Sir Syed University of Engineering and Technology

Software Engineering Department

Introduction to Software Engineering – Practical

Software Requirements Specification

Version v1.0

Project Title: Dehari – Home Based Services Booking App.

Team Members:

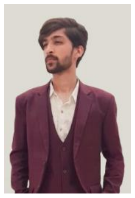
Name	Roll Number	Role
Abdul Rehman (Leader)	2024F-BSE-323	Whole Project Developer
 Sarim Mustafa	2024F-BSE-080	Database Designer and Data Collector
Syed Faiz Alam	2024F-BSE-085	Project Designer

Table of Contents

3.3 Use Case Descriptions

Provide use case descriptions with actors, steps, and alternative flows.

3.4 System Features

Describe each system feature and its priority.

3.5 Diagrams

- Use Case Diagram
- System Flow Diagram (Flow Chart)
- Class Diagram
- Sequence Diagram
- Mind Mapping (diagram)

5. External Interface Requirements

4.1 User Interfaces

Screens, menus, forms (use mockups or descriptions)

4.2 Hardware Interfaces

Interaction with hardware components

4.3 Software Interfaces

Interaction with other software components

4.4 Communication Interfaces

Protocols and methods for external communication

6. Other Non-functional Requirements

5.1 Performance Requirements Speed, response time, etc.

5.2 Security Requirements Login, data protection, etc.

5.3 Quality Attributes

Maintainability, usability, portability, etc.

7. Appendices

Additional information, forms, surveys, mock data, etc.

1. Introduction

1.1 Purpose

This Software Requirements Specification (SRS) document is created for the Dehari – Home Based Services Booking App. It explains about the system, the features of the system, and how users can use it. This document is for students, developers, supervisors, and evaluators to help them understand what the system is supposed to do and how it perform.

The main goal of the app is to let people find and book home service providers. for example electricians, plumbers, and other skilled workers. Rather than searching around or asking for recommendations, users can just open the app, browse services, and book directly.

The purpose of this document is to:

- **This document will explain the goal of the project:**
  Dehari app is developed to help users find local service providers like electricians, plumbers, and other professionals for home-based services.

- **Shows the overall app's functionality:**
  It explains all the main features like login, booking services, service provider registration, and the admin panel for managing users and services.

- **Easy to understand the project Dehari:**
  If you are coding, testing, or even reviewing, this document will help you understand everything.

- **It include both functional and non-functional requirements:**
  You will find what the system should do, how it should behave, and the conditions it should meet (like speed, usability, and security).

- **Guide for future updates or improvements:**
  If someone wants to update/improve the app later or fix bugs, this document will help them understand the original design and every logic behind it.

At the end, this document will help you clear everything. It gives you the full guidance of how the app works, what's inside it, and how different users interact with it so you can build it, test it, or present it without confusion.

1.2 Scope

The *Dehari – Home Based Services Booking App* is a desktop-based system that helps users easily connect with local home service providers. It is designed to make the long process of finding skilled professionals like electricians, plumbers, and cleaners easier, faster and more convenient. Dehari app cuts all hassle of finding service providers on social media or doing phone calls. with Dehari, they can just open the app, browse service categories, view available service providers, and make a booking all in one place.

The Dehari app is built using Java Swing, MySQL database, and NetBeans IDE. It also uses FlatLaf to give the interface a clean, modern look. The app runs on desktop environments and does not support mobile or web browsers.

For Regular Users (Customers):

- Sign up and log in securely
- Browse multiple types of home services
- See service providers displayed in separate cards under each category
- Click on a service to view more details about the providers
- Add personal information when booking a service
- See a full list of all past bookings in the "Orders" page
- Book services without calling or meet the provider first

For Service Providers:

- Register on the platform by filling out a detailed form
- Enter information like service type, city, experience, and skills
- Upload a profile picture and description to showcase their work
- Appear in user searches based on the selected service type
- Get booked by customers directly through the app

For Admins:

- Log in using a valid admin account
- Access a separate admin panel with more controls
- View a dashboard showing:
    - Total number of registered users
    - Total booked services
    - Overall profit earned from service provider fees
-
- Perform CRUD operations (Create, Read, Update, Delete) on these sections:
    - Registered users
    - Service providers
    - Booked services

- Monitor how much is earned per booking
  (Users are charged 500 PKR per service, and the platform earns 150 PKR from each service provider around 30% commission)
- View all users and providers in the system
- Control system access (e.g., block users, approve providers)
- Admin access is account-based regular users cannot access admin pages

System Capabilities:

- User roles: Admin, Service Provider, Customer
- Real-time booking history saved in the database
- Admin-only access to service management
- Desktop-only interface with form validation
- Simple UI with focus on functionality and ease of use
- MySQL used to manage all stored data (users, services, bookings)

The system does not include payment integration, chat features, or map-based service tracking. Its focus is on service browsing, booking, and role-based access control. This app is built mainly for learning and demonstration purposes, but the logic is strong enough to scale if needed later.

1.3 Definitions, Acronyms, and Abbreviations

This section explains the short forms and terms used throughout the document. If you're reading about something and aren't sure what it means, check here first.

Term/ Acronym	Meaning
SRS	Software Requirements Specification – this document
GUI	Graphical User Interface – the front-end of the app that users interact with
CRUD	Create, Read, Update, Delete – the basic functions used to manage data
MySQL	A database used to store all the data like users, services, and bookings
IDE	Integrated Development Environment – we used NetBeans to write and test code
PKR	Pakistani Rupees – the currency used in payments and service charges

Admin	A user with full access to the system. Can manage users, bookings, and more
User / Customer	Someone who uses the app to book a service
Service Provider	A person offering a home-based service like electrician, plumber, etc.
Booking	When a customer selects a provider and requests their service
FlatLaf	A library used to make the app's GUI look modern and clean
Dashboard	The admin's homepage that shows totals like users, bookings, and profits
Role-based Access	Different parts of the app are only shown based on who's logged in
Access Denied	Message shown when someone tries to access an area they're not allowed to
Service Card	A visual block showing each service (with image, name, etc.) in the UI
Order History	A list of all bookings made by the user
Timestamp	Date and time saved for bookings, registrations, and updates
Backend	The database and logic working behind the scenes (not visible to the user)
Frontend	The part of the app the user can see and interact with

1.4 References

Here are the main sources and tools used while planning and developing the Dehari – Home Based Services Booking App:

- Java official documentation for understanding core Java and Swing components
  https://docs.oracle.com/javase/8/docs/
- MySQL documentation for database design, queries, and relationships
  https://dev.mysql.com/doc/
- NetBeans IDE used for coding and building the GUI
  https://netbeans.apache.org/
- FlatLaf used to style the desktop application interface
  https://www.formdev.com/flatlaf/

- Stack Overflow for debugging help and coding solutions https://stackoverflow.com
- YouTube tutorials for building Java Swing applications and database connectivity
- Class notes and instructor guidance used for understanding how to structure the SRS and follow university format

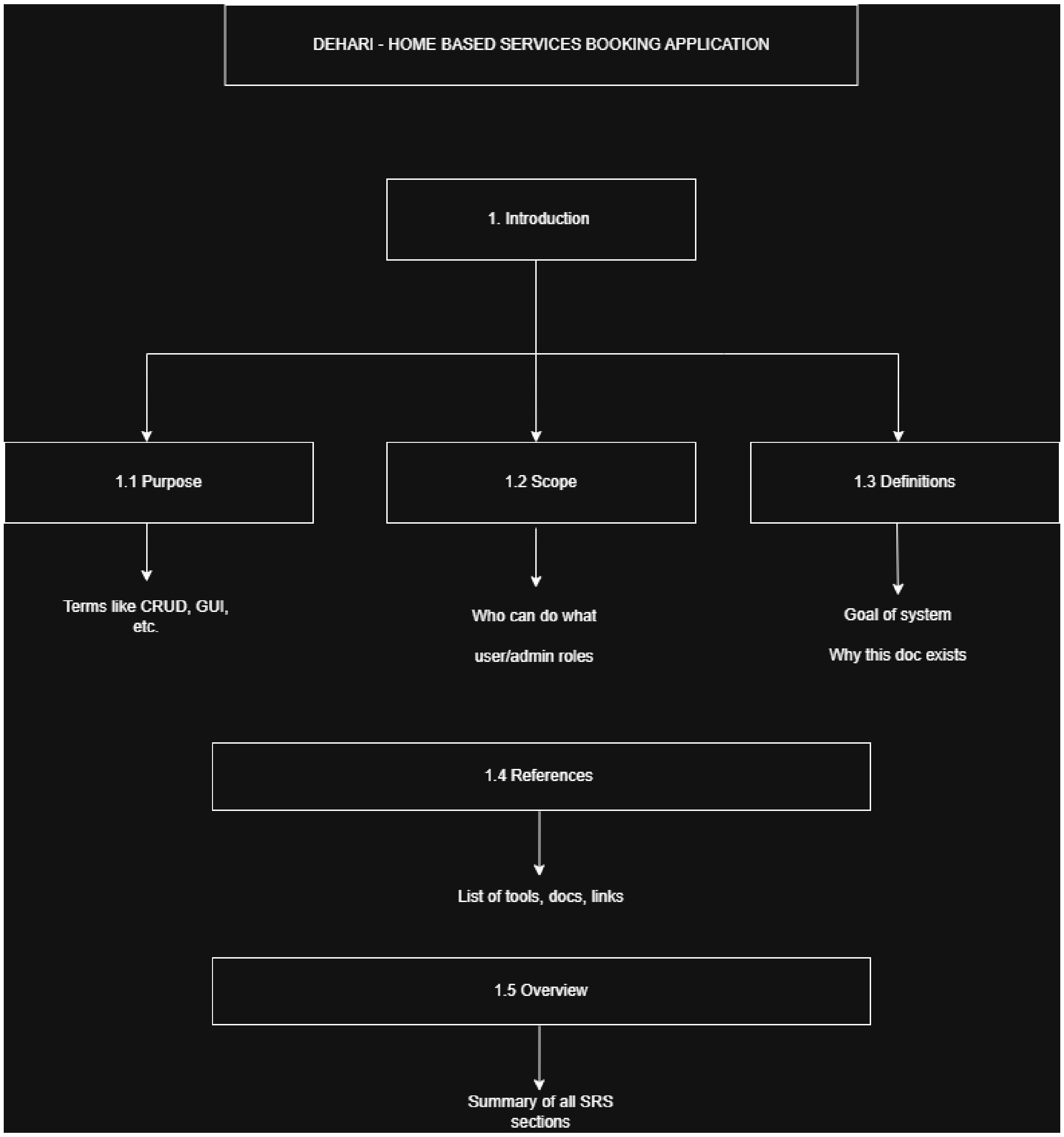
1.5 Overview

This Dehari app document has different sections to explain the app step-by-step. Every part focus on something specific like how the Dehari app works, who can use it, what features are available, and what the system should look like and behave like.

Here is a quick look at what you will find in the coming sections:

- **Section 1 – Introduction:**
  Gives an idea about what this app is, who is the purpose of this app, and why the document is so important.
- **Section 2 – Overall Description:**
  In this section we will see who the users are, what the system will do, what tools were used, and any design limits.
- **Section 3 – Scheduling:**
  This section has Gantt chart, network diagram, resource sheet, and other important diagrams.
- **Section 4 – Specific Requirements:**
  Breaks down functional and non-functional requirements, use cases, and diagrams like use case and class diagrams.
- **Section 5- External interface requirements:**
  Demonstrates how the system communicates with the user, the hardware and software (and screens, forms and backend connection).
- **Section 6 Other Non functional Requirements:**
  Concentrates on matters such as performance, security and quality of systems.
- **Section 7 - Appendices:**
  Additional detail such as, mockups, sample information and glossary.

This arrangement can make you have a clear mind on every increment of the project; idea to rolling system.



BLOCK FLOW DIAGRAM

2. Overall Description

2.1 Product Perspective

Dehari- Home Based Services Booking App will be constructed as a desktop application. It is made to run on windows and it is not dependent on a broader software or some independent platform. It possesses even in-built functions and has a local database based on MySQL.

These are more like fully functional mini-systems (in a sense they do everything by themselves such as logging in the users as well as handling reservations within one single desktop program).

This is what the pieces interlock:

- **Frontend (User interface):**
  In this application we have all the screen, forms and navigation created in Java Swing. To enhance the looks and make it modern and clean FlatLaf has been used.

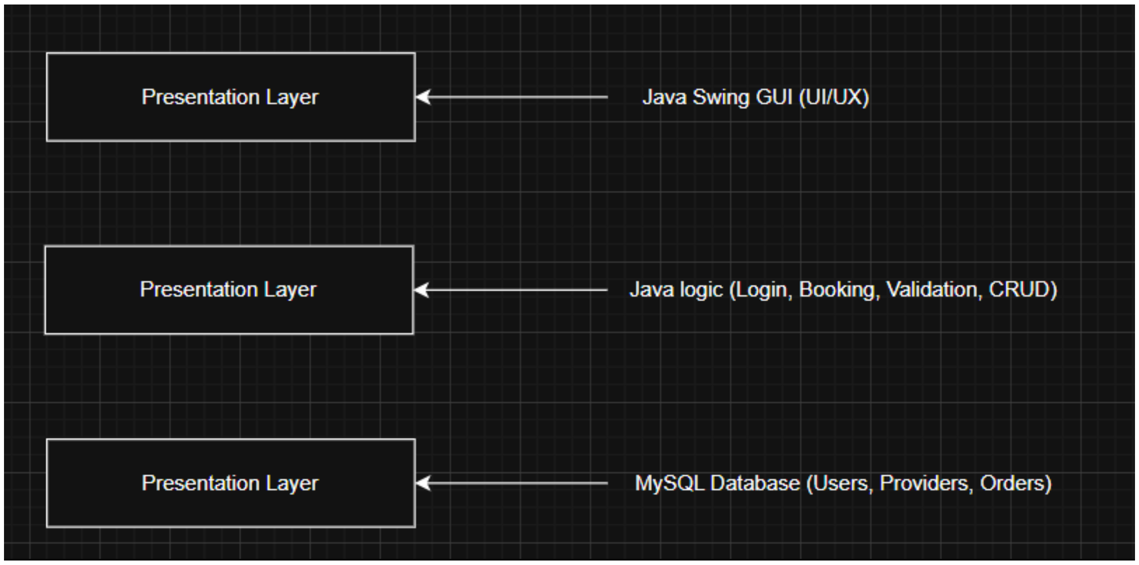
- **Back End (Logic/ Data Handling):**
  Java takes care of all the operations such as logging in, validation of the user, booking of the services, saving provider details, or all this.

- **Database (MySQL):**
  All the stuff is held here users, providers, bookings, reviews, and others. It is completely integrated to the app, and is updated on real-time with reference to the activities made by the user.

- **Type of system:**
  - It is the 3-user role system:
    - Customer/User Bounty Books services
    - Service Provider- Registers and provides services
    - Admin-handles users and services as well as views system statistics

- **Installation:**
  All you require is to have NetBeans and MySQL up and running. There is no requirement of third party cloud or web server.

The system is specific such that it is easy to use in local environments such as labs, demonstration environments or client applications running on a desktop. It does not requisite internet after installing it.


Presentation Layer
Java Swing GUI (UI/UX)
Presentation Layer
Java logic (Login, Booking, Validation, CRUD)
Presentation Layer
MySQL Database (Users, Providers, Orders)

3-Layer Architecture Diagram

2.2 Product Functions

Dehari app is designed to ensure that anyone can find and book the local service providers through his or her desktop with ease. It has three types of users i.e. customers, service providers and admins and each of them have different sets of features.

The possibilities of each role in the system are the following:

1. **Customers / Regular Users:**

   These are individuals who wish to hire a house service, say plumber, an electrician, or any other local aid.

   - **Browse Services:**
     - The users are able to view a combination of various service providers.
     - The cards profile the basic information of name, type of a particular service; rating and tiny profile of each card of services.

   - **See Provider Information:**
     - On clicking on a card, detailed information of that service provider appears: experience, location, skills and contact.

- **Reserve a Service:**
    - The user applies their choice of provider and provides their details as well as confirms a booking.
    - The booking is logged in the database together with provider and user details.

- **See Previous Bookings:**
    - All their past bookings with dates and prices information are available on the page named "Orders".

- **Log in and Profiles:**

    - Every user possesses his or her account. They will be able to sign in using security token, edit profile information, and maintain own personal information.

2. **To the Service Providers:**

   These are the employees or the professionals who will give the home-care services.

   - **Declaration of Provider:**
     The providers have an option of registering themselves by completing all their information - name, contact, type of skills, and services to be provided, experience, etc.

   - **Manage Profile:**
     They may post a profile photo, say something about work, and change service details.

   - **Be Booked by User:**
     Upon registration, the providers appear in the list of services that the user can book.

3. **For Admin:**
   On the management side of the app, admins are officials who have the entire control.

- **Dashboard Access:**
  - Admin dashboard reveals the sum of the users, sum of the bookings and sum of the profits.
  - It also shows system data such as:
    - The number of users in total
    - The total amount of booked services
    - The amount of total profit gained (system keeps 150 PKR per service)

- Manage Users:
  - Admins have the possibility of seeing all users, updating their information, and removing the accounts when necessary.

- **Control Service Providers:**
  - Admins will be able to view the profile of a provider, confirm the details, and delete or ban providers in case of the need.

- **Manage Bookings:**
  - The admin can see all of the bookings with the names of the users, the names of the providers, prices and dates.
  - Admin would be able to delete or modify records when it is necessary.

- **Access Control:**
  - Several features of how only administrators have access are restricted to regular users.
  - When a user, lacking the privileges of an administrator, chooses to view pages of the administrator, the system generates a response of: Access Denied.

Each of the functions is directly linked with the database, thus, each of the actions (booking, profiles editing, or deleting users, etc.) are instantly saved and displayed throughout the application.

In such a configuration, the app is practical, accessible, and friendly to work with too, regardless of the user booking a service, a provider with your skills to offer, or an administrator keeping everything in order.

2.3 User Classes and Characteristics

There are three major classes of users in your system. Each of them has various accessibility, opportunities, and roles within the app.

1. **Regular User / Customer**

   Such are regular users that are willing to book a residential service.

   - **Main Characteristics:**
     - Is able to visit suppliers of other servicesAvailable providers can be browsed
     - Is able to read provider profiles that are comprehensive utilized info
     - Have the capability of booking the services by filling their personal guide of information
     - They can log in, manage personal profile, and check booking history
     - Unable to log in the admin panel and edit system data

   Example:
   Ali requires electrician at home. He logs in the application, looks at several profiles, reserves a service, and screens the status of his orders.

2. **Service Provider**

   The users provide services on the platform. They are such as freelancers or local experts.

   - **Main Characteristics:**
     - Even can be registered with details such as name, experience, skills and city etc.
     - Users can book it
     - Are able to update their profile, add services and demonstrate their availability
     - Unable to view other providers and bookings as well as accessing their dashboard
     - Show up in the search results depending on the services they provide

   Example:
   Adeel is a 5 years experienced plumber. He makes his own profile using his skills and begins to get bookings using Dehari.

3. Admin

   The admins are equivalent to system managers. Everything is under their control.

- **Main Characteristics:**
  - Is able to log in to a special dashboard
  - Has the ability to see the number of users, the number of bookings and profit (150 PKR per booking)
  - Add, modify, or destroy any user, provider or booking
  - Has the ability to block or ban service providers in case of a necessity
  - Not able to be mimed by other common users (admin access is denied)
  - A non-admin seeing the admin URL gets the Access Denied page

Example:
Sara is on the administration. She sees the current profit, which services are used most, creates a user, removes a duplicate user, and monitors all the last bookings.

The system is divided into classes of users which are separated by roles and access. The app identifies you to allow you access to some areas. This makes the system safe and tidy.

2.4 Operating Environment

Dehari app is a Java program, thus the desktop client requires just a couple of items to be operational. It is based on Windows PCs and linked to MySQL database. The system does not require the internet connection after being installed, as it operates entirely offline, when the local database and environment is prepared.

It is easy to execute it on a low-end laptop or desktop. It is also not heavy and it does not require premium specification.

For Users (Customers & Service Providers)
The GUI will engage people with Swing. It ought to be deployed as a .jar file or it can be run using an IDE such as NetBeans (in a test environment).

They need:
- An operating system of Windows (Windows 7 or higher)
- Java Runtime Environment (JRE)
- Screen resolution (at least 1024768 on proper layout)

For Developers

- In case you are developing or re-designing the app, you will require:
- Java Development Kit (JDK 8 above)
- NetBenes IDE (suggested to use with GUI & MySQL integration)
- MySQL Server (best MySQL 5.7+ or MySQL Workbench)
- Modern UI skin (FlatLaf Look and Feel library)
- Local MySQL database connection MySQL database configuration

Admin (on same or another system)
The same app includes the admin panel, but requires a user to log in as an admin. The system requirements are thus similar to that of the normal users. One does not require a separate app.

Summary Software Stack
No external API and hosting are needed in the system. All the interactions are locally based, such that Java takes the logic and MySQL stores the data.

Component	Requirement / Tool	Notes
Operating System	Windows 7 / 8 / 10 / 11	Required for both users and developers
Java Runtime Environment	JRE 8 or above	Required to run the compiled application
Java Development Kit	JDK 8+	Required only for development
IDE	NetBeans	For coding, GUI building, and DB integration
GUI Framework	Java Swing + FlatLaf	Swing for frontend; FlatLaf for modern look
Database	MySQL Server 5.7+	Localhost setup; stores all users, bookings, providers
Database Tool	MySQL Workbench	For easy DB table viewing and testing
System Type	Standalone Desktop Application	Doesn't require web or cloud hosting
Screen Resolution	1024×768 or higher	To avoid layout break or scroll issues
Internet Connection	Not required after initial setup	All functions run locally after launch

2.5 Design and Implementation Constraints

Working on the creation of Dehari – Home Based Services Booking App, we were considered a few limitations. Such limitations influenced the design, development, and the use of technology that we could apply.

They are not always problems sometimes they are just rules or even standards by which development was governed.

Platform and Technology Limitations

- **Desktop Only:**
  The application will undoubtedly be desktop-only - no web and mobile compatibility.
- **Java Swing:**
  The choice of swing was made because of the time constraints and course demands (it was not considered probable to implement the project in JavaFX).
- **Backend: MySQL:**
  The database was restricted to MySQL that is to be set up locally. Neither cloud database at all nor ORM (such as Hibernate) were employed.
- **Styling: FlatLaf:**
  In the case of GUI look and feel, modernizing of UI was done using FlatLaf only. No support of UI based on CSS as with JavaFX.

Development Environment Limits

- **You Need to Use NetBeans:**
  A choice of NetBeans was made because of some familiarity and because Swing GUI Builder could be used. IntelliJ and others were not taken into account.
- **Manual Layouting:**
  The layout of the GUI is accomplished primarily by handSwing components use beyond the Swing abstraction layerso that it is not easily scalable, much less responsive.
- No Frameworks or third party APIs:
  It does not use Spring, Maven, REST APIs and so on. It is not online, all the logic is written manually.

Functional & Security Limits

- **Hardcoded Role Based Access:**

  Roles of the user (admin, user, provider) are saved and verified manually based on a conditional logic - no role-based framework is implemented.

- **Access Control: Graphical User Interface (GUI)only:**

  The Admin functions can only be seen upon login, but no in-depth route security (as in web apps).

- **No Multi-user Live Access:**

  This being a regional desktop program, several users cannot access it simultaneously unless one puts it on a different system.

Performance and Conformance Limits

- **MySQL Localhost:**

  It is an app that is compatible with a local MySQL server. When it is not running the app does not start up properly.

- **Limitations on Java version:**

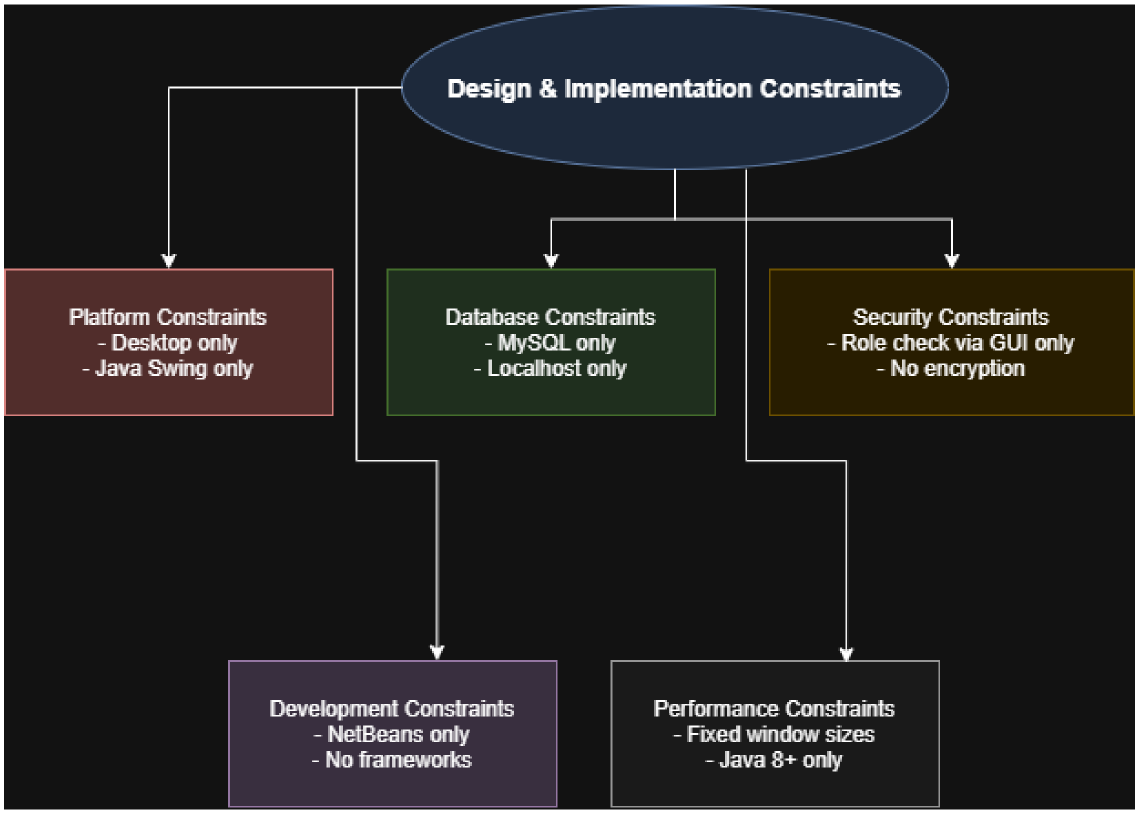
  The program was exercised using Java 8+. The use of older versions could bring about incompatibility.

- **Stable Window Sizes:**

  The screens are not responsive, they have a fixed size and in some lower resolutions, this may be a problem.

Constraint Type	**Details**
Platform Limitation	Desktop-only (no mobile/web version)
Programming Tools	Java, Swing, MySQL, NetBeans only
UI/UX Constraints	FlatLaf only, non-responsive fixed layout
No Third-Party Libraries	No REST APIs, Spring, Maven, or ORM used

Role Management	Simple condition-based role system (admin/user/provider)
Security Approach	GUI-based access control only
Database Limitation	MySQL only, requires manual setup
Internet Independence	System is fully offline after installation


Design & Implementation Constraints
Platform Constraints - Desktop only - Java Swing only
Database Constraints - MySQL only - Localhost only
Security Constraints - Role check via GUI only - No encryption
Development Constraints - NetBeans only - No frameworks
Performance Constraints - Fixed window sizes - Java 8+ only

Design and Implementation Constraints Diagram

2.6 User Documentation

Simple documentation is carried out to ensure that all the users understand how to utilize the Dehari system. The app is meant to be user friendly, regardless of whether you are the customer booking a service or the administrator who looks after the back-end operations. And you will not have to be a technical person in using it; the interface is easy to understand, and the buttons are marked correctly. to facilitate the process we have prepared certain documentation backup.

The table below provides breakdown of major actions that can be taken by Users (Customers & Providers) and Admins, what they do, and where you can find help regarding them:

User Documentation Table:

Role	Feature / Action	Description	Help Provided
Customer	Register Account	Sign up by entering name, email, and password	Registration form with basic validation
	Log In	Access your profile using email and password	Login form with error alerts
	Browse Services	View available service providers under categories like electrician, plumber	Cards shown on home screen
	View Provider Details	Click on a provider card to see their profile, experience, and skills	Detailed profile window popup
	Book a Service	Fill a form with name and contact info and confirm booking	Success message and order is saved
	View Past Orders	Access “Orders” page to check all previous service bookings	Orders listed with date and service info

Service Provider Documentation Table:

Feature / Action	What It Does	Help or Notes
Register as Service Provider	Fills a form with details like full name, email, password, contact info, and services offered	Includes dropdowns, text fields, and optional profile picture
Log In	Enters email and password to access provider dashboard	Shows error if credentials are wrong
View Own Profile	Sees the full profile info entered during registration	Automatically loaded from the database
Update Profile (if implemented)	Edits details like contact info, skills, or city	If feature exists, form with update button is provided
Track Bookings (optional)	View bookings made for their service (admin-visible by default)	This may depend on admin dashboard access or future enhancement

Admin Documentation Table:

Feature / Action	What It Does	Help or Notes
Log In as Admin	Accesses admin dashboard by entering valid admin credentials	Access denied for regular users
View Admin Dashboard	See summary of app activity: total users, total bookings, total profit	Dashboard loads automatically after login
View Earnings	Displays profit: 500 PKR per booking (user pays), 150 PKR goes to platform	This is shown in dashboard summary
Manage Users	View, edit, or delete customer accounts	Table with edit/delete buttons
Manage Service Providers	Add, update, or remove registered service providers	Form and table-based UI provided

View All Bookings	Lists all services booked by users, including provider and user details	Used for record keeping, support, or reporting
Access Control	Restricts admin panel to admin accounts only	Role check is handled in login logic

2.7 Assumptions and Dependencies

In this section we describe what we think will remain true during the time the system will be operated as well as what the app will require to operate correctly. This is not something that will be seen in the code however it is useful to know the environment the app was written in.

When such of them are modified, such as in the case of trying to run the app on a mobile device, or deploying it online, they will most likely need to change or add onto the system.

Assumptions
We are making some simple assumptions as to the use to which this app can be put:

- **Basic computer software are known to the users how to use them**

  It is an assumption that the users will be able to provide form filling, button clicking, and reading of labels. It does not require any special training besides simple use of mouse and keyboard.

- **The application will be a desktop or a laptop app**

  It is created using Java swing, therefore we are making an assumption that the users would open it on a computer. It is neither mobile phone or tablet-tested nor designed.

- **Java installation is already present (Java 8 or higher version)**

  The .jar file will require Java to be installed in the system of the user. Java 8+ is recommended since older versions of Java could not have supported some libraries.

- **The (MySQL) database is installed and operational on localhost**

  MySQL should be installed in the same system (localhost) that is running the app. It has no remote server - It connects to the local MySQL instance.

- **This system is applied into trusted/offline environment**

  As there is no high level of encryption neither the ability to log in on a web interface, we presume the usage of the app would be in a secure (personal) computing environment such as a lab.

- **The admins are added either manually or with the help of the database**

  The app does not have a button to create an admin. Editing of the MySQL database should be done to create an admin account.

- **The user obtains necessary files and folders**

  The required variants such as .jar executable, database .sql file, jdbc driver, and other assets (images, themes) are supposed to be packed and not absent.

- **There is no Multiple-user concurrency to be expected**

  The system is not designed to work on many users who would make bookings simutaneously. It is developed more like a personal or standalone app rather than a system at the enterprise level.

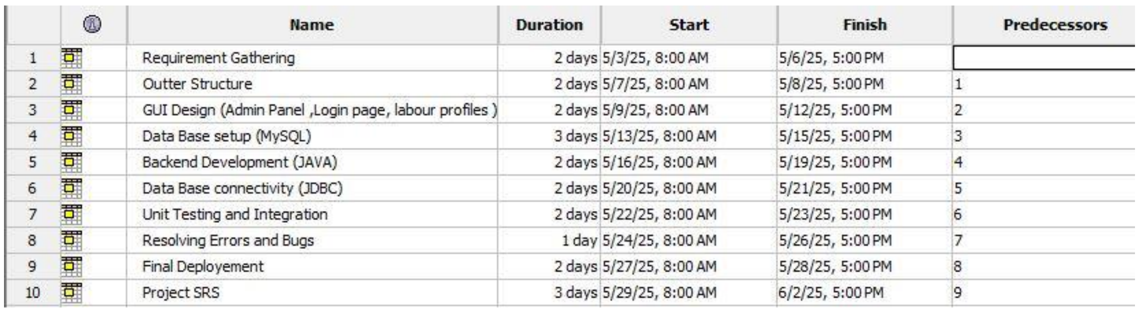
Dependencies:

Dependency	What It's For
Java SE Development Kit (JDK)	Required to compile, run, and build the Java Swing app (preferably Java 8 or above).
NetBeans IDE	Used to design the GUI, organize code, and run the project easily during development.
Java Swing	The core GUI framework used to build the entire interface — windows, buttons, forms, etc.
FlatLaf	External look-and-feel library to give the app a modern interface design (instead of default Java UI).
MySQL Server	Stores all data — users, service providers, bookings, and admin info — in a structured relational form.
MySQL Workbench or CLI	Optional tool to view, edit, and manage database tables (used during testing or debugging).

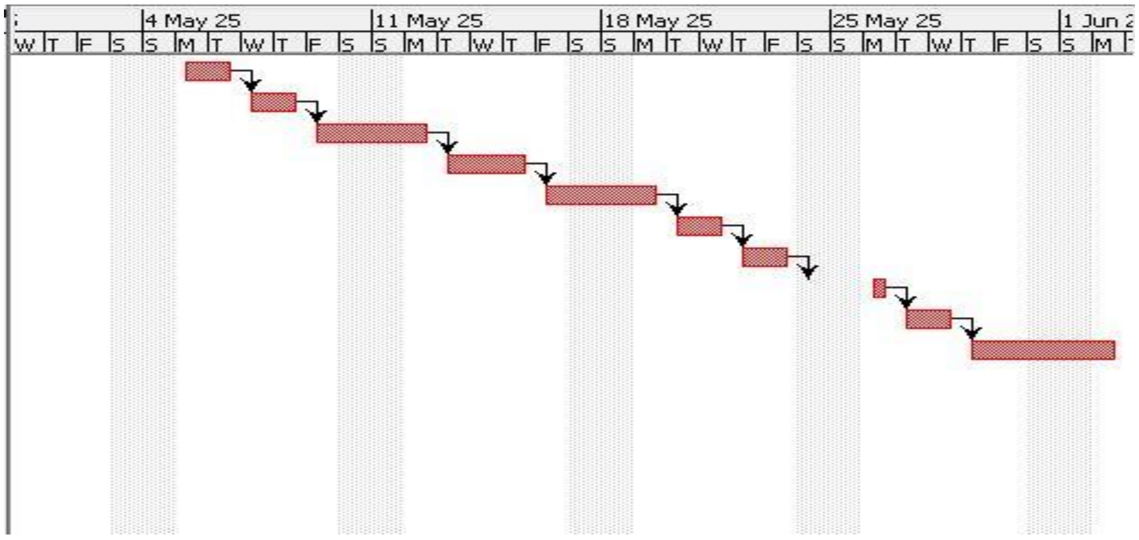
MySQL JDBC Connector	A `.jar` file that lets Java communicate with the MySQL database through JDBC.
Localhost (127.0.0.1)	Assumes the database is hosted on the same machine (no cloud or online DB).
Database Schema File (.sql)	The script used to create all required tables like `users`, `service_providers`, and `booked_services`.
Swing UI assets (optional)	Icons, images, and styling elements bundled in the app to make the UI cleaner and more user-friendly.

3. Scheduling (Gantt Charts, Resources)

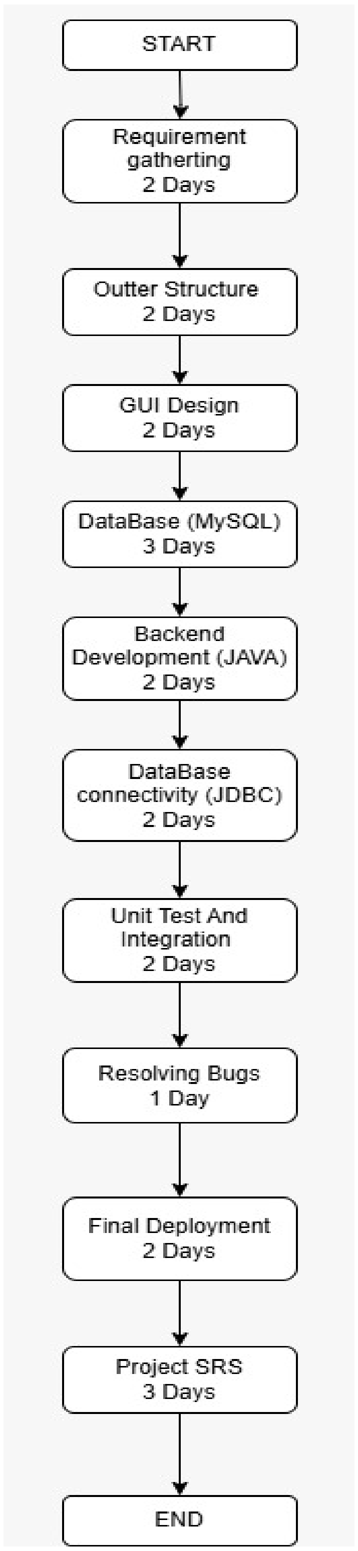
3.1. Gantt Chart of your Whole Project



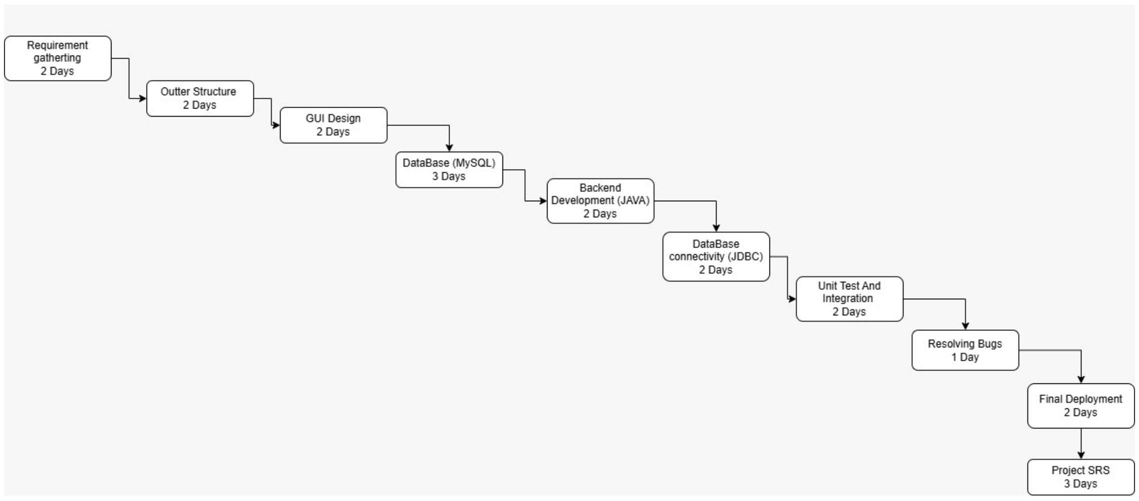
		Name	Duration	Start	Finish	Predecessors
1		Requirement Gathering	2 days	5/3/25, 8:00 AM	5/6/25, 5:00 PM	
2		Outter Structure	2 days	5/7/25, 8:00 AM	5/8/25, 5:00 PM	1
3		GUI Design (Admin Panel ,Login page, labour profiles)	2 days	5/9/25, 8:00 AM	5/12/25, 5:00 PM	2
4		Data Base setup (MySQL)	3 days	5/13/25, 8:00 AM	5/15/25, 5:00 PM	3
5		Backend Development (JAVA)	2 days	5/16/25, 8:00 AM	5/19/25, 5:00 PM	4
6		Data Base connectivity (JDBC)	2 days	5/20/25, 8:00 AM	5/21/25, 5:00 PM	5
7		Unit Testing and Integration	2 days	5/22/25, 8:00 AM	5/23/25, 5:00 PM	6
8		Resolving Errors and Bugs	1 day	5/24/25, 8:00 AM	5/26/25, 5:00 PM	7
9		Final Deployement	2 days	5/27/25, 8:00 AM	5/28/25, 5:00 PM	8
10		Project SRS	3 days	5/29/25, 8:00 AM	6/2/25, 5:00 PM	9

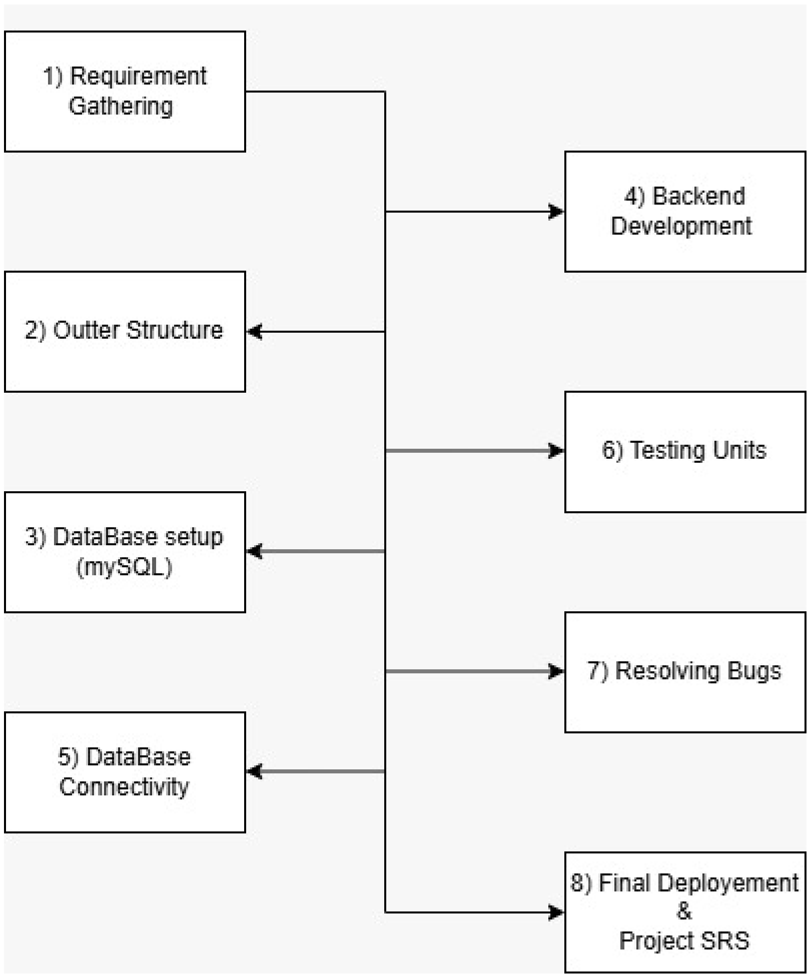


3.2. Pert Chart



3.3 Network Diagram


Requirement gatherting 2 Days
Outter Structure 2 Days
GUI Design 2 Days
DataBase (MySQL) 3 Days
Backend Development (JAVA) 2 Days
DataBase connectivity (JDBC) 2 Days
Unit Test And Integration 2 Days
Resolving Bugs 1 Day
Final Deployment 2 Days
Project SRS 3 Days



3.4. Resource Sheet

Resource Name	Type	Role / Description	Charges PKR (monthly)	Max. Availability (hrs/week)	Notes
Project Manager	Work	Oversees project timeline & coordination	10000	20 hrs	Handles communication & milestones
UI/UX Designer	Work	Designs app screens & user flow	8000	25 hrs	Wireframes & high-fidelity designs
Frontend Developer	Work	Builds UI using React Native / Flutter	8000	30 hrs	Mobile-focused frontend
Backend Developer	Work	Develops APIs, authentication, booking logic	9000	30 hrs	Node.js / Django / Laravel
QA Tester	Work	Tests app for bugs, performance, UX	6000	20 hrs	Manual and automated testing
DevOps Engineer	Work	Handles deployment, CI/CD, and server setup	5000	10 hrs	Optional if using managed platforms
Swing Frame work	Material	GUI Frame Work	-	-	-
SQL DataBase Server	Material	DataBase Software	-	-	-
Payment Gateway (API)	Material	Easypaisa/JazzCash/Bank-raast Transfer	-	-	Pay-per-transaction

4. Specific Requirements

4.1 Functional Requirements

These are the core things what the system should do.. They have been formatted in a user and system standpoint (they contain actions of customers, service providers, and admins). All the requirements are identified with their own individual ID that can be tracked easily.

User Functional Requirements

- **FR1: Registration of the user**

The system should enable a user to open an account and requires minimum information in order to open the account, e.g. full name, email address and password. The information is stored safely into the database.

- **FR2: Login of the user**

  Each registered user ought to be able to log into their email and password. Offering access should be preceded by credential checking by the system.

- **FR3: Services Newsfeed**

  The visitors will be able to see home-based services done such as electrician, plumber, carpenter, and so on are available. In every service there are several cards indicating various providers

- **FR4: See the Provider Profiles**

  Upon clicking a service card, a user is able to get comprehensive details of a service provider, including; experience, skills, ratings and contact information.

- **FR5: Schedule a Service**

  The users may make a reservation of a service by entering his or her personal information. When it is confirmed, a booking has been saved and the user receives a success message.

- **FR6: Past Orders**

Users have an opportunity to go in the “My Orders” page to see all their past bookings with the name of the provider and the type of service at the moment of booking.

Service Provider-Functional Requirements

- **FR7:Registration**

  Providers will have a chance to register with elaborate details on what they offer, the experience, place, and skills. A profile is made and stored.

- **FR8: Provider Log in**

  Service providers who have registered in the system will also be able to log in using their details to access or change their profile.

Admin Functional Requirements

- **FR9: Admin Login**

  Admins can log in independently utilizing their accreditations. As it were clients with admin rights can get to the admin dashboard.

- **FR10: Client Administration**

  The admin can see all clients, and has the capacity to upgrade or erase accounts on the off chance that required.

- **FR11: Supplier Administration**

  Admins can see, alter, or evacuate benefit suppliers from the framework utilizing the dashboard.

- **FR12: Bookings Administration**

Admins can see all bookings made by clients, with points of interest like client title, benefit supplier, benefit sort, and date.

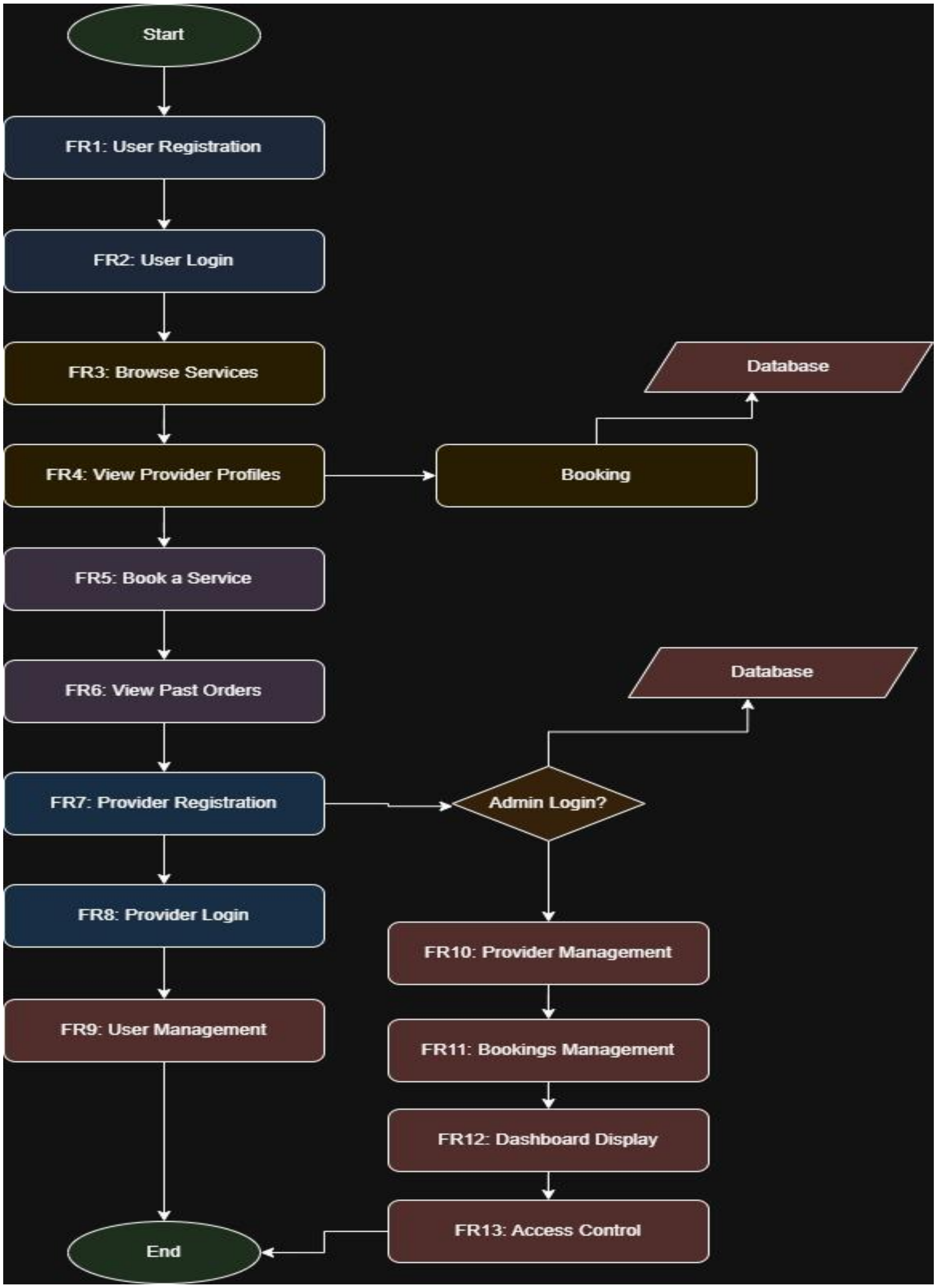
- **FR13: Dashboard Show**

The admin dashboard shows:

  - Add up to number of clients
  - Add up to number of booked administrations
  - Add up to benefit (calculated as 150 PKR per booking)

- **FR14: Get to Control**

  As it were clients stamped as admin within the database can get to the admin highlights. Others are denied get to.



System Flow Diagram

4.2. Non-functional Requirements

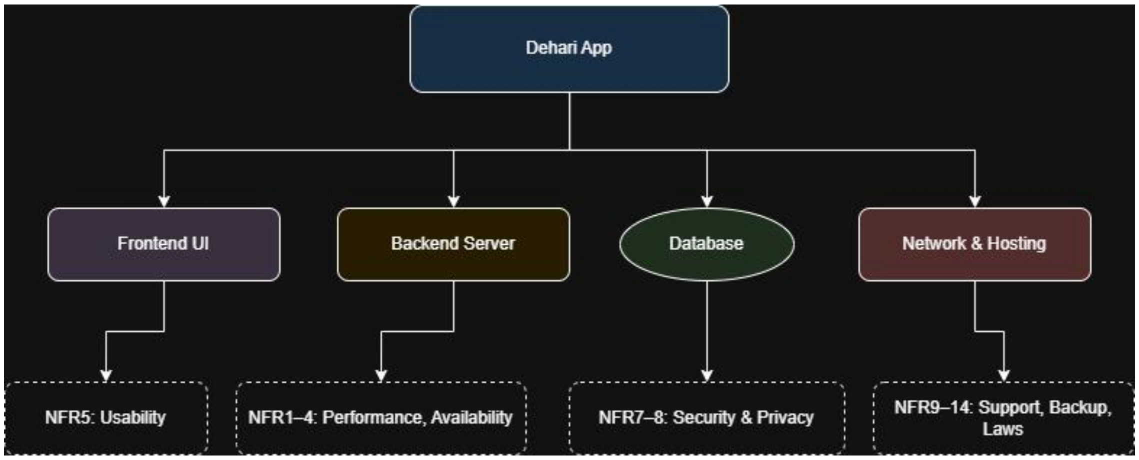
Non-functional necessities portray how the Dehari Home-Based Administrations Booking App ought to perform, not what it does. These components characterize the quality, behavior, and generally encounter of the framework. Indeed in the event that the app's highlights work, without great execution, security, and convenience — clients won't be cheerful.

We'll cover regions like speed, security, accessibility, practicality, and more.

Non-Functional Requirements Table:

ID	Category	Requirement Description
NFR1	**Performance**	The app should load any page within **3 seconds** under normal network conditions.
NFR2	**Scalability**	The system should support **at least 1,000 concurrent users** without crashing.
NFR3	**Reliability**	The system should have **99% uptime** during service hours.
NFR4	**Availability**	Services should be available **24/7**, except during scheduled maintenance.
NFR5	**Usability**	Users should be able to **book a service within 5 clicks or less** from the homepage.
NFR6	**Portability**	The system should run on **Windows**, **macOS**, and **Linux** via Java runtime.
NFR7	**Security**	All passwords must be **hashed and stored securely** using industry best practices.
NFR8	**Data Privacy**	Users' personal information should be **encrypted** in storage and during transmission.
NFR9	**Maintainability**	Codebase should be modular with **clearly commented methods** for easy updates.
NFR10	**Backup**	Daily **automated backups** must be taken of all bookings and user data.

NFR11	**Supportability**	System logs should be available to the admin for tracking any issues.
NFR12	**Error Handling**	Friendly error messages should be displayed when any issue occurs.
NFR13	**Localization**	App should support **English** now, and allow adding other languages easily later.


Dehari App
Frontend UI
Backend Server
Database
Network & Hosting
NFR5: Usability
NFR1–4: Performance, Availability
NFR7–8: Security & Privacy
NFR9–14: Support, Backup, Laws

Non-Functional Requirements UML diagram

NFR Mapping:

Component	**Connects to NFRs**	**NFR Summary**
Frontend	→ `NFR5: Usability`	Easy to use, book in a few clicks
Backend	→ `NFR1-4: Performance, Availability`	Fast, responsive, always available
Database	→ `NFR7-8: Security & Privacy`	Encrypt data, secure access
Network & Hosting	→ `NFR9-14: Backup, Logs, Compliance`	Logs, legal, backups, maintainability

Box Naming:

Box Name	Label in Diagram
Main System Block	`Dehari App`
Component 1	`Frontend UI`
Component 2	`Backend Server`
Component 3	`Database`
Component 4	`Network & Hosting`
NFR Group 1	`NFR1-4: Performance, Availability`
NFR Group 2	`NFR5: Usability`
NFR Group 3	`NFR7-8: Security & Privacy`
NFR Group 4	`NFR9-14: Support, Backup, Laws`

4.3 Use Case Descriptions

These utilize cases portray how clients connected with the framework. Each utilize case appears who is doing what, why, and what happens step by step — counting interchange streams and exemptions.

The most on-screen characters are:

Customer/User
Service Provider
Admin

Detailed Use Case Tables:

Use Case ID	UC1
Use Case Name	User Registration
Actor(s)	Customer, Service Provider
Preconditions	User is not logged in.

Description	The user fills out a registration form with name, email, password, etc., and submits it. System saves the info and creates an account.
Main Flow	1. User opens the registration page. 2. Fills in required fields (name, email, password, user type). 3. Clicks "Register". 4. System validates and saves data. 5. Account is created and user is redirected to login.
Alternate Flow	- If email is already registered, show error. - If required fields are empty, highlight and show message.
Postconditions	New account is created and stored in the `users` table.

Use Case ID	**UC2**
Use Case Name	User Login
Actor(s)	Customer, Service Provider, Admin
Preconditions	Account already exists.
Description	User provides email and password. System checks credentials and logs them in.
Main Flow	1. User opens login page. 2. Enters email and password. 3. Clicks "Login". 4. System checks credentials. 5. Redirects to respective dashboard (user/admin/provider).
Alternate Flow	- If wrong password, show error. - If user is banned or inactive, deny login and show message.
Postconditions	User is logged in and session is active.

Use Case ID	**UC3**
Use Case Name	Browse Services
Actor(s)	Customer
Preconditions	User must be logged in.
Description	User views service categories and list of service providers for each.

Main Flow	1. User logs in and navigates to services page. 2. Selects a service category. 3. List of providers with cards is shown. 4. User can click on a provider for full profile details.
Postconditions	User views service provider info from database.

Use Case ID	**UC4**
Use Case Name	Book a Service
Actor(s)	Customer
Preconditions	User is logged in and has browsed providers.
Description	User selects a provider, enters booking info, and places an order.
Main Flow	1. User selects provider. 2. Enters contact details. 3. Clicks "Book Now". 4. System stores booking in `booked_services`. 5. Confirmation message shown.
Alternate Flow	- If required info is missing, prompt user.
Postconditions	Booking stored with date/time in database.

Use Case ID	**UC5**
Use Case Name	View Booking History
Actor(s)	Customer
Preconditions	User is logged in.
Description	User checks their past service bookings.
Main Flow	1. User goes to "My Orders". 2. System fetches past bookings from `booked_services`. 3. List is shown in table format.
Postconditions	Booking history is shown.

Use Case ID	**UC6**
Use Case Name	Admin Dashboard
Actor(s)	Admin
Preconditions	Admin is logged in.
Description	Admin accesses dashboard to see total users, booked services, and profit.
Main Flow	1. Admin logs in. 2. System checks role. 3. Loads dashboard with stats: total users, booked services, and total profit (150 PKR per booking from provider).
Postconditions	Admin sees overview of platform activity.

Use Case ID	**UC7**
Use Case Name	Manage Users/Providers
Actor(s)	Admin
Preconditions	Admin logged in.
Description	Admin can view, delete, or update users and service providers.
Main Flow	1. Admin clicks "Manage Users/Providers". 2. System loads data. 3. Admin performs CRUD actions (Create, Read, Update, Delete).
Postconditions	Changes are reflected in database.

Use Case ID	**UC8**
Use Case Name	Service Provider Registration
Actor(s)	Service Provider
Preconditions	Provider does not already have an account.

Description	Provider signs up with extra info like skills, experience, service type.
Main Flow	1. Provider selects "Become a Provider". 2. Fills in form (skills, experience, etc.). 3. System stores profile in `service_providers`.
Postconditions	Provider profile is visible in service list.

Use Case ID	**UC9**
Use Case Name	Access Control
Actor(s)	System (for all users)
Preconditions	User is logged in.
Description	System checks user type (admin, provider, user) and shows allowed pages.
Main Flow	1. On login, user type is checked. 2. If user is admin → redirect to admin panel. 3. If user is normal user → go to services. 4. Unauthorized access shows “Access Denied”.
Postconditions	Users stay within their allowed area.

4.4 System Features

All of the features below are contain the essence of what your system is actually doing. These are the sections the users will be dealing with on a daily basis with the ability to book the services, to log in, to manage the accounts and so on. Admins themselves possess their set of controls to maintain things to order and running.

- **Registration & Login**
  The customer will be able to create an account by providing his simple data as name, email address and password. They have discretion of whether they want to be a regular user, a service provider or an admin (in special cases). After signing up, they are able to enter into the system with their details. The login system verifies the information supplied in the database and subsequently redirects the user to the right part of the application depending on their position.

- **Filter of Services and Providers**
  There are various services like plumbing, electrical work, etc. that the user can find after they log in. They are presented in the form of cards of services providers. In every card, there is a summary such as the name of the provider, the type of services provided, and the rating. Viewpeople may click on a card to get additional information such as experience, skills, city, and contact details. This enhances comparison and selection of who to book by the users.

- **Reserve a Service**
  Once the user has located a proper provider, he/she is able to proceed and make a booking. The booking is confirmed using several contact details (that are auto-filled in case the user is logged in). After this is submitted, it will be stored in the booked_services table and a message of confirmation should appear. It is fast, easy, and everything is within the application.

- **History Bookings**
  The users will have their history of all the previous bookings under them. This consists of such information as the name of the provider, the type of service, the cost and the date of the booking. The tracking of the previously used services including the follow-ups are also handy. This information is being directly retrieved via bookedservices table by the ID of logged-in user.

- **Registering of Service Provider**
  Individuals desiring to turn in their services (such as painters or electricians) may sign up individually as service providers. Registration form is a little bit longer and it requests additional data such as a type of services, skills, years of experience, etc. After this has been done their profile will appear to the user who enquires on the kind of service. Their information is stored in service_providers table.

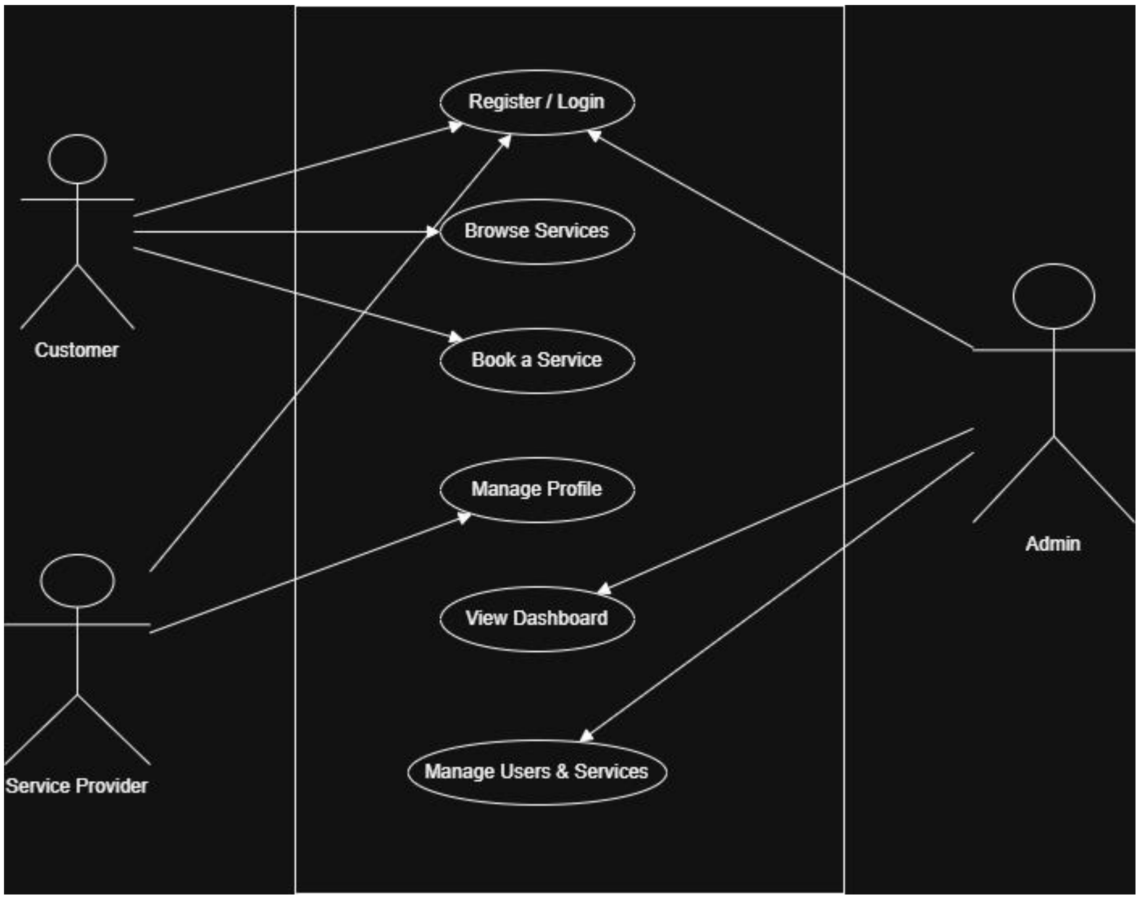
- **Role-Based Access Control**
  Each user will see a different aspect of the system depending on his type of a user (normal user, service provider, or admin). The frequent users will have an opportunity to search and make reservations. One can create his/her profile as a service provider. Dashboards and managing tools are available to admins. When a person attempts to access an area that he/she is not supposed to, the system intercepts the area and appears with the message of access being denied.

- **Admin Dashboard OVERVIEW**
  Once an admin logs in, he/she is redirected to a dashboard where he sees the big image of the platform. It displays the number of total users, the number of services booked as well as the profit realized by the system (at 150 PKR per booking). All these figures are auto-calculated by numbering the users and services in the database and performing basic arithmetic.

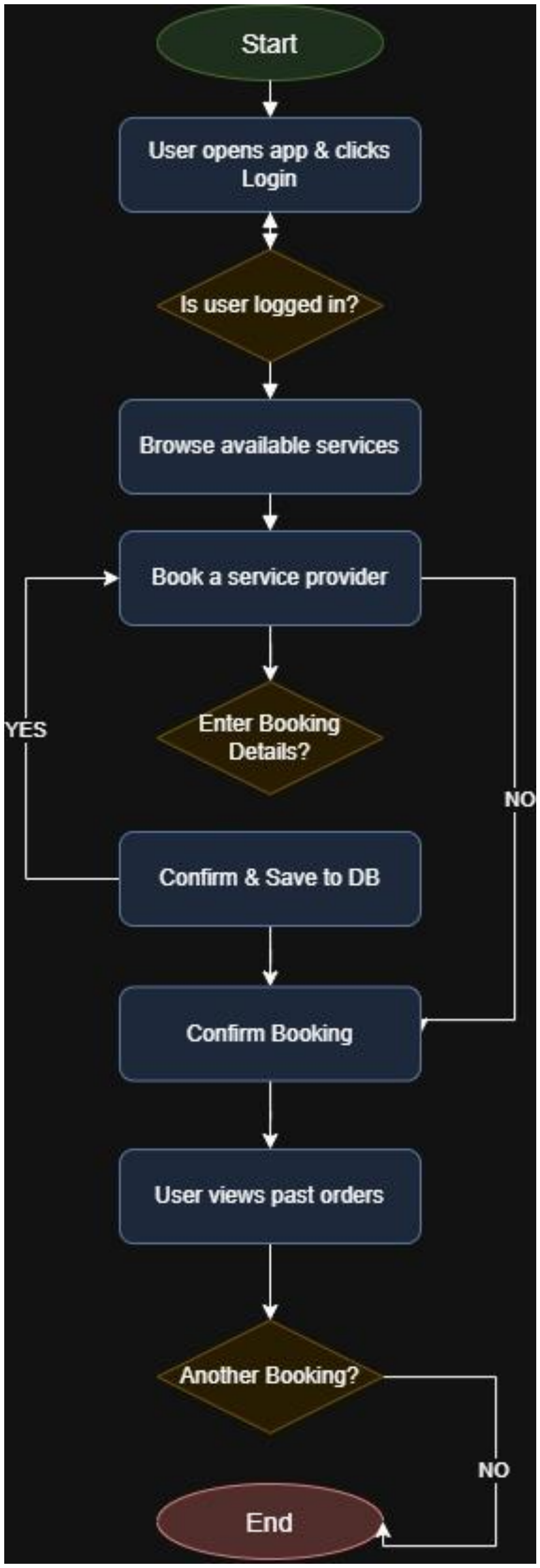
- **CRUD Admin Management Tools**
  It is possible to administer all the data within the system by the admins. They may create, read, update or delete users, service providers or even bookings as it may be required. Those tools are included in the admin panel and allow them to deal with issues, delete accounts that are not used, or change some information without entering the database manually.

4.5. Diagrams

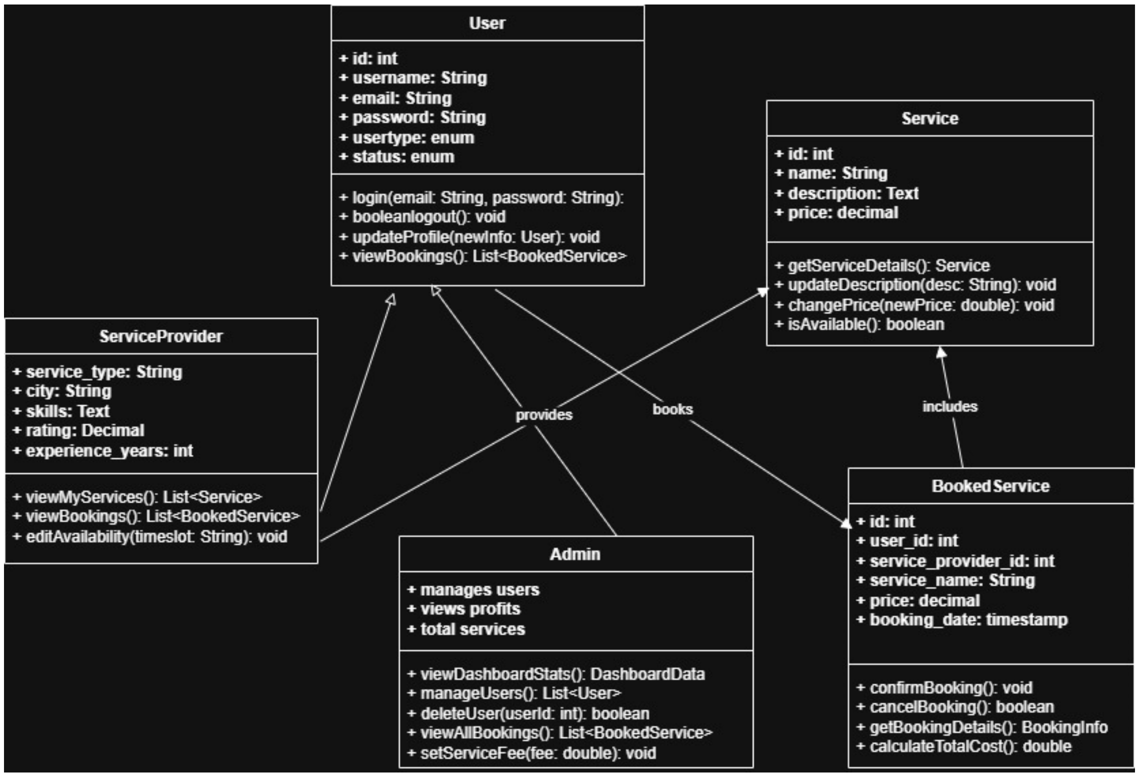
Use Case Diagram



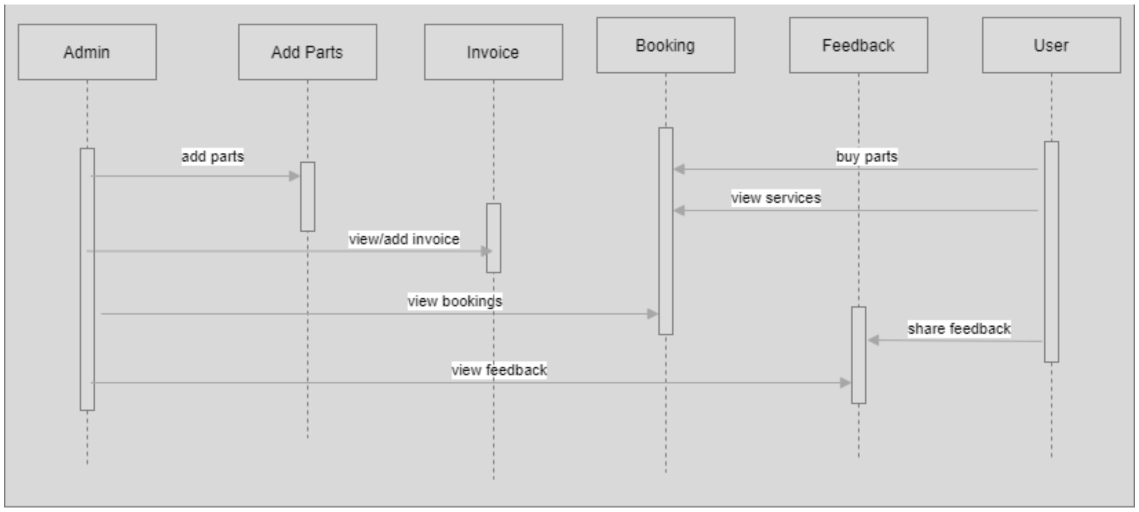
System Flow Diagram (Flow Chart)



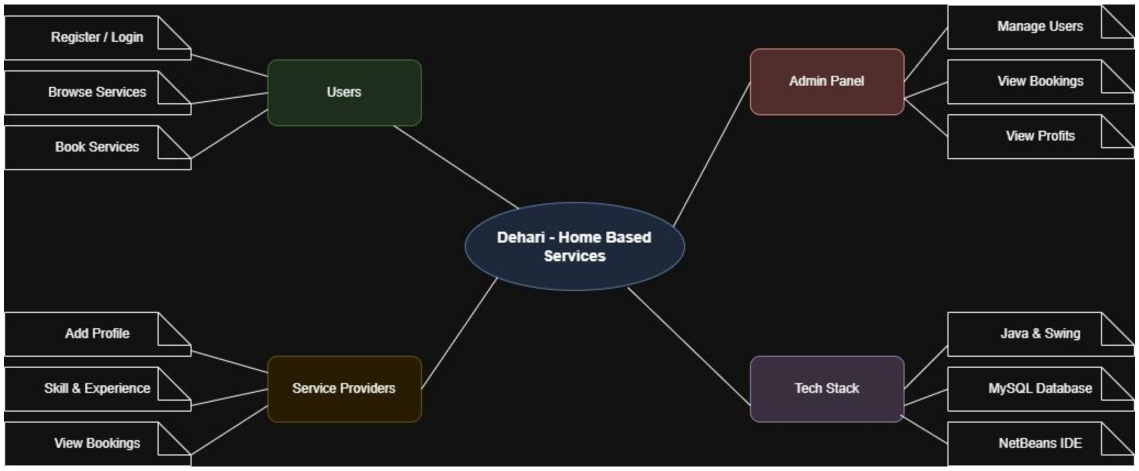
Class Diagram


User
+ id: int
+ username: String
+ email: String
+ password: String
+ usertype: enum
+ status: enum
+ login(email: String, password: String):
+ booleanlogout(): void
+ updateProfile(newInfo: User): void
+ viewBookings(): List<BookedService>
Service
+ id: int
+ name: String
+ description: Text
+ price: decimal
+ getServiceDetails(): Service
+ updateDescription(desc: String): void
+ changePrice(newPrice: double): void
+ isAvailable(): boolean
ServiceProvider
+ service_type: String
+ city: String
+ skills: Text
+ rating: Decimal
+ experience_years: int
+ viewMyServices(): List<Service>
+ viewBookings(): List<BookedService>
+ editAvailability(timeslot: String): void
provides
books
includes
Admin
+ manages users
+ views profits
+ total services
+ viewDashboardStats(): DashboardData
+ manageUsers(): List<User>
+ deleteUser(userId: int): boolean
+ viewAllBookings(): List<BookedService>
+ setServiceFee(fee: double): void
BookedService
+ id: int
+ user_id: int
+ service_provider_id: int
+ service_name: String
+ price: decimal
+ booking_date: timestamp
+ confirmBooking(): void
+ cancelBooking(): boolean
+ getBookingDetails(): BookingInfo
+ calculateTotalCost(): double

Sequence Diagram



Mind Mapping (diagram)



5. External Interface Requirements

5.1. User Interfaces

The Dehari - Home Based Services Booking App is scripted with a simple, easy to understand graphical user interface (GUI) by the application of Java Swing, and FlatLaf has been incorporated to give the app a trendy view. All the types of users (Customer, Service Provider, Admin) are accessing particular screens depending on their roles.

Customer (user) interface

The following are the screens that are located when a user registers, logs in, and books the services:

- **Login/Signup Screen**
    - Single field to fill in email address, password and create account or sign in.
    - invalid: displays red text error.
    - The link to register as provider is there.
- **Home Page / Dashboard**
    - Presents cards of service availability such as Electrician, Plumber, etc.
    - On every card, there is the image, name, and rating.
    - A click brings about a list of service providers.

- **Service Listing Service Page**
  - Displays the list of all the providers of the chosen service.
  - Listed on cards - name, experience, rating, and the button of Booking.
- **Booking Form**
  - AutoPopulated user name and choice provider details.
  - User adds address and confirms the same.
- History of orders page
  - Listing of all previous booking, date, price, provider.

Service Provider Interface

This is what the providers encounter when they log into their account and update profile:

- **Registration Page**
  - Type form to add name, email, password, city, contact, skills and profile picture.
  - dropdown service type (plumber, electrician etc.).
- **Login Page**
  - Just like user log in but verifies provider credentials.
- **Dashboard Page**
  - Displays the profile card, reservations done by the user as well as reviews.
  - Changeable details and picture.

Admin Interface

Admin can manage users, services, and trace financial information.

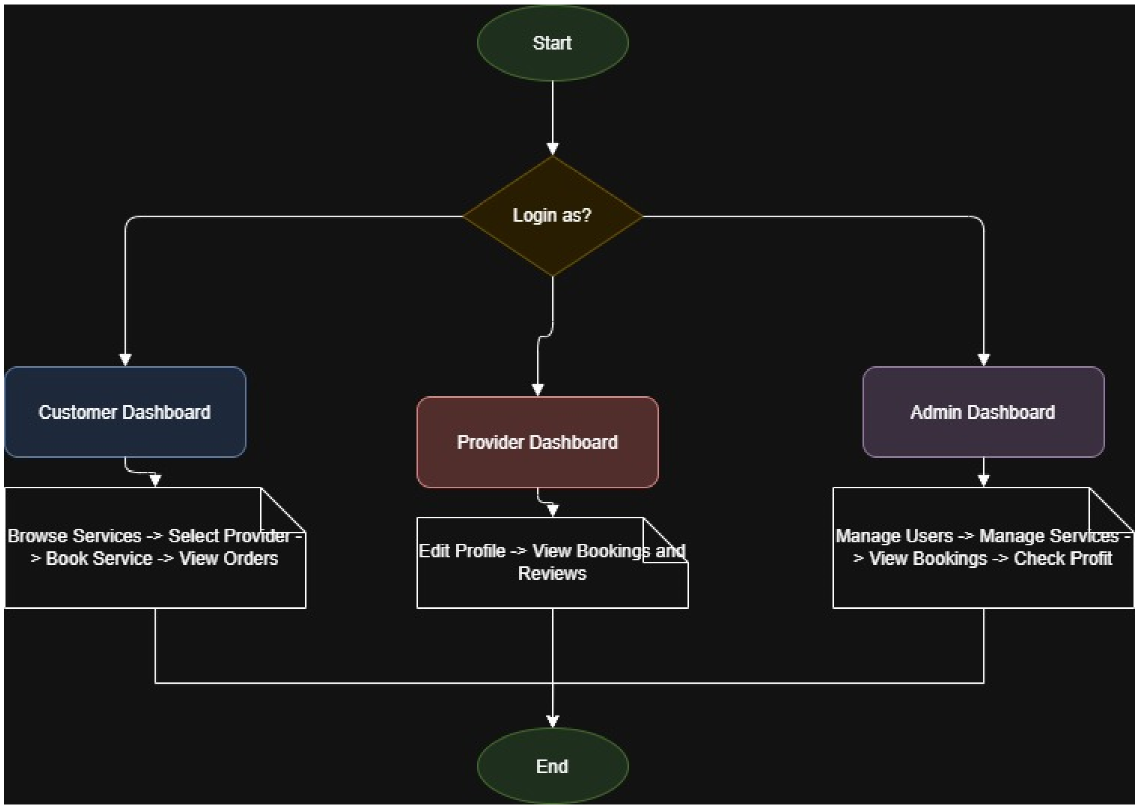
- Admin Home Pages
  - Deny-all; Admin-only access of hardcoded credentials or flag in the DB.
- **Home Admin Dashboard**
  - Pictures of:
    - Total number of Registered Users
    - Cumulative Booked Services
    - Total Profit (caculated through booking **fee)**
- **User Management Page**
  - User table (ID, name, type, status).
  - Admin is able to deactivate, activate or delete users.
- **Service Provider Management**
  - All providers with possibility to edit/delete.
- **Booking records page**
  - See all the bookings per users, name/date accessible.

Summary Table

User Role	**Screen**	**Purpose**
Customer	Login/Signup	Access the system or create account
Customer	Home/Dashboard	View available services
Customer	Service Provider Listing	View and select providers
Customer	Booking Form	Enter info and confirm booking
Customer	Order History	View previous bookings
Service Provider	Registration/Login	Create provider account or login
Service Provider	Dashboard/Profile	Manage personal info and view bookings
Admin	Login	Secure access to admin dashboard

Admin	Dashboard Home	View key metrics (users, bookings, profit)
Admin	User Management	Manage all users
Admin	Service Provider Management	View/edit/delete providers
Admin	Booking Records	Track all booked services

User Interface Flow Diagram


Start
Login as?
Customer Dashboard
Provider Dashboard
Admin Dashboard
Browse Services -> Select Provider -> Book Service -> View Orders
Edit Profile -> View Bookings and Reviews
Manage Users -> Manage Services -> View Bookings -> Check Profit
End

5.2. Hardware Interfaces

The application is not meant to perform on special or complex computers but normal ordinary computers. The chief interface of hardware comes between your local machine hardware (such as a laptop or PC) and the database of the system.

No dedicated hardware appliances such as fingerprint readers, card readers, and sensors are used.

The system employs easy client-server architecture in which:

- The app is used by the users and administrators via a computer (i.e., keyboard, the mouse, and the display).
- MySQL database server exists behind the scenes and communicates with the app.
- The communications occur on common ports and wire or Wi-Fi network connection.

All we need is to ensure that the hardware upon which the system runs fulfilled certain basic requirements.

Hardware Interaction Overview

Interface Type	Description
Input Devices	Keyboard and mouse (used for login, form input, navigation, etc.)
Output Devices	Monitor or laptop screen (to show GUI using Java Swing with FlatLaf)
Storage Device	Local storage used temporarily during runtime (JVM operations)
Database Server	MySQL database must be hosted on a local machine or LAN-accessible server
Network Interface	Wi-Fi or Ethernet connection (needed to connect the app to the database)

Example Use

A client places a booking service by using a laptop. The request is relayed by the app to the MySQL server by using the local network.

With the help of a PC, an admin has access to their dashboard. All the CRUD functions (such as editing user information) are taken care by a single database server.

5.3. Software Interfaces

In this section the intercommunication between the various components of the software is described as well as with outside systems. The software interfaces are rather simple in the case of Dehari. No third party APIs or other tools are used, including only the app, the database, and the libraries involved.

Primary Interfaces of the System

1. **Java Swing + FlatLaf - User Interface**

    - All the UI components such as buttons, forms and menus are programmed using the Java Swing inside the app.
    - To provide the interface with a modern flat style, the FlatLaf library is applied.
    - It is through these interfaces that interactions of users (admin, customer, provider) with the system logic is smooth.

2. **Java Application to MySQL Database**

    - The app has an access to the MySQL database through JDBC (Java Database Connectivity).
    - Queries are performed in order to:
        - Sign up or login.
        - Orders, service provider or fetch.
        - Add, edit, delete user, services, bookings (admin).

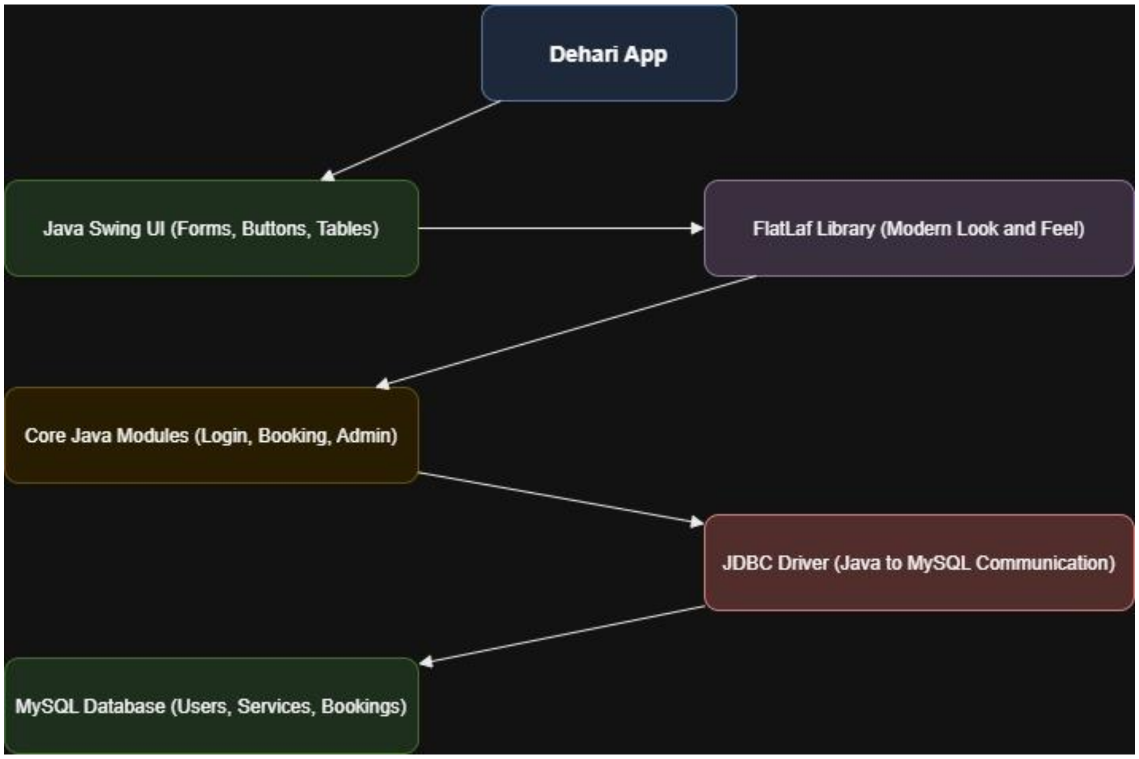
- Data communication with the database is all carried out using SQL statements inside Java.

3. **Internal Module Interfaces**

- Communication between different java classes/modules takes place internally:
    - Credential verification is done by Login module and redirect to dashboards.
    - Booking module processes a booking and makes storage of data.
    - Admin module retrieves reports, displays profits, and authorizes users/services.

Software Stack

Component	Interface Role
Java (JDK 8+)	Core development language
Swing GUI Toolkit	Builds the front-end interface
FlatLaf Library	Enhances GUI design with modern styling
JDBC Driver	Connects Java app to MySQL database
MySQL Server	Stores all persistent data
NetBeans IDE	Used for development and project management


Dehari App
Java Swing UI (Forms, Buttons, Tables)
FlatLaf Library (Modern Look and Feel)
Core Java Modules (Login, Booking, Admin)
JDBC Driver (Java to MySQL Communication)
MySQL Database (Users, Services, Bookings)

5.4. Communication Interfaces

The Dehari application does not depend on complicated or any third-party messaging systems. All the communication between the app and the database is local using normal networking configurations. This version of the system does not have any outward APIs, SMS or email integrations.

The following is the primary redirects of communication within the app:

- **LAN based / Localhost Connection**

  The MySQL database and java application interact either through a localhost or local IP address, in a LAN. It can be used in either classical or lab or offline demos setup.

- **JDBC Protocol**

  MySQL database is communicated with the help of Java Database Connection (JDBC) that is used in the app. JDBC passes SQL queries using TCP/IP and displays SQL results to be further processed.

- **Port Configuration**

  The default port is 3306 in which MySQL is usually working and all the requests are directed to the port by the Java application. In case it is employed on a different network, firewall configurations and port access have to be done.

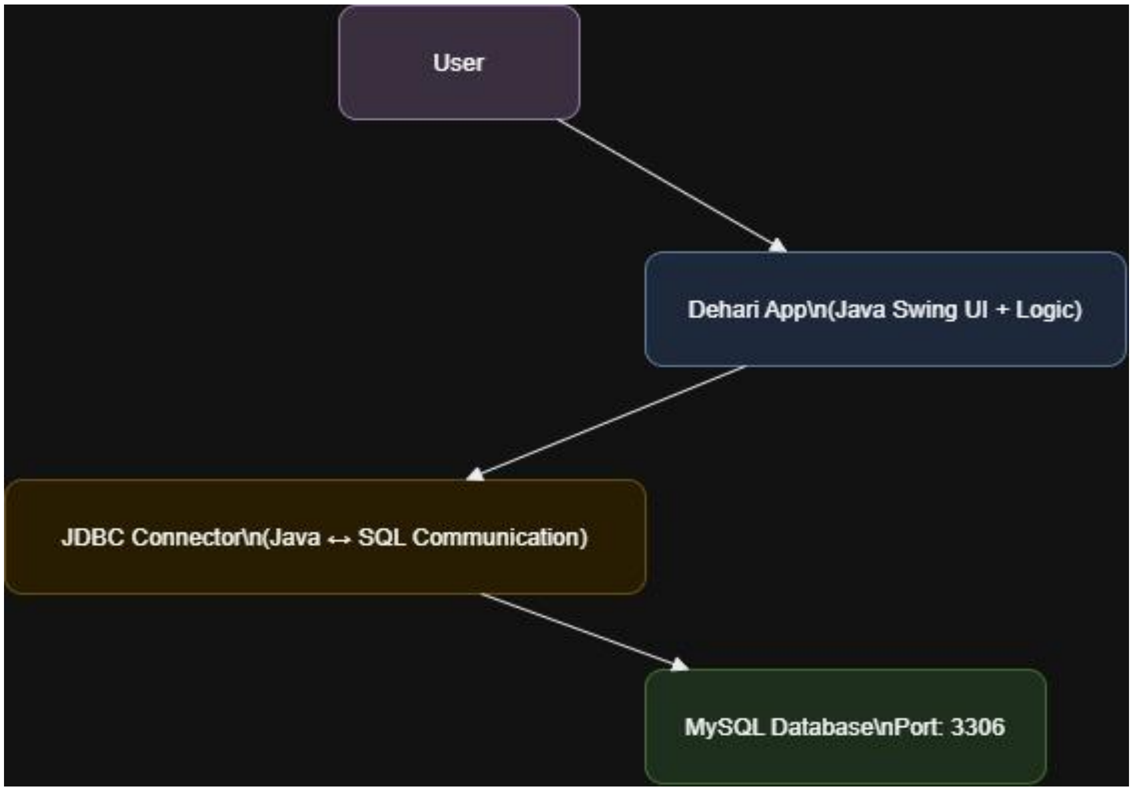
- **There is no Internet Addiction**

  The present configuration has no need of an internet connection. All things are on a local basis. This will make the app sustainable even when there is no cloud or external server.

- **Transmission of Data Format**

  App-to-database communications take place as SQL queries and query results. The current version is free of JSON, XML and REST APIs.

This is a perfect setup in small-scale university projects in which the simplicity, offline accessible and the complete control of the parts, overweigh the external integrations. In the event you think of moving the app to the cloud, you would probably add REST APIs, SSL encryption and use token-based communication.


User
Dehari App\n(Java Swing UI + Logic)
JDBC Connector\n(Java ↔ SQL Communication)
MySQL Database\nPort: 3306

6. Other Non-functional Requirements

6.1. Performance Requirements

Performance is all about the speed and the effectiveness that the system can handle in normal and heavy load. In the case of Dehari - Home Based Services Booking App, performance guarantees effortless browsing, fast bookings, and rapid responds of the maintenances without considering how many users are in the app.

Important Performances Anticipations:

- **Quickest User navigation**

  The pages such as Home, Services and Orders pages should load up immediately. The click on buttons or the transition between sections should not be felt as slow by its users.

- **Fast Search and Filtering**

  After a user has searched through a service provider or put filters in terms of city or category, the results must display within 2 seconds, even in case there are hundreds of entries.

- **Speed of service booking**

  Once submitted, the overall process of choice of a provider, information filling and booking must take no more than 3 sec. to confirm.

- **Administrator dashboard reply**

  Admin functions such as looking all the users, providers or reservation records are supposed to load in 2-3 seconds, even on high volumes of data.

- Concurrent Usage

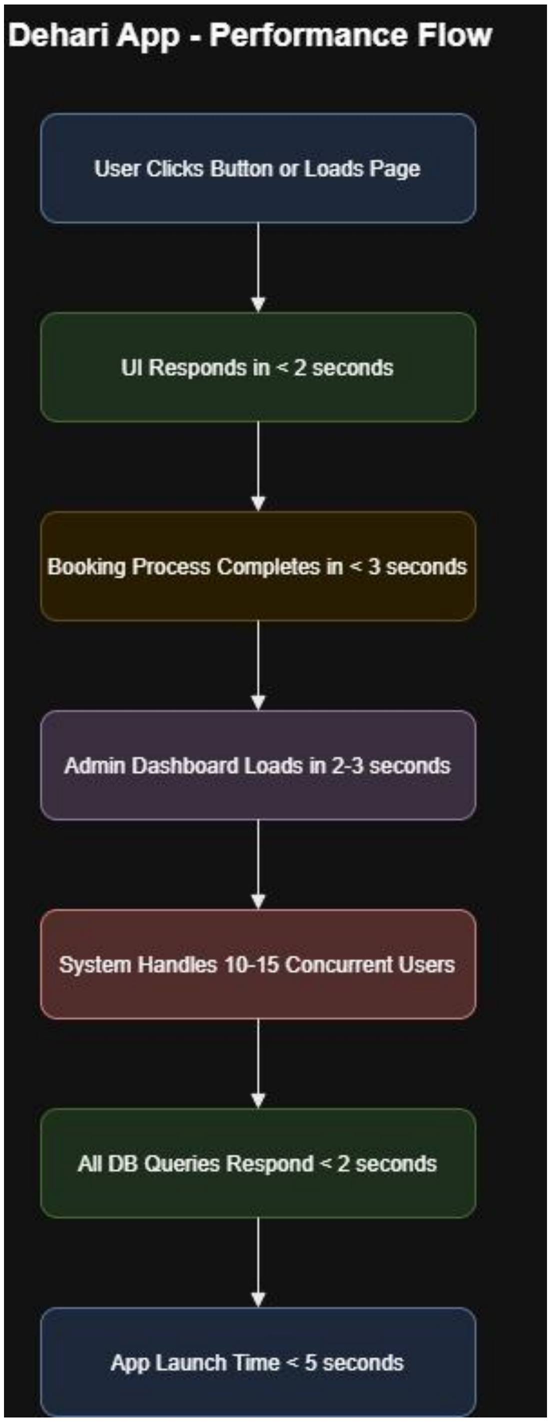
  The system is supposed to serve not less than 10-15 users to make different tasks simultaneously (e.g., booking or logging in or viewing services) without a decrease in its performance or failure.

- **The response of Database Query**

  Any SQL queries using JDBC ought to be answered within 1-2 seconds. They should be indexed properly and optimized SQL.

- **Short App Loading Time**

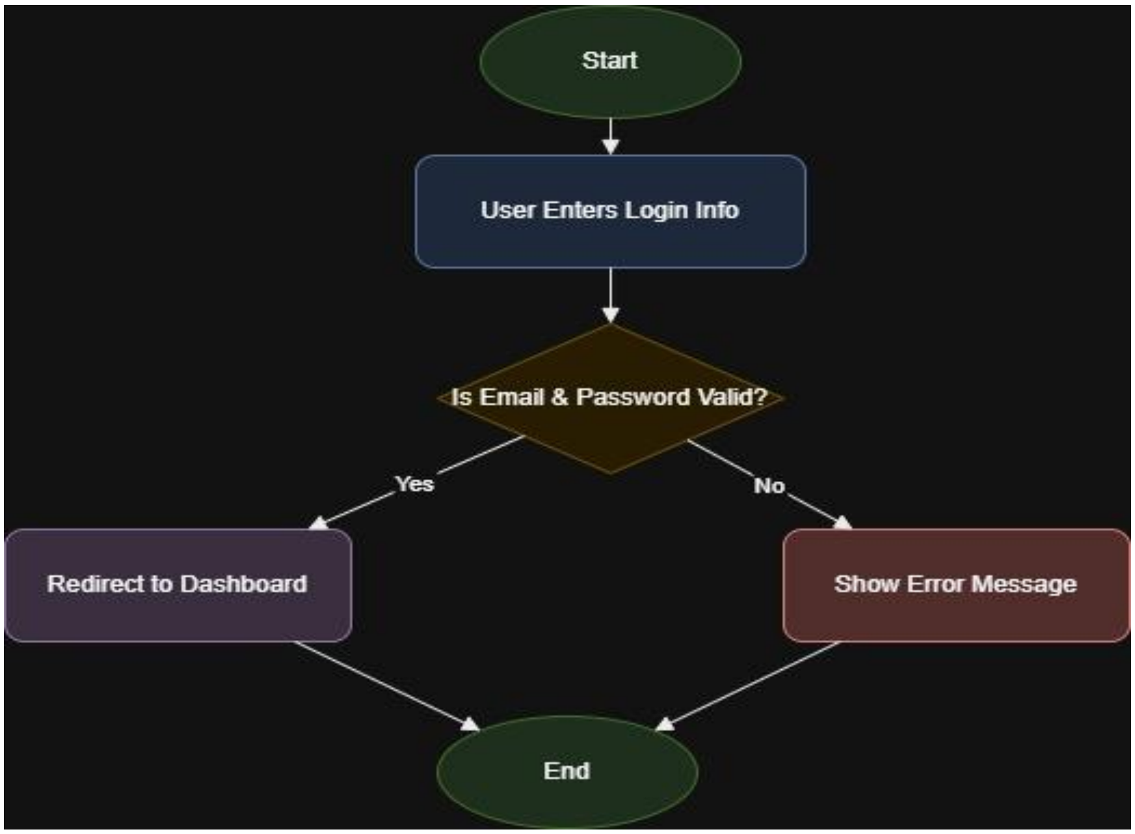
  The main window and services panel must be active within 5 seconds in a typical machine once the app is launched.


Dehari App - Performance Flow
User Clicks Button or Loads Page
UI Responds in < 2 seconds
Booking Process Completes in < 3 seconds
Admin Dashboard Loads in 2-3 seconds
System Handles 10-15 Concurrent Users
All DB Queries Respond < 2 seconds
App Launch Time < 5 seconds

6.2. Security Requirements-

Requirement ID	Requirement Title	Description	Applies To
SR1	User Authentication	Users must log in with valid email and password. No access to protected pages without login.	Users, Service Providers, Admins
SR2	Role-Based Access Control (RBAC)	Only admins can access the admin panel. Normal users and providers are restricted.	Admin Module
SR3	Password Encryption	All passwords are stored in the database using hashing (e.g., SHA-256 or BCrypt) to prevent leakage.	All User Accounts
SR4	Session Management	Users are automatically logged out after inactivity or closing the app window. Prevents session hijacking.	All User Sessions
SR5	Input Validation	All user inputs (e.g., forms, search) must be validated to prevent SQL injection, XSS, or invalid data entry.	Frontend & Backend
SR6	Access Denied Message	Users who try to access unauthorized sections will see a clear "Access Denied" message instead of being redirected silently.	Admin & User Modules
SR7	Limited Login Attempts	After 3–5 failed login attempts, temporarily lock the account or	Login Page

		delay the response to stop brute-force attacks.	
SR8	Secure Database Connection	The app should connect to MySQL using secure credentials. Avoid exposing the root user. Use minimal privilege accounts.	JDBC & DB Configuration
SR9	Audit Logs (optional/advanced)	Admin actions (e.g., deleting users, editing services) can be logged in the system for future auditing and transparency.	Admin Panel
SR10	Account Status Control	Admins can ban/unban users or providers. Banned users should not be able to log in or use any feature.	User & Provider Management

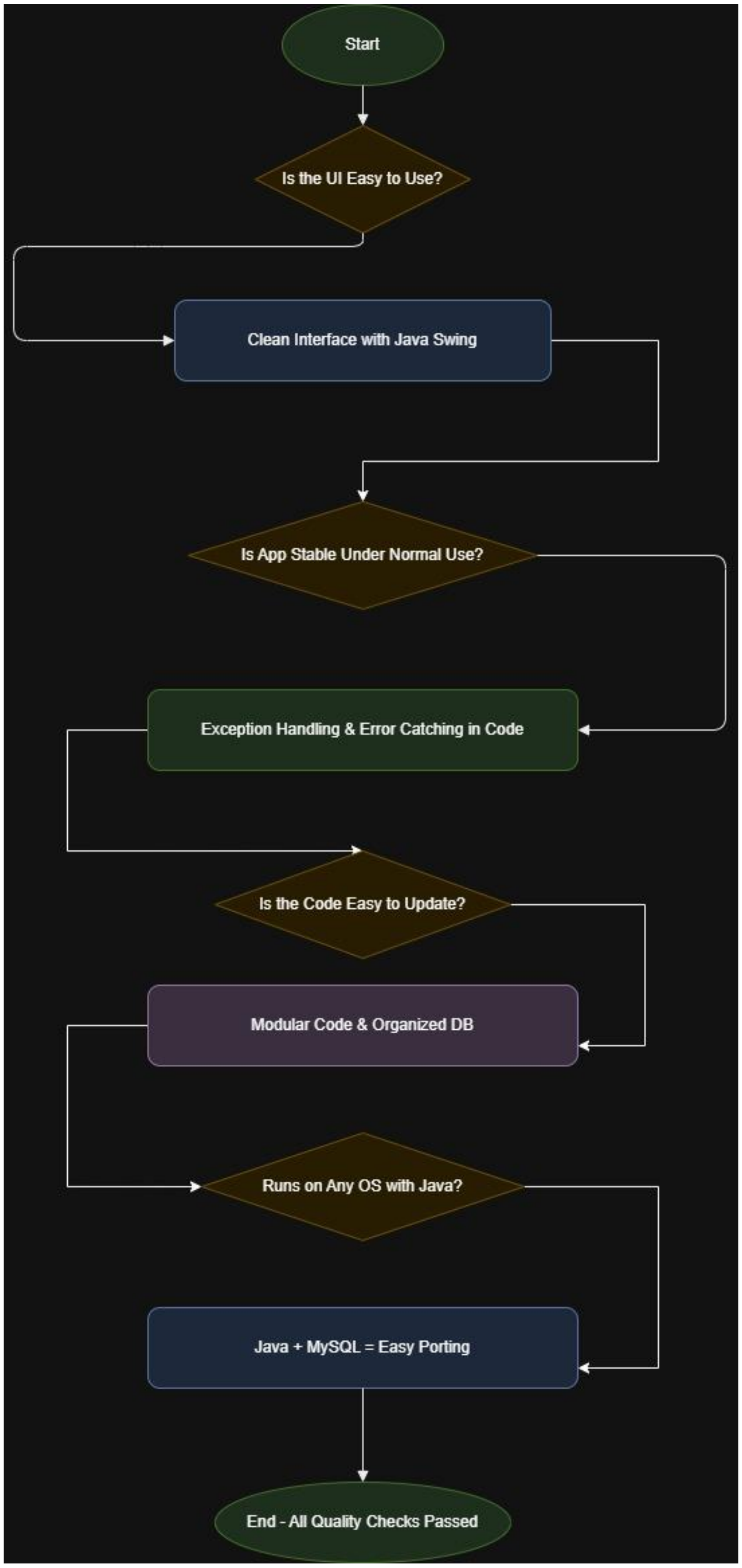


6.3. Quality Attributes

Quality The quality attributes describe the manner in which the Dehari app works efficiently, more than working. This is in term of its usability, reliability, ease of updating and compatibility with other machines or platforms. Such things are important as real-life applications would be especially in scenarios where REAL PEOPLE access it every day to locate home-based services.

Each element of an important quality is divided and presented in the following table, which describes how Dehari treats it:

Attribute	Description	How Dehari Handles It
Usability	The app should be easy to use for both users and service providers.	Simple navigation, clean GUI made with Java Swing and FlatLaf. Clear buttons and flows.
Reliability	The app should work consistently without crashing.	Built-in exception handling in Java; tested flows like login, booking, and data retrieval.
Maintainability	It should be easy to fix bugs or update features.	Organized code structure, clear database schema, and modular design.
Portability	It should run on different machines without needing major changes.	Built with Java — runs on any OS with JDK and MySQL installed.
Security	Protect data from unauthorized access.	Login required for all key areas. Admin/User roles enforced. Passwords are hashed.
Scalability	It should be able to grow as more users or services are added.	Relational DB design and expandable backend logic.
Availability	It should be accessible anytime without downtime.	Local desktop app means users are not dependent on live server uptime.
Responsiveness	It should react quickly to user actions (clicks, form submission, loading).	Fast response times, minimal delay due to optimized database queries.


Start
Is the UI Easy to Use?
Clean Interface with Java Swing
Is App Stable Under Normal Use?
Exception Handling & Error Catching in Code
Is the Code Easy to Update?
Modular Code & Organized DB
Runs on Any OS with Java?
Java + MySQL = Easy Porting
End - All Quality Checks Passed

7. Appendices

This part incorporates additional knowledge that would help in the project understanding. It holds the mock information, trial forms, and UI-specific properties that were not discussed in the major parts yet may be useful in the development, testing process, or further upgrading.

7.1. Mock Data Samples

ID	Username	Email	Role	Status
1	Abdul Rehman	rehmanabdul1445@gmail.com	User	active
2	admin	admin@gmail.com	Admin	active
3	zain_plumb	zain@services.com	provider	active

Service ID	Service Provider	Service Name	City	Price (PKR)
101	Zain Plumbing	Tap Repair	Karachi	500
102	Ali Electric	Wiring Fix	Karachi	500
103	Ali Carpenter	Carpenter	Karachi	500

7.2. Sample Login Form (Mockup Description)

- Single simplistic design that uses two text-fields, Email and Password.
- Login label write as submit buttons
- Forgot Password? Link below.
- Individual button to use "Register" in case the user does not have an account
- Same form, different role access is used in the case of admin login

7.3. Survey/Feedback Form Sample

- Text field full name
- Optional contact text field email
- Aspect of rating (1-5 stars)
- Additional comment textarea field
- Submit button posts the feedback to the administrator