CEBU INSTITUTE OF TECHNOLOGY
UNIVERSITY

IT342-G4
SYSTEMS INTEGRATION AND ARCHITECTURE 1

FUNCTIONAL REQUIREMENTS SPECIFICATION (FRS)

Project Title: InkSlot: Tattoo Shop Booking System

Prepared By: Natasha Kate A. Leonardo

Date of Submission: February 3, 2026

Version: 0.1

Table of Contents

1. Introduction....3
1.1. Purpose....3
1.2. Scope....3
1.3. Definitions, Acronyms, and Abbreviations....4
2. Overall Description....5
2.1. System Perspective....5
2.2. User Classes and Characteristics....5
2.3. Operating Environment....6
2.4. Assumptions and Dependencies....7
3. System Features and Functional Requirements....8
3.1. Feature 1: User Registration and Authentication....8
3.2. Feature 2: Artist Profile Management....10
3.3. Feature 3: Booking Management System....11
3.4. Feature 4: Schedule Management....13
3.4. Feature 5: Guest Booking Interface....14
4. Non-Functional Requirements....15
4.1. Performance Requirements....15
4.2. Security Requirements....15
4.3. Usability Requirements....16
4.4. Reliability Requirements....17
4.5. Maintainability Requirements....17
4.6. Portability Requirements....18
4.7. Scalability Requirements....18
4.8. Compatibility Requirements....19
5. System Models (Diagrams)....20
5.1. ERD....20
5.2. Use Case Diagram....21
5.3. Activity Diagram....22
5.4. Class Diagram....23
5.5. Sequence Diagram....24
6. Appendices....25

1. Introduction

1.1. Purpose

This Software Requirements Specification (SRS) document provides a comprehensive description of the InkSlot tattoo shop booking system. The document details the functional and non-functional requirements, system architecture, and design specifications for the application.

Intended Audience:

- Software developers and engineers implementing the system
- Project managers overseeing development
- Quality assurance and testing teams
- System administrators responsible for deployment
- Stakeholders evaluating system capabilities
- Academic instructors and evaluators (for CSIT327 laboratory activity)

1.2. Scope

System Name: InkSlot - Tattoo Shop Booking System

What the system will do:

- Provide a streamlined online booking platform for tattoo shop appointments
- Enable guest users to browse artist profiles, portfolios, and available time slots
- Allow guests to book appointments without mandatory account creation
- Facilitate artist registration and authentication for managing their professional presence
- Enable artists to showcase their work through portfolio uploads
- Allow artists to set and manage their availability schedules
- Display shop pricing rates based on tattoo size categories
- Handle booking confirmations via email and phone notifications
- Support design upload functionality for clients to share tattoo concepts
- Track appointment statuses (pending, confirmed, completed, cancelled)

What the system will NOT do:

- Process payments or handle financial transactions (future enhancement)
- Provide AI-based tattoo design generation or pricing (future enhancement)
- Support video consultations or real-time chat
- Manage inventory or tattoo supplies
- Handle multi-location shop management
- Provide customer loyalty programs or membership systems

System Boundaries:

- Frontend: React-based web application

- Backend: Spring Boot REST API
- Database: Relational database (PostgreSQL/MySQL)
- Deployment: Cloud hosting (Render, Railway, or Vercel)
- Version Control: Git with GitHub repository
- User access: Web browsers only (no mobile native apps in current scope)
- Authentication scope: Artists and admin only; guests access without authentication

1.3. Definitions, Acronyms, and Abbreviations

Term	Definition
API	Application Programming Interface - interface for communication between frontend and backend
Artist	Tattoo artist employed by the shop who can register, manage profile, and view bookings
Booking	An appointment reservation made by a guest for a tattoo session
CRUD	Create, Read, Update, Delete - basic database operations
ERD	Entity Relationship Diagram - visual representation of database schema
Guest	Unauthenticated user who can browse and create bookings without registration
JWT	JSON Web Token - authentication token format used for securing API endpoints
Portfolio	Collection of images showcasing an artist's previous tattoo work
REST	Representational State Transfer - architectural style for web services
SPA	Single Page Application - web application that loads a single HTML page
Spring Boot	Java-based framework for building backend applications
SRS	Software Requirements Specification - this document
UML	Unified Modeling Language - standardized modeling notation
UI	User Interface - visual elements users interact with
UX	User Experience - overall experience of a person using the system

2. Overall Description

2.1. System Perspective

InkSlot is a standalone web-based booking system designed specifically for tattoo shop appointment management. The system operates as a full-stack application with clear separation between presentation, business logic, and data layers.

System Architecture:

- Frontend Layer: React-based Single Page Application (SPA) providing responsive user interface
- Backend Layer: Spring Boot REST API handling business logic, authentication, and data processing
- Data Layer: Relational database storing user accounts, artist profiles, bookings, and schedules
- Communication: RESTful HTTP requests with JSON data format and JWT-based authentication

External Interfaces:

- Email service integration for booking confirmations (SMTP server)
- SMS service for phone notifications (future integration with Twilio/similar)
- File storage system for image uploads (portfolio images, tattoo designs)

System Context:
InkSlot serves as a customer-facing scheduling tool for tattoo shops, bridging the gap between artists and potential clients. It replaces traditional phone-based or walk-in booking systems with a modern, accessible online platform available 24/7.

2.2. User Classes and Characteristics

Guest User (Client/Customer)

- Characteristics: General public interested in getting tattoos, varying technical proficiency
- Frequency of Use: Occasional (typically booking once every few months to years)
- Technical Expertise: Low to moderate; expects intuitive interface
- Primary Goals: Browse artist profiles, view portfolios, check availability, book appointments
- Privileges: Read-only access to public content; can create bookings without authentication
- Special Needs: Mobile-responsive design; clear pricing information; easy booking process

Authenticated Artist

- Characteristics: Professional tattoo artists employed by the shop, moderate technical proficiency
- Frequency of Use: Daily to weekly for managing profiles, schedules, and viewing bookings
- Technical Expertise: Moderate; requires user-friendly dashboard
- Primary Goals: Showcase work, manage availability, track appointments, update profile information
- Privileges: Full CRUD access to own profile, portfolio, and schedule; read access to bookings
- Special Needs: Image upload capability; calendar-based schedule management; booking notifications

Admin/Shop Owner *(Future Enhancement)*

- Characteristics: Shop owner or manager with administrative oversight
- Frequency of Use: Regular monitoring and management
- Technical Expertise: Moderate to high
- Primary Goals: Manage all artists, oversee bookings, update shop rates, generate reports
- Privileges: Full system access including artist account management
- Special Needs: Analytics dashboard; bulk operations; user management tools

2.3. Operating Environment

Client-Side Requirements:

- Hardware: Desktop, laptop, tablet, or smartphone with minimum 2GB RAM
- Operating Systems: Windows 10+, macOS 10.14+, Linux (Ubuntu 18.04+), iOS 12+, Android 8.0+
- Web Browsers:
    - Google Chrome (version 90+)
    - Mozilla Firefox (version 88+)
    - Safari (version 14+)
    - Microsoft Edge (version 90+)
- Network: Stable internet connection (minimum 1 Mbps recommended)
- Display: Minimum resolution 320x568 (mobile), recommended 1366x768 (desktop)

Server-Side Requirements:

- Operating System: Linux (Ubuntu 20.04 LTS) or Windows Server 2019+
- Runtime Environment: Java Development Kit (JDK) 17 or higher
- Application Server: Embedded Tomcat (included with Spring Boot)
- Database Server: PostgreSQL 13+ or MySQL 8.0+
- Memory: Minimum 4GB RAM (8GB recommended for production)

- Storage: Minimum 20GB disk space (scalable based on image uploads)
- Network: Static IP address with proper firewall configuration

Development Tools:

- IDE: Visual Studio Code
- Version Control: Git with GitHub
- Build Tools: Spring Boot (Java) for backend; ReactJS for frontend; Android Kotlin for mobile
- Deployment: Cloud hosting (Render)

Third-Party Services:

- Email Service: SMTP server (Gmail SMTP, SendGrid, or AWS SES)
- File Storage: Local file system or cloud storage (AWS S3, Cloudinary - future)
- SMS Service: Twilio or similar (future integration)

2.4. Assumptions and Dependencies

Assumptions:

1. Users have basic familiarity with web-based booking systems
2. Artists have professional tattoo work to showcase in portfolios
3. The tattoo shop has consistent pricing structure across all artists
4. Guests provide valid email addresses and phone numbers for contact
5. Internet connectivity is available to all users during system access
6. The shop operates during standard business hours with predefined time slots
7. Booking confirmations occur within 24-48 hours of submission
8. Artists are responsible for keeping their schedules up-to-date
9. The system operates in a single timezone (shop's local timezone)
10. Image uploads are limited to reasonable file sizes (e.g., 5MB per image)

Dependencies:

1. Java Runtime Environment: System requires JDK 17+ for Spring Boot execution
2. Database Server: Functional PostgreSQL or MySQL instance must be available
3. Email Service: SMTP server must be accessible for sending booking confirmations
4. Browser Compatibility: Users must have modern web browsers with JavaScript enabled
5. Network Infrastructure: Reliable hosting environment with adequate bandwidth
6. SSL Certificate: HTTPS required for production deployment (security requirement)
7. Domain Name: Registered domain for production deployment

8. File System Access: Write permissions required for storing uploaded images
9. Spring Boot Framework: Compatible versions of Spring dependencies
10. React Framework: Node.js for frontend build process
11. CORS Configuration: Proper cross-origin settings for frontend-backend communication
12. JWT Library: Authentication depends on Java JWT implementation (jjwt or similar)
13. Cloud Hosting Platform: Render, Railway, or Vercel for production deployment
14. Git/GitHub: Version control system for collaborative development and deployment
15. Android Kotlin: For future mobile application development (Phase 2 enhancement)

External Dependencies:

- Spring Security for authentication
- Spring Data JPA for database operations
- React Router for frontend navigation
- Axios or Fetch API for HTTP requests
- BCrypt for password hashing

3. System Features and Functional Requirements

3.1. Feature 1: User Registration and Authentication

Description:
The system provides secure registration and authentication mechanisms for artists to create accounts and access protected resources. This includes user account creation with email verification, login with JWT token generation, and logout functionality. The authentication system ensures that only authorized artists can manage their profiles, portfolios, and schedules.

Functional Requirements:

FR-1.1: Artist Registration

- The system shall allow new artists to register by providing email, password, full name, and phone number
- The system shall validate that the email address is in proper format and not already registered
- The system shall enforce password requirements (minimum 8 characters, at least one uppercase, one lowercase, one number)
- The system shall hash passwords using BCrypt before storing in the database

- The system shall automatically assign the role "ARTIST" to newly registered users
- The system shall create a corresponding artist profile record upon successful user registration
- The system shall display appropriate error messages for validation failures or registration errors
- The system shall redirect users to the login page after successful registration

FR-1.2: Artist Login

- The system shall provide a login form accepting email and password
- The system shall validate credentials against stored user records
- The system shall generate a JWT token upon successful authentication
- The system shall include user role information in the JWT token payload
- The system shall return the token and user information to the client
- The system shall reject login attempts with incorrect credentials with appropriate error messages
- The system shall implement rate limiting to prevent brute force attacks (maximum 5 attempts per 15 minutes)
- The system shall redirect authenticated artists to their dashboard upon successful login

FR-1.3: Artist Logout

- The system shall provide a logout function accessible from any authenticated page
- The system shall clear authentication tokens from client-side storage upon logout
- The system shall invalidate the current session (if session-based authentication is used)
- The system shall redirect users to the home page after logout
- The system shall prevent access to protected pages after logout

FR-1.4: Token Management

- The system shall set JWT token expiration to 24 hours
- The system shall validate tokens on every request to protected endpoints
- The system shall return 401 Unauthorized for invalid or expired tokens
- The system shall extract user information from valid tokens for request processing

FR-1.5: Access Control

- The system shall protect artist-specific endpoints from unauthenticated access
- The system shall verify user roles before granting access to restricted features
- The system shall allow guests to access public pages without authentication

- The system shall implement proper authorization checks for all CRUD operations

FR-1.6: OAuth Authentication (Optional)

- The system shall support OAuth 2.0 authentication via Google
- The system shall allow artists to register/login using their Google account
- The system shall retrieve user email and name from OAuth provider
- The system shall create user account automatically upon first OAuth login
- The system shall link OAuth accounts to existing email-based accounts if email matches

3.2. Feature 2: Artist Profile Management

Description:
Artists can create and maintain comprehensive professional profiles showcasing their expertise, experience, and personal information. This feature enables artists to present themselves to potential clients through biographical information, years of experience, specialties, and profile images.

Functional Requirements:

FR-2.1: View Artist Profile

- The system shall allow authenticated artists to view their complete profile information
- The system shall display profile data including bio, years of experience, specialties, and profile image
- The system shall provide a dedicated profile page in the artist dashboard
- The system shall allow guests to view public-facing artist profiles without authentication

FR-2.2: Edit Profile Information

- The system shall provide an edit form for artists to update their profile information
- The system shall allow artists to modify bio (text field up to 1000 characters)
- The system shall allow artists to update years of experience (integer value 0-50)
- The system shall allow artists to edit specialties (text field up to 500 characters)
- The system shall validate all input fields before submission
- The system shall save changes to the database upon successful validation
- The system shall display success confirmation after profile update
- The system shall show error messages for validation failures

FR-2.3: Profile Image Upload

- The system shall allow artists to upload a profile photo (JPG, PNG formats)

- The system shall restrict image file size to maximum 5MB
- The system shall validate image format and dimensions (minimum 200x200 pixels)
- The system shall store uploaded images in the server file system or cloud storage
- The system shall save the image URL/path in the database
- The system shall display the uploaded image in the profile preview
- The system shall allow artists to replace existing profile images
- The system shall delete old images when new ones are uploaded

FR-2.4: Profile Display for Guests

- The system shall display all artist profiles on a public browsing page
- The system shall show profile image, name, bio, years of experience, and specialties
- The system shall provide a link to view individual artist's portfolio
- The system shall display artist availability status (currently available/unavailable)

FR-2.4: OAuth Security

- The system shall use OAuth 2.0 protocol for third-party authentication
- The system shall not store OAuth provider passwords
- The system shall securely store and refresh OAuth tokens
- The system shall validate OAuth tokens with provider on each request

3.3. Feature 3: Booking Management System

Description:
A comprehensive booking system that allows guests to create appointment reservations and enables artists to view and manage their bookings. The system handles booking creation, status tracking, and notification workflows.

Functional Requirements:

FR-3.1: Create Booking (Guest)

- The system shall provide a booking form accessible to guests without authentication
- The system shall require guests to input: client name, email, phone number
- The system shall require guests to select: preferred artist, appointment date, appointment time
- The system shall allow optional design image upload (JPG, PNG, max 10MB)
- The system shall allow guests to specify approximate tattoo size (text input or dropdown)
- The system shall validate all required fields before submission

- The system shall check artist availability for selected date/time
- The system shall calculate estimated price based on size category and shop rates
- The system shall create booking record with status "PENDING"
- The system shall send confirmation email to guest with booking details
- The system shall display booking confirmation page with appointment information and booking reference number

FR-3.2: View Bookings (Artist)

- The system shall provide a bookings dashboard for authenticated artists
- The system shall display all bookings assigned to the logged-in artist
- The system shall show booking details: client info, date, time, design image, size, estimated price, status
- The system shall allow filtering bookings by status (pending, confirmed, completed, cancelled)
- The system shall allow sorting bookings by date (ascending/descending)
- The system shall display bookings in a table or card layout with pagination
- The system shall show booking count for each status category

FR-3.3: Update Booking Status (Artist)

- The system shall allow artists to change booking status from "PENDING" to "CONFIRMED"
- The system shall allow artists to mark bookings as "COMPLETED" after the appointment
- The system shall allow artists to "CANCEL" bookings with optional cancellation reason
- The system shall send email notification to client when status changes
- The system shall log status change history with timestamp and user
- The system shall prevent modification of completed or cancelled bookings

FR-3.4: View Design Upload

- The system shall display uploaded design images in the booking details view
- The system shall provide image zoom/preview functionality
- The system shall allow artists to download design images for reference

FR-3.5: Booking Notifications

- The system shall send email confirmation to guest immediately after booking creation
- The system shall send email notification to assigned artist when new booking is created
- The system shall send status update emails when artists modify booking status
- The system shall include booking details in all notification emails

3.4. Feature 4: Schedule Management

Description:
Artists can define and manage their availability schedules, allowing guests to view real-time availability when booking appointments. This prevents double-booking and ensures accurate scheduling.

Functional Requirements:

FR-4.1: Set Availability

- The system shall provide a schedule management interface for authenticated artists
- The system shall allow artists to add available time slots by selecting date, start time, and end time
- The system shall validate that start time is before end time
- The system shall prevent overlapping time slots for the same artist
- The system shall allow artists to mark time slots as available or unavailable
- The system shall save schedule entries to the database
- The system shall display success confirmation after schedule update

FR-4.2: View Schedule

- The system shall display artist's schedule in calendar view format
- The system shall show available and unavailable time slots with visual distinction (colors/icons)
- The system shall allow artists to view schedule by day, week, or month
- The system shall highlight dates with existing bookings
- The system shall show booked time slots as unavailable

FR-4.3: Modify Schedule

- The system shall allow artists to edit existing schedule entries
- The system shall allow artists to delete future schedule entries
- The system shall prevent deletion of past schedule entries
- The system shall prevent modification of time slots with confirmed bookings
- The system shall warn artists if modifying schedule affects pending bookings

FR-4.4: Check Availability (Guest)

- The system shall display available time slots for selected artist and date
- The system shall only show time slots marked as available with no confirmed bookings
- The system shall refresh availability in real-time when date or artist selection changes
- The system shall indicate unavailable dates with visual styling
- The system shall show earliest available appointment date for each artist

3.5. Feature 5: Guest Booking Interface

Description:
A public-facing interface allowing guests to browse artist information, view portfolios, check pricing, and create bookings without requiring authentication.

Functional Requirements:

FR-5.1: Browse Artists

- The system shall display a list of all active artists on the public homepage
- The system shall show artist profile images, names, and brief bio excerpts
- The system shall indicate each artist's current availability status
- The system shall provide a "View Portfolio" link for each artist
- The system shall allow sorting artists by name or years of experience

FR-5.2: View Artist Portfolio

- The system shall display all portfolio images uploaded by the selected artist
- The system shall show images in grid layout with image descriptions
- The system shall provide image lightbox/modal for full-size viewing
- The system shall show portfolio metadata (upload date, description)
- The system shall allow guests to navigate between different artists' portfolios

FR-5.3: View Shop Rates

- The system shall display pricing information on a dedicated rates page
- The system shall show all size categories (small, medium, large) with base prices
- The system shall provide detailed descriptions of what each size category includes
- The system shall display disclaimer that final pricing may vary based on complexity
- The system shall allow access to rates page without authentication

FR-5.4: Upload Portfolio (Artist)

- The system shall allow authenticated artists to upload portfolio images
- The system shall accept JPG, PNG image formats with maximum 5MB file size
- The system shall require optional text description for each portfolio image (up to 500 characters)
- The system shall validate image format and size before upload
- The system shall store images with associated artist_id in database
- The system shall allow artists to upload multiple images in batch
- The system shall display uploaded images immediately in portfolio view

FR-5.5: Manage Portfolio (Artist)

- The system shall allow artists to view all their portfolio images in dashboard

- The system shall allow artists to edit descriptions of existing portfolio images
- The system shall allow artists to delete portfolio images
- The system shall confirm deletion action before removing images
- The system shall remove image files from storage when deleted from database

4. Non-Functional Requirements

4.1. Performance Requirements

NFR-1.1: Response Time

- The system shall load the homepage within 2 seconds on standard broadband connection (10 Mbps)
- The system shall return API responses within 500 milliseconds for 95% of requests
- The system shall load artist profile pages within 1.5 seconds
- The system shall complete booking creation within 3 seconds including database operations and email sending

NFR-1.2: Throughput

- The system shall support at least 100 concurrent users without performance degradation
- The system shall handle minimum 50 booking requests per hour
- The system shall process at least 20 image uploads simultaneously

NFR-1.3: Resource Usage

- The system shall utilize maximum 80% CPU under normal load conditions
- The system shall maintain memory usage below 4GB under normal operations
- The system shall optimize database queries to execute within 100 milliseconds

4.2 Security Requirements

NFR-2.1: Authentication Security

- The system shall implement JWT-based authentication with secure token generation
- The system shall hash all passwords using BCrypt with minimum cost factor of 10
- The system shall enforce password complexity requirements (minimum 8 characters, mixed case, numbers)

- The system shall implement rate limiting on login attempts (5 attempts per 15 minutes)
- The system shall expire JWT tokens after 24 hours of inactivity

NFR-2.2: Data Protection

- The system shall use HTTPS/TLS 1.3 for all client-server communications in production
- The system shall validate and sanitize all user inputs to prevent SQL injection attacks
- The system shall implement Cross-Site Scripting (XSS) protection on all input fields
- The system shall protect against Cross-Site Request Forgery (CSRF) attacks
- The system shall not store sensitive information in browser localStorage (except auth tokens)

NFR-2.3: Authorization

- The system shall implement role-based access control (RBAC)
- The system shall verify user roles before granting access to protected endpoints
- The system shall ensure artists can only modify their own profiles and portfolios
- The system shall prevent unauthorized access to other users' data

NFR-2.4: File Upload Security

- The system shall validate file types and reject executable files
- The system shall scan uploaded images for malicious content
- The system shall enforce file size limits (5MB profile images, 10MB design uploads)
- The system shall generate unique filenames to prevent file overwriting

4.3 Usability Requirements

NFR-3.1: User Interface

- The system shall provide an intuitive and consistent user interface following Material Design or Bootstrap principles
- The system shall clearly label all form fields with appropriate placeholders
- The system shall display error messages in red text near the relevant input field
- The system shall provide visual feedback for loading states (spinners, progress bars)
- The system shall highlight required form fields with asterisks (*)

NFR-3.2: Accessibility

- The system shall comply with WCAG 2.1 Level AA accessibility standards

- The system shall provide alt text for all images
- The system shall support keyboard navigation for all interactive elements
- The system shall maintain color contrast ratio of at least 4.5:1 for text
- The system shall be compatible with screen readers

NFR-3.3: Learnability

- The system shall provide tooltips or help text for complex features
- The system shall require minimal training for guest users to book appointments
- The system shall provide clear navigation with breadcrumbs for multi-step processes

4.4 Reliability Requirements

NFR-4.1: Availability

- The system shall maintain 99% uptime during business hours (9 AM - 9 PM local time)
- The system shall have maximum planned downtime of 2 hours per month for maintenance
- The system shall gracefully handle server errors with user-friendly error pages

NFR-4.2: Fault Tolerance

- The system shall continue operating if email service is temporarily unavailable (queue emails)
- The system shall log all errors to a persistent log file for troubleshooting
- The system shall recover from database connection failures with automatic retry mechanisms

NFR-4.3: Data Integrity

- The system shall use database transactions to ensure data consistency
- The system shall prevent duplicate bookings for the same artist at the same time
- The system shall maintain referential integrity through foreign key constraints
- The system shall backup database daily and retain backups for 30 days

4.5 Maintainability Requirements

NFR-5.1: Code Quality

- The system shall follow Java coding conventions (Oracle Code Conventions)
- The system shall maintain code documentation with Javadoc comments for all public methods
- The system shall achieve minimum 70% code coverage with unit tests

- The system shall use meaningful variable and method names following camelCase convention

NFR-5.2: Modularity

- The system shall implement layered architecture (Controller, Service, Repository)
- The system shall separate concerns between frontend and backend
- The system shall use dependency injection for loose coupling
- The system shall create reusable components for common UI elements

NFR-5.3: Logging

- The system shall log all authentication attempts (successful and failed)
- The system shall log all booking creation, modification, and cancellation events
- The system shall use appropriate log levels (DEBUG, INFO, WARN, ERROR)
- The system shall rotate log files when they reach 100MB size

4.6 Portability Requirements

NFR-6.1: Platform Independence

- The system shall run on any operating system supporting Java 17+ (Windows, Linux, macOS)
- The system shall be deployable to cloud platforms (AWS, Azure, Google Cloud)
- The system shall support containerization using Docker

NFR-6.2: Database Portability

- The system shall support both PostgreSQL and MySQL with minimal configuration changes
- The system shall use standard SQL queries avoiding database-specific syntax
- The system shall abstract database operations using Spring Data JPA

4.7 Scalability Requirements

NFR-7.1: Horizontal Scalability

- The system shall support load balancing across multiple backend instances
- The system shall use stateless authentication (JWT) to enable horizontal scaling
- The system shall externalize file storage to support distributed deployments

NFR-7.2: Data Scalability

- The system shall handle up to 10,000 user accounts without performance degradation
- The system shall support up to 100,000 booking records efficiently

- The system shall implement pagination for large data sets (50 records per page)

4.8 Compatibility Requirements

NFR-8.1: Browser Compatibility

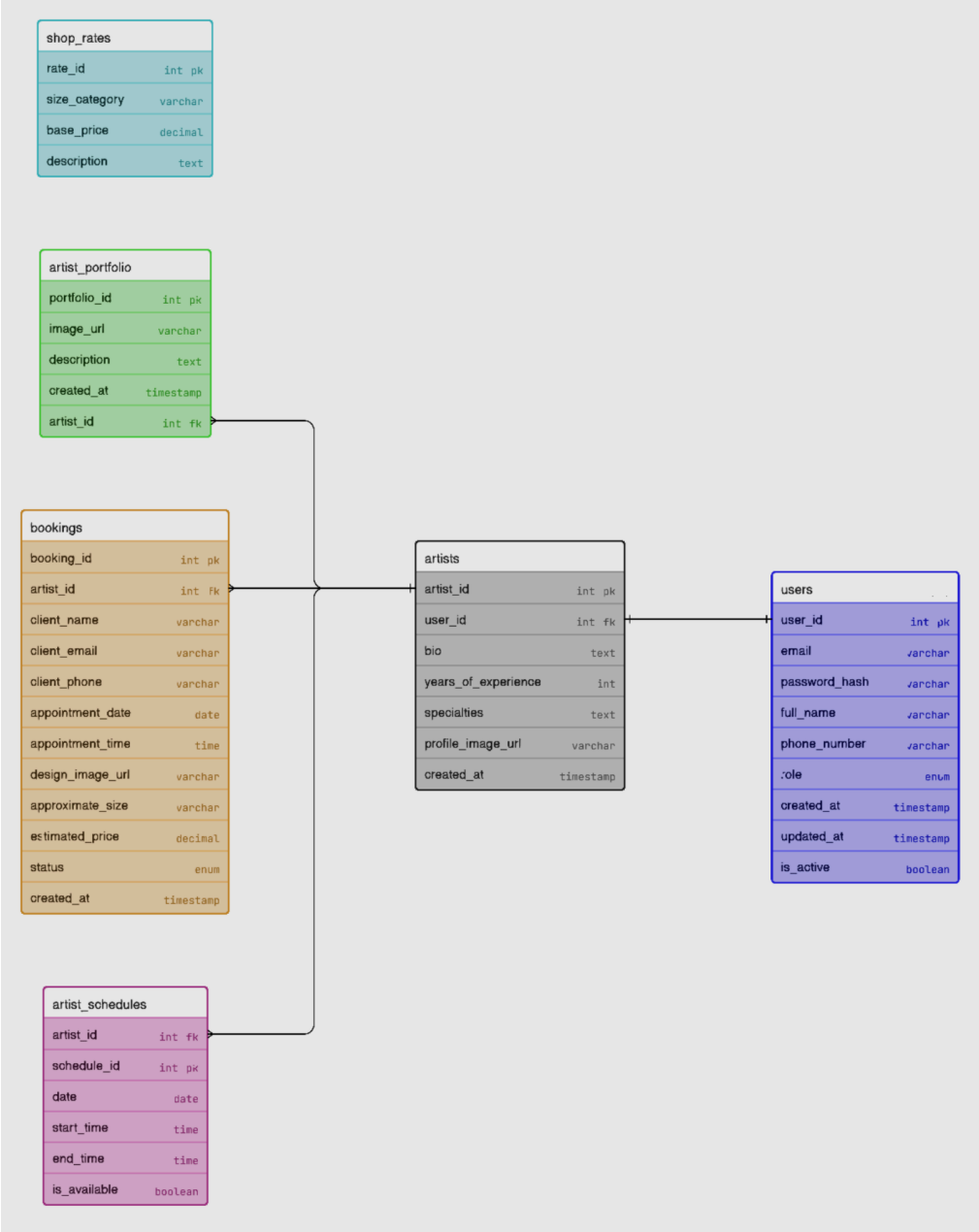
- The system shall function correctly on Chrome 90+, Firefox 88+, Safari 14+, Edge 90+
- The system shall degrade gracefully on older browsers with appropriate warnings
- The system shall support mobile browsers (Chrome Mobile, Safari Mobile)

NFR-8.2: Responsive Design

- The system shall adapt to screen sizes from 320px (mobile) to 2560px (desktop)
- The system shall provide touch-friendly controls on mobile devices (minimum 44x44px tap targets)
- The system shall maintain usability across portrait and landscape orientations

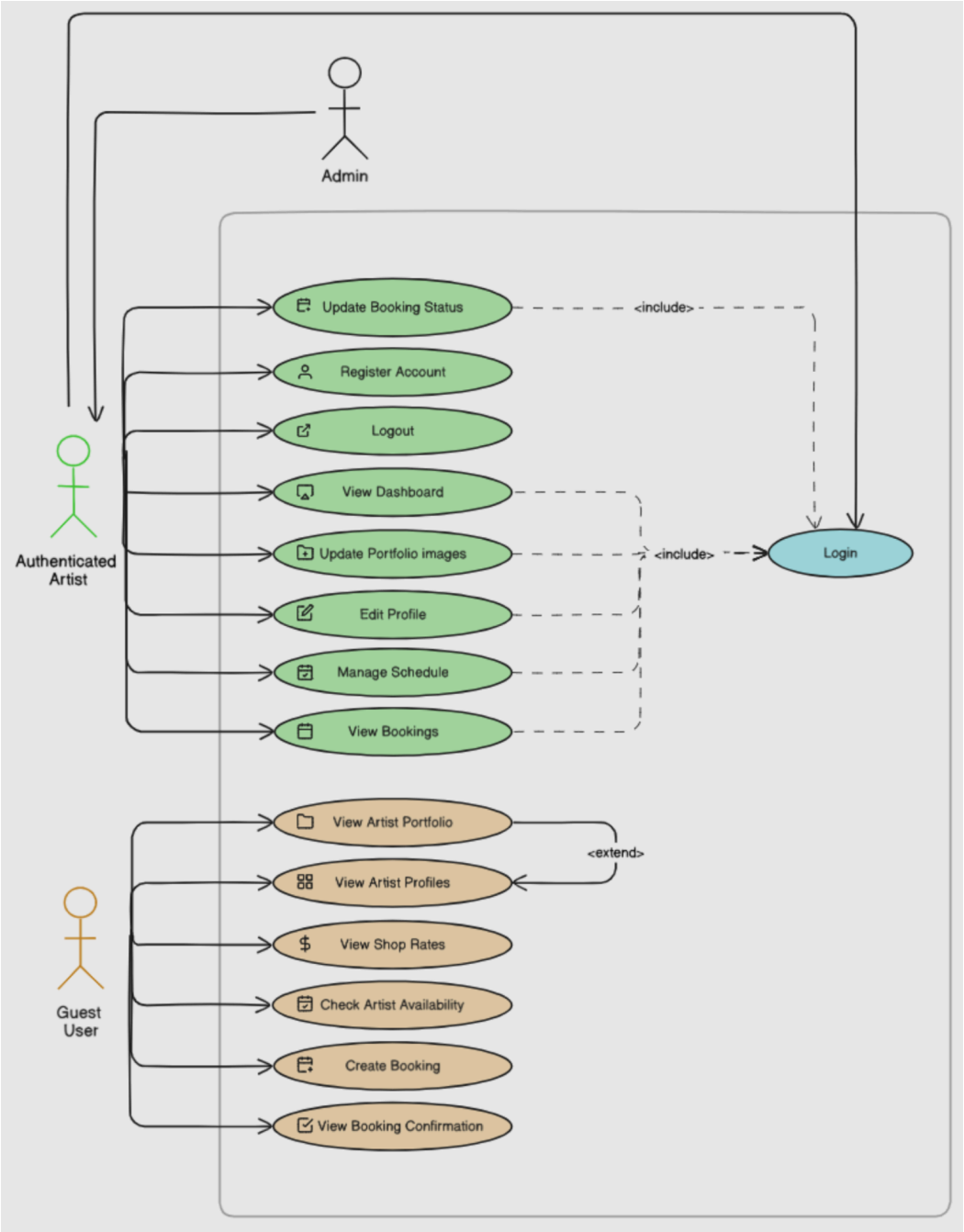
5. System Models (Diagrams)

5.1. ERD



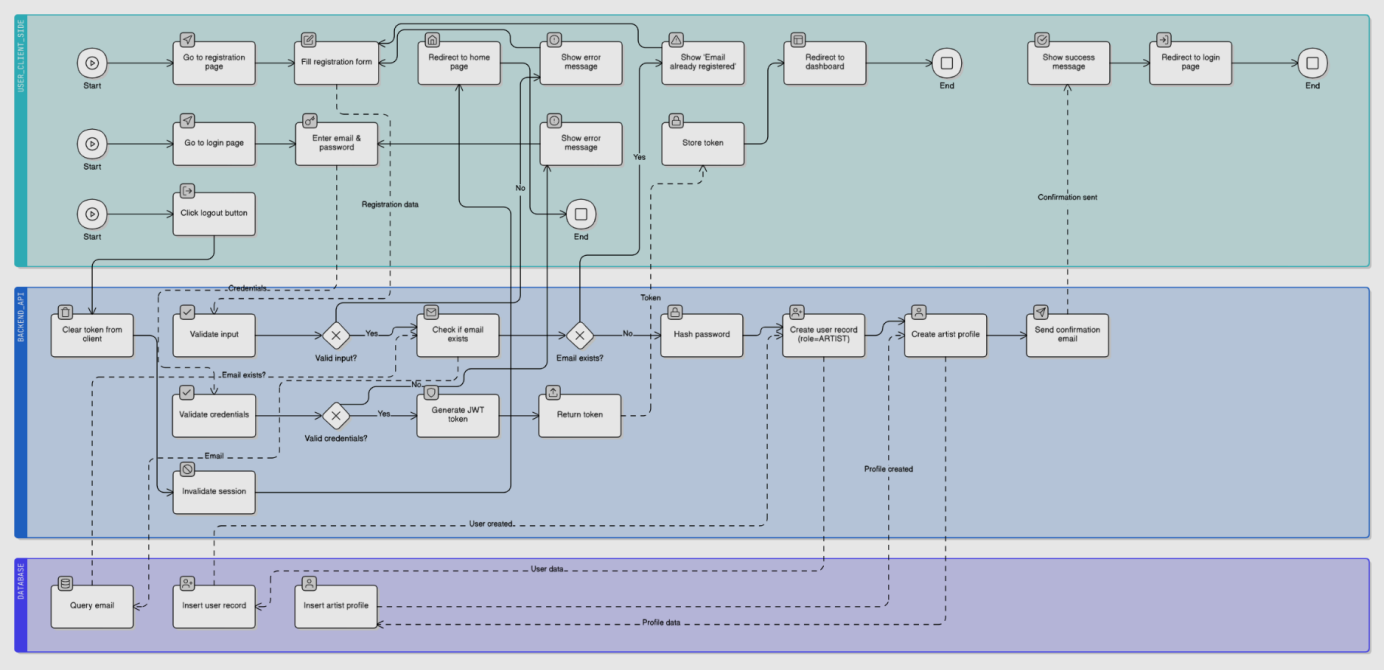
The Entity Relationship Diagram illustrates the database schema for InkSlot, showing six main entities: users, artists, artist_portfolio, bookings, artist_schedules, and shop_rates. The diagram depicts the relationships between entities using crow's foot notation, highlighting primary keys (PK), foreign keys (FK), and cardinality constraints. Key relationships include the one-to-one association between users and artists, and one-to-many relationships from artists to their portfolios, bookings, and schedules.

5.2. Use Case Diagram



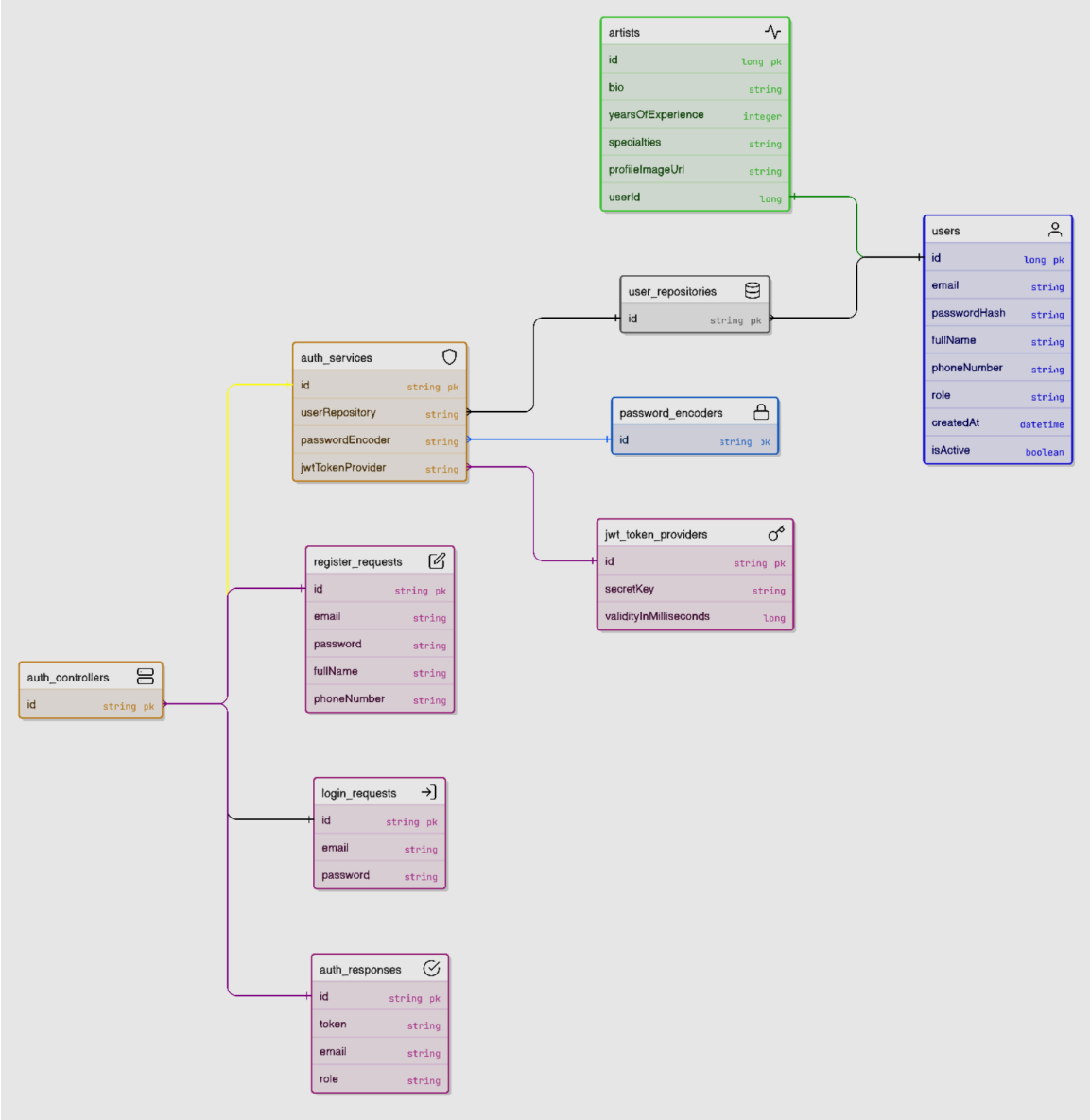
The Use Case Diagram identifies the primary actors (Guest User, Authenticated Artist, Admin) and their interactions with the InkSlot system. It illustrates functional requirements through use cases such as browsing artist profiles, creating bookings, managing portfolios, and handling authentication. The diagram employs include and extend relationships to show dependencies, notably that authentication (Login) is a precondition for all artist-specific actions, and viewing portfolios extends from viewing artist profiles.

5.3. Activity Diagram



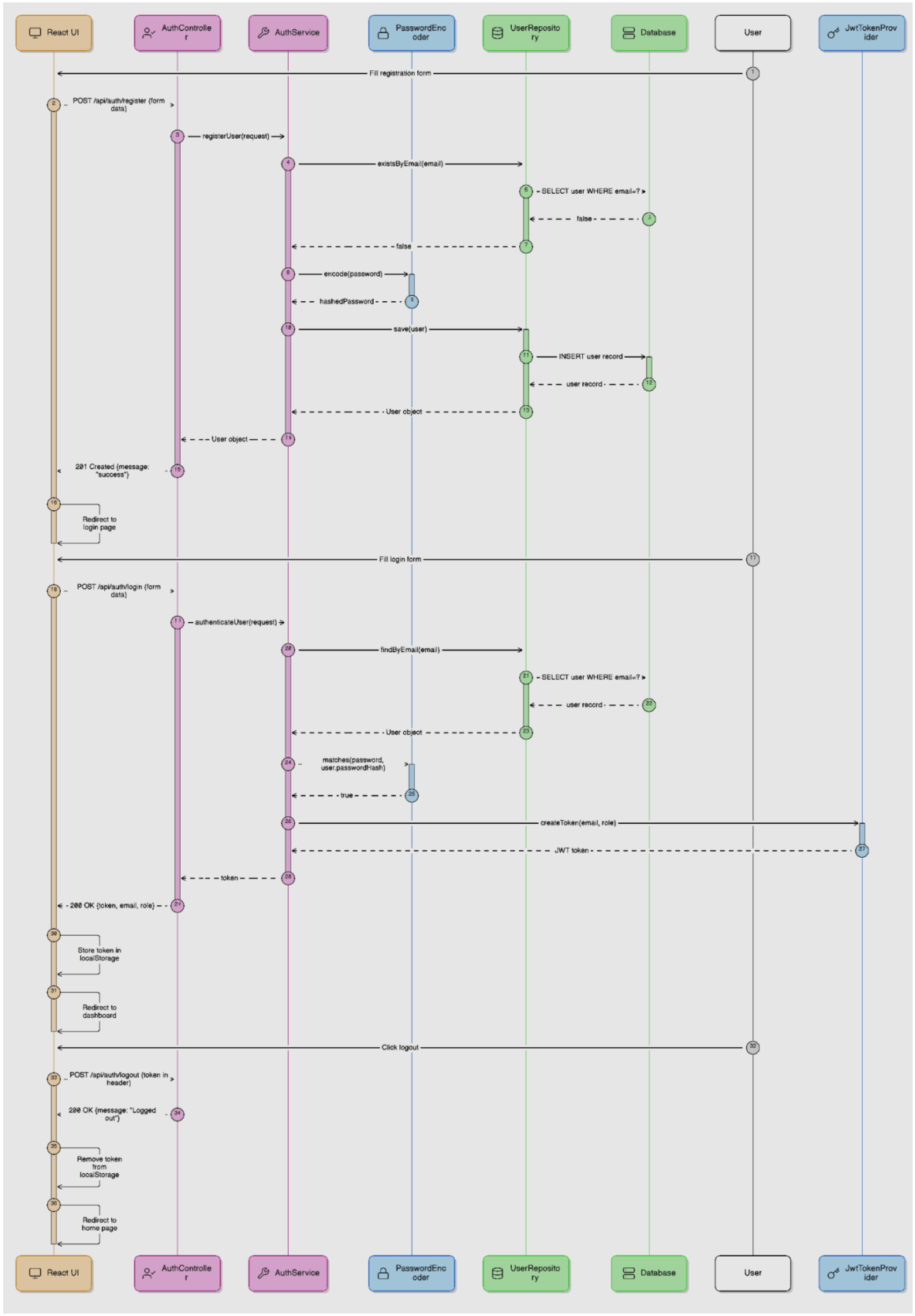
The Activity Diagram models the workflow for three critical processes: Artist Registration, Artist Login, and Artist Logout. It uses swimlanes to delineate responsibilities between the User/Client Side, Backend API, and Database layers. The diagram includes decision points for validation checks, error handling paths, and successful completion flows. Key decision nodes validate registration inputs, check email uniqueness, verify login credentials, and handle authentication token management.

5.4. Class Diagram



The Class Diagram presents the object-oriented structure of the InkSlot backend using Spring Boot architecture. It defines classes such as User, Artist, AuthController, AuthService, UserRepository, PasswordEncoder, and JwtTokenProvider, along with Data Transfer Objects (DTOs) for request and response handling. The diagram specifies class attributes with data types, methods with return types, and relationships including dependencies, associations, and compositions. This structure follows the layered architecture pattern with clear separation between controllers, services, and repositories.

5.5. Sequence Diagram



The Sequence Diagrams illustrate the temporal interactions between system components for three key operations: Artist Registration, Artist Login, and Artist Logout. Each diagram shows the message flow between React UI, Spring Boot controllers, service layer, repositories, and the database. Lifelines represent participating objects, while activation boxes indicate processing periods. The diagrams detail authentication workflows including password hashing, JWT token generation, database operations, and client-side token storage, demonstrating the complete request-response cycle for authentication operations.

6. Appendices

Appendix A: Technology Stack

Frontend

- Framework: React 18.x
- Routing: React Router DOM 6.x
- State Management: React Context API or Redux Toolkit
- HTTP Client: Axios
- UI Framework: Material-UI (MUI) or Bootstrap 5
- Form Handling: Formik or React Hook Form
- Validation: Yup validation schema
- Build Tool: Vite or Create React App
- Package Manager: npm or yarn
- Deployment: Vercel or Netlify

Backend

- Framework: Spring Boot 3.x
- Language: Java 17+
- Build Tool: Maven or Gradle
- Security: Spring Security with JWT
- OAuth: Spring Security OAuth 2.0 (Google)
- ORM: Spring Data JPA / Hibernate
- Validation: Hibernate Validator (JSR-380)
- API Documentation: SpringDoc OpenAPI (Swagger)
- Email: Spring Mail with SMTP
- Logging: SLF4J with Logback
- File Upload: Apache Commons FileUpload or Spring Multipart

Database

- Primary Options: PostgreSQL 13+ or MySQL 8.0+
- Alternative: MongoDB 5.0+ (NoSQL option)

- Connection Pooling: HikariCP (default in Spring Boot)
- Migration Tool: Flyway or Liquibase (optional)

Development Tools

- Version Control: Git with GitHub (REQUIRED)
- IDE: IntelliJ IDEA, Visual Studio Code, Eclipse
- API Testing: Postman, Insomnia
- Database Client: pgAdmin (PostgreSQL), MySQL Workbench, DBeaver
- Package Manager: npm/yarn (frontend), Maven/Gradle (backend)

Deployment

- Application Server: Embedded Tomcat (Spring Boot default)
- Cloud Hosting: Render, Railway, or Vercel (REQUIRED for project)
- Version Control Integration: GitHub (automatic deployment via CI/CD)
- CI/CD: GitHub Actions (automatic deployment on push)
- Frontend Deployment: Vercel or Netlify
- Database Hosting: Provided by cloud platform (Render Postgres, Railway, etc.)
- Containerization: Docker (optional, supported by Render/Railway)

Mobile (Phase 2)

- Platform: Android
- Language: Kotlin
- Architecture: MVVM (Model-View-ViewModel)
- Networking: Retrofit for REST API calls
- Authentication: OAuth 2.0 integration with Google Sign-In
- UI Framework: Jetpack Compose or Material Design 3
- Local Database: Room (SQLite wrapper)
- Dependency Injection: Hilt

Appendix B: API Endpoint Specifications

Authentication Endpoints

Register

POST /api/auth/register

Request Body: {

"email": "string",

"password": "string",

"fullName": "string",

"phoneNumber": "string"

}

Response: 201 Created

{

"message": "Registration successful",

"userId": "long",

"token": "string (JWT)"

}

Login

POST /api/auth/login

Request Body: {

"email": "string",

"password": "string"

}

Response: 200 OK

{

"token": "string (JWT)",

"email": "string",

"role": "string",

"userId": "long",

"expiresIn": "long (milliseconds)"

}

OAuth Google Login

POST /api/auth/oauth/google

Request Body: {

"idToken": "string (Google ID token)"

}

Response: 200 OK

{

"token": "string (JWT)",

"email": "string",

"role": "string",

"userId": "long",

"isNewUser": "boolean",

"message": "Authentication successful"
}

Logout

POST /api/auth/logout
Headers: Authorization: Bearer {token}
Response: 200 OK
{
"message": "Logged out successfully"
}

Refresh Token

POST /api/auth/refresh
Headers: Authorization: Bearer {token}
Response: 200 OK
{
"token": "string (new JWT)"
}

Appendix C: Database Schema Details

Users Table

```
CREATE TABLE users (
  user_id SERIAL PRIMARY KEY,
  email VARCHAR(255) UNIQUE NOT NULL,
  password_hash VARCHAR(255),
  full_name VARCHAR(255) NOT NULL,
  phone_number VARCHAR(20),
  role VARCHAR(20) NOT NULL DEFAULT 'ARTIST' CHECK (role IN ('ARTIST',
'ADMIN')),
  google_id VARCHAR(255) UNIQUE,
  oauth_provider VARCHAR(50),
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
  is_active BOOLEAN DEFAULT TRUE,
```

```
 CONSTRAINT chk_auth_method CHECK (password_hash IS NOT NULL OR google_id
IS NOT NULL)
);

CREATE INDEX idx_users_email ON users(email);
CREATE INDEX idx_users_google_id ON users(google_id);
```

Artists Table

```
CREATE TABLE artists (
 artist_id SERIAL PRIMARY KEY,
 user_id INTEGER UNIQUE NOT NULL,
 bio TEXT,
 years_of_experience INTEGER CHECK (years_of_experience >= 0),
 specialties TEXT,
 profile_image_url VARCHAR(500),
 created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
 updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
 FOREIGN KEY (user_id) REFERENCES users(user_id) ON DELETE CASCADE
);

CREATE INDEX idx_artists_user_id ON artists(user_id);
```

Artist Portfolio Table

```
CREATE TABLE artist_portfolio (
 portfolio_id SERIAL PRIMARY KEY,
 artist_id INTEGER NOT NULL,
 image_url VARCHAR(500) NOT NULL,
 description TEXT,
 created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
 updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
 FOREIGN KEY (artist_id) REFERENCES artists(artist_id) ON DELETE CASCADE
);

CREATE INDEX idx_portfolio_artist_id ON artist_portfolio(artist_id);
```

Bookings Table

```
CREATE TABLE bookings (
 booking_id SERIAL PRIMARY KEY,
 artist_id INTEGER NOT NULL,
 client_name VARCHAR(255) NOT NULL,
```

```
  client_email VARCHAR(255) NOT NULL,
  client_phone VARCHAR(20) NOT NULL,
  appointment_date DATE NOT NULL,
  appointment_time TIME NOT NULL,
  design_image_url VARCHAR(500),
  approximate_size VARCHAR(100),
  estimated_price DECIMAL(10, 2),
  status VARCHAR(20) NOT NULL DEFAULT 'PENDING' CHECK (status IN
('PENDING', 'CONFIRMED', 'COMPLETED', 'CANCELLED')),
  cancellation_reason TEXT,
  reference_number VARCHAR(20) UNIQUE NOT NULL,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
  FOREIGN KEY (artist_id) REFERENCES artists(artist_id) ON DELETE CASCADE
);


CREATE INDEX idx_bookings_artist_id ON bookings(artist_id);
CREATE INDEX idx_bookings_status ON bookings(status);
CREATE INDEX idx_bookings_date ON bookings(appointment_date);
CREATE UNIQUE INDEX idx_booking_time_conflict ON bookings(artist_id,
appointment_date, appointment_time) WHERE status != 'CANCELLED';
```

Artist Schedules Table

```
CREATE TABLE artist_schedules (
  schedule_id SERIAL PRIMARY KEY,
  artist_id INTEGER NOT NULL,
  date DATE NOT NULL,
  start_time TIME NOT NULL,
  end_time TIME NOT NULL,
  is_available BOOLEAN DEFAULT TRUE,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
  FOREIGN KEY (artist_id) REFERENCES artists(artist_id) ON DELETE CASCADE,
  CONSTRAINT chk_time_order CHECK (start_time < end_time),
  UNIQUE (artist_id, date, start_time)
);

CREATE INDEX idx_schedules_artist_id ON artist_schedules(artist_id);
CREATE INDEX idx_schedules_date ON artist_schedules(date);
```

Shop Rates Table

```
CREATE TABLE shop_rates (
  rate_id SERIAL PRIMARY KEY,
  size_category VARCHAR(50) NOT NULL UNIQUE,
  base_price DECIMAL(10, 2) NOT NULL,
  description TEXT,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP
);

CREATE INDEX idx_rates_category ON shop_rates(size_category);
```

OAuth Tokens Table (Optional - for storing refresh tokens)

```
CREATE TABLE oauth_tokens (
  token_id SERIAL PRIMARY KEY,
  user_id INTEGER NOT NULL,
  provider VARCHAR(50) NOT NULL,
  refresh_token TEXT,
  expires_at TIMESTAMP,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  FOREIGN KEY (user_id) REFERENCES users(user_id) ON DELETE CASCADE
);


CREATE INDEX idx_oauth_tokens_user_id ON oauth_tokens(user_id);
```

Appendix D: Error Code Reference

Error Code	HTTP Status	Description	User Action
ERR_AUTH_001	401	Invalid credentials (email/password)	Verify email and password are correct
ERR_AUTH_002	401	Token expired	Log in again to generate new token
ERR_AUTH_003	403	Insufficient permissions	Contact administrator for access
ERR_AUTH_004	401	Token invalid or malformed	Clear browser cache and log in again

ERR_OAUTH_001	400	Invalid Google ID token	Try OAuth login again
ERR_OAUTH_002	400	OAuth provider error	Try again later or use email login
ERR_REG_001	409	Email already registered	Use different email or log in
ERR_REG_002	400	Invalid email format	Check email format (user@domain.com)
ERR_REG_003	400	Password too weak	Use 8+ chars, uppercase, lowercase, number
ERR_REG_004	400	Missing required fields	Fill all required fields (marked with *)
ERR_BOOK_001	409	Time slot not available	Select different date/time
ERR_BOOK_002	400	Invalid date (past date)	Select future date
ERR_BOOK_003	400	Artist not found	Select valid artist from list
ERR_BOOK_004	400	Booking reference not found	Check booking reference number
ERR_FILE_001	400	File size exceeds limit	Reduce image size (max 5MB)
ERR_FILE_002	400	Invalid file format	Use JPG or PNG format only
ERR_FILE_003	413	Payload too large	Upload smaller images
ERR_VAL_001	400	Validation failed	Check form for errors
ERR_VAL_002	400	Invalid input data	Verify data types and formats
ERR_DB_001	500	Database connection error	Try again later - contact support
ERR_DB_002	500	Database constraint violation	Ensure unique values (email)
ERR_SRV_001	500	Internal server error	Contact support team
ERR_SRV_002	503	Service unavailable	System under maintenance, try later
ERR_RATE_001	404	Rate not found	Invalid rate ID
ERR_SCHEDULE_001	409	Schedule conflict	Time slot overlaps with existing schedule

Appendix E: Security Best Practices

Password Policy

- Minimum 8 characters
- At least one uppercase letter (A-Z)
- At least one lowercase letter (a-z)
- At least one number (0-9)
- Special characters recommended (!@#$%^&*)
- Cannot contain user's email or name
- Password history: prevent reuse of last 3 passwords (future enhancement)
- Password reset: link expires after 15 minutes

JWT Token Security

- Algorithm: HS512 (HMAC with SHA-512)
- Token payload includes: userId, email, role, issued-at time, expiration time
- Secret key: minimum 256 bits (32 characters), stored in environment variables
- Token expiration: 24 hours for access token
- Refresh token: 7 days (optional for Phase 2)
- Storage: httpOnly cookies (preferred) or localStorage (development)
- CORS restrictions: only accept requests from registered frontend domains

OAuth 2.0 Security (Google)

- Redirect URI must be registered with Google Cloud Console
- ID tokens validated with Google's public certificates
- Refresh tokens stored securely (if using offline access)
- OAuth scopes limited to: openid, email, profile only
- No sensitive data stored from OAuth provider except email and name

Input Validation

- All inputs sanitized using OWASP Java Encoder
- Email validation using RFC 5322 regex pattern
- Phone number validation: 7-15 digits
- Date validation: must be future date for bookings
- Time validation: must be within business hours
- Text length validation: enforce maximum character limits
- SQL injection prevention: prepared statements via JPA (automatic)
- XSS protection: HTML encoding on all user inputs
- CSRF tokens for state-changing operations (GET vs POST)

File Upload Security

- Whitelist allowed extensions: .jpg, .jpeg, .png only
- File type verification: check MIME type AND magic bytes (not just extension)
- Maximum file size: 5MB for images
- Virus scanning: ClamAV integration (optional for production)
- Uploaded files stored outside web root directory (/uploads/private/)
- Unique filename generation: UUID + timestamp (prevents overwriting)
- Image reprocessing: strip EXIF metadata, resize to standard dimensions
- Serve files via controlled endpoint with authentication check

Database Security

- All credentials stored in environment variables (never in code)
- Password hashing: BCrypt with cost factor 10
- Database connections use SSL/TLS
- SQL transactions for atomic operations
- Foreign key constraints enforce referential integrity
- Row-level security: artists can only view their own bookings
- Sensitive columns encrypted: passwords (irreversible hash)

Network Security

- HTTPS/TLS 1.3 required for all production communications
- CORS headers configured restrictively
- Rate limiting: 5 login attempts per 15 minutes per IP
- API request throttling: 100 requests per minute per user
- Security headers: X-Frame-Options, X-Content-Type-Options, Content-Security-Policy
- HSTS (HTTP Strict Transport Security): 1 year max-age

Logging and Monitoring

- Log all authentication attempts (success/failure)
- Log all booking modifications with user identification
- Log all file uploads with user info
- Sanitize logs: don't log passwords, tokens, or sensitive data
- Retain logs for 90 days minimum
- Monitor for suspicious patterns: repeated failed logins, unusual IP addresses
- Alert on security events: breached passwords, unauthorized access attempts

Appendix F: Future Enhancements

Phase 1 Completion (Current Scope - Sprint 2 & 3)

1. User Authentication (Email/Password + OAuth Google)

2. REST API integration (all CRUD endpoints)
3. Database operations (MySQL/PostgreSQL)
4. Error handling and validation (comprehensive)
5. Cloud deployment (Render/Railway/Vercel)
6. Git/GitHub version control

Phase 2 Features (Android Mobile App + Enhancement - Sprint 4 & 5)

1. Native Android Application: Kotlin-based mobile app with MVVM architecture
2. Mobile Authentication: Email/password and Google Sign-In for Android
3. Mobile UI: Native Android UI components with Material Design 3
4. API Integration: Full REST API connectivity to Spring Boot backend
5. Offline Support: Local SQLite cache for offline viewing of artist profiles
6. Push Notifications: Firebase Cloud Messaging for booking reminders
7. QR Code Booking: Quick booking via artist QR codes
8. Payment Integration: Google Pay integration for deposits

Phase 3 Features (Advanced Enhancement - Future)

1. Payment Processing: Stripe/PayPal integration for full payment handling
2. AI Price Estimation: Machine learning model for automatic pricing based on design complexity
3. SMS Notifications: Twilio integration for appointment reminders via SMS
4. Video Consultation: WebSocket-based video calling between artists and clients
5. Real-time Chat: Messaging system for artist-client communication
6. Admin Dashboard: Comprehensive analytics, user management, revenue reports
7. Artist Ratings & Reviews: Client review system with star ratings
8. Loyalty Program: Points-based rewards for repeat customers
9. Multi-location Support: Franchise management with multiple tattoo shops
10. Advanced Scheduling: Recurring appointments, break times, buffer time management

Appendix G: Glossary of Terms

Term	Definition
Artist Dashboard	Authenticated area where artists manage profiles, portfolios, schedules, and bookings

Availability Slot	Time period during which an artist is available for appointments (date + time range)
Booking Reference	Unique identifier (e.g., INK-2026-001) assigned to each appointment for tracking
BCrypt	Password hashing algorithm that includes salt and cost factor for security
CRUD	Create, Read, Update, Delete - four basic operations for data management
Design Upload	Client's tattoo design image submitted during booking process
Guest User	Unauthenticated visitor who can browse and create bookings without registration
JWT (JSON Web Token)	Compact, URL-safe token format used for stateless authentication
OAuth 2.0	Industry-standard authorization framework for secure third-party authentication
Portfolio	Collection of an artist's previous tattoo work showcased to potential clients
Protected Endpoint	API route requiring valid JWT authentication token for access
REST API	Representational State Transfer - architectural style for web services using HTTP methods
Role-Based Access Control (RBAC)	Authorization approach granting permissions based on user roles (Guest, Artist, Admin)
Session	Period during which a user remains authenticated after logging in
Single Page Application (SPA)	Web app that loads once and dynamically updates content without page reloads
Status Workflow	Booking progression through states: Pending -> Confirmed -> Completed/Cancelled
Webhook	HTTP callback mechanism for event-driven notifications (e.g., payment processing)
Atomic Transaction	Database operation that completes entirely or not at all (ACID properties)
Foreign Key	Database constraint linking a column to a primary key in another table

Index	Database optimization structure accelerating data retrieval speed
Pagination	Breaking large datasets into smaller pages (e.g., 20 items per page)
Rate Limiting	Technique restricting number of requests per time period
CORS	Cross-Origin Resource Sharing - mechanism allowing cross-domain API requests
Environment Variables	Configuration settings stored outside code (DATABASE_URL, JWT_SECRET, etc.)
Dependency Injection	Design pattern providing dependencies to classes rather than having them create dependencies

Appendix H: References and Resources

Official Documentation

- Spring Boot: https://spring.io/projects/spring-boot
- React: https://react.dev
- Spring Security: https://docs.spring.io/spring-security/reference/
- JWT.io: https://jwt.io
- Google OAuth 2.0: https://developers.google.com/identity/protocols/oauth2
- PostgreSQL: https://www.postgresql.org/docs/
- MySQL: https://dev.mysql.com/doc/

Development Standards & Best Practices

- REST API Design Guidelines: https://restfulapi.net
- OWASP Top 10: https://owasp.org/www-project-top-ten/
- WCAG 2.1 Accessibility: https://www.w3.org/WAI/WCAG21/quickref/
- Java Code Conventions: https://www.oracle.com/java/technologies/javase/codeconventions-contents.html
- Spring Best Practices: https://spring.io/guides
- OAuth 2.0 Security: https://tools.ietf.org/html/rfc6749

Tutorials & Learning Resources

- Baeldung (Spring Boot): https://www.baeldung.com
- MDN Web Docs: https://developer.mozilla.org
- FreeCodeCamp: https://www.freecodecamp.org
- Udemy Spring Boot Courses: https://www.udemy.com/courses/spring/

UML & System Design

- UML 2.5 Specification: https://www.omg.org/spec/UML/
- PlantUML: https://plantuml.com
- Lucidchart: https://www.lucidchart.com
- Draw.io: https://www.draw.io

Development Tools

- IntelliJ IDEA: https://www.jetbrains.com/idea/
- Visual Studio Code: https://code.visualstudio.com/
- Postman: https://www.postman.com/
- Git Documentation: https://git-scm.com/doc

Deployment & Hosting

- Render Documentation: https://render.com/docs
- Railway Documentation: https://docs.railway.app/
- Vercel Documentation: https://vercel.com/docs
- Docker: https://docs.docker.com/

Security Resources

- OWASP Cheat Sheet Series: https://cheatsheetseries.owasp.org/
- PortSwigger Web Security: https://portswigger.net/web-security
- Snyk Security Blog: https://snyk.io/blog/