**APPLICATION DEVELOPMENT – II**

**Software DESIGN specification**

*Submitted in partial fulfilment for the award of the degree*

*Of*

**Master of Science**

***in***

**Information Technology**

*by*

**Kalyani Payyavula**

**(15MIN0241)**



**School of Information Technology and Engineering**

June, 2019

**CONSULT DOCTOR ONLINE**

Software Design Specification

1.0

22.06.2019

Kalyani Payyavula

Software Engineer

(EMP.ID - 318562)

Table of Contents

[1. INTRODUCTION 6](#_Toc11774960)

[1.1 PURPOSE 6](#_Toc11774961)

[1.2 SCOPE 7](#_Toc11774962)

[1.3 REFERENCES 7](#_Toc11774963)

[1.4 DEFINITIONS, ACRONYMS AND ABBREVATIONS 8](#_Toc11774964)

[1.5 OVERVIEW 8](#_Toc11774965)

[2 SYSTEM OVERVIEW 8](#_Toc11774966)

[2.1 PRODUCT PERSPECTIVE 9](#_Toc11774967)

[2.2 PRODUCT FUNCTIONS 10](#_Toc11774968)

[2.3 EXTERNAL INTERFACE REQUIREMENTS 11](#_Toc11774969)

[2.3.1 User Interfaces 11](#_Toc11774970)

[2.3.2 Hardware Interfaces 11](#_Toc11774971)

[2.3.3 Software Interfaces 11](#_Toc11774972)

[2.3.4 Communications Interfaces 11](#_Toc11774973)

[2.4 FUNCTIONAL REQUIREMENTS 12](#_Toc11774974)

[2.4.1 Home Page Requirements: 12](#_Toc11774975)

[2.4.2 New user registration Requirements: 12](#_Toc11774976)

[2.4.3 Doctor Availability Requirements: 12](#_Toc11774977)

[2.4.4 Book Appointment Requirements: 13](#_Toc11774978)

[2.4.5 Check History Requirements: 13](#_Toc11774979)

[2.4.6 Profile Management Requirements: 14](#_Toc11774980)

[2.4.7 Doctor Requirements: 14](#_Toc11774981)

[2.5 NON- FUNCTIONAL REQUIREMENTS 15](#_Toc11774982)

[2.5.1 Performance: 15](#_Toc11774983)

[2.5.2 Reliability: 15](#_Toc11774984)

[2.5.3 Security: 15](#_Toc11774985)

[2.5.4 Maintainability: 15](#_Toc11774986)

[2.5.5 Portability: 15](#_Toc11774987)

[3 SYSTEM ARCHITECTURE 16](#_Toc11774988)

[3.1 ARCHITECURAL DESIGN 16](#_Toc11774989)

[3.1.1 Single-Page Applications: 16](#_Toc11774990)

[3.1.2 MERN Stack: 17](#_Toc11774991)

[3.1.3 React with Flux Architecture: 19](#_Toc11774992)

[3.1.4 High level Design of Applicaiton 21](#_Toc11774993)

[3.2 DESIGN RATIONALE 22](#_Toc11774994)

[3.2.1 Benefits of React JavaScript Library: 22](#_Toc11774995)

[3.2.2 Benefits of Node.js 23](#_Toc11774996)

[4 DATA DESIGN 26](#_Toc11774997)

[4.1 DATA DESCRIPTION 26](#_Toc11774998)

[4.2 DATA DICTIONORY 26](#_Toc11774999)

[5 COMPONENT DESIGN 28](#_Toc11775000)

[5.1 USER INTERFACE 28](#_Toc11775001)

[5.2 HARDWARE INTERFACES 28](#_Toc11775002)

[5.3 RELIABILITY 28](#_Toc11775003)

[5.4 PERFORMANCE 28](#_Toc11775004)

[5.5 SUPPORTABILITY 28](#_Toc11775005)

[5.6 SAFETY 29](#_Toc11775006)

[5.7 SECURITY 29](#_Toc11775007)

[5.8 DESIGN CONSTRAINTS 29](#_Toc11775008)

[6 HUMAN INTERFACE DESIGN 29](#_Toc11775009)

[6.1 OVER VIEW OF USER INTERFACE: 29](#_Toc11775010)

[6.2 SCREEN IMAGES 31](#_Toc11775011)

[6.2.1 Home Page: 31](#_Toc11775012)

[6.2.2 Book Appointment page: 32](#_Toc11775013)

[6.2.3 Register user page: 33](#_Toc11775014)

[6.2.4 Check history page: 34](#_Toc11775015)

[6.2.5 Book Appointment for authenticated user 35](#_Toc11775016)

[6.2.6 Edit Profile page: 36](#_Toc11775017)

[6.2.7 Doctor Home Page: 37](#_Toc11775018)

[7 APPENDICES 38](#_Toc11775019)

[8 CONCLUSION: 38](#_Toc11775020)

**Document Approval**

The following Software Design Specification has been accepted and approved by the following:

|  |  |  |  |
| --- | --- | --- | --- |
| **SIGNATURE** | **NAME** | **TITLE** | **DATE** |
|  | Kalyani Payyavula | Consult Doctor Online | 21/06/2019 |
| **SIGNATURE** | **Evaluator Name** | **Comments** | **DATE** |
|  |  |  |  |

# INTRODUCTION

As the population continues to grow, so does the need for healthcare services. Exponential increase in the number of patients, creates new challenges for facility administrators and staff. Legacy processes and procedures that clinics followed may no longer be effective in handling a raise in new patients. One process affected almost every patient and doctor from an increase in patients is appointment-scheduling. This seemingly simple task can quickly become burdensome and challenging to staff members. Most of the clinics and doctors tend to do scheduling appointment over the phone which consumes most of the time and resources. In the next few years most of the individuals prefer to do common tasks online instead of picking up the phone and calling doctor.

Few benefits of online appointment are Time Savings, Monetary Saving and 24 -hour convenience. Let’s take simple example of clinic which does 100 appointments daily on phone. Each appointment call can take an average of 4 minutes which result in 400 minutes or seven hours. One staff member is dedicated to take appointments. Both time and expense required for this staff member can be saved. Sending automated remainders to patients helps to decrease the number of ‘no-shows’. This can directly impact on the revenue of the clinic. Scheduling appointments over the phone usually requires an individual to phone in during office hours. This can cause inconvenience for most patients, as they too are working at this time. An online scheduling system allows for 24-hour scheduling, not just during normal facility of office hours.

Facilities or Clinics with younger patients and clients can have an easy transition to online-appointment, as this group is typically internet savvy and actually prefer to schedule appointments online. On the other hand, hospitals with large number of senior patients and clients would not have an that easy transition to online appointment. However, few surveys show that Internet users in the age group of 50-64 grew exponentially day by day. In order to address the group which doesn’t use internet, clinics may continue to do phone appointment for few more years.

## PURPOSE

In my observation almost, all clinics or doctors are trying to solve the same problem, allow patients to book appointments round the clock. I would like to provide a common solution which can solve the problem of all patients and doctors. Provide a configurable application which can be configured for a particular clinic or doctor. Allow administrator to add/delete doctors to the clinic. Patients should be able to book appointment right from the mobile browser or desktop browser. Scheduling appointment online can benefit everyone involved in the scheduling process, as administrators and staff can conduct their task more efficiently and accurately, while customers and clients have the ability to book their appointments and reservations quickly and more conveniently.

## SCOPE

This document will cover design aspects on allowing users to book doctor appointments online. This covers both functional and non-functional aspects of the use case. In the functional scope it covers about the use case of patient, doctor and clinic admin. It covers in detail about how the application allows users to register, login and book appointments. It also covers about how users can check the history of the appointments. To make the use case complete this talk about how users can cancel/edit the appointments which are already in place.

In the non-functional aspects of the use case this document talks about how to avoid hackers to make appointment. This covers on how the system should be build robust so that it won’t execute the scripts injected by hackers. This also covers requirements on system performance, portability and saving user identity.

This document doesn’t speak about the design or technical aspects of the implementation. For more information on design, implementation and testing, I would recommend to refer respective documents.

## REFERENCES

|  |  |
| --- | --- |
| Reference |  |
| Document location | <https://github.com/KalyaniPayyavula/consult_doctor/tree/master/documents> |
| Implementation & Testing documents | <https://github.com/KalyaniPayyavula/consult_doctor/tree/master/documents> |
| Apollo Appointment | <https://www.askapollo.com/physical-appointment/> |
| Manipal Appointment | <https://www.manipalhospitals.com/appointments> |
| Sakra Hospital Appointment | <https://www.sakraworldhospital.com/request-appointment.php> |
| Online appointments | <https://www.appointmentplus.com/> |
| Single Page Application | <https://msdn.microsoft.com/en-us/magazine/dn463786.aspx> |
| MERN Stack | <https://www.mongodb.com/blog/post/the-modern-application-stack-part-1-introducing-the-mean-stack> |
| React with Fluz Architecture | <https://dzone.com/articles/a-detailed-study-of-flux-the-reactjs-application-a> |
| Express | <https://expressjs.com/> |
| MongoDB | https://www.mongodb.com/ |

## DEFINITIONS, ACRONYMS AND ABBREVATIONS

FR – Functional Requirements

NFR – Non Functional Requirements

NURR – New User Registration Requirements

DAR – Doctor Availability Requirements

BAR – Book Appointment Requirements.

PM – Profile Management

CH – Check History

## OVERVIEW

The rest of this document contains the following in the mentioned order:

* General description of the project and its requirements.
* Specific requirements for the project including the functionality, usability, reliability, performance security, safety, design constraints, etc.
* Supporting information in order to understand the project including diagrams which represents the system

# SYSTEM OVERVIEW

The appointment-scheduling process, historically viewed as a necessary burden in medical offices, healthcare facilities and wellness centers, can be completely automated through an inefficient online scheduling software program. The benefits of implementing this technology touch everyone involved in the scheduling process, as administrators and staff can conduct their tasks more efficiently and accurately, while customers and clients have the ability to book their appointments and reservations quickly and more conveniently.

The Benefits of an Online Scheduling System:

Commonly referred to by such names as online scheduling software, online booking applications and online scheduler, an online scheduling system is a Web-based application that allows individuals to conveniently and securely book their appointments and reservations online through any Web-connected device, such as a computer, laptop, smartphone or tablet. They typically access the online scheduling system through a “Book Now” button found on a Web site or page, or from a URL provided to them by the medial, healthcare or wellness facility. Once a date and time are selected, the system will automatically confirm the booking and instantly record it within the system, without any staff action needed.

In addition to online scheduling, online scheduling systems also come equipped with other beneficial features like automated e-mail and text message reminders, which the system sends out to patients and booked individuals on a specific date prior to their scheduled appointment; recording and record-keeping capabilities that make it quick and simple to access data associated with a specific appointment; and repeat patient reminders, which the system sends out automatically when a specified amount of time has expired between appointments.

24-Hour Convenience:

Scheduling appointments over the phone usually requires an individual to phone in during office hours, as few facilities offer round-the-clock phone booking. This is an inconvenience for most patients, as they too are working at this time. Additionally, many individuals prefer to schedule their appointments online rather than over the phone. An online scheduling system allows for 24-hour scheduling, not just during normal facility or office hours.

Time savings:

Staff spends less time on the phone booking and managing appointments, thereby freeing up their schedule for more important and pressing tasks. Booking individuals also save time, as they no longer have to commit a part of their busy schedule to calling their medical, healthcare or wellness provider (or remain on hold, which adds minutes to the scheduling process).

As an example, let’s look at a large medical facility that typically schedules approximately 100 appointments daily. Each appointment call is fielded by an administrative support staffer, who spends an average of four minutes on the phone. This equates to an average of 400 minute or almost seven hours of time spend each day just to booking appointments over the phone.

That’s time savings just from scheduling appointments alone. Other tasks automated by an online scheduling system, such as automated appointment reminders, add additional time savings to daily operations.

## PRODUCT PERSPECTIVE

This could have an effect on the success of an online scheduling system whose goal is to provide online scheduling. Facilities, centers and practices with younger patients and clients may have an easy transition, as this group is typically Internet savvy and actually prefer to schedule appointments online. On the other end of the spectrum are sites with a large number of senior patients and clients. HIPAA compliance. Given the online interaction and transfer of information between a facility and its patients, compliance of Health Insurance Portability and Accountability Act (HIPAA) provisions is an important consideration when utilizing an online scheduling system.

## PRODUCT FUNCTIONS

There are many eventual users who will use my application. The general characteristics of the eventual users of the product that will affect the specific requirements are as below:

1. **New Patient:** New Patient is someone who is approaching doctor or facility for the first time.

Functionalities:

* Register himself with all personal details.
* Check doctor availability and book appointment.
* Check history of appointments.
* Edit/ delete appointments already made.
* Edit his personal profile.
* Take appointment for his/her family member.

1. **Doctor:**  User authorized by admin as doctor.

Functionalities:

* Updates his/her availability.
* Can have access to patient details who has taken appointment.
* Mark appointment as completed/handled.
* Give future appointments for patients.
* Ability to cancel appointments on emergency.

1. **Admin:**  User authorized to add doctor profile and generate access credentials for doctors.

Functionalities:

* Updates list of doctors available in the clinic.
* Has permissions to add/remove doctors.
* Controls appointment remainders.
* Maintains the system and sole responsible for maintaining appointments.

**2.3.** GENERAL CONSTRAINTS

While the actual implementation of an online scheduling system is typically seamless and relatively simple, there are considerations that medical, healthcare and wellness facilities should keep in mind when transitioning to Web-based booking. They include:

Optional or required? One question administrator should answer is whether or not to make online scheduling a requirement. Requiring that all appointments be made online can certainly free up staff responsibilities and schedules, but it can also be a hindrance to those without easy access to the Internet or who prefer to schedule their appointments over the phone. Many facilities give their patients and clients the option of booking online, which typically brings good results.

## EXTERNAL INTERFACE REQUIREMENTS

### User Interfaces

* System shall provide browser interface to use the system.
* Application shall launch successfully in all popular browsers like IE, Google Chrome, Safari and Firefox.
* System shall also allow users to use Mobile browsers.
* System shall work fine with Android and iOS default browsers.

### Hardware Interfaces

* Application shall not take any input from hardware.
* Application shall save data to database only on valid session.
* Application shall not allow any external hardware to read user data.
* This system doesn’t read/write data to any hardware.

### Software Interfaces

* System shall not accept data from any external software.
* System shall write data to database server.
* System shall read data form database server after successful authentication.

### Communications Interfaces

* System shall communicate with database to store user details and doctor details.
* User details are only available to doctor with read access.
* User is the only interface which inserts data into the system.
* Doctor can only read the data of user, he cannot modify it.

## FUNCTIONAL REQUIREMENTS

### Home Page Requirements:

|  |  |
| --- | --- |
| Requirement ID | Requirement |
| FR-HomePage-1 | System shall display list of doctors and their specialization. |
| FR-HomePage-2 | List of doctors available in a clinic can be edited / modified. |
| FR-HomePage-3 | Home page should include doctor timings, doctor qualifications. |
| FR-HomePage-4 | Home page shall display a link to check availability of particular doctor. |
| FR-HomePage-5 | Home page shall display an option to register new user. |
| FR-HomePage-6 | Home page shall allow already registered user to login. |
| FR-HomePage-7 | Home page shall allow Guest user to check his Appointments. |

### New user registration Requirements:

|  |  |
| --- | --- |
| Requirement ID | Requirement |
| FR-NURR-1 | System shall allow user to register with his personal details so that he can use them later to book appointment. |
| FR-NURR-2 | System shall ask for following details.   * User Id \* * Password\* * Confirmation password\* * Email\* * Mobile Number\* * Address: Line 1, Line 2, Pin Code   Fields marked with \* are mandatory. |
| FR-NURR-3 | System shall validate user provided details and provide a confirmation message to the user. |

### Doctor Availability Requirements:

|  |  |
| --- | --- |
| Requirement ID | Requirement |
| FR-DAR-1 | System shall display list of available dates with green color. |
| FR-DAR-2 | On selecting a date with green color, system shall display an overlay to the user with list of available Slots. |
| FR-DAR-3 | System shall allow user to check doctor availability for next 3 months. |
| FR-DAR-4 | If doctor is not available for any reason system shall display such reason on the screen. |

### Book Appointment Requirements:

|  |  |
| --- | --- |
| Requirement ID | Requirement |
| FR-BAR-1 | System shall allow user to pick an available slot and proceed with appointment booking. |
| FR-BAR-2 | System shall block a slot for a user and doesn’t show that slot for any other user. |
| FR-BAR-3 | System shall allow user to book appointment even without login as guest user. |
| FR-BAR-4 | Guest user will be asked to enter details like  Name\*  Mobile number\*  Email id\*  Problem description\*  Address:  Fields marked with \* are mandatory. |
| FR-BAR-5 | System shall allow registered user to login and book appointment. In this use case System shall pick user details directly. |
| FR-BAR-6 | System shall display an option to select if the appointment is for his/her family member. |
| FR-BAR-7 | When user selects family member option he will be asked to enter the details of family member. |
| FR-BAR-8 | Family details shall be saved by the system so that they will not be asked to enter again. |
| FR-BAR-9 | Each slot shall be 15 mins. This should be configurable by doctor. |
| FR-BAR-10 | Each user can book a maximum of 2 appointments per doctor. |

### Check History Requirements:

|  |  |
| --- | --- |
| Requirement ID | Requirement |
| FR-CH-1 | System shall allow registered user to check his history of appointments. |
| FR-CH-2 | If an appointment is in future user will be allowed to edit/cancel that appointment. |
| FR-CH-3 | When user cancel an appointment, it will be made available to other users for use. |
| FR-CH-4 | If user wishes to edit the time of appointment user shall be presented with available slots in the same page. |
| FR-CH-5 | When user cancel appointment, free the slot. |

### Profile Management Requirements:

|  |  |
| --- | --- |
| Requirement ID | Requirement |
| FR-PM-1 | System shall allow user to check his profile. |
| FR-PM-2 | System shall allow user to edit his profile if user feels so. |
| FR-PM-3 | System shall allow user to add his family members details. |
| FR-PM-4 | User can never change his user id once registered. |

### Doctor Requirements:

|  |  |
| --- | --- |
| Requirement ID | Requirement |
| FR-DR-1 | System shall allow admin to add doctor to clinic. |
| FR-DR-2 | Once doctor is added to clinic he/she will be provided login credentials with which they are allowed to enter their availability in the clinic. |
| FR-DR-3 | System shall allow doctor to check his filled slots for the next 3 months. |
| FR-DR-4 | System shall allow doctor to check the profile of the patient. |
| FR-DR-5 | System shall allow doctor to enter notes about the patient findings. |
| FR-DR-6 | System shall allow doctor to check previous visits of the patient and his findings. |

## NON- FUNCTIONAL REQUIREMENTS

A non-functional requirement is a [requirement](https://en.wikipedia.org/wiki/Requirement) that specifies criteria that can be used to judge the operation of a system, rather than specific behaviors. Non-functional requirements place restrictions on the product being developed, the development process, and specify external constraints that the product must meet.

### Performance:

|  |  |
| --- | --- |
| NFR-Perf-1 | Application shall launch in no less than 1 seconds on good internet connection. |
| NFR-Perf-1 | User doesn’t see any lag while checking the doctor availability. |
| NFR-Perf-1 | Application shall launch in mobile browsers with no lag. |
| NFR-Perf-1 | Application shall perform well on Smart phones with 3G connections as well. |

### Reliability:

|  |  |
| --- | --- |
| NFR-Reli-1 | Application must be robust enough to make sure that their data is not compromised. |
| NFR-Reli-2 | Only doctor can check the data of the patient. |
| NFR-Reli-3 | Office staff shall not have access to the patient problem statement. |
| NFR-Reli-4 | If a family member books an appointment for his children he can access the data. |

### Security:

|  |  |
| --- | --- |
| NFR-Secu-1 | Application should avoid hackers to book appointment. |
| NFR-Secu-2 | Application Must take care that no scripts can be inserted as input. |
| NFR-Secu-3 | Application Must not execute user inputs directly on the database. |
| NFR-Secu-4 | Application shall not allow hacker to access the database. |

### Maintainability:

|  |  |
| --- | --- |
| NFR-Main-1 | Application should be built in such a way that it can be maintained with minimum cost. |
| NFR-Main-2 | Adding new features to the application should be with minimum cost. |
| NFR-Main-3 | Application should be robust enough to upgrade the libraries. |

### Portability:

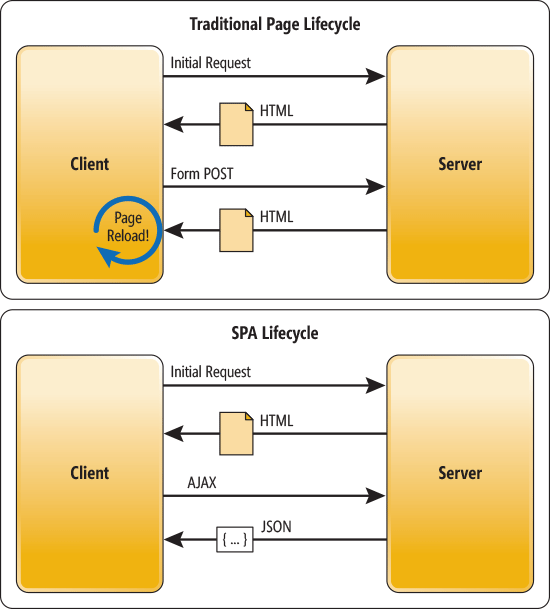
|  |  |
| --- | --- |
| NFR-Port-1 | Application can be launched on different desktop browsers. |
| NFR-Port-2 | Application shall adapt well with mobile browsers. |
| NFR-Port-3 | All popular browsers can launch the application. |
| NFR-Port-4 | Application can be converted to mobile application with minimum cost. |

# SYSTEM ARCHITECTURE

## ARCHITECURAL DESIGN

### Single-Page Applications:

Single-Page Applications (SPAs) are Web apps that load a single HTML page and dynamically update that page as the user interacts with the app.



In a traditional Web app, every time the app calls the server, the server renders a new HTML page. This triggers a page refresh in the browser. In an SPA, after the first page loads, all interaction with the server happens through AJAX calls. These AJAX calls return data—not markup—usually in JSON format. The app uses the JSON data to update the page dynamically, without reloading the page.

One benefit of SPAs is obvious: Applications are more fluid and responsive, without the jarring effect of reloading and re-rendering the page. Another benefit might be less obvious and it concerns how you architect a Web app. Sending the app data as JSON creates a separation between the presentation (HTML markup) and application logic (AJAX requests plus JSON responses).

In a pure SPA, all UI interaction occurs on the client side, through JavaScript and CSS. After the initial page load, the server acts purely as a service layer. The client just needs to know what HTTP requests to send. It doesn’t care how the server implements things on the back end.

With this architecture, the client and the service are independent. You could replace the entire back end that runs the service, and as long as you don’t change the API, you won’t break the client. The reverse is also true—you can replace the entire client app without changing the service layer. For example, you might write a native mobile client that consumes the service.

### MERN Stack:

I would like to choose MERN stack architecture for implementing this application. Here is the high-level architecture of MERN depicting how each component are connected.

#### MongoDB: A cross-platform document database

[MongoDB](https://www.mongodb.com/) is a NoSQL (non-relational) document-oriented database.

While conventional relational databases have a typical schema design based on columns and tables, MongoDB is schema-less. Data is stored in flexible documents with a JSON (JavaScript Object Notation)-based query language. The content, size, and number of fields in the documents can differ from one to the next. This means that the data structure to be changed over time.

MongoDB is known for being flexible and easy to scale. You can see multiple examples of [real-life MongoDB applications here](https://www.mongodb.com/use-cases).

#### Express: A back-end web application framework

[Express](https://expressjs.com/) is a web application framework for Node.js, another MERN component. Instead of writing full web server code by hand on Node.js directly, developers use Express to simplify the task of writing server code. There’s no need to repeat the same code over and over, as you would with the Node.js HTTP module.

The Express framework is designed for building robust web applications and APIs. It’s known for its fast speed and minimalist structure, with many features available as plugins.

#### React: A JavaScript library for building user interfaces

[React](https://reactjs.org/) was originally created by a software engineer at Facebook, and was later open-sourced. It is maintained by Facebook, as well as a community of development companies and individual developers.

The React library can be used for creating views rendered in HTML. React views are declarative. This means that developers don’t have to worry about managing the effects of changes in the view’s state (the object that determines how components behave) or changes in the data.

Instead of relying on templates to automate the creation of repetitive HTML or DOM (Document Object Model) elements, React uses a full-featured programming language (JavaScript) to construct repetitive or conditional DOM elements.

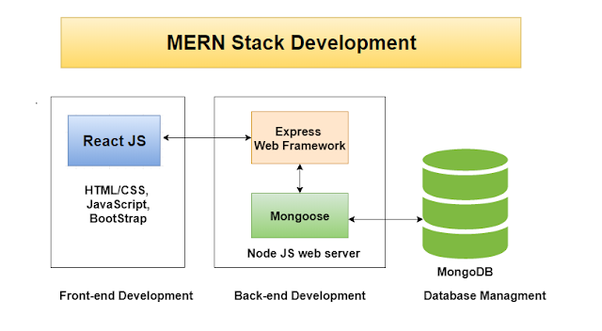
With React, the same code can run on both the server and the browser.

React anchors the MERN stack. In a way, it’s the defining feature of the stack. It’s the one component that differentiates MERN from MEAN, another popular JavaScript stack that uses AngularJS (a front-end web application framework) instead of the React library.

#### Node.js: A cross-platform JavaScript runtime environment

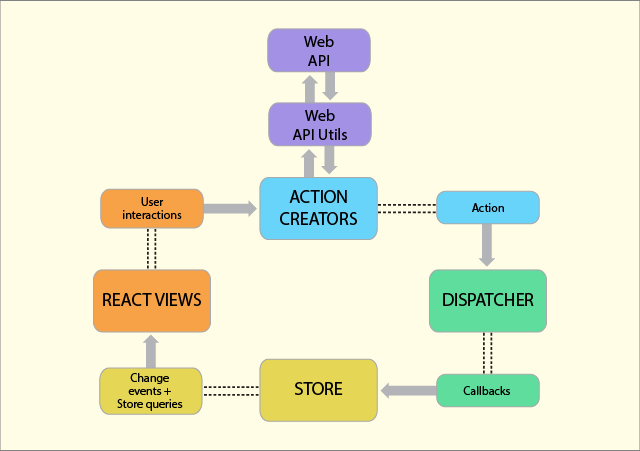
[Node.js](https://nodejs.org/en/) was initially built for Google Chrome, and later open-sourced by Google in 2008. It is built on Chrome’s V8 JavaScript engine. It’s designed to build scalable network applications, and can execute JavaScript code outside of a browser.

Node.js works without an enclosing HTML page, instead using its own module system based on CommonJS, to put together multiple JavaScript files.



### React with Flux Architecture:

The components in Flux's architecture interact more like an EventBus and less like an MVC. As mentioned earlier, Flux is not actually a library or a framework, it is a new kind of architecture that Facebook created to work with React. Hence the main function of Flux is to complement React and promote Unidirectional Data Flow.



In a typical Flux architecture, you will find the following components:

Actions - Helpers that pass data to the Dispatcher.

Dispatcher - Receives these Actions and broadcasts payloads to registered callbacks.

Stores - Act as containers for application state and logic. The real work in the application is done in the Stores. The Stores are registered to listen in on the actions of the Dispatcher and update the Views according to these actions.

Controller Views - React Components grab the state from the stores and then pass it down to the child components.

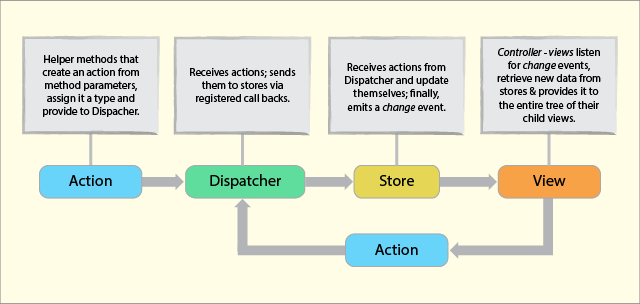
The Controllers in the MVC and Flux are different. In Flux, the Controllers are Controller-Views and are found at the very top of the hierarchy. Views are React components. All the functionality is usually found in the Store. The Store is where all the work is done and tells the Dispatcher which events/actions it is listening for.

When an event happens, the Dispatcher would send the “payload” to the Store that is registered to listen for that particular action. Now it is up to the Store to update the View, which in turn triggers an action. The action to that will occur is also predetermined, like name, the type of action, and so on.

The View propagates the Action through a central Dispatcher and this will be sent to various Stores. These Stores contain an application’s business logic and other data. It updates all the Views. It works best with React’s programming style and the Store sends updates without the need to provided detailed code on how to transition views between states.

This proves that the Flux pattern follows a unidirectional data flow. The Action, Dispatcher, Store, and View are independent nodes with specific inputs and outputs. The data flows through the Dispatcher, the central hub, which in turn manages all the data. The Dispatcher acts as a registry with registered callbacks that the Stores respond to. Stores will emit a change which will be picked by the Controller-Views.

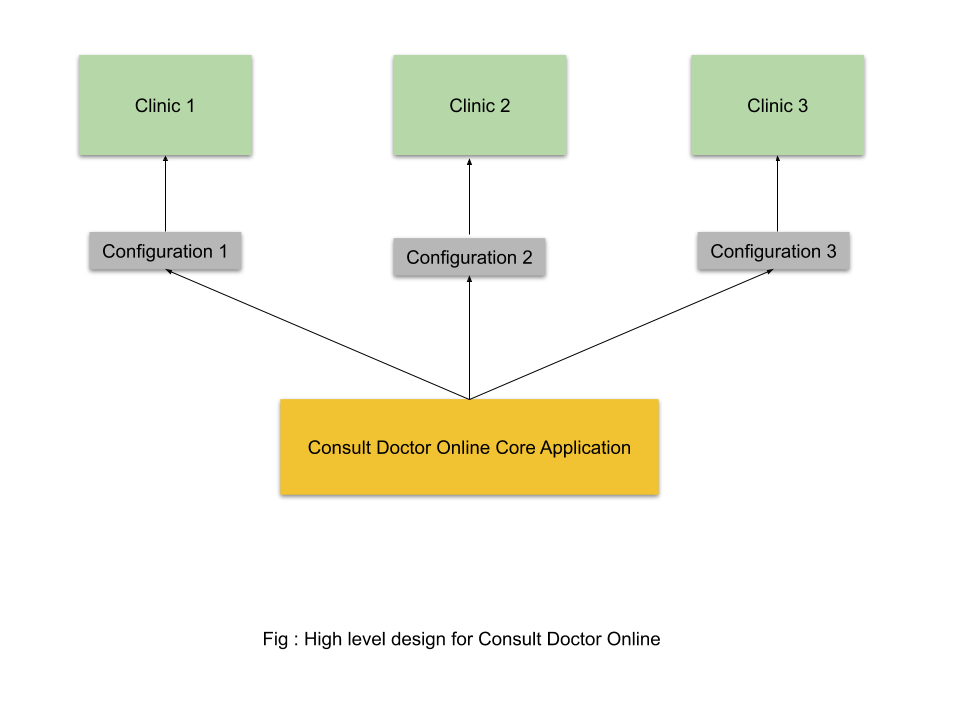
This is what happens when a View is propagated in the system:



This proves that there are no two-way bindings, the structure is akin to functional relative programming, and more something like flow-based programming. The dependencies that occur in the stores are kept in a strict hierarchy, while the Dispatcher handles the updates. This structure also solves the problems that come naturally with two-way binding. To create a Dispatcher, you need to bring the Dispatcher from the Flux. You can handle this by using libraries such as Dispatcher.js.

### High level Design of Applicaiton

Main Intention behind this application is to develop a single application which can be configured for a particular clinic. In this way we can save time and effort to develop application for an individual Clinic or facility.



Configurations can be any JSON file which can load the doctor information. Deployments can be taken care in such a way that each clinic can own an application as if it is developed for them. In this way we can take care of security.

## DESIGN RATIONALE

### Benefits of React JavaScript Library:

#### **Better User Experience**

Unlike other JavaScript frameworks, React uses the Virtual DOM – the abstract form of Real DOM. This makes it easier for the React app developers to update changes performed by the users in the application without affecting the other parts of the interface.

This results in building a highly dynamic UI with the exquisite user experience.

#### **Time-Saving**

In the case of React, the app development companies can reuse the code components at distinct levels at any point of time. Besides, the components are isolated to each other and changes in one does not affect the other, which makes it easier to manage the updates.

This makes the React app development easier, time-saving, and efficient for developers.

#### **Quick Development**

React allows the developers to reuse the existing code and apply hot reloading into the process. This approach not only improves the app performance but also accelerates the development speed.

#### **Faster Testing**

React extensively uses Redux which cut down the hassle of storing and managing component states in large-sized and complex applications with enormous dynamic elements.

It helps the developers to add application state in a single object and empower every component of the app to access the application state without involving child components or using callback. This makes it easier to test the application and log data changes, along with the use of hot reloading and other such tools.

#### **Code Stability with One-directional data binding**

ReactJS let the developers work directly with the components and employ downward data binding to ensure that the parent entities do not get affected by the changes of child entities. This approach makes the code stable and supports the idea of development in the future.

### Benefits of Node.js

#### **Node.js offers an Easy Scalability**

One of the key advantages of Node.js is that developers find it easy to scale the applications in horizontal as well as the vertical directions. The applications can be scaled in horizontal manner by the addition of additional nodes to the existing system.

Moreover, Node.js also offers you the option of adding extra resources to the single nodes during the vertical scaling of the application. So, it is highly scalable and provides better option than other JavaScript servers.

#### **Easy to Learn**

Since JavaScript is one of the most popular programming languages, most of the front-end developers have a good grasp over it.

It becomes much easier for them to start using the Node.js at the backend. It is easier to learn Node.js and consumes less time to work with it.

#### **Node.js is used as a Single Programming Language**

Node.js offers the developers the luxury of writing the server-side applications in the JavaScript. This allows the [Node.js developers](https://www.mindinventory.com/hire-node-js-developers.php) to write both the front-end as well as the back-end web application in JavaScript using a runtime environment.

And they don’t need to use any other server-side programming language. It also makes the deployment of the web applications simpler because almost all the web browsers support JavaScript.

#### **The Benefit of Fullstack JS**

Node.js has been regarded as a full-stack JavaScript for serving both the client and the server-side applications.

Therefore, the advantage is that you don’t have to hire separate developers for backend as well as the front-end development. It saves both your valuable money and time.

#### **Known for Offering High Performance**

It has been mentioned earlier that Node.js interprets the JavaScript code via Google’s V8 JavaScript engine. This engine complies the JavaScript code directly into the machine code. This makes it easier and faster to implement the code in a effective manner.

The speed of the code execution also enhanced by runtime environment as it supports the non-blocking I/O operations.

#### **The Support of Large and Active Community**

Node.js is blessed to have a large and active community of developers who keep on continuously contributing towards its further development and improvement.

In fact, the groups of developers are well supported by the JavaScript programmers providing ready-made and easy solutions and codes in GitHub. It is expected that the developers will initiate many further developers in the future.

#### **The Advantage of Caching**

The open-source runtime environment of the Node.js also provides the facility of caching single modules. Whenever there is any request for the first module, it gets cached in the application memory.

The developers don’t have to re-execute the codes as caching allows applications to load the web pages faster and responds more swiftly to the user.

#### **Offers the Freedom to Develop Apps**

Another advantage that Node.js offers to the developers is the freedom to develop the apps and software.

This is one essential feature, which remains absent in Ruby on Rails imposing certain guidelines. You can begin everything from the scratch while developing applications.

#### **Getting Support for Commonly Used Tools**

With Node.js, the developers can get an extended support for the various commonly used tools. Let’s take an example. Suppose, you want to test the source code of Node.js application; you can do so by using the Jasmin and other such unit-testing tools.

Similarly, if you want to identify and install the project dependencies, you can make use of npm, a powerful package manager. You can use grunt for task running of the project.

#### **Handles the Requests Simultaneously**

Since the Node.js is providing the option of non-blocking I/O systems, it relatively helps you to process several requests concurrently.

The system can handle the concurrent request handling efficiently better than others including Ruby or Python. The incoming requests get lined up and are executed quickly and systematically.

#### **Node.js is Highly Extensible**

The Node.js is known to be highly extensible, which means that you can customize and further extend Node.js as per their requirements.

You can also make use of JSON to provide the scope for exchange of data between the web server and the client. It also is facilitated with built-in APIs for developing HTTP, TCP, and DNS etc. servers.

# DATA DESIGN

## DATA DESCRIPTION

The database subsystem stores al1 the data that the system can provide to its users and the inter-relationship between these data. Upon user requests, the Event Handler subsystem may accordingly search database for information, update data, or Save new data into the database through Client admission subsystem and Server admission subsystem. The Database subsystem is connected to Server admission subsystem by ODBC as middle ware. Through the query, it can get data from or save data to database. The query is attached with each component if that component needs to access database.

## DATA DICTIONORY

* + 1. **Registration Details Table**

The table holds records of registered users with their respective preferred usernames and passwords. It also has the contacts {phone numbers, and email address} of users. This same table is used by the user to get the username and password for logging in.

* Table structure for user registration details:

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Type** | **Null** | **Description** |
| ***User.ID*** | String(50) | No | Login ID of user. |
| Password | String(20) | No | Password to Login. |
| Email | String(50) | No | EMAIL ID of the user. |
| Mobile Number | String(50) | No | Mobile number of user. |

* Table structure for User Address.

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Type** | **Null** | **Description** |
| ***User.Id*** | String(25) | No | User Id. |
| Line1 | String(25) | No | Line 1 of the address. |
| Line 2 | String(25) | Yes | Will be null if user doesn’t enter. |
| City | String(25) | No | City entered buy user. |
| State | String(25) | No | State entered by user. |
| Zip Code | NumberInt(6) | No | Zip code of the user |
|  |  |  |  |

* Table structure for Doctor.

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Type** | **Null** | **Description** |
| ***Docotor.Id*** | String(20) | No | Doctor ID |
| Availability Hours | Object | No | Mongo DB object type. |
| Filled Slots | Object | No | Mongo DB object type. |
| Available Slots | Object | No | Mongo DB object type. |
|  |  |  |  |

# COMPONENT DESIGN

## USER INTERFACE

It will have a login screen. Login screen will have the option to take user Id and Password. Doctors will be provided interface to update their availability. Once the doctor attends the patient necessary action will be taken against the appointment.

## HARDWARE INTERFACES

As this application, can be accessed via internet, only web server is required as a hardware interface to be able to run it. It is assumed that web server is in a secure environment with necessary firewall and network setting done.

Mobiles phones and computers can be used here.

## RELIABILITY

The system should be reliable. Security is a major concern for an online consultation. Process used in this system should be secure enough to be able to meet the requirements mentioned for online appointment. It requires database connections and network connections. Changes can be done in the databases to store appointment details. All changes need to be confirmed and if the transfer is complete the confirmation should be displayed. The changes should be monitored.

## PERFORMANCE

There might be many users accessing to the web server simultaneously. As an online tool performance shouldn’t be affected much and response time for submitted page should be less than a minute.

## SUPPORTABILITY

Version 1.0 is the first version of the consult doctor online. Future release features will be considered during the design. Phone support can be provided with all dial in details in the initial version release. In the future versions this can be improved by supporting an online chat facility. Future enhancements or improvements can be considered during the design and development

## SAFETY

Web server should be secure. Regular backups should be in place.

## SECURITY

Customer’s registration information is confidential.

## DESIGN CONSTRAINTS

Consult doctor will be an online application and it can run from a machine that has an internet access. Internet connectivity is required. The online consultation application needs to be designed for a clinic or facility. It should be designed in such a way that almost all the clinics should get the benefit from this. Database access and web services access are required.

# HUMAN INTERFACE DESIGN

## OVER VIEW OF USER INTERFACE:

This application provides a responsive web design. Responsive web design is an approach whereby a designer creates a web page that “responds to” or resizes itself depending on the type of device it is being seen through.  That could be an oversized desktop computer monitor, a laptop or devices with small screens such as [smartphones](https://smallbiztrends.com/2018/05/get-ready-for-the-mobile-first-index.html) and tablets.

Responsive Web design has become an essential tool for anyone with a digital presence. With the growth of smartphones, tablets and other mobile computing devices, more people are using smaller-screens to view web pages.

Let’s take a traditional “fixed” website.  When viewed on a desktop computer, for instance, the website might show three columns. But when you view that same layout on a smaller tablet, it might force you to scroll horizontally, something users don’t like. Or elements might be hidden from view or look distorted.  The impact is also complicated by the fact that many tablets can be viewed either in portrait orientation or turned sideways for landscape view.

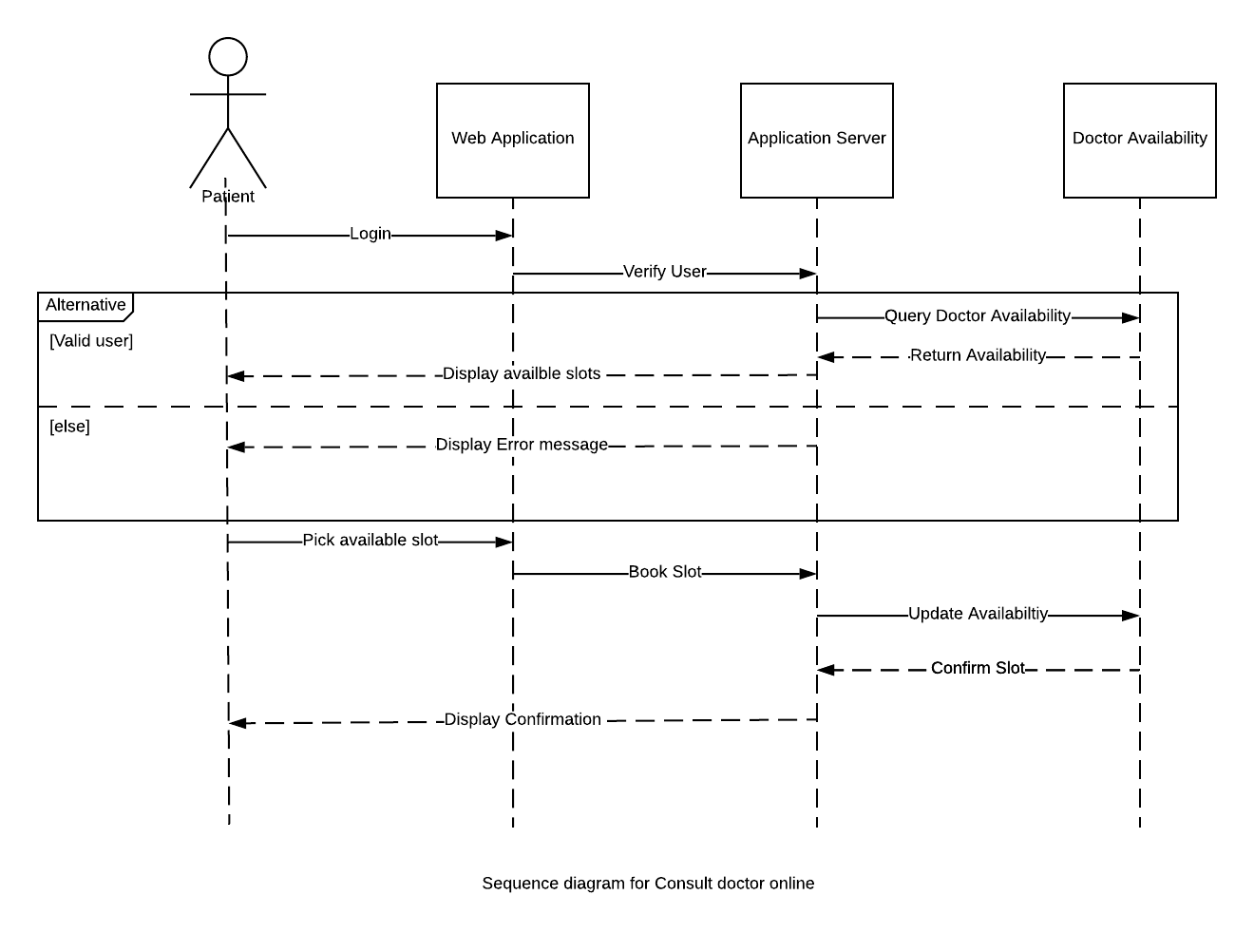
On a tiny smartphone screen, websites can be even more challenging to see. Large images may “break” the layout. Sites can be slow to load on smartphones if they are graphics heavy.

However, if a site uses responsive design, the tablet version might automatically adjust to display just two columns. That way, the content is readable and easy to navigate. On a smartphone, the content might appear as a single column, perhaps stacked vertically.  Or possibly the user would have the ability to swipe over to view other columns.  Images will resize instead of distorting the layout or getting cut off.

The point is: with responsive design, the website automatically adjusts based on the device the viewer sees it in.

**User interacts with the system**

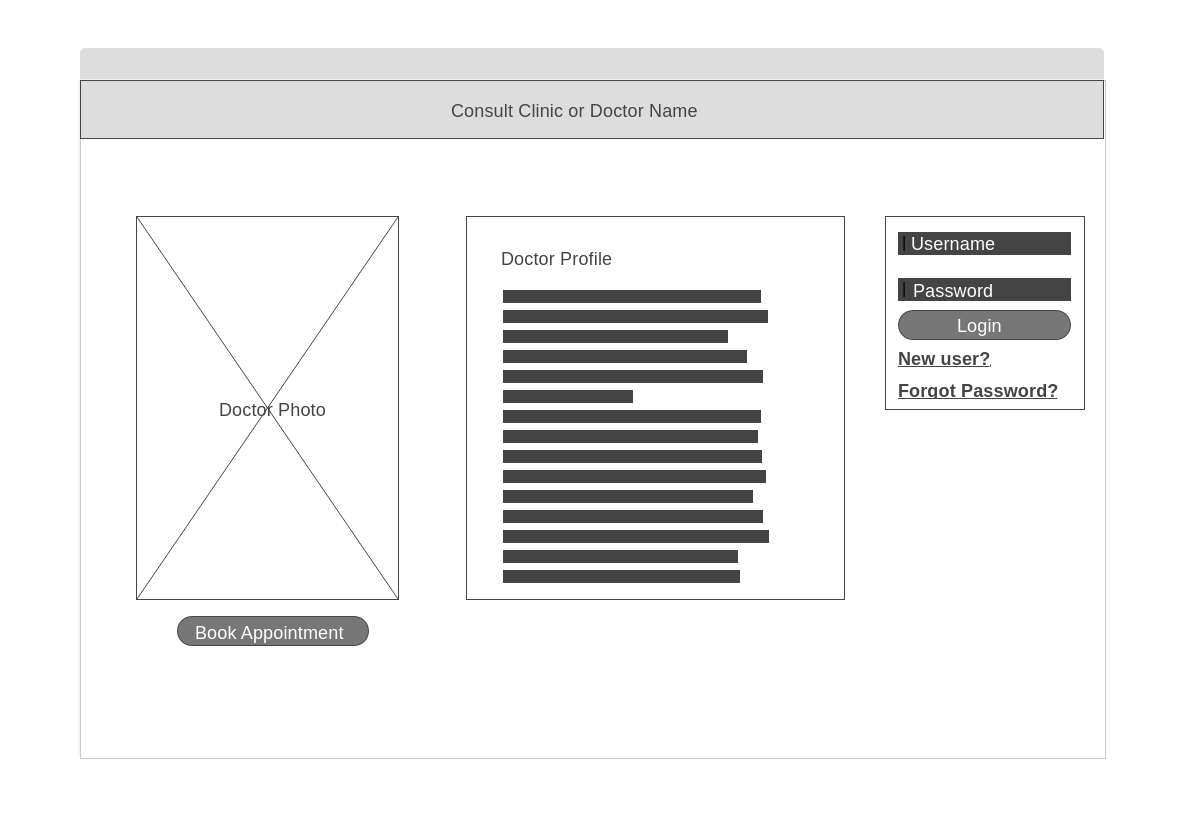
The sequence diagram shows how the user can book appointment with doctor online.



## SCREEN IMAGES

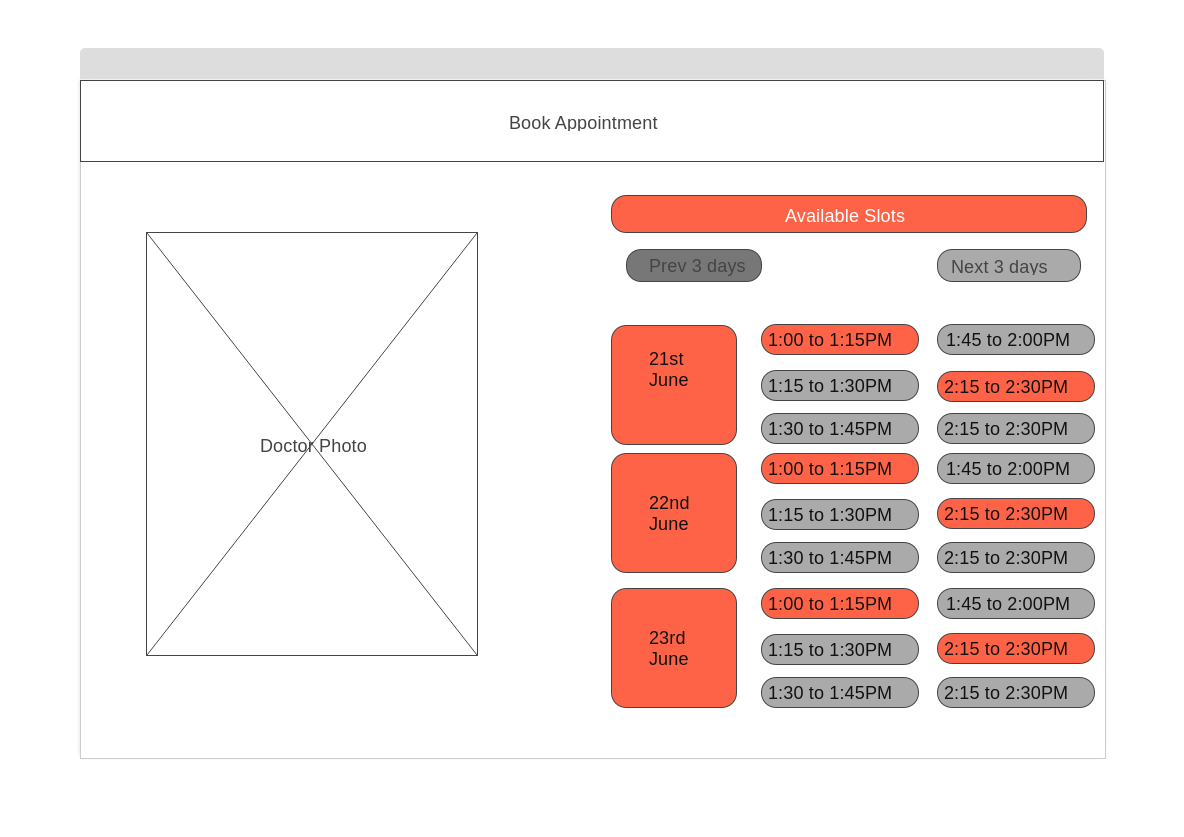
### Home Page:

This is the page which user will land in when he visits the application.



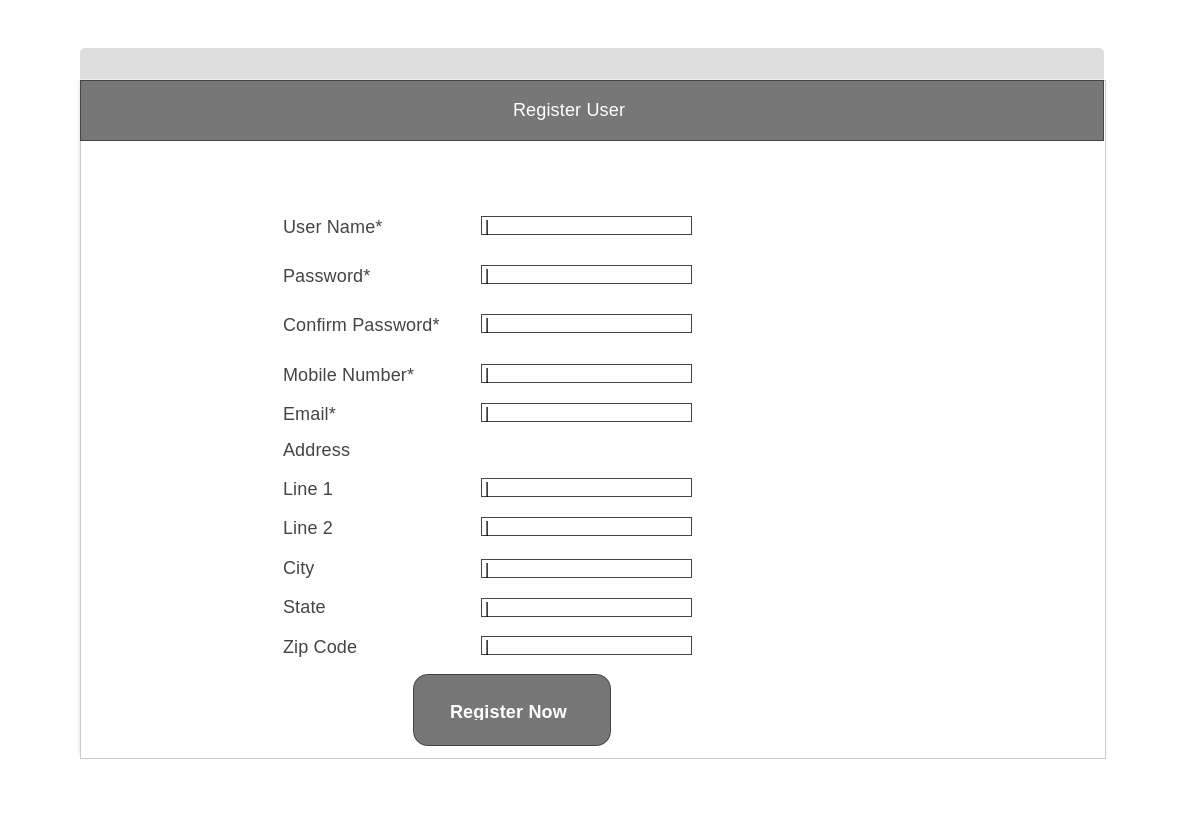
### Book Appointment page:

This page will help user to book appointment as guest.



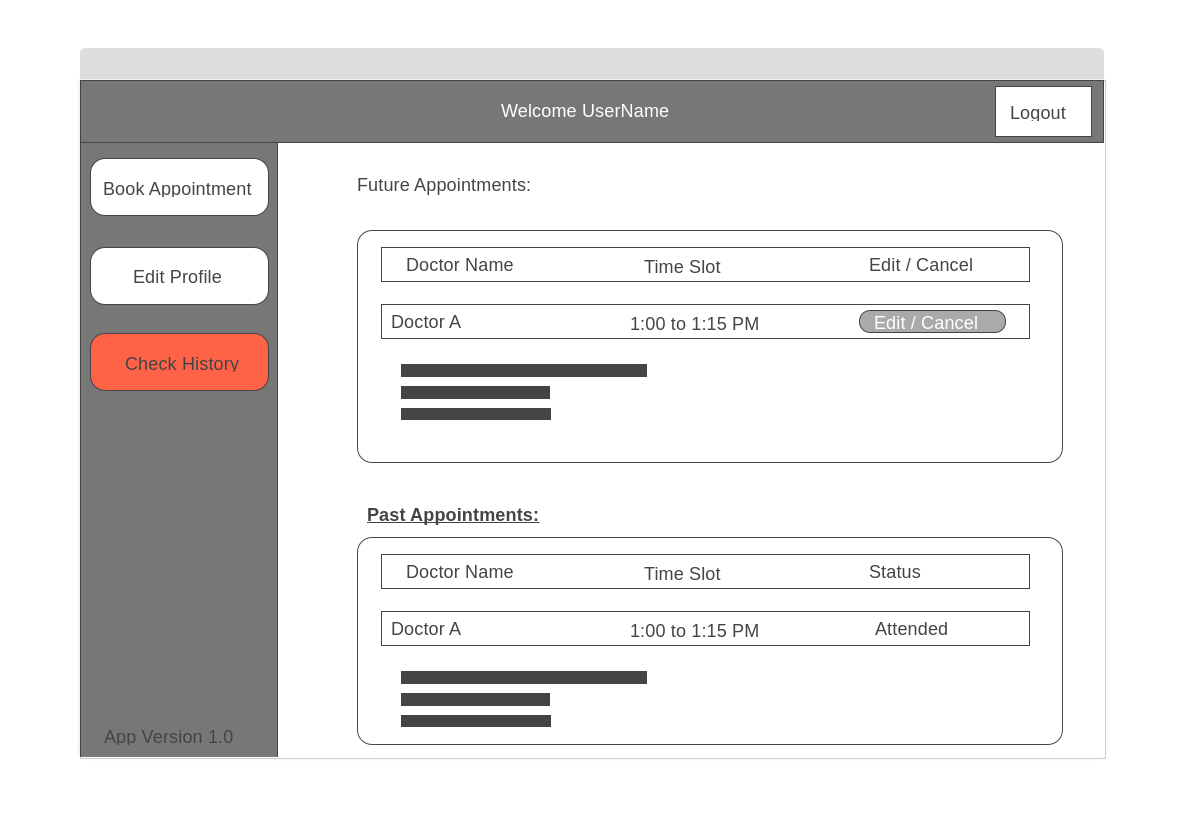
### Register user page:

This page will help user to register with clinic. Next time he is going to book his appointment he need not enter his details.

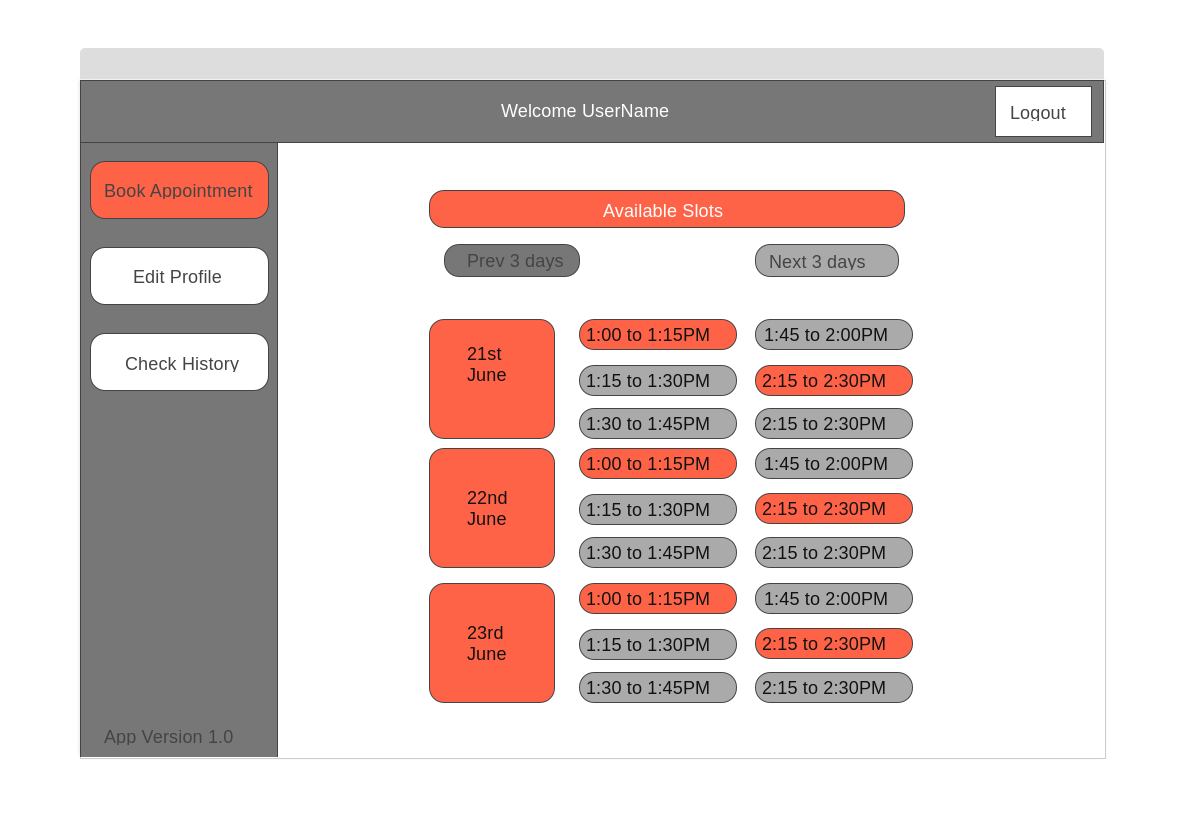


### Check history page:

In this page user can Edit/ Cancel his appointments. He can also check his previous visits to the clinic.

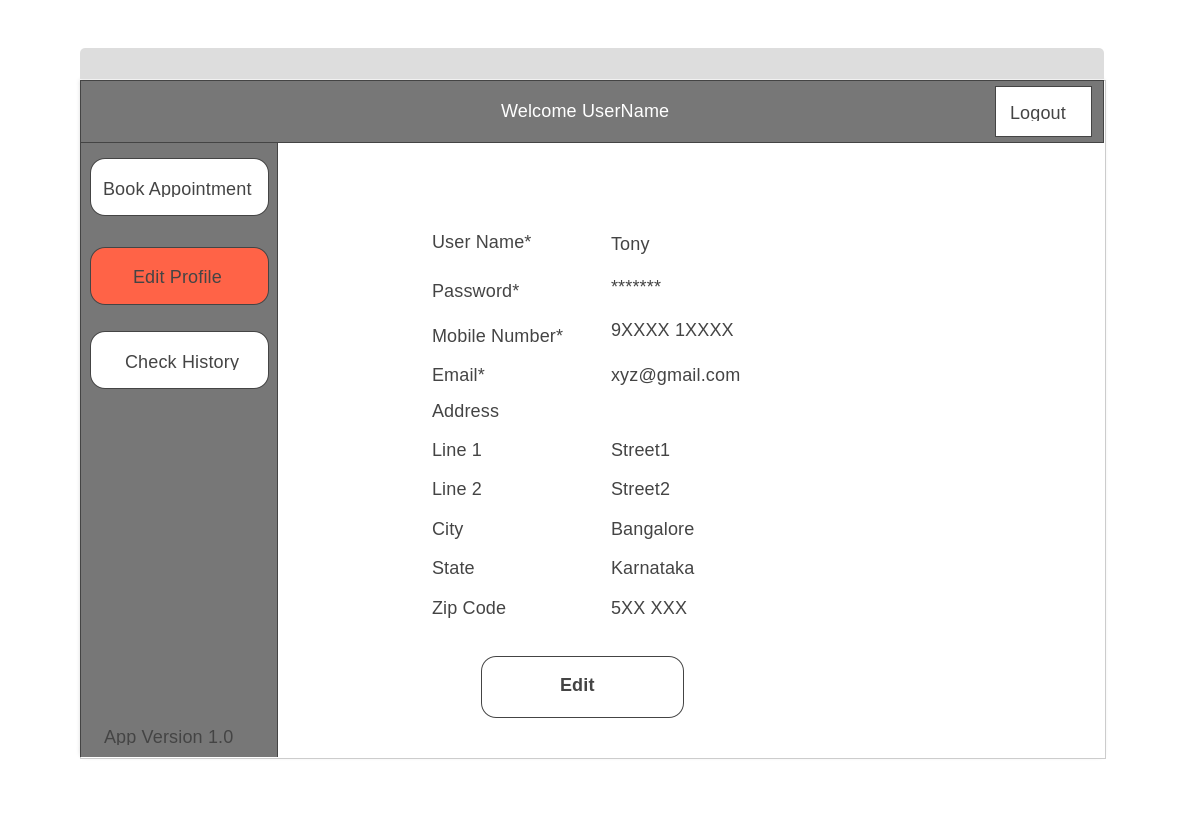


### Book Appointment for authenticated user



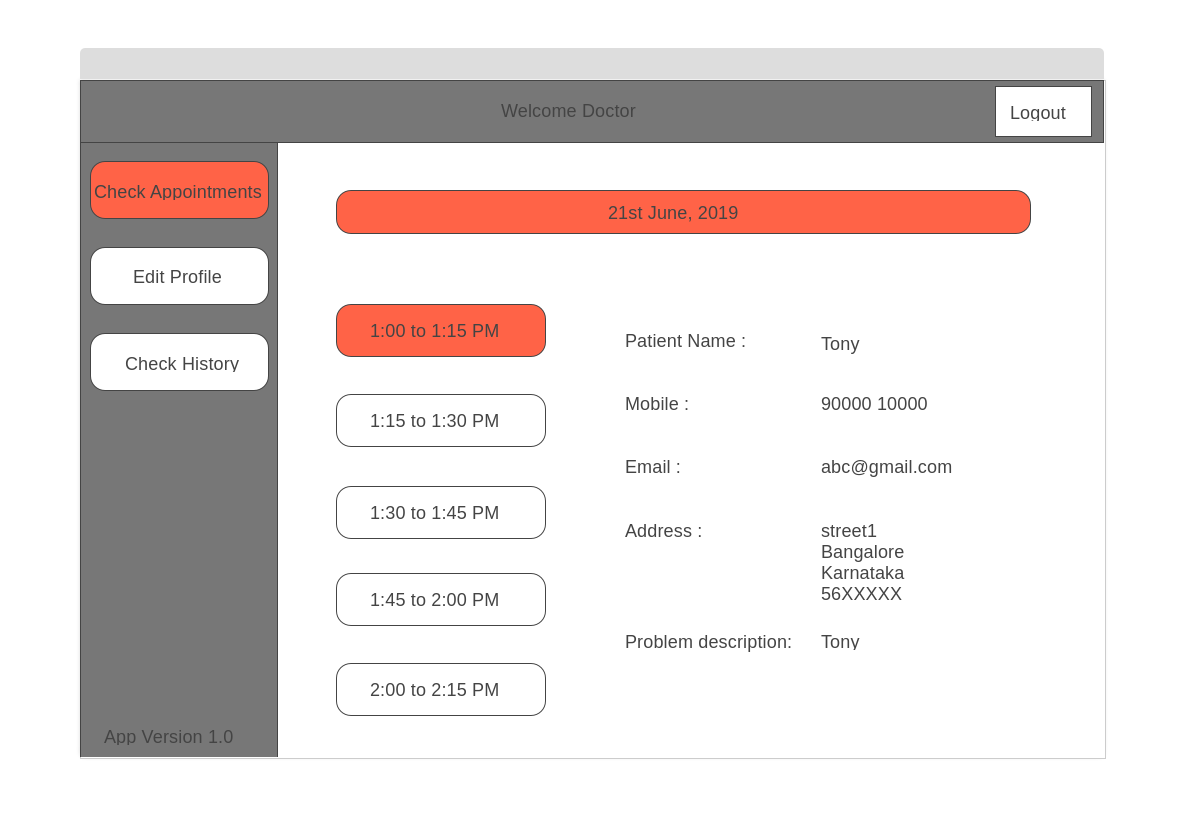
### Edit Profile page:

This page will display user details given while user registers. User can always edit that if he feels to do so.



### Doctor Home Page:

This is the page doctor will see once he is authenticated with application.



# APPENDICES

Each clinic or facility would like to have their own appointment system. Main advantage of doing this is 24-hour availability and Time Savings. This solution helps us to resolve the problem of all clinics or facilities rather than building concrete solution for each clinic.

# CONCLUSION:

This design specification document prepared for a better design of Consult Doctor online. The functional and other requirements of the systems are described here.