SOFTWARE REQUIREMENTS SPECIFICATION

for

DigiMED

Prepared by

Adam Mitry, Ahmed Abdelkader,

Hoda Hussein, Mohamed Khalil Brik,

Rana Taher, Yasmina Mahdy

Version 2.0

November 1, 2024

Revision History

Date	Reason for Changes	Version
2024-10-23	Initial draft	1.0
2024-11-1	Added sprint 2 requirements	2.0

Contents

1 Introduction

1.1 Purpose

The purpose of this document is to provide a comprehensive overview of the software requirements for DigiMED, a digital healthcare platform aimed at facilitating efficient and secure interaction between doctors, patients, and healthcare administrators. This Software Requirements Specification (SRS) will guide the development process by clearly defining both functional and non-functional requirements, ensuring the final product meets the needs of all users and complies with healthcare industry standards.

1.2 Scope

This core subset of DigiMED's features is centered on providing essential functionalities within two main user views: **Patient View** and **Admin View**.

Patient View: Aimed at empowering patients with access to their medical information and appointment management, the Patient View includes:

- **Appointment Scheduling**: Allows patients to view, book, and manage their appointments with healthcare providers.
- **Medical Records Access**: Enables patients to securely view essential parts of their medical history, providing transparency and continuity of care.
- **Prescription Management**: Provides patients with access to their current prescriptions, enhancing adherence to medication schedules.

Admin View: Designed to enable administrators to manage essential system components and ensure smooth platform operation, the Admin View includes:

- **User Management**: Allows admins to view, create, edit, and delete profiles for doctors, patients, and other healthcare roles. Admins can assign user roles and manage basic access permissions.
- **Scheduling Management**: Enables admins to create and modify doctor schedules, ensuring efficient use of resources. Admins can also oversee patient appointments, allowing for easy management of the appointment pipeline.

- **Access Control**: Provides role-based permissions management, ensuring that each user has appropriate access based on their role, helping maintain data security and regulatory compliance.

1.3 Terms

Patient View: The DigiMED interface for patients, providing access to basic health information, appointments, and prescriptions.

Admin View: The DigiMED interface for administrators, offering control over essential backend functions, including scheduling and user management.

Admin: Authorized personnel responsible for managing core user profiles, basic access, and scheduling.
Patient: A user accessing DigiMED to manage their healthcare needs, including appointments, medical records, and prescriptions.

Appointment: A scheduled healthcare session between a patient and a doctor.

Schedule: A doctor's availability and booked appointments, managed by admins for optimized scheduling.

Prescription: A doctor-authorized medication plan available to patients through DigiMED.

Access Control: The ability for admins to assign permissions, limiting or allowing user access to specific features.

1.4 Abbreviations

SRS: Software Requirements Specification

HIPAA: Health Insurance Portability and Accountability Act

GDPR: General Data Protection Regulation

UI: User Interface

UX: User Experience

DBMS: Database Management System

2 Product Overview

2.1 Product Perspective

2.1.1 System Interfaces

2.1.2 User Interfaces

2.1.3 Hardware Interfaces

2.1.4 Software Interfaces

2.1.5 Communication Interfaces

2.1.6 Memory Constraints

2.1.7 Operations

2.1.8 Site Adaptation Requirements

2.1.9 Interface with Services

2.2 Product Functions

2.3 User Characteristics

2.4 Limitations

2.5 Assumptions and Dependencies

2.6 Apportioning of Requirements

1. **Share and Update a Doctor's Schedule:** Enables administrative staff to share and make updates to a doctor's schedule as necessary.
2. **Access to Doctor's Schedule and Appointments:** Grants certain administrative users access to view and manage doctor schedules and appointments.
3. **Contact Patients Scheduled with Doctor:** Allows administrative staff to contact patients with scheduled appointments.
4. **Optimize Doctor's Schedule:** Enables optimization of a doctor's schedule based on patient needs and availability.

5. **Update Patient Medical Records:** Allows doctors to update and add new details to a patient's medical records.
6. **Write a Prescription for Patient:** Enables doctors to create and issue prescriptions directly for their patients.
7. **View My Scheduled Appointments:** Allows doctors to view all upcoming appointments in their schedules.
8. **Overwrite My Schedule:** Provides doctors the option to adjust their schedules directly when needed.
9. **Contact Patients' Previous Doctors:** Allows administrative staff to contact patients' previous doctors for additional medical information.
10. **Approve or Modify Online Appointment Requests:** Grants doctors the ability to approve or request modifications for online appointment bookings.

2.7 Specified Requirements

Profile Management

1. **Create a Profile:** Allows users to create a new patient profile with required personal and contact information.
2. **Delete a Profile:** Enables patients or admins to permanently delete an existing profile from the system.
3. **Edit a Profile:** Allows users to update personal details within an existing patient profile.
4. **Retrieve a Profile:** Enables authorized users to access a patient’s profile information for review.

Patient Medical Records

1. **Edit an Existing Record:** Allows doctors to modify details in an existing medical record for accuracy or updates.
2. **Retrieve a Record:** Enables users with permission to view a specific patient’s medical record.
3. **Delete a Record (Admin Only):** Permits admins to permanently remove a patient’s medical record from the system.
4. **Create a Record:** Allows doctors to add new medical records for patients with relevant details.

Access Management to Records

1. **Restrict Record Access by Role:** Enforces permissions based on user roles, allowing or denying access to patient medical records.

Scheduling System

1. **Create a Schedule:** Allows doctors to set available appointment slots for patient bookings.
2. **Create an Appointment in a Schedule:** Enables patients to book an available time slot in a doctor's schedule.
3. **Edit an Existing Schedule:** Allows doctors to modify their availability in an existing schedule.
4. **Delete a Schedule:** Enables doctors to remove an unwanted schedule, canceling associated slots.
5. **Delete an Appointment:** Permits patients to cancel a previously booked appointment.
6. **Retrieve a Schedule:** Enables patients to view available appointment slots for a specific doctor.
7. **Access Management to Scheduling:** Controls scheduling access rights based on user roles and permissions.

3 Specific Requirements

3.1 External Interfaces

3.2 Functions

1. Profile Management

1.1. Create a Profile

- **Scenario:** A new patient wants to register in the healthcare system.
- **Function:** A user can create a profile for themselves or for a new patient.
- **Description:** Through the profile management service, a user can fill out their details to create a new patient profile.
- **Inputs:** Patient username (string), Date of Birth (string), Contact information (string),Password (string).
- **Source:** The client request source is from the API Gateway; the user registers through the Patient UI.
- **Outputs:** Confirmation of the newly created profile with a unique Patient ID.
- **Destination:** The patient's profile is saved in the Database management system.
- **Action:** The user fills in the necessary fields and submits the form to create a new profile.
- **Requirements and Pre-condition:** The system must allow new registrations and validate the input fields for correctness.
- **Post-condition and Side effects:** The profile is stored in the database, and the patient may receive a confirmation email or email asking for re_authentication in case the profile wasn't properly authenticated .

1.2. Delete a Profile

- **Scenario:** A patient no longer wants to be part of the app..
- **Function:** A user can delete their profile or that of a patient (if an admin).
- **Description:** The profile management system allows users to permanently remove their profiles.
- **Inputs:** Patient ID (string), Password (string).
- **Source:** The client request source is from the API Gateway; the user initiates the delete through the Patient UI or Admin UI.
- **Outputs:** Confirmation of successful deletion or an error message if the process fails.
- **Destination:** The patient's profile is removed from the Database management system.
- **Action:** The user selects the option to delete their profile and confirms the action.
- **Requirements and Pre-condition:** The system must verify user permissions for deletion.
- **Post-condition and Side effects:** The profile is permanently removed, and related data may also be deleted.

1.3. Edit a Profile

- **Scenario:** A patient wants to update their contact information.
- **Function:** A user can edit their profile.
- **Description:** The profile management system allows users to make changes to their existing profiles.
- **Inputs:** Patient ID (string), updated information (string).
- **Source:** The client request source is from the API Gateway; the user edits their profile through the Patient UI.
- **Outputs:** Confirmation of the updated profile.
- **Destination:** The patient's profile is updated in the Database management system.
- **Action:** The user updates the necessary fields and submits the changes.
- **Requirements and Pre-condition:** The system must validate input data before updating.
- **Post-condition and Side effects:** The profile is updated, and the patient may receive a confirmation email.

1.4. Retrieve a Profile

- **Scenario:** A doctor needs to view a patient’s profile before an appointment.
- **Function:** A user can retrieve a profile.
- **Description:** The profile management system allows users to access patient profiles based on the Patient ID.
- **Inputs:** Patient ID (string).
- **Source:** The client request source is from the API Gateway; the user requests the profile through the Doctor UI or Patient UI.

- **Outputs:** The patient's profile information.
- **Destination:** Data is retrieved from the Database management system.
- **Action:** The user searches for the patient using their Patient ID.
- **Requirements and Pre-condition:** The system must allow profile retrieval for authorized users only.
- **Post-condition and Side effects:** The requested profile is displayed, allowing for updates or confirmations.

---

2. Patient Medical Records

2.1. Edit an Existing Record

- **Scenario:** A doctor needs to update a patient's medical history.
- **Function:** A doctor can edit an existing medical record.
- **Description:** Through the medical record management system, a doctor can modify a patient's existing records.
- **Inputs:** Patient ID (string), Record ID (string), updated record details (string).
- **Source:** The client request source is from the API Gateway; the doctor edits the record through the Doctor UI.
- **Outputs:** Confirmation of the updated medical record.
- **Destination:** The medical record is updated in the Database management system.
- **Action:** The doctor selects the record and submits the changes.
- **Requirements and Pre-condition:** The system must ensure only authorized personnel can edit records.
- **Post-condition and Side effects:** The record is updated, and a notification may be sent to the patient.

2.2. Retrieve a Record

- **Scenario:** A patient wants to view their medical history.
- **Function:** A user can retrieve a medical record.
- **Description:** The medical record management system allows users to access specific records.
- **Inputs:** Patient ID (string), Record ID (string).
- **Source:** The client request source is from the API Gateway; the user requests the record through the Patient UI or Doctor UI.
- **Outputs:** The requested medical record.
- **Destination:** Data is retrieved from the Database management system.
- **Action:** The user searches for the record using Patient ID and Record ID.
- **Requirements and Pre-condition:** The system must validate user permissions before allowing access.
- **Post-condition and Side effects:** The requested record is displayed for the user.

2.3. Delete a Record (Admin Only)

- **Scenario:** An admin needs to remove a patient's outdated medical record.
- **Function:** An admin can delete a medical record.
- **Description:** The medical record management system allows admins to permanently remove records.
- **Inputs:** Patient ID (string), Record ID (string).
- **Source:** The client request source is from the API Gateway; the admin initiates the delete through the Admin UI.
- **Outputs:** Confirmation of successful deletion.
- **Destination:** The medical record is removed from the Database management system.
- **Action:** The admin selects the record and confirms deletion.
- **Requirements and Pre-condition:** The system must verify admin permissions for deletion.
- **Post-condition and Side effects:** The record is permanently removed, and related data may also be deleted.

2.4. Create a Record

- **Scenario:** A doctor wants to add a new medical record for a patient.
- **Function:** A doctor can create a new medical record.
- **Description:** The medical record management system allows doctors to create new entries for patient records.
- **Inputs:** Patient ID (string), record details (string).
- **Source:** The client request source is from the API Gateway; the doctor creates the record through the Doctor UI.
- **Outputs:** Confirmation of the newly created medical record.
- **Destination:** The record is saved in the Database management system.
- **Action:** The doctor fills in the necessary details and submits the form.
- **Requirements and Pre-condition:** The system must validate input fields for correctness before saving.
- **Post-condition and Side effects:** The record is stored, and the patient may be notified of the update.

2.5. Access Management to Records

- **Scenario:** Different users require different levels of access to medical records.
- **Function:** The system manages user access levels to medical records.
- **Description:** The medical record management system restricts or grants access to medical records based on user roles.
- **Inputs:** User ID (string), access level (string).
- **Source:** The client request source is from the API Gateway.
- **Outputs:** Confirmation of access rights granted or denied.
- **Destination:** Access rights are managed through the Database management system.
- **Action:** The admin or authorized personnel adjusts user access levels.

- **Requirements and Pre-condition:** The system must have defined roles and permissions.
- **Post-condition and Side effects:** Users receive notifications if their access is changed.

---

3. Scheduling System

3.1. Create a Schedule

- **Scenario:** A doctor wants to set their availability for appointments.
- **Function:** A doctor can create a new schedule.
- **Description:** The scheduling management service allows doctors to specify their available time slots.
- **Inputs:** Doctor ID (string), available dates and times (string).
- **Source:** The client request source is from the API Gateway; the doctor creates the schedule through the Doctor UI.
- **Outputs:** Confirmation of the newly created schedule.
- **Destination:** The schedule is saved in the Database management system.
- **Action:** The doctor fills out the scheduling form and submits it.
- **Requirements and Pre-condition:** The system must validate the input for availability conflicts.
- **Post-condition and Side effects:** The schedule is stored, and notifications may be sent to patients.

3.2. Create an Appointment in a Schedule

- **Scenario:** A patient wants to book an appointment with a doctor.
- **Function:** A user can create an appointment in a doctor's schedule.
- **Description:** The appointment system allows users to book time slots in the available schedules.
- **Inputs:** Patient ID (string), Doctor ID (string), selected date and time (string).
- **Source:** The client request source is from the API Gateway; the user books the appointment through the Patient UI.
- **Outputs:** Confirmation of the booked appointment.
- **Destination:** The appointment is stored in the Database management system.
- **Action:** The user selects the time slot and confirms the appointment.
- **Requirements and Pre-condition:** The system must ensure the time slot is available before confirming the appointment.
- **Post-condition and Side effects:** The appointment is confirmed, and notifications may be sent to both doctor and patient.

3.3. Edit an Existing Schedule

- **Scenario:** A doctor needs to adjust their available hours.

- **Function:** A doctor can edit an existing schedule.
- **Description:** The scheduling system allows doctors to modify their availability.
- **Inputs:** Schedule ID (string), updated availability details (string).
- **Source:** The client request source is from the API Gateway; the doctor edits the schedule through the Doctor UI.
- **Outputs:** Confirmation of the updated schedule.
- **Destination:** The schedule is updated in the Database management system.
- **Action:** The doctor selects the schedule and submits the updated details.
- **Requirements and Pre-condition:** The system must validate the new availability against existing appointments.
- **Post-condition and Side effects:** The schedule is updated, and notifications may be sent to affected patients.

3.4. Delete a Schedule

- **Scenario:** A doctor no longer wants to keep a specific schedule.
- **Function:** A doctor can delete a schedule.
- **Description:** The scheduling system allows doctors to remove their schedules when they are no longer needed.
- **Inputs:** Schedule ID (string).
- **Source:** The client request source is from the API Gateway; the doctor deletes the schedule through the Doctor UI.
- **Outputs:** Confirmation of successful deletion.
- **Destination:** The schedule is removed from the Database management system.
- **Action:** The doctor selects the schedule to delete and confirms the action.
- **Requirements and Pre-condition:** The system must check for any existing appointments before allowing deletion.
- **Post-condition and Side effects:** The schedule is removed, and patients with existing appointments may be notified of the change.

3.5. Delete an Appointment

- **Scenario:** A patient wishes to cancel their appointment.
- **Function:** A user can delete an appointment.
- **Description:** The scheduling management system allows users to cancel their scheduled appointments.
- **Inputs:** Appointment ID (string).
- **Source:** The client request source is from the API Gateway; the user cancels the appointment through the Patient UI.
- **Outputs:** Confirmation of appointment cancellation.
- **Destination:** The appointment is removed from the Database management system.
- **Action:** The user selects the appointment to cancel and confirms the cancellation.
- **Requirements and Pre-condition:** The system must validate cancellation policies before confirming.

- **Post-condition and Side effects:** The appointment is removed, and both the doctor and patient may be notified.

3.6. Retrieve a Schedule

- **Scenario:** A patient wants to view available appointment slots for a doctor.
- **Function:** A user can retrieve a doctor's schedule.
- **Description:** The scheduling system allows users to see available time slots based on the doctor's schedule.
- **Inputs:** Doctor ID (string).
- **Source:** The client request source is from the API Gateway; the user requests the schedule through the Patient UI.
- **Outputs:** The available time slots for the doctor.
- **Destination:** Data is retrieved from the Database management system.
- **Action:** The user searches for the doctor's schedule using the Doctor ID.
- **Requirements and Pre-condition:** The system must allow schedule retrieval for authorized users.
- **Post-condition and Side effects:** The user receives a list of available time slots to choose from.

3.7. Access Management to Scheduling

- **Scenario:** Different users require different levels of access to scheduling features.
- **Function:** The system manages user access levels to scheduling features.
- **Description:** The scheduling management system restricts or grants access to scheduling functions based on user roles.
- **Inputs:** User ID (string), access level (string).
- **Source:** The client request source is from the API Gateway.
- **Outputs:** Confirmation of access rights granted or denied.
- **Destination:** Access rights are managed through the Database management system.
- **Action:** The admin or authorized personnel adjusts user access levels.
- **Requirements and Pre-condition:** The system must have defined roles and permissions.
- **Post-condition and Side effects:** Users receive notifications if their access is changed.

3.3 Usability Requirements

3.4 Performance Requirements

3.5 Logical Database Requirements

3.6 Design Constraints

3.7 Standards Compliance

3.8 Software System Attributes

4 Verification

5 Supporting Information