

Software Requirements Specification

for

ShebaBondhu - A One-Stop Home Services System

Prepared by
Nafisa Anzum Kotha (2154901047)
Jawad Anzum Fahim (2254901022)
Ruponti Muin Nova (2254901048)
Sadia Alam (2254901048)
Muttakin Mahmud (2254901120)

*Bangladesh University of Professionals
Information and Communication Engineering
Software Testing & Maintenance (STM)*

September 2025

Contents

Revision History	1
1 Introduction	2
1.1 Purpose	2
1.2 Intended Audience	2
1.3 Intended Use	3
1.3.1 Business Analysts (BAs)	3
1.3.2 Project Managers (PMs)	3
1.3.3 Developers	3
1.3.4 QA/QC Engineers	3
1.3.5 End Users (Customers and Service Providers)	3
1.3.6 Marketing Staff	3
1.3.7 Testers	4
1.3.8 Investors	4
1.4 Product Scope	4
1.4.1 Purpose	4
1.4.2 Benefits and Objectives	4
1.4.3 Alignment with Goals	4
1.4.4 Business Strategy Connection	4
1.5 Risk Definition	4
2 Overall Description	6
2.1 User Classes and Characteristics	6
2.2 User Needs	7
2.3 Operating Environment	7
2.4 Constraints	7
2.5 Assumptions	7
3 Requirements	9
3.1 Functional Requirements	9

3.1.1	Authentication – Role-based User Stories & Acceptance Criteria	9
3.1.2	Live Tracking	11
3.1.3	SOS / Emergency Request	11
3.1.4	Searching	11
3.1.5	Rating & Review	12
3.1.6	Notifications	12
3.1.7	Real-time Chat	13
3.1.8	Personal Profile (Homeowner & Service Provider)	14
3.1.9	Package Booking	15
3.1.10	Service History	15
3.2	Non Functional Requirements	16
3.2.1	Performance	16
3.2.2	Security	17
3.2.3	Safety	17
3.2.4	Quality	17
Appendix		18
A Documentation Tools		18
A.1	API Documentation: JSDoc	18
A.1.1	Installation & Usage	18
A.2	Version Control	18
A.3	Diagram Tools	18
A.4	Document Preparation	18
B Coding Standards		19
B.1	Recommended Style Guide: Airbnb	19
B.2	Key Conventions	19
B.2.1	Imports	19
B.2.2	Naming	19
B.2.3	Variables	19
B.2.4	Functions	20
B.2.5	Formatting	20
B.3	Next.js Structure	20
B.3.1	Pages (page.tsx)	20
B.3.2	API Routes (route.ts)	20
B.3.3	Database (lib/dbConnect.ts)	20
B.3.4	Models (models/*.ts)	20
B.3.5	Components (components/*.tsx)	20

C Use Case Diagrams	21
D Extended Use Case Descriptions	22
E Database Schema	23
E.1 Entity-Relationship Overview	24
F Testing Strategy	25
F.1 Recommended Tool: Jest	25
F.2 Implementation	25
G Architectural Pattern	26
G.1 Selected Pattern: MVVM	26
G.2 Why MVVM Fits ShebaBondhu	26
G.3 MVVM Layer Responsibilities	26
G.4 Folder Structure (MVVM)	26
G.5 Benefits of MVVM for ShebaBondhu	27

Revision History

Revision	Date	Author(s)	Description
1.0	05.10.2025	Nova	Initial SRS document created
1.1	07.10.2025	Nova, Kotha, Sinthia, Muttaki, Jawad	Functional requirements added
2.0	04.11.2025	Nova, Sinthia, Kotha, Muttaki	Testing strategy finalized
2.1	05.11.2025	Sinthia, Kotha, Muttaki	Documentation tools finalized
2.2	05.11.2025	Nova, Jawad	Coding standards finalized
3.0	09.11.2025	Nova	Database schema finalized
3.1	09.11.2025	Sinthia	Use case diagrams finalized
3.2	15.11.2025	Nova, Sinthia, Muttaki, Kotha	Architectural pattern finalized
4.0	17.11.2025	Sinthia	Extended use case descriptions added

Chapter 1

Introduction

1.1 Purpose

The purpose of *ShebaBondhu* is to create a reliable digital platform that makes everyday household services easier to access for people in Bangladesh. At present, most people rely on word of mouth, local contacts, or unverified advertisements to find workers such as electricians, plumbers, cleaners, and mechanics. This process often takes time, creates uncertainty about the quality of work, and sometimes puts users at risk because there is no proper verification of the service providers. *ShebaBondhu* addresses these problems by offering a one-stop solution where users can quickly search, book, and communicate with trusted professionals from their mobile phones. The system is designed not only for convenience but also for safety and trust. Every service provider will have a profile that can be verified and reviewed by users, which helps people feel confident about who they are hiring. Real-time tracking further adds to this trust by allowing users to see the provider's location and status. In addition, the platform includes an SOS feature that allows users to request immediate help in case of urgent needs such as sudden electrical failures, water leaks, or household emergencies.

Another important purpose of *ShebaBondhu* is to create opportunities for local service providers. Many skilled workers in Bangladesh work informally and struggle to reach new customers. By using this platform, they can present their services to a wider audience, build a reputation through ratings and reviews, and increase their income in a fair and transparent way. The package booking option also benefits both sides by supporting long-term arrangements, such as monthly cleaning or regular maintenance, instead of just one-time services.

1.2 Intended Audience

The Software Requirements Specification for *ShebaBondhu* is designed for a wide range of stakeholders who will take part in the development, operation, and use of the system. The main audiences are:

- **Business Analysts (BAs):** They will analyze business needs and make sure the requirements of the system align with the project's goals.
- **Project Managers (PMs):** They will use this document to track progress, allocate resources, and ensure that development stays on schedule and within scope.
- **Developers:** They will depend on the requirements to design, implement, and integrate features correctly.
- **QA/QC Engineers:** They will use the document to create test cases and confirm that the platform works as expected.

- **End Users (Customers and Service Providers):** Customers will rely on the app to book and track services, while providers will use it to offer their services and manage bookings.
- **Marketing Staff:** They will use this document to understand the target users and design effective promotion strategies.
- **Testers:** They will verify that the system behaves according to the requirements written in this document.
- **Investors:** They will review the SRS to understand the project's scope, risks, and potential for growth.

1.3 Intended Use

The intended use of this SRS is to guide all groups involved in the development and use of *ShebaBondhu*. Each group will use the document differently based on their responsibilities.

1.3.1 Business Analysts (BAs)

Business analysts will use the document to understand business needs and ensure they are translated into system requirements. It will help them analyze user interactions, workflows, and required features so they can create clear documentation for the development team.

1.3.2 Project Managers (PMs)

Project managers will use the SRS to understand the scope of the project and to plan resources, deadlines, and deliverables. They will rely on it to make sure the development process aligns with the defined requirements and that the final product meets user expectations.

1.3.3 Developers

Developers will depend on this document to design and implement the features of the app. The SRS provides them with detailed requirements about the system's functionality, constraints, and expected behavior, which guides them during coding and integration.

1.3.4 QA/QC Engineers

Quality assurance and quality control engineers will use the document to prepare test cases and validate the platform. They will make sure that *ShebaBondhu* meets performance, usability, and security standards described in the requirements.

1.3.5 End Users (Customers and Service Providers)

Customers and service providers will not directly use the SRS for development, but it describes the features they can expect when using the app. Customers will be able to search for services, make bookings, and track providers, while service providers will use it to manage their work and interact with clients.

1.3.6 Marketing Staff

Marketing teams will use the SRS to understand the target users, the scope of services offered, and the unique features of the system. This will help them design campaigns and communication strategies that highlight the strengths of *ShebaBondhu*.

1.3.7 Testers

Testers will rely on the document to design test scenarios and verify that the app behaves according to the defined requirements. It will guide them in identifying bugs, checking workflows, and ensuring the system delivers the expected results.

1.3.8 Investors

Investors will use this SRS to get a clear picture of the system's goals, features, and risks. It will help them make informed decisions about supporting the project by understanding its scope and future potential.

1.4 Product Scope

1.4.1 Purpose

ShebaBondhu is a mobile and web platform that helps people find and book trusted household service providers like plumbers, electricians, cleaners, and appliance repair experts. Instead of searching around or relying on random contacts, users can quickly search, book, and track skilled workers in one place.

1.4.2 Benefits and Objectives

- **Makes life easier:** Users can search and book services anytime without hassle.
- **Emergency help:** SOS option lets people call for urgent help fast.
- **Builds trust:** Ratings and reviews show which providers are reliable.
- **Keeps things organized:** Service history, package bookings, and notifications help users track their services.
- **Supports workers:** Skilled providers can reach more customers and grow their income.
- **Safe and reliable:** Live tracking and secure login keep both users and providers confident

1.4.3 Alignment with Goals

This project matches our goal to make daily household services simple, safe, and dependable in Bangladesh. It encourages people to use verified professionals and creates a fair system for both users and service providers.

1.4.4 Business Strategy Connection

By offering all services in one trusted app, ShebaBondhu can attract more users, improve customer satisfaction, and grow steadily in the household service market.

1.5 Risk Definition

- **User Inactivity:** Risk that users or providers may not actively engage, reducing platform effectiveness.
- **Administrator Workload:** Handling complaints, verification, and SOS requests may overwhelm admins.

- **Communication Breakdown:** Misunderstanding between users, providers, or team members could affect service quality.
- **Service Quality Variability:** Inconsistent performance by providers may harm platform reputation.
- **Changes in Scope or Features:** Late feature changes may delay development.
- **Security Vulnerabilities:** Threats to user data, payment systems, or location tracking features.
- **Connectivity Issues:** Poor internet connectivity could disrupt live tracking, chat, or SOS response.
- **Fraudulent Activities:** Fake accounts or providers could misuse the platform if verification fails.

Chapter 2

Overall Description

The *ShebaBondhu* system is a home services web application designed to connect house owners with skilled service providers such as electricians, plumbers, carpenters, and cleaners. The platform simplifies the process of finding and booking reliable services by allowing users to search based on category, location, and ratings. It ensures that house owners can quickly find trusted professionals while enabling service providers to gain more visibility and job opportunities. The system is intended to provide a smooth, secure, and user-friendly experience for both groups.

2.1 User Classes and Characteristics

The system has two primary user classes: house owners and service providers.

House Owners (Customers)

These are individuals who seek household services such as plumbing, electrical work, or cleaning. They generally possess basic to moderate digital literacy and are familiar with browsing and using mobile or web applications. Their main expectation is to find trustworthy service providers quickly, compare them based on ratings, availability, and cost, and easily book the required service. They value convenience, affordability, and security in the overall process.

Service Providers (Professionals)

These are skilled workers such as electricians, plumbers, and carpenters who wish to offer their services through the platform. Many of them may have limited technical expertise, so the system must be simple and easy to navigate. Their key requirements include registering an account, creating or updating their profile, setting availability, and responding to incoming service requests. They are mainly focused on increasing their visibility and gaining more work opportunities through positive ratings and reviews.

2.2 User Needs

The needs of the two user groups are as follows:

House Owners

House owners need an efficient way to search for services by category and location, compare providers based on reputation, and book them directly through the application. They require secure login and profile management, the ability to view service history, and features like real-time tracking, SOS for emergencies, and direct chat for coordination. They also expect a transparent feedback system through ratings and reviews.

Service Providers

Service providers need a straightforward registration process and profile management tools to list their services, set pricing, and manage availability. They require notifications for new service requests, an option to communicate with customers, and visibility in search results based on ratings and proximity. They also depend on an honest rating and review system to build trust and attract more customers.

2.3 Operating Environment

The *ShebaBondhu* system will operate as a web application and will be tested on multiple operating systems including Windows, macOS, and Linux. It will also be developed with future mobile app conversion in mind, supporting Android and iOS devices. On the client side, users will access the application through modern browsers such as Google Chrome, Firefox, and Edge. The system will feature a mobile-responsive design to ensure usability across devices.

On the server side, the application will use the Next.js framework for frontend and backend integration, along with necessary technologies such as HTML, CSS, and JavaScript. Data will be stored in a database-MongoDB, maintaining user accounts, service provider details, and booking records. The hardware environment includes client devices like PCs, laptops, and smartphones, while the server will be cloud-hosted on platforms such as AWS, Azure, or optionally on local servers. A stable internet connection is required for smooth operation, especially for features like live tracking, chat, and notifications.

2.4 Constraints

The system is subject to several constraints. Technically, it requires continuous internet connectivity as no offline mode is available. The platform must efficiently handle multiple users simultaneously without performance degradation. Security is a priority, with measures such as password hashing, encrypted data storage, and secure communication protocols.

From a user perspective, service providers may have limited digital skills, so the user interface must be intuitive and simple. House owners rely heavily on ratings and reviews, so safeguards must prevent fake or misleading feedback. Additionally, the system must comply with local regulations for online transactions, data protection, and user privacy, especially if payment gateways are integrated.

2.5 Assumptions

The development and use of *ShebaBondhu* assumes that all users, including house owners and service providers, will have access to internet-enabled devices. Service providers are assumed to be skilled and reliable, while ratings and reviews from house owners are assumed to be honest. Payment integration, if implemented, will

rely on popular local gateways such as bKash, Nagad, or Rocket. Initially, the system will support only one region or city, with plans for expansion to additional regions in the future.

Chapter 3

Requirements

3.1 Functional Requirements

3.1.1 Authentication – Role-based User Stories & Acceptance Criteria

User Story 1: User Registration

- **Homeowner:** As a homeowner, I want to register using my email, phone number, or Google account, so that I can quickly find and book trusted service providers.
- **Service Provider:** As a service provider, I want to register using my name, phone number, email address, service type (multi-select: plumbing, cleaning, etc.), address, password, Years of Experience, Certification/License (optional), and Upload ID Proof, so that I can showcase my services and gain customer trust.

Acceptance Criteria:

- **Given** I am a new user on the registration page,
- **When** I provide valid details (email/phone/social login),
- **Then** my account should be created and I should be redirected to my role-specific login page.
- **And if** I provide invalid or duplicate details,
- **Then** I should remain on the registration page until I provide valid information.

User Story 2: Secure Login

- **Homeowner:** As a homeowner, I want to log in securely with my credentials, so that I can access my profile and book services.
- **Service Provider:** As a service provider, I want to log in securely, so that I can manage my bookings and communicate with homeowners.

Acceptance Criteria:

- **Given** I am a registered user on the login page,
- **When** I enter valid credentials like my email and password,
- **Then** I should receive an OTP through email and be redirected to my role-specific homepage.
- **When** I am using a Google account,
- **Then** I will be redirected to my user-specific homepage without 2-step verification.
- **When** I enter invalid credentials,
- **Then** I should see an error message (“Invalid username or password”).

User Story 3: Password Reset

- **Homeowner:** As a homeowner who forgot my password, I want to reset it via OTP or email verification, so that I can safely regain access to my homepage and profile.
- **Service Provider:** As a service provider who forgot my password, I want to reset it via OTP or email verification, so that I can safely regain access to my service management homepage.

Acceptance Criteria:

- **Given** I am on the login page,
- **When** I click “Forgot Password” and provide my email/phone,
- **Then** I should receive a verification link/OTP.
- **When** I verify successfully,
- **Then** I should be able to create a new password and log in again.
- **When** I input the wrong OTP,
- **Then** I will not be able to reset my password and will be shown an error.

User Story 4: Stay Logged In

- **Homeowner:** As a busy homeowner, I want the app to keep me logged in, so that I can book services quickly without entering my credentials every time.
- **Service Provider:** As a busy service provider, I want the app to keep me logged in, so that I can respond to homeowner requests faster.

Acceptance Criteria:

- **Given** I have selected the “Remember Me” option at login,
- **When** I return to the website later,
- **Then** I should stay logged in without re-entering credentials (until I log out manually).

3.1.2 Live Tracking

User Story: Real-time Location Tracking

- **Homeowner:** As a homeowner, I want to track the location of the service provider in real time, so that I know when they will arrive and can plan accordingly.
- **Service Provider:** As a service provider, I want my location to be shared with homeowners during a service visit, so that they can monitor my arrival and reduce missed appointments.

Acceptance Criteria:

- **Given** a service is booked and accepted,
- **When** the service provider starts the journey to the homeowner,
- **Then** the homeowner should see the provider's live location on the map in real time.
- **If** the provider's location cannot be accessed,
- **Then** the homeowner should see a message indicating the location is unavailable.

3.1.3 SOS / Emergency Request

User Story: Emergency Alert

- **Homeowner:** As a homeowner, I want to send an emergency SOS request during a service, so that I can receive immediate assistance in urgent situations.
- **Service Provider:** As a service provider, I want to receive real-time SOS notifications from homeowners, highlighted prominently on my dashboard, so that I can respond quickly and give urgent attention when needed.

Acceptance Criteria:

- **Given** a homeowner is on the active service page,
- **When** the “SOS” button is pressed,
- **Then** an immediate alert is sent to nearby service providers and to family or friends.
- **And** the request is visually prioritized on the provider dashboard (e.g., red highlight or pop-up).
- **And** the homeowner receives confirmation that the SOS has been sent, along with an estimated response time.

3.1.4 Searching

User Story: Search for Services

- **Homeowner:** As a homeowner, I want to search for service providers by category (e.g., plumber, electrician) and location, so that I can quickly find the right professional available near me.
- **Service Provider:** As a service provider, I want my services to appear in relevant searches based on my skills, ratings, and location, so that homeowners can easily discover and book me.

Acceptance Criteria:

- **Given** a homeowner is on the search page,
- **When** they enter a keyword (e.g., "plumber") and location,
- **Then** they should see a list of matching service providers with name, rating, and availability.
- **If** no providers are found,
- **Then** the homeowner should see a message: "No providers available. Please try another search."
- Providers should only appear in search results if they are active and available for bookings.

3.1.5 Rating & Review

User Story: Provide and Receive Feedback

- **Homeowner:** As a homeowner, I want to rate and review the service provider after the service is completed, so that I can share my experience and help others make informed choices.
- **Service Provider:** As a service provider, I want to receive ratings and reviews from homeowners, so that I can improve my service quality and build trust with future customers.

Acceptance Criteria:

- **Given** a service is marked as completed,
- **When** the homeowner opens the booking history,
- **Then** they should see an option to rate the provider (1–5 stars) and leave a written review.
- **If** the homeowner submits a rating and review, it should be stored and reflected in the provider's overall rating.
- A provider cannot rate themselves; only homeowners who booked and completed a service can leave ratings.
- **If** the homeowner skips rating, they should still be able to submit it later from booking history.

3.1.6 Notifications

User Story: Receive Timely Updates

- **Homeowner:** As a homeowner, I want to receive notifications about booking confirmations, cancellations, or service updates, so that I always know the current status of my requests.
- **Service Provider:** As a service provider, I want to receive notifications about new bookings, cancellations, or changes from homeowners, so that I can respond quickly and manage my schedule.

Acceptance Criteria:

- **Given** I am logged in as a homeowner or service provider,
- **When** a booking is confirmed, canceled, updated, or my profile information changes,
- **Then** I should see a notification popup or message in my notification panel.
- **Given** I have unread notifications,
- **When** I open the notifications panel,
- **Then** I should see all recent notifications marked as read once I view them.
- **Given** I have disabled notifications in my settings,
- **When** a booking status changes,
- **Then** I should not receive popup notifications but should still see updates in the history tab.

3.1.7 Real-time Chat

User Story: Direct Communication

- **Homeowner:** As a homeowner, I want to chat with my booked service provider in real-time, so that I can share details or clarify instructions quickly.
- **Service Provider:** As a service provider, I want to chat directly with the homeowner, so that I can confirm details, ask questions, and provide updates about the service.

Acceptance Criteria:

- **Given** I am logged in as a homeowner or service provider,
- **When** I open a booking and tap “Chat,”
- **Then** I should see a chat window showing previous messages (if any) and be able to type and send a new message.
- **Given** I send a message,
- **When** the recipient is online,
- **Then** they should receive the message instantly with a notification sound or icon badge.
- **Given** I am offline,
- **When** someone sends me a message,
- **Then** I should get a notification about the unread message the next time I log in.
- **Given** the conversation is completed,
- **When** I close the chat window,
- **Then** the chat history should remain available for later reference under that booking.

3.1.8 Personal Profile (Homeowner & Service Provider)

User Story 1: Complete Profile after Registration

- **Homeowner:** As a homeowner, I want to complete my personal profile with details like full name, phone number, and address (area-wise) if these are incomplete during registration, so that I can book services in my location.
- **Service Provider:** As a service provider, I want to complete my service profile with details like service type, service area, years of experience, certifications, and ID proof if these are incomplete during registration, so that homeowners can trust me and book my services.

Acceptance Criteria:

- **Given** I log in for the first time,
- **When** I go to my homepage without filling my profile,
- **Then** I should be prompted to complete my profile before accessing booking/management features.
- **When** I submit all mandatory fields (name, address, phone, service type for providers),
- **Then** I should be able to proceed to booking (homeowner) or service management (provider).

User Story 2: Update Profile

- **Homeowner:** As a homeowner, I want to update my profile details (address, phone number, etc.), so that my service requests always reach the right location if I move to a different place.
- **Service Provider:** As a service provider, I want to update my service details (new certifications, service types, updated service area), so that I can attract more homeowners with accurate information.

Acceptance Criteria:

- **Given** I am logged in,
- **When** I go to the profile page and edit my information,
- **Then** my changes should be saved and reflected immediately.
- **When** I leave mandatory fields blank or invalid,
- **Then** I should see an error message and changes should not be saved.

User Story 3: Delete Profile / Account

- **Homeowner:** As a homeowner, I want to delete my account, so that I can stop using the platform if I don't need services anymore.
- **Service Provider:** As a service provider, I want to delete my account, so that my profile and services are no longer visible to homeowners.

Acceptance Criteria:

- **Given** I am on my profile page,
- **When** I click “Delete Account” and confirm,
- **Then** my account should be deactivated/removed from the platform.
- **When** I cancel the delete action,
- **Then** my account should remain active.

User Story 4: Profile Picture Upload

- **User:** As a user, I want to upload a profile picture, so that my account looks authentic and trustworthy.

3.1.9 Package Booking

User Story 1: Create and Offer Service Packages

- **Service Provider:** As a service provider, I want to create and offer service packages (e.g., monthly cleaning, quarterly pest control), so that I can attract long-term customers and ensure a stable, predictable income.

User Story 2: Discover and Book Service Packages

- **Homeowner:** As a homeowner, I want to discover and book service packages, so that I can schedule recurring services conveniently without manual re-booking and potentially receive a better price.

Acceptance Criteria:

- **Given** I am a service provider on my dashboard,
- **When** I navigate to the "Manage Services" section and choose to create a "Package,"
- **Then** I should be able to define the package details: name, description, list of services included, frequency (e.g., weekly, monthly), duration (e.g., 3 months), and total price.
- **And** the created package should be visible on my public profile for homeowners to book.
- **Given** I am a homeowner viewing a service provider's profile,
- **When** I select a service package,
- **Then** I should see all the details of the package, including the schedule of services and the total cost.
- **When** I proceed to book the package and select a start date,
- **Then** my booking for the entire duration of the package should be confirmed, and the recurring appointments should appear in my "Upcoming Services" list.
- **Given** a package is booked,
- **When** it's time for a scheduled service within the package,
- **Then** both the homeowner and service provider should receive a notification reminder, just like a one-time service.

3.1.10 Service History

User Story: View Past Services

- **Homeowner:** As a homeowner, I want to view a simple list of all my previously completed services, so that I can easily track my expenses, remember providers I liked, and re-book a service quickly.
- **Service Provider:** As a service provider, I want to view a history of all the jobs I have completed, so that I can track my earnings, review my work volume, and see my customer history.

Acceptance Criteria:

- **Given** I am a logged-in user (either homeowner or service provider),
- **When** I navigate to the "Service History" or "Past Bookings" section in my profile,
- **Then** I should see a list of all my completed services, sorted by the most recent date first.
- **Given** I am viewing the service history list,
- **When** I look at an entry in the list,
- **Then** it should display key information at a glance:
 - For Homeowner: Service Type, Provider's Name, Completion Date, and Final Cost.
 - For Service Provider: Service Type, Homeowner's Name, Completion Date, and My Earnings.
- **Given** I want more details about a past service,
- **When** I click on a specific service from the history list,
- **Then** I should be taken to a detailed summary page showing the original booking details, payment receipt, and the rating/review that was given.
- **And** as a homeowner, I should see a "Book Again" button to easily initiate a new booking with the same provider for the same service.
- **Given** I am a new user with no completed bookings,
- **When** I visit the Service History page,
- **Then** I should see a message like "You have no completed services yet."

3.2 Non Functional Requirements

3.2.1 Performance

The system must be responsive and efficient under load to ensure a positive user experience.

- **Response Time:** All user interactions, such as button clicks, form submissions, and page transitions, must be acknowledged by the system in under 500 milliseconds.
- **Page Load Time:** Critical pages (including Homepage, Search Results, Service Provider Profile, and Booking page) must load completely within 3 seconds on a standard 10 Mbps internet connection.
- **Concurrency:** The system must be able to handle at least 500 concurrent users performing typical actions (searching, booking, chatting) without any noticeable degradation in performance.
- **Live Tracking Latency:** The real-time location tracking feature must update the provider's location on the homeowner's map at least every 10 seconds with a data refresh latency of less than 2 seconds.

3.2.2 Security

The system must protect user data and prevent unauthorized access.

- **Data Encryption:** All communication between the client and server must be encrypted using current TLS (Transport Layer Security) protocols. Sensitive user data (e.g., passwords, ID proofs, personal information) must be encrypted at rest in the database.
- **Authentication and Authorization:** Passwords must be stored using a strong, salted hashing algorithm (e.g., Argon2, bcrypt). The system must enforce strict Role-Based Access Control (RBAC) to ensure that users can only access data and functionalities relevant to their role (Homeowner vs. Service Provider).
- **Vulnerability Protection:** The application must be protected against common web vulnerabilities as listed in the OWASP Top 10, including SQL Injection, Cross-Site Scripting (XSS), and Cross-Site Request Forgery (CSRF).
- **Secure Payments:** All payment processing must be handled through a PCI-DSS compliant third-party payment gateway to ensure the security of financial information.

3.2.3 Safety

The system includes features designed to enhance the physical safety and trust of its users.

- **Provider Verification:** All service providers must undergo a mandatory verification process, including ID proof submission, before their profile becomes active on the platform.
- **SOS Feature Reliability:** The in-app SOS emergency feature must have a 99.99% success rate in transmitting the alert to the designated contacts and nearby providers under standard network conditions.
- **Data Integrity and Backup:** The system must ensure the integrity of user data. Automated daily backups of the entire database must be performed, with a defined data recovery plan to restore service within 2 hours in case of a critical failure.

3.2.4 Quality

The overall quality of the system will be judged by its reliability, usability, and maintainability.

- **Availability (Uptime):** The platform must have a minimum uptime of 99.8%, excluding scheduled maintenance periods. Scheduled maintenance will be communicated to users at least 24 hours in advance.
- **Usability:** The user interface must be intuitive and easy to navigate. A new user should be able to register, search for a service, and complete a booking without requiring external assistance or a tutorial.
- **Compatibility:** The web application must be fully functional and render correctly on the latest versions of major web browsers (Google Chrome, Mozilla Firefox, Safari, and Microsoft Edge).
- **Accessibility:** The platform should adhere to basic accessibility guidelines (WCAG 2.1 Level AA) to be usable by people with disabilities, including features like proper color contrast and screen reader compatibility.
- **Maintainability:** The source code must be well-documented, modular, and follow standard coding practices to facilitate future updates, bug fixes, and feature additions.

Appendix A

Documentation Tools

This section describes the tools used for creating and maintaining project documentation for *ShebaBondhu*.

A.1 API Documentation: JSDoc

- **Tool:** JSDoc
- **Purpose:** Auto-generates HTML documentation from JavaScript code comments
- **Why Chosen:** Lightweight, no compilation needed, integrates with VS Code

A.1.1 Installation & Usage

1. Install: `npm install -save-dev jsdoc`
2. Add comments in code using `/** ... */` format
3. Add script in package.json: `"docs": "jsdoc src -d docs"`
4. Generate docs: `npm run docs`
5. View: Open `docs/index.html` in browser

A.2 Version Control

- **Repository:** GitHub
- **Documentation:** GitHub wiki

A.3 Diagram Tools

- **Tool:** Canva, dbDiagram
- **Purpose:** System architecture, use case, and database diagrams

A.4 Document Preparation

- **Tool:** L^AT_EX
- **Purpose:** SRS and formal project documentation

Appendix B

Coding Standards

This section outlines the coding conventions for the *ShebaBondhu* project based on industry-standard style guides.

B.1 Recommended Style Guide: Airbnb

- **Chosen:** Customized Airbnb JavaScript Style Guide
- **Why:** Popular in React/Next.js ecosystem, strict rules reduce ambiguity, excellent ESLint support via `eslint-config-airbnb`
- **Alternatives Evaluated:** Google Style Guide, StandardJS

B.2 Key Conventions

B.2.1 Imports

- Always at top of file, grouped: Node built-ins → External libs → Internal modules
- Use absolute imports with `@/` prefix (e.g., `@/components/Navbar`)
- Sort alphabetically within groups, no unused imports

B.2.2 Naming

- **Variables/Functions:** camelCase (e.g., `userName`, `fetchUser`)
- **Constants:** UPPER_CASE (e.g., `API_URL`)
- **Components/Classes/Types:** PascalCase (e.g., `BookingCard`, `IUser`)

B.2.3 Variables

- Use `const` by default
- Use `let` only when reassignment needed
- Never use `var`

B.2.4 Functions

- Prefer arrow functions for anonymous functions
- Async functions must use `try/catch`
- Keep functions pure, avoid side-effects

B.2.5 Formatting

- Indentation: 4 spaces (VS Code default)
- Semicolons: Required
- Quotes: Single quotes for strings
- Trailing commas in multi-line arrays/objects

B.3 Next.js Structure

B.3.1 Pages (`page.tsx`)

- Components → PascalCase
- Logic first, JSX return at bottom
- Use semantic HTML, avoid inline styles

B.3.2 API Routes (`route.ts`)

- Use async functions with `try/catch`
- Always return `Response.json()` with status codes

B.3.3 Database (`lib/dbConnect.ts`)

- Singleton pattern to avoid multiple connections
- Named export function

B.3.4 Models (`models/*.ts`)

- One model per file, PascalCase names
- Define interface → schema → export model

B.3.5 Components (`components/*.tsx`)

- PascalCase file and function names
- Props must be typed with interface
- Pure functional components only

Appendix C

Use Case Diagrams

This section presents the use case diagrams that illustrate the interactions between actors (users) and the *ShebaBondhu* system.

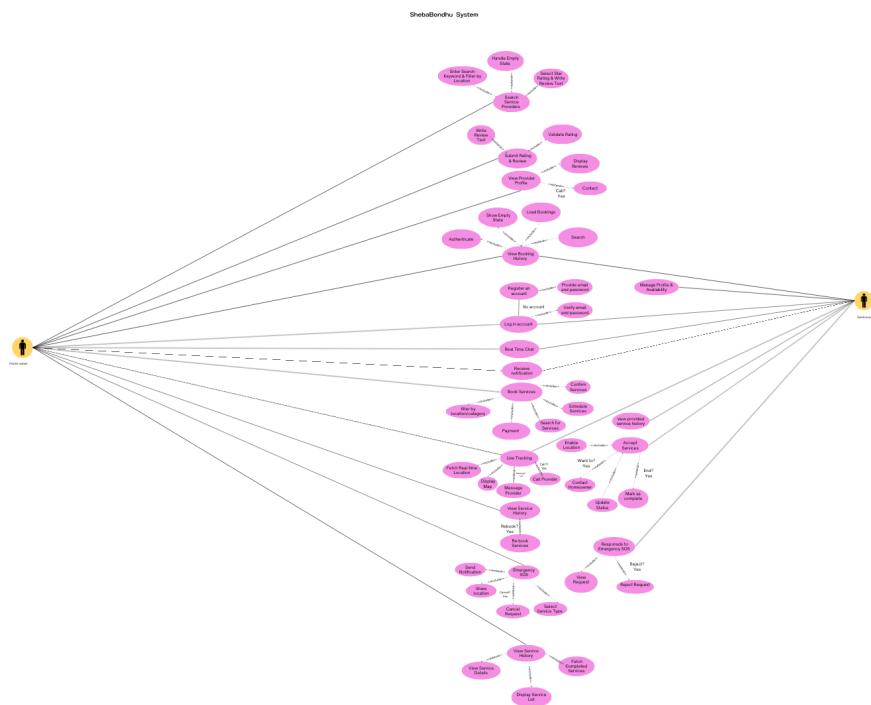


Figure C.1: Main System Use Case Diagram

Appendix D

Extended Use Case Descriptions

This section provides detailed descriptions of key use cases including preconditions, main flow, alternative flows, and postconditions.

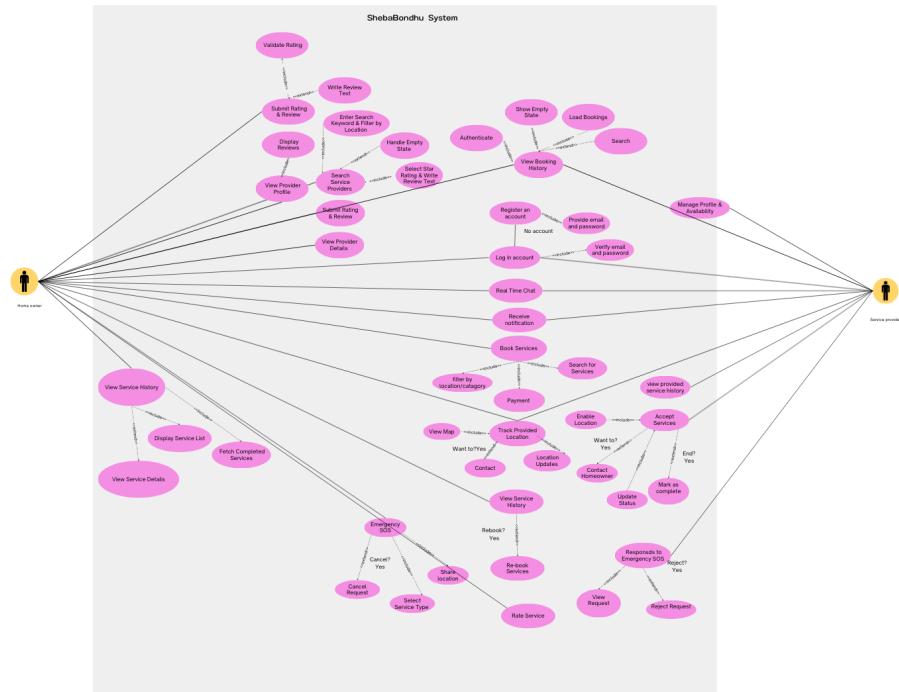


Figure D.1: Extended Use Case Diagram

Appendix E

Database Schema

E.1 Entity-Relationship Overview

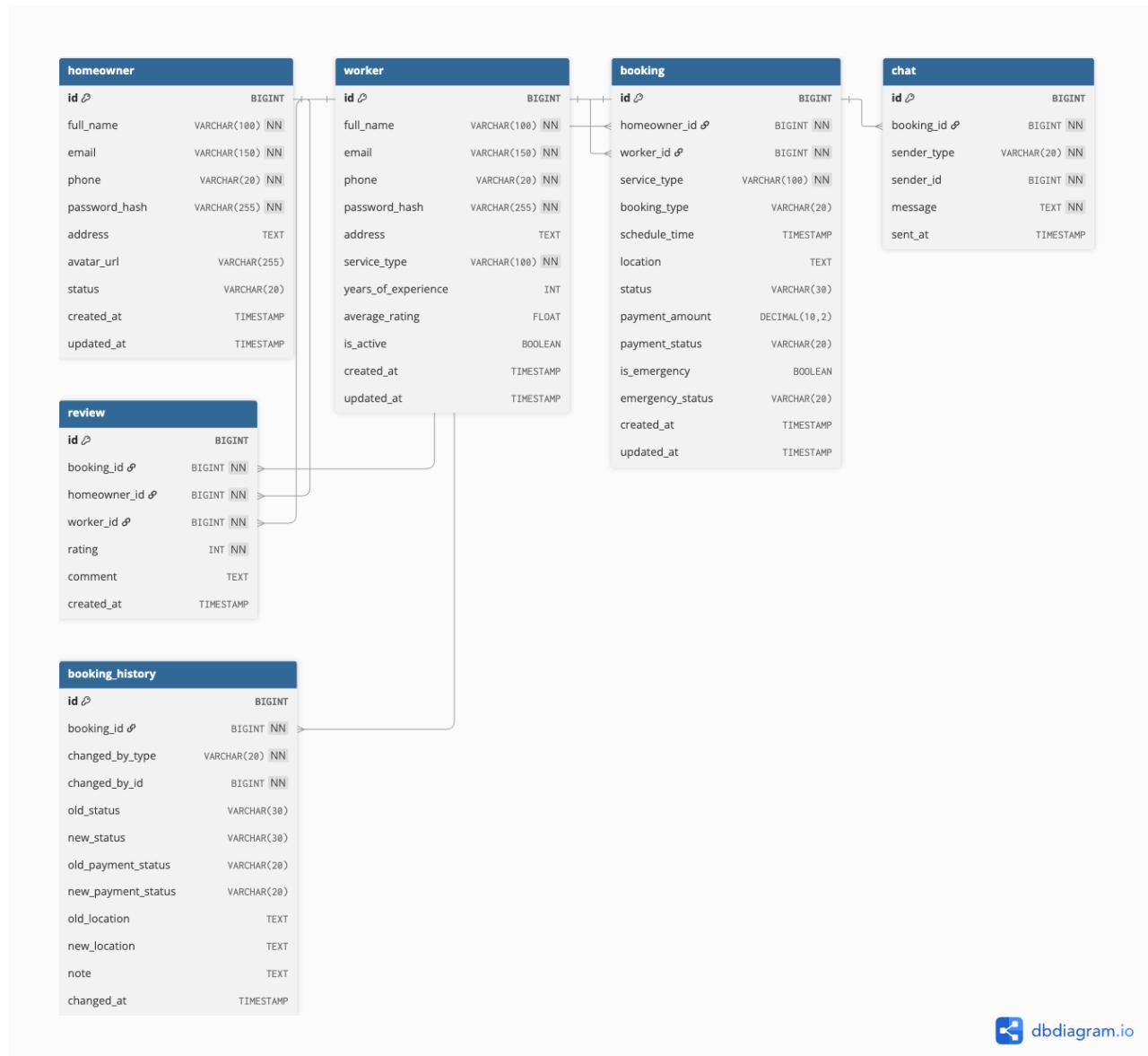


Figure E.1: Schema Diagram

Appendix F

Testing Strategy

This section outlines the testing tools and approach for *ShebaBondhu*.

F.1 Recommended Tool: Jest

- **Tool:** Jest (by Meta/Facebook)
- **Type:** Unit & Integration Testing
- **Why Chosen:** Zero-config, works with Next.js out-of-the-box, built-in mocking, snapshot testing, coverage reporting

F.2 Implementation

1. Install: `npm install -save-dev jest @types/jest`
2. Add script in `package.json`: `"test": "jest"`
3. Create test files: `*.test.js` or `*.spec.js`
4. Run tests: `npm test`
5. Coverage report: `npm test --coverage`

Appendix G

Architectural Pattern

This section describes the architecture pattern selection for *ShebaBondhu*.

G.1 Selected Pattern: MVVM

- **Pattern:** Model-View-ViewModel (MVVM)
- **Why Chosen:** Aligns with React + Next.js component-based development, supports interactive dashboards and real-time updates

G.2 Why MVVM Fits ShebaBondhu

- Real-time updates (service acceptance, tracking status)
- Different dashboards for roles (homeowner, provider)
- Heavy client-side state handling
- Clean separation of UI logic from data state
- Potential future mobile app conversion

G.3 MVVM Layer Responsibilities

Layer	Responsibility
Model	Fetch and update raw data (API calls)
ViewModel	Holds state, handles logic, updates View
View	Renders UI based on ViewModel, observes state
Page	Connects View + ViewModel

Table G.1: MVVM Layer Responsibilities

G.4 Folder Structure (MVVM)

```
/components
  ServiceList.js      <-- View
```

```
/viewmodels
  ServiceViewModel.js  <-- ViewModel
/models
  ServiceModel.js      <-- Model (API functions)
/pages
  dashboard.js         <-- Next.js page
```

G.5 Benefits of MVVM for ShebaBondhu

- **Separation of Concerns:** Each layer has distinct responsibility
- **Reusable Logic:** ViewModel can be used across multiple components
- **Automatic UI Updates:** UI reacts to ViewModel state changes
- **Testability:** Business logic testable without rendering UI
- **Scalability:** Easy to extend with new features