

Table of Contents

| | |
|---|----|
| HireLink - Local Service Provider Platform | 3 |
| Academic Project Documentation | 3 |
| HireLink | 3 |
| Table of Contents | 3 |
| 1. Executive Summary..... | 4 |
| Key Highlights | 4 |
| Project Objectives..... | 4 |
| 2. Introduction | 4 |
| 2.1 Problem Statement..... | 4 |
| 2.2 Proposed Solution | 4 |
| 2.3 Scope of the Project..... | 5 |
| 2.4 Target Audience..... | 5 |
| 3. Software Requirements Specification (SRS)..... | 5 |
| 3.1 Functional Requirements | 5 |
| 3.2 Non-Functional Requirements..... | 7 |
| 3.3 Use Case Specifications | 7 |
| 4. System Architecture..... | 9 |
| 4.1 High-Level Architecture..... | 9 |
| 4.2 Technology Stack | 10 |
| 5. Data Flow Diagrams (DFD)..... | 11 |
| 5.1 Context Diagram (Level 0 DFD) | 11 |
| 5.2 Level 1 DFD | 12 |
| 5.3 Level 2 DFD - Booking Management | 13 |
| 6. Entity-Relationship Diagrams..... | 14 |
| 6.1 Complete ER Diagram..... | 14 |
| 6.2 Entity Descriptions | 14 |
| 6.3 Relationship Summary | 15 |
| 7. Database Design | 15 |
| 7.1 Data Dictionary | 15 |
| 7.2 Booking Status State Machine | 18 |
| 8. Workflow Diagrams..... | 20 |

| | |
|--|----|
| 8.1 User Registration Workflow | 20 |
| 8.2 Service Booking Workflow | 22 |
| 9. Technical Specifications | 23 |
| 9.1 Frontend Architecture | 23 |
| 9.2 Backend Architecture | 23 |
| 10. API Documentation | 24 |
| 10.1 API Overview | 24 |
| 10.2 Authentication Endpoints | 24 |
| 10.3 Booking Endpoints | 25 |
| 10.4 API Error Responses..... | 26 |
| 11. User Interface Design | 26 |
| 11.1 Design Principles..... | 26 |
| 11.2 Color Palette | 26 |
| 11.3 Key Pages..... | 26 |
| 12. Security Implementation | 27 |
| 12.1 Authentication Flow | 27 |
| 12.2 Security Measures | 27 |
| 12.3 Role-Based Access Control | 28 |
| 13. Testing Strategy | 28 |
| 13.1 Testing Pyramid | 28 |
| 13.2 Test Categories..... | 28 |
| 14. Deployment Architecture..... | 29 |
| 14.1 Deployment Diagram | 29 |
| 14.2 Docker Configuration..... | 29 |
| 14.3 Environment Configuration..... | 30 |
| 15. Conclusion..... | 30 |
| 15.1 Project Summary | 30 |
| 15.2 Future Enhancements..... | 30 |
| 15.3 Lessons Learned | 30 |
| Appendices..... | 31 |
| Appendix A: Glossary | 31 |
| Appendix B: Demo Accounts..... | 31 |
| Appendix C: API Endpoints Summary..... | 31 |

HireLink - Local Service Provider Platform

Academic Project Documentation

UIT, ADOOR, Academic Year 2025-2026

Academic Year 2025-2026

HireLink

A Full-Stack Web Application for Connecting Customers with Local Service Providers

Submitted in partial fulfillment of the requirements for the BCA , Graduate Academic Project

Project Team: HireLink Development Team

Submission Date: January 2026

Table of Contents

1. Executive Summary
2. Introduction
3. Software Requirements Specification (SRS)
4. System Architecture
5. Data Flow Diagrams (DFD)
6. Entity-Relationship Diagrams
7. Database Design
8. Workflow Diagrams
9. Technical Specifications
10. API Documentation
11. User Interface Design
12. Security Implementation
13. Testing Strategy
14. Deployment Architecture

15. Conclusion

1. Executive Summary

HireLink is a comprehensive full-stack web application designed to bridge the gap between customers seeking home services and verified local service providers. The platform facilitates seamless booking, tracking, and management of services such as electrical work, plumbing, carpentry, cleaning, painting, and more.

Key Highlights

| Aspect | Description |
|------------------|--|
| Platform Type | Web-based Service Marketplace |
| Target Users | Customers, Service Providers, Administrators |
| Primary Services | Electrical, Plumbing, Carpentry, Cleaning, Painting, AC & Appliances |
| Technology Stack | React.js, Spring Boot, MySQL, Docker |
| Architecture | Microservices-ready Monolith |

Project Objectives

1. Create a user-friendly platform for booking home services
 2. Implement a robust verification system for service providers
 3. Enable real-time booking management and tracking
 4. Provide transparent pricing and rating mechanisms
 5. Ensure secure authentication and data protection
-

2. Introduction

2.1 Problem Statement

The home services industry faces significant challenges in connecting customers with reliable service providers:

- **Lack of transparency** in pricing and service quality
- **Difficulty in verifying** provider credentials
- **Inefficient scheduling** and booking processes
- **No standardized feedback** mechanisms
- **Trust deficit** between customers and providers

2.2 Proposed Solution

HireLink addresses these challenges by providing a centralized digital platform that:

1. **Verifies Service Providers** through KYC documentation and background checks
2. **Standardizes Pricing** with transparent cost structures
3. **Enables Easy Booking** with intuitive scheduling interfaces
4. **Implements Rating Systems** for quality assurance
5. **Provides Real-time Tracking** of service status

2.3 Scope of the Project

In Scope: - User registration and authentication (Customers, Providers, Admins) - Service category management - Booking creation, tracking, and management - Provider profile and service management - Review and rating system - Location-based service discovery - Multi-role dashboard interfaces

Out of Scope (Future Enhancements): - Payment gateway integration - Real-time chat/messaging - Mobile native applications - AI-based service recommendations

2.4 Target Audience

| User Type | Description |
|-------------------|--|
| Customers | Individuals seeking home services for residential needs |
| Service Providers | Skilled professionals offering home services |
| Administrators | Platform managers overseeing operations and verification |

3. Software Requirements Specification (SRS)

3.1 Functional Requirements

3.1.1 User Management Module

| ID | Requirement | Priority |
|--------|---|----------|
| FR-001 | Users shall be able to register with email/phone and password | High |
| FR-002 | Users shall be able to login using credentials | High |
| FR-003 | System shall support OTP-based verification | High |
| FR-004 | Users shall be able to update their profile information | Medium |
| FR-005 | Users shall be able to manage multiple addresses | Medium |
| FR-006 | System shall support role-based access | High |

| ID | Requirement | Priority |
|-----------------------------|-------------|----------|
| (Customer, Provider, Admin) | | |

3.1.2 Service Provider Module

| ID | Requirement | Priority |
|--------|--|----------|
| FR-007 | Providers shall be able to create and manage service listings | High |
| FR-008 | Providers shall submit KYC documents for verification | High |
| FR-009 | Providers shall set availability schedules | Medium |
| FR-010 | Providers shall define service areas by pincode/radius | Medium |
| FR-011 | Providers shall accept/reject booking requests | High |
| FR-012 | System shall track provider statistics (ratings, bookings, earnings) | High |

3.1.3 Service Category Module

| ID | Requirement | Priority |
|--------|---|----------|
| FR-013 | System shall support hierarchical service categories | High |
| FR-014 | Categories shall have configurable pricing structures | Medium |
| FR-015 | Featured categories shall be highlighted on homepage | Low |
| FR-016 | Categories shall support multiple pricing units | Medium |

3.1.4 Booking Module

| ID | Requirement | Priority |
|--------|--|----------|
| FR-017 | Customers shall be able to create service bookings | High |
| FR-018 | System shall generate unique booking numbers | High |
| FR-019 | Bookings shall support multiple status states | High |
| FR-020 | Customers shall track booking status in real-time | High |
| FR-021 | System shall prevent duplicate active bookings | Medium |

| ID | Requirement | Priority |
|--------|--|----------|
| FR-022 | Bookings shall support cancellation with reason tracking | Medium |

3.1.5 Review and Rating Module

| ID | Requirement | Priority |
|--------|--|----------|
| FR-023 | Customers shall rate completed services | High |
| FR-024 | System shall support multi-dimensional ratings | Medium |
| FR-025 | Reviews shall be visible on provider profiles | High |
| FR-026 | System shall calculate and display average ratings | High |

3.2 Non-Functional Requirements

3.2.1 Performance Requirements

| ID | Requirement | Metric |
|---------|--|------------|
| NFR-001 | Page load time shall be under 3 seconds | < 3s |
| NFR-002 | API response time shall be under 500ms | < 500ms |
| NFR-003 | System shall support 1000 concurrent users | 1000 users |
| NFR-004 | Database queries shall execute under 100ms | < 100ms |

3.2.2 Security Requirements

| ID | Requirement | Implementation |
|---------|--|----------------------|
| NFR-005 | Passwords shall be encrypted using bcrypt | BCrypt hashing |
| NFR-006 | API endpoints shall be secured with JWT tokens | JWT Authentication |
| NFR-007 | Sensitive data shall be encrypted at rest | AES Encryption |
| NFR-008 | System shall implement role-based access control | Spring Security RBAC |

3.2.3 Usability Requirements

| ID | Requirement |
|---------|--|
| NFR-009 | UI shall be responsive across devices (mobile, tablet, desktop) |
| NFR-010 | System shall provide clear error messages |
| NFR-011 | Navigation shall be intuitive with maximum 3 clicks to any feature |

3.3 Use Case Specifications

Use Case UC-001: User Registration

| Field | Description |
|-------------|-------------|
| Use Case ID | UC-001 |

| Field | Description |
|---------------|---|
| Name | User Registration |
| Actors | Customer, Service Provider |
| Precondition | User has valid email/phone number |
| Main Flow | 1. User navigates to registration page 2. User selects account type 3. User enters required details 4. System validates input 5. System sends OTP verification 6. User verifies OTP 7. Account is created |
| Postcondition | User account is created and verified |

Use Case UC-002: Book a Service

| Field | Description |
|---------------|--|
| Use Case ID | UC-002 |
| Name | Book a Service |
| Actors | Customer |
| Precondition | Customer is logged in |
| Main Flow | 1. Customer browses/searches services 2. Customer selects a service 3. Customer views service details 4. Customer clicks “Book Now” 5. Customer selects date and time 6. Customer enters service address 7. Customer describes the issue 8. System creates booking |
| Postcondition | Booking is created with unique booking number |

4. System Architecture

4.1 High-Level Architecture

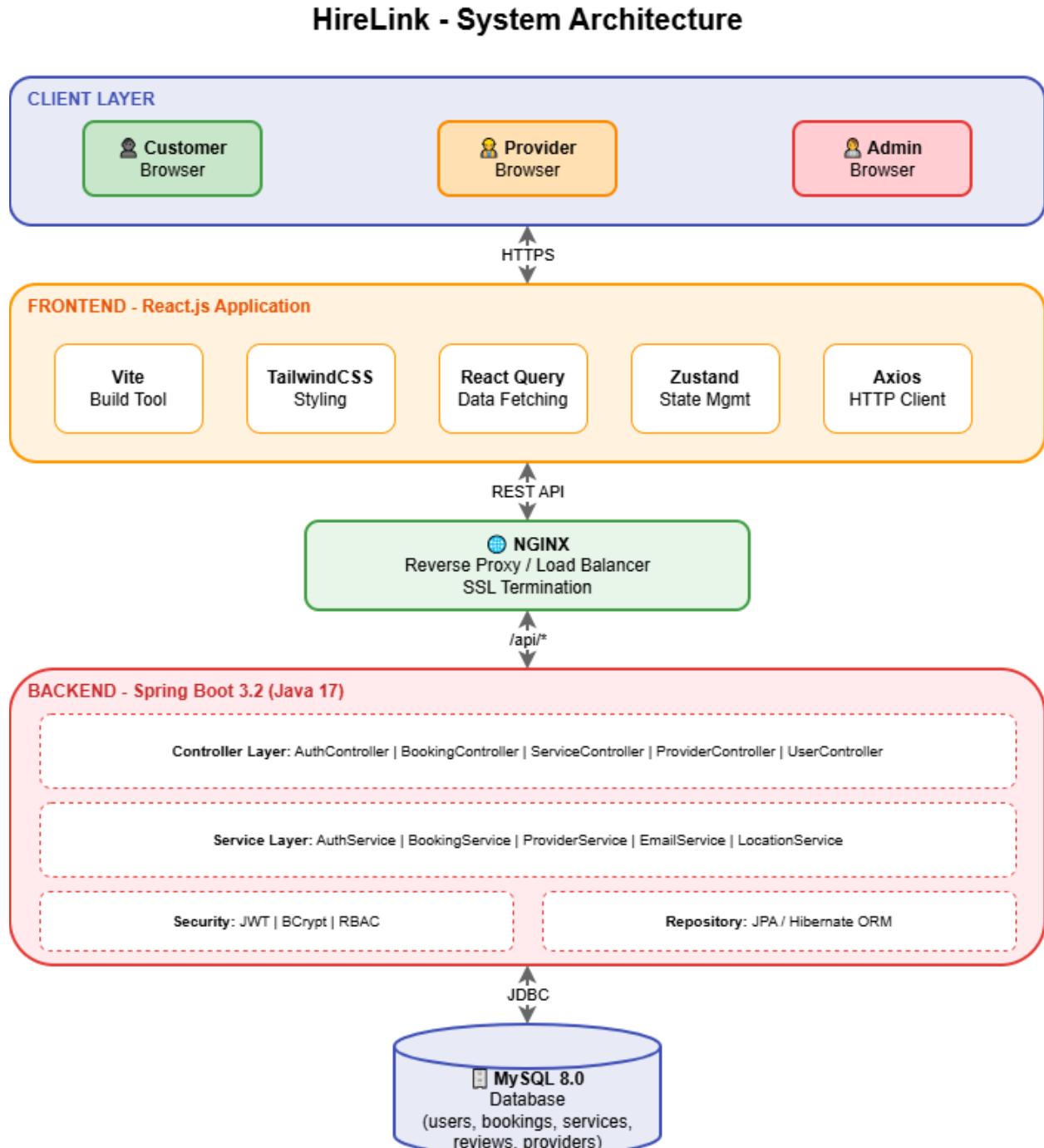


Figure 4.1: High-Level System Architecture showing the three-tier architecture with Client Layer (React.js), Application Layer (Spring Boot), and Data Layer (MySQL)

The HireLink system follows a three-tier architecture:

Client Layer: - React.js single-page application - Vite build tool for fast development - TailwindCSS for styling - Zustand for state management - React Query for data fetching

Application Layer: - Spring Boot 3.2 REST API - Spring Security for authentication - JWT token-based authorization - JPA/Hibernate for ORM

Data Layer: - MySQL 8.0 relational database - Optimized indexes for performance - Stored procedures for complex operations

4.2 Technology Stack

| Layer | Technology | Version | Purpose |
|--------------------|-----------------|---------|--------------------------------|
| Frontend Framework | React.js | 18.x | UI Component Library |
| Build Tool | Vite | 5.x | Fast Development & Build |
| Styling | TailwindCSS | 3.x | Utility-First CSS |
| State Management | Zustand | 4.x | Lightweight State Store |
| API Client | React Query | 5.x | Data Fetching & Caching |
| Backend Framework | Spring Boot | 3.2.x | REST API Framework |
| Language | Java | 17 | Backend Programming |
| Security | Spring Security | 6.x | Authentication & Authorization |
| ORM | Hibernate/JPA | 6.x | Object-Relational Mapping |
| Database | MySQL | 8.0 | Relational Database |
| Containerization | Docker | 24.x | Application Containers |

5. Data Flow Diagrams (DFD)

5.1 Context Diagram (Level 0 DFD)

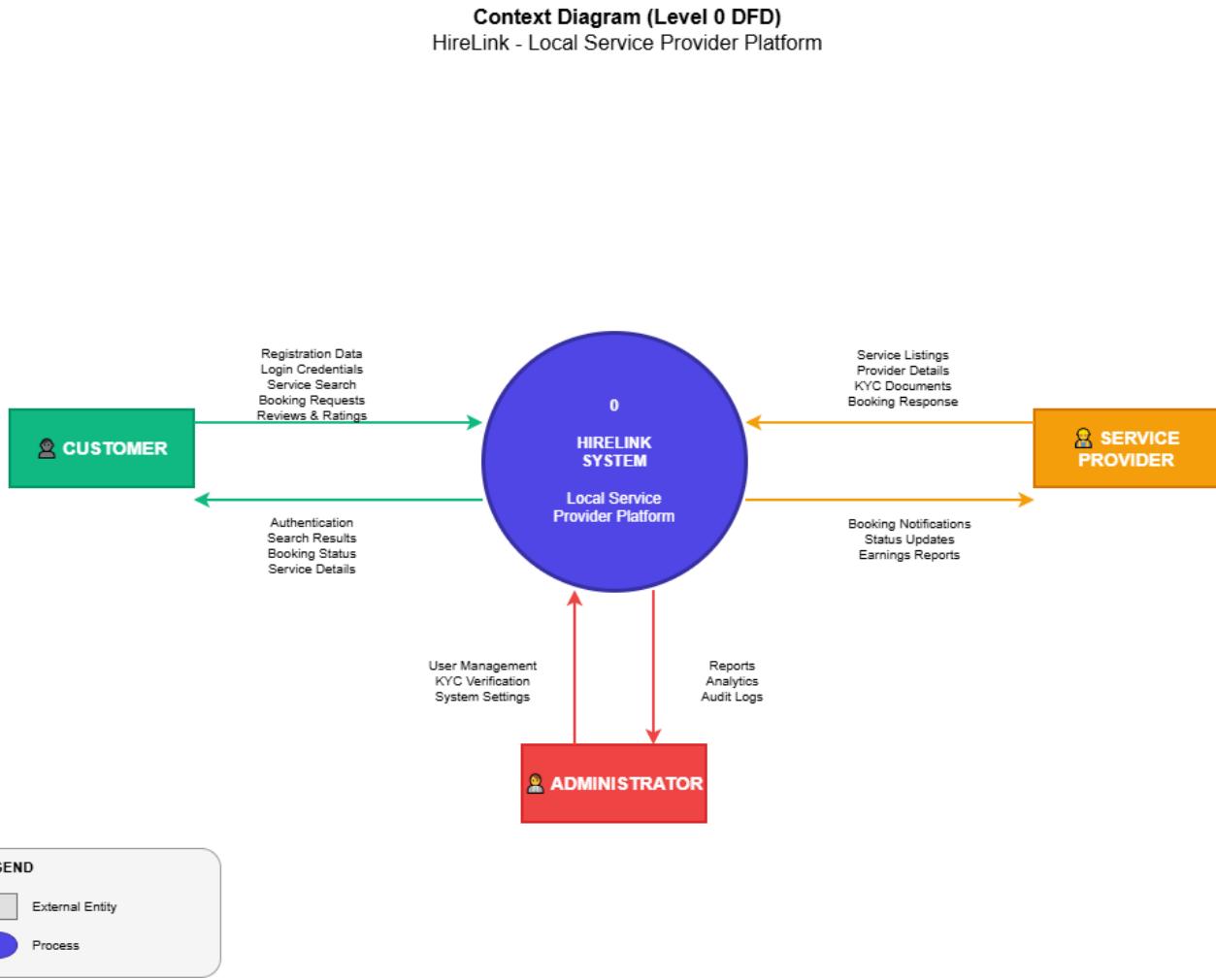


Figure 5.1: Context Diagram showing the HireLink system as a single process with three external entities: Customer, Service Provider, and Administrator

External Entities: - **Customer:** Sends registration data, login credentials, service search queries, booking requests, and reviews - **Service Provider:** Sends service listings, KYC documents, and booking responses - **Administrator:** Manages users, verifies KYC, and configures system settings

5.2 Level 1 DFD

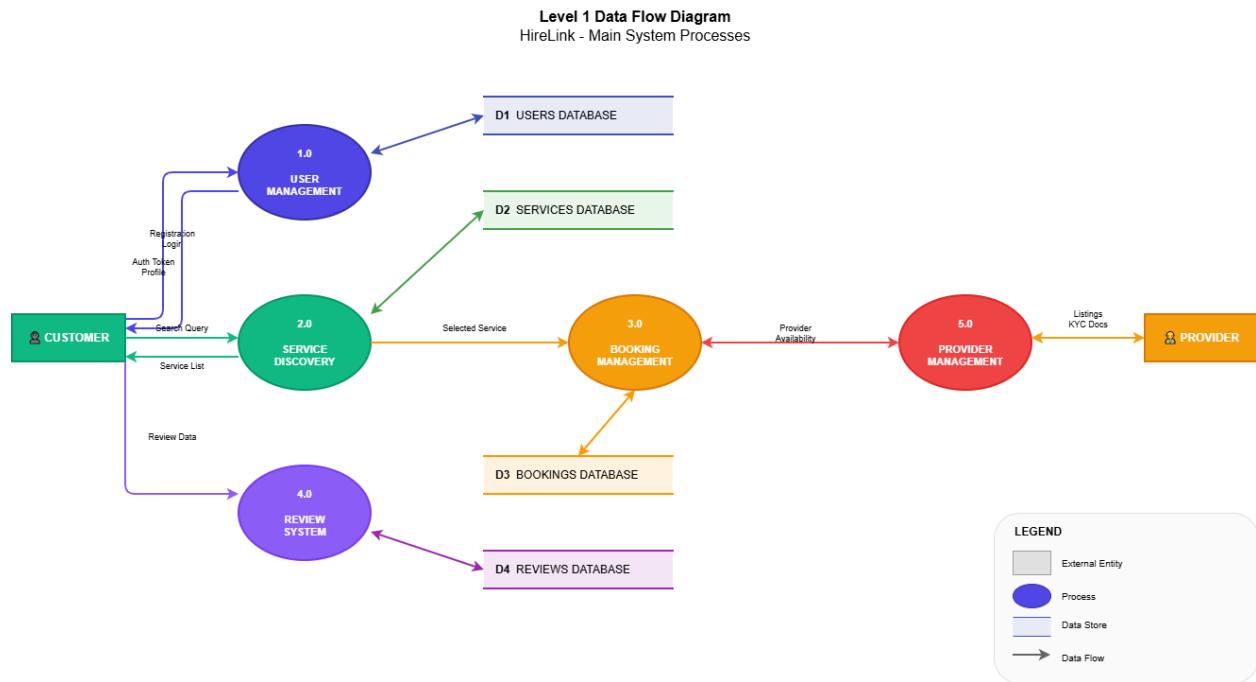


Figure 5.2: Level 1 DFD showing the main processes: User Management, Service Discovery, Booking Management, Review System, and Provider Management

Main Processes:

| Process | Description |
|-------------------------|--|
| 1.0 User Management | Handles registration, authentication, and profile management |
| 2.0 Service Discovery | Manages service search and category browsing |
| 3.0 Booking Management | Creates and manages service bookings |
| 4.0 Review System | Handles ratings and reviews |
| 5.0 Provider Management | Manages provider profiles and verification |

Data Stores:

| Data Store | Description |
|-----------------------|---|
| D1: Users Database | Stores user accounts and provider information |
| D2: Services Database | Stores service categories and listings |
| D3: Bookings Database | Stores booking records and status history |
| D4: Reviews Database | Stores customer reviews and ratings |

5.3 Level 2 DFD - Booking Management

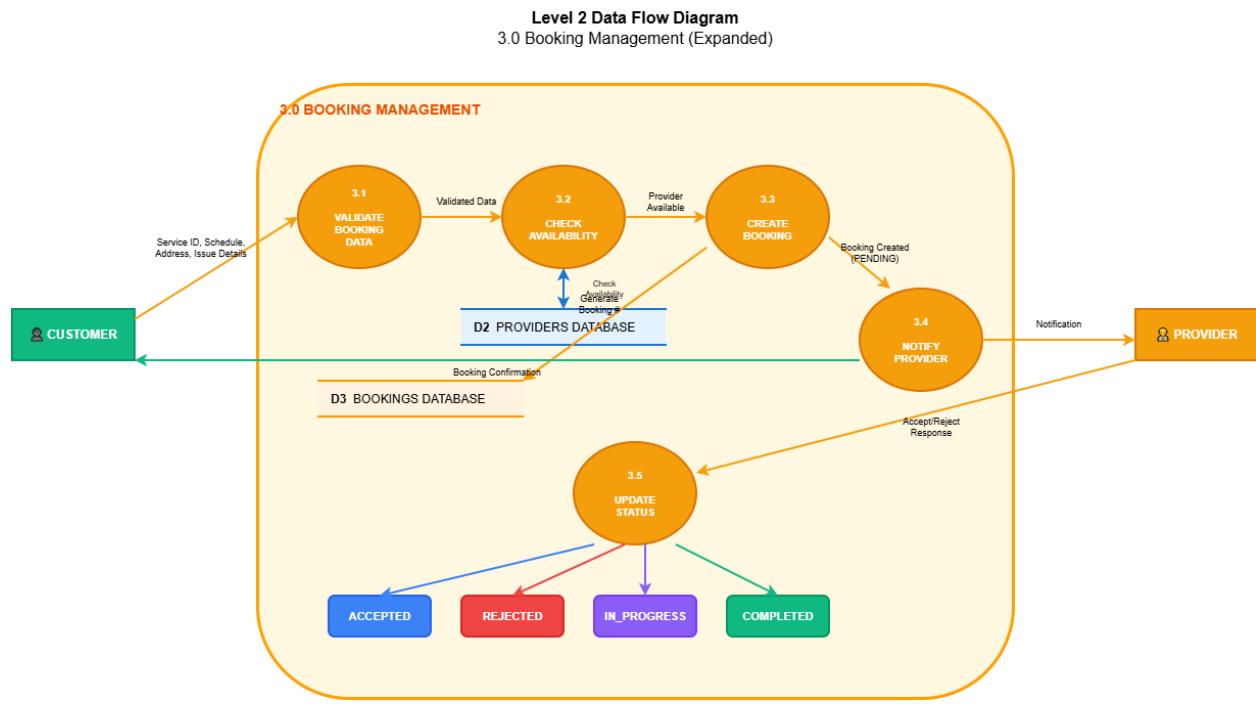


Figure 5.3: Level 2 DFD for Booking Management showing sub-processes: Validate Booking Data, Check Availability, Create Booking, Notify Provider, and Update Status

Sub-Processes:

| Process | Description |
|---------------------------|---|
| 3.1 Validate Booking Data | Validates customer input for booking |
| 3.2 Check Availability | Verifies provider availability |
| 3.3 Create Booking | Generates booking record with unique number |
| 3.4 Notify Provider | Sends notification to assigned provider |
| 3.5 Update Status | Manages booking status transitions |

6. Entity-Relationship Diagrams

6.1 Complete ER Diagram

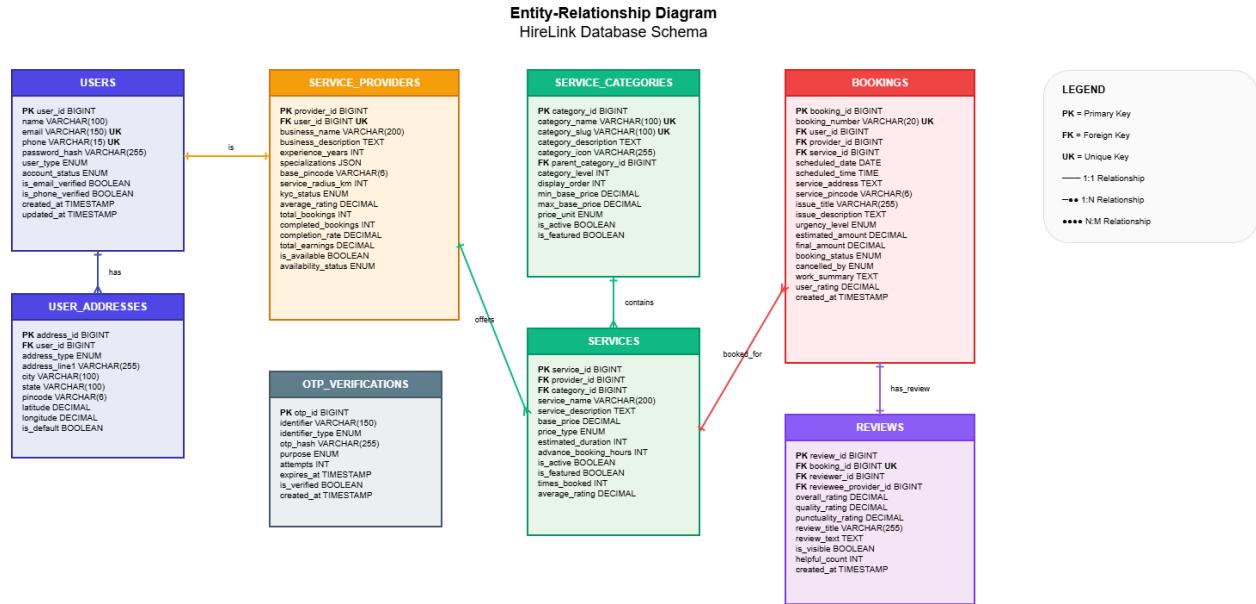


Figure 6.1: Entity-Relationship Diagram showing all database entities and their relationships

6.2 Entity Descriptions

| Entity | Description | Key Attributes |
|--------------------|----------------------------|---|
| USERS | All system users | user_id (PK), name, email, phone, user_type |
| USER_ADDRESSES | User address records | address_id (PK), user_id (FK), city, pincode |
| SERVICE_PROVIDERS | Provider business profiles | provider_id (PK), user_id (FK), business_name |
| SERVICE_CATEGORIES | Hierarchical categories | category_id (PK), category_name, parent_id |
| SERVICES | Service listings | service_id (PK), provider_id (FK), base_price |
| BOOKINGS | Service bookings | booking_id (PK), booking_number, status |
| REVIEWS | Customer reviews | review_id (PK), booking_id (FK), rating |
| OTP_VERIFICATIONS | OTP records | otp_id (PK), identifier, otp_hash |

6.3 Relationship Summary

| Parent Entity | Child Entity | Relationship | Cardinality |
|-------------------|------------------|--------------|-------------|
| Users | UserAddresses | Has | 1:N |
| Users | ServiceProviders | Is | 1:1 |
| Users | Bookings | Makes | 1:N |
| Users | Reviews | Writes | 1:N |
| ServiceProviders | Services | Offers | 1:N |
| ServiceProviders | Bookings | Receives | 1:N |
| ServiceCategories | Services | Contains | 1:N |
| Services | Bookings | BookedFor | 1:N |
| Bookings | Reviews | HasReview | 1:1 |

7. Database Design

7.1 Data Dictionary

Table: users

| Column | Data Type | Constraints | Description |
|-------------------|--------------|--------------------|-----------------------------|
| user_id | BIGINT | PK, AUTO_INCREMENT | Unique user identifier |
| name | VARCHAR(100) | NOT NULL | User's full name |
| email | VARCHAR(150) | UNIQUE | Email address |
| phone | VARCHAR(15) | UNIQUE | Phone number |
| password_hash | VARCHAR(255) | | BCrypt hashed password |
| user_type | ENUM | NOT NULL | CUSTOMER, PROVIDER, ADMIN |
| account_status | ENUM | | ACTIVE, INACTIVE, SUSPENDED |
| is_email_verified | BOOLEAN | DEFAULT FALSE | Email verification status |
| is_phone_verified | BOOLEAN | DEFAULT FALSE | Phone verification status |
| created_at | TIMESTAMP | NOT NULL | Record creation time |

Table: service_providers

| Column | Data Type | Constraints | Description |
|-------------|-----------|--------------------|----------------------------|
| provider_id | BIGINT | PK, AUTO_INCREMENT | Unique provider identifier |

| Column | Data Type | Constraints | Description |
|----------------------|--------------|--------------|----------------------------------|
| user_id | BIGINT | FK, UNIQUE | Reference to users table |
| business_name | VARCHAR(200) | | Business/brand name |
| business_description | TEXT | | Detailed description |
| experience_years | INT | DEFAULT 0 | Years of experience |
| base_pincode | VARCHAR(6) | | Primary service pincode |
| service_radius_km | INT | DEFAULT 10 | Service coverage radius |
| kyc_status | ENUM | | NOT_SUBMITTED, PENDING, VERIFIED |
| average_rating | DECIMAL(3,2) | DEFAULT 0 | Average service rating |
| total_bookings | INT | DEFAULT 0 | Total booking count |
| completed_bookings | INT | DEFAULT 0 | Completed booking count |
| is_available | BOOLEAN | DEFAULT TRUE | Current availability |

Table: bookings

| Column | Data Type | Constraints | Description |
|-----------------|--------------|--------------------|---------------------------|
| booking_id | BIGINT | PK, AUTO_INCREMENT | Unique booking identifier |
| booking_number | VARCHAR(20) | UNIQUE, NOT NULL | Human-readable booking ID |
| user_id | BIGINT | FK, NOT NULL | Customer reference |
| provider_id | BIGINT | FK, NOT NULL | Provider reference |
| service_id | BIGINT | FK, NOT NULL | Service reference |
| scheduled_date | DATE | NOT NULL | Scheduled service date |
| scheduled_time | TIME | NOT NULL | Scheduled start time |
| service_address | TEXT | NOT NULL | Service location address |
| service_pincode | VARCHAR(6) | NOT NULL | Service location pincode |
| issue_title | VARCHAR(255) | | Brief issue summary |

| Column | Data Type | Constraints | Description |
|-----------------------|---------------|-------------------|-------------------------------|
| issue_descrip tion | TEXT | | Detailed issue description |
| urgency_level | ENUM | DEFAULT 'MEDIUM' | LOW, MEDIUM, HIGH |
| estimated_a mount | DECIMAL(10,2) | | Initial cost estimate |
| final_amount | DECIMAL(10,2) | | Final billing amount |
| booking_statu s | ENUM | DEFAULT 'PENDING' | Current booking status |
| created_at | TIMESTAMP | NOT NULL | Booking creation time |

7.2 Booking Status State Machine

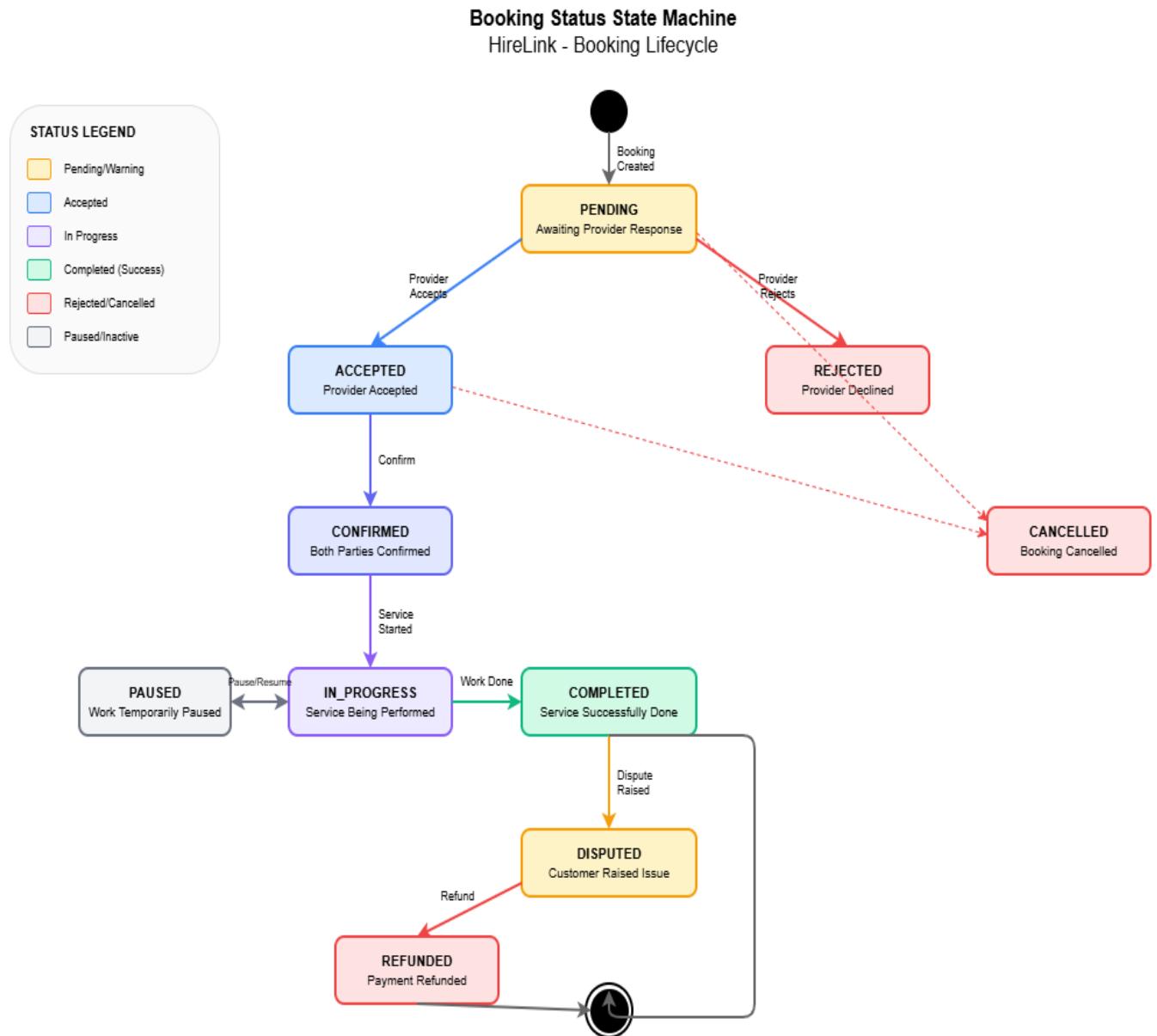


Figure 7.1: Booking Status State Machine showing all possible status transitions

Status Definitions:

| Status | Description |
|-------------|---|
| PENDING | Initial state, awaiting provider response |
| ACCEPTED | Provider accepted the booking |
| REJECTED | Provider declined the booking |
| CONFIRMED | Booking confirmed by both parties |
| IN_PROGRESS | Service is currently being performed |

| Status | Description |
|-----------|--------------------------------|
| PAUSED | Work temporarily paused |
| COMPLETED | Service successfully completed |
| CANCELLED | Booking cancelled by any party |
| DISPUTED | Customer raised a dispute |
| REFUNDED | Payment refunded after dispute |

8. Workflow Diagrams

8.1 User Registration Workflow

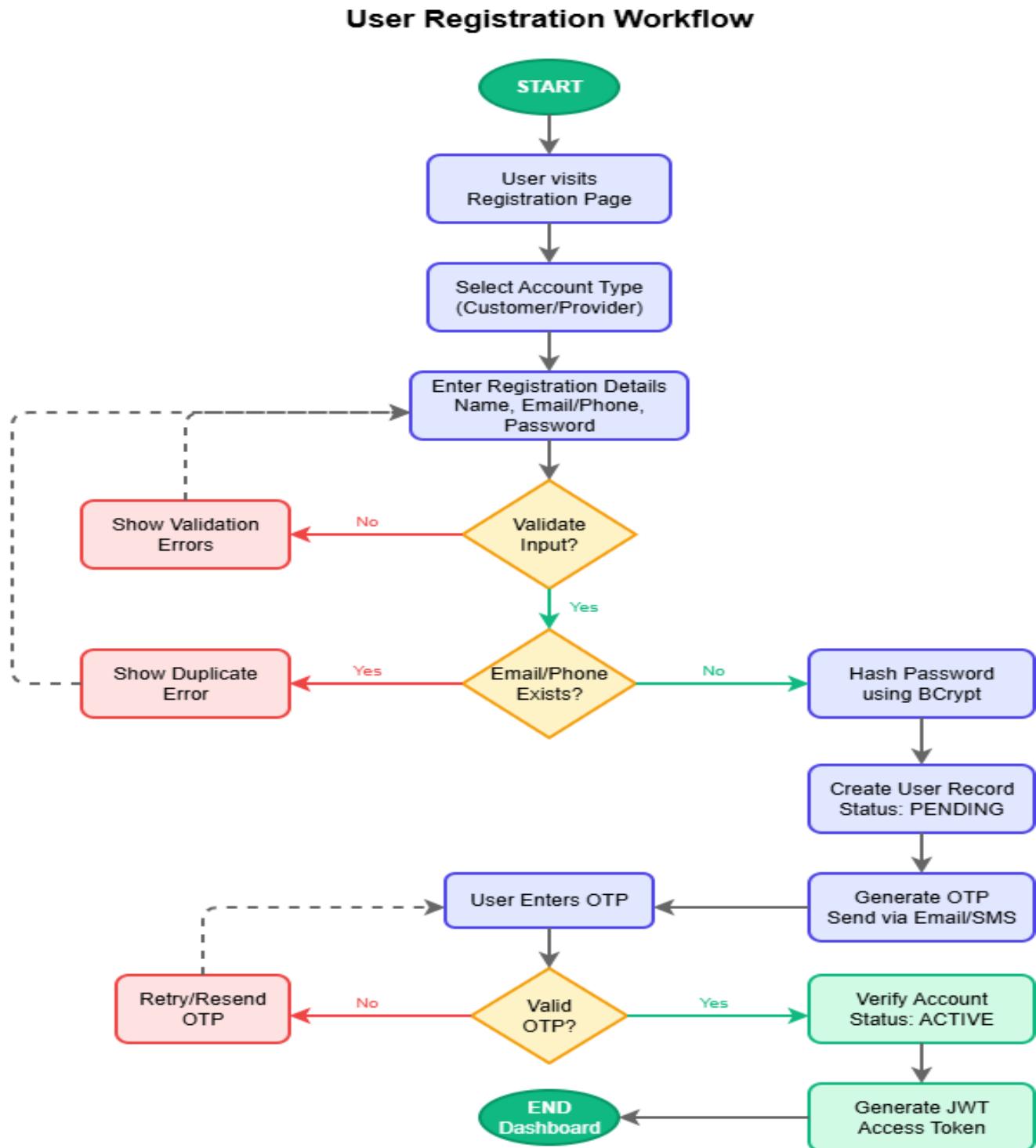


Figure 8.1: User Registration Flowchart showing the complete registration process from start to dashboard redirect

Workflow Steps:

1. User visits registration page
2. User selects account type (Customer/Provider)
3. User enters registration details (Name, Email/Phone, Password)
4. System validates input data
5. System checks for existing email/phone
6. System hashes password using BCrypt
7. System creates user record with PENDING status
8. System generates and sends OTP
9. User enters OTP
10. System verifies OTP
11. System activates account
12. System generates JWT token
13. User is redirected to dashboard

8.2 Service Booking Workflow

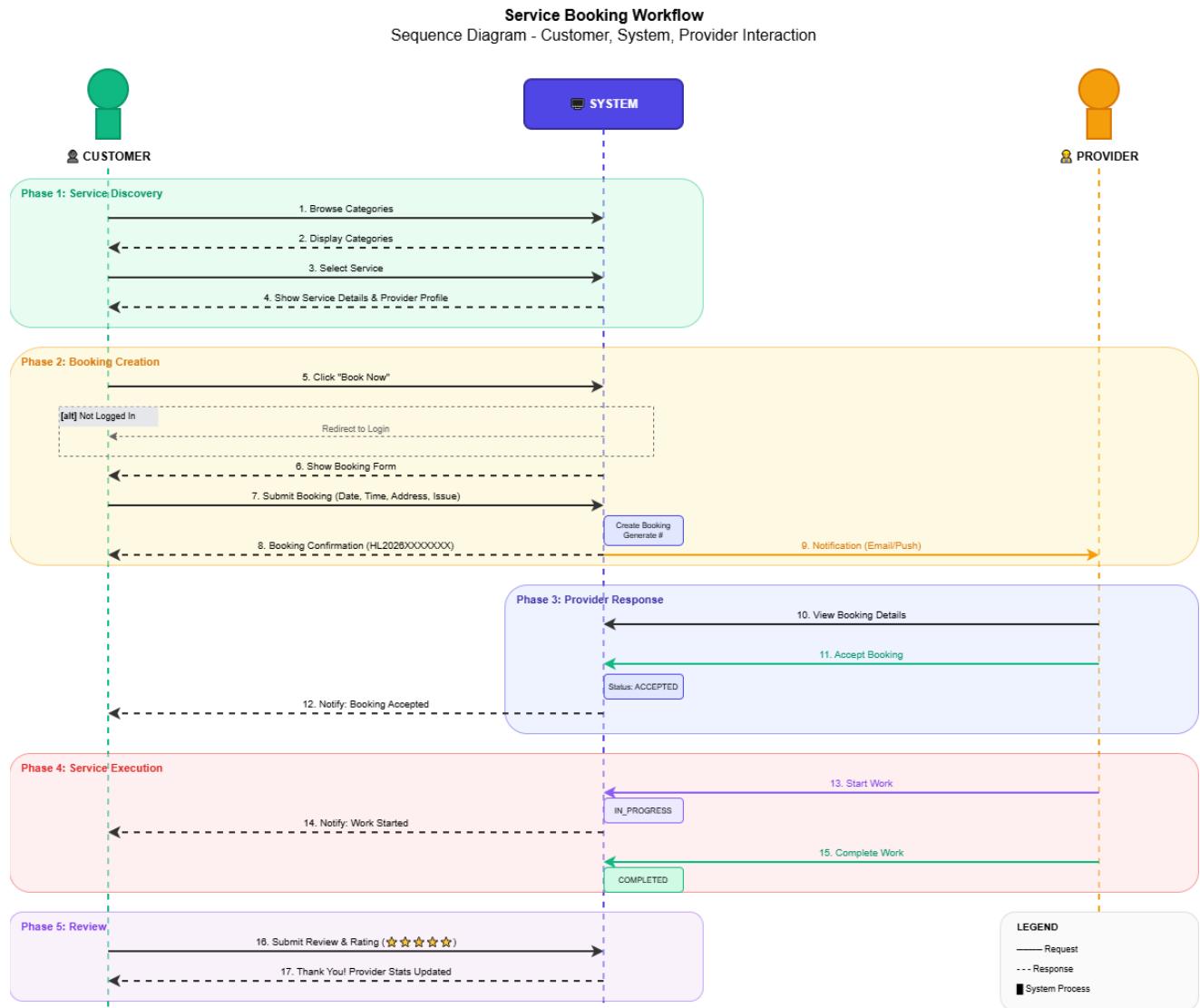


Figure 8.2: Service Booking Sequence Diagram showing interactions between Customer, System, and Provider

Workflow Phases:

Phase 1: Service Discovery - Customer browses categories - System displays available services - Customer selects service and views details

Phase 2: Booking Creation - Customer initiates booking - System validates login status - Customer enters booking details - System creates booking with PENDING status - System generates booking number (HL2026XXXXXXX) - Provider receives notification

Phase 3: Provider Response - Provider reviews booking request - Provider accepts or rejects - System updates status - Customer receives notification

Phase 4: Service Execution - Provider starts work (IN_PROGRESS) - Provider completes work (COMPLETED) - Customer receives completion notification

Phase 5: Review - Customer submits review and rating - System updates provider statistics

9. Technical Specifications

9.1 Frontend Architecture

Project Structure:

```
frontend/
  └── src/
    ├── components/      # Reusable UI components
    ├── pages/           # Page components (routes)
    ├── services/         # API service layer
    ├── context/          # State management
    ├── styles/           # Styling configuration
    ├── App.jsx           # Main application
    └── main.jsx          # Entry point
  └── public/            # Static assets
  └── package.json       # Dependencies
```

Key Dependencies:

| Package | Purpose |
|-----------------------|-------------------------|
| react | UI Component Library |
| react-router-dom | Client-side Routing |
| @tanstack/react-query | Data Fetching & Caching |
| zustand | State Management |
| axios | HTTP Client |
| tailwindcss | CSS Framework |

9.2 Backend Architecture

Project Structure:

```
backend/
  └── src/main/java/com/hirelink/
    ├── controller/      # REST API controllers
    ├── service/          # Business logic
    ├── repository/        # Data access
    ├── entity/           # JPA entities
    ├── dto/               # Data transfer objects
    └── security/         # Security components
```

```
└─ exception/      # Exception handling  
  pom.xml          # Maven dependencies
```

Key Dependencies:

| Dependency | Purpose |
|------------------------------|-----------------------|
| spring-boot-starter-web | REST API Framework |
| spring-boot-starter-data-jpa | Data Access Layer |
| spring-boot-starter-security | Authentication |
| mysql-connector-j | Database Driver |
| jjwt-api | JWT Token Generation |
| lombok | Boilerplate Reduction |

10. API Documentation

10.1 API Overview

| Base URL | Format | Authentication |
|----------|--------|------------------|
| /api | JSON | JWT Bearer Token |

10.2 Authentication Endpoints

POST /api/auth/register

Request:

```
{  
  "name": "John Doe",  
  "email": "john@example.com",  
  "phone": "9876543210",  
  "password": "securePassword123",  
  "userType": "CUSTOMER"  
}
```

Response (201 Created):

```
{  
  "success": true,  
  "message": "Registration successful",  
  "data": {  
    "userId": 1,  
    "name": "John Doe",  
    "userType": "CUSTOMER"  
  }  
}
```

POST /api/auth/login

Request:

```
{  
  "identifier": "john@example.com",  
  "password": "securePassword123"  
}
```

Response (200 OK):

```
{  
  "success": true,  
  "data": {  
    "accessToken": "eyJhbGciOiJIUzI1NiIs...".  
    "refreshToken": "eyJhbGciOiJIUzI1NiIs...".  
    "tokenType": "Bearer",  
    "expiresIn": 86400  
  }  
}
```

10.3 Booking Endpoints

POST /api/bookings

Request:

```
{  
  "serviceId": 1,  
  "providerId": 1,  
  "scheduledDate": "2026-02-15",  
  "scheduledTime": "10:00:00",  
  "serviceAddress": "42, Shanti Nagar",  
  "servicePincode": "560001",  
  "issueTitle": "Ceiling Fan Not Working",  
  "urgencyLevel": "MEDIUM"  
}
```

Response (201 Created):

```
{  
  "success": true,  
  "data": {  
    "bookingId": 1,  
    "bookingNumber": "HL202602150001",  
    "bookingStatus": "PENDING",  
    "estimatedAmount": 300.00  
  }  
}
```

10.4 API Error Responses

| HTTP Code | Error Type | Description |
|-----------|-----------------------|-------------------------------------|
| 400 | Bad Request | Invalid input or validation failure |
| 401 | Unauthorized | Missing or invalid authentication |
| 403 | Forbidden | Insufficient permissions |
| 404 | Not Found | Resource does not exist |
| 500 | Internal Server Error | Server-side error |

11. User Interface Design

11.1 Design Principles

- Responsive Design:** Mobile-first approach with breakpoints for tablet and desktop
- Consistent Visual Language:** Unified color palette, typography, and spacing
- Accessibility:** WCAG 2.1 compliant with proper contrast ratios
- Intuitive Navigation:** Maximum 3 clicks to reach any feature
- Feedback & Loading States:** Clear visual feedback for all user actions

11.2 Color Palette

| Color Name | Hex Code | Usage |
|-------------|----------|-------------------------|
| Primary-600 | #4F46E5 | Primary actions, links |
| Primary-700 | #4338CA | Primary hover states |
| Accent-500 | #F59E0B | CTA buttons, highlights |
| Success-500 | #22C55E | Success states |
| Warning-500 | #EAB308 | Warning messages |
| Error-500 | #EF4444 | Error states |
| Gray-900 | #111827 | Primary text |
| Gray-500 | #6B7280 | Secondary text |

11.3 Key Pages

| Page | Description |
|----------------|---|
| Home | Hero section, service categories, featured providers |
| Categories | Grid of all service categories |
| Service Detail | Service information, provider profile, booking button |
| Booking Form | Date/time selection, address input, issue description |
| My Bookings | List of customer's bookings with status filters |
| Booking Detail | Complete booking information, status timeline |

| Page | Description |
|------------------|--|
| Provider Profile | Business info, services offered, reviews |
| Profile | User settings, addresses, preferences |

12. Security Implementation

12.1 Authentication Flow

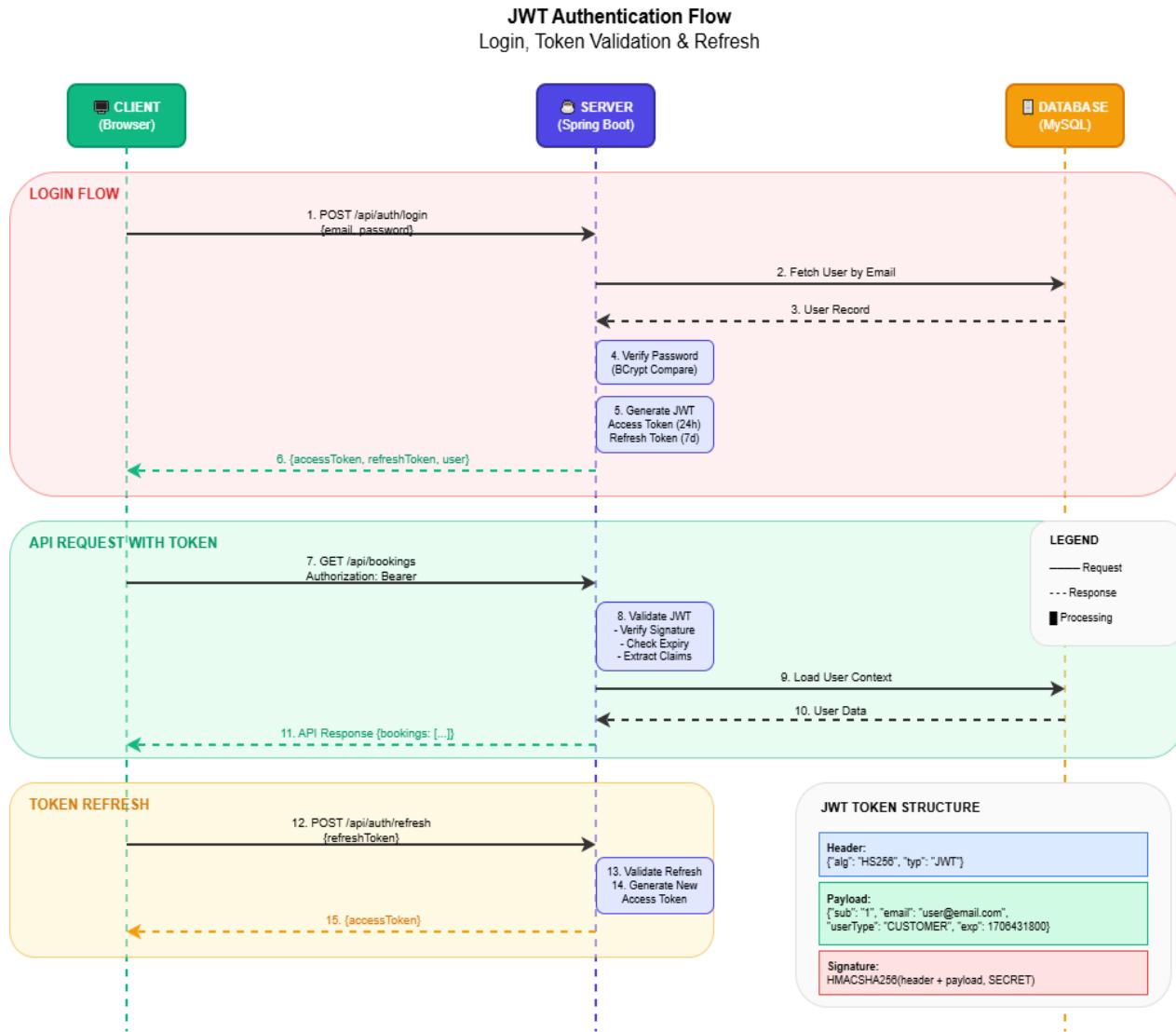


Figure 12.1: JWT Authentication Flow showing the login process and token validation

12.2 Security Measures

| Aspect | Implementation |
|------------------|-----------------------|
| Password Storage | BCrypt with 12 rounds |

| Aspect | Implementation |
|--------------------------|---------------------------|
| Token Type | JWT (JSON Web Token) |
| Access Token Expiry | 24 hours |
| Refresh Token Expiry | 7 days |
| Input Validation | Jakarta Bean Validation |
| SQL Injection Prevention | JPA Parameterized Queries |
| XSS Prevention | React automatic escaping |

12.3 Role-Based Access Control

| Role | Permissions |
|-------------|---|
| CUSTOMER | View services, Create bookings, Write reviews |
| PROVIDER | Manage services, Accept/reject bookings |
| ADMIN | User management, KYC verification |
| SUPER_ADMIN | All permissions + System settings |

13. Testing Strategy

13.1 Testing Pyramid

| Level | Coverage | Framework |
|-------------------|----------|------------------|
| Unit Tests | 80% | JUnit 5, Jest |
| Integration Tests | 15% | Spring Boot Test |
| E2E Tests | 5% | Cypress |

13.2 Test Categories

Unit Tests: - Backend service methods - Repository operations - Frontend components - Utility functions

Integration Tests: - API endpoint testing - Database operations - Authentication flow

End-to-End Tests: - Complete user workflows - Cross-browser testing



14. Deployment Architecture

14.1 Deployment Diagram

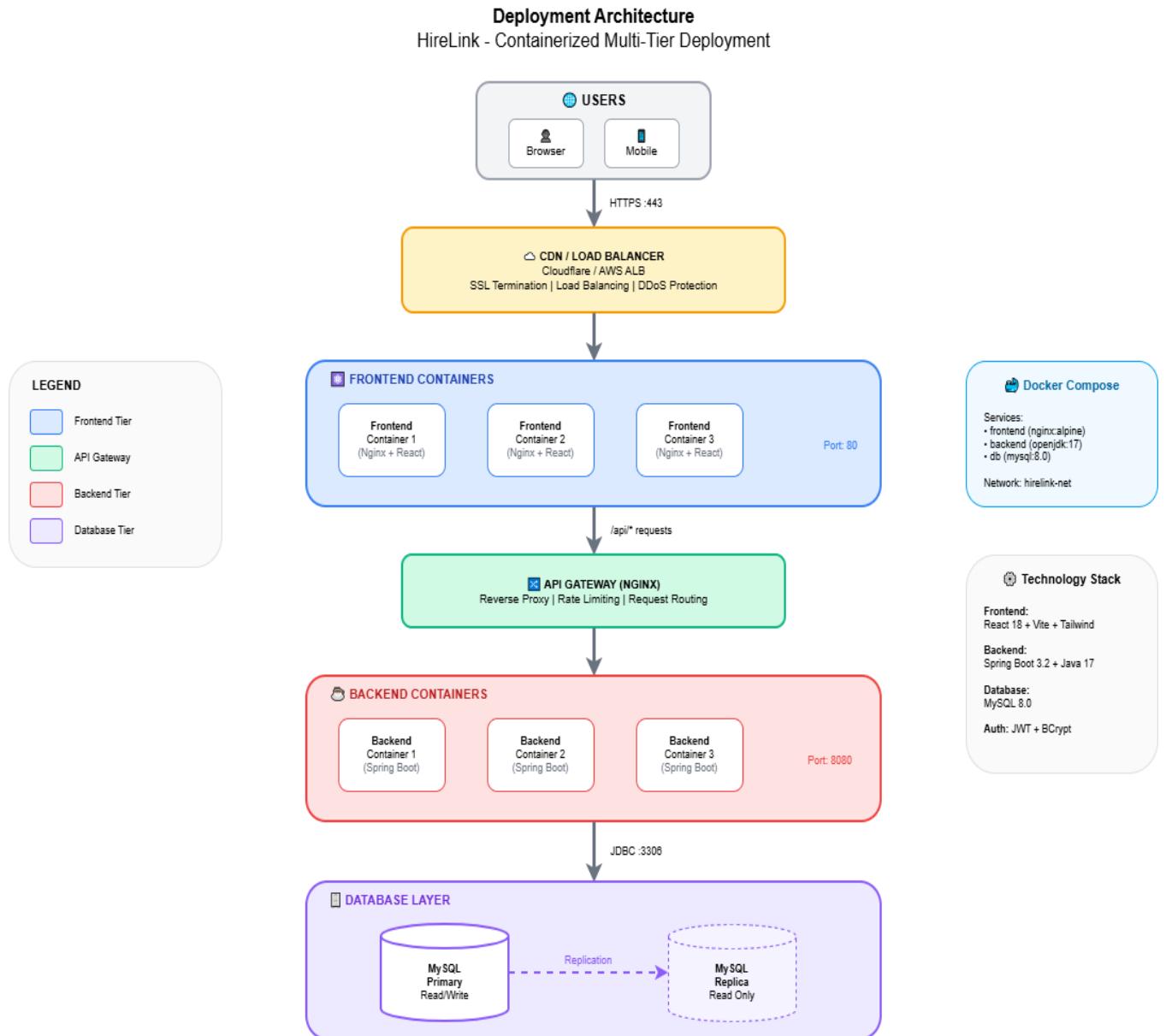


Figure 14.1: Deployment Architecture showing containerized services with load balancing

14.2 Docker Configuration

Services:

| Service | Technology | Port |
|----------|---------------|------|
| Frontend | Nginx + React | 80 |
| Backend | Spring Boot | 8080 |

| Service | Technology | Port |
|----------|------------|------|
| Database | MySQL 8.0 | 3306 |

14.3 Environment Configuration

| Environment | URL | Purpose |
|-------------|---------------------|------------------------|
| Development | localhost:3000 | Local development |
| Staging | staging.hirelink.in | Pre-production testing |
| Production | www.hirelink.in | Live application |

15. Conclusion

15.1 Project Summary

HireLink successfully addresses the challenges in the home services industry by providing a comprehensive digital platform that connects customers with verified local service providers.

Key Achievements:

| Objective | Status |
|------------------------------------|------------|
| User Registration & Authentication | ✓ Complete |
| Service Category Management | ✓ Complete |
| Booking System | ✓ Complete |
| Provider Management | ✓ Complete |
| Review System | ✓ Complete |
| Location-Based Discovery | ✓ Complete |
| Responsive UI | ✓ Complete |

15.2 Future Enhancements

1. Payment Gateway Integration (Razorpay/Stripe)
2. Real-Time Notifications (WebSocket)
3. Mobile Applications (iOS/Android)
4. AI Recommendations
5. Multi-Language Support
6. Analytics Dashboard

15.3 Lessons Learned

1. Spring Boot + React provides excellent developer productivity
2. Zustand offers simpler state management than Redux
3. Proper database normalization prevents scalability issues

4. Security should be implemented from the start
 5. Docker significantly simplifies deployment
-

Appendices

Appendix A: Glossary

| Term | Definition |
|------|-----------------------------------|
| JWT | JSON Web Token for authentication |
| KYC | Know Your Customer verification |
| OTP | One-Time Password |
| API | Application Programming Interface |
| CRUD | Create, Read, Update, Delete |
| ORM | Object-Relational Mapping |

Appendix B: Demo Accounts

| Role | Email | Password |
|----------|------------------------------|-------------|
| Customer | priya.sharma@email.com | password123 |
| Provider | ramesh.electrician@email.com | password123 |
| Admin | admin@hirelink.in | password123 |

Appendix C: API Endpoints Summary

| Method | Endpoint | Description |
|--------|---------------------------|-------------------|
| POST | /api/auth/register | User registration |
| POST | /api/auth/login | User login |
| GET | /api/categories | List categories |
| GET | /api/services/{id} | Service details |
| POST | /api/bookings | Create booking |
| GET | /api/bookings/my-bookings | User's bookings |
| PUT | /api/bookings/{id}/status | Update status |
| GET | /api/providers/{id} | Provider profile |
| POST | /api/bookings/{id}/review | Add review |

Document Version: 1.0 **Last Updated:** January 2026 **Academic Institution:** UIT, ADOOR

This document is submitted as part of the academic project requirements for the Graduate Academic Project at ENSATE, Academic Year 2025-2026.