**Software Requirements**

**Specification**

**for**

**HealthMate**

**Version 1.0 approved**

**Prepared by Cherbane Akram**

**Younsi Chakib**

**Benziada Moncef**

**Larik Rihab**

**Nadji Fares**

**Numidia Institute of Technology**

**Copyright © 1999 by Karl E. Wiegers. Permission is granted to use, modify, and distribute this document.**

**Software Requirements Specification for HealthMate Page ii**

# Table of Contents

[Table of Contents ii](#_Toc8033)

[Revision History ii](#_Toc8034)

[1. Introduction 1](#_Toc8035)

[1.1 Purpose 1](#_Toc8036)

[1.2 Document Conventions 1](#_Toc8037)

[1.3 Intended Audience and Reading Suggestions 1](#_Toc8038)

[1.4 Product Scope 1](#_Toc8039)

[1.5 References 2](#_Toc8040)

[2. Overall Description 2](#_Toc8041)

[2.1 Product Perspective 2](#_Toc8042)

[2.2 Product Functions 3](#_Toc8043)

[2.3 User Classes and Characteristics 3](#_Toc8044)

[2.4 Operating Environment 3](#_Toc8045)

[2.5 Design and Implementation Constraints 4](#_Toc8046)

[2.6 User Documentation 4](#_Toc8047)

[2.7 Assumptions and Dependencies 4](#_Toc8048)

[3. External Interface Requirements 5](#_Toc8049)

[3.1 User Interfaces 5](#_Toc8050)

[3.2 Hardware Interfaces 5](#_Toc8051)

[3.3 Software Interfaces 6](#_Toc8052)

[3.4 Communications Interfaces 6](#_Toc8053)

[4. System Features 7](#_Toc8054)

[4.1 System Feature 1 7](#_Toc8055)

[4.2 System Feature 2 (and so on) 8](#_Toc8056)

[5. Other Nonfunctional Requirements 10](#_Toc8057)

[5.1 Performance Requirements 10](#_Toc8058)

[5.2 Safety Requirements 10](#_Toc8059)

[5.3 Security Requirements 10](#_Toc8060)

[5.4 Software Quality Attributes 11](#_Toc8061)

[5.5 Business Rules 11](#_Toc8062)

[6. Other Requirements 12](#_Toc8063)

[Appendix A: Glossary 12](#_Toc8064)

[Appendix B: Analysis Models 13](#_Toc8065)

[Appendix C: To Be Determined List 13](#_Toc8066)

# 1. Introduction

## 1.1 Purpose

## *HealthMate is a mobile application designed to provide medical diagnosis assistance through an AI chatbot. This document specifies the requirements for version 1.0 of HealthMate. The scope of this SRS includes the entire system, encompassing the frontend built with Flutter, backend implemented using FastAPI, Firebase for authentication, and the Random Forest-based AI diagnosis model.*

## 1.2 Document Conventions

## *This SRS document adheres to IEEE standards for software documentation. It uses hierarchical section numbering (e.g., 1.1, 1.2) for organization. Priorities for requirements are explicitly stated where applicable. Functional requirements are labeled with "REQ" identifiers for traceability, and non-functional requirements are clearly distinguished.*

## 1.3 Intended Audience and Reading Suggestions

## The intended audience for this document includes:

## Developers: To understand the technical requirements for implementation.

## Project Managers: To ensure alignment with project goals and scope.

## Testers: To develop test cases based on requirements.

## Users and Stakeholders: To verify that the application meets their needs.

## Regulatory Authorities: To ensure compliance with healthcare data standards.

## Readers are encouraged to begin with Section 1 for an overview, proceed to Section 2 for system context, and review detailed requirements in Sections 3 and 4.

## 1.4 Product Scope

## *HealthMate aims to assist users in receiving quick and accurate medical insights based on symptoms. Users input their symptoms, and the AI-powered chatbot provides a probable diagnosis, detailed explanations, and emergency orientation if necessary. The application is designed for the general public, emphasizing accessibility, security, and responsiveness. HealthMate aligns with corporate goals of leveraging AI to enhance healthcare accessibility and aligns with standards such as HIPAA for data security.*

## 1.5 References

# 1. IEEE Software Requirements Specification Template

# 2. Flutter Framework Documentation: <https://flutter.dev>

# 3. Firebase Authentication Documentation: https://firebase.google.com/docs/auth

# 4. FastAPI Documentation: https://fastapi.tiangolo.com

# *5. HIPAA Compliance Guidelines:* [*https://www.hhs.gov/hipaa*](https://www.hhs.gov/hipaa)

# 2. Overall Description

## 2.1 Product Perspective

## HealthMate is a new, self-contained mobile application designed to enhance accessibility to medical diagnoses. It does not build on any prior product family but instead integrates AI, mobile app development, and healthcare technologies to create a unique user experience. The system comprises the following key components:

## Frontend: Built with Flutter, offering cross-platform support for Android and iOS.

## Backend: A FastAPI-powered service that manages AI predictions and database interactions.

## AI Model: A Random Forest algorithm trained to process user-provided symptoms and provide probable diagnoses.

## Authentication: Managed using Firebase to ensure secure and seamless user registration and login. The system interfaces include RESTful APIs for communication between the mobile frontend and backend, and Firebase for authentication.

## 2.2 Product Functions

## HealthMate facilitates the following major functions:

## User registration and login, with options for email/password or Google authentication.

## AI-driven symptom analysis and prediction of potential diagnoses.

## Detailed explanations and emergency orientation based on AI predictions.

## A user-friendly interface for inputting symptoms and accessing results.

## Secure storage and management of user data, compliant with healthcare standards

## 2.3 User Classes and Characteristics

## The primary user classes for HealthMate include:

## General Users: Individuals seeking quick, symptom-based health insights. These users typically have minimal technical expertise and are looking for ease of use.

## Healthcare Enthusiasts: Users who may leverage the app for preliminary self-diagnosis before consulting medical professionals.

## Regulatory Authorities and Organizations: Focused on verifying compliance with healthcare data and security standards.

## Each user class requires specific features to ensure their satisfaction and usability.

## 2.4 Operating Environment

## HealthMate operates in the following environment:

## Hardware: Mobile devices supporting Android (API Level 21 or above) and iOS (12.0 or later).

## Software: Built using Flutter for the frontend, FastAPI for the backend, and Firebase for authentication.

## Network: Requires stable internet connectivity for real-time API interactions.

## 2.5 Design and Implementation Constraints

## The development of HealthMate is subject to the following constraints:

## Compliance: Adherence to HIPAA and other healthcare data security standards.

## Technological Constraints: Use of Flutter, FastAPI, Firebase, and the Random Forest algorithm for core functionalities.

## Performance: Ensuring low latency in AI predictions and seamless app performance.

## Security: Implementation of data encryption and secure communication protocols.

## 2.6 User Documentation

## The following documentation will be delivered alongside HealthMate:

## User Manuals: Detailed instructions on app usage.

## In-App Tutorials: Interactive guides for first-time users.

## FAQs: A comprehensive list of frequently asked questions and troubleshooting tips. Documentation will be available in digital formats compatible with the app.

## 2.7 Assumptions and Dependencies

# The project is based on the following assumptions and dependencies:

# Users will have access to mobile devices with stable internet connectivity.

# Firebase services will remain available and functional.

# The Random Forest model will provide accurate and timely predictions.

# Third-party libraries and APIs used in development will maintain backward compatibility.

# 3. External Interface Requirements

## 3.1 User Interfaces

## The user interface (UI) for HealthMate is structured around a seamless and intuitive chatting experience, featuring:

## Login and Authentication Page: Allows users to log in with their credentials or authenticate via Google.

## Chatting Page: The central interface where users interact with the AI chatbot. This page includes:

## Message Input Field: A text box for users to enter their symptoms or queries.

## Chat History Display: Messages are displayed in a conversational format, showing both user inputs and chatbot responses.

## Send Button: For sending messages to the chatbot.

## Diagnosis Summary View: A separate view or inline response from the chatbot that provides a summary of the diagnosis, additional details, and emergency orientation if necessary.

## Error Handling: Standardized responses and UI prompts for scenarios such as failed API responses or invalid inputs.

## The design follows material design guidelines for consistency and user familiarity. Further UI specifications, including layouts and interaction flows, are documented separately.

## 3.2 Hardware Interfaces

## HealthMate interfaces with the following hardware components:

## Mobile Devices: Supports Android (API Level 21 or later) and iOS (12.0 or later) devices.

## Input/Output: Relies on the device’s touchscreen for interaction and display for chatting functionalities.

## No additional peripherals are required.

## 3.3 Software Interfaces

## HealthMate interacts with multiple software components to enable its functionalities:

## Backend API: Communicates with the FastAPI backend via RESTful APIs to handle chatbot interactions and AI diagnosis requests.

## Firebase: Provides secure user authentication and management services.

## Chatbot Engine: Manages the conversational flow between the user and the AI model. The Random Forest algorithm processes symptom inputs and predicts potential diagnoses, which are formatted into natural language responses.

## Third-Party Libraries: Flutter libraries are used to support UI components and handle network requests.

## Data Flow:

## Input: User messages sent through the chat interface.

## Output: Chatbot responses, including diagnostic predictions and supporting information. All communication is encrypted using HTTPS, and API specifications for the chatbot interactions are documented separately.

## 3.4 Communications Interfaces

# The communication functionalities include:

# Protocols: All communications between the app and backend use secure HTTPS to ensure encryption and data integrity.

# Chat System: Real-time exchange of messages between the user and the chatbot, optimized for low latency to simulate a natural conversation.

# Error Messaging: The chatbot provides user-friendly error messages if issues occur, such as a failure to understand input or an API timeout.

# Email Integration: Supports account-related communications, including verification and password recovery.

# All communications adhere to RESTful API standards and industry encryption practices (e.g., TLS 1.2 or higher).

# 4. System Features

## This section outlines the major features of HealthMate, focusing on its core functionalities and user interactions.

## 4.1 System Feature 1

#### **4.1.1 Description and Priority**

#### This feature allows users to engage in a conversational interface with the AI chatbot, providing symptoms as text input. The chatbot processes the input using a Random Forest-based AI model and returns a probable diagnosis with additional explanations or emergency guidance if necessary.

#### **Priority**: High

#### **Component Ratings**:

#### **Benefit**: 9 (Crucial for application functionality)

#### **Penalty**: 8 (Critical if unavailable)

#### **Cost**: 5 (Moderate cost to implement and maintain)

#### **Risk**: 6 (Moderate technical challenges and dependencies)

#### **4.1.2 Stimulus/Response Sequences**

#### **Stimulus**: User logs into the app and navigates to the chat interface. **Response**: The app loads the chatbot page with an input field and displays a welcome message.

#### **Stimulus**: User inputs symptoms (e.g., "I have a fever and headache"). **Response**: The app sends the input to the backend via API, processes the symptoms, and displays a diagnosis response (e.g., "The symptoms indicate a possible case of influenza. Please consult a doctor if symptoms persist.").

#### **Stimulus**: User asks for additional details or clarification. **Response**: The chatbot provides explanatory information or emergency recommendations if necessary.

#### **Stimulus**: Invalid input (e.g., "1234" or nonsensical text). **Response**: The chatbot prompts the user to re-enter symptoms with a clear message (e.g., "I couldn't understand that. Please provide valid symptoms.").

#### **4.1.3 Functional Requirements**

#### **REQ-1**: The system must accept textual symptom inputs from the user.

#### **REQ-2**: The backend must process symptom data using the Random Forest AI model and return a diagnosis.

#### **REQ-3**: The chatbot must format AI responses into conversational text before presenting them to the user.

#### **REQ-4**: The system must handle invalid or incomplete inputs by prompting the user with appropriate messages.

#### **REQ-5**: The chatbot must be capable of providing additional details or emergency guidance based on the diagnosis.

#### **REQ-6**: The system must ensure low-latency responses for seamless user interaction.

#### **REQ-7**: All communication must be encrypted and secure to maintain user privacy and data security.

**4.2 User Authentication and Account Management**

**4.2.1 Description and Priority**

This feature handles user registration, login, and account management, ensuring secure access to the application.

* **Priority**: High
* **Component Ratings**:
  + **Benefit**: 9 (Essential for controlled access)
  + **Penalty**: 8 (Significant if unavailable)
  + **Cost**: 4 (Relatively low implementation cost with Firebase)
  + **Risk**: 3 (Low risk due to Firebase's reliability)

**4.2.2 Stimulus/Response Sequences**

1. **Stimulus**: User selects "Sign Up" and enters personal details (e.g., name, email, password).  
   **Response**: The system registers the user and confirms account creation.
2. **Stimulus**: User selects "Log In" and provides credentials.  
   **Response**: The system verifies the credentials and grants access to the main interface.
3. **Stimulus**: User opts for Google authentication.  
   **Response**: The system authenticates via Google and logs the user in.
4. **Stimulus**: Password recovery request.  
   **Response**: The system sends a password reset link to the registered email.

**4.2.3 Functional Requirements**

* **REQ-1**: The system must allow users to register with their name, email, and password or via Google.
* **REQ-2**: The system must authenticate users securely using Firebase Authentication.
* **REQ-3**: The system must enable password recovery through email.
* **REQ-4**: User account data must be securely stored and encrypted.
* **REQ-5**: The system must handle authentication errors gracefully with user-friendly messages.

# 5. Other Nonfunctional Requirements

## 5.1 Performance Requirements

## The system must respond to user inputs on the chatbot interface within 2 seconds under normal network conditions to maintain a conversational flow.

## Backend API calls, including AI model predictions, must process and return responses within 1 second.

## The app must support at least 1,000 concurrent users without significant degradation in performance.

## The system should handle 500 API requests per second during peak usage with a maximum latency of 2 seconds.

## These performance requirements ensure a responsive and efficient user experience, particularly in real-time chatbot interactions.

## 5.2 Safety Requirements

## The app must provide clear disclaimers that diagnoses are for informational purposes only and not a substitute for professional medical advice.

## Any emergency orientation must explicitly guide users to seek professional medical help in urgent situations.

## The system must ensure no unintended data modification or loss during data transmission or storage.

## The app must include safeguards to prevent misuse, such as restricting AI responses to valid medical contexts only.

## Compliance with healthcare safety standards (e.g., HIPAA) is mandatory.

## These safeguards are essential to prevent harm due to misinterpretation of the AI's outputs or system failures.

## 5.3 Security Requirements

## All communications between the app and backend must be encrypted using TLS 1.2 or higher.

## User authentication must be managed through Firebase, ensuring secure login and account management.

## Sensitive user data, such as medical information and personal details, must be stored in an encrypted format.

## The app must log all access attempts and notify users of suspicious activity.

## The system must comply with HIPAA standards for handling healthcare data, ensuring user privacy and security.

## Sessions must automatically time out after 15 minutes of inactivity to prevent unauthorized access.

## These security measures ensure the protection of user data and maintain trust in the application.

## 5.4 Software Quality Attributes

## Adaptability: The app must support new medical conditions and AI model updates without requiring major system rework.

## Availability: The system must maintain an uptime of 99.9%, ensuring availability during peak usage.

## Correctness: Diagnoses must be accurate based on the trained Random Forest model, with periodic validation to maintain reliability.

## Flexibility: The architecture must support adding new features with minimal disruption to existing functionality.

## Interoperability: The app must integrate seamlessly with Firebase, FastAPI, and the underlying AI model.

## Maintainability: Codebase must be modular, well-documented, and follow industry coding standards for easy maintenance.

## Portability: The app must function on both Android and iOS platforms without significant platform-specific issues.

## Reliability: The system must handle 99.5% of requests successfully without crashing or errors.

## *Usability: The app must score at least 85% in usability testing, ensuring that users find it intuitive and easy to use.*

## 5.5 Business Rules

# Only authenticated users may access the chatbot and diagnosis functionalities.

# The app must restrict access to user data to the respective user and authorized administrators.

# Diagnoses provided by the app must be presented with clear disclaimers, emphasizing their informational nature.

# Users must agree to the terms and conditions, including privacy policies, before registering.

# The app must enforce the use of strong passwords for account creation.

# These business rules guide the operational principles of HealthMate and ensure compliance with ethical and legal standards.

# 6. Other Requirements

# Database Requirements:

# The system must use a secure, scalable cloud-based database to store user information, medical data, and chatbot logs.

# Sensitive data, such as user credentials, must be encrypted using AES-256 encryption.

# The database must allow real-time updates and synchronization with the app backend.

# Internationalization Requirements:

# The app must support multiple languages, including English, French, and Arabic.

# All text in the app must be easily translatable using resource files.

# Legal Requirements:

# The app must comply with HIPAA for handling healthcare-related data.

# Terms of Service and Privacy Policy documents must be accessible within the app, and users must accept them during registration.

# Reuse Objectives:

# The Random Forest AI model and the backend API architecture must be designed for reuse in future health-related applications.

# Modular components, such as the chatbot interface and Firebase integration, must be reusable for other projects.

# Scalability:

# The architecture must support scaling up to accommodate increasing numbers of users and concurrent requests.

# Appendix A: Glossary

# AI: Artificial Intelligence. Technology that simulates human intelligence for decision-making and predictions.

# HIPAA: Health Insurance Portability and Accountability Act. A US regulation ensuring data privacy and security in healthcare.

# Random Forest: A machine learning algorithm used for classification and regression tasks.

# RESTful API: A type of web service that uses standard HTTP methods for interaction between systems.

# Firebase: A platform developed by Google for creating mobile and web applications, used here for authentication.

# FastAPI: A modern Python-based web framework for building APIs.

# TLS*:* Transport Layer Security. A protocol for encrypting internet communications.

# Appendix B: Analysis Models

# Data Flow Diagram (DFD): Illustrates the flow of data between users, the chatbot, backend, and database.

# Class Diagram: Shows key components like User, Chatbot, AIModel, and their interactions.

# State-Transition Diagram: Details the app's states, such as Login, Chatting, and Viewing Results, and transitions triggered by user actions.

# Entity-Relationship Diagram (ERD): Represents the relationships between database entities like Users, Diagnoses, and Logs.

# These models are maintained separately and can be provided upon request.

# Appendix C: To Be Determined List

1. Additional languages to be supported in future versions.

2. Integration with wearable devices for enhanced data collection.

3. AI model updates to include more complex diagnoses and new datasets.

4. Potential partnerships with healthcare organizations for validation.

5. Legal consultation for compliance with international healthcare data regulations beyond HIPAA.