TECHNICAL DOCUMENTATION

HireLink

System Architecture, Database Design & API Specifications

Technology Stack

React | Spring Boot | MySQL

Table of Contents

1. System Overview

1.1 Technology Stack

Layer	Technology	Version / Details
Frontend	React.js	v18.x with Hooks, Context API
UI Framework	Tailwind CSS / Material UI	Responsive components
Backend	Spring Boot (Java)	v3.x, Java 17+
Database	MySQL	v8.0 with InnoDB engine
ORM	Spring Data JPA / Hibernate	Entity management
Authentication	Spring Security + JWT	OAuth 2.0 support
API	RESTful APIs	JSON, OpenAPI 3.0 spec
Build Tools	Maven / npm	Dependency management
Cloud/Hosting	AWS / Heroku / Railway	EC2, RDS, S3

1.2 System Architecture

HireLink follows a three-tier architecture pattern separating presentation, business logic, and data layers. This ensures scalability, maintainability, and clear separation of concerns.

Architecture Diagram (Text Representation):

```
┌──────────────────────────────────────────────────────────────┐
│                    PRESENTATION LAYER                          │
│  ┌──────────────┐  ┌──────────────┐  ┌──────────────────┐  │
│  │   Web App    │  │  Mobile App  │  │ Admin Dashboard  │  │
│  │   (React)    │  │ (React PWA)  │  │    (React)       │  │
│  └──────────────┘  └──────────────┘  └──────────────────┘  │
└──────────────────────────────────────────────────────────────┘
                              │
                              ▼
┌──────────────────────────────────────────────────────────────┐
│                      API GATEWAY LAYER                         │
│             (Spring Boot REST Controllers)                     │
└──────────────────────────────────────────────────────────────┘
                              │
                              ▼
┌──────────────────────────────────────────────────────────────┐
│                    BUSINESS LOGIC LAYER                        │
│  ┌────────────┐  ┌────────────┐  ┌────────────┐  ┌──────────┐  │
│  │   User     │  │  Booking   │  │  Payment   │  │    AI    │  │
│  │  Service   │  │  Service   │  │  Service   │  │ Service  │  │
│  └────────────┘  └────────────┘  └────────────┘  └──────────┘  │
└──────────────────────────────────────────────────────────────┘
                              │
                              ▼
┌──────────────────────────────────────────────────────────────┐
│                      DATA ACCESS LAYER                         │
│  ┌───────────────────┐     ┌──────────────────────────────┐  │
│  │   MySQL (RDS)       │     │   File Storage (S3/Firebase) │    │
│  └─────────────────────┘     └──────────────────────────────┘    │
└──────────────────────────────────────────────────────────────────┘
```

2. Data Flow Diagrams (DFD)

2.1 Context Diagram (Level 0)

The context diagram shows HireLink as a single process with external entities:

```
              ┌─────────────┐
              │    User     │
              └──────┬──────┘
                     │ Search, Book, Pay, Review
                     ▼
┌───────────────────────────────────┐
│                                   │◄── Provider Profile, Accept
│          HIRELINK SYSTEM          │    ┌────────────────┐
│                                   │────│Service Provider│
└───────────────────────────────────┘    └────────────────┘
      │             │             │
      ▼             ▼             ▼
┌──────────┐  ┌─────────────┐  ┌─────────────┐
│ Payment  │  │  SMS/Email  │  │  Maps API   │
│ Gateway  │  │   Gateway   │  │  (Google)   │
```

2.2 Level 1 DFD - Main Processes

Key processes within HireLink:

Process	Input	Output	Data Store
P1: Registration	User details, OTP	Auth token, Profile	D1: Users
P2: Search	Location, Filters	Provider list	D2: Providers, D3: Services
P3: Booking	Service, Time, Address	Booking confirmation	D4: Bookings
P4: Payment	Amount, Method	Receipt, Status	D5: Payments
P5: Review	Rating, Comment	Updated provider rating	D6: Reviews
P6: AI Matching	User history, Preferences	Ranked providers	D7: ML Models

3. Database Design

3.1 Entity Relationship Diagram (Text)

```
┌────────────────────┐              ┌────────────────────┐
│        USER        │              │  SERVICE_PROVIDER  │
├────────────────────┤              ├────────────────────┤
│ PK: user_id        │◄────1:1────►│ PK: provider_id    │
│ name               │              │ FK: user_id        │
│ phone              │              │ business_name      │
│ email              │              │ bio                │
│ user_type          │              │ experience_years   │
└────────────────────┘              │ service_area       │
          │                         │ is_verified        │
          │ 1:M                     └────────────────────┘
          ▼                                   │ 1:M
┌────────────────────┐                        ▼
│      BOOKING       │◄──────────  ┌────────────────────┐
├────────────────────┤              │      SERVICE       │
│ PK: booking_id     │              ├────────────────────┤
│ FK: user_id        │              │ PK: service_id     │
│ FK: provider_id    │              │ FK: provider_id    │
│ FK: service_id     │              │ category           │
│ booking_date       │              │ name               │
│ status             │              │ description        │
└────────────────────┘              │ price              │
```

3.2 Database Schema (MySQL)

3.2.1 Users Table

```
CREATE TABLE users (
    user_id BIGINT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(100) NOT NULL,
    phone VARCHAR(15) UNIQUE NOT NULL,
    email VARCHAR(100) UNIQUE,
    password_hash VARCHAR(255) NOT NULL,
    user_type ENUM('USER', 'PROVIDER', 'ADMIN') DEFAULT 'USER',
    profile_image_url VARCHAR(500),
    is_verified BOOLEAN DEFAULT FALSE,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP
);
```

3.2.2 Service Providers Table

```
CREATE TABLE service_providers (
    provider_id BIGINT PRIMARY KEY AUTO_INCREMENT,
    user_id BIGINT UNIQUE NOT NULL,
    business_name VARCHAR(150),
    bio TEXT,
    experience_years INT DEFAULT 0,
    latitude DECIMAL(10, 8),
    longitude DECIMAL(11, 8),
    service_radius_km INT DEFAULT 10,
    aadhaar_number_encrypted VARCHAR(255),
    is_kyc_verified BOOLEAN DEFAULT FALSE,
    avg_rating DECIMAL(2,1) DEFAULT 0.0,
    total_reviews INT DEFAULT 0,
    is_available BOOLEAN DEFAULT TRUE,
    FOREIGN KEY (user_id) REFERENCES users(user_id)
);
```

3.2.3 Services Table

```
CREATE TABLE services (
    service_id BIGINT PRIMARY KEY AUTO_INCREMENT,
    provider_id BIGINT NOT NULL,
    category
ENUM('PLUMBING','ELECTRICAL','CARPENTRY','MASON','PAINTING','AC_APPLIANCE',
'OTHER'),
    name VARCHAR(150) NOT NULL,
    description TEXT,
    base_price DECIMAL(10,2) NOT NULL,
    price_type ENUM('FIXED', 'HOURLY', 'PER_SQFT') DEFAULT 'FIXED',
    is_active BOOLEAN DEFAULT TRUE,
    FOREIGN KEY (provider_id) REFERENCES service_providers(provider_id)
);
```

3.2.4 Bookings Table

```
CREATE TABLE bookings (
    booking_id BIGINT PRIMARY KEY AUTO_INCREMENT,
    user_id BIGINT NOT NULL,
    provider_id BIGINT NOT NULL,
    service_id BIGINT NOT NULL,
    scheduled_date DATE NOT NULL,
    scheduled_time TIME NOT NULL,
    address TEXT NOT NULL,
    pincode VARCHAR(10),
    issue_description TEXT,
    status
ENUM('PENDING','CONFIRMED','IN_PROGRESS','COMPLETED','CANCELLED') DEFAULT
'PENDING',
    final_amount DECIMAL(10,2),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (user_id) REFERENCES users(user_id),
    FOREIGN KEY (provider_id) REFERENCES service_providers(provider_id),
    FOREIGN KEY (service_id) REFERENCES services(service_id)
);
```

4. API Specifications

4.1 Authentication APIs

Method	Endpoint	Description
POST	/api/v1/auth/register	Register new user with phone/email
POST	/api/v1/auth/send-otp	Send OTP to phone number
POST	/api/v1/auth/verify-otp	Verify OTP and return JWT token
POST	/api/v1/auth/login	Login with credentials
POST	/api/v1/auth/refresh-token	Refresh JWT token

4.2 Provider APIs

Method	Endpoint	Description
GET	/api/v1/providers	Search providers with filters
GET	/api/v1/providers/{id}	Get provider details by ID
POST	/api/v1/providers/register	Register as service provider
PUT	/api/v1/providers/{id}	Update provider profile
GET	/api/v1/providers/nearby	Get nearby providers (lat, lng, radius)

4.3 Booking APIs

Method	Endpoint	Description
POST	/api/v1/bookings	Create new booking request
GET	/api/v1/bookings/{id}	Get booking details
PATCH	/api/v1/bookings/{id}/status	Update booking status
GET	/api/v1/bookings/user/{userId}	Get user's booking history

5. Project Structure

5.1 Backend (Spring Boot)

```
hirelink-backend/
├── src/main/java/com/hirelink/
│   ├── HirelinkApplication.java
│   ├── config/
│   │   ├── SecurityConfig.java
│   │   ├── JwtConfig.java
│   │   └── CorsConfig.java
│   ├── controller/
│   │   ├── AuthController.java
│   │   ├── UserController.java
│   │   ├── ProviderController.java
│   │   └── BookingController.java
│   ├── service/
│   ├── repository/
│   ├── entity/
│   ├── dto/
│   └── exception/
├── src/main/resources/
│   ├── application.properties
│   └── application-dev.properties
└── pom.xml
```

5.2 Frontend (React)

```
hirelink-frontend/
├── src/
│   ├── components/
│   │   ├── common/
│   │   ├── auth/
│   │   ├── provider/
│   │   └── booking/
│   ├── pages/
│   ├── hooks/
│   ├── context/
│   ├── services/
│   ├── utils/
│   ├── App.jsx
│   └── index.js
├── public/
└── package.json
```

— End of Document —