Software Requirements Specification

for

MedVoyage

Version 1.0 approved.

Prepared by Yasmeen Haleem, Bhavani Rachakatla, Vidhi Bhatt, Pravallika Bollavaram, Manushree Buyya, Emmie Abels, Demir Altay, Kalyan Cheerla

Team Squad8

10/02/2023

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Yasmeen Haleem, Bhavani Rachakatla, Pravallika Bollavaram, Manushree Buyya, Vidhi Bhatt, Emmie Abels, Demir Altay, Kalyan Cheerla	10/02/2023	Initial Draft	1.0

1. Introduction

1.1 Purpose

The purpose of this document is to elaborate the specific requirements the product (online patient /clinic/doctor booking system) serves to implement where the developers have kept the necessary stakeholders in mind. This project seeks to be an easy and convenient online patient booking system, where patients, health care professionals such as physicians or doctor's teams can access information pertinent to everyone. It is a valuable resource for patients to effortlessly sign up and schedule a doctor's appointment by observing schedules of doctors that match their preferences.

This document is necessary as it provides any stakeholders such as developers, patients, or health care professionals, the software this project intends to utilize and the necessary functionalities aiming to satisfy all users.

Overall, this allows a smooth transition with ease to support patients and their needs, and healthcare professionals to document patient information. The scope of this project is a simple and functioning online web application for medical appointments which the *"Project Scope*" section elaborates in detail in terms of corporate/business goals and purpose or benefits of the product. The developers, engineers, and other teams will be able to access revisions and specific requirements made in this document. This requirements record is the official catalog of the software functionalities and requirements for this online patient booking system project that the developers of this project aim to fulfill.

1.2 Document Conventions

The standards for this document are simple and all specific requirements are stated through equal priority levels with bolded fonts to emphasize heading or sub-headings. The glossary items are provided at the end of the document and are in bold to highlight important terms that may be new to different users or individuals reading the document. Important headings, subheadings, diagram references are labelled in bold and italics to highlight the significant sections of the project.

1.3 Intended Audience and Reading Suggestions

This document is intended for developers, users, testers, project and risk managers, possible investors, possible marketing partners or experts. This requirements document contains the project description, project significance and purpose, possible stakeholders, functional/non-functional software requirements, and several interfaces and system requirements pertinent to a functioning web-based application.

The initial sections, such as one and some of two, address project purpose, description, overall significance of the project involving few features and constraints, references, project scope, stakeholders involved, top-level requirements through visuals, and important glossary items.

Sections one and two of these documents may be useful to various users intending to utilize this application or marketing staff that intends to highlight this project's usability and ease of use. All the sections, but most importantly, sections two to six are suggested for developers, risk and project managers, testers, investors, etc. to monitor and understand specific system requirements and analysis.

1.4 Project Scope

Many individuals have issues connecting with doctor's offices to book appointments and get scheduled as quickly and efficiently as possible. The project involves a web-based application system, particularly, an online patient booking system to allow easy sign-up for medical appointments based on preferences and needs. Simply summarizing, patients will sign up for the system, enter significant personal information protected by our protection encryption and data protection systems, and select physician(s) that meet their medical, location, time, expenses, etc. needs.

The objective is to provide a very simple web-based application sign up and useability that enhances user experience and enables users to sign up and make appointments within minutes. For now, we plan to input the system typically in the Dallas-Fort Worth area with possibilities of extending accessibility around the globe.

The patients should be able to view appointments close by and during the times they can manage in their busy schedules. The business strategies we plan to implement are enhanced user accessibility, useability, and customer satisfaction by understanding our target audience needs (patient or health care professional), while marketing our ease of use through core software functionalities that improve fast loading, fast sign-ups, customized time schedules and preferences of an individual patient or healthcare provider. Succinctly, this strategy helps meet our corporate goals of improving in person or call-in appointment systems already in place and deliver valuable user experience for a crucial necessity of healthcare.

1.5 References

Squad8's MedVoyage Project Proposal:
"SE Group." *To SharePoint – MedVoyage Project Proposal Microsoft Support*, support.microsoft.com/en-us/office/sign-in-to-sharepoint-324a89ec-e77b-4475-b64a-13a0c14c45ec. Accessed 1 Oct. 2023.
https://myunt.sharepoint.com/:b:/s/CSCE5430-SEGroupProject/ES7EwfrnAkhCp6lQ9-7nkZEBQ2CjCve6QW-dqJisjgDq6g?e=9ipjek

Squad8's MedVoyage Project Proposal Presentation:
"SE Group." *To SharePoint – MedVoyage Project Proposal Presentation Microsoft Support*, support.microsoft.com/en-us/office/sign-in-to-sharepoint-324a89ec-e77b-4475-b64a-13a0c14c45ec. Accessed 1 Oct. 2023.
https://myunt.sharepoint.com/:p:/s/CSCE5430-SEGroupProject/ERTE6JCz-lJLu7jjbexvhZoBwPAmY09jw0OQn64ksgNSFQ?e=U3AaUr

2. Overall Description

2.1 Product Perspective

For the end user, the product will be a system to book or modify appointments. Once they sign up for the web application, the system will allow users to choose the appointments based on location/distance preferences, doctor specialty preferences, healthcare providers preferences, and time and availability preferences. The healthcare providers or doctors' teams can manage and access patient data relevant to their clinic or hospital.

This product replaces traditional appointment booking methods like in-person or call-in booking, enabling more convenience. The document will later dive into system and user requirements, along with the project's system implementations, designs, and functionality. The system will be based on varied databases and software systems that help book, manage, and modify appointments. The project is designed to meet stakeholder needs, whether it is time efficiency or usability. Moreover, the application meets the essential top-level needs of patients and healthcare providers through an easy sign-up procedure, strong password security options, fast-loading pages, preference match options, etc. It allows healthcare providers to manage and access patient data.

Overall, the system structure plan is to include details of doctor availability for the patients to access and choose appointments from and where the doctor's clinic or hospital is located, including the distance from the users' entered address or location to the doctor's location. The patient will be able to select a doctor based on various criteria including specialty, distance, and doctor availability. The system will attempt to find an appointment that matches the patients' criteria or give the patient similar alternatives if there is nothing that matches the criteria entered. Doctors and Clinic staff will be able to manage these appointments and visualize patient data to confirm available appointments that the patient entered in the personal information section after signing up for the web application.

The below figure 2.1 displays the major components of the system, connections between the various subsystems, including external interfaces. The system structure runs on a Django Web framework, with a frontend of JavaScript, HTML/CSS, data container of MySQL, API services, backup/cloud services running on web application accessible through the internet. The subsystems within the frontend and backend can be defined through an authentication page that allows a sign in, which is later able to update the database. An appointment booking sub-system is a scheduler system that allows calendar format booking. Additionally, under this Django backend web application system, a doctor availability tracker is implemented through the project to provide doctor availability information to patients and staff members. Th product's other subsystem will implement sharing of data for better communication interfaces between the system. This subsystem will be examined through different API services, that can be connected to external APIs as well. The subsystem containing patient and doctor information can be stored utilizing database systems such as what this product will utilize data storage through MySQL. This can be done through various queries where MySQL aids in storage support for retrieving or storing the patient and doctor information.

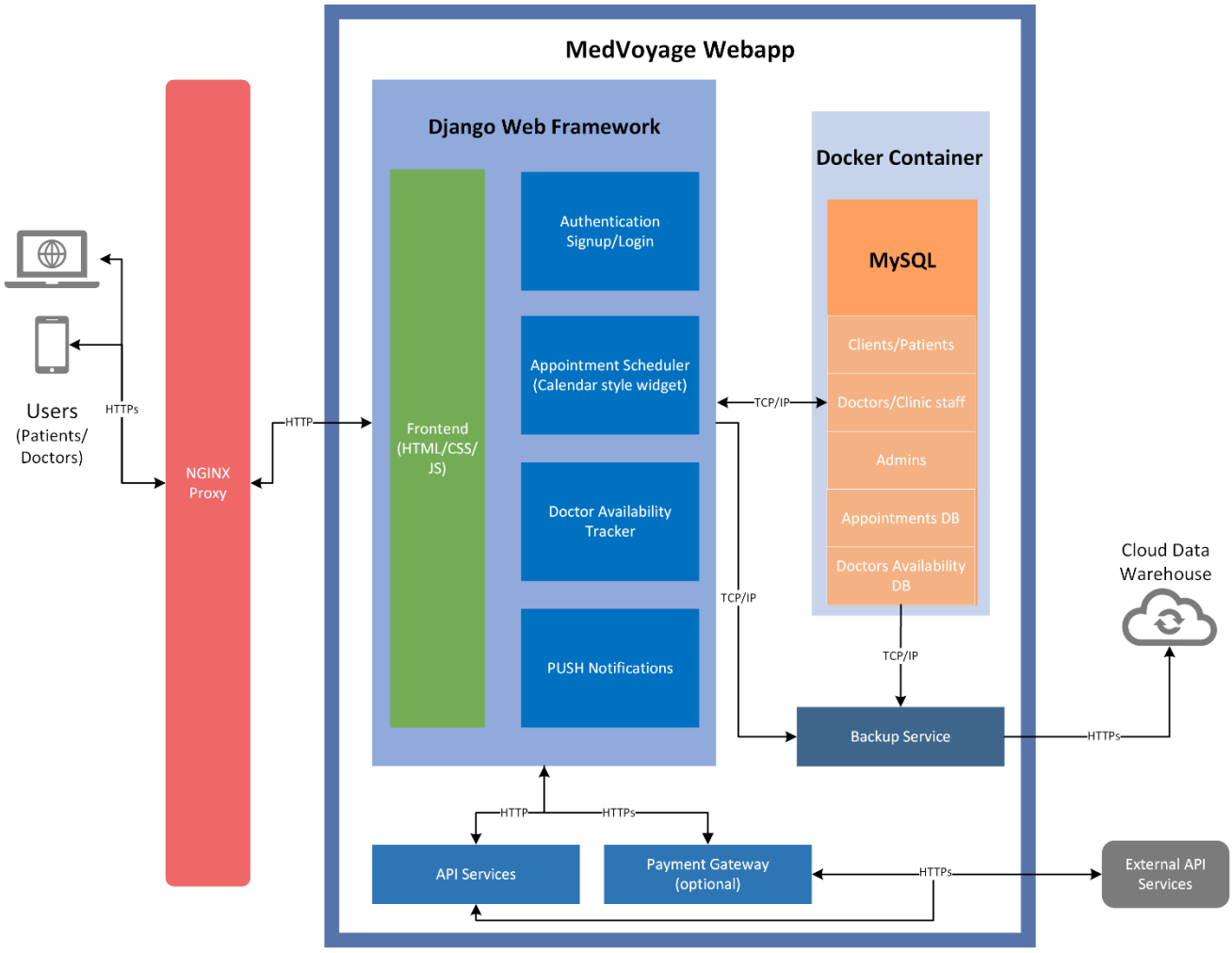

Figure 2-1: Detailed MedVoyage WebApp system diagram with dataflow

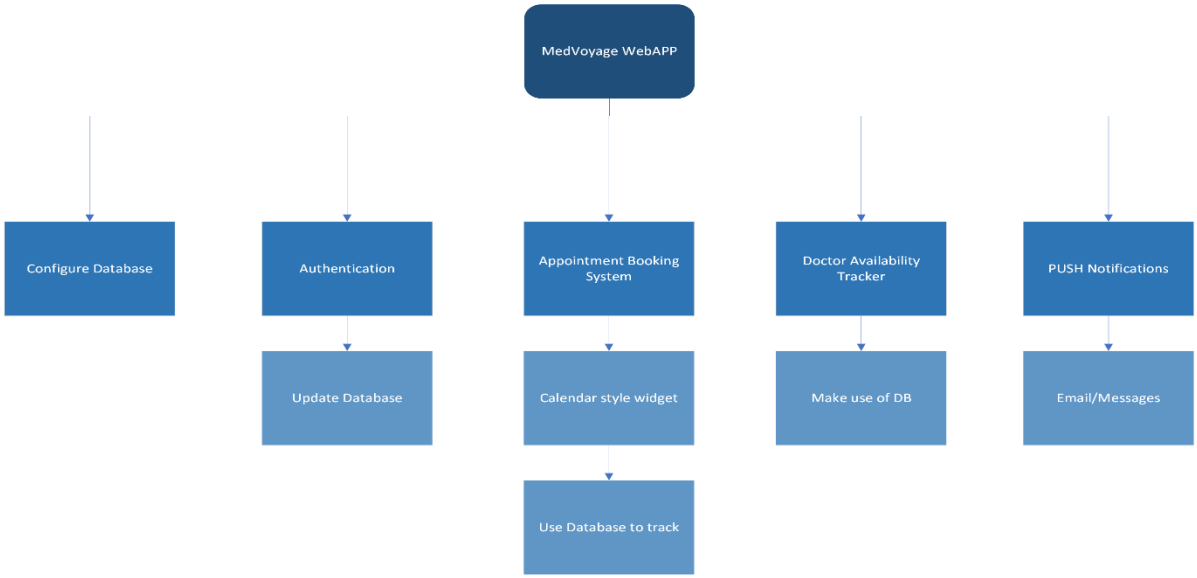

Figure 2-2: Software feature diagram with MedVoyage's core features

2.2 Product Features

The MedVoyage application will facilitate an easy, convenient, and hassle-free clinical appointment booking. When the users sign up, they will be provided with information related to the available doctors by their specialties. Once they select their preferred doctor, the application provides the doctor's calendar displaying their availability. Once a suitable day and time for their appointment is chosen, they can book it. Once the appointment is confirmed, the users will receive a notification. The users are also allowed to reschedule/cancel their appointments. Additionally, they could upload files related to their health issues so the doctor can review them before their appointment. The user's health data would be strictly confidential and not exposed to anyone, but the doctor concerned, ensuring privacy and security. The above diagram displays major features implemented by the product. The high-level features above demonstrate the main synopsis of structures the product will include to keep the system running. The features also delve into more detail as shown in figure 2.2 illustrated by the light blue squares characterizing the sub-features after major features perform adequately.

2.3 User Classes and Characteristics

The application could have the following user classes: Patient, Doctor, Clinic Staff, System Administrator, and Application Managers.

Patient users would be the primary end-users of this application. The Patients will be able to book, reschedule, or cancel an appointment for themselves along with adding or editing their personal contact information. They are assumed to have basic computer or mobile device skills. We want the system to be easy to use and accessible for them regardless of their skills, experience, or environment.

Doctors would be the second most common type of user in the system. They would use the application to manage their appointment availability. They will be able to set times that are available for appointments, view all appointments that have been booked with them, see currently available appointments slots on their own schedule, and view the contact information of patient users who have booked appointments with them. They are assumed to be proficient with computer systems and medical software.

Clinic Staff users would be the third most common type of user in the system. These users will use the system daily and have the second highest amount of privilege in the system since they would assist the patients with appointments. Clinic Staff users will be able to view, reschedule, or cancel appointments made by patient users and be able to book an appointment for a Patient user if the Patient user is scheduling a follow-up appointment at the clinic or needs assistance with appointment booking. The Clinic Staff user is assumed to have computer skills and general scheduling systems.

System Administrators would be the least common type of user in the system. The users will have the highest level of privilege in the system. These users will have access to all system features since they are special users that will only be accessing the system when necessary for system maintenance or troubleshooting and have privileges to add, update or remove patients, doctors, and clinic staff data as needed.

Application Managers are the ones who manage the technical aspects of the system and are responsible for maintaining and ensuring the proper functioning of the application. They will have full control over the entire system and will be responsible for ensuring the system's security and stability, as well as resolving any technical issues. They work along with the system administrators for technical tasks.

2.4 Operating Environment

The application would function in a web-centric environment accessible via the Internet. All the application components are developed using web-based frameworks and technologies and could be hosted on web server that supports web-based apps, using technologies like Apache, Nginx, or cloud hosting. The implementation languages would be Python and frameworks such as Django or Flask. MySQL and PostgreSQL would be the choices for managing the databases. The application would be designed to be compatible with various web browsers such as Mozilla Firefox, Google Chrome, Microsoft Edge, and Safari across different operating systems. Security measures like data encryption, access control, and regular security updates would be in place to protect sensitive patient or doctor data.

2.5 Design and Implementation Constraints

The following includes some of the constraints that may impact the design and implementation of the application:

Regulatory Compliance:
Compliance with healthcare regulations such as the Health Insurance Portability and Accountability Act (HIPAA) is paramount. We must ensure that patient data remains secure and that the web application adheres to all relevant healthcare data related privacy laws.

Security Considerations:
We figured out that strict security measures are essential. The team may be limited by the need to implement encryption, access controls, secure authentication, and regular security audits to protect patient/ doctors' data.

Database Selection:
The users/clinics/doctors may have specific preferences or mandates for the type of database management system (DBMS) that must be used. This could limit the options to select a DBMS based on project requirements.

Programming Language and Frameworks:
The team members may have language and framework preferences based on their existing technology stack or in-house expertise. These preferences could limit the choice of programming languages and frameworks for development.

Communication Protocols:
The application would use standard HTTP/HTTPS to communicate with the end-users' web browsers. The same protocols would be used in communication with the API layer while the native API would communicate to the stored data using the native DBMS protocol.

Design and User Interface Standards:
The patient/clinic/doctor may establish design conventions or user interface (UI) standards. Adherence to these standards may constrain design choices ensuring consistency with established standards.

Scalability Requirements:
Scalability requirements, such as the need to handle a specific number of users or appointment bookings, may limit the architectural choices and technologies that can be used.

2.6 User Documentation

The applications' operational instructions would be provided in the form of a GitBook manual and/or documentation set.

Technical documentation of the application may include systems' architectural design, application workflows, release notes detailing updates, improvements, bug fixes and the API's involved.

For the process of booking, visualizing, monitoring, and confirming appointments, it is critical for the end-users to easily comprehend this project's system. Consequently, there could be an on-line help tool where the users may search for items needed for clarity. Essentially, there will be a navigation page available for the users to aid in the web-application's processing.

2.7 Assumptions and Dependencies

- The user should know the basics of using a computer or mobile device
- The user should be able to navigate to the website via the web browser
- The user should have access to the internet
- The project assumes that it will comply with healthcare regulations like HIPAA (Health Insurance Portability and Accountability Act) or GDPR (General Data Protection Regulation), and any changes in regulatory requirements can necessitate adjustments to the project.
- If users, including healthcare providers and patients, will readily adopt and use the application as intended. Additionally, user behavior and adoption can vary and may require adjustments to user interfaces and training materials.
- If the required patient data, including medical records and insurance information, will be available in a specific format or accessible through APIs. Changes in data sources or formats can impact data integration.
- Dependency on hosting and cloud providers (e.g., AWS, Azure, GCP). Changes in provider policies, pricing, or services can impact deployment and scalability.
- Dependency on third-party compliance audits or certifications (e.g., HIPAA compliance audits). Failures in audits can lead to non-compliance and necessitate changes.
- Dependency on web browsers and their compatibility with the application's frontend. Updates or changes in browsers may require adjustments in the frontend code
- Dependency on external services for features like email delivery or SMS notifications. Changes in service providers or APIs may affect functionality.
- Dependencies on specific database management systems (e.g., MySQL, PostgreSQL) and their versions are essential, and upgrades or migrations may be required if these dependencies change.
- Dependency on legal agreements or licenses for software components, including open-source libraries. Changes in licensing terms can impact the project's legal compliance.
- Dependency on security services, such as firewalls and intrusion detection systems. Changes in security needs or technologies may require adjustments.

3. System Features

3.1 MedVoyage Basic Functionality

3.1.1 Front Landing page – MedVoyage online booking system will be introduced.

3.1.2 About us page – Here details about the goal of the MedVoyage application will be explained and the general information about the clinic and creators will be shared.

3.1.3 Contact us page – Details regarding the contact information and email addresses will be shared to contact the team.

3.1.4 Navigation bar – a proper navigation bar will be added.

3.1.5 Mobile user-friendly design – mobile-friendly design will be implemented.
3.1.6 Custom Error pages (for HTTP 400, 300, 500 status codes) – different error pages will be worked upon to be displayed during error scenarios.
3.1.7 Configure the Database – MySQL database configuration will be performed.
3.1.8 Unified signup form – a unified user registration form with fields for login-type, name, email, password, about them, and contact information will be implemented.
3.1.9 Unified Login form – a login form with login-type, email and password will be implemented for patients and doctors to login.
3.1.10 Handle different authorizations for patients and doctors – proper different authorization DBs will be implemented in the system.
3.1.11 Authorization DB integration – proper integration of login and signup entries in MySQL databases.
3.1.12 Client profile update form – a form page to allow clients/patients to update their profiles.
3.1.13 Doctor profile update form – a form page to allow doctors to update their profiles.
3.1.14 Forgot password form – form to help users (both doctors & patients) to retrieve their passwords that have been forgotten through email.
3.1.15 Logout page - A logout button will be created where patients and doctors can press to logout properly.
3.1.16 Password change form – a provision to update password for users in their profile.

3.2 MedVoyage Doctor appointment booking system

3.2.1 Doctor Availability tracker – Proper form for doctors to add their availability details.
3.2.2 Calendar style widget-An interactive calendar widget for the patients to choose appointment dates and times.
3.2.3 Appointment Booking System-A format booking system with available time slots and doctor’s information.
3.2.4 Client upcoming appointments list -A list to store the upcoming patient's appointment date and time.
3.2.5 Client previous appointments list-A list to store the previous appointment history of the patients.
3.2.6 Doctor day-to-day appointments list-A list to store the day-to-day appointments for each doctor.
3.2.7 Enabling notification capabilities-An in-app notification system for the patients to set preferred notification method for reminders. method.
3.2.8 Appointment deletion capabilities-A provision for patient/doctor where under extreme circumstances they will have an option to delete their appointments.
3.2.9 Appointment Doctor confirmation capability-A notification sent to the doctor to confirm the appointment after patient books one.
3.2.10 Appointment Doctor rejection capability-Once the patient has booked an appointment, the doctor will have special permissions to reject the appointment based on some special circumstances, like health issues or weather-related issues.
3.2.11 Basic search to get relevant doctors by name-a search interface system with dropdown menus, text inputs, and checkboxes to search doctor’s name.
3.2.12 Enhancing privacy and data security-a Two-Factor Authentication (2FA) for all users for privacy and data security.

3.2.13 Clear dashboard for doctors-A dashboard will be created where doctors will be able to update their information and available timings.

3.3 Admin system and extra capabilities

3.3.1 Create admin authentication system-An authentication system for admin to handle special requests by the patients and doctors, to perform steps to maintain data privacy.
3.3.2 Admin dashboard system-The admin will have access to a customized dashboard to perform the above-mentioned tasks.
3.3.3 Appointment notification remainders-The notification reminders will be sent by email.
3.3.4 Enhancing search system to add filters-Special filters like region, gender, age, doctors' specialization will be added for better performance.
3.3.5 Updatable emergency contact information for patients-The patients' contact information will be updated where they will be able to add close family members whom they want to contact in case of emergencies.
3.3.6 Notification preferences capabilities-A special feature will be added where the patients, doctors and admin could set different preferences for notifications like emails, WhatsApp, telegram etc.
3.3.7 Adding privacy policy and Terms of Service-As HIPAA rules must be strictly followed, some terms of service and privacy policy will be updated.
3.3.8 Updating sign-up pages to have Terms of Service and privacy policies-To create terms of services and policies and integrate in the sign-up pages.
3.3.9 Data backup and recovery system-A system to establish a secure connection to the database and perform necessary data validation, data backup will be generated.
3.3.10 Patients' medical history upload-A provision will be made where patients can upload the pdf of their medical records.
3.3.11 Doctors' electronic prescription capability-A provision will be made where doctors could upload the pdf of their prescription to the patient

3.4 Extra features

3.4.1 Symptom checker:

Many users may not be aware of the health issues that they are suffering from or which specialist they should consider. To assist such patients, we can create a dashboard where users input their symptoms. After submitting the details, they will receive information about their condition, recommended specialists to consult, and a list of doctors specializing in that field. This can be implemented by:

- designing a ML model
- providing a dropdown menu where users can select their symptoms and once, they click on the submit button, they will receive information about their condition, the recommended specialists, and a list of available doctors.

3.4.2 Health Challenges and Rewards:

The team will create challenges and on completion, the user earns rewards, and these rewards can be used to get discounts on appointments. One of the challenges can be steps count challenge where we tell users

that if they reach some number of steps, they get some reward points, as we are developing a web application, it is difficult for us to track the step count, but we can make use of third-party services by:

- Choosing a third-party fitness application.
- Creating a developer account.
- Obtaining API credentials
- And then registering our web application

Now, we will implement authentication mechanisms that allow users to link their fitness accounts to our web application.

3.4.3 HealthCare Subscription Plan:

If we have patients who come for monthly checkups, we will offer them yearly plans which will cost them less when compared to paying appointment fees.

3.4.4 Health Severity Tracker:

In the user dashboard, we can add a tracker feature where users can monitor all their health records for their own purposes or to update the doctors. For suppose:

- If a patient is diagnosed with diabetes and starts medication, they are willing to keep track of reports every time they get the test done. Then, he can use this tracker to update.
- If it is a migraine patient and they are on a 6-month course, they have kept track of migraine attacks so that when they meet the doctor next time, they can tell if their frequency is increased or decreased.

4. External Interface Requirements

4.1 User Interfaces

MedVoyage's UI is a web-based front-end that users including patients and healthcare providers, access through web browsers on various devices, including desktop computers, laptops, tablets, and smartphones. This project will be accessible and compatible with all web browsers such as Google Chrome or Internet Explorer. These will function under GUI standards so that each image and page in the web application is smooth and uniform, along with proper icons and buttons that are helpful to less experienced users.
MedVoyage's UI design theme and layout and color choices is yet to be determined, but standard web conventions for ease of navigation and clarity will be followed including simple login menus, links, registration page, and clear end-use controls such as form input fields and buttons with clear actions. The notifications will be provided using emails and WhatsApp.

4.2 Hardware Interfaces

The project will be a web-based application, so the computer hardware will need to be connected to the internet. Support devices that will assist in the hardware interface for the system include a modem, an Internet Service Provider, a router, ethernet cables, etc. It is also essential for the given web browser to support JavaScript.

4.3 Software Interfaces

MedVoyage application will utilize a Django web framework to streamline development and manage backend operations. The html and CSS scripts will be used to create the webpages for the front end. MySQL is the database that will be used for storing the user profiles the doctors and patients and clinics and the appointment details and other related data. MedVoyage application will run on virtual server on google cloud. MedVoyage web application will be accessed by users using various client-side operating systems like (windows or mac) through web browsers. Data sharing will occur through structured formats such as XML via API endpoints. The http and https protocols will be used, and engine x and self-authentication will be used for securing data. If the MedVoyage application handles payments, integration with a specific payment gateway (e.g., swift) will be necessary for processing financial transactions securely.

4.3 Communications Interfaces

The application would use standard HTTP/HTTPS to communicate with the end-users' web browsers. The same protocols would be used in communication with the API layer while the native API would communicate to the stored data using the native DBMS protocol. Emails will be formatted in HTML and plain text, containing essential appointment details and action links. Emails will not contain sensitive patient information. Secure email transmission will be ensured through SSL/TLS encryption. HTTPS will be enforced to encrypt data in transit, ensuring secure communication between browsers and the server. User interfaces will be rendered in HTML, CSS, and JavaScript. The system will send email notifications for appointment confirmations, reminders, and updates. Users will fill in electronic forms for registration, appointment booking, and updating medical history. Form data will be transmitted over HTTPS, ensuring encryption during submission. Data exchange will occur in structured formats XML via API endpoints. The system will employ real-time data synchronization to update user dashboards and appointment statuses instantly.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

1. User Interface Responsiveness:
    a. Performance Requirement: When a user clicks a button or tab that opens in a new window, the page should load within 2 seconds.
    b. Rationale: Modern users have limited patience for slow loading pages. If a page takes too long to load, users may become frustrated, lose interest, or even switch to a competitor's application. The 2 second threshold is chosen to ensure a responsive and engaging user experience, which is essential for user retention and maintaining a competitive edge in the market.

2. Database Query Optimization:
    a. Performance Requirement: When querying the database and loading a large amount of data, implement a chunked loading system or display a loading symbol within 1 second all while maintaining the rest of the page is still responsive while the loading is happening.
    b. Rationale: Loading large datasets can take a fair amount of time. By loading data in chunks or providing a loading symbol within 1 second and keeping the rest of the page responsive, we ensure that the user is aware of what is happening, which improves the UX and prevents them from perceiving the application as unresponsive. This will address two things:
        i. User Feedback

1. The loading symbol and or the chunked loading will provide almost immediate feedback to the user, indicating that the application has received and is processing their request. This will prevent users from thinking that the application is unresponsive or frozen.

ii. Performance Optimization

1. Chunked loading will ensure that the user gets to start interacting with the loaded data, while the rest is loading in the background, enhancing our users experience.

3. Authentication Speed:
   a. Performance Requirement: When a user attempts to login, the authentication process should be complete within 3 seconds.
   b. Rationale: Quick authentication is necessary to prevent users from experiencing unnecessary delays during the login process. Users will be more likely to continue using the application if they can login quickly. Longer authentication times can and will lead to user frustration and wasted time, discouraging users from using the application.

5.2 Safety Requirements

1. Automatic Session Timeouts:
   a. Safety Requirement: Implement automatic session timeouts for the users after a certain time of inactivity.
   b. Rationale: Automatic session timeouts prevent unauthorized access and data exposure if the user forgets to logout.
2. User Education and Confirmation:
   a. Safety Requirement: Implement features such as asking users to confirm their actions during critical operations like booking or cancelling an appointment.
   b. Rationale: This is to safeguard against unintended user actions to prevent erroneous bookings/cancellations.

3. Testing

   a. Performance Testing: The project shall perform load testing, stress testing, and scalability testing to ensure the system's performance meets non-functional requirements like response time, throughput, and resource utilization.
   b. Usability Testing: The project shall evaluate the user interface and user experience to ensure it meets non-functional requirements related to usability, accessibility, and user satisfaction.
   c. Security Testing: The project shall ensure that all the meets non-functional requirements related to data protection, privacy, and security by identifying and mitigating vulnerabilities.
   d. Reliability and Availability Testing: We shall validate the system's ability to meet non-functional requirements related to uptime, fault tolerance, and availability.
   e. Compatibility Testing: The project shall confirm that the system functions correctly across different browsers, devices, and platforms, aligning with non-functional requirements for compatibility.
   f. Compliance Testing: The project shall ensure that the system adheres to relevant regulations and standards, such as HIPAA or GDPR, addressing non-functional requirements for compliance.
   g. Performance Monitoring: The project shall do ongoing monitoring of the system's performance aligns with non-functional requirements to maintain performance levels over time.

5.3 Security Requirements

1. Two-Factor Authentication:
    a. Security Requirement: To implement Two-Factor Authentication (2FA) for all users
    b. Rationale: This ensures an additional layer of security against unauthorized access making it difficult for attackers to gain access.

2. Data Encryption:
    a. Security Requirement: Encryption of sensitive data during storage and transmission
    b. Rationale: This ensures that the confidentiality and integrity of the user and clinical data is maintained, reducing the risk of data breach and unauthorized access.

3. Compliance with HIPAA Security and GDPR Data Protection Rules:
    a. Security Requirement: To implement security controls for protecting users' health information and implement data protection measures like data encryption
    b. Rationale: This is to ensure the secure handling, transmission and storage of healthcare related information and robust protection of personal data for users, mitigating the risks associated with data breaches, unauthorized access, and non-compliance penalties.

4. Password Security
    a. Password Length: Passwords should be at least 8 characters long. The product shall implement a validation rule that checks the length of passwords entered by users. If a password is less than 8 characters, display an error message and require users to choose a longer password.
    b. Special Character: Passwords should include at least one special character (e.g., !, @, #, $, %, etc.). We shall use regular expressions or character validation to check if the password contains at least one special character. If not, we shall display an error message and prompt the patient/doctor/clinic staff to include a special character in their password.
    c. Password Complexity: We shall encourage prospective patients/doctors/clinic staff to create complex passwords by providing guidelines and suggestions. The product shall offer password strength indicators or tooltips that guide users on how to create a strong password. For instance, the product shall provide real-time feedback on password strength, such as "Weak," "Medium," or "Strong," based on complexity. While the patient/doctor/clinic staff is entering their password, the system evaluates it and displays "Weak" for "12345" but "Strong" for "P@ssw0rd."
    d. Password Hashing: We shall store passwords securely by hashing them using a strong hashing algorithm (e.g., bcrypt). When a patient/doctor/clinic staff sets or changes their password, hash it before storing it in the database. When verifying login credentials, hash the entered password and compare it to the stored hash for authentication. If a patient/doctor sets their password as "MySecurePassword," the system stores it as a secure hash like "$2y$12$C98TxL....." in the database.
    e. Password Reset Policy: We shall Implement a secure password reset policy, requiring patient/doctor/clinical staff to verify their identity before resetting passwords. When patient/doctor/clinical staff request a password reset, send a verification link to their registered email address. Only allow password resets after successful email verification or security questions authentication.

By implementing these password security measures, the product shall enhance the overall security of accounts within the online clinic booking web application. This helps in not only protecting the

patient/doctor information, but also contributes to compliance with data protection regulations such as HIPAA and ensures that user accounts remain secure even in the event of a security breach.

5.4 Software Quality Attributes

1.Correctness	: All the implemented features must be verified through automated and manual testing.
2.Interoperability	: Usage of RESTful APIs for easy integration with other systems.
3.Portability	: The application should be able to work on Android/iOS/ modern web browsers.
4.Testability	: Have broad unit tests such that at least 80% of the code is covered.
5.Usability	: By prioritizing ease of use, we aim to deliver a smooth and efficient user experience that satisfies immediate needs and encourage long-term user engagement.
6.Maintainability	: Maintaining detailed documentation for all the features and architecture of the application.

6. Other Requirements

6.1 Implementation Phases of MedVoyage

The above product will be implemented in three development phases as shown below:

Phase 1:

Requirements: MedVoyage Basic Functionality

Details: Front Landing page, about us page, Contact us page, Navigation bar, Mobile user-friendly design, Custom Error pages (for HTTP 400, 300, 500 status codes), Configure the Database, Unified signup form, Unified Login form, Handle different authorizations for patients and doctors, Authorization DB integration, Client profile update form, Doctor profile update form, Forgot password form, Logout page, Password change form.

Due Date: 23rd October 2023.

Phase 2:

Requirements: MedVoyage Doctor appointment booking system

Details: Doctor Availability tracker, Calendar style widget, Appointment Booking System, Client upcoming appointments list, Client previous appointments list, Doctor Day-to-day appointments list, Enabling notification capabilities, Appointment reminders, Appointment deletion capabilities, Appointment Doctor confirmation capability, Appointment Doctor rejection capability, Basic search to get relevant doctors by name, Enhancing privacy and data security, Clear dashboard for doctors.

Due Date: 13th November 2023.

Phase 3:

Requirements: Admin system and extra capabilities

Details: Create admin authentication system, Admin dashboard system, Appointment notification remainders, enhancing search system to add filters, Updatable emergency contact information for patients, Notification preferences capabilities, Adding privacy policy and Terms of Service, Data backup and recovery system, Patients' medical history upload, Doctors' electronic prescription capability, Updating sign-up pages, Data backup and recovery system.

Due Date: 4th December 2023.

6.2 Member Contributions table

Name	Contribution
Yasmeen Haleem	Added details for functional requirements like Appointment Management, Appointment Tracker, Database and Backend in section 3 and in non-functional requirements added password security and testing and added assumptions and user interfaces and hardware interfaces.
Bhavani Rachakatla	Section 2 Product Perspective, Product Features, User Classes and Characteristics, Operating Environment, Design and Implementation Constraints, User Documentation Section 5 Safety Requirements, Security Requirements and Software Quality Attributes under the Non-Functional Requirements Section 6 – Details, Requirements Updated the GANTT Chart, Created and updated note-deliverable2.txt file in GitHub
Vidhi Bhatt	Inserted all of Section 1 (purpose, document conventions, intended audience and reading suggestions, project scope, references) Section 2.1 (product perspective not including future diagram), hardware & user interface
Pravallika Bollavaram	Added extra features under the Functional requirements
Manushree Buyya	Added few features in the Section 3
Emmie Abels	Added Section 2.3, made edits to Section 2.1 for clarity, and fixed formatting throughout the document
Demir Altay	Added Section 5.1, helped writing system administrator portion of section 2.3
Kalyan Cheerla	Worked on design and system diagrams & outlined few parts of different sections.

Appendix A: Glossary

Term	**Definition/Acronym**
Web-app	Website application
User	Patients, Healthcare Professionals, or their teams
API	Application Programming Interface
SQL	Structured Query Language
GUI	Graphical User Interface
HTTP	Hyper Text Transfer Protocol
HTTPs	Secure Hyper Text Transfer Protocol
TLS	Transport Layer Security
Container	A software container is a package of software's that help isolating and contains all the elements to run a service/application.
Docker	Docker is an open-source container platform that packages applications in containers.
PCI DSS	Payment Card Industry Data Security Standards
T&C's	Terms and Conditions
GCP	Google Cloud Platform
GDPR	General Data Protection Regulation
HIPAA	Health Insurance Portability and Accountability Act
DBMS	Database Management Systems

Appendix B: Analysis Diagrams & Models

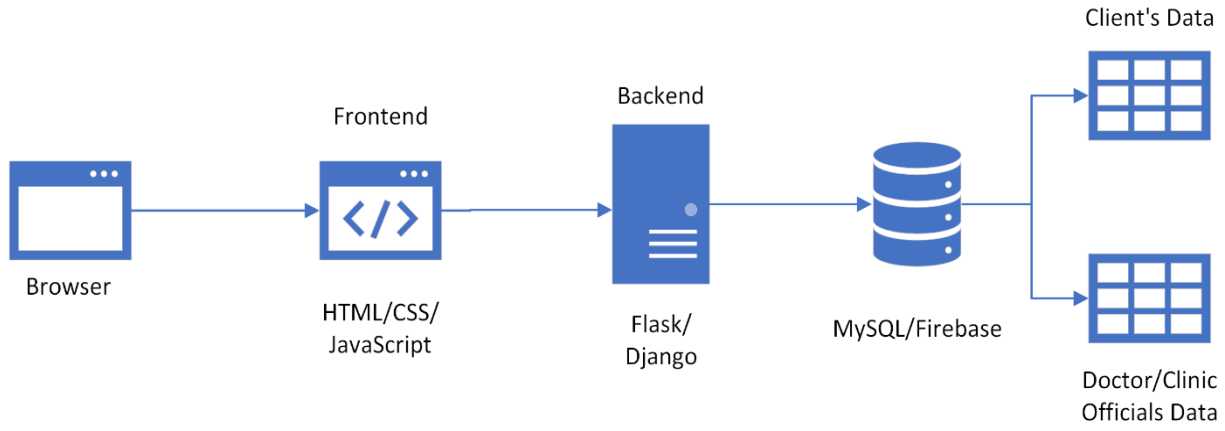

Figure 6-1: Details about front-end, back-end, database

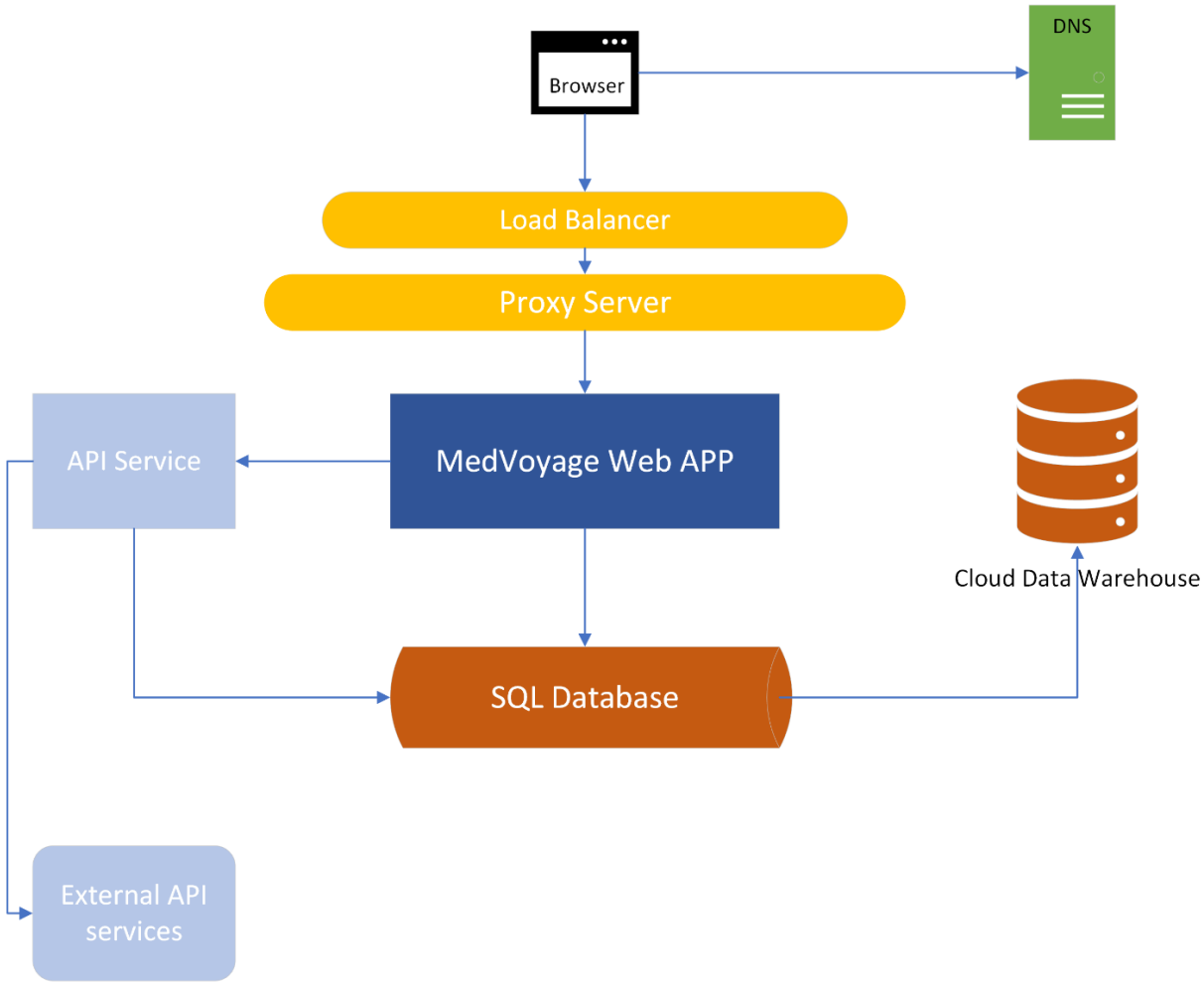

Figure 6-2: High-level System design diagram

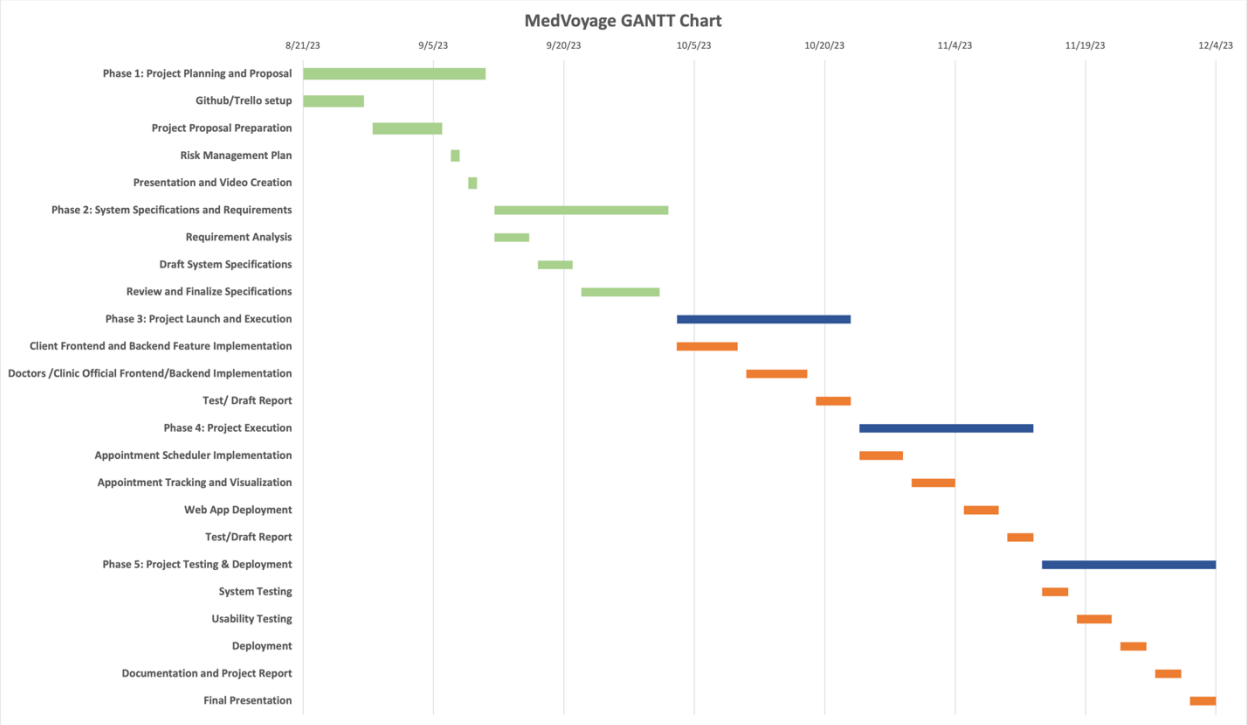

Figure 6-3: MedVoyage Gantt Chart for Deliverable 2

Appendix C: Issues List (TBD in the future)

Issue	**Details**