# Software Requirements Specification

for
Private Clinic Management System
Version 1.0 approved

Prepared by Panayiotis Christodoulou
Cyprus University of Technology
May 2025

# Table of Contents

# 1. Introduction

## 1.1 Purpose

This Software Requirements Specification (SRS) defines the requirements for the Private Clinic Management System. This local-only system is intended for use by a single doctor in a private practice and is developed as part of a final year thesis project. The system provides functionalities for managing patient records, medical histories, appointment scheduling, and medical file uploads.

## 1.2 Document Conventions

Requirements in this document are written using standardized language. The words 'must', 'shall', and 'required' indicate mandatory features. 'Should' implies a desirable feature, and 'may' indicates optional behavior. Headings follow IEEE SRS formatting.

## 1.3 Intended Audience and Reading Suggestions

This document is intended for:
- Academic supervisors and evaluators
- Developers or maintainers of the system
- Testers responsible for validation
- End-users for overview of the system features

### 1.4 Project Scope

The Private Clinic Management System is a web application aimed at simplifying day-to-day operations in a small private clinic. The doctor can manage patient details, medical files, and appointments. Patients can register, update their personal data, and request appointments. Uploaded test results (PDF, PNG, JPG) are associated with patient profiles for medical reference.

### 1.5 References

No external documents are referenced. This system is self-contained, designed for educational purposes and local-only deployment.

## 2. Overall Description

### 2.1 Product Perspective

This system is a standalone web-based product, not dependent on or integrated with any external system. It is developed from scratch using PHP, MySQL, HTML, and CSS. It operates in a local environment such as XAMPP or LAMP stack.

### 2.2 Product Features

- Doctor account for full system access and management
- Patient registration and login
- Patient profile management
- Appointment scheduling (by patient)
- Appointment management (by doctor)
- Upload and storage of examination files (PDF, PNG, JPG)
- Password recovery via PHPMailer

### 2.3 User Classes and Characteristics

There are two user classes:
- Doctor: Has full access to all system features including patient data, appointments, and uploads.
- Patient: Limited access to own profile, appointment booking, and personal data only. Must be registered and logged in.

### 2.4 Operating Environment

The system is designed to run locally under:
- Apache Web Server (via XAMPP)
- MySQL/MariaDB database
- PHP 8.x
- Modern web browser (Chrome, Firefox, Edge)

## 2.5 Design and Implementation Constraints

- Local-only architecture, no public network access
- Developed solely using open-source technologies
- Designed to support only one doctor (single admin role)
- Session-based role validation for access control
- File uploads limited to specific formats for safety

## 2.6 User Documentation

The system includes:

- Online form instructions and field validations
- Separate user manual for doctor and patient provided as documentation

## 2.7 Assumptions and Dependencies

- The system assumes the hosting machine has a working local server stack (e.g., XAMPP)
- PHPMailer must be configured for password recovery to function
- Users will enter valid data as expected; basic client-side validation is implemented

# 3. System Features

## 3.1 User Registration and Login

### 3.1.1 Description and Priority

Allows patients to register and log into the system. Priority: High.

### 3.1.2 Stimulus/Response Sequences

User submits registration form → System validates input → User account is created.

### 3.1.3 Functional Requirements

REQ-1: The system shall allow new users to register with a unique email address.
REQ-2: The system shall hash and store passwords securely.
REQ-3: The system shall authenticate users and redirect them to the appropriate dashboard.

## 3.2 Appointment Booking

### 3.2.1 Description and Priority

Patients can request appointments with the doctor. Priority: High.

### 3.2.2 Stimulus/Response Sequences

Patient selects a date and submits a request → System records appointment.

### 3.2.3 Functional Requirements

REQ-4: The system shall allow patients to select available dates for appointments.
REQ-5: The system shall notify the doctor of new appointments.
REQ-6: The doctor shall be able to cancel or reschedule appointments.

## 3.3 Patient Record Management

### 3.3.1 Description and Priority

Doctors manage patients' personal data and medical history. Priority: High.

### 3.3.2 Stimulus/Response Sequences

Doctor accesses patient profile → Doctor adds or updates medical notes.

### 3.3.3 Functional Requirements

REQ-7: The doctor shall have access to the full profile of registered patients.
REQ-8: The doctor shall be able to edit and update patient medical records.

## 3.4 File Upload for Examinations

### 3.4.1 Description and Priority

Doctors can upload medical files (test results) linked to each patient. Priority: Medium.

#### 3.4.2 Stimulus/Response Sequences

Doctor selects a patient and uploads a file → File is stored and linked to the patient.

#### 3.4.3 Functional Requirements

REQ-9: The system shall allow the upload of PDF, PNG, and JPG files.
REQ-10: The uploaded files shall be accessible only by the doctor.
REQ-11: Files shall be linked to the appropriate patient profile.

## 4. External Interface Requirements

### 4.1 User Interfaces

The user interface is web-based and accessible via browser. It includes:
- Login and registration forms with input validation
- Patient dashboard: view/edit profile, view appointments
- Doctor dashboard: manage appointments, access patient data
- Upload interface for examination files

All forms provide basic feedback on errors and confirmations. Interface is mobile-friendly and minimalistic.

### 4.2 Hardware Interfaces

There are no specialized hardware interface requirements. The system runs on a standard PC with a local server stack (e.g., XAMPP).

### 4.3 Software Interfaces

- PHP 8.x: Used for server-side logic
- MySQL/MariaDB: Database storage for users, appointments, uploads
- PHPMailer: Handles email sending for password reset
- HTML/CSS/JS: Frontend presentation layer

The PHP files interface with the MySQL database using mysqli for CRUD operations.

### 4.4 Communications Interfaces

The system does not require internet connectivity for regular use.
Communication occurs locally between the web server and database. Email reset functionality depends on external SMTP configuration in PHPMailer.

## 5. Other Nonfunctional Requirements

### 5.1 Performance Requirements

The system should respond to user actions (form submissions, page navigation) within 2-3 seconds in a local environment. Data queries should execute in under 1 second on a modestly sized dataset.

### 5.2 Safety Requirements

As the system runs locally and does not store critical medical data such as diagnoses or prescriptions, safety risks are minimal. File types are restricted to prevent malicious uploads, and error messages are user-friendly.

### 5.3 Security Requirements

- Passwords are hashed before storage using `password_hash()`
- Session-based authentication is used for access control
- Uploaded files are validated for type and size
- PHPMailer should use TLS if SMTP is configured
- User data is only accessible to the corresponding user and doctor

### 5.4 Software Quality Attributes

- Usability: Simple, intuitive UI for both roles
- Maintainability: Modular PHP files with reusable code blocks
- Portability: Can run on any system with PHP and MySQL
- Reliability: Expected to function continuously in local use
- Extensibility: Future versions may include multiple doctors, specialty filters, etc.

## 6. Other Requirements

- All content is in English; UTF-8 encoding is used throughout
- The system should be deployable without requiring internet
- Manual installation guide will be included
- All database tables will follow normalization standards up to 3NF

## Appendix A: Glossary

- CRUD: Create, Read, Update, Delete
- ERD: Entity-Relationship Diagram
- DFD: Data Flow Diagram
- UI: User Interface
- SRS: Software Requirements Specification

## Appendix B: Analysis Models

See ERD and DFD diagrams in the Design Document.

## Appendix C: Issues List

No major open issues at the time of submission. All TBD items have been resolved.