
# Software Requirements Specification

## MyWelly Application

Health Consultation Booking Platform

**Prepared by:**

Safar Fatima Ezzahra, Elansari Zineb, Bellaou Salma

**Supervisor:**

Fouad Imane

**Date:** November 30, 2025

**Version:** 2.0

# Contents

# 1 Introduction

## 1.1 Purpose

This document specifies the software requirements for **MyWelly**, a comprehensive healthcare management platform designed to connect patients, doctors, and laboratories in Morocco. The purpose of this Software Requirements Specification (SRS) document is to provide a detailed description of the MyWelly web application, defining all functional and non-functional requirements of the system.

This document serves as a contractual agreement between stakeholders and the development team, ensuring that all parties have a clear understanding of the system's capabilities, constraints, and expected behavior.

The primary purpose of MyWelly is to:

- Streamline appointment booking between patients and healthcare providers
- Provide a secure platform for managing medical records and prescriptions
- Enable transparent doctor ratings and reviews
- Reduce waiting times and improve healthcare accessibility in Morocco

**This document is intended for:**

- Development team members implementing the Spring Boot application
- Project managers for planning, tracking, and resource allocation
- Healthcare professionals (doctors and laboratory staff) who will use the platform
- Patients who will benefit from the service
- System administrators responsible for deployment and maintenance
- Quality assurance teams for testing and validation

## 1.2 Product Scope

### 1.2.1 Software Product Identification

**MyWelly** is a web-based healthcare management system built using Spring Boot framework, targeting Morocco's healthcare sector. The application provides a comprehensive digital ecosystem that unites patients, doctors, and laboratories on a secure, scalable platform accessible through standard web browsers.

### 1.2.2 What the System Will Do

The MyWelly system provides the following capabilities:

1. **User Management**
    - Maintain centralized database of patients, doctors, and laboratories
    - Implement Spring Security with BCrypt password encryption

- Support role-based access control (PATIENT, DOCTOR, LABORATORY)
- Lock accounts after 5 consecutive failed login attempts

2. **Doctor Discovery**
   - Enable search by specialty, location, name, and price range
   - Display doctor profiles with ratings, reviews, and consultation fees
   - Support multiple filter criteria simultaneously
   - Show real-time availability and working hours
3. **Appointment Management**
   - Allow real-time appointment booking with conflict detection
   - Notify doctors of bookings, changes, and cancellations
   - Track appointment history and status
   - Automatically mark appointments as completed after 2 hours
4. **Review System**
   - Enable verified patient reviews with 1-5 star ratings
   - Calculate and display average ratings for doctors
   - Show review comments and timestamps
   - Update doctor ratings in real-time
5. **Account Management**
   - Allow users to update email and phone numbers
   - Enable password changes with confirmation
   - Support account deletion with data removal
   - Allow doctors to update location and certificates
6. **Security and Compliance**
   - Implement HTTPS encryption for all communications
   - Store passwords using BCrypt hashing
   - Provide secure session management

### 1.2.3 What the System Will NOT Do

The following features are explicitly out of scope:

1. **Telemedicine Services**
   - No video consultations or remote diagnosis features
   - No real-time chat between patients and doctors
2. **Financial Processing**
   - No direct payment processing or online payments
   - No insurance claim processing or verification
   - No billing or invoicing system

### 1.2.4 User Classes and Benefits

1. **Patients** - Individuals seeking healthcare services
   *Benefits:*

   - 24/7 appointment booking from any device with internet access
   - Easy search and comparison of doctors by specialty, location, rating, and price
   - View detailed doctor profiles with verified patient reviews
   - Access appointment history and track past consultations
   - Transparent consultation fee visibility before booking
   - Secure account with encrypted personal information

2. **Doctors** - Healthcare professionals managing their practice
   *Benefits:*

   - Automated appointment management system
   - Dashboard showing upcoming appointments and patient details
   - Ability to cancel appointments
   - View to patient reviews
   - Track performance metrics (total reviews, average rating)
   - Update professional information (location, certificates, fees)
   - Secure storage of patient consultation history
   - Increase visibility and attract new patients

3. **Laboratories** - Diagnostic testing facilities
   *Benefits:*

   - Increased visibility in the healthcare ecosystem
   - Ability to receive patient reviews and ratings
   - Profile management with location and working hours
   - Track reputation through rating system

4. **System Administrators** - Technical staff managing the platform
   *Benefits:*

   - Centralized user account management
   - Monitor system performance and usage
   - Manage database integrity and backups
   - Ensure security compliance and data protection
   - Handle user support and issue resolution

## 1.3 Definitions, Acronyms, and Abbreviations

- **SRS:** Software Requirements Specification
- **FR:** Functional Requirement
- **NFR:** Non-Functional Requirement
- **API:** Application Programming Interface
- **HTTPS:** Hypertext Transfer Protocol Secure
- **GDPR:** General Data Protection Regulation
- **MAD:** Moroccan Dirham (currency)
- **BCrypt:** Blowfish-based password hashing function
- **JPA:** Java Persistence API
- **H2:** Java-based relational database management system
- **Spring Security:** Authentication and access-control framework
- **Thymeleaf:** Server-side Java template engine
- **DTO:** Data Transfer Object

## 1.4 References

1. IEEE Std 830-1998: IEEE Recommended Practice for Software Requirements Specifications
2. Spring Boot Documentation: `https://spring.io/projects/spring-boot`
3. Spring Security Reference: `https://spring.io/projects/spring-security`
4. A mobile App : Doctorlib
5. MyWelly Feasibility Study Report (October 2025)

# 2 Overall Description

## 2.1 Product Perspective

### 2.1.1 System Context

MyWelly is a standalone web application that operates within Morocco's healthcare ecosystem. The system addresses critical inefficiencies including long wait times, fragmented communication between healthcare stakeholders, lack of transparent pricing, and absence of reliable doctor reviews.

The application serves as a central digital platform connecting three primary user types (patients, doctors, and laboratories) in a coordinated healthcare network. It replaces traditional phone-based appointment booking with an automated 24/7 online system while maintaining the integrity of in-person medical consultations.

### 2.1.2 System Architecture

MyWelly follows a three-tier architecture:

1. **Presentation Layer** - Thymeleaf-based web interface with responsive design
2. **Business Logic Layer** - Spring Boot application with service classes
3. **Data Layer** - H2/PostgreSQL relational database with JPA repositories

### 2.1.3 External Interfaces

**User Interfaces:**

The system provides a responsive web interface accessible through standard browsers (Chrome, Firefox, Safari, Edge). The interface features:

- Clean, intuitive design with green color scheme
- Role-based navigation menus
- Form validation with error messages
- Dashboard views customized per user type
- Mobile-responsive layouts

**Hardware Interfaces:**

No special hardware required. The system runs on standard web servers and is accessible from:

- Desktop computers (Windows, macOS, Linux)
- Laptops
- Tablets
- Smartphones

**Software Interfaces:**

- **Spring Boot Framework 3.x** - Core application framework

- **Spring Security** - Authentication and authorization
- **Spring Data JPA** - Database access layer
- **H2 Database** - Development database (file-based)
- **PostgreSQL/MySQL** - Production database (recommended)
- **Thymeleaf** - Server-side template engine
- **Web Browser** - Client-side rendering

**Communication Interfaces:**

- **HTTPS** - Secure encrypted communication for all web traffic
- **HTTP REST** - Internal API communication between layers

## 2.2 Product Functions

The main functional modules of MyWelly are organized by user role:

### 2.2.1 Patient Functions

1. **Account Management**
   - Register with personal information (name, date of birth, gender, ID number)
   - Login with email and password
   - Update email and phone number
   - Change password
   - Delete account permanently
2. **Doctor Search and Discovery**
   - Search doctors by name, specialty, location
   - Filter results by price range
   - View doctor profiles with ratings and reviews
   - Compare multiple doctors
3. **Appointment Management**
   - Book appointments with selected doctors
   - View upcoming appointments
   - Cancel scheduled appointments
   - Access appointment history
4. **Review Submission**
   - Rate doctors (1-5 stars)
   - Write review comments
   - Submit reviews after completed appointments

### 2.2.2 Doctor Functions

1. **Account Management**
   - Register with professional details (name, specialty, certificates)
   - Login securely
   - Update location and contact information
   - Manage working hours and consultation fees
2. **Appointment Management**
   - View upcoming appointments dashboard
   - See patient details for scheduled appointments
   - Cancel appointments with reason notification
   - Track appointment statistics
3. **Profile Management**
   - View and update professional information
   - Monitor average rating and total reviews

### 2.2.3 Laboratory Functions

1. **Account Management**
   - Register laboratory with facility details
   - Login to secure dashboard
   - Update location and contact information
2. **Profile Management**
   - View laboratory information
   - Monitor ratings and reviews

## 2.3 User Characteristics

**Patients:**
Patients usually have only basic digital literacy to navigate web applications, create accounts, book appointments, and receive notifications.

**Doctors:**
Doctors possess moderate to advanced technical skills. They are familiar with managing digital records, using scheduling tools, and handling medical data. Their capabilities include updating consultation notes, prescribing digitally, and managing patient history efficiently. They require reliable dashboards .

**Laboratories:**
Laboratory staff are accustomed to working with medical information systems and data entry tools.

**System Administrators:**
System administrators have technical expertise in databases, security, and system performance. They are capable of maintaining servers, ensuring data integrity, handling user accounts, and monitoring the system's compliance with data protection laws.

## 2.4 Constraints

### 2.4.1 Regulatory Requirements

- Must comply with Moroccan Data Protection Law (Law 09-08)
- Must adhere to GDPR standards for data privacy
- Must implement secure storage and transmission of medical information
- Must provide data deletion capabilities upon user request

### 2.4.2 Technical Constraints

- **Platform:** Web-based application (no native mobile apps in initial version)
- **Database:** Relational database required (H2 for development, PostgreSQL/MySQL for production)
- **Framework:** Spring Boot 3.x with Java 17+
- **Browser Support:** Modern browsers (Chrome 90+, Firefox 88+, Safari 14+, Edge 90+)
- **Network:** Stable internet connection required for all operations
- **Security:** HTTPS mandatory for all communications

### 2.4.3 Development Standards

- Follow Spring Boot best practices and conventions
- Use Spring Security for authentication and authorization
- Follow MVC architecture pattern
- Use Lombok for reducing boilerplate code
- Implement proper exception handling
- Write clean, documented, and maintainable code

### 2.4.4 Performance Constraints

- Support minimum 1,000 concurrent users
- Page load time under 3 seconds
- Database query response time under 1 second
- Session timeout after 30 minutes of inactivity

## 2.5 Assumptions and Dependencies

### 2.5.1 Assumptions

1. Users have access to devices with web browsers and internet connectivity
2. Doctors and laboratories will provide accurate and up-to-date information
3. Email addresses provided by users are valid and regularly checked
4. Users have basic digital literacy to navigate web applications
5. Healthcare providers will honor appointments booked through the system
6. The application will be hosted on reliable infrastructure with 99% uptime

### 2.5.2 Dependencies

1. **Java Runtime Environment:** JDK 17 or higher
2. **Spring Boot Framework:** Version 3.x
3. **Database:** H2 (development) or PostgreSQL/MySQL (production)
4. **Web Server:** Apache Tomcat (embedded in Spring Boot)
5. **Internet Connection:** Reliable broadband for server hosting
6. **SSL Certificate:** For HTTPS encryption
7. **Healthcare Provider Participation:** Doctors and laboratories must register and maintain profiles

# 3 Specific Requirements

## 3.1 Functional Requirements

### 3.1.1 User Registration and Authentication

**REQ-AUTH-001: User Registration**

*Description:* The system shall allow users to register as Patient, Doctor, or Laboratory.
*Inputs:*

- User type selection (PATIENT, DOCTOR, LABORATORY)
- Email address
- Password
- Role-specific information

*Processing:*

- Validate email format and uniqueness
- Encrypt password using BCrypt
- Store user credentials in database
- Create associated profile (Patient, Doctor, or Laboratory entity)

*Outputs:*

- Success message and redirect to login page
- Error message if email already exists

**REQ-AUTH-002: Patient Registration Details**

*Description:* When registering as a patient, the system shall collect specific information.
*Required Fields:*

- Full name
- Date of birth
- Gender (Male/Female)
- ID number
- Phone number
- Username

**REQ-AUTH-003: Doctor Registration Details**

*Description:* When registering as a doctor, the system shall collect professional information.
*Required Fields:*

- Doctor name
- Medical specialty
- Location (city)

- Phone number
- Working hours
- Certificates
- Consultation fee (in MAD)

**REQ-AUTH-004: Laboratory Registration Details**
*Description:* When registering as a laboratory, the system shall collect facility information.
*Required Fields:*

- Laboratory name
- Location (city)
- Phone number
- Working hours

**REQ-AUTH-005: User Login**
*Description:* The system shall authenticate users using email and password.
*Inputs:*

- Email address
- Password

*Processing:*

- Validate credentials against database
- Check if account is locked
- Create secure session
- Redirect based on user role

*Outputs:*

- Redirect to appropriate dashboard on success
- Error message on invalid credentials
- Account locked message after 5 failed attempts

**REQ-AUTH-006: Failed Login Tracking**
*Description:* The system shall track failed login attempts and lock accounts after 5 consecutive failures.
*Processing:*

- Increment failed login counter on each failed attempt
- Lock account when counter reaches 5
- Reset counter on successful login

**REQ-AUTH-007: Logout**
*Description:* The system shall allow users to securely logout.
*Processing:*

- Invalidate current session
- Delete session cookies
- Redirect to home page

### 3.1.2 Doctor Search and Discovery

**REQ-SEARCH-001: Doctor Search by Name**
*Description:* The system shall allow patients to search for doctors by name.
*Inputs:*

- Search query (doctor name or partial name)

*Processing:*

- Perform case-insensitive search in doctor names
- Return matching doctors ordered by average rating

*Outputs:*

- List of doctors matching search criteria
- Count of results found

**REQ-SEARCH-002: Filter by Specialty**
*Description:* The system shall allow filtering doctors by medical specialty.
*Available Specialties:*

- Cardiology
- Dermatology
- Pediatrics
- Orthopedics
- Neurology
- General Practice

**REQ-SEARCH-003: Filter by Location**
*Description:* The system shall allow filtering doctors by city location.
*Available Locations:*

- Casablanca, Rabat, Marrakech, Tangier, Agadir, Laayoune, Guelmim, Ben Guerir

**REQ-SEARCH-004: Filter by Price Range**
*Description:* The system shall allow filtering doctors by consultation fee range.
*Inputs:*

- Minimum price (optional)

- Maximum price (optional)

*Processing:*

- Filter doctors where consultation fee is within specified range

**REQ-SEARCH-005: Multiple Filters**
*Description:* The system shall support applying multiple search filters simultaneously.
*Example:*

- Cardiology + Casablanca + Price 300-500 MAD

**REQ-SEARCH-006: Doctor Profile Display**
*Description:* The system shall display comprehensive doctor information in search results.
*Information Displayed:*

- Doctor name
- Medical specialty
- Location (city)
- Average rating (stars)
- Total number of reviews
- Consultation fee (in MAD)

**REQ-SEARCH-007: Doctor Profile Details**
*Description:* The system shall provide detailed doctor profile pages.
*Information Displayed:*

- Full professional information
- Phone number
- Working hours
- Certificates and qualifications
- Average rating with star visualization
- List of patient reviews with ratings and comments
- "Book Appointment" button

### 3.1.3 Appointment Management

**REQ-APPT-001: Appointment Booking**
*Description:* The system shall allow patients to book appointments with doctors.
*Inputs:*

- Selected doctor
- Appointment date (current date or future)
- Appointment time (from available slots)

*Processing:*

- Validate date is not in the past
- Check for time slot conflicts
- Create appointment record with status SCHEDULED

*Outputs:*

- Error if time slot already booked

**REQ-APPT-002: Available Time Slots**
*Description:* The system shall provide predefined time slots for appointment booking.
*Available Time Slots:*

- Morning: 09:00, 09:30, 10:00, 10:30, 11:00, 11:30
- Afternoon: 14:00, 14:30, 15:00, 15:30, 16:00, 16:30

**REQ-APPT-003: Conflict Detection**
*Description:* The system shall prevent double-booking of appointment slots.
*Processing:*

- Check database for existing appointments with same doctor, date, and time
- Block booking if conflict exists
- Show error message to user

**REQ-APPT-004: Appointment Cancellation by Patient**
*Description:* The system shall allow patients to cancel their scheduled appointments.
*Processing:*

- Update appointment status to CANCELLED_BY_PATIENT
- Remove from upcoming appointments list

**REQ-APPT-005: Appointment Cancellation by Doctor**
*Description:* The system shall allow doctors to cancel appointments with reason.
*Inputs:*

- Appointment ID
- Cancellation reason (required text field)

*Processing:*

- Update appointment status to CANCELLED_BY_DOCTOR
- Store cancellation reason

**REQ-APPT-006: Upcoming Appointments View**
*Description:* The system shall display upcoming appointments for patients and doctors.
*Information Displayed:*

- For Patients: Doctor name, specialty, location, date, time, fee, status
- For Doctors: Patient name, phone number, date, time, status

*Sorting:* Chronologically ordered by date and time (earliest first)

**REQ-APPT-007: Appointment History**
*Description:* The system shall maintain complete appointment history for patients.
*Statuses Included:*

- COMPLETED
- CANCELLED_BY_PATIENT
- CANCELLED_BY_DOCTOR
- NO_SHOW

*Sorting:* Reverse chronologically (most recent first)

**REQ-APPT-008: Automatic Appointment Completion**
*Description:* The system shall automatically mark appointments as completed 2 hours after scheduled time.
*Processing:*

- Scheduled task runs every hour (cron: 0 0 * * * *)
- Find all SCHEDULED appointments older than 2 hours
- Update status to COMPLETED

**REQ-APPT-009: Appointment Reminder Email** *Description:* The system shall send email reminders to patients about upcoming appointments. *Email Content:*

- Doctor name and specialty
- Appointment date and time
- Doctor location and phone number
- Consultation fee

### 3.1.4 Review and Rating System

**REQ-REVIEW-001: Submit Doctor Review** *Description:* The system shall allow patients to review doctors after completed appointments. *Inputs:*

- Doctor ID
- Rating (1-5 stars, required)
- Comment text (required, maximum 1000 characters)

*Processing:*

- Create review record in database
- Associate with patient and doctor

- Timestamp the review
- Recalculate doctor's average rating
- Update doctor's total review count

**REQ-REVIEW-002: Rating Calculation** *Description:* The system shall automatically calculate and update average ratings. *Processing:*

- Query all reviews for the doctor
- Calculate arithmetic mean of all ratings
- Round to one decimal place
- Update doctor.averageRating field
- Update doctor.totalReviews count

**REQ-REVIEW-003: Display Reviews** *Description:* The system shall display reviews on doctor profile pages. *Information Displayed:*

- Patient full name (reviewer)
- Rating (star visualization)
- Comment text

*Sorting:* Most recent reviews first

### 3.1.5 Account Management

**REQ-ACCOUNT-001: Update Email** *Description:* The system shall allow users to update their email address. *Inputs:*

- New email address

*Processing:*

- Check email uniqueness
- Update user.email field
- Maintain session

**REQ-ACCOUNT-002: Change Password** *Description:* The system shall allow users to change their password. *Inputs:*

- New password (minimum 6 characters)
- Password confirmation

*Processing:*

- Validate password match
- Encrypt using BCrypt

- Update user.password field

**REQ-ACCOUNT-003: Update Doctor Location** *Description:* The system shall allow doctors to update their clinic location. *Inputs:*

- New location (city selection)

*Processing:*

- Update doctor.location field
- Refresh profile information

**REQ-ACCOUNT-004: Delete Account** *Description:* The system shall allow users to permanently delete their accounts. *Processing:*

- Display confirmation dialog
- Delete user record (cascades to related entities)
- Cancel all future appointments
- Invalidate session
- Redirect to logout

*Warning:* Must inform user this action is irreversible

### 3.1.6 Dashboard Features

**REQ-DASH-001: Patient Dashboard** *Description:* The system shall provide a comprehensive dashboard for patients. *Components:*

- Welcome message with patient name
- Upcoming appointments section (with cancel buttons)
- Appointment history section
- Quick link to find doctors
- Success/error alerts for recent actions

**REQ-DASH-002: Doctor Dashboard** *Description:* The system shall provide a comprehensive dashboard for doctors. *Components:*

- Welcome message with doctor name
- Statistics cards (upcoming appointments, total reviews, average rating)
- Upcoming appointments list with patient details
- Recent reviews section
- Appointment cancellation forms with reason field

**REQ-DASH-003: Laboratory Dashboard** *Description:* The system shall provide a comprehensive dashboard for laboratories. *Components:*

- Laboratory name and information
- Contact information display

## 3.2 Non-Functional Requirements

### 3.2.1 Performance Requirements

**REQ-PERF-001: Response Time** *Description:* The system shall provide fast response times for user interactions. *Requirements:*

- Page load time: less than 3 seconds
- Database queries: less than 1 second
- Search results: less than 2 seconds

**REQ-PERF-002: Concurrent Users** *Description:* The system shall support multiple simultaneous users. *Requirement:* Minimum 1,000 concurrent users without performance degradation
**REQ-PERF-003: Database Performance** *Description:* The system shall optimize database operations. *Requirements:*

- Use connection pooling
- Implement proper indexing on frequently queried fields
- Optimize JPA queries to avoid N+1 problems

### 3.2.2 Security Requirements

**REQ-SEC-001: Password Encryption** *Description:* The system shall encrypt all passwords using BCrypt. *Requirements:*

- BCrypt strength factor: 10 or higher
- Never store plain-text passwords
- Hash passwords before database storage

**REQ-SEC-002: HTTPS Communication** *Description:* All communications shall be encrypted using HTTPS/TLS. *Requirements:*

- SSL/TLS certificate required
- Redirect HTTP to HTTPS
- Use TLS 1.2 or higher

**REQ-SEC-003: Session Management** *Description:* The system shall implement secure session management. *Requirements:*

- Session timeout after 30 minutes of inactivity
- Invalidate sessions on logout

**REQ-SEC-004: Role-Based Access Control** *Description:* The system shall enforce role-based access control. *Requirements:*

- Patient URLs: accessible only to PATIENT role
- Doctor URLs: accessible only to DOCTOR role
- Laboratory URLs: accessible only to LABORATORY role

**REQ-SEC-005: SQL Injection Prevention** *Description:* The system shall prevent SQL injection attacks. *Implementation:*

- Use JPA with prepared statements
- Parameterize all database queries
- Validate and sanitize user inputs

### 3.2.3 Reliability Requirements

**REQ-REL-001: System Availability** *Description:* The system shall maintain high availability. *Requirement:* 99 **REQ-REL-002: Data Backup** *Description:* The system shall perform regular data backups. *Requirements:*

- Daily automated backups
- Backup retention: minimum 30 days
- Backup verification procedures
- Disaster recovery plan

**REQ-REL-003: Error Handling** *Description:* The system shall handle errors gracefully. *Requirements:*

- Display user-friendly error messages
- Log technical details for debugging
- Prevent system crashes
- Provide error recovery options

### 3.2.4 Usability Requirements

**REQ-USE-001: User Interface Design** *Description:* The system shall provide an intuitive user interface. *Requirements:*

- Consistent color scheme (green theme)
- Clear navigation menus
- Responsive design for mobile and desktop
- Form validation with helpful error messages
- Confirmation dialogs for critical actions

**REQ-USE-002: Accessibility** *Description:* The system shall be accessible to users with varying abilities. *Requirements:*

- Readable fonts and sufficient contrast
- Logical tab order for keyboard navigation
- Descriptive labels for form fields
- Clear feedback for user actions

### 3.2.5 Maintainability Requirements

**REQ-MAINT-001: Code Quality** *Description:* The system shall maintain high code quality standards. *Requirements:*

- Follow Spring Boot best practices
- Use Lombok to reduce boilerplate
- Implement proper exception handling
- Maintain separation of concerns (MVC pattern)

**REQ-MAINT-002: Documentation** *Description:* The system shall include comprehensive documentation. *Requirements:*

- JavaDoc comments for complex methods
- README with setup instructions
- Database schema documentation
- API endpoint documentation
- Deployment guide

**REQ-MAINT-003: Logging** *Description:* The system shall implement comprehensive logging. *Requirements:*

- Log levels: DEBUG, INFO, WARN, ERROR
- Log authentication attempts
- Log critical operations
- Include timestamps and user context

### 3.2.6 Data Requirements

**REQ-DATA-001: Data Integrity** *Description:* The system shall maintain data integrity. *Requirements:*

- Foreign key constraints
- Not-null constraints for required fields
- Unique constraints for emails and usernames
- Cascading deletes for related data

**REQ-DATA-002: Data Privacy** *Description:* The system shall protect user privacy. *Requirements:*

- Allow users to delete their data
- Encrypt sensitive data in transit and at rest
- Limit data access based on roles

**REQ-DATA-003: Data Retention** *Description:* The system shall define data retention policies. *Requirements:*

- Keep appointment history for minimum 1 year
- Archive old records after 2 years
- Delete user data upon account deletion

# 4 System Models

## 4.1 Data Model

The MyWelly system uses a relational database with the following key entities:

### 4.1.1 Entity Descriptions

**User**

- Primary entity for authentication
- Fields: id, email, password (encrypted), userRole, createdAt, failedLoginAttempts, accountLocked
- Relationships: OneToOne with Patient, Doctor, or Laboratory

**Patient**

- Stores patient-specific information
- Fields: id, userId, fullName, dateOfBirth, gender, idNumber, phoneNumber, username, noShowCount, isBlocked
- Relationships: OneToMany with Appointment, OneToMany with Review

**Doctor**

- Stores doctor-specific information
- Fields: id, userId, name, specialty, location, phoneNumber, workingHours, certificates, consultationFee, averageRating, totalReviews
- Relationships: OneToMany with Appointment, OneToMany with Review

**Laboratory**

- Stores laboratory information
- Fields: id, userId, labName, location, phoneNumber, workingHours.

**Appointment**

- Stores appointment records
- Fields: id, patientId, doctorId, appointmentDate, appointmentTime, status, cancellationReason, createdAt
- Status values: SCHEDULED, COMPLETED, CANCELLED$_BY_PATIENT$, $CANCELLED_BY_DOCTO$
- Relationships: ManyToOne with Patient, ManyToOne with Doctor

**Review**

- Stores patient reviews
- Fields: id, patientId, doctorId, laboratoryId, rating, comment
- Constraints: rating between 1 and 5
- Relationships: ManyToOne with Patient, ManyToOne with Doctor

## 4.2 Use Case Diagrams

The system supports the following primary use cases:

### 4.2.1 Sign Up Use Case

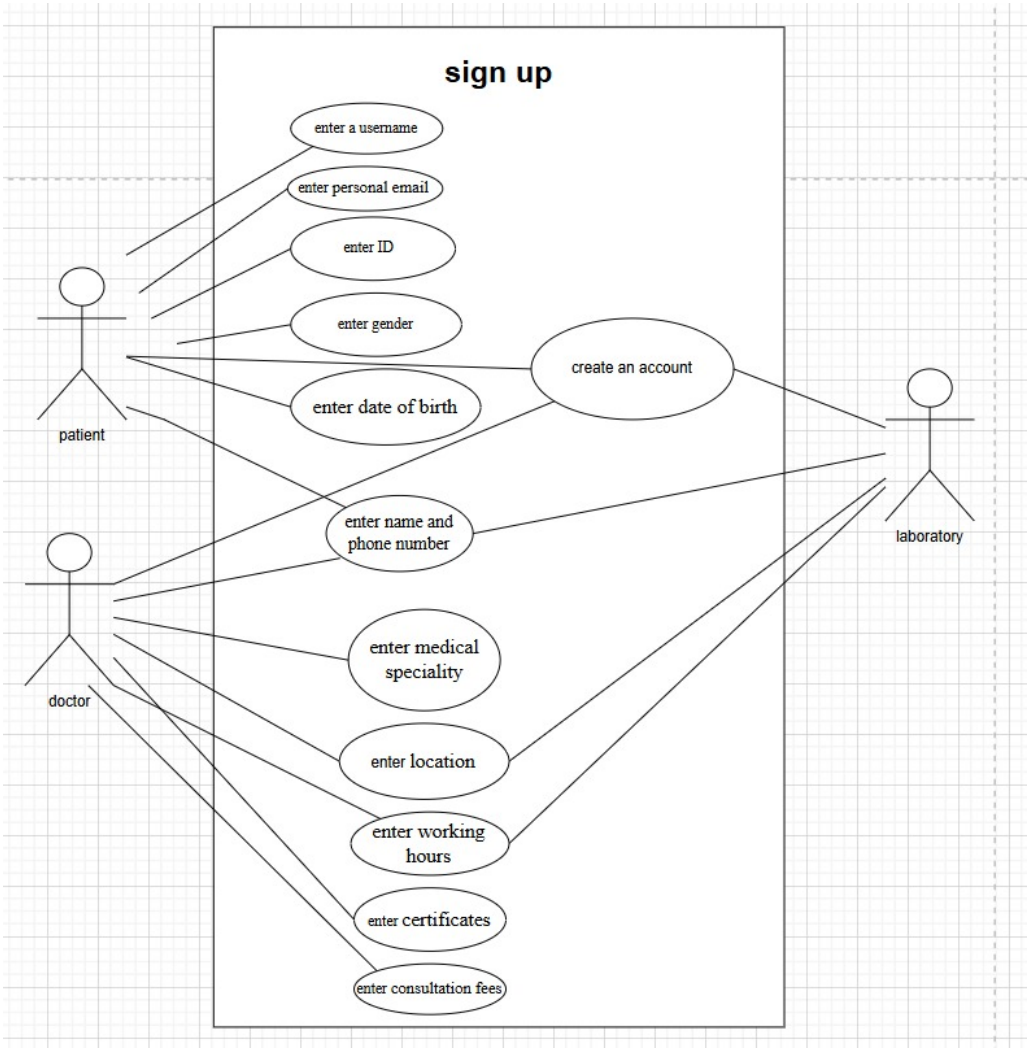


Figure 1: Sign Up Use Case Diagram

The sign-up process allows three types of users (patients, doctors, and laboratories) to register on the platform with their specific required information.

### 4.2.2 Login Use Case

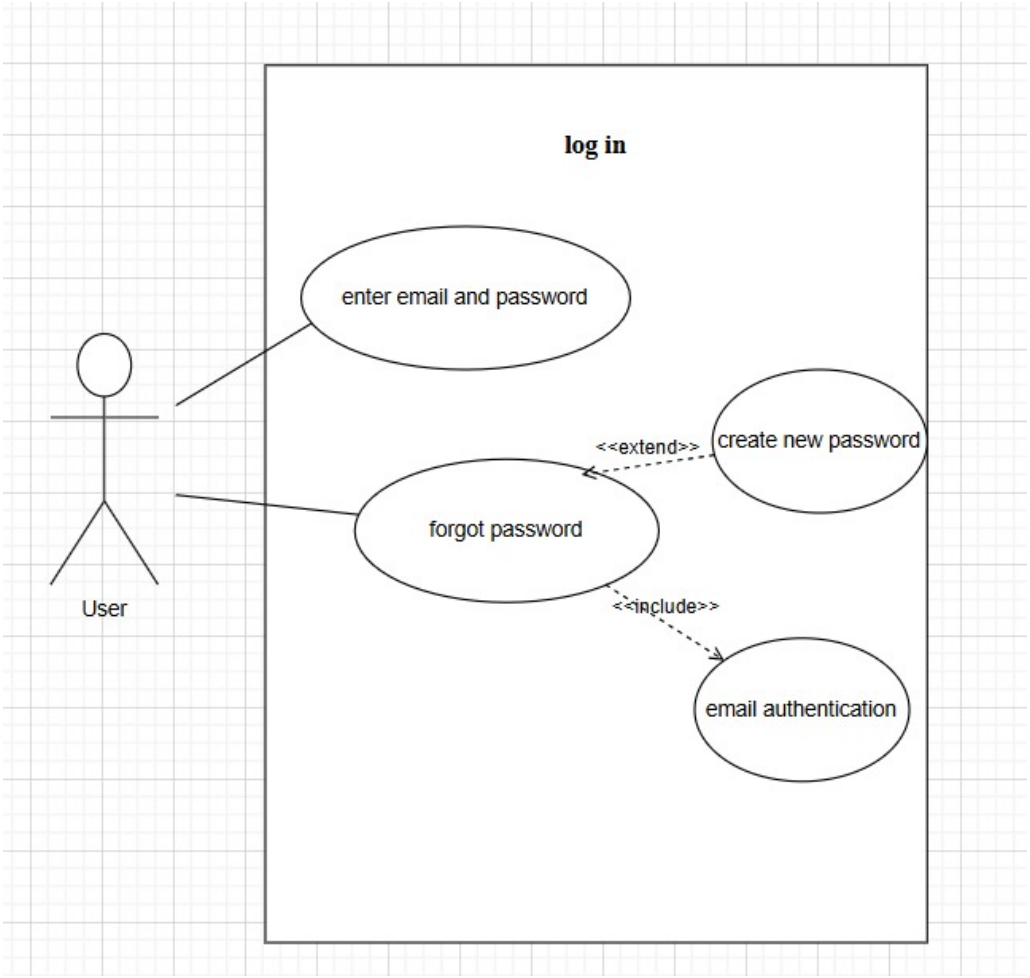


Figure 2: Login Use Case Diagram

Users can securely log in using their email and password, with a "forgot password" recovery option available.

### 4.2.3 Account Management Use Case

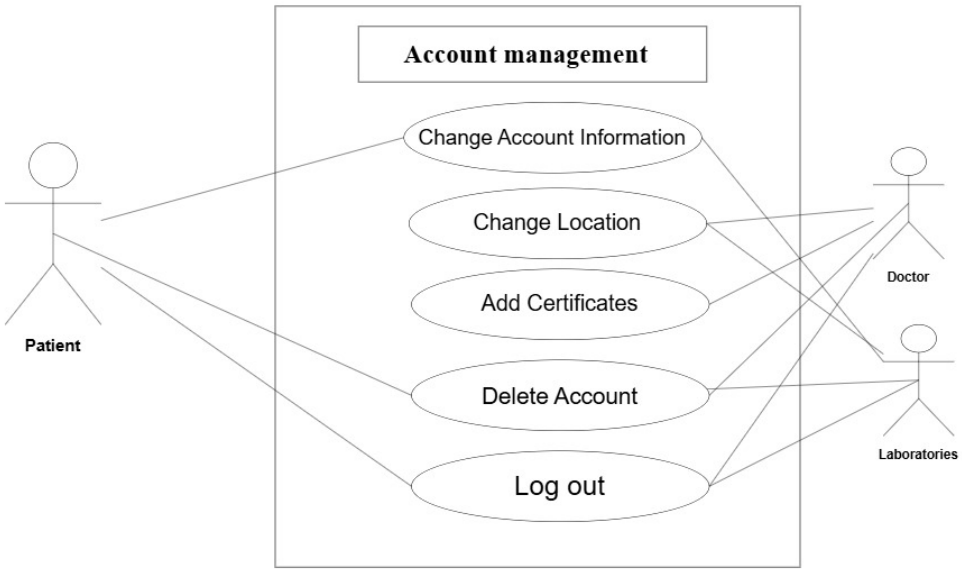


Figure 3: Account Management Use Case Diagram

All user types can manage their account information, including changing passwords, updating locations, and deleting accounts.

### 4.2.4 Browse and Search Use Case

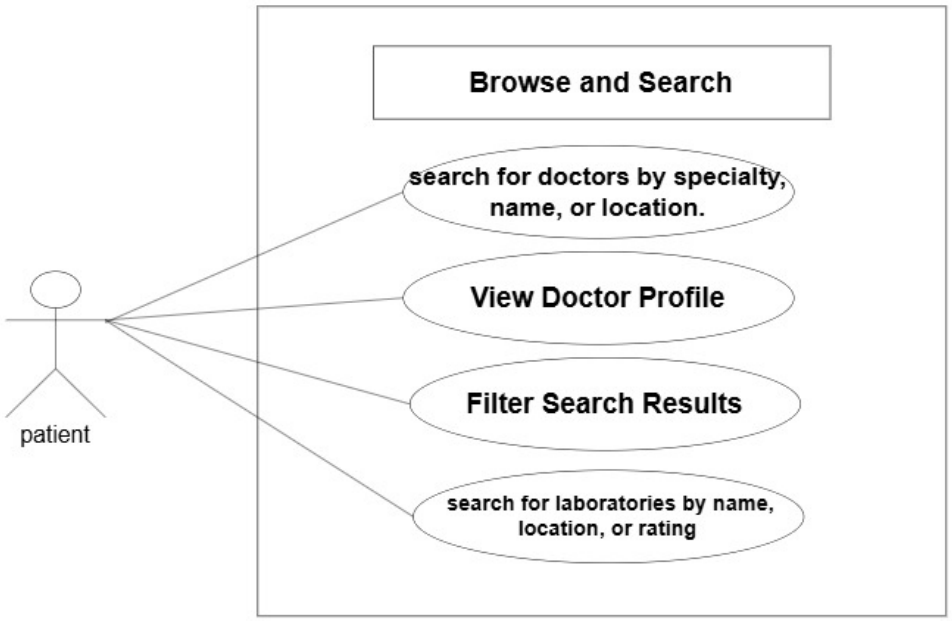


Figure 4: Browse and Search Use Case Diagram

Patients can search for doctors by specialty, name, or location, view detailed profiles, and filter search results. They can also search for laboratories.

### 4.2.5 Appointment Management Use Case

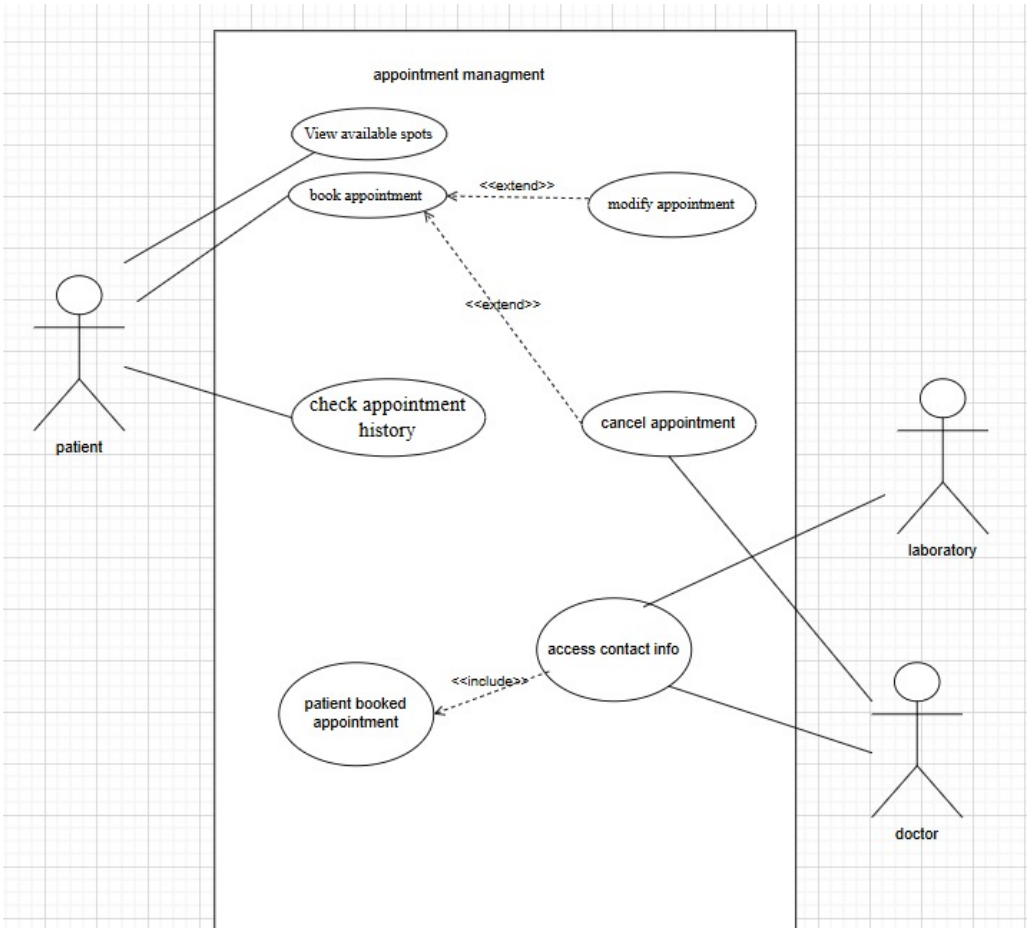


Figure 5: Appointment Management Use Case Diagram

Comprehensive appointment management features include booking, modifying, canceling appointments, viewing history, and automated notifications for all parties involved.

### 4.2.6 Reviews Use Case

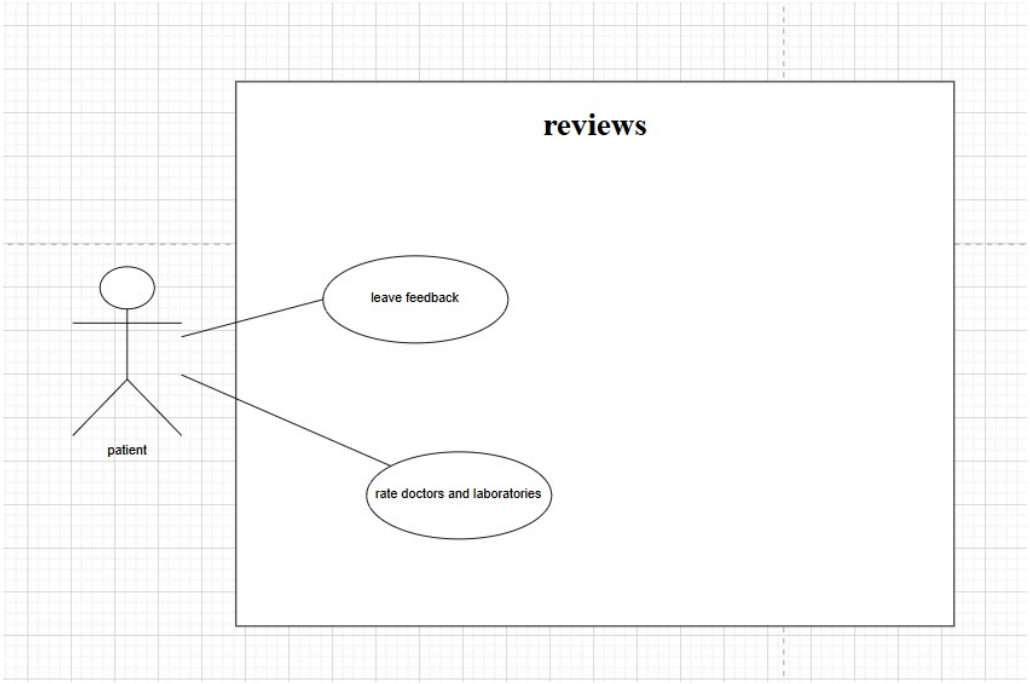


Figure 6: Reviews Use Case Diagram

Patients can leave feedback and rate doctors and laboratories after their appointments using a 5-star rating system.

### 4.2.7 Patient Use Cases Summary

1. Sign up as patient
2. Login to system
3. Search for doctors
4. View doctor profile
5. Book appointment
6. Cancel appointment
7. View appointment history
8. Submit review for doctor
9. Update email and password
10. Delete account

### 4.2.8 Doctor Use Cases Summary

1. Sign up as doctor
2. Login to system
3. View dashboard with statistics
4. View upcoming appointments
5. Cancel appointment with reason
6. View patient reviews
7. Update location
8. Update email and password

### 4.2.9 Laboratory Use Cases Summary

1. Sign up as laboratory
2. Login to system
3. View dashboard
4. View reviews
5. Update contact information

# 5 Appendices

## 5.1 Technology Stack

| Component | Technology |
|---|---|
| Backend Framework | Spring Boot 3.x |
| Programming Language | Java 17+ |
| Security | Spring Security with BCrypt |
| Database (Prod) | H2 (file-based) |
| Database (Dev) | PostgreSQL or MySQL |
| ORM | Spring Data JPA / Hibernate |
| Template Engine | Thymeleaf |
| Web Server | Apache Tomcat (embedded) |
| Build Tool | Maven or Gradle |
| Code Reduction | Lombok |
| Scheduling | Spring @Scheduled |

Table 1: Technology Stack

## 5.2 Database Schema

**Tables:**

- users
- patients
- doctors
- laboratories
- appointments
- reviews

**Key Indexes:**

- users.email (unique)
- patients.username (unique)
- doctors.specialty, doctors.location
- appointments (patientId, doctorId, date, time, status)
- reviews (doctorId, laboratoryId)

## 5.3 Future Enhancements

Potential features for future versions:

1. Telemedicine and video consultations
2. Online payment processing
3. Prescription management system

4. Laboratory test result uploads
5. Mobile native applications (Android/iOS)
6. Medication reminders
7. Health tracking and analytics
8. Insurance integration
9. Multi-language support (Arabic, French)
10. Advanced search filters
11. Doctor availability calendar
12. SMS notifications
13. Patient medical records storage
14. Integration with national health systems

# 6 Glossary

- **BCrypt:** A password hashing function designed for secure password storage
- **Cascading Delete:** Automatic deletion of related records when a parent record is deleted
- **CSRF:** Cross-Site Request Forgery, a type of web security attack
- **DTO:** Data Transfer Object, used to transfer data between layers
- **Entity:** A Java class mapped to a database table using JPA
- **HTTPS:** HTTP Secure, encrypted version of HTTP protocol
- **JPA:** Java Persistence API, specification for ORM in Java
- **Lombok:** Java library that reduces boilerplate code
- **MVC:** Model-View-Controller, software architectural pattern
- **ORM:** Object-Relational Mapping, technique for database interaction
- **Repository:** Spring Data interface for database operations
- **REST:** Representational State Transfer, architectural style for APIs
- **Service Layer:** Business logic layer in Spring applications
- **Spring Boot:** Java framework for building enterprise applications
- **Spring Security:** Framework for authentication and authorization
- **Thymeleaf:** Server-side Java template engine for HTML
- **XSS:** Cross-Site Scripting, a type of web security vulnerability


*Transforming Healthcare in Morocco*