**Software Requirments**

**Specification**

**for**

**Hello Doctor**

Version 1.0 approved

Prepared by Usama, Kashif, Fabiha

Organization FAST NUCES

06/05/2023

**1. Introduction**

**1.1 Purpose**

The purpose of this doctor-patient application is to provide a convenient and user-friendly platform for patients to schedule appointments with their doctors using their mobile devices. With the Laravel backend, the application is built with a robust and secure architecture to ensure the protection of patient information. By streamlining the appointment scheduling process, patients can easily manage their healthcare needs and receive timely care from their trusted providers. Ultimately, this application aims to improve access to healthcare services and enhance the patient experience by offering a convenient, reliable, and efficient way to book appointments.

**1.2 Document Conventions**

1. File Naming Convention: All files should be named in a consistent and descriptive manner, using camelCase. For example, "mainScreen.dart" or "authService.dart".
2. Code Formatting Convention: Code should be formatted using the Flutter code formatting standards, which can be enforced through the use of tools like Dartfmt. This includes proper indentation, spacing, and commenting.
3. Function and Variable Naming Convention: Function and variable names should be descriptive and meaningful, using camelCase. For example, "userEmail" or "getDoctorAppointments".
4. Database Naming Convention: Database tables, columns, and relationships should be named in a consistent and logical manner, using snake\_case. For example, "appointments" or "doctor\_specialties".
5. SQL Query Formatting Convention: SQL queries should be formatted using a consistent and readable style, with proper indentation, spacing, and commenting. This can help make queries more readable and maintainable over time.
6. Commenting Convention: Code should be well-documented using comments, which provide context and explain the purpose of functions, variables, and other elements. Comments should be written in a clear and concise manner, using proper grammar and punctuation.

**1.3 Intended Audience and Reading Suggestions**

**Intended Audience:**

The doctor-patient application you have created using Laravel is aimed at providing a convenient and user-friendly platform for patients to schedule appointments with their doctors using their mobile devices. As such, the intended audience for this application would be:

1. Patients seeking an easier and more efficient way to book appointments with their healthcare providers.
2. Doctors and healthcare providers are looking to streamline their appointment scheduling process and enhance the patient experience.
3. Healthcare administrators seeking to improve patient access to care and optimize healthcare delivery.

**Reading Suggestions:**

To fully understand and utilize the doctor-patient application you have created, users should have a basic understanding of:

1. How to use a mobile device and navigate mobile applications.
2. Basic healthcare terminology and procedures.
3. The importance of regular healthcare check-ups and appointments.
4. The need to protect their personal information and adhere to data privacy policies.

**1.4 Product Scope**

The scope of the Flutter project's software requirements specification (SRS) is to create a mobile application that enables patients to schedule appointments with their healthcare providers in a convenient and user-friendly manner. The application will be designed to run on both Android and iOS mobile devices and will allow patients to search for and select healthcare providers, view their availability, and book appointments at their preferred times.

Key features of the application will include:

1. User Authentication: Patients will be required to create an account and log in to access the application's features. Authentication will be managed securely, with password hashing and secure storage of user information.
2. Appointment Scheduling: Patients will be able to schedule appointments with their chosen healthcare provider at a time that is convenient for them. They will be able to view provider availability in real-time and select an appointment slot that suits their schedule.
3. Appointment Reminders: Patients will receive appointment reminders via push notifications or email, ensuring that they do not forget their scheduled appointments.
4. Review and Rating: Patients will be able to leave reviews and ratings for their healthcare providers, helping to improve the quality of care and service provided.
5. Secure Data Storage: The application will store patient and provider information securely, adhering to data privacy and security standards.

**1.5 References**

1. Google. (2021). "Flutter for Mobile App Development." Google Developers website.
2. Laravel. (2021). "Laravel Documentation." Laravel website.
3. MySQL. (2021). "MySQL Documentation." MySQL website.
4. Flutter. (2021). "Package Management with Pub." Flutter website.
5. PHP. (2021). "PHP: Hypertext Preprocessor." PHP website.

**2. Overall Description**

**2.1 Product Perspective**

The product perspective for the doctor patient application involves creating a user-friendly mobile platform for scheduling appointments with healthcare providers. It will run on both Android and iOS devices and integrate with external healthcare information systems and payment gateways. Key stakeholders include healthcare providers, patients, and app developers. Defining the hardware and software components required for the application, as well as the external interfaces, is also important. By defining the product perspective, the software requirements specification can guide the development process and ensure the final product meets the needs of its intended users.

**2.2 Product Functions**

The doctor patient application has several key functions including:

1. Adding and deleting data to manage appointments and keep track of medical history
2. Adding favorite doctors for easy access to trusted healthcare providers
3. Scheduling appointments based on availability of healthcare providers and patient preferences
4. Updating user profiles to ensure accuracy of medical records and personal information
5. Login and logout functionality for securing user data
6. Registering new users to expand the user base and make the application accessible to a wider audience.

**2.3 User Classes and Characteristics**

In the doctor patient application, there are two primary user classes: patients and doctors. Patients are individuals who use the application to schedule appointments with doctors, manage their medical history and personal information, and communicate with doctors. Doctors are professionals who use the application to manage their appointments, communicate with patients, and access medical records. The characteristics of patients may include limited technical skills, a need for intuitive and user-friendly interfaces, and a desire for secure and private communication with doctors. Doctors may have higher technical skills and require additional features for managing appointments and medical records. Understanding the characteristics of these user classes is essential for designing and implementing an application that meets their specific needs and preferences.

**2.4 Operating Environment**

The operating environment for the doctor patient application is primarily mobile devices such as smartphones and tablets running on Android and iOS operating systems. The application will require internet connectivity to access external healthcare information systems and payment gateways. It may also require integration with third-party software or hardware components, such as electronic health record systems and telehealth platforms. The application will need to be developed using software frameworks and tools such as Flutter, Laravel, and SQL databases to ensure compatibility and performance across different mobile devices and platforms. Additionally, the application will need to adhere to security and privacy regulations, such as HIPAA compliance, to protect sensitive patient information. Understanding the operating environment is critical for ensuring that the application can function seamlessly and securely on different devices and platforms.

**2.5 Design and Implementation Constraints**

In the context of the doctor patient application developed using Flutter, there are several design and implementation constraints that need to be addressed. One of the primary constraints is the need to ensure compliance with HIPAA regulations to protect patient data. The application must also be designed to operate seamlessly on a wide range of mobile devices running on both Android and iOS operating systems, which requires careful consideration of hardware limitations and screen sizes. The application will also need to interface with third-party software, such as electronic health record systems and payment gateways, which may pose compatibility issues that need to be resolved. Finally, security considerations, such as data encryption and user authentication, will need to be incorporated into the application design to ensure that sensitive patient information is protected. Addressing these design and implementation constraints is critical for developing a high-quality, reliable, and secure application that meets the needs of its users.

**2.6 User Documentation**

User documentation is an essential component of the doctor patient application, as it provides users with instructions and guidance on how to use the application effectively. The documentation will include user manuals, FAQs, and tutorials, which will be easily accessible through the application's interface. The documentation will be written in clear and concise language, making it easy for users to understand and follow. It will also be regularly updated to reflect any changes or updates to the application's features and functionalities. The user documentation will be designed to provide an optimal user experience, which will help users make the most of the application's features and functions.

**2.7 Assumptions and Dependencies**

The doctor patient application is designed with the assumption that users have basic knowledge of using mobile applications. It is dependent on stable internet connectivity and access to patient data and appointment scheduling databases to function effectively. These assumptions and dependencies need to be considered while using the application to ensure optimal performance and user experience.

**3. External Interface Requirements**

**3.1 User Interfaces**

The user interface for the doctor patient application will be designed to provide an intuitive and seamless experience for users. The interface will be simple and easy to navigate, with logical and consistent screen layouts. The application will feature standard buttons and functions such as login, logout, and appointment scheduling, which will appear on every screen. The user interface will also incorporate standard error message display standards to provide users with clear feedback in case of any issues or errors. Additionally, the application will adhere to GUI standards and product family style guides to maintain a consistent look and feel across all screens. The design details of the user interface will be documented in a separate user interface specification, which will outline the software components for which a user interface is needed.

**3.2 Hardware Interfaces**

The doctor patient application will require minimal hardware interfaces, as it will be primarily used on mobile devices. The application will be compatible with Android and iOS platforms and will require devices with the following minimum specifications:

1. Android: Android 5.0 or later, with a minimum of 2GB of RAM
2. iOS: iOS 11 or later, with a minimum of 2GB of RAM

The application will also require stable internet connectivity for accessing patient data and appointment scheduling databases.

**3.3 Software Interface**

The doctor patient application will require several software interfaces to function effectively. The application will interface with the Laravel framework on the server-side, which will provide the application with data storage, retrieval, and processing capabilities. The application will also interface with MySQL databases to store and retrieve patient data and appointment schedules. On the client-side, the application will interface with the mobile device's operating system to access device functionalities such as push notifications and location services. The application will also interface with third-party libraries and APIs such as Firebase for push notifications and Google Maps for location services. The application will be developed using the Flutter SDK, which will provide a robust software interface for developing cross-platform mobile applications.

**3.4 Communications Interfaces**

The doctor patient application will rely on several communication interfaces to function effectively. The primary communication interface will be the internet, which will be used to transfer data between the client-side application and the server-side Laravel framework. The application will use HTTPS protocol for secure data transmission. The application will also use push notifications to communicate with users about upcoming appointments, and this will be facilitated using the Firebase Cloud Messaging service. Additionally, the application will interface with third-party APIs such as Google Maps to provide location-based services.

**4. System Features**

The doctor patient application will provide various features to enhance the user experience. These include user authentication, appointment scheduling, doctor search, favorite doctor list, patient profile management, appointment reminders, appointment cancellation, and rating and feedback. These features are designed to provide a seamless and user-friendly experience to both doctors and patients using the application.

**4.1 System Feature 1 - Appointment Scheduling**

4.1.1 Description and Priority:

The Appointment Scheduling feature allows users to schedule appointments with their preferred doctors. It is of high priority, as it is a core functionality of the application.

4.1.2 Stimulus/Response Sequences:

The user selects the preferred doctor and available time slot, the system confirms the appointment and sends a notification to the doctor and patient.

4.1.3 Functional Requirements:

REQ-1: The system must display a list of available doctors.

REQ-2: The system must allow the user to select a preferred doctor and available time slot.

REQ-3: The system must confirm the appointment and send a notification to the doctor and patient.

REQ-4: The system must prevent double booking of appointments for the same time slot.

**4.2 System Feature 2 - User Profile Management**

4.2.1 Description and Priority:

The feature allows users to create, edit, and manage their profiles, including their personal and contact information. This feature has a high priority as it is essential for providing personalized healthcare services to users and maintaining accurate records.

4.2.2 Stimulus/Response Sequences:

The user navigates to the profile management screen, where they can view and edit their profile information. The system validates the inputs and saves the changes. In case of any errors, appropriate error messages are displayed.

4.2.3 Functional Requirements:

The system should allow users to create a new profile or edit an existing profile.

The profile should include fields for personal and contact information such as name, age, address, phone number, and email.

The system should validate the inputs and display error messages for any invalid or incomplete information.

The system should allow users to upload their profile picture.

The system should provide the ability to delete a profile.

The system should maintain the confidentiality and privacy of user information by implementing appropriate security measures.

**5. Other Nonfunctional Requirements**

**5.1 Performance Requirements**

The doctor patient application is expected to have a high level of performance and should be able to handle a large number of concurrent users without any significant delay in response time. The application should have an average response time of less than 3 seconds for all major user actions. The system should also be able to handle a large amount of data without affecting performance. The performance requirements will be measured using load testing tools, and any issues identified will be addressed before release.

**5.2 Safety Requirements**

Safety requirements specify how the system must behave to avoid accidents or hazardous situations. In the case of a doctor-patient application, safety requirements may include ensuring the confidentiality of patient information, preventing access to the system by unauthorized individuals, and ensuring that the application does not provide inaccurate medical information or advice that could harm patients. It may also include guidelines for ensuring that the application complies with any relevant regulations or standards related to data privacy and security in the healthcare industry.

**5.3 Security Requirements**

1. The application uses encryption to protect sensitive user data, such as login credentials and medical records.
2. The application implements secure communication protocols to prevent interception or tampering of data in transit.
3. The application protects against common web application security threats, such as cross-site scripting (XSS) and SQL injection attacks.
4. The application adhere to industry-standard security best practices and undergo regular security testing and auditing to ensure ongoing protection against emerging threats

**5.4 Software Quality Attributes**

Software quality attributes are a critical aspect of any software project. For my Flutter project, the following attributes are essential:

1. Usability: The application must be easy to use, intuitive, and user-friendly.
2. Reliability: The application must be reliable and stable, with minimal downtime and errors.
3. Maintainability: The application must be easy to maintain and upgrade, with clear documentation and coding standards.
4. Portability: The application must be portable across different devices and platforms.
5. Scalability: The application must be scalable to accommodate a growing number of users and data.

**5.5 Business Rules**

1. The application complies with all relevant privacy and data protection laws andregulations.
2. User authentication is required for all access to the application, with strong password policies and multi-factor authentication options available.
3. Access to patient data is restricted to authorized healthcare providers and the patients themselves.
4. All data transmitted between the application and the server must be encrypted using a secure protocol.

**6. Other Requirements**

1. Appointment: A scheduled meeting between a patient and a doctor at a specific time and date.
2. Doctor: A medical professional who provides healthcare services to patients.
3. Login: The process of accessing the application by providing valid credentials.
4. Logout: The process of exiting the application and ending the user session.
5. Patient: An individual who receives healthcare services from a doctor or medical institution.
6. Profile: The personal information and settings of a user in the application.
7. Registration: The process of creating a new user account in the application.
8. User: An individual who interacts with the application, including both patients and doctors.

**Appendix B: Analysis Models**

**Appendix A: Glossary**

This appendix contains the analysis models used in the development of the software product. These models include use case diagrams, activity diagrams, sequence diagrams, and class diagrams. These models provide a visual representation of the system requirements and the interactions between the different components of the system. They are useful for understanding the system behavior and for communicating the system design to stakeholders.

Use case diagrams are used to describe the different ways that users interact with the system. They show the different use cases and the actors that are involved in each use case. Activity diagrams are used to show the workflow of a particular use case. They show the different activities that are performed and the order in which they are performed.

Sequence diagrams are used to show the interactions between different components of the system. They show the messages that are sent between the different components and the order in which they are sent. Class diagrams are used to show the different classes that are used in the system and the relationships between them. They show the attributes and operations of each class and the relationships between the different classes.

These analysis models are an important part of the software development process. They help to ensure that the system requirements are well-defined and that the system design is appropriate for the intended use. They also help to identify potential problems early in the development process, which can save time and money in the long run.

**Appendix C: To Be Determined List**

The To Be Determined (TBD) List is a document that contains a list of items or issues that have not been fully defined or resolved during the analysis phase of a project. These items may include functional or non-functional requirements, user interface design elements, system architecture components, or any other aspect of the project that requires further clarification or decision-making. The TBD List serves as a reminder to the project team of the outstanding items that need to be addressed and helps to ensure that nothing is overlooked as the project progresses. It is important to regularly review and update the TBD List to ensure that all items are eventually resolved and included in the final product.