Software Requirements Specification

for

“Fixly”

Prepared by

Group ID: 33

Pansuriya Devangi	**202412057**	**202412057@daiict.ac.in**
Katharotiya Harsh	**202412036**	**202412036@daiict.ac.in**
Shekhda Miraj	**202412104**	**202412104@daiict.ac.in**

Instructor: **Prof. Jayprakash Lalchandani**

Course: **Software Engineering(IT632)**

Date: **28/02/2025**

Contents

1 Introduction

Fixly is a service provider platform, transforming the way households access essential services. This intuitive application makes it easy to find and book skilled professionals, including plumbers, carpenters, electricians, cleaners, home repair specialists, caterers, and more—all from the convenience of your home. It is designed to revolutionize how service seekers connect with trusted professionals for their everyday needs. The aim of this application is to create a reliable and efficient service provider platform that connects users with skilled professionals for various household services.

1.1 Document Purpose

This Software Requirements Specification (SRS) document outlines the functional and NFRs for the development of Fixly, a web-based customer service platform. Fixly is designed to bridge the gap between service seekers and skilled professionals by providing a seamless online marketplace for essential household services. The platform facilitates service booking, provider management, and ensuring a reliable and convenient experience for users.

This document covers all major system components, including user and service provider management, booking functionalities, rating and review mechanisms, and profile management. It serves as a comprehensive reference for stakeholders, including developers, project managers, and clients, to understand the system’s features, constraints, and objectives.

1.2 Product Scope

The key objectives of this product include:

- Provide an Efficient Service Marketplace: Create a platform where users can easily browse, book, and manage professional service providers for their household needs.
- Enhance Accessibility and Convenience: Ensure that users can access a wide range of services at their fingertips without the need for extensive searching or unreliable contacts.
- Strengthen Service Provider Network: Onboard skilled professionals and help them reach a broader customer base, ensuring a steady flow of work opportunities.
- Ensure Seamless Booking and Management: Offer a smooth and intuitive booking system that allows users to schedule single or multiple services efficiently.
- Improve Transparency and Trust: Implement user reviews, ratings, and service history tracking to build trust and ensure quality service delivery.
- Optimize User and Service Provider Management: Provide robust features for adding, updating, and managing user and service provider information efficiently.
- Provide a One-Stop Solution for All Household Services: Create a unified platform where users can conveniently find and book a wide range of household services—eliminating the need to search multiple sources for different service providers.
- The main benefits of this platform include convenience, efficiency, reliability, and increased work opportunities for service providers, ultimately improving customer satisfaction and professional engagement.

The primary goals of Fixly are:

- Create a seamless service marketplace where users can easily book household services.
- Ensure accessibility and convenience by providing a user-friendly and intuitive platform.
- Enhance trust and transparency through ratings, reviews.
- Enhance Accessibility: Offer a platform that makes essential household services available anytime, anywhere.
- Support Skilled Professionals: Provide independent service providers with a platform to promote their skills and expand their customer base.
- Optimize User Experience: Deliver a well-structured, user-friendly system that ensures fast, hassle-free service booking and management.

1.3 Intended Audience and Document Overview

Intended Audience

This Software Requirements Specification (SRS) document is intended for the following stakeholders:

- **Professor & Evaluators:** To assess the project's feasibility, completeness, and adherence to software engineering principles.
- **Developers (Project Team):** To understand the functional and non-functional requirements and guide the implementation of the platform.
- **Users:** To review the system's capabilities and ensure it meets business and user needs.
- **Testers:** To verify that the platform functions correctly by validating features against the documented requirements.

Document Overview:

This document provides a detailed Software Requirements Specification (SRS) for the Fixly service platform. It outlines the system's purpose, features, and constraints to ensure clarity for all stakeholders.

The document is structured as follows:

1. **Introduction –** Describes the purpose, scope, intended audience, and conventions used in the document.

2. **Overall Description –** Provides an overview of the system, including its features, constraints, and dependencies.

3. **Specific Requirements –** Details functional and external interface requirements, along with use case model, DFD and ER Model.

4. **Non-Functional Requirements –** Covers performance, security, and software quality attributes.

5. **Other Requirements –** Lists any additional requirements relevant to the system.

6. **Appendices –** Includes supporting documents such as a data dictionary and group log.

This document serves as a guideline for developers, testers, and evaluators, ensuring the system meets all functional and non-functional requirements.

1.4 Definitions, Acronyms and Abbreviations

Definitions

- Service Provider – A professional or skilled worker who offers various household or personal services through the platform.
- User – An individual who registers on the platform to book services from service providers.
- Booking Request – A formal request made by a user to hire a service provider for a specific task.
- Service Confirmation – A notification sent to both the user and the provider upon successful booking.
- Rating & Review System – A feature that allows users to rate and provide feedback on service providers based on their experience.
- Authentication – A security mechanism used to verify users and service providers before granting access.
- Dashboard – A personalized user interface where users can manage their bookings, profile, and transaction history.
- Payment Gateway – A system that facilitates online transactions between users and service providers.
- Admin Panel – A backend interface used by administrators to manage users, service providers, and bookings.
- Stakeholder - A stakeholder is any individual, group, or entity that has an interest in or is affected by the development, deployment, and operation of the software system.

Acronyms and Abbreviations

- SRS – Software Requirements Specification
- DFD – Data Flow Diagram
- ERD – Entity-Relationship Diagram
- JWT – JSON Web Token (used for secure authentication)
- API – Application Programming Interface
- UI – User Interface
- NFR - Non - Functional Requirements
- DBMS – Database Management System

- CRUD – Create, Read, Update, Delete (operations for database management)

1.5 Document Conventions

Main Title: Arial, Bold, Size 18

Subtitles: Arial, Bold, Size 14

Body Text: Arial, Normal, Size 14

Text Alignment: Justified

1.6 References

https://www.taskrabbit.com
https://www.urbancompany.com
https://www.google.com

2.Overall Description

1.7 Product Overview

Fixly is more than just a service provider platform—it's a bridge connecting customers with skilled professionals for a variety of household and commercial needs. Designed with simplicity and efficiency in mind, Fixly offers an intuitive interface where users can easily book services, manage appointments, and communicate with service providers. Whether it's plumbing, cleaning, or technical support, customers can schedule services, monitor their progress, and share feedback. Meanwhile, service providers have the tools to manage their offerings ,track schedules, and engage with clients.

1.8 Product Functionality

1. Service Provider Booking : Effortlessly find and book household service providers with just a few clicks. Users can book single or multiple service providers at a time, ensuring a seamless experience for all household needs.

2. User and Service Provider Management : Add, Delete, Update : Manage user and service provider information efficiently.

3.Filter Services: Easily filter services based on various factors to find the perfect match for your requirements.

4. Booking History : Users can conveniently view their previous bookings in their dashboard, providing a transparent record of past transactions.

5. Ratings and Reviews System :
Implement a transparent rating and review system where users can provide feedback on service providers, helping maintain quality standards and build trust within the platform.

6. Profile Management : User Profile Update: Users can easily update their profile details for a personalized experience.

7. Easy Service Booking :

Simplify the process of booking services by providing an intuitive interface that enables users to select their required service and confirm their booking instantly ensuring a hassle-free experience without the need for appointments.

8.Skilled Professional registration :

Provide an easy and streamlined registration process for skilled professionals to join the platform as service providers. Fixers can create a profile, list their expertise, set availability - expanding their reach and securing more work opportunities.

1.9 Design & Implementation Constraints

The development of the Service Provider Booking Platform is subject to several constraints that will impact its design and implementation. These constraints include:

- Technology Stack: The platform will be developed using ReactJS, HTML, CSS, and JavaScript for the frontend, while the backend will be powered by Node.js and Express.js.
- Hardware Limitations: The system must operate efficiently on standard web servers and support scalable cloud infrastructure to handle varying workloads.
- Integration Constraints: The platform must integrate with Razorpay for secure payment processing.
- Security Considerations: Robust security measures, including user authentication, data encryption, and access control, must be implemented to safeguard sensitive user information.
- Authentication: JWT-based authentication will be implemented to provide secure access control and user verification.
- Scalability & Performance Constraints : The backend, built using Node.js and Express.js, should ensure fast API responses and scalable database interactions with MongoDB.

1.10 Assumptions and Dependencies

Assumptions:

1. Users will have access to a stable internet connection to use the platform.

2. Service providers will provide accurate and up-to-date information about their services.

3. The system will be accessed primarily through web browsers and mobile devices.

4. Users and service providers will follow the platform's terms and conditions.

Dependencies:

1. **Technology Stack:** The platform relies on ReactJS for the frontend, Node.js with Express.js for the backend, and MongoDB as the database.

2. **Authentication System:** Uses JWT-based authentication for secure login and access control.

3. **Hosting & Deployment:** The system depends on platforms like Heroku, Vercel, or Netlify for deployment.

4. **External APIs:** third-party services are integrated, they must be available and functional.

5. **User Engagement:** The platform’s success depends on active participation from both users and service providers.

3.Specific Requirements

1.11 External Interface Requirements

3.1.1 User Interfaces

The system provides an interactive and easy-to-use interface for both customers and service providers. The main components of the user interface include:

1. **Web-Based Dashboard**
    - The system will be accessible via a web application built with ReactJS, HTML, CSS, and JavaScript.
    - A responsive design will ensure compatibility with desktops, tablets, and mobile devices.
2. **User Interaction**
    - The interface will have a clean and intuitive layout to ensure easy navigation.
    - Users can interact with the platform using buttons, dropdowns, search bars, and filtering options.
    - Key actions such as service booking, profile updates, and viewing booking history will be accessible from the dashboard.
3. **Service Provider Panel**
    - Service providers will have their own dashboard to manage bookings, update availability, and respond to customer requests.
    - Notifications will inform service providers about new booking requests and customer messages.
4. **Booking System**
    - Users can search, filter, and book services through an interactive form.
    - Booking confirmations will be displayed in real-time.
5. **Notifications and Alerts**
    - Users will receive real-time notifications via in-app messages and emails for booking confirmations and service updates.
6. **Ratings & Reviews**
    - Customers can rate and review service providers to maintain platform credibility.
    - Service providers can view their ratings and respond to feedback.

3.1.2 Hardware Interfaces

The system will interact with the following hardware components:

1. **User Devices**
    - Supports laptops, desktops, tablets, and mobile phones for web access.
2. **Hosting Servers**
    - The backend will be hosted on cloud platforms like Heroku, Vercel, or Netlify to ensure scalability.
3. **Database Server**
    - The system will store and retrieve data using MongoDB, which will run on a cloud-hosted database server.
4. **Authentication System**
    - JWT-based authentication will be used to verify user identities securely.

3.1.3 Software Interfaces

The platform will integrate with various software components to ensure seamless functionality:

1. **Frontend-Backend Interaction**
    - The frontend (ReactJS) will communicate with the backend (NodeJS, Express.js) using RESTful APIs.
2. **Database Connection**
    - The backend will interact with the MongoDB database for storing user data, bookings, and reviews.
3. **External API Integration**
    - Integration with external service providers for authentication and data validation.
4. **Hosting & Deployment**
    - The application will be deployed on Heroku, Vercel, or Netlify for hosting.
    - Git & GitHub will be used for version control and collaboration.

3.2 Functional Requirements

These requirements ensure that the system operates efficiently for both users (customers) and service providers while maintaining security and usability.

1. **User Registration & Authentication**
    - Users and service providers can sign up and log in securely.
    - Profile management options will be available.
2. **Service Search & Booking**
    - Users can search for service providers based on category, availability, location, and rating.
    - Users can book a single or multiple service providers at a time.
    - Booking confirmation notifications will be sent to users and service providers.
3. **Service Provider Management**
    - Service providers can create and manage their profiles, including expertise, availability, and pricing.
    - They can accept or reject booking requests.
4. **Booking History & Tracking**
    - Users can view their past and upcoming bookings on a dashboard.
    - Service providers can track completed and pending bookings.
5. **Ratings and Reviews**
    - Users can provide ratings and reviews for service providers.
    - Service providers can respond to customer feedback.
6. **Notification System**
    - Users and service providers receive real-time notifications about booking confirmations, updates, and cancellations.
7. **Profile Management**
    - Users can update personal information, such as name, contact details, and preferences.
    - Service providers can update their availability, skills, and pricing.

3.3 Use Case Model

Use Cases :

Use Case 1: Registration

Actor: Customer, Service Provider
Description: Allows users to create an account in the system.
Basic Flow:

1. User selects the "Registration" option.
2. The system prompts for user details (name, email, phone, password).
3. User enters details and submits.
4. System verifies details and creates an account.
5. User receives a confirmation message.

Alternative Flow:

- If the password does not meet criteria, the system prompts the user to enter a valid password.
- If email is already registered, the system notifies the user.

Use Case 2: Login

Actor: Customer, Service Provider, Admin
Description: Users log into the system to access their accounts.
Basic Flow:

1. User selects the "Login" option.
2. The system prompts for email and password.
3. User enters credentials and submits.
4. The system verifies the credentials.
5. If correct, the user is granted access to their dashboard.

Alternative Flow:

- If credentials are incorrect, the system displays a "Login Error" message.

- If the user forgets the password, the system allows password recovery.

Use Case 3: Book Service

Actor: Customer
Description: A customer books a service provided by a service provider.
Basic Flow:

1. Customer searches and selects a service.
2. System filters available providers based on ratings and availability.
3. Customer submits a booking request.
4. System notifies the provider.
5. Provider accepts/rejects the request.
6. System updates booking status.
7. Customer and provider receive confirmation.

Alternative Flow:

- If no providers are available, the system notifies the customer.

Use Case 4: Manage Service Listings

Actor: Service Provider
Description: Service providers can add, update, or delete their service listings.
Basic Flow:

1. Provider logs into the system.
2. Provider selects "Manage Service Listings."
3. Provider adds or edits service details.
4. System updates the listing.

Alternative Flow:

- If service details are incomplete, the system prompts for missing information.

Use Case 5: Search & Browse Services

Actor: Customer
Description: Customers can search for services based on category, location, and provider ratings.
Basic Flow:

1. Customer selects the "Search Services" option.
2. System displays a search bar and filter options.
3. Customer enters search criteria.
4. System retrieves and displays matching services.

Alternative Flow:

- If no services match the criteria, the system displays "No results found."

Use Case 6: View & Give Rating

Actor: Customer
Description: Customers can rate and review services after completion.
Basic Flow:

1. Customer selects a completed booking.
2. System has a page for a rating and review.
3. Customer submits feedback.
4. System updates the provider's rating.

Alternative Flow:

- If the customer doesn't provide a rating, the system sends a reminder.

Use Case 7: Cancel Booking

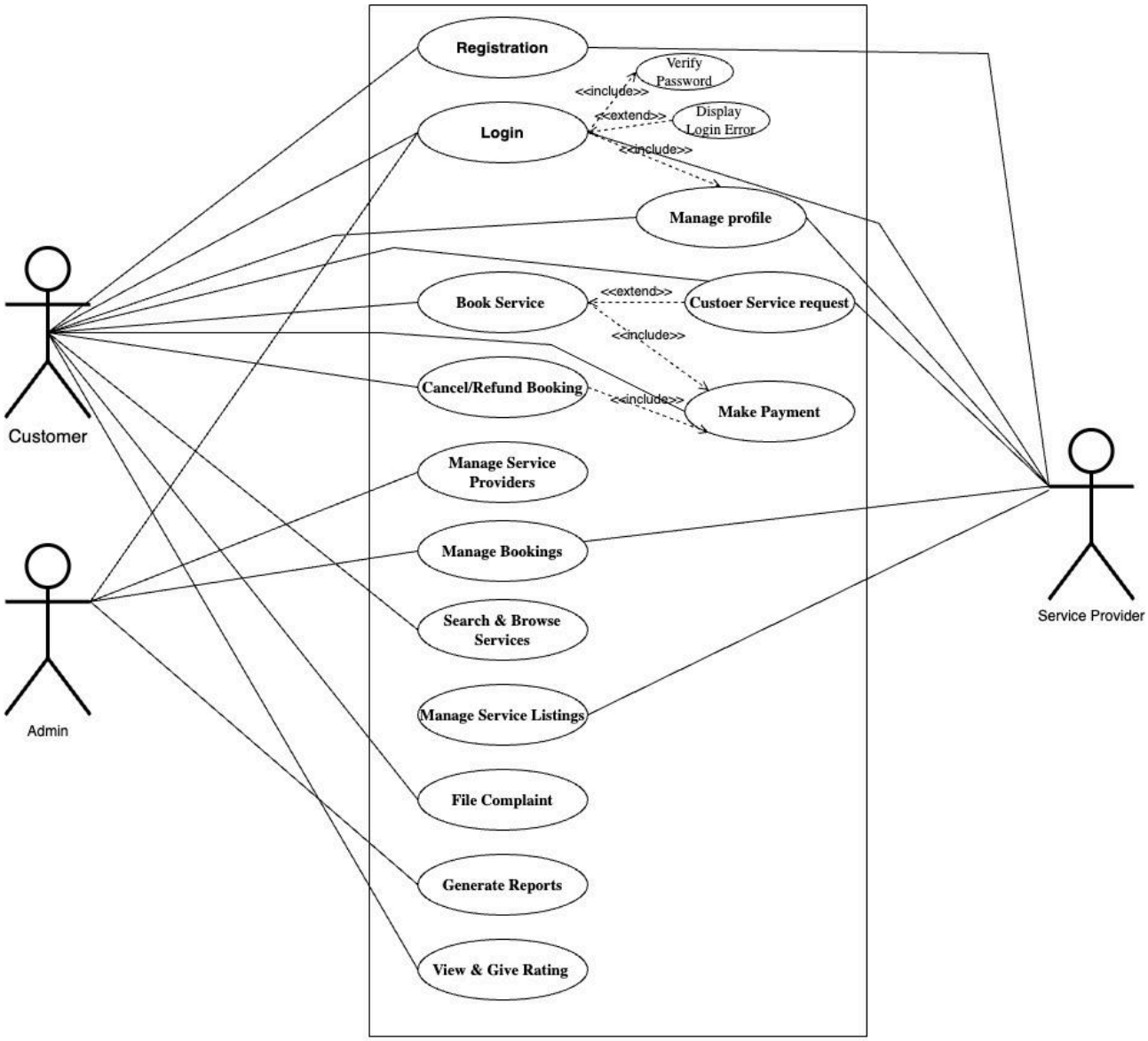
Actor: Customer
Description: Allows a customer to cancel request for a booking.
Basic Flow:

1. Customer selects a booked service.
2. System displays cancellation policy.
3. Customer submits a cancellation request.
4. System processes the request.
5. Customer receives confirmation.

Alternative Flow:

- If the cancellation deadline has passed, the system prevents cancellation.

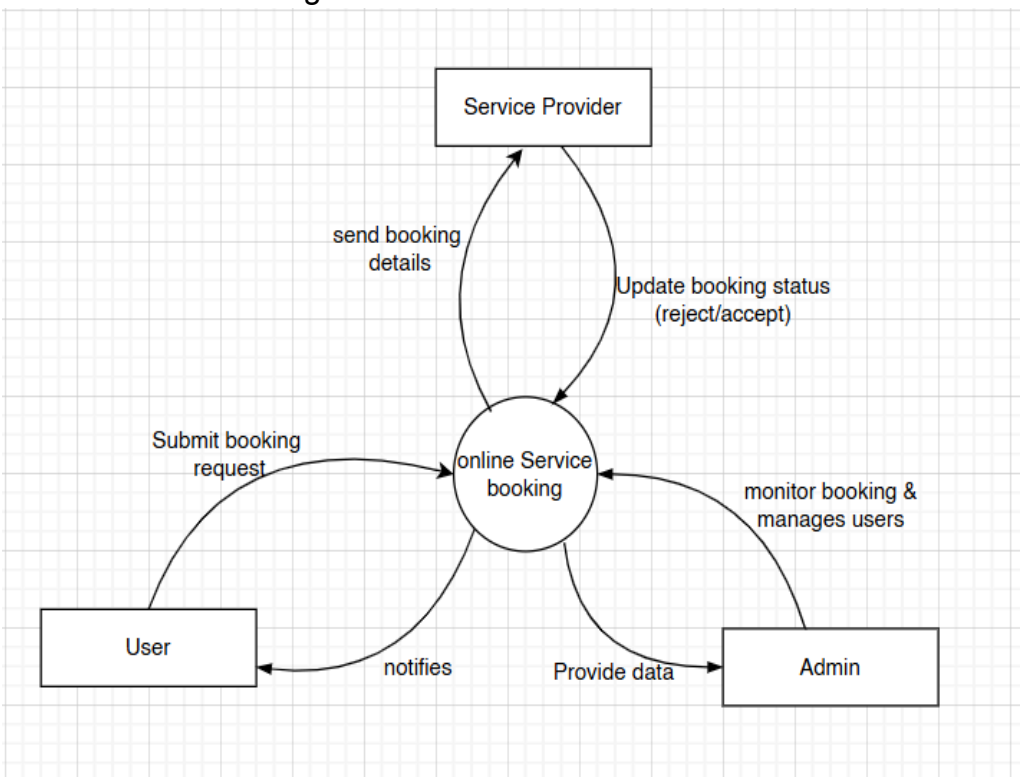
Use Case Diagram



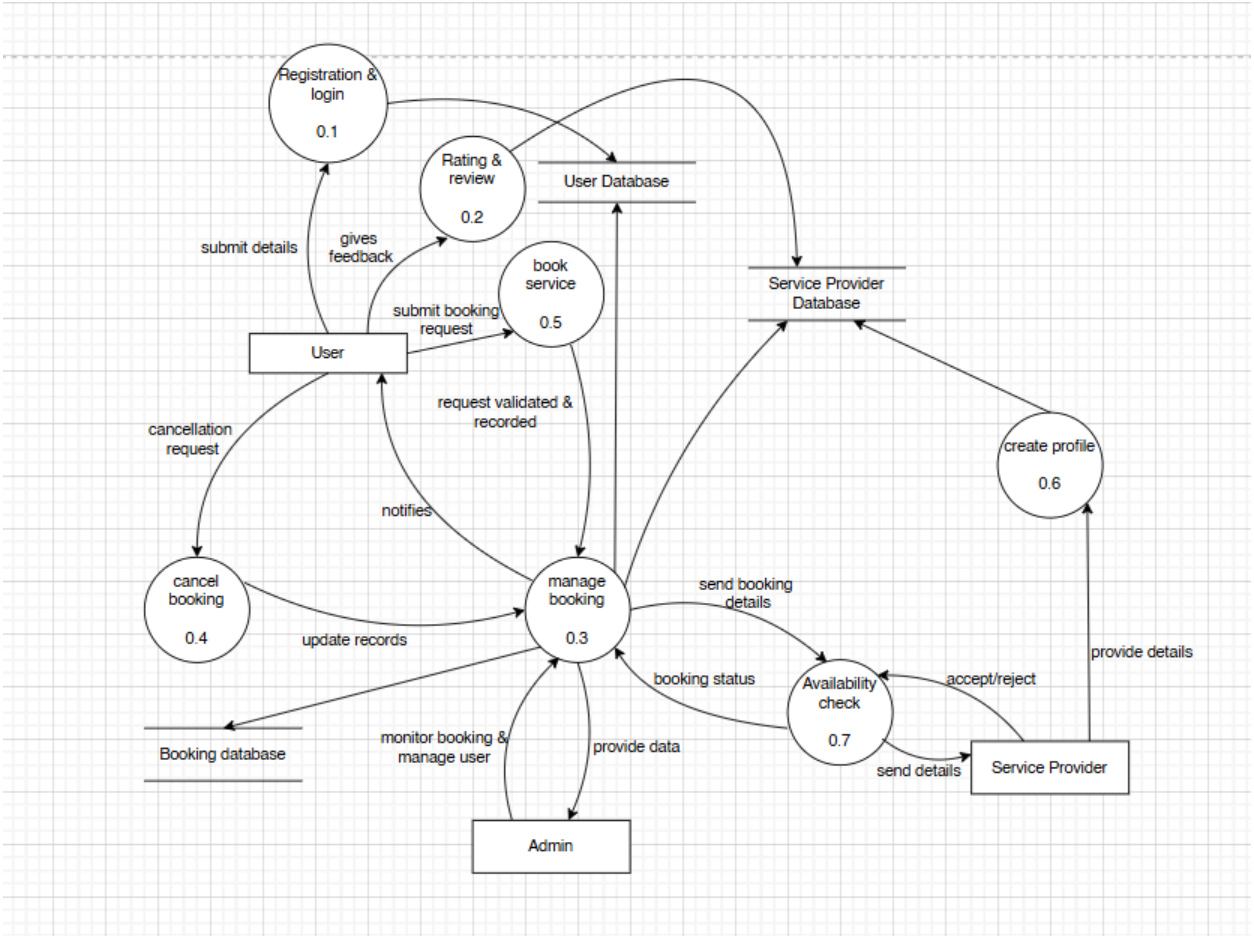
3.4 DFD Diagram

Level 0

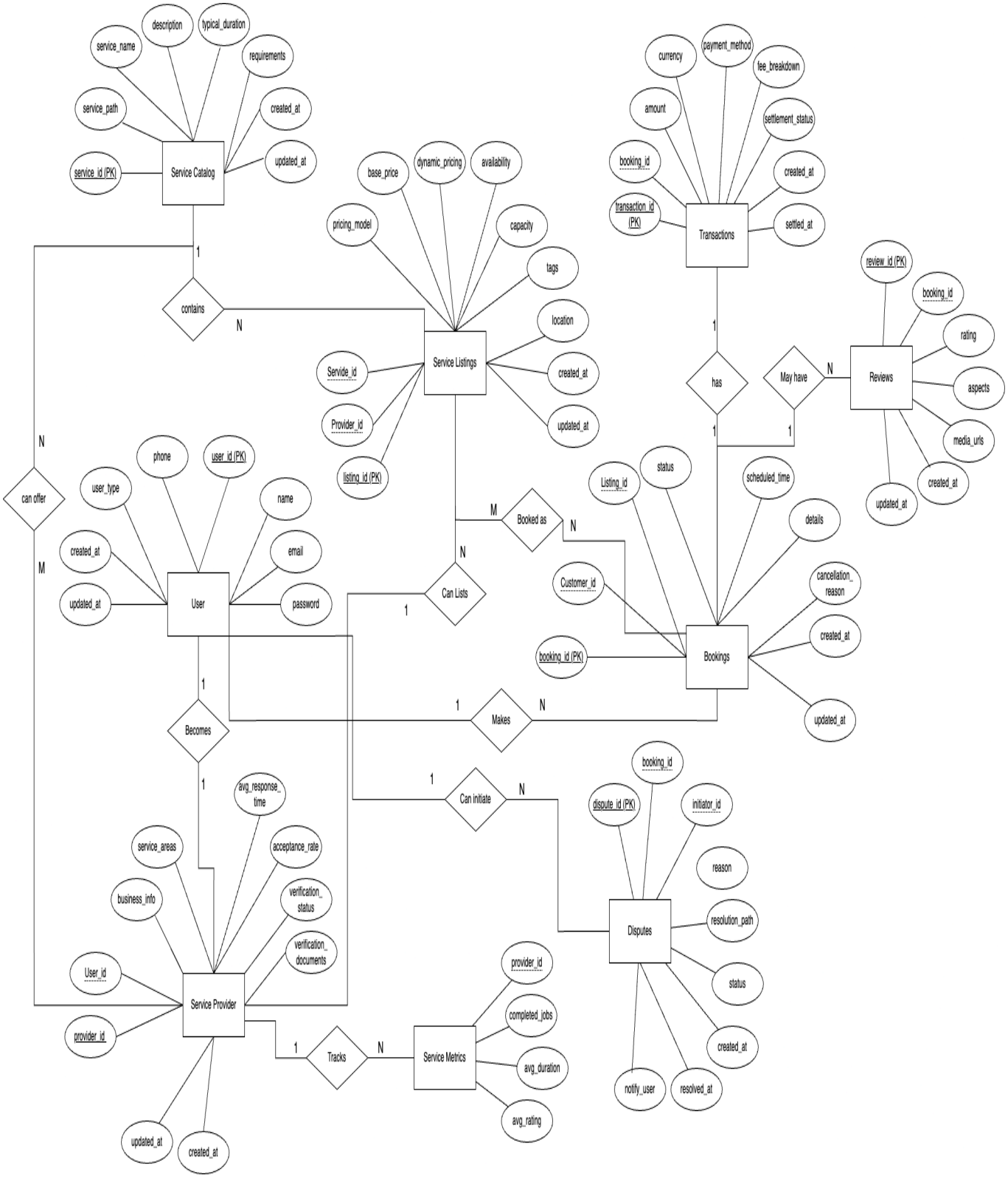
Online service booking - 0



Level 1



3.5 ER Diagram


Service Catalog
service_id (PK)
service_name
description
typical_duration
requirements
service_path
created_at
updated_at
contains
Service Listings
Servide_id
Provider_id
listing_id (PK)
base_price
dynamic_pricing
availability
pricing_model
capacity
tags
location
created_at
updated_at
Transactions
transaction_id (PK)
booking_id
currency
payment_method
fee_breakdown
amount
settlement_status
created_at
settled_at
has
May have
Reviews
review_id (PK)
booking_id
rating
aspects
media_urls
created_at
updated_at
can offer
User
user_id (PK)
phone
user_type
name
email
password
created_at
updated_at
Booked as
Can Lists
Bookings
booking_id (PK)
Customer_id
Listing_id
status
scheduled_time
details
cancellation_reason
created_at
updated_at
Becomes
Makes
Service Provider
provider_id
User_id
avg_response_time
service_areas
acceptance_rate
business_info
verification_status
verification_documents
Tracks
Service Metrics
provider_id
completed_jobs
avg_duration
avg_rating
Can initiate
Disputes
dispute_id (PK)
booking_id
initiator_id
reason
resolution_path
status
created_at
resolved_at
notify_user

2 Other Non-functional Requirements

4. Other Non-functional Requirements

4.1 Performance Requirements

1. The platform shall support simultaneous bookings from up to 500 users per minute without performance degradation.

2. The response time for searching services and providers shall not exceed 2 seconds under normal load conditions.

3. Notifications for booking confirmations shall be sent to both customers and service providers within 5 seconds of successful payment.

4. The system shall maintain an uptime of 99.9%, allowing only 1 hour of downtime per month for maintenance.

5. Payment transactions using Razorpay shall be processed within 3 seconds to ensure a smooth user experience.

6. The platform shall handle 5,000 concurrent users without system failure or slowdown.

4.2 Safety and Security Requirements

1. JWT-based authentication shall be implemented to ensure secure user logins and sessions.

2. Service providers must pass a verification process before being allowed to list their services, reducing fraudulent activities.

3. Communication between clients and providers shall be end-to-end encrypted to maintain privacy and prevent unauthorized access.

4. The platform shall implement rate limiting and CAPTCHA verification to prevent bot attacks and brute-force attempts.

5. All user data, including transaction history and personal details, shall be backed up daily to prevent data loss.

6. In the event of a security breach, affected users shall be notified within 24 hours, and necessary measures will be taken immediately.

4.3 Software Quality Attributes

4.3.1 Reliability

The system shall maintain 99.9% uptime to ensure uninterrupted service availability.

A failover mechanism will be implemented to ensure the platform remains functional in case of server failures.

4.3.2 Scalability

The platform shall be designed with microservices architecture, allowing horizontal scaling as user demand increases.

The database queries and API endpoints shall be optimized to handle large numbers of concurrent requests efficiently.

4.3.3 Maintainability

The codebase shall follow modular development principles, ensuring ease of debugging and future updates..

Comprehensive API documentation shall be maintained for easy integration of new features.

4.3.4 Security & Privacy

Role-Based Access Control (RBAC) shall be implemented to restrict access to sensitive data and operations.

Passwords shall be hashed using decrypt before storage to prevent unauthorized access.

The system shall enforce automatic session expiration after a period of inactivity.

4.3.5 Usability

The platform shall have a responsive UI, ensuring a seamless experience across mobile and desktop devices.

The booking process shall require no more than 3 clicks, improving user experience and engagement.

Real-time notifications and reminders shall keep users updated about service bookings and payments.

This ensures that Fixly remains fast, secure, scalable, and user-friendly, while maintaining high security and reliability standards.

5. Data Dictionary

1. User Table:

Field Name	Data Type	Constraints	Description
UserID	INT	PRIMARY KEY	A unique identifier for each user.
UserType	VARCHAR(50)	CHECK (UserType IN ('Customer', 'Admin', 'ServiceProvider'))	Specifies the type of user: 'Customer', 'Admin', or 'ServiceProvider'.
Username	VARCHAR(100)	UNIQUE, NOT NULL	A unique username for the user.
Password	VARCHAR(255)	NOT NULL	The password for the user.
FirstName	VARCHAR(100)	NOT NULL	The first name of the user.
LastName	VARCHAR(100)	NOT NULL	The last name of the user.
Email	VARCHAR(100)	UNIQUE, NOT NULL	A unique email address for the user.
Phone	VARCHAR(15)	UNIQUE, NOT NULL	A unique phone number for the user.
Address	TEXT		The address of the user.
RegistrationDate	DATETIME	DEFAULT CURRENT_DATE	The date and time when the user registered.
ProfilePicture	TEXT		A URL or path to the user's profile picture.

2. Service Provider Table:

Field Name	Data Type	Constraints	Description
ServiceProviderID	INT	PRIMARY KEY	A unique identifier for each service provider.
UserID	INT	REFERENCES User(UserID) ON DELETE CASCADE	Links the service provider to a user.
ServiceCategory	VARCHAR(100)	NOT NULL	The category of services provided.
ServiceDescription	TEXT		A description of the services provided.
Availability	BOOLEAN	DEFAULT TRUE	Indicates if the provider is available (TRUE) or not (FALSE).
Rating	DECIMAL(3,2)	DEFAULT 0	The rating of the service provider (out of 5).
VerificationStatus	VARCHAR(50)	CHECK (VerificationStatus IN ('Pending', 'Verified', 'Rejected'))	The verification status of the service provider.

3. Service Category Table:

Field Name	**Data Type**	**Constraints**	**Description**
CategoryID	INT	PRIMARY KEY	A unique identifier for each service category.
CategoryName	VARCHAR(100)	UNIQUE, NOT NULL	The name of the service category.
CategoryDescription	TEXT		A description of the service category.

4. Service Listing Table :

Field Name	Data Type	Constraints	Description
ListingID	INT	PRIMARY KEY	A unique identifier for each service listing.
ServiceProviderID	INT	REFERENCES ServiceProvider(ServiceProviderID) ON DELETE CASCADE	Links the listing to a service provider.
CategoryID	INT	REFERENCES ServiceCategory(CategoryID) ON DELETE SET NULL	Links the listing to a service category.
ServiceTitle	VARCHAR(255)	NOT NULL	The title of the service being offered.
ServicePrice	DECIMAL(10,2)	NOT NULL	The price of the service.
ServiceDetails	TEXT		A detailed description of the service.
ServiceImage	TEXT		A URL or path to the image of the service.

5. Booking Table:

Field Name	Data Type	Constraints	Description
BookingID	INT	PRIMARY KEY	A unique identifier for each booking.
CustomerID	INT	REFERENCES User(UserID) ON DELETE CASCADE	Links the booking to a customer.

ServiceProviderID	INT	REFERENCES ServiceProvider(ServiceProviderID) ON DELETE SET NULL	Links the booking to a service provider.
ServiceListingID	INT	REFERENCES ServiceListing(ListingID) ON DELETE CASCADE	Links the booking to a service listing.
BookingDateTime	DATETIME	DEFAULT CURRENT_DATE	The date and time when the booking was made.
ServiceDateTime	DATETIME	NOT NULL	The date and time when the service is scheduled.
BookingStatus	VARCHAR(50)	CHECK (BookingStatus IN ('Pending', 'Confirmed', 'Completed', 'Cancelled'))	The status of the booking.
SpecialInstructions	TEXT		Special instructions for the service provider regarding the booking.

6. Payment Table:

Field Name	Data Type	Constraints	Description
PaymentID	INT	PRIMARY KEY	A unique identifier for each payment.
BookingID	INT	REFERENCES Booking(BookingID) ON DELETE CASCADE	Links the payment to a booking.
PaymentAmount	DECIMAL(10,2)	NOT NULL	The total amount of the payment.

PaymentMethod	VARCHAR(50)	CHECK (PaymentMethod IN ('Credit Card', 'Debit Card', 'UPI', 'PayPal'))	The payment method used by the customer.
PaymentDateTime	DATETIME	DEFAULT CURRENT_DATE	The date and time when the payment was made.
PaymentStatus	VARCHAR(50)	CHECK (PaymentStatus IN ('Pending', 'Completed', 'Failed'))	The status of the payment.

7. Rating & Review Table :

Field Name	Data Type	Constraints	Description
ReviewID	INT	PRIMARY KEY	A unique identifier for each review.
BookingID	INT	REFERENCES Booking(BookingID) ON DELETE CASCADE	Links the review to a booking.
CustomerID	INT	REFERENCES User(UserID) ON DELETE SET NULL	Links the review to a customer.
Rating	DECIMAL(2,1)	CHECK (Rating BETWEEN 1 AND 5)	The rating given by the customer (between 1 and 5).
ReviewText	TEXT		The written review text left by the customer.
ReviewDateTime	DATETIME	DEFAULT CURRENT_DATE	The date and time when the review was submitted.

8. Service Category Table:

Field Name	Data Type	Constraints	Description
ComplaintID	INT	PRIMARY KEY	A unique identifier for each complaint.

CustomerID	INT	REFERENCES User(UserID) ON DELETE CASCADE	Links the complaint to a customer.
BookingID	INT	REFERENCES Booking(BookingID) ON DELETE SET NULL	Links the complaint to a booking.
ComplaintDateTime	DATETIME	DEFAULT CURRENT_DATE	The date and time when the complaint was made.
ComplaintText	TEXT		The description of the complaint made by the customer.
ComplaintStatus	VARCHAR(50)	CHECK (ComplaintStatus IN ('Pending', 'Resolved', 'Rejected'))	The status of the complaint: 'Pending', 'Resolved', or 'Rejected'.

9. Report Table :

Field Name	Data Type	Constraints	Description
ReportID	INT	PRIMARY KEY	A unique identifier for each report.
AdminID	INT	REFERENCES User(UserID) ON DELETE SET NULL	Links the report to an admin.
ReportType	VARCHAR(100)	NOT NULL	The type of report generated by the admin.
GeneratedAt	DATETIME	DEFAULT CURRENT_DATE	The date and time when the report was generated.
ReportData	TEXT		The content or data of the generated report.

6.GROUP LOG

Agenda	Task Description	Time Commitment
Finalized core system features	Initial research on service booking platform requirements	1 day
Product analysis	Overall Description of product & its functionality	4 hour
Refined system scope	Brainstormed functional and non-functional requirements	5 hour
Creating System Diagrams	Created use case diagram & DFD & ER Diagram	4 hour
Ensured completeness and accuracy of the interface requirements	Added External Interface Requirement	3 hour