CEBU INSTITUTE OF TECHNOLOGY
UNIVERSITY

IT342- GO2
SYSTEMS INTEGRATION AND ARCHITECTURE 1

FUNCTIONAL REQUIREMENTS SPECIFICATION (FRS)

Project Title: DigiHealth: Digital Clinic & Appointment Booking System

Prepared By: [Jessie Noel Lapure /DigiHealth/ Developer]

Date of Submission: [10/22/2025]

Version: 1.0

Table of Contents

1. Introduction

This Functional Requirements Specification (FRS) outlines the Minimum Viable Product (MVP) features for the DigiHealth: Digital Clinic & Appointment Booking System. Given the revised timeline with final presentation in the 1st week of December 2025, this document prioritizes core functionalities that can be realistically implemented, tested, and demonstrated within 7-8 weeks (October 20 - December 5, 2025).

Purpose: This Functional Requirements Specification (FRS) defines the features, behavior, and constraints of the **DigiHealth Online Clinic & Appointment System** — a web and mobile platform connecting patients directly to licensed doctors for virtual and in-clinic consultations.

Scope Adjustment: This document covers the system's functional requirements, non-functional requirements, and design assumptions necessary for implementation, deployment, and maintenance in real-world use.

2. Project Overview

Background

Campus and small community health clinics rely on manual processes that create inefficiencies. DigiHealth MVP will demonstrate core digital solutions within the constrained academic timeline.

Goals (MVP Focus)

- Primary Goal: Develop a functional demo showcasing digital appointment booking and basic patient record management
- Secondary Goal: Implement secure authentication and role-based access
- Presentation Goal: Demonstrate working mobile app and web dashboard by December 1st week

MVP Scope (What We WILL Build)

- Mobile Application (Android): Patient registration and appointment booking
- Web Dashboard: Doctor access to patient records and appointments
- Backend APIs: Core data management and authentication
- Database: Patient, appointment, and basic user data storage
- Authentication: Google OAuth 2.0 integration

3. Team

Name	Role
Jessie Noel Lapure	Full Stack Developer / Project Manager
William Bustamante	Full Stack Developer
Joel Verano	Full Stack Developer
Matthew Rimar Martus	Full Stack Developer

4. Functional Requirements

FR-1: Patient Registration (Mobile App)

The system shall allow patients to register using either Google OAuth 2.0 authentication or manual form input of patient personal information such as name, contact number, email address, and require completion of a basic medical profile including age, gender, allergies, and existing medical conditions to support medical consultations.

FR-2: Patient Login (Mobile App)

The system shall allow registered patients to log in using Google OAuth 2.0 or their registered email and password, validate user credentials securely, and provide authenticated access to patient functionalities such as appointment booking and profile management.

FR-3: Doctor Registration (Web App)
The system shall allow doctors to register an account by providing their full name, specialization, license number, contact details, and schedule availability. The system shall require administrator approval before newly registered doctors become visible for patient appointment selection.

FR-4: Doctor Login and Schedule Management
The system shall allow approved doctors to securely log in and access their dashboard, which displays upcoming appointments, patient information, and schedule management tools for adjusting availability and booking slots

FR-5: Appointment Booking (Mobile App)

The system shall allow patients to browse available doctors by name or specialization, view open time slots, and select a preferred appointment date and time. The system shall confirm bookings and notify both the patient and doctor of appointment details.

FR-6: Appointment Management

The system shall allow doctors to view all scheduled appointments, mark appointments as completed or cancelled, and update booking statuses in real time and automatically reflect updates on the patient's appointment list.

FR-7: Patient Record Management (Web Dashboard)
The system shall allow doctors to view and update basic patient records containing consultation notes, provide search functionality to find patients by name or ID, and display patient appointment history. Restrict access to ensure that only assigned doctors can view or edit a patient's medical record.

FR-8: Role-Based Access Control
The system shall implement role-based access for three distinct user types: Patient, Doctor, and System Administrator. Each role shall have specific permissions, ensuring that sensitive data and system features are accessible only to authorized users.

FR-9: Administrator Management
The system shall allow the system administrator to manage user accounts by creating, approving, or deactivating doctor accounts, configuring clinic details, defining appointment policies, and monitoring overall system status.

FR-10: System Reports and Analytics
The system shall generate summary reports displaying appointment statistics, patient registration counts, and doctor activity metrics through simple charts or tables accessible via the system administrator dashboard.

FR-12: Admin System Monitoring
The system shall provide a system status dashboard, display database connection status, include API health check functionality, and allow viewing of error logs within.

5. Non-Functional Requirements

NFR-1: Platform Requirements
The system shall support Android mobile application with minimum viable functionality for patient registration and appointment booking.

NFR-2: Performance Requirements
The system shall provide response times under 5 seconds for basic operations, support concurrent access for 5-10 users during demonstration purpose, and maintain stable uptime during presentation demonstration periods.

NFR-3: Security Requirements
The system shall implement Google OAuth 2.0 for authentication, use HTTPS encryption for data transmission, and apply secure storage practices to protect sensitive patient information in compliance with privacy regulations.

NFR-4: Data Privacy and Compliance
The system shall adhere to applicable data protection laws, ensuring that patient information is accessible only to authorized medical professionals. The system shall comply with HIPAA-equivalent standards and the Philippine Data Privacy Act of 2012

NFR-5: Reliability and Availability

The system shall maintain a minimum uptime of ninety-nine percent during operational hours and perform automatic daily database backups to prevent data loss.

NFR-6: Usability Requirements
The system shall provide an intuitive and user-friendly interface for both mobile and web platforms, ensuring easy navigation for patients, doctors, and administrators with consistent design and accessible layout.

NFR-7: Scalability Requirements
The system shall be designed with modular architecture to support future expansion such as teleconsultation, online payments, and multi-clinic support without significant code refactoring.

NFR-8: Maintainability Requirements
The system shall be developed following standard coding practices and proper

documentation, allowing developers to update, fix, or enhance system components efficiently with minimal service interruption.

6. Use Cases

Use Case 1: Patient Registration

Actor: Patient
Precondition: Patient has the DigiHealth mobile app installed.
Main Flow:

1. The Patient opens the DigiHealth mobile app.
2. The Patient taps **"Register"** and chooses between Google OAuth 2.0 sign-in or manual registration.
3. The Patient provides basic profile information (name, phone, email and emergency contact).
4. The Patient completes the basic medical profile including age, gender, allergies, and existing medical conditions.
5. The System validates input and creates a new patient account.
6. The System stores basic details in the database.

Postcondition:
A new patient account is successfully created and ready for login.

Use Case 2: Patient Login

Actor: Patient
Precondition: Patient has an existing registered account.
Main Flow:

1. The Patient opens the DigiHealth app.
2. The selects **"Login"** using Google OAuth 2.0 credentials or manual login.
3. The System verifies credentials or token provided.
4. The system establishes a secure session for the authenticated user.
5. The System redirects the patient to the appointment booking dashboard.

Postcondition:
The patient is logged in and can now access appointment booking features..

Use Case 3: Doctor Registration

Actor: Doctor
Precondition: The doctor has access to the web dashboard and a stable internet connection.
Main Flow:

1. The doctor accesses the DigiHealth web portal.
2. The doctor selects “Register” and provides details including name, specialization, license number, contact information, and availability.
3. The system validates the information and forwards the registration request to the administrator.
4. The administrator reviews and approves the registration.
   The system activates the doctor account and adds it to the doctor listing.

Postcondition: A verified doctor account is created and visible for patient appointment booking.

Use Case 4: Doctor Login and Dashboard Access

Actor: Doctor
Precondition: The doctor has an approved account.
Main Flow:

1. The doctor accesses the DigiHealth web dashboard.
2. The doctor enters login credentials or uses Google OAuth 2.0 authentication.
3. The system validates the credentials and authorizes access.
4. The system displays the doctor’s personalized dashboard showing upcoming appointments and patient details.
5. The doctor can adjust availability or manage appointments as needed.

Postcondition: The doctor successfully accesses the dashboard and can manage schedules and consultations.

Use Case 5: Browse Doctors and Book Appointment

Actor: Patient
Precondition: The patient is logged in to the system.
Main Flow:

1. The patient opens the appointment booking section.
2. The system displays a searchable list of available doctors with their specialties and schedules.
3. The patient selects a doctor and chooses an available date and time slot.
4. The patient reviews the appointment details and confirms the booking.
5. The system saves the appointment data in the database.
6. The system sends confirmation notifications to both doctor and patient.

Postcondition: The appointment is successfully booked and visible in both patient and doctor dashboards.

Use Case 6: Manage Appointments (Doctor)

Actor: Doctor
Precondition: The doctor is logged in and has active appointments.
Main Flow:

1. The doctor accesses the appointment management section of the dashboard.
2. The system displays the list of upcoming patient appointments.
3. The doctor reviews each appointment and can mark it as completed or cancelled.
4. The system updates the appointment status in real time.
5. The system sends status update notifications to the corresponding patient.

Postcondition: Appointment records are updated and synchronized across doctor and patient interfaces.

Use Case 7: Patient Record Management

Actor: Doctor
Precondition: The doctor has an active appointment with the patient.
Main Flow:

1. The doctor selects a completed appointment or patient from the dashboard.
2. The system displays the patient's personal information and previous consultation notes.
3. The doctor inputs new medical notes, prescriptions, or observations.
4. The system saves and updates the patient's medical record in the database.
5. The doctor can view or edit previous notes for reference.

Postcondition: The patient's medical record is securely updated and accessible only to the assigned doctor.

Use Case 8: User and System Management

Actor: System Administrator
Precondition: The administrator is authenticated.
Main Flow:

1. The administrator logs in to the web dashboard.
2. The administrator reviews pending doctor registration requests.
3. The administrator approves, rejects, or deactivates user accounts as necessary.
4. The administrator updates clinic settings such as booking policies, hours, and system information.
5. The system records and saves all configuration and user management changes.

Postcondition: System settings and user accounts are updated successfully.

Use Case 9: Generate Reports and Analytics

Actor: System Administrator
Precondition: The administrator is logged in to the web dashboard.
Main Flow:

1. The administrator accesses the reporting and analytics module.
2. The system retrieves appointment and registration data from the database.
3. The system generates visual summaries such as total patients, total appointments, and doctor activity metrics.
4. The administrator views the reports and can export them if needed.
5. The system logs the report generation for audit purposes.
   Postcondition: Reports are successfully generated and available for review or export.

7. System Interfaces

External Integrations

- Google OAuth 2.0 API: For authentication (both mobile and web)
- Cloud Database: MySQL hosted on cloud platform
- Hosting Platform: Railway/Heroku/AWS Free Tier APIs

Internal Interfaces

- Mobile App ↔ Backend: REST API calls for appointments and patient data
- Web Dashboard ↔ Backend: REST API calls for staff operations
- Backend ↔ Database: JPA/Hibernate for data persistence

8. Assumptions and Constraints

Assumptions

- Development Environment: Team has access to required development tools
- User Testing: Limited to team testing and basic functionality verification
- Data: Sample data will be used for demonstration purpose

Revised Timeline Constraints

- Total Timeline: 7-8 weeks (October 14 - December 6, 2025)
- Requirements & Design: 1 week
- Development Phase: 4 weeks maximum
- Testing Phase: 1 week
- Presentation Prep: 1 week
- Final Presentation: December 1st week, 2025

Resource Constraints

- Team Size: 4 student developers with academic commitments
- Budget: Limited to free tier services and minimal costs
- Infrastructure: Free hosting platforms only

Feature Constraints (MVP Approach)

- No QR Code functionality in initial version
- No advanced analytics - basic charts only if time permits
- No complex inventory management - manual entry only
- No real-time notifications - basic email notifications at most
- No mobile iOS version - Android only
- NO advanced security features
- NO detailed audit trails
- NO complex system administration
- NO server configuration interfaces

9. Acceptance Criteria

Acceptance Criteria

Core Functionality Demonstration:

- ✅ Patient registration works using both Google OAuth 2.0 and manual form input.
- ✅ Patient profile setup collects complete personal and basic medical information.
- ✅ Patient login is successful and redirects to the appointment booking dashboard.
- ✅ Patients can browse doctors by name or specialization.
- ✅ Appointment booking confirms date and time selections for both patient and doctor.
- ✅ Doctors can register, select their specialization, and await admin approval.
- ✅ Approved doctors can log in and manage their schedule and appointments.
- ✅ Administrators can approve or deactivate doctor accounts and update clinic settings.

Technical Demonstration:

- ✅ All API endpoints respond within five seconds during testing.
- ✅ Mobile and web applications connect successfully to the shared backend database.
- ✅ User data remains persistent after logout, restart, or database refresh.
- ✅ The system supports at least one hundred concurrent users during testing.

Security and Data Privacy

- ✅ Google OAuth 2.0 authentication functions correctly for web and mobile access.
- ✅ All data transmissions use HTTPS encryption for security.
- ✅ Patient medical information is accessible only to the assigned doctor.
- ✅ Data handling complies with the Data Privacy Act of 2012 and HIPAA-equivalent standards.

Usability and Interface

- ✅ Mobile application interface is user-friendly, responsive, and easy to navigate.
- ✅ Web dashboard for doctors and admins loads properly on common browsers.
- ✅ Confirmation and error messages display clearly for all major user actions.
- ✅ User feedback messages appear for successful bookings, updates, and profile changes.

Administrative and Reporting Features

- ✅ Administrator can configure clinic details, booking policies, and operational hours.
- ✅ Reports display accurate totals for patients, appointments, and doctor activity.
- ✅ Report data can be exported for documentation or review.

System Reliability and Maintenance

- ✅ Application uptime remains stable at 99% during testing.
- ✅ Automatic daily database backup completes successfully.
- ✅ Error handling and recovery mechanisms function without crashes.
- ✅ Activity logs for admin and doctor actions are recorded properly.

10. Appendices

Appendix A: Revised Project Timeline

Phase	Duration	Start Date	End Date	Key Deliverables
Requirements & Design	1 week	Oct 14, 2025	Oct 22, 2025	FRS Document, UI Mockups, Database Schema
Core Development (MVP)	4 weeks	Nov 1, 2025	Nov 24, 2025	Android App, Web Dashboard, Backend APIs

Testing & Integration	1 week	Nov 25, 2025	Dec 1, 2025	Bug fixes, Integration testing
Presentation Preparation	1 week	Dec 4, 2025	Dec 5, 2025	Demo setup, Presentation materials

Total Timeline: 8 weeks
Presentation Date: December 4-5, 2025 (estimated)

Appendix B: MVP Development Priorities

Priority Level	Features	Development Time	Status
MUST HAVE	Patient registration, App Appointment booking, Staff dashboard, Authentication	14-18 days	Required for demo
SHOULD HAVE	Patient search, Role-based access	5-7 days	Important for Completeness

COULD HAVE	Basic inventory, Simple reports	5-7 days	Time permitting
DEFERRED	QR check-in, Advanced analytics, Queue management…	TBD	Post-presentation

Appendix C: Risk Mitigation (MVP Focus)

Risk	Likelihood	Impact	Mitigation Strategy
Timeline overrun	High	High	Focus on MVP features only, defer complex functionality
Technical integration issues	Medium	High	Start with simple authentication, avoid complex integrations
Team coordination challenges	Medium	Medium	Weekly team meetings, clear role assignments
Presentation demo failures	Medium	High	Prepare backup demo videos, test thoroughly before presentation

Appendix D: Technology Stack

Backend:

- Spring Boot 2.7+ (REST APIs)
- MySQL 8.0+ (Database)
- Google OAuth 2.0 (Authentication)

Frontend:

- React 18+ (Web Dashboard)
- MUI/Tailwind (Styling)
- Chart.js (Basic charts if implemented)

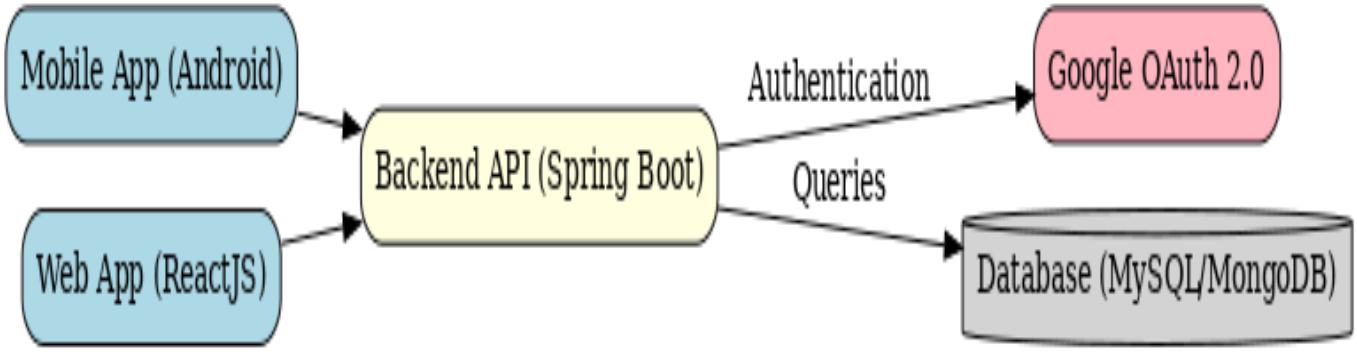
Mobile:

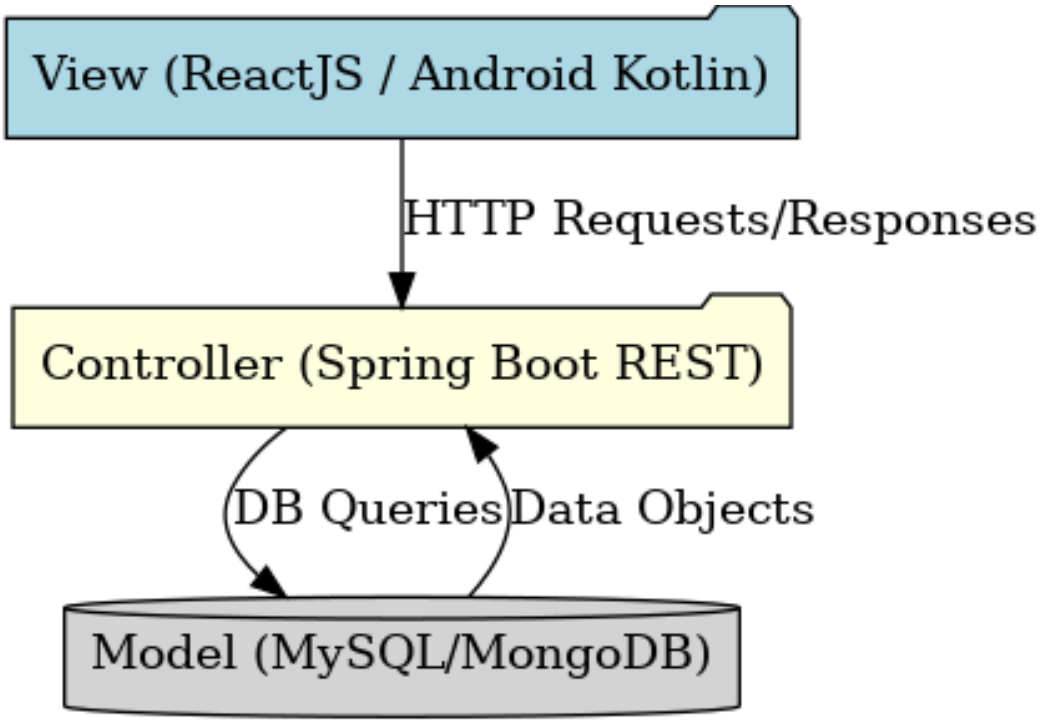
- Kotlin (Android Development)
- Android Studio (Development Environment)
- Retrofit (HTTP Client)

Hosting:

- Railway/Heroku (Backend)
- Cloud database service (MySQL)

Appendix E: Diagrams





Appendix F: [Figma Prototype]

Appendix G: Document Control:

- Version: 1.0 (MVP - Realistic Timeline)
- Last Updated: October 22, 2025
- Document Owner: William Bustamante
- Approval Authority: Mr. Frederick Revilleza
- Final Presentation: December 1st week, 2025

11. Approval Sign-off

	Full Name	Signature	Date
Prepared By:	Jessie Noel D. Lapure		
Developer	Matthew Rimar S. Martus		
Developer	Joel M. Virano Jr		
Developer	William P. Bustamante		
Lead Developer	Jessie Noel D. Lapure		
Reviewed By:			
Reviewed By:			
Approved By:	Mr. Frederick Revilleza		