# Software Requirements Specification (SRS)

## MedMate – Smart Prescription & Medicine Tracker

**Team Orbit**

Course: CS-458 | Class: II-A | Department: UBIT

Version 1.0 | Date: August 28, 2025

### Development Team Members

Muhammad Hasan Khan (B23110006099) - Project Manager

Muhammad Hamza Abbas (B23110006096) - Deployment Manager

Husain Ahmad Khan (B23110006045) - QA Tester

Muhammad Mujtaba (B23110006107) - Lead Developer

Rayyan Ibrahim (B23110006137) - UI/UX Designer

## 1. Introduction

### 1.1 Purpose

This Software Requirements Specification (SRS) document provides a comprehensive description of the MedMate Smart Prescription & Medicine Tracker system. The document is intended for the development team, project stakeholders, course instructors, and future maintainers of the system. It defines the functional and non-functional requirements, system architecture, constraints, and acceptance criteria for the MedMate web application.

The MedMate system aims to revolutionize prescription and medicine management through a comprehensive digital solution that addresses critical healthcare challenges including lost paper prescriptions, missed medication doses, and fragmented healthcare records.

### 1.2 Scope

MedMate is a web-based application designed to provide secure digital prescription management, automated medication reminders, and family-based healthcare coordination. The system will be developed using the MERN stack (MongoDB, Express.js, React.js, Node.js) and deployed on cloud platforms.

#### 1.2.1 Included Functionality

* User registration, authentication, and secure account management for patients and caregivers
* Digital prescription upload, storage, and management system
* Automated medication reminder and notification system
* Family profile management allowing single-user management of multiple dependents
* Secure data encryption and privacy protection
* Responsive web-based platform optimized for various devices
* Admin panel for system monitoring and user management

#### 1.2.2 Excluded Functionality

* Native mobile applications (Android/iOS)
* Direct integration with third-party pharmacy databases or ERP systems
* Online prescription issuing by doctors
* E-commerce features including payment processing for medicine orders
* Multi-language support (English only)
* AI-powered medical advice or recommendations
* Prescription scanning with OCR technology
* Full HIPAA/GDPR compliance certification

### 1.3 Definitions, Acronyms, and Abbreviations

**API**

Application Programming Interface - A set of protocols and tools for building software applications

**CCB**

Change Control Board - A group responsible for reviewing and approving project changes

**CRUD**

Create, Read, Update, Delete - Basic database operations

**JWT**

JSON Web Token - A compact, URL-safe means of representing claims to be transferred between two parties

**MERN**

MongoDB, Express.js, React.js, Node.js - Full-stack JavaScript technology stack

**MVP**

Minimum Viable Product - A product with minimum features sufficient to satisfy early customers

**RBAC**

Role-Based Access Control - A security approach that restricts system access based on user roles

**REST**

Representational State Transfer - An architectural style for designing networked applications

**SRS**

Software Requirements Specification - A document that describes the intended behavior of a software system

**UI/UX**

User Interface/User Experience - Design aspects focused on user interaction and experience

**WBS**

Work Breakdown Structure - A hierarchical decomposition of project work

### 1.4 References

* MedMate Project Charter v1.0
* MedMate Business Case Document
* MedMate Project Plan and WBS
* MedMate Scope Management Plan
* IEEE Std 830-1998 - IEEE Recommended Practice for Software Requirements Specifications
* MERN Stack Documentation
* MongoDB Atlas Documentation
* React.js Official Documentation
* Node.js and Express.js Documentation

### 1.5 Overview

This SRS document is organized into five main sections. Section 1 provides an introduction to the document and project. Section 2 gives an overall description of the MedMate system including product perspective, functions, and user characteristics. Section 3 details the specific functional and non-functional requirements. Section 4 presents system models and architectural designs. Section 5 contains appendices including glossary and supporting information.

## 2. Overall Description

### 2.1 Product Perspective

MedMate is a standalone web application that operates independently without requiring integration with existing healthcare systems in the initial version. The system consists of a client-server architecture with the following major components:

* **Frontend Client:** React.js-based web interface accessible through modern web browsers
* **Backend Server:** Node.js and Express.js RESTful API server handling business logic
* **Database:** MongoDB Atlas cloud database for data persistence
* **File Storage:** Cloudinary or Firebase for prescription image storage
* **Authentication Service:** JWT-based authentication and authorization system

### 2.2 Product Functions

The MedMate system provides the following major functions:

#### 2.2.1 User Management

* User registration and profile creation
* Secure login and logout functionality
* Password management and recovery
* Role-based access control (Patient, Caregiver, Admin)
* Family profile management for multiple dependents

#### 2.2.2 Prescription Management

* Digital prescription upload (manual entry or image upload)
* Prescription storage and organization by user, date, and doctor
* Prescription viewing, editing, and deletion
* Prescription history tracking
* Search and filter functionality

#### 2.2.3 Medication Reminder System

* Automated medication reminders based on prescription schedules
* Multiple reminder settings per prescription
* Email and in-app notification delivery
* Reminder acknowledgment and snooze functionality
* Medication adherence tracking

#### 2.2.4 Administrative Functions

* User management and monitoring
* System health monitoring
* Prescription activity logs
* Security audit trails

### 2.3 User Classes and Characteristics

|  |  |  |  |
| --- | --- | --- | --- |
| **User Class** | **Description** | **Characteristics** | **Privileges** |
| Patients | Primary users managing their own medications | Basic to intermediate computer skills, health-conscious, may include elderly users | Full access to personal prescriptions, reminders, and profile management |
| Caregivers | Family members managing medications for dependents | Intermediate computer skills, responsible for multiple family members' health | Manage multiple family profiles, all patient privileges for assigned dependents |
| Administrators | System administrators monitoring and maintaining the platform | Advanced technical skills, responsible for system operations | User management, system monitoring, access to logs and analytics |
| Elderly Users | Senior citizens requiring medication management assistance | Limited computer skills, may require caregiver assistance, need accessibility features | Basic prescription viewing and reminder acknowledgment |

### 2.4 Operating Environment

#### 2.4.1 Client Environment

* **Web Browsers:** Chrome 90+, Firefox 88+, Safari 14+, Edge 90+
* **Operating Systems:** Windows 10+, macOS 10.15+, Linux (Ubuntu 18.04+), iOS 14+, Android 9+
* **Screen Resolutions:** Minimum 1024x768, responsive design for mobile devices
* **Internet Connection:** Broadband connection recommended, minimum 1 Mbps

#### 2.4.2 Server Environment

* **Frontend Hosting:** Vercel cloud platform
* **Backend Hosting:** Render or Heroku cloud platform
* **Database:** MongoDB Atlas cloud service
* **File Storage:** Cloudinary or Firebase cloud storage
* **CDN:** Global content delivery network for static assets

### 2.5 Design and Implementation Constraints

#### 2.5.1 Technology Constraints

* Must use MERN stack as per course requirements
* Limited to web-based platform only (no native mobile apps)
* Must utilize cloud-based hosting solutions
* English language only in initial version

#### 2.5.2 Regulatory Constraints

* Must include appropriate medical disclaimers
* Basic data privacy measures required
* No medical advice or AI recommendations
* User consent required for data collection

#### 2.5.3 Resource Constraints

* 4-month development timeline
* 5-person development team
* Limited budget using free/open-source tools
* Academic project scope limitations

### 2.6 User Documentation

The following user documentation will be provided:

* User Manual with step-by-step instructions
* Quick Start Guide for new users
* FAQ document addressing common issues
* Video tutorials for key features
* API documentation for developers
* System administration guide

## 3. Specific Requirements

### 3.1 Functional Requirements

#### 3.1.1 User Authentication and Management

**FR-001: User Registration**

**Description:** The system shall allow new users to create accounts with email and password.

**Priority:** High

**Inputs:** Email address, password, confirm password, full name, phone number

**Processing:** Validate email format, check password strength, verify email uniqueness, hash password

**Outputs:** User account creation confirmation, welcome email

**FR-002: User Login**

**Description:** The system shall authenticate users using email and password credentials.

**Priority:** High

**Inputs:** Email address, password

**Processing:** Validate credentials, generate JWT token, establish session

**Outputs:** Authentication token, user dashboard access

**FR-003: Password Management**

**Description:** The system shall provide password reset functionality via email.

**Priority:** Medium

**Inputs:** Email address for reset request

**Processing:** Generate reset token, send email with reset link

**Outputs:** Password reset email, secure password update form

**FR-004: Family Profile Management**

**Description:** The system shall allow users to create and manage profiles for family members.

**Priority:** High

**Inputs:** Family member name, relationship, date of birth, medical notes

**Processing:** Create linked profile, establish caregiver relationship

**Outputs:** Family member profile, access permissions

#### 3.1.2 Prescription Management

**FR-005: Prescription Upload**

**Description:** The system shall allow users to upload prescriptions via manual entry or image upload.

**Priority:** High

**Inputs:** Prescription details (medication name, dosage, frequency, doctor name, date) or prescription image

**Processing:** Validate prescription data, store image in cloud storage, create database record

**Outputs:** Prescription confirmation, unique prescription ID

**FR-006: Prescription Storage and Organization**

**Description:** The system shall store prescriptions organized by user, date, and doctor.

**Priority:** High

**Inputs:** Prescription data, user ID, metadata

**Processing:** Encrypt sensitive data, create indexes, maintain relationships

**Outputs:** Organized prescription database, searchable records

**FR-007: Prescription Viewing and Management**

**Description:** The system shall allow users to view, edit, and delete their prescriptions.

**Priority:** High

**Inputs:** User request, prescription ID, updated data

**Processing:** Authorize access, validate changes, update records

**Outputs:** Prescription details, update confirmation

**FR-008: Prescription Search and Filter**

**Description:** The system shall provide search and filter capabilities for prescriptions.

**Priority:** Medium

**Inputs:** Search terms, filter criteria (date range, doctor, medication type)

**Processing:** Query database, apply filters, sort results

**Outputs:** Filtered prescription list, search results

#### 3.1.3 Medication Reminder System

**FR-009: Reminder Creation**

**Description:** The system shall allow users to set automated medication reminders.

**Priority:** High

**Inputs:** Prescription ID, reminder times, frequency, duration

**Processing:** Calculate reminder schedule, create recurring events

**Outputs:** Reminder schedule confirmation, active reminder list

**FR-010: Reminder Notifications**

**Description:** The system shall send notifications via email and in-app alerts.

**Priority:** High

**Inputs:** Scheduled reminder, user preferences

**Processing:** Generate notification, deliver via configured channels

**Outputs:** Email notification, in-app alert

**FR-011: Reminder Management**

**Description:** The system shall allow users to acknowledge, snooze, or skip reminders.

**Priority:** Medium

**Inputs:** User action (acknowledge, snooze, skip), reminder ID

**Processing:** Update reminder status, reschedule if needed, log action

**Outputs:** Updated reminder status, adherence record

#### 3.1.4 Administrative Functions

**FR-012: User Management**

**Description:** The system shall provide admin capabilities to view and manage users.

**Priority:** Medium

**Inputs:** Admin credentials, user management actions

**Processing:** Verify admin permissions, execute user operations

**Outputs:** User list, management confirmations

**FR-013: System Monitoring**

**Description:** The system shall provide monitoring capabilities for prescription activity.

**Priority:** Low

**Inputs:** Admin request, monitoring parameters

**Processing:** Query activity logs, generate reports

**Outputs:** Activity reports, system health metrics

### 3.2 Non-Functional Requirements

#### 3.2.1 Performance Requirements

**NFR-001: Response Time**

**Description:** The system shall respond to user requests within 3 seconds under normal load.

**Priority:** High

**Metric:** 95% of requests completed within 3 seconds

**NFR-002: Throughput**

**Description:** The system shall support at least 100 concurrent users.

**Priority:** Medium

**Metric:** 100 concurrent active sessions without performance degradation

**NFR-003: Scalability**

**Description:** The system shall be designed to scale horizontally with increased load.

**Priority:** Medium

**Metric:** Ability to add server instances to handle increased traffic

#### 3.2.2 Security Requirements

**NFR-004: Authentication Security**

**Description:** The system shall use JWT-based authentication with bcrypt password hashing.

**Priority:** High

**Implementation:** JWT tokens, bcrypt hashing, session management

**NFR-005: Data Encryption**

**Description:** The system shall encrypt sensitive health data at rest and in transit.

**Priority:** High

**Implementation:** HTTPS, database encryption, encrypted file storage

**NFR-006: Access Control**

**Description:** The system shall implement role-based access control (RBAC).

**Priority:** High

**Implementation:** User roles, permission matrices, authorization middleware

#### 3.2.3 Usability Requirements

**NFR-007: Accessibility**

**Description:** The system shall be accessible to elderly users with appropriate UI design.

**Priority:** High

**Features:** Large fonts, high contrast, clear icons, simple navigation

**NFR-008: Responsiveness**

**Description:** The system shall provide responsive design for various screen sizes.

**Priority:** High

**Support:** Desktop, tablet, and mobile device compatibility

**NFR-009: User Experience**

**Description:** The system shall provide intuitive user interface with minimal learning curve.

**Priority:** High

**Metrics:** New users can complete basic tasks within 10 minutes

#### 3.2.4 Reliability Requirements

**NFR-010: Availability**

**Description:** The system shall maintain 95% uptime availability.

**Priority:** High

**Metric:** Maximum 36 hours downtime per month

**NFR-011: Data Backup**

**Description:** The system shall perform automated daily backups of all user data.

**Priority:** High

**Implementation:** MongoDB Atlas automated backups, 30-day retention

**NFR-012: Error Handling**

**Description:** The system shall provide graceful error handling and recovery.

**Priority:** Medium

**Features:** User-friendly error messages, automatic retry mechanisms

### 3.3 Interface Requirements

#### 3.3.1 User Interface Requirements

* Clean, modern web interface using React.js and TailwindCSS/MUI
* Responsive design compatible with desktop and mobile browsers
* Accessibility features for elderly users (large fonts, high contrast)
* Intuitive navigation with breadcrumb trails
* Form validation with real-time feedback
* Loading indicators for long-running operations

#### 3.3.2 Hardware Interface Requirements

* Standard web browser environment (no special hardware required)
* Camera access for prescription image capture (optional)
* Audio support for notification alerts

#### 3.3.3 Software Interface Requirements

* MongoDB Atlas database interface
* Cloudinary/Firebase API for file storage
* Email service provider API for notifications
* JWT library for authentication
* bcrypt library for password hashing

#### 3.3.4 Communication Interface Requirements

* HTTPS protocol for all client-server communication
* RESTful API endpoints for frontend-backend communication
* SMTP protocol for email notifications
* WebSocket connections for real-time notifications (future enhancement)

## 4. System Models

### 4.1 System Architecture

The MedMate system follows a three-tier architecture pattern:

#### 4.1.1 Presentation Tier (Frontend)

* **Technology:** React.js with Redux/Zustand for state management
* **Styling:** TailwindCSS or Material-UI for responsive design
* **Hosting:** Vercel cloud platform
* **Responsibilities:** User interface rendering, user interaction handling, client-side validation

#### 4.1.2 Application Tier (Backend)

* **Technology:** Node.js with Express.js framework
* **Authentication:** JWT-based token management
* **Hosting:** Render or Heroku cloud platform
* **Responsibilities:** Business logic, API endpoints, authentication, data validation

#### 4.1.3 Data Tier (Database)

* **Technology:** MongoDB with MongoDB Atlas cloud hosting
* **File Storage:** Cloudinary or Firebase for prescription images
* **Responsibilities:** Data persistence, data integrity, backup and recovery

### 4.2 Data Models

#### 4.2.1 User Entity

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Type** | **Description** | **Constraints** |
| \_id | ObjectId | Unique user identifier | Primary key, auto-generated |
| email | String | User email address | Required, unique, valid email format |
| password | String | Hashed password | Required, bcrypt hashed |
| firstName | String | User first name | Required, 2-50 characters |
| lastName | String | User last name | Required, 2-50 characters |
| phoneNumber | String | Contact phone number | Optional, valid phone format |
| role | String | User role | Enum: patient, caregiver, admin |
| createdAt | Date | Account creation date | Auto-generated |
| updatedAt | Date | Last update date | Auto-updated |

#### 4.2.2 Prescription Entity

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Type** | **Description** | **Constraints** |
| \_id | ObjectId | Unique prescription identifier | Primary key, auto-generated |
| userId | ObjectId | Reference to user | Required, foreign key |
| medicationName | String | Name of medication | Required, 2-100 characters |
| dosage | String | Medication dosage | Required |
| frequency | String | How often to take | Required |
| doctorName | String | Prescribing doctor | Required |
| prescriptionDate | Date | Date prescription was issued | Required |
| imageUrl | String | URL to prescription image | Optional |
| instructions | String | Special instructions | Optional |
| isActive | Boolean | Whether prescription is active | Default: true |

#### 4.2.3 Reminder Entity

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Type** | **Description** | **Constraints** |
| \_id | ObjectId | Unique reminder identifier | Primary key, auto-generated |
| prescriptionId | ObjectId | Reference to prescription | Required, foreign key |
| userId | ObjectId | Reference to user | Required, foreign key |
| reminderTime | Date | When to send reminder | Required |
| isRecurring | Boolean | Whether reminder repeats | Default: false |
| frequency | String | Recurrence pattern | Enum: daily, weekly, monthly |
| status | String | Reminder status | Enum: active, completed, snoozed |
| lastSent | Date | Last notification sent | Optional |

### 4.3 API Specifications

#### 4.3.1 Authentication APIs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Endpoint** | **Method** | **Description** | **Request Body** | **Response** |
| /api/auth/register | POST | User registration | email, password, firstName, lastName | User object, JWT token |
| /api/auth/login | POST | User login | email, password | User object, JWT token |
| /api/auth/logout | POST | User logout | None | Success message |
| /api/auth/reset-password | POST | Password reset request | email | Success message |

#### 4.3.2 Prescription APIs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Endpoint** | **Method** | **Description** | **Parameters** | **Response** |
| /api/prescriptions | GET | Get user prescriptions | userId, page, limit | Array of prescriptions |
| /api/prescriptions | POST | Create prescription | Prescription data | Created prescription |
| /api/prescriptions/:id | PUT | Update prescription | Prescription updates | Updated prescription |
| /api/prescriptions/:id | DELETE | Delete prescription | None | Success message |

#### 4.3.3 Reminder APIs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Endpoint** | **Method** | **Description** | **Parameters** | **Response** |
| /api/reminders | GET | Get user reminders | userId | Array of reminders |
| /api/reminders | POST | Create reminder | Reminder data | Created reminder |
| /api/reminders/:id/acknowledge | PUT | Acknowledge reminder | None | Updated reminder |
| /api/reminders/:id/snooze | PUT | Snooze reminder | snooze duration | Updated reminder |

## 5. Appendices

### 5.1 Project Timeline and Milestones

|  |  |  |  |
| --- | --- | --- | --- |
| **Phase** | **Duration** | **Key Deliverables** | **Responsible Team Member** |
| Project Management | 50 days | Project plan, risk management, communication coordination | Muhammad Hasan Khan |
| Requirements & System Design | 30 days | Requirements analysis, system architecture, data models | Muhammad Mujtaba |
| Backend Development | 15 days | Database setup, API development, authentication | Muhammad Mujtaba |
| Frontend Development | 30 days | UI/UX design, React components, backend integration | Rayyan Ibrahim |
| Testing & QA | 4 days | Unit testing, system testing, user acceptance testing | Husain Ahmad Khan |
| Deployment | 1 day | Production deployment, security configuration | Muhammad Hamza Abbas |
| Documentation | 6 days | Technical documentation, user training materials | Muhammad Hasan Khan |

### 5.2 Risk Assessment Matrix

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Risk** | **Probability** | **Impact** | **Risk Level** | **Mitigation Strategy** |
| Data Breaching | Low | High | High | Strong encryption, secure authentication, regular security audits |
| Server Crash | Medium | High | High | Reliable cloud platform, automated backups, monitoring |
| Timeline Slippage | Medium | Medium | Medium | Weekly sprints, clear milestones, regular progress reviews |
| Low User Adoption | High | Medium | High | User-friendly design, pilot testing, feedback collection |
| Browser Compatibility | Low | Low | Low | Cross-browser testing, responsive design, progressive enhancement |
| Medical Misuse | Medium | High | High | Clear disclaimers, no medical advice, user education |

### 5.3 Quality Assurance Plan

#### 5.3.1 Testing Strategy

* **Unit Testing:** Individual component and function testing using Jest and React Testing Library
* **Integration Testing:** API endpoint testing using Postman and automated test suites
* **System Testing:** End-to-end testing of complete user workflows
* **Performance Testing:** Load testing with simulated concurrent users
* **Security Testing:** Authentication, authorization, and data protection validation
* **Usability Testing:** User experience testing with target audience groups
* **Compatibility Testing:** Cross-browser and device compatibility verification

#### 5.3.2 Acceptance Criteria

* All functional requirements successfully implemented and tested
* System performance meets specified response time requirements
* Security measures implemented and validated
* User interface passes accessibility and usability testing
* System successfully deployed to production environment
* Documentation completed and reviewed
* Stakeholder approval obtained for all deliverables

### 5.4 Technology Stack Details

#### 5.4.1 Frontend Technologies

* **React.js 18+:** Component-based UI library for building interactive interfaces
* **Redux/Zustand:** State management for complex application state
* **TailwindCSS/Material-UI:** CSS framework for responsive and accessible design
* **React Router:** Client-side routing for single-page application navigation
* **Axios:** HTTP client for API communication
* **React Hook Form:** Form handling and validation library

#### 5.4.2 Backend Technologies

* **Node.js 16+:** JavaScript runtime for server-side development
* **Express.js:** Web application framework for RESTful API development
* **MongoDB:** NoSQL database for flexible data storage
* **Mongoose:** MongoDB object modeling library for Node.js
* **JSON Web Token (JWT):** Token-based authentication mechanism
* **bcrypt:** Password hashing library for secure authentication
* **Cloudinary/Firebase:** Cloud storage for prescription images

#### 5.4.3 Development and Deployment Tools

* **Git/GitHub:** Version control and collaborative development
* **GitHub Actions:** Continuous integration and deployment pipeline
* **Vercel:** Frontend hosting platform with global CDN
* **Render/Heroku:** Backend hosting with automatic scaling
* **MongoDB Atlas:** Managed MongoDB cloud service
* **Postman:** API development and testing tool

### 5.5 Security Measures

#### 5.5.1 Data Protection

* All sensitive data encrypted using AES-256 encryption
* HTTPS enforced for all client-server communication
* Password hashing using bcrypt with salt rounds
* JWT tokens with expiration and refresh mechanisms
* Input validation and sanitization to prevent injection attacks
* CORS configuration to restrict cross-origin requests

#### 5.5.2 Access Control

* Role-based access control (RBAC) implementation
* Session management with secure token handling
* User permission validation for all protected routes
* Audit logging for security-sensitive operations
* Rate limiting to prevent abuse and DoS attacks

#### 5.5.3 Privacy Compliance

* Clear privacy policy and terms of service
* User consent mechanisms for data collection
* Data minimization principles applied
* Right to data deletion and export capabilities
* Medical disclaimers to prevent misuse of health information

### 5.6 Glossary

**Adherence**

The extent to which patients take medications as prescribed by healthcare providers

**Caregiver**

A person who provides care and support to family members, particularly children or elderly relatives

**Digital Prescription**

An electronic version of a traditional paper prescription that can be stored and managed digitally

**Family Profile**

A user account feature that allows management of multiple family members' medical information from a single login

**Medication Reminder**

An automated notification system that alerts users when it's time to take their prescribed medications

**MERN Stack**

A full-stack JavaScript framework consisting of MongoDB, Express.js, React.js, and Node.js

**RESTful API**

A software architecture style that defines a set of constraints for creating web services

**Responsive Design**

A web design approach that makes web pages render well on various devices and screen sizes

### 5.7 Supporting Information

This document serves as the comprehensive requirements specification for the MedMate Smart Prescription & Medicine Tracker project. It has been developed by Team Orbit for the CS-458 course at the Department of Computer Science, UBIT.

For additional information or clarification regarding any requirements specified in this document, please contact the project team at the provided contact information or through the course instructor, Miss Maryam Feroze.

**Document Version:** 1.0

**Last Updated:** August 28, 2025

**Review Date:** As per project milestones

**Approval:** Pending stakeholder review