Batch: A1	**Roll No.: 16010123012**
Experiment / assignment / tutorial No.: 01	

TITLE: Requirement **S**pecification Document

AIM: To learn and understand the way of analysing the gathered information in the previous phase for the development process and prepare requirement specification document. A concept of software engineering.

Expected Course outcome of Experiment:

Process of gathering requirements and converting them into specifications.
Document created will be used by both, the customer and the developer, to understand WHAT is going to be developed.

Books/ Journals/ Websites referred:

1. Roger Pressman, Software Engineering: A practitioners Approach, McGraw Hill, 2010 ,6th edition
2. Ian Somerville, Software Engineering , Addison Wesley,2011,9th edition
3 http://en.wikipedia.org/wiki/Software_requirements_specification

---

Pre Lab/ Prior Concepts:

Requirements analysis in systems engineering and software engineering, encompasses those tasks that go into determining the needs or conditions to meet for a new or altered product, taking account of the possibly conflicting requirements of the various stakeholders, such as beneficiaries or users. It is an early stage in the more general activity of requirements engineering which encompasses all activities concerned with eliciting, analyzing, documenting, validating and managing software or system requirements.

Requirements analysis is critical to the success of a systems or software project. The requirements should be documented, actionable, measurable, testable, traceable, related to identified business needs or opportunities, and defined to a level of detail sufficient for system design.

Conceptually, requirements analysis includes three types of activities:

- **Eliciting requirements**: the task of identifying the various types of requirements from various sources including project documentation, (e.g. the project charter or definition), business process documentation, and stakeholder interviews. This is sometimes also called requirements gathering.
- **Analysing requirements**: determining whether the stated requirements are clear, complete, consistent and unambiguous, and resolving any apparent conflicts.
- Recording requirements: Requirements may be documented in various forms, usually including a summary list and may include natural-language documents, use cases or process specifications.

New systems change the environment and relationships between people, so it is important to identify all the stakeholders, taken into account all their needs and ensure they understand the implications of the new systems. Analysts can employ several techniques to elicit the requirements from the customer. These may include the development of scenarios, the identification of use cases, the use of workplace observation or ethnography, holding interviews, or focus groups (more aptly named in this context as requirements workshops, or requirements review sessions) and creating requirements lists. Prototyping may be used to develop an example system that can be demonstrated to stakeholders. Where necessary, the analyst will employ a combination of these methods to establish the exact requirements of the stakeholders, so that a system that meets the business needs is produced

Different types of Requirements

- Functional requirements
- Usability requirements
- Reliability requirements
- Performance requirements
- Security requirements

A typical SRS document template is shared subsequently. This document acts as a reference and will be used by both, the customer (for whom the software system is to be developed), and the organization which develops the solution. Typically, prepared by the development organization at the early stage of development by the professionals after interacting with the customer.

Software Requirements Specification for:

MediSlot

Version 1.0

Prepared by Aaryan Sharma, Aditey Kshirsagar, Aditya Baheti

K.J Somaiya School of Engineering

23rd July, 2025

Table of Contents

Introduction

Purpose

MediSlot is designed to streamline and automate the process of booking, managing, and monitoring outpatient medical appointments in clinics and hospitals. This document covers the core functionality of the appointment scheduling module, including patient booking, doctor availability management, and admin control.

Product Scope

MediSlot is a web-based medical appointment scheduling system designed to simplify and optimize the process of booking, rescheduling, and managing appointments in hospitals and clinics. It provides dedicated interfaces for patients and doctor offering role-based access and real-time updates on appointment availability.
The primary goal of MediSlot is to reduce manual effort, eliminate booking conflicts, and improve patient experience through smart time-slot management and automated reminders. By digitizing appointment workflows, the system contributes to the broader goals of improving healthcare efficiency, reducing patient wait times, and supporting data-driven decision-making in healthcare operations.

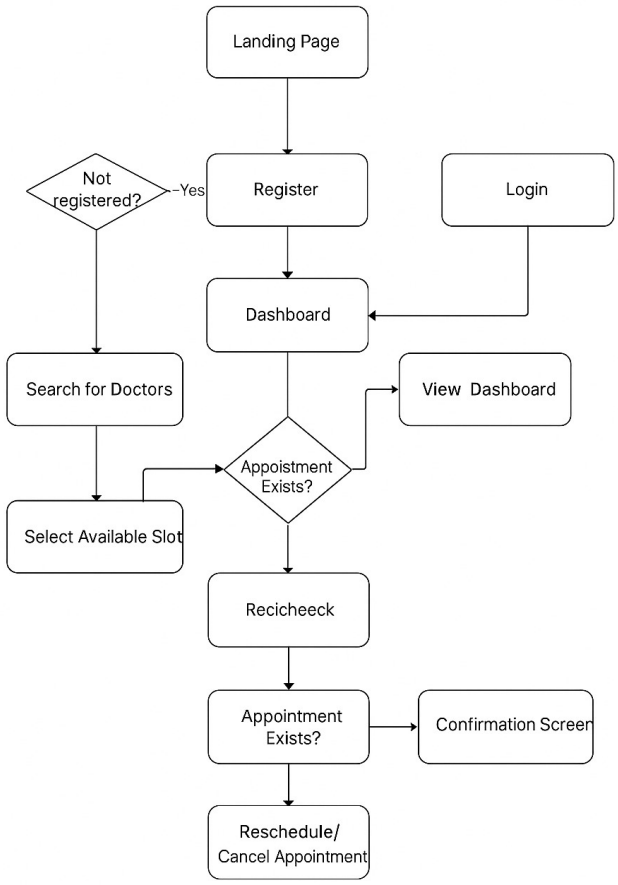
References

calendly.com

Overall Description

Product Perspective

MediSlot is a new, self-contained web-based product designed to address the inefficiencies in traditional medical appointment scheduling systems. It is not a follow-on member of an existing product line, but rather a standalone system aimed at hospitals, clinics, and diagnostic centers seeking to digitize and automate their outpatient scheduling processes. The system will interact with users through a browser-based interface and operate independently of any proprietary hospital software, ensuring broad compatibility.



Product Functions

1. Patient Functions:

- Register and securely log in to the platform
- Search and book for available doctors by specialization and date
- Reschedule or cancel existing appointments
- View upcoming and past appointments
- Receive automated SMS/email notifications for appointment confirmations, cancellations, and reminders

2. Medical Professional Functions:

- Log in to their dashboard using secure credentials

- Set, update, or block their availability for appointments
- View scheduled appointments by date
- Cancel or reschedule appointments when necessary
- Access a summary of appointments

Operating Environment

MediSlot operates in a web-based environment. Users can access the platform through modern web browsers such as Google Chrome, Mozilla Firefox, Microsoft Edge, and Safari. The platform is compatible with a wide range of operating systems, including Windows, macOS, Linux, Android, and iOS. No additional software installation is required, as MediSlot is a responsive web application that functions seamlessly on desktops, laptops, tablets, and smartphones. The system requires only a stable internet connection and a modern web browser for full functionality.

Design and Implementation Constraints

- **Regulatory Compliance:**
  - The system must comply with data privacy and healthcare regulations to ensure secure handling of patient information.
  - Any third-party services used for notifications, authentication, or data storage must also adhere to these standards.
- **Technology Constraints:**
  - The platform must be built as a web-based application using widely supported technologies (e.g., HTML5, CSS3, JavaScript, React.js for frontend, and Node.js for backend).
  - The database must use a relational database management system such as MySQL or PostgreSQL for compatibility and scalability.
- **Hardware Limitations:**
  - MediSlot must operate on standard web servers with typical hosting specifications. Memory and processing must be optimized to support at least 500 concurrent users.
  - Users must access the platform through devices with an active internet connection; offline functionality is not supported.
- **Interface Constraints:**

- The system must integrate seamlessly with SMTP servers for email notifications and optional SMS gateway APIs for text reminders.
- Integration with future hospital EHR systems must be possible without significant redesign.

- **Security Considerations:**
  - All data transfers must be secured using SSL/TLS encryption, and sensitive data must be encrypted in the database.
  - The system must implement role-based access control (RBAC) to limit functionality based on user type (Patient, Doctor, Admin).

- **Design Conventions:**
  - The software must follow clean code standards and modular design principles to ensure maintainability and scalability.
  - Developers must provide documentation for all modules to facilitate future updates and third-party maintenance.

User Documentation

User documentation for MediSlot will include online help guides, step-by-step tutorials, and a comprehensive user manual. The online help guides will be integrated within the application to assist users in performing specific actions such as booking appointments, setting doctor availability, and managing user profiles. Step-by-step tutorials will help new users, both patients and doctors to quickly understand how to navigate the platform, register, log in, and utilize key features effectively. Additionally, a detailed user manual will serve as a complete reference, covering all functionalities of MediSlot, including appointment management, dashboard usage, and account handling.
All documentation will be delivered in digital formats, accessible directly within the MediSlot web application designed with a user-first approach, the documentation will ensure clarity and ease of use for all users, regardless of their technical background. Since the entire platform prioritizes intuitive user flow and simplicity, the accompanying documentation will support this goal by offering clear, concise, and helpful guidance at every step.

Assumptions and Dependencies

The development and operation of MediSlot are based on several assumptions and dependencies. It is assumed that users, patients and doctors, have access to a stable internet connection and a modern web browser (such as Chrome, Firefox, or Edge) on their devices. The application depends on a backend database system (PostgreSQL) to store user data, appointment details, and doctor availability. It is also assumed that the hosting environment will support the required web server and scripting technologies used

in the development of MediSlot, such as Node.js, or similar frameworks. Additionally, the project may rely on third-party components or APIs for email notifications, calendar synchronization, and authentication and any changes in these services may affect the functionality of the platform. The system assumes minimal downtime for maintenance and that users will have basic digital literacy to navigate the platform.

External Interface Requirements

User Interfaces

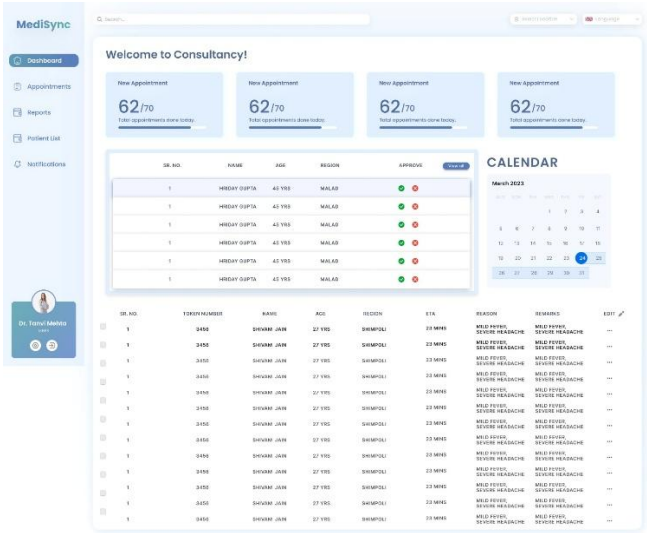
- **Design Standards:**
  - The interface will follow modern Material Design principles, ensuring consistency in color schemes, typography, and layout across all screens.
  - The platform will be fully web-responsive, adapting to different screen sizes on desktops, tablets, and smartphones.
  - Standard elements such as the navigation bar, help icon, and logout button will appear on every screen for easy access.

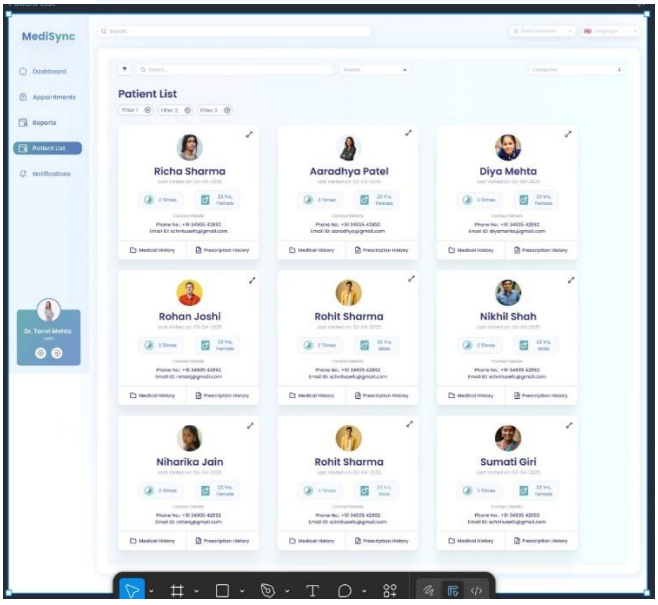
- **Screen Layout & Elements:**
  - Dashboard: Summarizes user-specific information. For patients, it displays upcoming appointments and booking options. For doctors, it shows scheduled appointments and availability settings.
  - Appointment Booking Screen: Includes search filters (by doctor, specialization, or hospital), available time slots, and booking confirmation prompts.
  - Availability Management (for Doctors): Provides an intuitive calendar interface for setting or blocking available slots.
  - Admin Panel: Allows admins to manage doctor accounts, view system statistics, and monitor user activity.

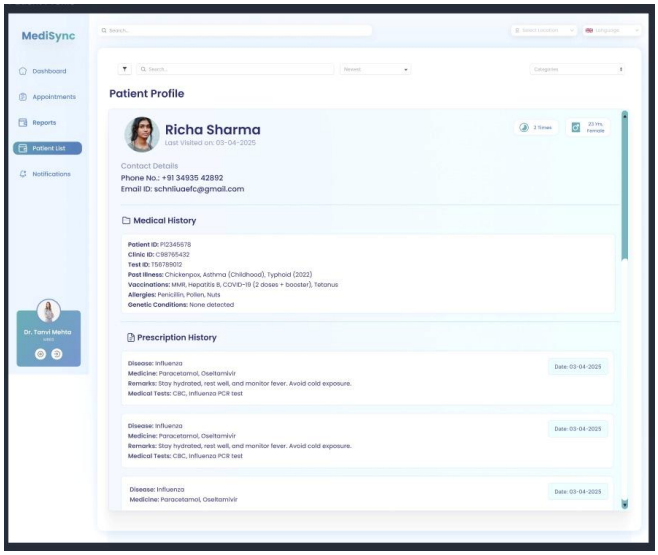
- **Error Messages & Validation:**
  - Error messages will be clear, descriptive, and non-technical.
  - Mandatory fields will be validated before submission, and incorrect inputs will be highlighted in red with contextual hints.

- **Standard Buttons and Functions:**
  - Buttons such as Confirm, Cancel, Save Changes, and Back will maintain uniform placement and styling across screens.
  - A Help/FAQ section will be accessible from every page.

  - **Sample UI:**


MediSync
Welcome to Consultancy!
CALENDAR


MediSync
Patient List
Richa Sharma
Aaradhya Patel
Diya Mehta
Rohan Joshi
Rohit Sharma
Nikhil Shah
Niharika Jain
Rohit Sharma
Sumati Giri



Hardware Interfaces

MediSlot operates as a web-based platform, requiring no direct hardware interfaces. It is designed to function seamlessly across a variety of user devices, including desktops, laptops, tablets and smartphones. The only essential hardware requirement is an active internet connection on the device being used to access the system. All interactions take place through standard web browsers, eliminating the need for any specialized hardware components.

Software Interfaces

MediSlot will interact with several software components and services to deliver its functionality. The system will be hosted on a server running a supported operating system and it will use a web server to handle client requests.
A relational database management system (RDBMS) such as MySQL or PostgreSQL will be used for data storage, managing information related to patients, doctors, appointments, and notifications. The backend application will communicate with the database using SQL queries and Object-Relational Mapping (ORM) frameworks if required.
MediSlot will also integrate with third-party services via RESTful APIs:

- Email notifications will be sent using services like SMTP.
- Authentication services may be supported using APIs for OTP-based verification or OAuth 2.0-enabled identity providers.

Incoming data from users will include registration details, login credentials, appointment requests, and schedule updates, typically transmitted as JSON payloads through secure

HTTPS requests. Outgoing data will consist of booking confirmations, appointment history, reminders, and system notifications, also structured in JSON or XML formats.

Communications Interfaces

MediSlot operates as a web-based application that relies on standard communication protocols to function effectively. All communication between the client and server takes place over the HTTP or HTTPS protocols, ensuring secure and encrypted data transfer. The platform is designed to work seamlessly through modern web browsers, allowing users (patients and doctors) to access features like booking, managing appointments, and viewing medical information in real-time. Email notifications are integrated into the system using SMTP protocols to confirm bookings, send reminders, and notify users of any updates. The application may also support communication with external services via RESTful APIs, for purposes such as authentication or SMS-based notifications. All form data exchanged through the platform is structured in standard formats such as JSON or URL-encoded formats to ensure compatibility and efficient data processing.

System Features

System Feature 1: Patient Appointment Booking

4.1.1 Description and Priority

This feature allows patients to search for doctors, view available time slots, and book appointments. It is a High priority feature as it forms the core functionality of the MediSlot system. Without this feature, the platform fails to meet its primary objective.

4.1.2 Stimulus/Response Sequences

- **Stimulus:** The patient logs in and selects the “Book Appointment” option.
  Response: The system displays a list of doctors categorized by specialization.
- **Stimulus:** The patient selects a doctor and preferred date.
  Response: The system shows all available time slots for that doctor on the selected date.
- **Stimulus:** The patient selects a time slot and confirms booking.
  Response: The system saves the appointment, sends a confirmation email/SMS, and updates the doctor’s schedule.
- **Error Condition:** The patient selects a slot that has just been booked by another user.
  Response: The system notifies the patient that the slot is unavailable and prompts them to select another slot.

4.1.3 Functional Requirements

- **REQ-1:** The system must allow patients to search for doctors by specialization, name, or location.
- **REQ-2:** The system must display the doctor's available time slots in real time.
- **REQ-3:** The system must prevent double-booking of the same slot.
- **REQ-4:** The system must send booking confirmations via email/SMS.
- **REQ-5:** The system must validate user inputs and display error messages for invalid selections.

System Feature 2: Doctor Availability Management

4.2.1 Description and Priority

This feature allows doctors to set, update, or block their availability for appointments. It is a High priority feature, as it ensures accurate scheduling and optimal use of doctors' time.

4.2.2 Stimulus/Response Sequences

- **Stimulus:** The doctor logs in and selects the "Manage Availability" option.
  Response: The system displays the current availability schedule.
- **Stimulus:** The doctor updates available days and time slots or blocks off time for leave.
  Response: The system saves the updated availability and reflects it in the appointment booking interface.
- **Error Condition:** The doctor tries to block a time slot that already has a confirmed appointment.
  Response: The system displays a warning and prompts the doctor to reschedule or cancel existing appointments first.

4.1.3 Functional Requirements

- **REQ-1:** The system must allow doctors to set their weekly availability and block time slots when unavailable.
- **REQ-2:** The system must update the booking interface in real time when availability changes.
- **REQ-3:** The system must validate that blocking a slot with confirmed appointments is not allowed unless those appointments are canceled or rescheduled.

- **REQ-4:** The system must notify affected patients if a doctor updates availability that impacts booked appointments.

System Feature 3: Appointment Management

4.3.1 Description and Priority

This feature allows both patients and doctors to reschedule or cancel appointments. It is a High priority feature, as it provides flexibility and ensures up-to-date schedules.

4.3.2 Stimulus/Response Sequences

- **Stimulus:** The patient/doctor selects an existing appointment and chooses the "Reschedule" option.
  Response: The system displays available time slots for the same doctor.
- **Stimulus:** The user selects a new time slot.
  Response: The system updates the appointment and sends notifications to both parties.
- **Stimulus:** The patient/doctor chooses the "Cancel" option.
  Response: The system cancels the appointment and notifies the other party.
- **Error Condition:** The user tries to reschedule to a fully booked time slot.
  Response: The system notifies the user and prompts them to select a different time slot.

4.1.3 Functional Requirements

- **REQ-1:** The system must allow patients and doctors to reschedule existing appointments.
- **REQ-2:** The system must allow patients and doctors to cancel existing appointments.
- **REQ-3:** The system must update appointment records and availability in real time.
- **REQ-4:** The system must send notifications of changes to both the doctor and the patient.
- **REQ-5:** The system must validate that rescheduled slots are available.

System Feature 4: Notifications & Reminders

4.2.1 Description and Priority

This feature ensures that patients and doctors receive automated notifications for confirmations, cancellations, and reminders. It is a Medium priority feature as it improves user engagement and reduces missed appointments.

4.2.2 Stimulus/Response Sequences

- **Stimulus:** A patient books an appointment.
  Response: The system sends an immediate confirmation via email/SMS.
- **Stimulus:** An appointment is scheduled for the next day.
  Response: The system sends a reminder 24 hours before the appointment.
- **Stimulus:** A doctor updates availability affecting booked appointments.
  Response: The system sends notifications to affected patients.

4.1.3 Functional Requirements

- **REQ-1:** The system must send booking confirmations to patients and doctors.
- **REQ-2:** The system must send automated reminders before appointments.
- **REQ-3:** The system must send notifications of cancellations and reschedules.
- **REQ-4:** Notifications must be delivered via both email and SMS (if enabled).

Other Nonfunctional Requirements

Performance Requirements

- The platform should respond to user actions, such as booking or cancelling appointments, within 2 seconds to ensure a smooth user experience.
- Concurrent users (up to 500 users) should experience minimal slowdown in platform responsiveness.
- The application must be fully web-responsive across all devices (desktop, tablet, mobile) and provide the same user experience.
- Appointment availability and booking updates must be processed in real time to avoid double-booking.
- User authentication should be seamless and secure across the entire system.
- Notifications (email/SMS) must be triggered within 1 minute of any booking, cancellation, or schedule update.

Safety Requirements

- The system must ensure data backups are performed daily to prevent data loss in case of system failure.
- All user data, including patient and doctor information, must be encrypted during transmission and securely stored in the database to avoid breaches.
- The system must include role-based access control to prevent unauthorized users from accessing sensitive medical information.
- The platform must comply with relevant data protection regulations for handling medical and personal information.
- Users must be warned and prevented from performing destructive actions (e.g., deleting accounts or appointments) without proper confirmation.
- Regular security audits and vulnerability checks must be performed to ensure system safety and reliability.

Security Requirements

- The system must implement secure user authentication using strong password policies and optional OTP-based verification for patients and doctors.
- Role-based access control must be enforced to ensure that only authorized users (patients, doctors, or admin) can access their respective features and data.
- All sensitive user data (personal details, medical information, login credentials) must be encrypted and at rest in the database.
- Session management must automatically log out inactive users after a predefined period to prevent unauthorized access.
- The system must comply with data privacy regulations for handling patient information.
- Regular security audits, penetration testing, and vulnerability assessments must be conducted to identify and fix security risks.
- Users must be notified of any security breach or unauthorized data access attempts as per compliance standards.
- The platform must protect against common web vulnerabilities such as SQL Injection, Cross-Site Scripting (XSS), and Cross-Site Request Forgery (CSRF).

Software Quality Attributes

- The system must maintain 99% uptime, excluding scheduled maintenance, to ensure continuous access for patients and doctors.
- All appointment bookings and updates must be processed accurately and consistently, ensuring no data loss or double-booking scenarios.
- The platform must be user-friendly, with an intuitive interface designed for users of varying technical skills. Key actions (e.g., booking an appointment) should be achievable within 3 clicks or less.
- The system must be modular and documented to allow easy updates, bug fixes, and feature enhancements without disrupting operations.
- Strong data encryption, authentication, and compliance with data protection standards must be ensured.

- The platform must be capable of scaling to support a larger user base by upgrading resources with minimal configuration changes.
- MediSlot must be compatible across all modern web browsers and devices, ensuring the same user experience on desktops, tablets, and smartphones.
- The system must support automated and manual testing, with logging mechanisms to track and debug errors efficiently.

Business Rules

- **Role-Based Access:**
  - Only **patients** can book, reschedule, or cancel appointments for themselves.
  - Only **doctors** can manage their availability schedules and view appointment details.
  - Only the **admin (receptionist)** can create, edit, or deactivate doctor accounts.

- **Appointment Policies:**
  - Patients cannot book overlapping appointments for the same time slot.
  - Patients must cancel or reschedule appointments at least **2 hours before** the scheduled time (configurable by the admin).
  - Doctors must notify patients through the system if they cancel or modify an appointment.

- **Data Access Restrictions:**
  - Patients cannot view other patients' data.
  - Doctors can only access information related to their scheduled patients.
  - Admins have full access to doctor data but not to detailed patient medical histories.

- **Notifications:**
  - All booking confirmations, cancellations, and reminders must be automatically communicated to patients and doctors via email/SMS.
  - **Security Compliance:**
  - All users must authenticate using valid credentials before accessing any system feature.
  - Sensitive information must never be shared outside the platform or with unauthorized individuals.

Other Requirements

- **Database Requirements:**
  - The system must use a secure and reliable relational database (e.g., MySQL or PostgreSQL) to store patient, doctor, and appointment data.
  - All database operations must be optimized for quick access and retrieval, ensuring minimal latency for large data sets.
  - Daily backups must be performed to prevent data loss.

- **Internationalization & Localization:**
  - The system should support multiple languages for the user interface (English as default) to cater to diverse user bases.
  - Time zones must be handled appropriately for appointment scheduling in different geographic regions.

- **Legal & Regulatory Requirements:**
  - The system must comply with applicable data privacy laws to safeguard sensitive medical information.
  - All user data must be stored and processed according to relevant healthcare standards.

  - **Reuse Objectives:**
    Core modules such as authentication, notification services, and appointment scheduling must be designed for reuse in future versions or other healthcare applications.

- **System Logging & Auditing:**
  - The system must maintain detailed logs of user activities, errors, and system events for auditing and debugging purposes.
  - Logs must be stored securely and accessible only to authorized administrators.

- **Integration Requirements:**
  - MediSlot must be capable of integrating with third-party services such as SMS gateways, payment systems and hospital EHR (Electronic Health Record) systems without extensive redevelopment.

Appendix A: Glossary

- **Admin (Receptionist):** A user with administrative privileges who can manage doctor accounts and oversee system operations.
- **Appointment Slot:** A predefined time interval available for booking by patients.
- **Authentication:** The process of verifying a user's identity using login credentials or OTP verification.
- **EHR (Electronic Health Record):** A digital version of a patient's health information that may integrate with MediSlot in the future.
- **Patient:** A user who books appointments with doctors via the MediSlot platform.
- **Doctor:** A user providing medical services who can set availability and manage scheduled appointments.
- **Notifications:** Messages sent via email/SMS to inform users about bookings, cancellations, or reminders.

- **RDBMS:** Relational Database Management System used for storing and managing structured data.
- **UI (User Interface):** The visual elements and design of the MediSlot platform.

Post Laboratory Activity:

1. You are required to prepare SRS document for any project. (It could be the mini project you have completed in semester IV)
   Software Requirements Specification (SRS)
   Project Title: Health & Wellness Blog – A Responsive Platform for Evidence-Based Lifestyle Content
   Version: 1.0
   Prepared by: Aaryan Sharma, Aditey Kshirsagar, Aditya Baheti
   Organisation: K. J. Somaiya School of Engineering
   Date: 29 July 2025

1 Introduction

1.1 Purpose

This document defines all functional and non-functional requirements for **Health & Wellness Blog**, a PHP–MySQL web application that delivers curated articles, category-based exploration, and community interaction to promote healthier living.

1.2 Product Scope

The platform lets authors publish rich-media wellness posts, readers filter and discuss them, and admins maintain an orderly content pipeline. It aims to become a low-friction knowledge hub for students and staff seeking practical guidance on nutrition, mental health, fitness, and preventive care.

1.3 References

- GitHub repository – *Aaryan-Sharma-5/Health-and-Wellness-Blog* GitHub
- Figma UI mock-ups

2 Overall Description

Aspect	Detail
2.1 Product Perspective	Stand-alone LAMP-stack site; no dependency on external CMS.
2.2 Product Functions	• Display featured and latest articles • Category filter & keyword search • Dark-mode toggle • Comment thread per post • Admin CRUD for posts, images & categories
2.3 User Classes	**Visitor:** browse & search content. **Registered Reader:** post comments, like articles. **Author:** create/edit own posts. **Admin:** full moderation & user management.
2.4 Operating Environment	Modern browsers (Chrome, Firefox, Edge, Safari); PHP 8.3, MySQL 8 on Apache 2.4; tested on Windows-XAMPP & Ubuntu-LAMP.
2.5 Design Constraints	Must load without JS errors in dark-mode; database tables follow InnoDB with UTF-8.
2.6 User Documentation	In-app tooltips, README install guide.

3 External Interface Requirements

3.1 User Interfaces

- **Home:** hero banner, featured carousel, category chips, dark-mode switch.
- **Article Page:** title, author card, reading-time badge, social share, comment section.
- **Dashboard (Admin):** post list, bulk actions, image uploader, category manager.

3.2 Hardware Interfaces

None, runs in standard browsers on any desktop / mobile device.

3.3 Software Interfaces

- MySQL via PDO for data persistence.
- Future: optional REST endpoint for a mobile app.

3.4 Communications Interfaces

HTTP/HTTPS; JSON for any future AJAX endpoints; current PHP renders server-side HTML.

4 System Features

4.1 Article Management (High)

- **FR-1:** Author can create, save draft, publish, edit, delete own articles.
- **FR-2:** Admin can moderate any article and mark as Featured.
- **FR-3:** System stores hero image and rich-text body (markdown/HTML).

4.2 Category & Filter (High)

- **FR-1:** Visitor can filter by category tag or search keyword.
- **FR-2:** Results update within 2 s, most-recent first.

4.3 Comment System (Medium)

- **FR-1:** Registered reader can post, edit, delete own comment.
- **FR-2:** Admin can remove abusive comments; simple profanity check.

4.4 Dark-Mode & Responsiveness (Medium)

- **FR-1:** Toggle persists in browser localStorage.
- **FR-2:** Layout adapts to screens ≥ 360 px wide without horizontal scroll.

4.5 Authentication & Roles (Medium)

- **FR-1:** Email-password signup/sign-in; hashed with Argon2id.
- **FR-2:** Role-based access controls gate Author and Admin routes.

5 Other Non-Functional Requirements

Category	Requirement
Performance	≤ 2 s first contentful paint on 4G; ≤ 500 ms DB query time.
Reliability	99 % uptime during semester; nightly DB dump.
Security	Prepared-statement queries; CSRF tokens on all forms; content-security-policy headers.
Usability	Key actions (read, search, comment) max 3 clicks from home.
Maintainability	MVC-ish PHP folder structure; PSR-12 coding standard; unit tests for DAO layer.

6 Business Rules

1. Featured articles limited to five; oldest auto-demoted when admin adds sixth.
2. Unverified emails cannot post comments.
3. Image uploads restricted to ≤ 5 MB, JPEG/PNG only.

7 Other Requirements

Planned phase-2 adds: newsletter subscription, user profiles with badge system, and REST API for a progressive-web-app frontend.

Appendix A Glossary

Term	Meaning
Dark Mode	UI colour scheme optimised for low-light viewing.
Featured	Article promoted to home-page carousel.

2. Prepare questionnaire for the allotted project considering your lab instructor is the client for requirement gathering.

    - What core features do you absolutely want in version 1 of MediSlot?
    - How should patients create an account—email + password, phone OTP, or something else?

- Should doctors go through a separate onboarding/approval step before they can accept bookings?
- Can a patient hold more than one active appointment at a time?
- How far in advance should booking slots open, days, weeks, or months?
- Do you want doctors to mark special "emergency" or walk-in slots beyond normal hours?
- Which reminder channels are mandatory, SMS, email, WhatsApp, or all of them?
- Should we add a real payment gateway now, or just simulate fee collection for the project demo?
- Which languages must the interface and notifications support at launch?
- Do you want an admin dashboard showing daily/weekly booking statistics?
- Should late cancellations or no-shows trigger any penalties or warnings inside the app?
- Do patients need the option to upload files (e.g., previous reports) while booking?
- Must the system work smoothly on mobile screens, or is desktop-only acceptable for now?
- Is integration with the institute's existing EHR/EMR or calendar system required in this phase?
- Are there any existing apps or sites you'd like us to use as a design or feature reference?

3. Consider following scenario: An institute is interested in developing a Library Information System (LIS) for the benefit of students and employees of the institute. LIS will enable the members to borrow a book (or return it) with ease while sitting at his desk/chamber. The system also enables a member to extend the date of his borrowing if no other booking for that particular book has been made. For the library staff, this system aids them to easily handle day-to-day book transactions. The librarian, who has administrative privileges and complete control over the system, can enter a new record into the system when a new book has been purchased, or remove a record in case any book is taken off the shelf. Any non-member is free to use this system to browse/search books online. However, issuing or returning books is restricted to valid users (members) of LIS only.

   The final deliverable would a web application (using the recent HTML 5), which should run only within the institute LAN. Although this reduces security risk of the software to a large extent, care should be taken no confidential information (e.g. passwords) is stored in plain text.

   Prepare SRS document for the same in the format discussed in the write-up.

Purpose

Provide a browser-based way for institute members to search, borrow, return, and renew books while letting library staff manage inventory and view transaction history.

Scope

Campus-LAN-only web app (HTML5 front end, lightweight PHP + MySQL back end). Public visitors can search; only authenticated members can transact.

User Roles and Permissions

- Visitor: catalogue search only.
- Member (student or staff): borrow, return, renew own loans.
- Library Staff: process member transactions at the desk.
- Librarian (admin): add / remove book records, override renewals, view all logs.

Core Features

1. Live catalogue search and book details.
2. One-click borrow / return with instant stock update.
3. Online renewal allowed when no active hold exists.
4. Inventory CRUD via secure librarian dashboard.
5. Immutable audit log retained for five years.

Operating Environment

Modern HTML5 browser on desktop or laptop; PHP 8 and MySQL 8 hosted on an intranet Apache server; USB barcode scanners work as HID keyboards.

Key Non-Functional Requirements

- Performance: search and loan actions must respond in under two seconds.
- Security: passwords hashed with Argon2; HTTPS enforced inside LAN.
- Availability: system up at least 99 % of library hours.
- Data protection: nightly database backups; audit logs stored five years.

Design Constraints

No external internet access; must comply with the India Copyright Act; UI must degrade gracefully if JavaScript is disabled.

Glossary

Hold – reservation placed on a loaned book.
Member – authenticated user with borrowing rights.

Post Lab Descriptive Questions

1. **What are different techniques to gather information for software development?**

   There are several methods to gather requirements, but not all are equally effective in every context. For a system like the Library Information System (LIS), the following are the most practical:

   1. Stakeholder Interviews – Talking directly with librarians, students, and faculty gives insight into day-to-day pain points and what they actually need the system to do.
   2. On-Site Observation – Watching how book issuing and returns happen today helps identify bottlenecks and inefficiencies that users may not even mention.
   3. Prototyping – Creating a quick mockup or screen flow helps users visualize the system and give feedback before it's built.
   4. Questionnaires – Helpful if you want to get broader student/faculty feedback about their expectations from the system, especially for features like book search or renewals.
   5. Requirement Workshops – Get stakeholders in a room (physically or virtually), and brainstorm features and flows.

2. **List verification and validation techniques for requirements.**

   1. Verification
      1. Reviews – Peer or stakeholder reviews of the requirement doc
      2. Inspections – Formal and structured document checking
      3. Walkthroughs – Step-by-step exploration of requirement documents
      4. Checklists – Predefined lists to catch missing or inconsistent items

   2. Validation
      1. Prototyping – Validate assumptions by showing a sample UI
      2. User Feedback – Gather client opinions on the draft SRS
      3. Acceptance Criteria Review – Ensure requirements are testable and meet client expectations
      4. Simulations – Use models to demonstrate system behavior