering.** His vision and execution aimed at creating a structure, definition and realism around the project and fostered the ideal environment for us to learn and do. This project is a result of his teaching, encouragement and inputs in the numerous meetings he had with us, despite his busy schedule.

The experience was novel one and we would like to thank all the people, who have let their valuable time for the completion of the report. Without their consideration it would have been difficult to complete the report.

**Abstract**

**Sevak: A Full-Stack Service Management Platform**

Sevak is a comprehensive web application designed to bridge the gap between clients and skilled service professionals, offering seamless access to household services such as plumbing, electrical repairs, cleaning, and more. Built as a full-stack solution, Sevak leverages modern web technologies to provide an efficient, scalable, and location-aware service management platform.

The system comprises two primary modules:

1. **Admin Module**:
   * Enables administrators to manage professionals and clients efficiently.
   * Features include user verification, service monitoring, booking oversight, and analytics.
   * Provides tools for status management (activation/deactivation), review moderation, and payment tracking.
2. **User Module**:
   * **Clients**: Browse services, book professionals, track appointments, and provide feedback.
   * **Professionals**: Create profiles, list services, manage bookings, and receive payments.

Sevak enhances service accessibility through real-time search, GPS-based professional recommendations, and secure payment integration. Built with security, scalability, and user experience in mind, the platform demonstrates the practical application of full-stack development in solving real-world service management challenges. Future enhancements could include AI-driven service matching, IoT integration for smart home services, and mobile app extensions.

**Key Technologies**: PHP, MySQL, JavaScript, HTML/CSS, AJAX, RESTful APIs.

**Impact**: Simplifies service discovery, improves professional-client interactions, and optimizes service delivery in urban and semi-urban areas.

**TABLE OF CONTENTS**

|  |  |
| --- | --- |
| **TOPIC** | **PAGE NO.** |
| 1. Introduction | 7 |
| 1.1 Purpose | 7 |
| 1.2 Scope | 7 |
| 1.3 Terms, Definitions, Acronyms and Abbreviations | 7 |
| 1.4 Objective | 7 |
| 2. General Description | 8 |
| 2.1 Product Perspective | 8 |
| 2.2. 2.2 Product Function | 8 |
| 2.3 Hardware Interface | 9 |
| 2.4 Software Interface | 9 |
| 2.5 User Characteristic | 9 |
| 2.6 General Constraints | 10 |
| 2.7 Tools and Technologies | 15 |
| 3. Specific Requirements | 16 |
| 3.1 Functional Requirements | 17 |
| 3.2 Non-Functional Requirements | 18 |
| 3.2.1 Availability | 18 |
| 3.2.2 Security | 18 |
| 3.2.3 Reliability | 18 |
| 3.2.4 Portability | 19 |
| 3.2.5 Maintainability | 19 |
| 4. System Architectural Design | 20 |
| 4.1 Detailed Description of Components | 20 |
| 4.1.1 Sequence Diagram | 20 |
| 4.1.2 Activity Diagram | 23 |
| 4.1.3 Data Flow Diagram | 26 |
| 4.1.4 Entity-Relation Diagram | 29 |
| 5. Implementation & User Interfaces | 31 |
| 6. Software Testing | 36 |
| 6.1 Testing | 36 |
| 6.2 Objectives of Software testing | 36 |
| 6.3 Principles of Software testing | 36 |
| 6.4 Testing Fundamentals | 37 |
| 6.5 Testing Information | 37 |
| 7. Conclusion | 38 |
| 8. References | 39 |

### Introduction

The requirements for creating a web-based platform that links clients with service providers providing a range of fundamental services, including plumbing, electrical repairs, house cleaning, and other domestic services, are described in this Software Requirements Specification (SRS) document. The website seeks to offer a smooth, intuitive user experience so that people can quickly schedule services and locate trustworthy experts in their neighborhoods.

#### Purpose

This platform's goal is to provide consumers with a one-stop shop for a range of home-related services. By offering a consolidated platform where consumers can peruse service categories, read reviews, schedule appointments, and pay for services all in one location, it seeks to streamline the recruiting process for qualified personnel.

#### Scope

Features that include service provider registration, client accounts, scheduling, payment processing, ratings, reviews, and search and filtering options for various service categories are all included in the website's scope. Several service categories will be supported by the platform.

*Target Audience:* Anyone in need of basic household services, including homeowners and renters, is the website's primary demographic. Because the platform is meant to be used by people with varying levels of technological expertise, simplicity, and usability will be given top priority in both design and features. Additionally, service providers can sign up on the platform to market their services to prospective clients.

### Terms, Definitions, Acronyms, and Abbreviations

* ***User***: An individual looking to book household services.
* ***Service Provider***: A professional offering household services.
* ***Admin***: Manages users, providers, bookings, and payments.
* ***UI/UX***: User Interface/User Experience.

#### Objectives

* To offer an extensive list of reliable service providers across various industries to make it simple for clients to search, make reservations, and pay.
* To set up a trustworthy evaluation and rating system to ensure the quality of services.
* To enable service providers to handle reservations, client communications, and payments effectively.

### General Description

The platform is a web-based service designed to connect users with professionals who offer essential household services, such as electrical repairs, plumbing, cleaning, and general home maintenance. With an intuitive user interface, the platform simplifies the process of finding, booking, and paying for services, catering to both customers and service providers.

### Product Perspective

This website will act as a centralized hub, offering various service categories where customers can browse and select from a pool of qualified professionals in their area. It will be optimized for both desktop and mobile devices, ensuring accessibility across different platforms. Additionally, the platform will support multiple payment methods for seamless transactions and feature a rating and review system to maintain service quality.

Sevak is an innovative online platform that connects users with trusted, vetted house help and service providers. With a focus on convenience and reliability, Sevak offers a wide array of services that include domestic help, cleaning, cooking, and other essential household services. The website is designed with user-friendliness in mind, allowing customers to easily browse available services, select service providers based on ratings, reviews, and specific needs, and schedule appointments at their convenience.

One of the key differentiators of Sevak is its stringent vetting process for all service providers. Background checks and skill assessments ensure that customers can trust the professionals who enter their homes. Furthermore, Sevak offers a transparent pricing structure with no hidden fees, making it clear what the customer is paying for upfront. The platform supports multiple payment options and provides a smooth experience for both customers and service providers

### Product Functions

Key features of the platform include:

* ***Customer Registration & Profile Management*:** Users can create accounts, manage their profiles, and track their service history.
* ***Service Provider Registration & Profile Management:*** Professionals can sign up, build profiles, and showcase their skills, experience, and availability.
* ***Search & Filter Options*:** Users can search for specific services (e.g., electricians, plumbers, housekeepers) based on location, service type, and availability.
* ***Booking & Scheduling*:** Customers can select a service provider and schedule appointments based on availability.
* ***Payment Integration*:** Secure payment processing with multiple payment options ensures smooth financial transactions.
* ***Ratings & Reviews*:** Customers can rate service providers and leave feedback after a service, assisting future users in making informed choices.
* ***Communication System*:** A built-in messaging or notification system facilitates communication between customers and service providers for service details, time adjustments, or clarifications.
* ***Admin Panel*:** A robust administrative backend allows platform managers to oversee users, service providers, transactions, and overall operations efficiently.
  1. Hardware Interfaces

*Server-side*

* + *Processor:* Intel Dual Core(i3) or more
  + *Processor Speed:* v1.3 GHZ or more
  + *RAM*: 8 GB
  + *Hard Disk:* 20 GB of free space

*Client-side*

* + *Processor:* Octa Core Max 2.01Ghz or more
  + *Processor Speed :* v1.3 GHZ or more
  + *RAM*: 4 GB
  + *Hard Disk:* 512MB of free space
  1. Software Interfaces

*Server-side*

* + *OS:* Windows Server 8 or onwards
  + *Webserver:* Xampp Server

*Client-side*

* + *OS:* Windows Server 8 or onwards
  + *Browser:* Any browser compatible with IE 5.0 or onwards

### User Characteristics

The platform will serve two primary user groups:

* **Customers**: Homeowners, renters, or anyone in need of basic services for their home. These users may not have technical expertise and will require an intuitive, simple interface for easy navigation.
* **Service Providers**: Professionals offering household services. They should have basic digital literacy to manage their profiles, bookings, and payments via the platform.

Additionally, administrators will have access to manage the entire platform, ensuring smooth operation and handling customer and provider support issues.

### General Constraints

#### **Functional Constraints**

* The website must allow users to register/log in.
* Users must be able to search for services by category and location.
* Secure payment gateway integration is required.
* Real-time booking and scheduling must be supported.
* Service providers must be verified before listing their services.

Use Case

*Use Case 1- Registration*

|  |  |  |
| --- | --- | --- |
| **Serial no:** | **Field name** | **Description of Field name** |
| 1 | Use case no | 1 |
| 2 | Registration | This use case document defines the procedure for registering into the system |
| 3 | Actors | Professional, Service requester |
| 4 | Pre-condition | None |
| 5 | Post-condition | If the use case is successful then the actor will successfully be able to register into the system otherwise the system state will remain unchanged. |
| 6 | Flow of events | *Basic flow:* This use case starts when the actor wishes to register into the system. The user selects the role. The system requests that the actor enter their details. The actor enters the required details. The system validates the details entered. If the details are validated, the user is directed to the login page. The use case ends  *Alternate flow*: None |
| 7 | Special requirements | None |
| 8 | Related use cases | None |

*Use Case 2- Login*

|  |  |  |
| --- | --- | --- |
| **Serial no:** | **Field name** | **Description of Field name** |
| 1 | Use case no | 2 |
| 2 | Registration | This use case document defines the procedure for Logging into the system |
| 3 | Actors | Professional, Service requester |
| 4 | Pre-condition | Actors should be Registered. |
| 5 | Post-condition | If the use case is successful then the actor will successfully be able to book the professional for a certain amount of time otherwise the system state will remain unchanged |
| 6 | Flow of events | *Basic flow:* This use case starts when the actor wishes to log in to the system. · The user selects the role. · The system requests that the actor enter their email ID and password. · The actor enters the required detail. · The system validates the details entered. If the details are validated, the user is directed to the corresponding page. · The use case ends.  *Alternate flow:* None |
| 7 | Special requirements | None |
| 8 | Related use cases | Registration, Login |

*Use Case 3- Services*

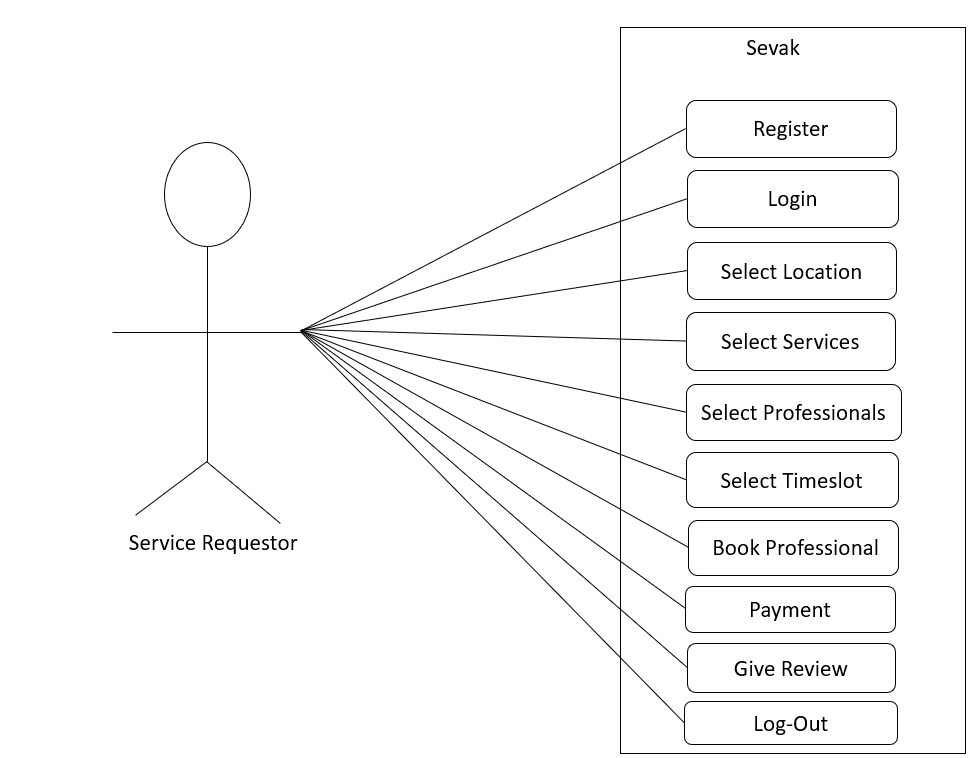
|  |  |  |
| --- | --- | --- |
| **Serial no:** | **Field name** | **Description of Field name** |
| 1 | Use case no | 3 |
| 2 | Registration | This use case document defines the procedure for updating a service. |
| 3 | Actors | Admin |
| 4 | Pre-condition | It is a must for the admin to log in to the system. |
| 5 | Post-condition | If the use case is successful then the actor will successfully be able to update the services in the system otherwise the system state will remain unchanged. |
| 6 | Flow of events | Basic flow- · This use case starts when the actor wishes to update the services. The admin updates the services and modifies the related profession performing these services. The use case ends.  Alternate flow: None |
| 7 | Special requirements | None |
| 8 | Related use cases | Registration, Login, Booking |

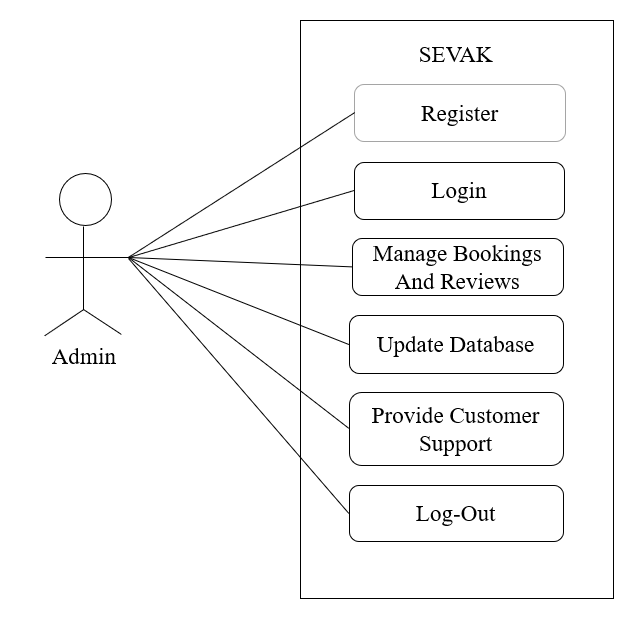
Use Case 4- Booking

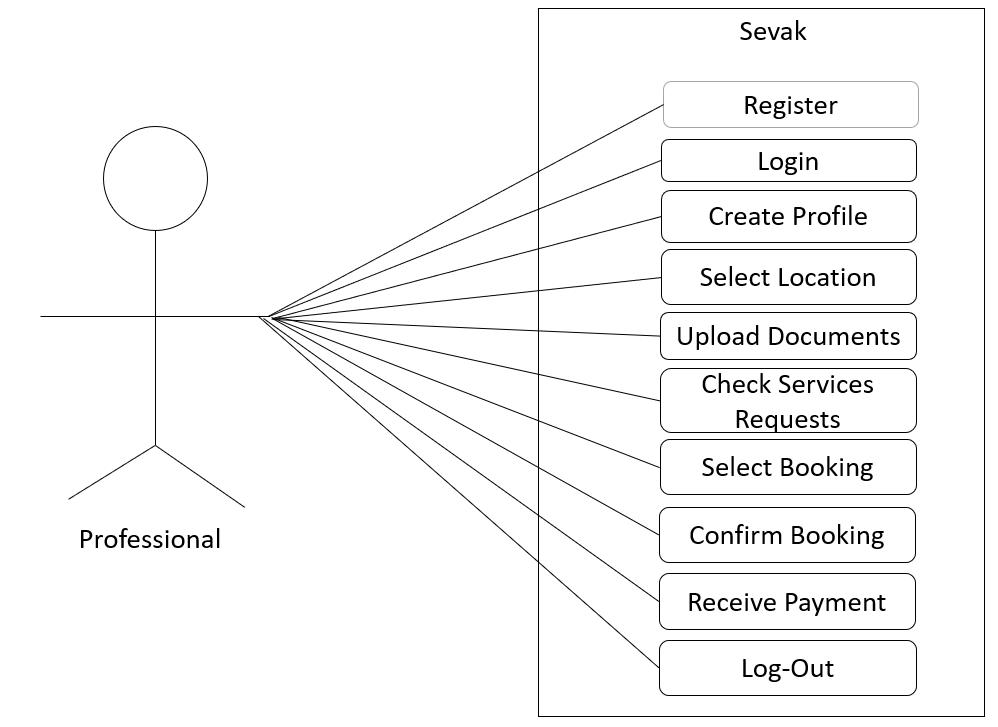
|  |  |  |
| --- | --- | --- |
| **Serial no:** | **Field name** | **Description of Field name** |
| 1 | Use case no | 4 |
| 2 | Registration | This use case document defines the procedure for booking services |
| 3 | Actors | Admin, Professional, Service requester |
| 4 | Pre-condition | Actors should be Registered. |
| 5 | Post-condition | If the use case is successful then the actor will successfully be able to register into the system otherwise the system state will remain unchanged. |
| 6 | Flow of events | *Basic flow-* This use case starts when the actor wishes to book the professional. · The user selects the services and then selects professionals from the particular field. Then check if the professional is available for that time or not if yes then proceed with payment and the booking is confirmed. And if not then select other professionals. The use case ends.  *Alternate flow*: None |
| 7 | Special requirements | None |
| 8 | Related use cases | None |

*Use Case 5-Logout*

|  |  |  |
| --- | --- | --- |
| **Serial no:** | **Field name** | **Description of Field name** |
| 1 | Use case no | 5 |
| 2 | Registration | This use case document defines the procedure for logging out of the system |
| 3 | Actors | Admin, Professional, Service requester |
| 4 | Pre-condition | The actor must log in to the system. |
| 5 | Post-condition | If the use case is successful then the actor is logged out from the system otherwise the system state is unchanged |
| 6 | Flow of events | Basic flow- This use case starts when the actor wishes to log out from the system. The actor requests to log in from the system. The actor is logged out of the system. The use case ends.  Alternate flow: None |
| 7 | Special requirements | None |
| 8 | Related use cases | Login |







#### **Non-Functional Constraints**

* **Performance**: The website must load within 3 seconds on standard connections.
* **Security**: Secure user authentication and encrypted payments.
* **Scalability**: Should handle at least 100,000 users without performance issues.
* **Usability**: Must have a responsive design for mobile and desktop users.

#### **Technical Constraints**

* Must be built using modern web technologies (e.g., React, Node.js, PHP, MySQL).
* Should support third-party API integrations (e.g., Google Maps for location services).
* Should be hosted on a cloud-based infrastructure (AWS, Azure, or similar).

#### **Legal and Compliance Constraints**

* Must comply with data protection laws (GDPR, CCPA).
* Terms and conditions must be clearly stated for both users and service providers
  1. Tools & Technologies
* Front End: HTML5, CSS3, JavaScript
* Back End: Php
* Database: MySQL 8.0 Design
* Tool: Visual Studio code
* Web Application Server: Apache ServeTop of Formr

Bottom of Form

## Specific Requirements

### User Module

### The User-Interface is used to access the user’s personal information. These data are stored in the form of tables using the MYSQL database. The user needs to enter their username and password.

### Validation and verification are performed at the back end. Assuming the given accreditations are right the administrator effectively login to the framework.

### Users should be able to register using an email, or phone number.

### Login authentication via OTP or password-based system.

### Users can update their profile details and contact information.

### User module is further divided into the professional module and the service requestor module.

### Professional Module

### The service requester must complete a verification process before choosing services.

### Service requester can choose from service listings with descriptions, pricing, and availability.

### Service requesters can pay using different methods.

### Service requester can search for services using location-based filtering.

### Service requester can provide ratings and reviews for completed services.

### Admin Module

* Admin can approve or reject service provider registrations.
* Admin can manage user accounts and block fraudulent users or providers.
* Admin can view and manage all transactions occurring on the platform.
* Admin can generate reports for bookings, earnings, and user activity.
* Admin can modify service categories and pricing guidelines.

*Service requester Module*

* + *The service requester* must complete a verification process before choosing services.
  + Service requester can choose from service listings with descriptions, pricing, and availability.
  + Service requesters can pay using different methods.
  + *Service requester* can search for services using location-based filtering.
  + Service requester can provide ratings and reviews for completed services.

*Requests Module*

* + Service requester can request the services of respective professionals.
  + Service requester can request for more than one service depending upon the availability of the professional.

*Bookings*

* Professional details, date, and time will be shown in this module after confirmed booking.
* Notifications are provided when the booking is confirmed.

### Payment Module

* Integrated with secure payment gateways (PayPal, Stripe, Razor-pay, etc.).
* Multiple payment options including credit/debit cards, UPI, and digital wallets.
* Automatic invoicing and receipts for each transaction.
* Secure refund processing in case of service cancellation.
* Users can save payment details for faster checkouts.

### Notification System

* Email and SMS notifications for booking confirmations, cancellations, and payment transactions.
* Reminder notifications for upcoming service bookings.
* Promotional notifications for discounts and special offers.

## Functional Requirements

### User Module

* User registration/login/logout.
* Registration and verification.

### Service Requestor Module

* Profile Management.
* Search and filter services.
* Book services and track status.
* Provide ratings and reviews.
* Booking management dashboard.

### Professional Module

* Service listing management.
* Booking management dashboard.
* Receive payments and track earnings.

### Admin Module

* Manage users and service providers.
* Approve or reject service providers.
* Monitor transactions and bookings.
* Generate reports.

### Payment Module

* Secure payment processing.
* Multiple payment options (Credit/Debit cards, UPI, Wallets).
* Transaction history and confirmation notification.

### Requests module

* Booking requests.
* Pending requests.

### Booking module

* Confirm Booking requests.
* Booking id and professional details
  1. Nonfunctional Requirements (Software System Attributes)

#### AVAILABILITY

* 99% uptime with cloud-based hosting.
* Autom.9ated backups and disaster recovery measures.

#### SECURITY

* Data encryption for user information and payments.
* Secure authentication (OAuth, JWT).

#### RELIABILITY

#### The platform should ensure minimal downtime and consistent performance

#### A failover mechanism should be implemented for high availability.

#### PORTABILITY

* The website should be accessible on multiple devices, including desktops, tablets, and smartphones.
* The system should be easily migratable to different hosting environments.
  + 1. MAINTAINABILITY
* The system should have a modular design for easy updates and bug fixes.
* Comprehensive documentation should be available for future maintenance and upgrades.

1. **System Architectural Design**

Our website ‘Sevak’ provides services to the service requestors which is accomplished by professionals.

A user can register and use the web services as both the service requestor and professional.

* A Service Requestor can create many Requests.
* A Request is for a specific Service.
* A specified Professional can be booked many times according to their timeslot.
* A Booking is associated with one Request and one Professional.
* A Service Requestor can leave Reviews after a service is completed.
* A Review is associated with one Booking.
* A Payment is associated with one Booking.
* Location will be taken to easier search of the requestors and professional
* Verified government documents will be taken from both the requestors and the professional during details filling
* History of the service requestor will be maintained to show his previous requests
* History of the professional will be maintained to show his previous work
* Admin can access all the database and manipulate it.
  1. Detailed Description of Components
     1. Sequence Diagram

A Sequence Diagram is a type of Unified Modelling Language (UML) diagram that illustrates how objects or components interact in a specific sequence of events over time. It focuses on the time-ordered flow of messages between different elements of a system, typically during a particular process or scenario.

Key Components of a Sequence Diagram:

*Actors:*

* + Represent external entities that interact with the system (e.g., users, other systems).
  + Depicted as stick figures or labels at the top of the diagram.

*Objects/Components:*

* + Represent the entities within the system that participate in the interaction (e.g., modules, classes, services).
  + Depicted as rectangles with names.

*Lifelines:*

* Vertical dashed lines extending from each actor or object, representing their existence during the interaction.

*Messages:*

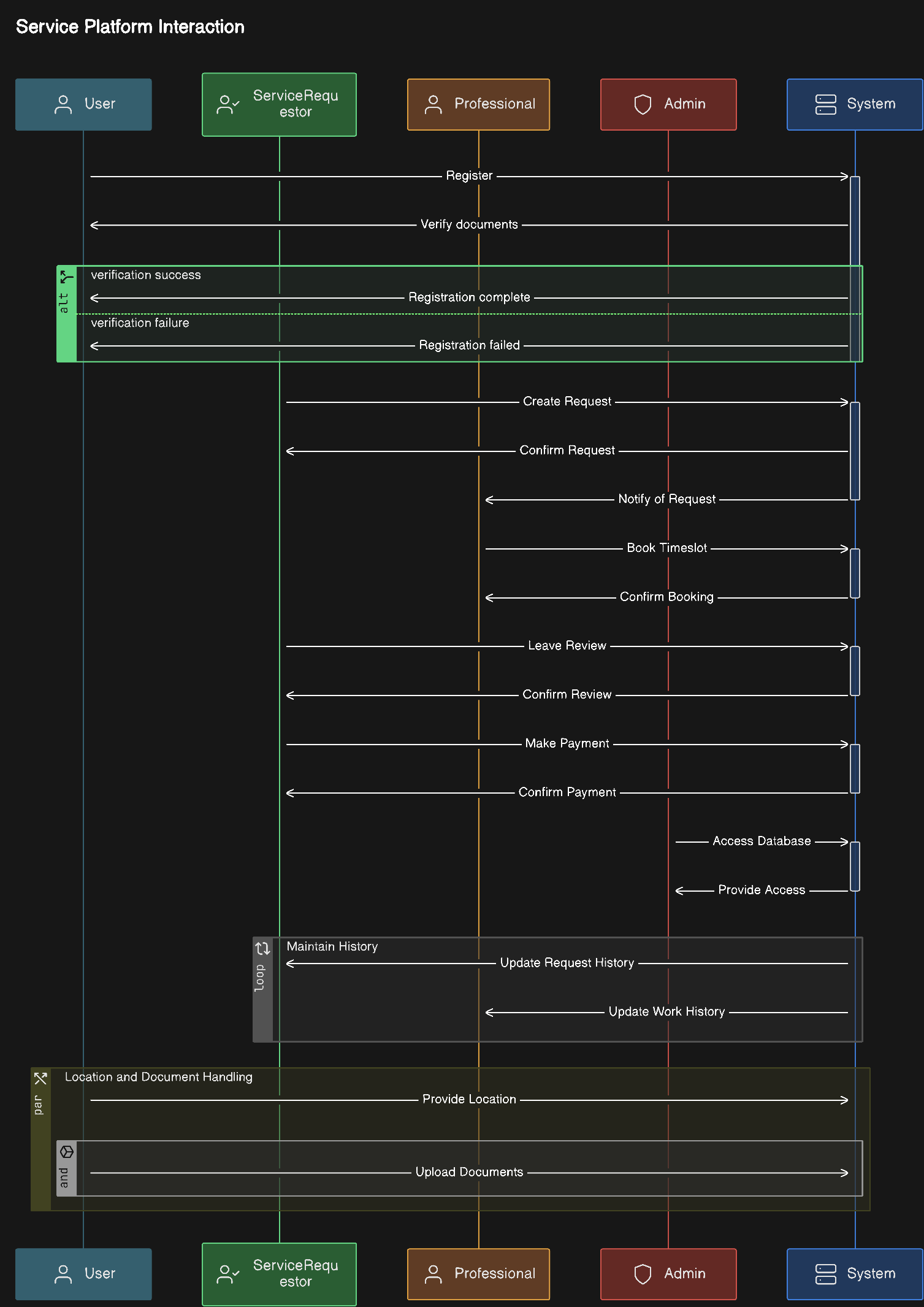
* Arrows that represent communication between actors and objects or between objects.

*Types of messages:*

* *Synchronous* (solid arrow with filled head): The sender waits for a response.
* *Asynchronous* (solid arrow with open head): The sender doesn’t wait for a response.
* *Return* (dashed arrow): Indicates a response or return value.

*Activation Bars:*

* Rectangles on lifelines showing when an object or actor is actively performing a process or operation.



* + 1. Activity Diagram

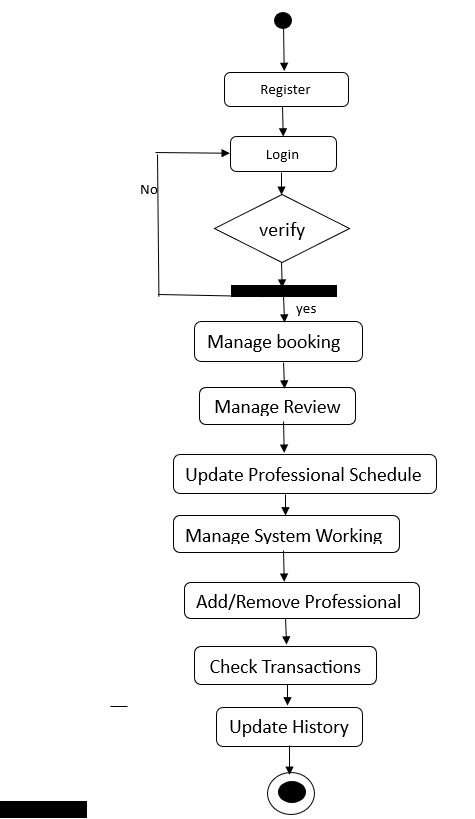
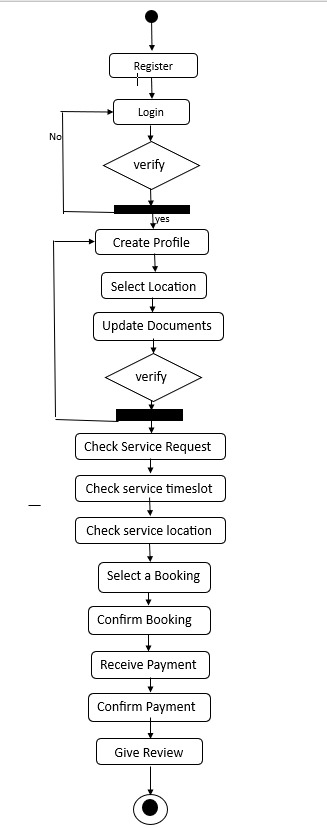
An Activity Diagram is a type of behavioral diagram in the Unified Modelling Language (UML) that represents the flow of activities or tasks in a process or system. It is widely used for modeling workflows, business processes, and use case behaviors, providing a high-level view of how activities interact and progress.

Key Features of an Activity Diagram:

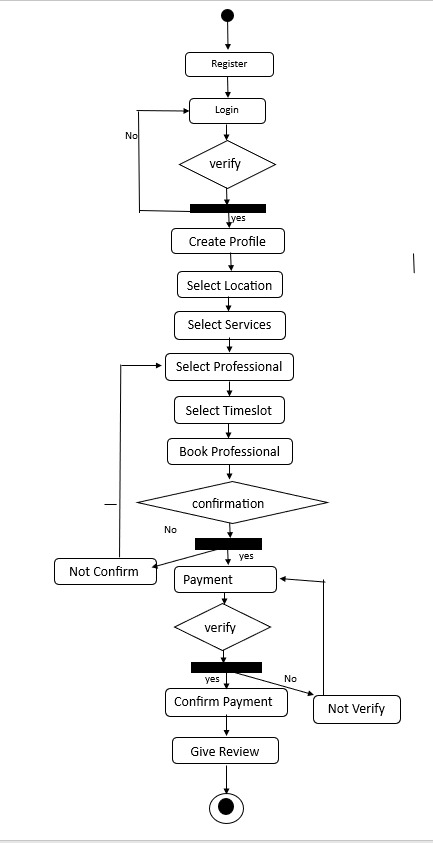
1. *Visual Representation:* Displays a sequence of activities and decisions in a graphical format.
2. *Flow of Control:* Focuses on the dynamic aspects of a system by showing how control flows from one activity to another.
3. *Parallel and Sequential Flows:* Can represent both parallel and sequential tasks in a process.
4. *Decision Points:* Includes branching (decision-making) and merging (converging different paths).

Components of an Activity Diagram:

* Activity Nodes: Represent specific actions or tasks (e.g., "Log in", "Submit Form").
* Control Flows: Arrows that show the sequence in which activities are performed.
* Start Node: A filled circle that marks the starting point of the workflow.
* End Node: A circle with a border surrounding a filled circle, indicating the termination of the process.
* Decision Nodes: Diamonds used to represent conditional branching.
* Fork and Join Nodes: Represent parallel processes.
  + Fork: Splits a flow into multiple concurrent flows.
  + Join: Combines multiple flows into one.



Professional activity Diagram Admin activity Diagram



Service requestor activity diagram

* + 1. Data Flow Diagram (DFD)

It is a graphical representation of the flow of data within a system. It maps out how information is input, processed, stored, and output by identifying the various components involved and the flow of data between them. DFDs are commonly used to model system processes and designs clearly and simply.

Key Components of a DFD:

1. Processes:
   * Represent activities or functions that transform data.
   * Depicted as circles or ovals.
   * Labelled with a process name (e.g., "Validate User").
2. Data Flows:
   * Represent the movement of data between components.
   * Depicted as arrows labelled with the data being transferred (e.g., "User Credentials").
3. Data Stores:
   * Represent storage locations where data is held.
   * Depicted as open-ended rectangles.
   * Labelled with a name that indicates the stored data (e.g., "User Database").
4. External Entities:
   * Represent sources or destinations of data outside the system.
   * Depicted as rectangles.
   * Labelled with the name of the entity (e.g., "User", "Payment Gateway").

Levels of DFDs:

1. Context Diagram (Level 0 DFD):
   * The highest-level DFD provides a big-picture view of the system.
   * Shows the system as a single process and its interactions with external entities.
2. Level 1 DFD:
   * Break down the single process from the context diagram into sub-processes.
   * Provides more detail about data flows and interactions within the system.
3. Level 2 DFDs:
   * Further decomposition of processes to illustrate detailed flows of data.

*Level 0: Context Diagram*

Entities:

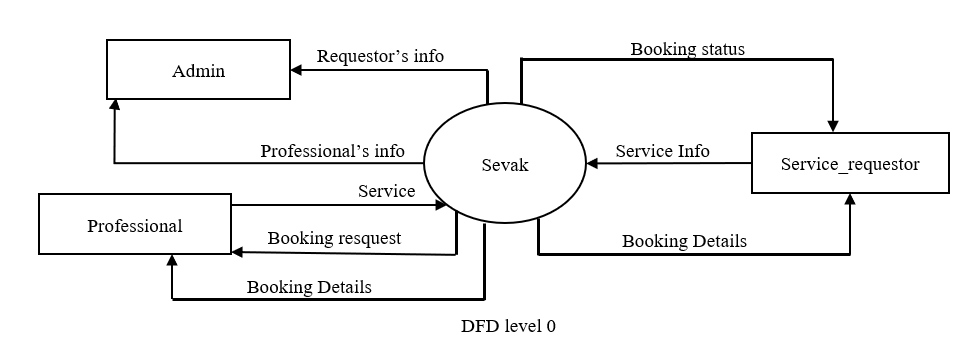
* + - Service Requestor
    - Professional
    - Admin

Processes:

* + - Service Request Processing
    - Booking & Payment
    - Review System
    - User Registration

Data Stores:

* + - Users Database
    - Requests Database
    - Booking Database
    - Payment Records
    - Reviews Database
    - Admin Database



*Level 1: Expanded Process Details*

1. User Registration

* + - Service Requestor & Professional submit required details (including government documents)
    - System verifies identity and stores in the Users Database

2. Service Request Processing

* + - The Service Requestor submits a request for a specific service
    - Location data is taken for optimized searching
    - Request stored in the Requests Database

3. Booking & Payment

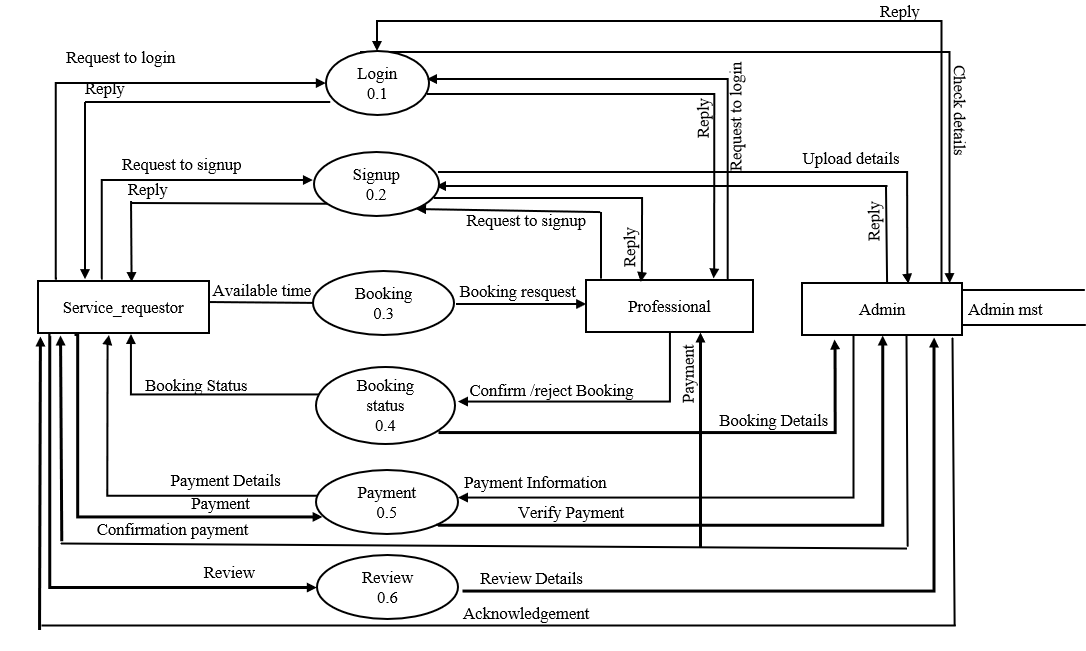
* + - Professionals choose available slots
    - A Booking is created linking Request and Professional
    - Payment is processed and recorded

4. Review System

* After service completion, the Requestor can leave a review
* Review is linked to the respective Booking

5. Admin Management

* + - Admin has full access to the database
    - Can monitor and manipulate records



DFD level 1

* + 1. Entity -Relation Diagram

An entity-relationship (ER) database is a database designed using an ER model, which is a visual representation of the database's structure. ER models use diagrams to show how entities are connected and interact with each other.

Its components are:

1. Entities:

* Represent real-world objects or concepts, such as customers, products, or courses.

1. Relationships:

* Define how entities are connected, such as one-to-one, one-to-many, or many-to-many relationships

1. Attributes:

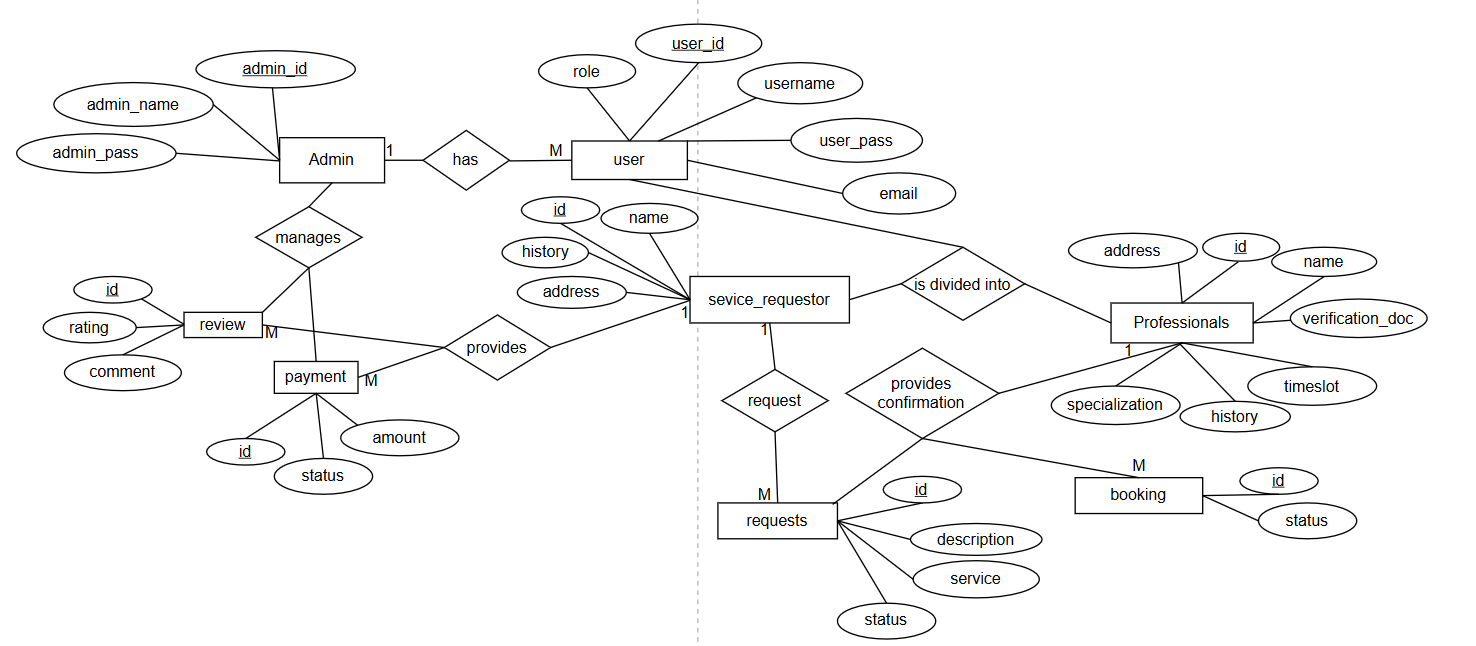
* Define the properties of an entity, such as name, age, or ID

1. Diagrams**:**

* Use rectangles to represent entities and diamonds to represent relationships.

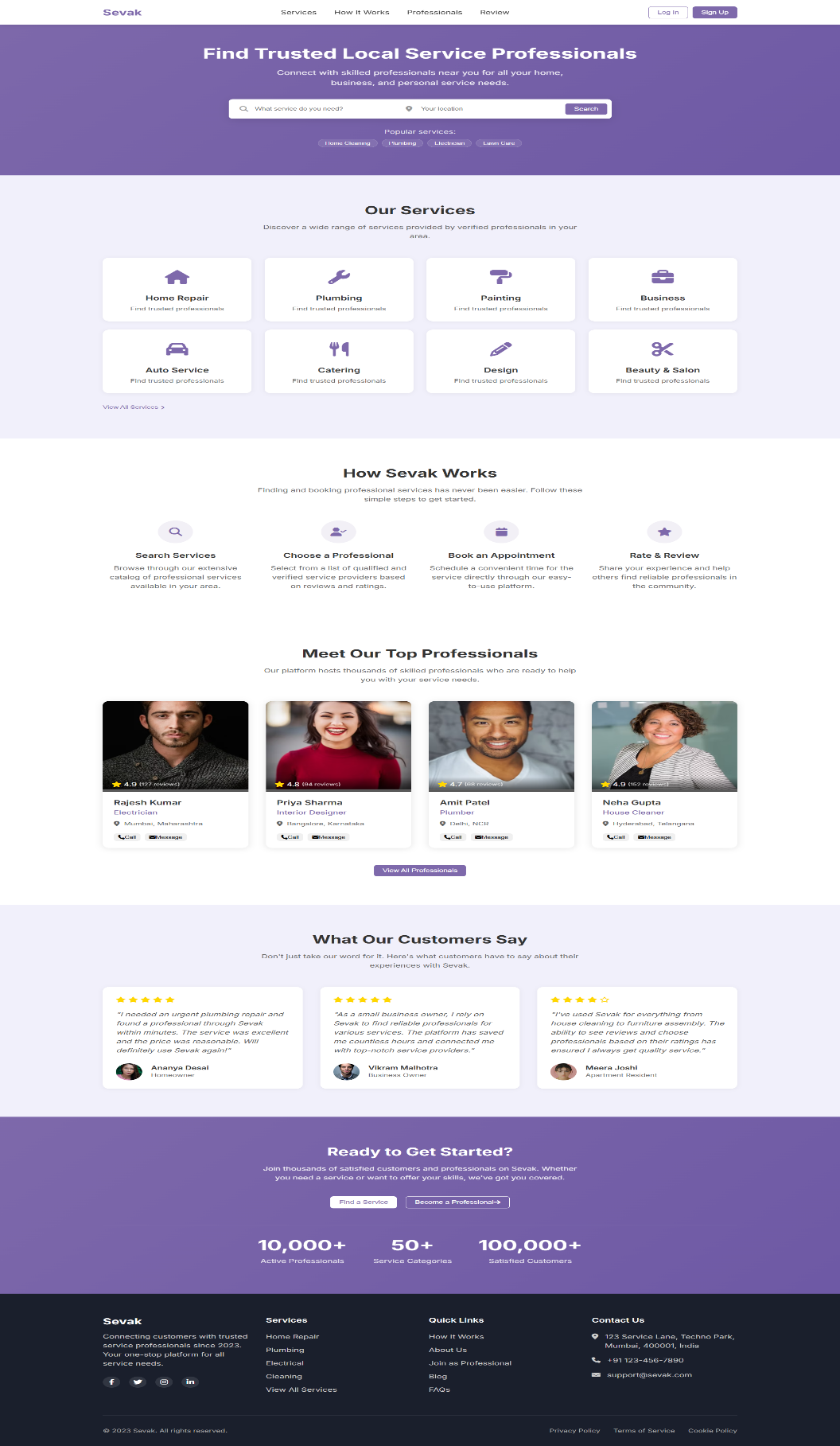
Why it's useful: -

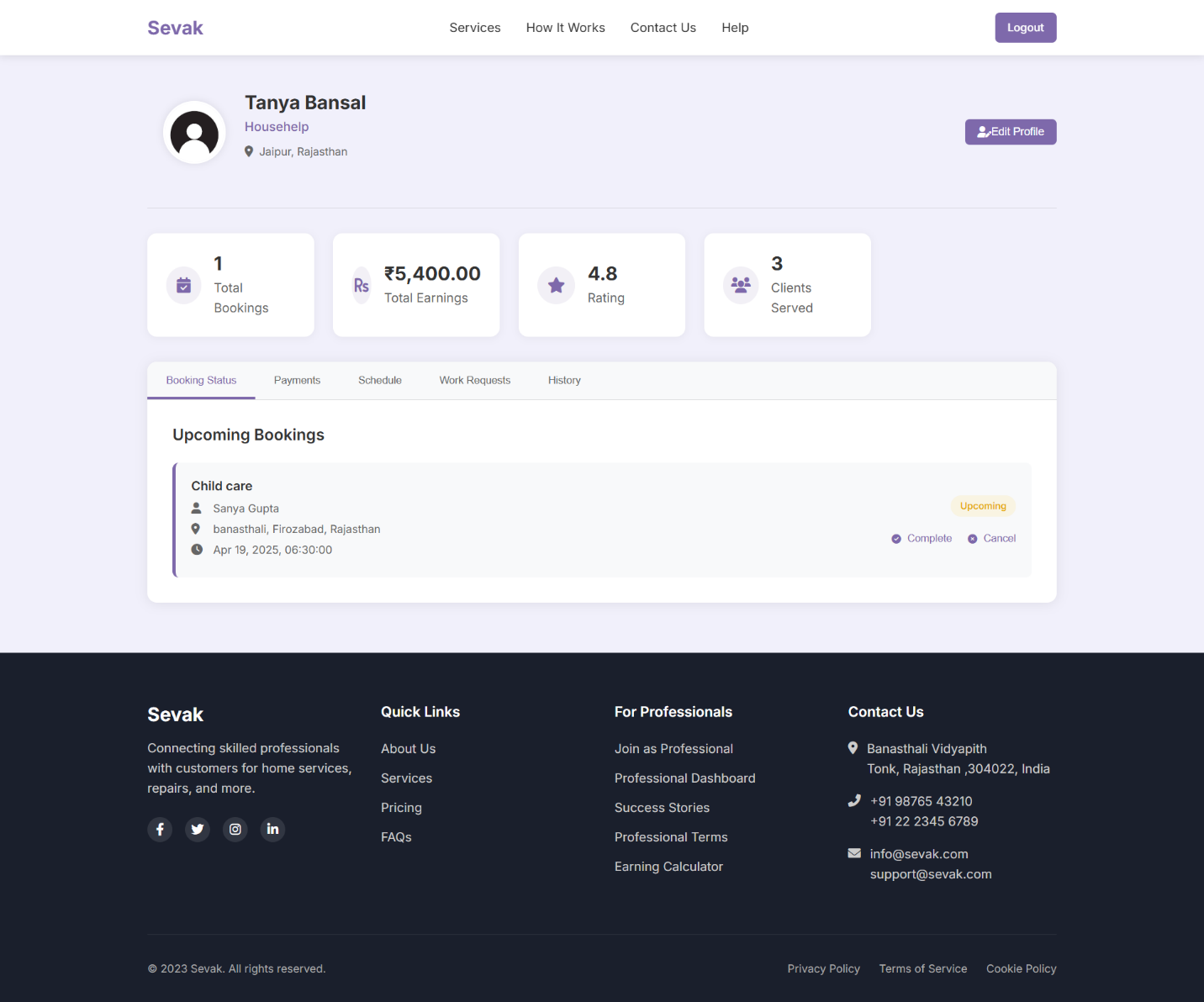
* ER models make it easier to design and understand databases
* They help organize data and understand how entities interact
* They are a systematic process that requires analyzing data requirements before implementing the database.

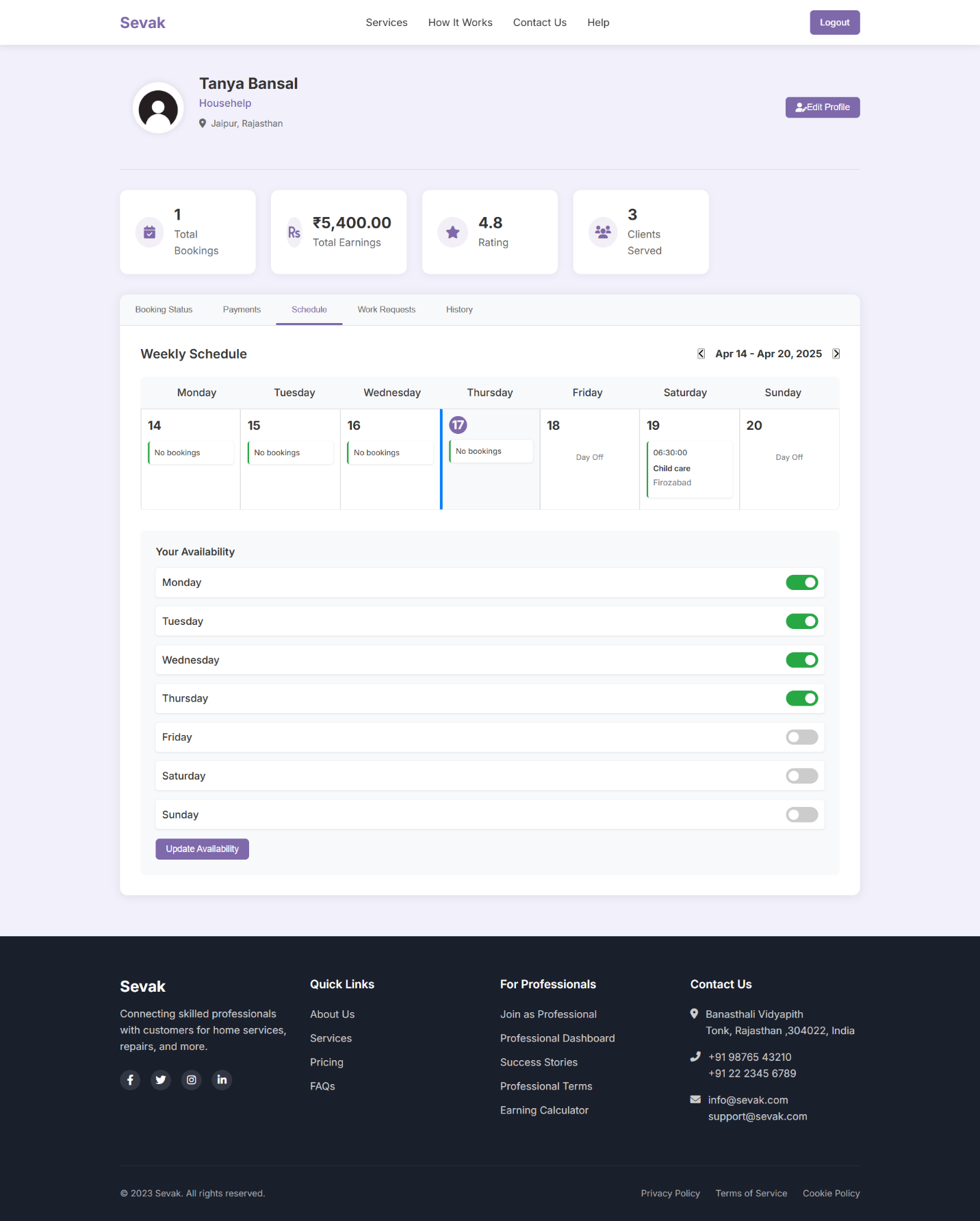


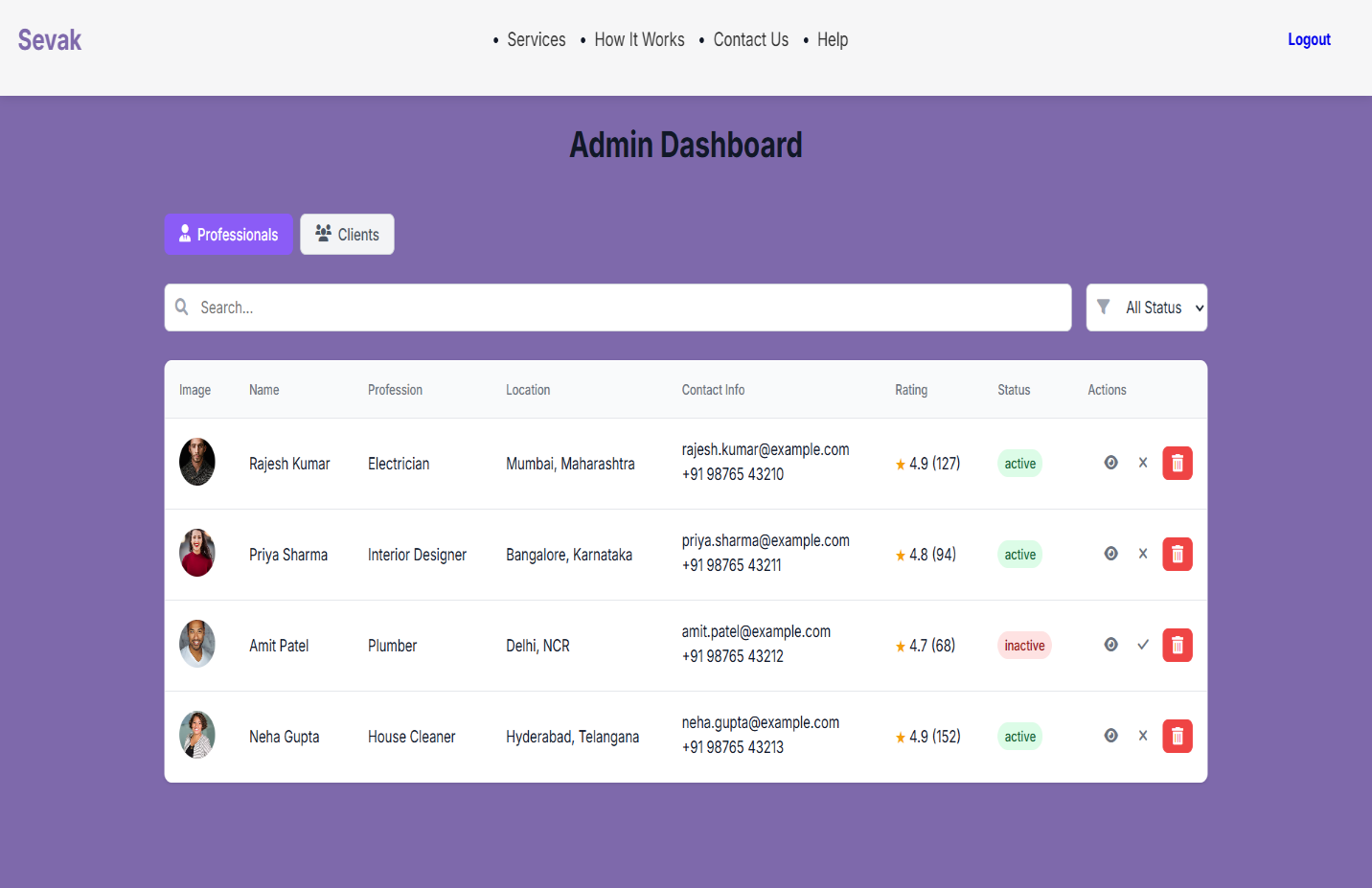
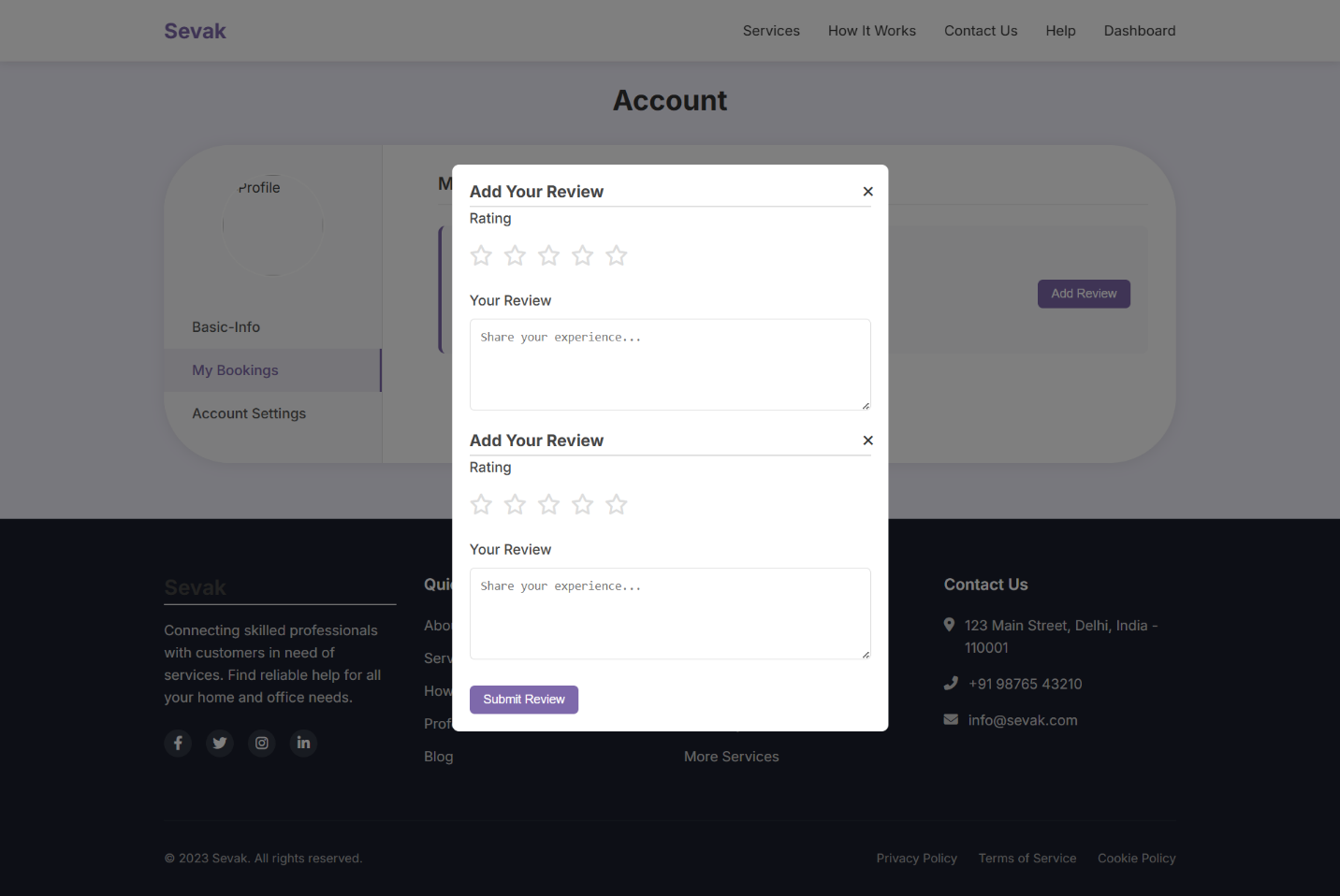
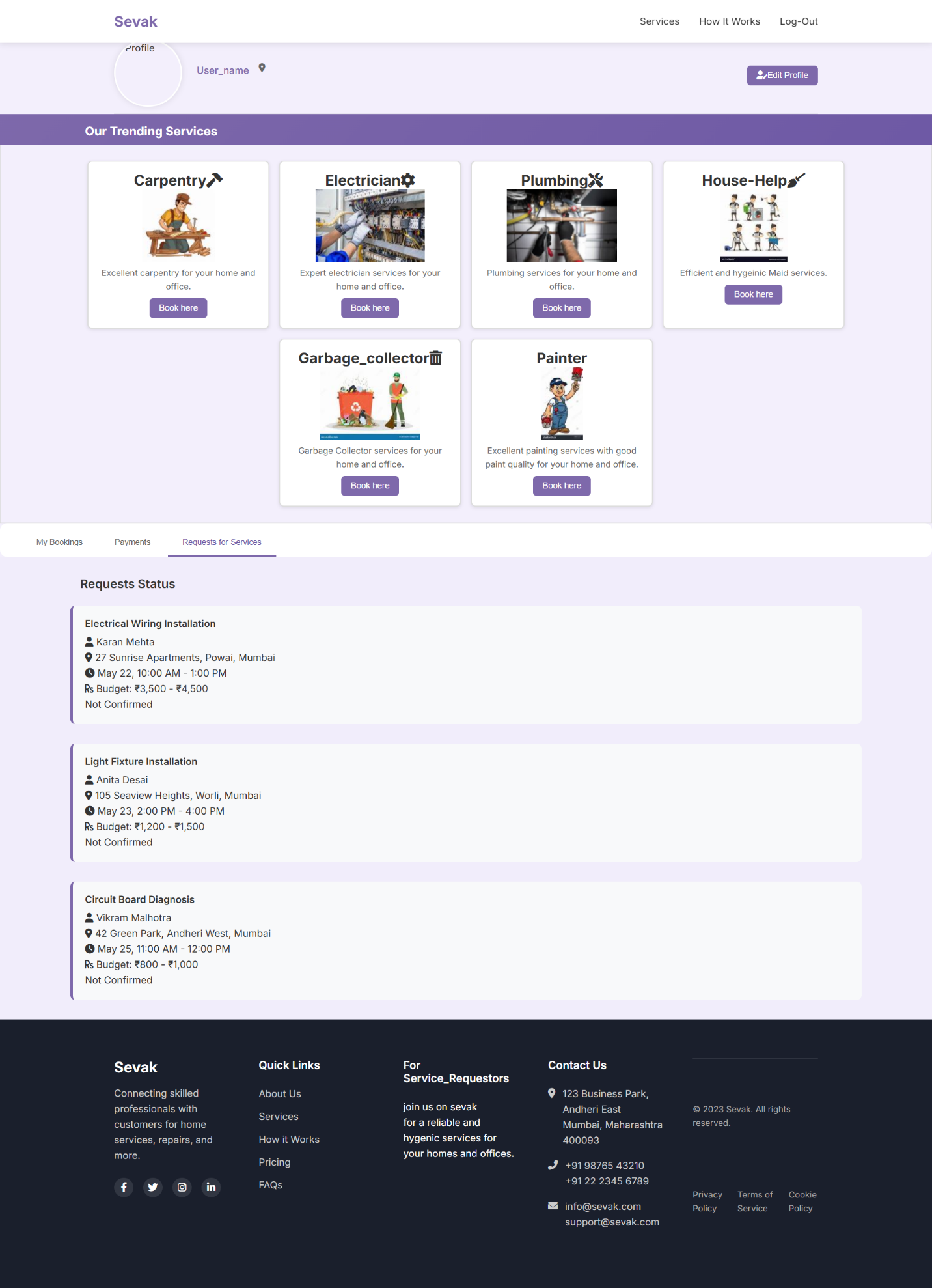
Entity-Relation Diagram

1. **Implementation & User Interfaces**

****

****

****

****

1. **Software Testing**
   1. Testing

Software testing is the process of executing a program with intension of finding errors in the code. It is a process of evolution of system or its parts by manual or automatic means to verify that it is satisfying specified or requirements or not.

Generally, no system is perfect due to communication problems between user and developer, time constraints, or conceptual mistakes by developer.

To purpose of system testing is to check and find out these errors or faults as early as possible so losses due to it can be saved. Testing is the fundamental process of software success.

Testing is not a distinct phase in system development life cycle but should be applicable throughout all phases i.e., design development and maintenance phase.

Testing is used to show incorrectness and considered to success when an error is detected.

* 1. Objectives of Software Testing
* Software Quality Improvement: The computer and a bug or fault in software causes severe losses. So, a great consideration is required for checking for quality of software.
* Verification and Validation: Verification means to test that we are building the product in right way.i.e., are we using the correct procedure for the development of software so that it can meet the user requirements. Validation means to check whether we are building the right product or not.
* Software Reliability Estimation: The objective is to discover the residual designing errors before delivery to the customer. The failure data during process are taken down in order to estimate the software reliability.
  1. Principles of Software Testing
* All tests should be traceable to end user requirements.
* Tests should be planned long before testing begins
* Testing should begin on a small scale and progress towards testing in large
* To be most effective testing should be conducted by an independent third party.

The primary objective for test case design is to derive a set of tests that has the highest livelihood for uncovering defects in software.

To accomplish this objective two different categories of test case design techniques are used. They are:

* White Box Testing: White box testing focus on the program control structure. Test cases are derived to ensure that all statements in the program have been executed at least once during testing and that all logical conditions have been executed.
* Black Box Testing: Black box testing is designed to validate functional requirements without regard to the internal workings of a program. Black box testing mainly focuses on the information domain of the software, deriving test cases by partitioning input and output in a manner that provides through test coverage. Incorrect and missing functions, interface errors, errors in data structures.
  1. Testing fundamentals

Testing is a process of executing program with the intent of finding error. A good test case is one that has high probability of finding an undiscovered error. If testing is conducted successfully, it uncovers the errors in the software. Testing cannot show the absence of defects, it can only show that software defects present.

* 1. Testing Information flow

Information flow for testing flows the pattern. Two class of input provided to test the process. The software configuration includes a software requirements specification, a design specification and source code.

Test configuration includes test plan and test cases and test tools. Tests are conducted and all the results are evaluated. That is test results are compared with expected results. When erroneous data are uncovered, an error is implied and debugging commences.

1. **Conclusion**

This system avoids manual work and the problems concern with it. This system was designed in such a way that future modifications can be done easily. The following conclusions can be deduced from the development of the project

* Automation of the entire system improves the efficiency.
* It provides a friendly graphical user interface which proves to be best.
* It gives appropriate access to the authorized users depending on their permissions.
* It effectively overcomes the delay in communications.
* Updating information becomes so much easier.
* System security, data security, and reliability are the striking features.
* The System has adequate scope for modification in the future if it is necessary.

1. **References**

This document uses references from the following sources of:

 Core Web Technologies

1. World Wide Web Consortium (W3C). (2023). *HTML5: The latest standard for markup*. <https://html.spec.whatwg.org/>
   * Relevance: Official HTML5 specifications used in Sevak.
2. Mozilla Developer Network (MDN). (2023). *CSS: Cascading Style Sheets*. <https://developer.mozilla.org/en-US/docs/Web/CSS>
   * Relevance: Documentation for styling Sevak’s frontend.
3. ECMA International. (2023). *ECMAScript 2023 (JavaScript) language specification*. <https://262.ecma-international.org/>
   * Relevance: Standard reference for JavaScript used in Sevak.
4. PHP Group. (2023). *PHP 8.2 documentation*. <https://www.php.net/docs.php>
   * Relevance: Official PHP backend scripting for Sevak.
5. Apache Friends. (2023). *XAMPP: Apache distribution for local development*. <https://www.apachefriends.org/index.html>
   * Relevance: Local server environment for Sevak’s development.
6. MySQL. (2023). *MySQL 8.0 reference manual*. <https://dev.mysql.com/doc/>
   * Relevance: If Sevak uses MySQL (included in XAMPP).
7. OWASP. (2023). *PHP security cheat sheet*. <https://cheatsheetseries.owasp.org/cheatsheets/PHP_Security_Cheat_Sheet.html>
   * Relevance: Secure coding practices for Sevak’s PHP backend.
8. Mozilla Developer Network (MDN). (2023). *Web security guidelines*. <https://developer.mozilla.org/en-US/docs/Web/Security>
   * Relevance: General security practices for Sevak’s frontend.