

|  |  |  |
| --- | --- | --- |
|  |  | |
|  |  |  | | |  | |
|  |  | Wintan Hospital  Online patient appointment system | | |  | |
|  |  | | | | |  |
|  | **Introduction**  Wintan Hospital is a leading private hospital in Sri Lanka. The hospital was established in September 2008. The vision of Wintan Hospital is providing ancillary services to the cancer hospital and caters to the growing needs of Maharagama, Kesbewa and Piliyandala electorates. The Wintan Hospital offers its clientele a private, satisfactory and alternative option to traditional medical consultation with eight consulting rooms and 38 visiting specialist doctors.  **Author**: K.P.I. Shenesh Perera  **Date**: 15/06/2019  IDM-Negombo | | | | |  |
|  |  | | |  | | |

Contents

[Research on existing online patient appointment systems 3](#_Toc12471358)

[Wintan Hospital: Project Plan (Project Schedule & Gantt Chart) 5](#_Toc12471359)

[User & System requirements 5](#_Toc12471360)

[Medium fidelity Wireframing for UIs 7](#_Toc12471361)

[Homepage 7](#_Toc12471362)

[Register 8](#_Toc12471363)

[Login 8](#_Toc12471364)

[Making Appointments 9](#_Toc12471365)

[Tech Stack 9](#_Toc12471366)

[Choice of design pattern: MVC 10](#_Toc12471367)

[What is the MVC design pattern? 10](#_Toc12471368)

[Why MVC? 11](#_Toc12471369)

[User Documentation 11](#_Toc12471370)

[Homepage 11](#_Toc12471371)

[Compare, contrast and evaluation of design to development 12](#_Toc12471372)

[Register 12](#_Toc12471373)

[Compare, contrast and evaluation of design to development 12](#_Toc12471374)

[Login 13](#_Toc12471375)

[Compare, contrast and evaluation of design to development 13](#_Toc12471376)

[Profile 14](#_Toc12471377)

[Compare, contrast and evaluation of design to development 14](#_Toc12471378)

# Research on existing online patient appointment systems

Choice of online patient appointment system: **Nawaloka Hospital PLC**

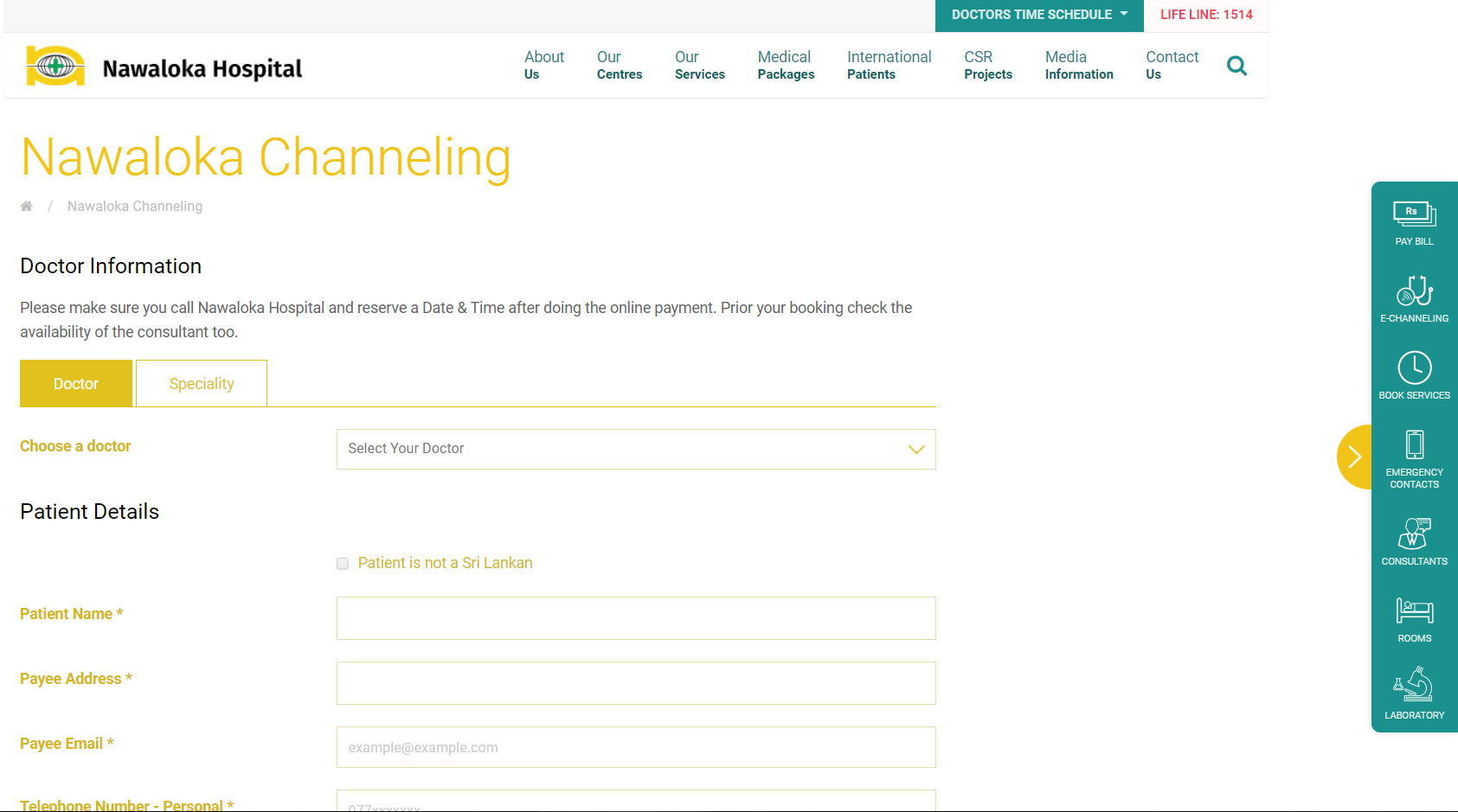


Figure 1.0, nawaloka.com, 03/06/2019, Shenesh Perera

The reason I chose Nawaloka Hospital’s online patient appointment system is because it is extremely similar to the same basic requirements Wintan Hospital requires. Nawaloka Hospital is known to be one the pioneers in the hospital industry, and as such the conclusions I draw from this research will be quite valid.

Figure 1.0, shows the patient appointment system Nawaloka hospital has. From first glance it becomes quite obvious that there is no registration or user access control setup. The form it presents asks some pretty sensitive information like for example the Payee Address.

The problem with this website is that it does not have TLS/SSL certification, in otherwords it is not a HTTPS based website. How does this become a problem? This patient appointment form alone requests for Payee address, having HTTPS on your website means your internet traffic is encrypted. When the form is submitted, usually form data is sent to the backend through an HTTP POST request. Although HTTP POST requests are known to be quite secure, in terms of encapsulating the data packets that are sent, but in no way is there any data obsfucation or encryption done.

This means that if a hacker were to sit in the middle of the client and the webserver and listen to the endpoint the form data is sent, then the hacker will be capable of stealing the form data without being caught.

In terms of design complexity and functionality, the patient appointment system of Nawaloka Hospital seems to be quite adept. The colors used aren’t difficult on the eyes, there is proper contrast between font and the colors used which is important for a website that attracts patients that may have sight problems. It also seems to be a mobile-first design, as such it is responsive across screens.

Conclusions are:

* Nawaloka Hospital’s online patient appointment system does not care for security or data protection.
* Design and color contrasts have been taken into count.
* Responsive web design has been done.
* From the looks of it, Nawaloka Hospital uses Apache HTTP server, with PHP as the server side language, and HTML/CSS/Javascript for client side design and development.
* Extremely slow, very high load time.
* Cluttered and disturbing animations have been used.
* Search Engine Optimization is quite adept.

# Wintan Hospital: Project Plan (Project Schedule & Gantt Chart)

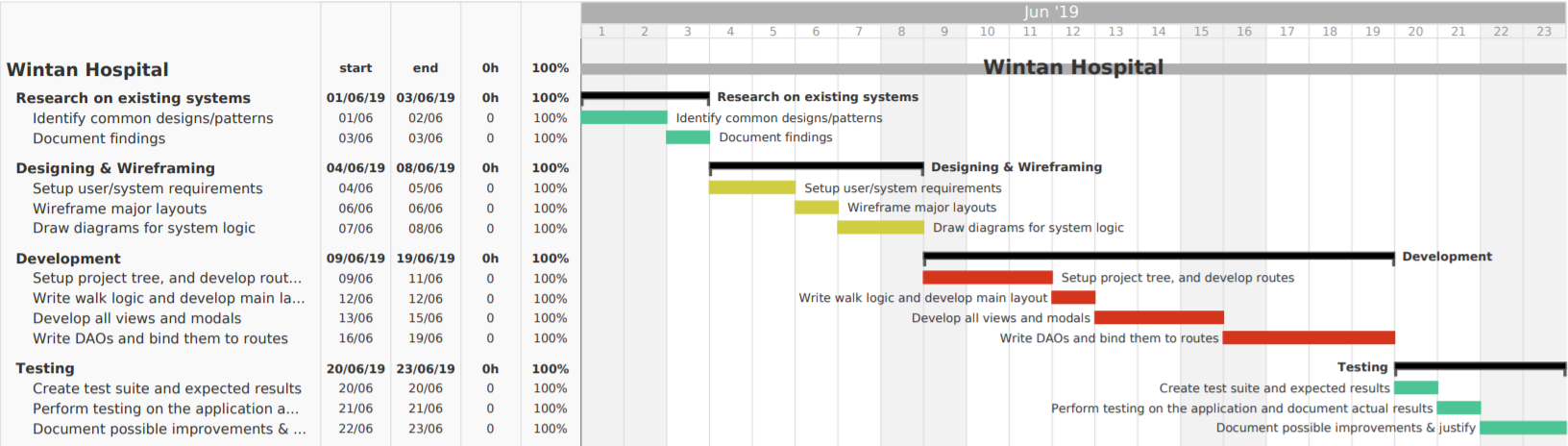


Figure 1.1, 01/06/2019, Shenesh Perera

**Note**: If the project plan is difficult to read, the image’s name is “project\_plan.jpg”.

**Summary of the project plan:**

|  |  |
| --- | --- |
| Identify Common Designs/Patterns | 2 days |
| Document Findings | 1 day |
| Setup user/system requirements | 2 days |
| Wireframe user layouts | 1 day |
| Draw diagrams for system logic | 2 days |
| Setup project tree & develop routes | 3 days |
| Write walk logic and develop main layout | 1 day |
| Develop all views and modals | 3 day |
| Write DAOs and bind them to routes | 4 days |
| Create test suite and expected results | 1 day |
| Perform testing on application and document results | 1 day |
| Document possible improvements & justifications | 2 days |

# User & System requirements

Through the research that was done on Nawaloka Hospital and the assignment scenario, conclusions and assumptions were made, as such the following are the user and system requirements. Do note, however that these requirements may or may not be fulfilled but the best attempt will be made.

User requirements:

* Patients should be capable of appointing a doctor
* Patients should be able to look at doctor schedules
* Patients should be able to see the appointments they’ve made
* Should not be complex or difficult to get started with
* Colors used must not be hard on the eyes
* The text colors must contrast the background colors

System requirements:

* Atleast 6 interlinked webpages
* A database to store patient information
* Frontend that will be used to collect data
* A backend that will process form data sent from the frontend
* Security must be ensured during form data processing
* Mobile-first/responsive frontend UI/UX design

# Medium fidelity Wireframing for UIs

Wireframing has been done in order to design the layouts before getting into development to satisfy the user and system requirements mentioned above. All user interfaces that collect data have been wireframed, **however during the actual development phase, there might be more user interfaces that may or may not collect data.**

In certain UI wireframes, the implicit form details were not included as I justify it as being a decision that should be made during development.

## Homepage

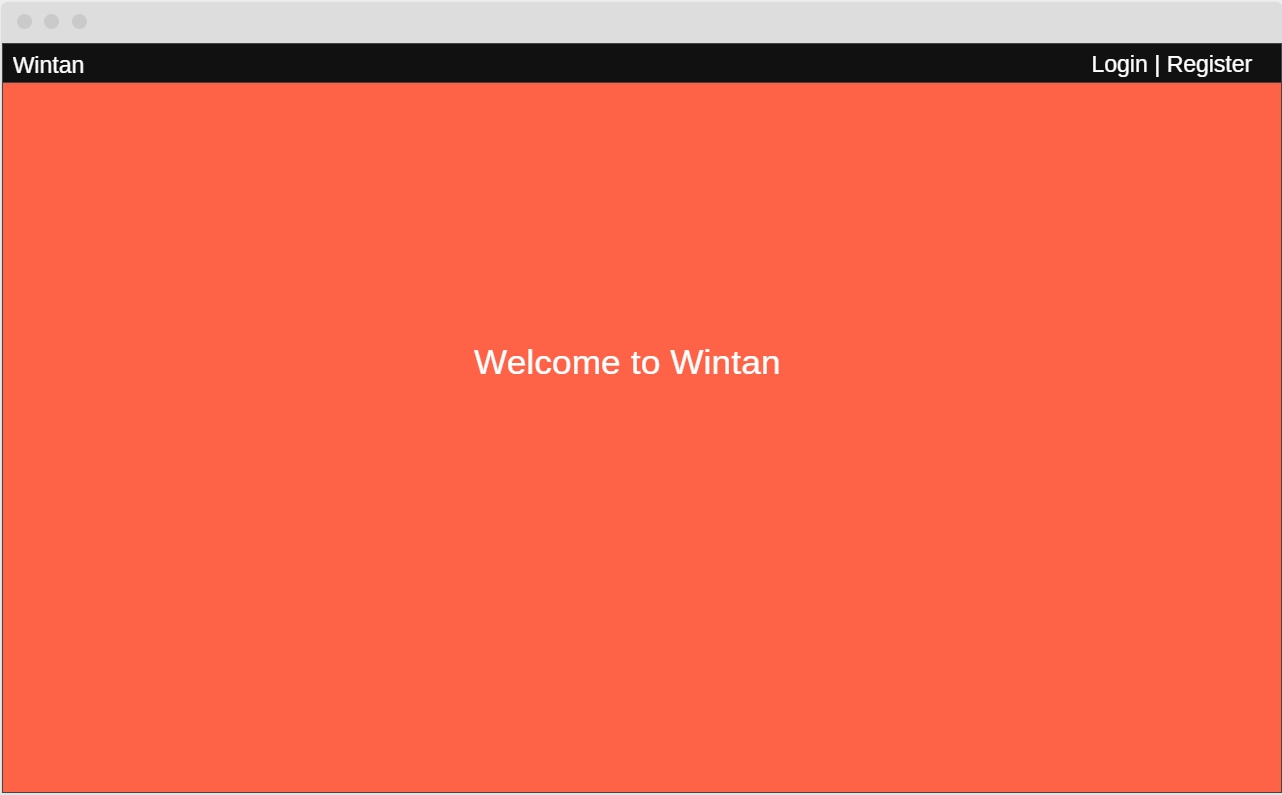


Figure 1.2, 06/06/2019

Figure 1.2, is the wireframe for the homepage. It includes the navigation bar on the top with only login and register buttons, a brand name will be set to the right side of the navigation bar. The orange box represents an image that will be relevant to this website, and there’ll be a center welcoming text.

## Register

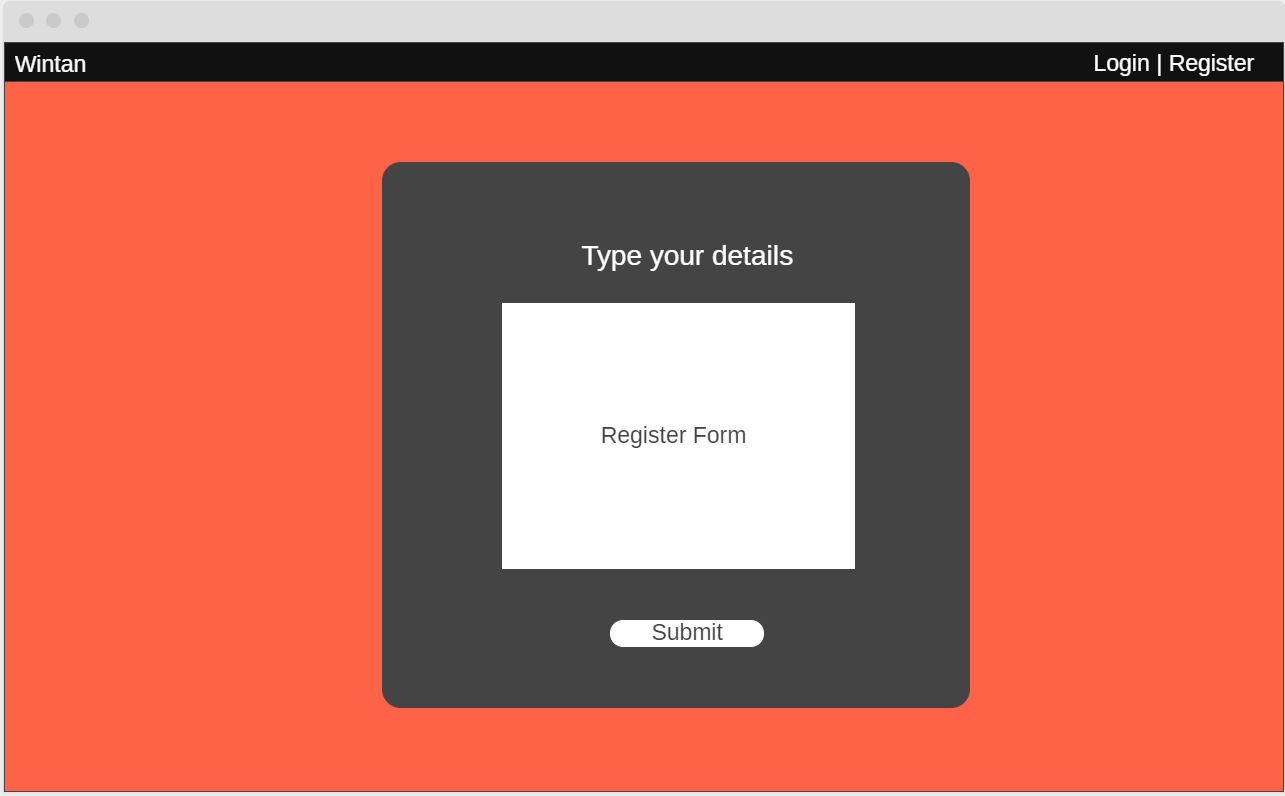


Figure 1.3, 06/06/2019

This will be the register form that contains the first data collection form, that will create an account for the patient. The actual contents of the register form will be decided in the implementation stage. However, there will be 3 distinct components in the register form. The header text, the form and the submit button.

## Login

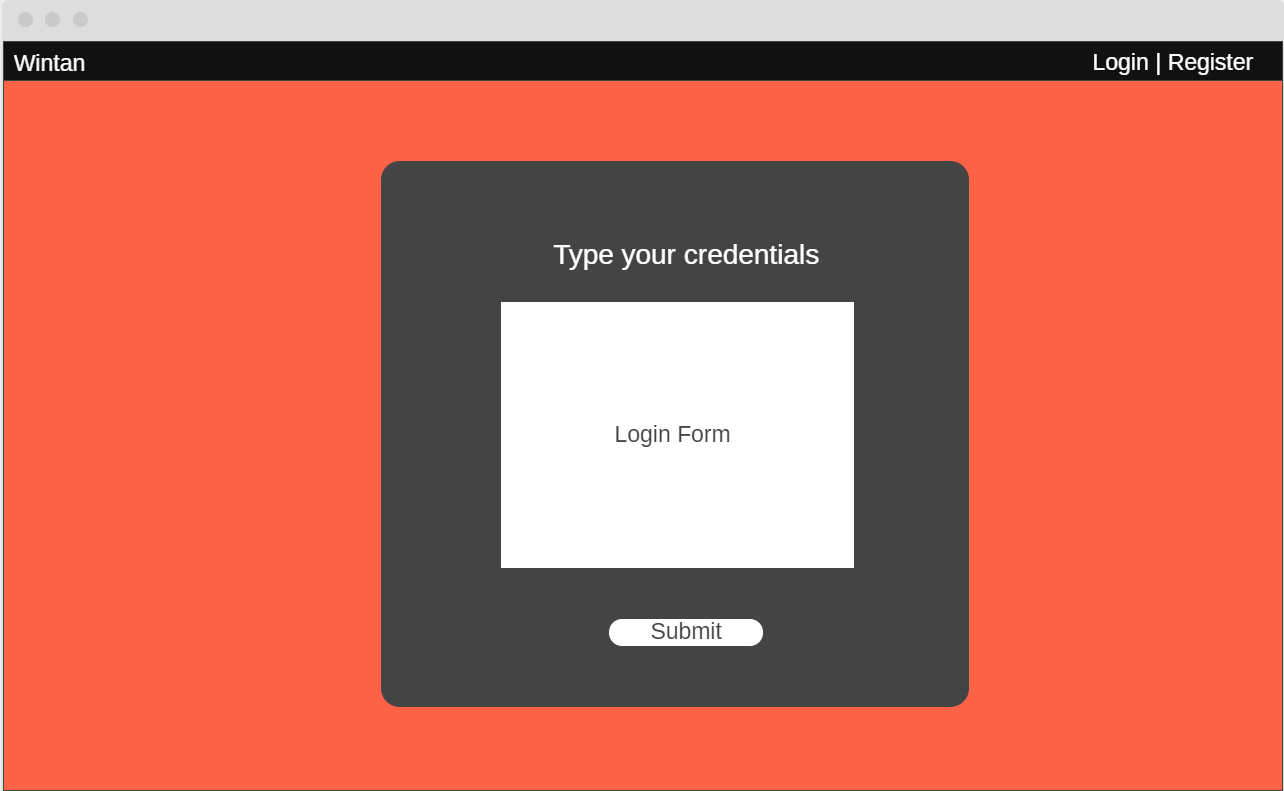


Figure 1.4, 06/06/2019

This will be the login form, similar in design to the register form. The 3 main components are also the same. However, the functionality of the login page will be considerably different.

## Making Appointments

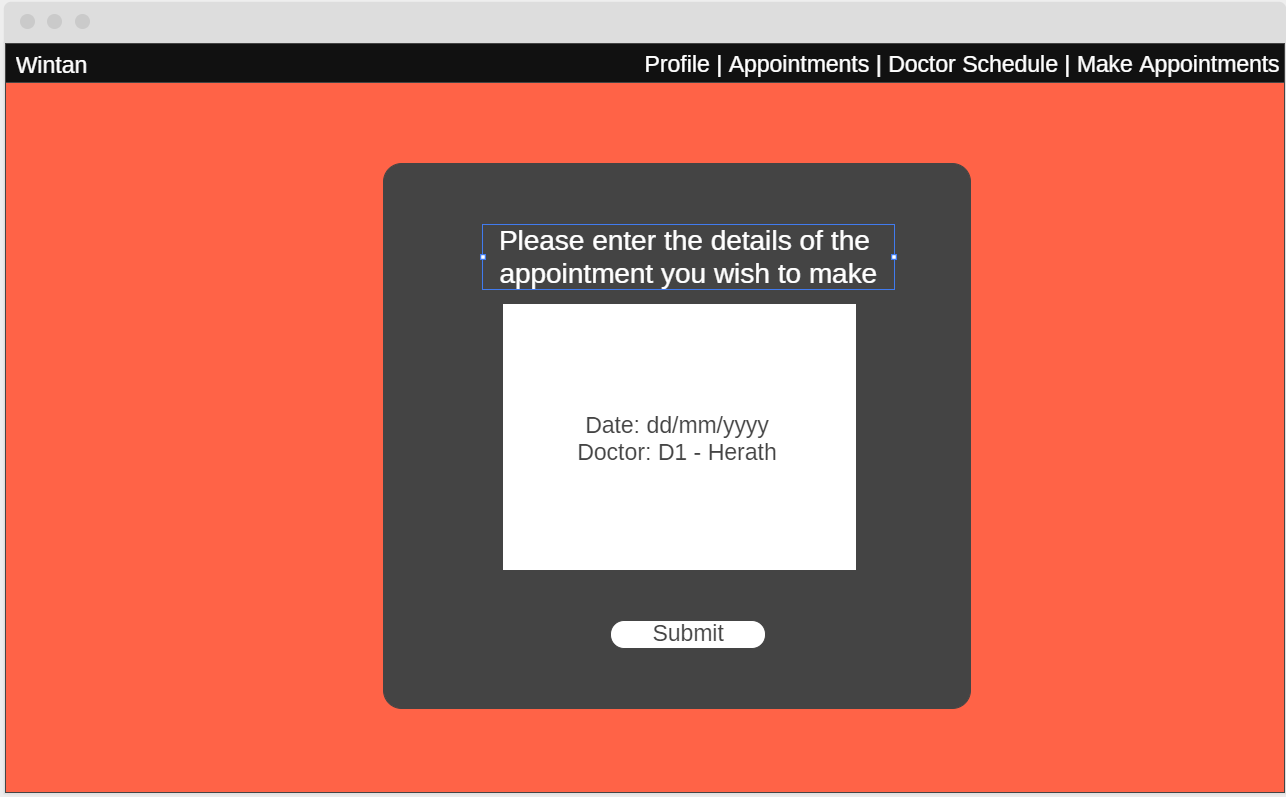


Figure 1.5, 06/06/2019

This will be the core of the patient appointment system. This page will only be visible once a user has registered and then logged in, the navigation bar will have updated showing 4 distinct components now, Profile, Appointments, Doctor Schedule and Make Appointments. There will be 3 main components in this form, the header text, the form itself and the submit button. I decided that there is no need to wait till the implementation phase in order to determine the contents of the form, as such there will be 2 fields. A date picker and a drop-box that will have all the doctors available for appointing.

That will be the end of the of all the wireframing. There’ll be additional forms and pages added in the actual implementation phase, but the contents and the structure of these pages cannot be decided before implementation.

# Why such a simple design?

The reason is because most people in Sri Lanka are technologically challenged. The target audience of this web application are patients, and necessarily these patients are usually above the age of 30. Most patients as such would not be particularly familiar with online patient appointment systems, and it is not rational to create something complex and difficult to use because that would only result in patients not using the Wintan Hospital’s online patient appointment system.

Having complex designs will push our target audience away from the online patient appointment system and simply rely on the legacy/traditional patient appointment system that relies on papers and calls in order to get an appointment.

The Nawaloka Hospital patient appointment system, despite being one of the most famous private hospitals in the Negombo area is barely ever used. The estimated traffic according to a simple traffic sniffing test was somewhere around 1 to 5 per hour, which is very low and is not efficient.

The simple and easy to use design will attract people to using the patient appointment system. The best and only way to improve the user experience is to provide the majority of the target audience that has a significant boost in user-friendliness than the traditional call/paper based appointment system.

Which is why, the simple design, easy appealing colors and less content are the core of this web application.

# Tech Stack

As this is a full-stack web application, the stack I choose will be split into frontend and backend this is the stack I will be using and the justification as to why I used said technologies.

Frontend Stack:

* HTML – For basic semantic markup and layouts
* CSS – For styling the layouts and semantic markup
* Javascript – To control the logic and the interactions of the layouts
* Bootstrap – To create a responsive navigation bar
* JQuery – As Bootstrap requires it

Backend stack:

* NodeJS – To handle static content and form data in an event-driven manner
* ExpressJS – For better and more performant server architecture
* PHP – To handle dynamic content of the web application in a synchronous manner
* SQLite – A web-enabled RDBMS that does not require a server, to store data

# Choice of design pattern: MVC

## What is the MVC design pattern?

Figure 1.6, 07/06/2019

The model view controller is a software pattern that is used for web development services such as applications; irrespective of the language. It is one of the most widely used paradigms in web development. The MVC design pattern is an application of the separation of concerns principle, which dictates that components that have different purpose or logic must be separated and isolated. As such, the web application is split into 3 main components. The Modals, Views and Controllers.

**Model**

The model is the main component within the pattern and it manages the data, logic and rules of the application. The model structures and stores the data within the database and it is subject to the controller’s commands.

**View**

This is the output representation of the data in the form of a screen and / or user interface (UX). You can imagine this to be the DOM or document object model that you use within an application to prescribe behavior to the view.

**Controller**

This accepts the input and sends commands to send to the model, the model then outputs the view to the user of the application.

## Why MVC?

It is important to understand that the MVC design pattern allows the components to be loosely coupled within the same environment. This allows you to create separate, re-usable components that can be easily modified.

This means that the time taken to develop applications is faster and more efficient in the long run. Especially when you consider the fact that if your customer needs to make a change to the MVC pattern. It wouldn’t take long to make those amendments or to change the models etc.

The model view controller concept is one of the basic tenets of computer programming and this is a very useful paradigm to the world of tech and as such this is a very suitable design pattern for a web application like this online patient appointment system as it helps to reinforce the logic conceptually of how loosely coupled components communicate.

# User Documentation

The user documentation will describe, explain and provide evidence for certain implications and the satisfaction of user and system requirements. Each page that exists within the web application will be promptly explained, every component available will be described as well.

**Critical criticism, justification and comparisons will be made to the design documentation, so please read every word of each section.**

## Homepage



Figure 1.7, 09/06/2019

The home page is the first page that will be seen when you visit the website. It welcomes every patient with a greeting and a subtext that ensures that Wintan Hospital is the hospital for their medical needs.

### Compare, contrast and evaluation of design to development

The wireframe design of the homepage is almost identical to the implemented homepage, there is an image in the background and a greeting text. There is also a navigation bar that holds the Login and Register buttons with the brand of the hospital on the left hand side corner of the navigation bar.

The colors that were used however are different. The navigation bar has a grey color and the text color is white. The buttons have green text and a green outline. Upon hovering the buttons, you will notice that the text will receive a green background and the text color will change to white.

The register button will lead to the register webpage and the login button will lead to the login webpage.

Register

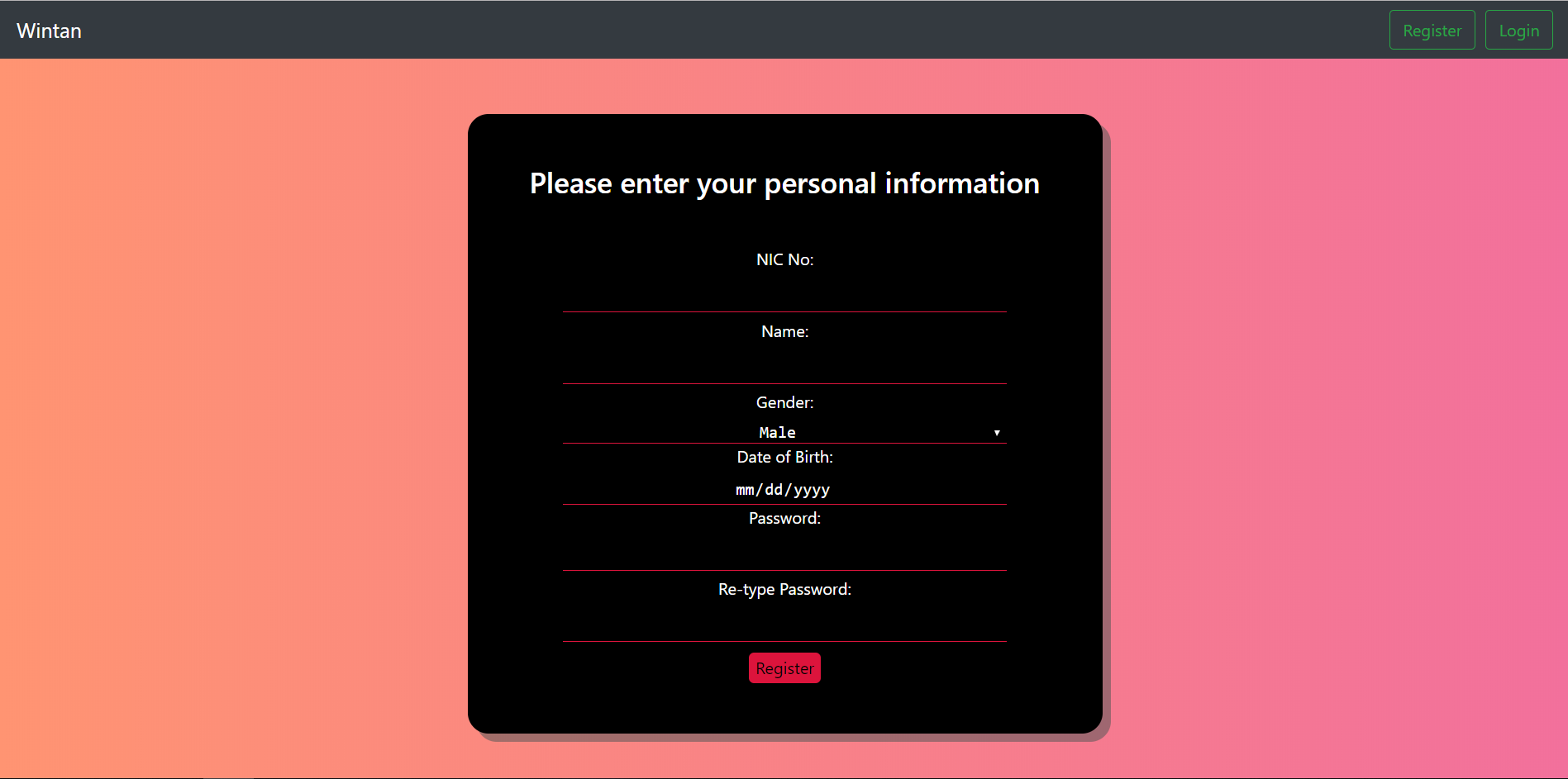


Figure 1.8, 09/06/2019

The register page is obtained when you click the register button on the navigation bar, it displays a form that will register a patient.

### Compare, contrast and evaluation of design to development

The wireframe did not have any content on the form that will be displayed in the register page, and there were 2 boxes, the white one inside the black one, but I decided to use a pure black single box but with the 3 main components. The header text, the form and the submit button (renamed to register).

The register form requests information like the NIC no., gender, name of the patient, these are the form data that will be sent to the backend through an HTTP POST request.

## Login

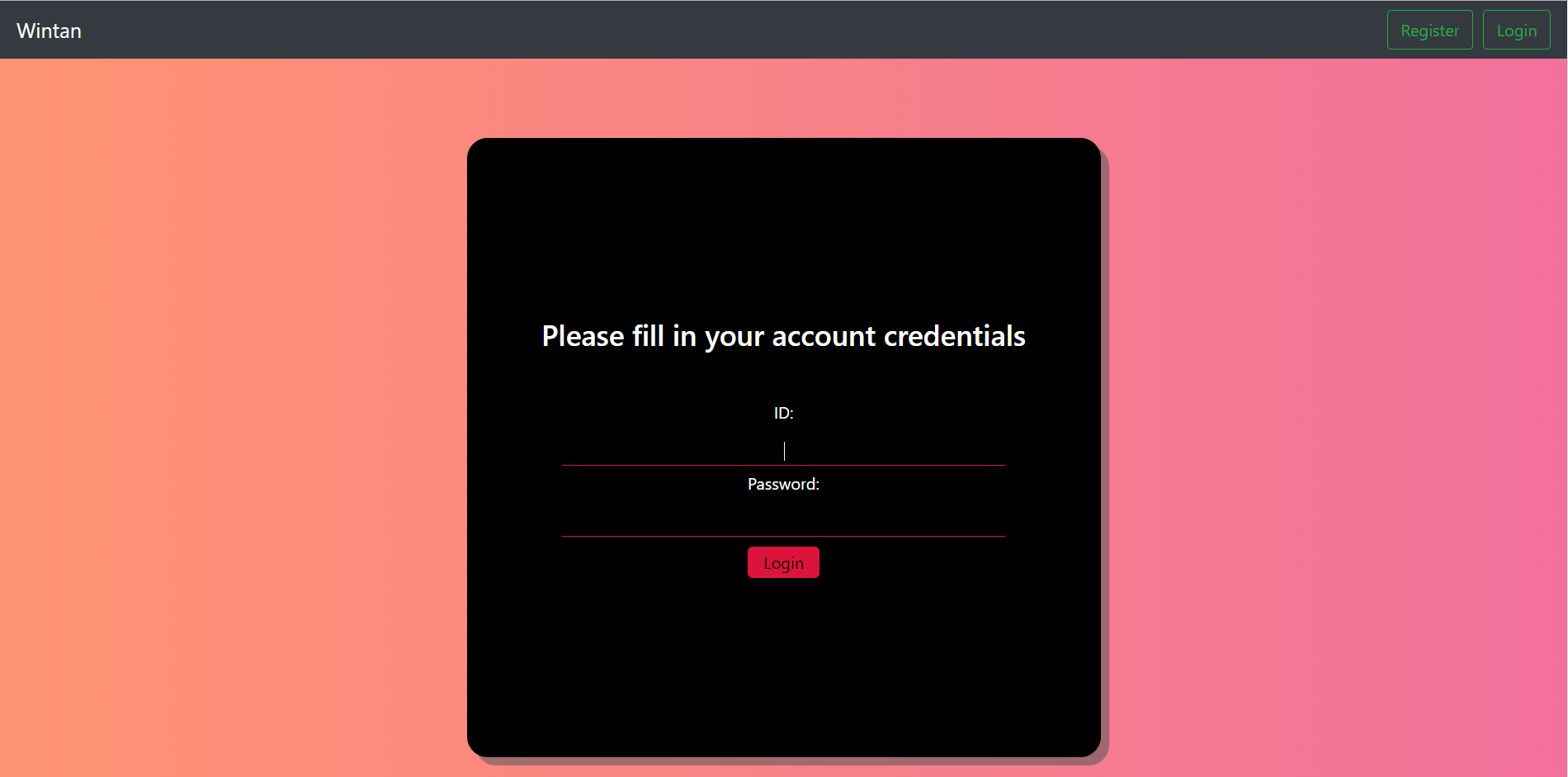


Figure 1.9, 09/09/2019

The login page is obtained when you click on the login button on the navigation bar. It too just like the register page, displays a form that has details to be filled with.

### Compare, contrast and evaluation of design to development

The login wireframe too did not have any content on the form, but in the implementation content was added. The structure of the login form is different, the 3 components, the header, the form and the submit button (renamed as login) exist however. After a user has registered, they may use their NIC No. as the ID and use the password they registered with.

However, though upon registration, till you logout you will automatically be logged in.

## Profile

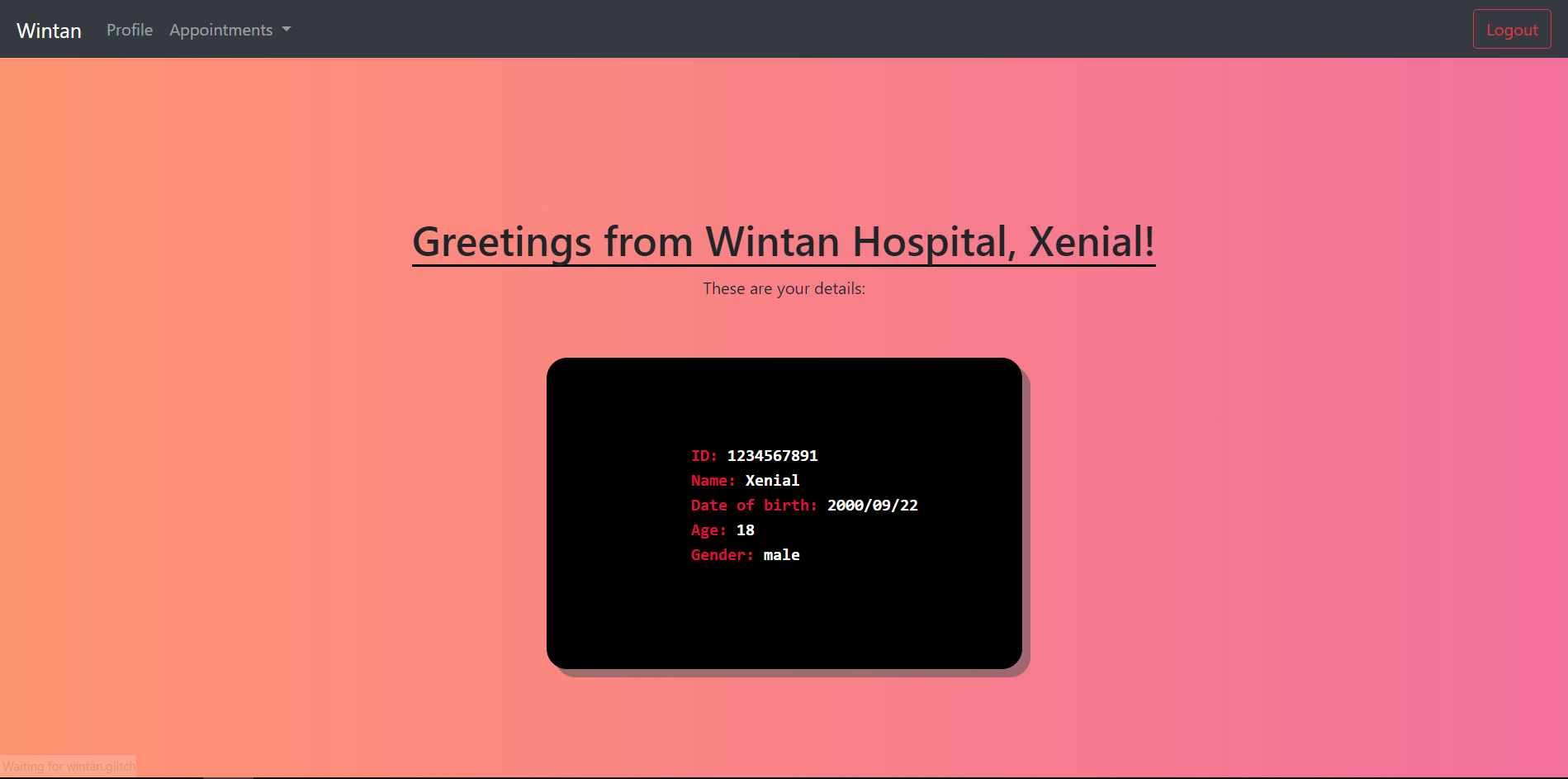


Figure 2.0, 09/09/2019

The profile page displays all the information of the logged in patient. The patient may only goto their profile after they have logged in, which means they must have registered in the wintan patient appointment system first.

### Compare, contrast and evaluation of design to development

The profile page was not wireframed during the design stage as this was added as a utility page in order to boost user-friendliness and the overall user experience(UX) of the patient so that they can look at what data they have registered with. All the registered data except the password is listed here, in addition to the data that has been entered, the age field will be calculated at the backend and shown in the profile page.

## Appoint a doctor-page

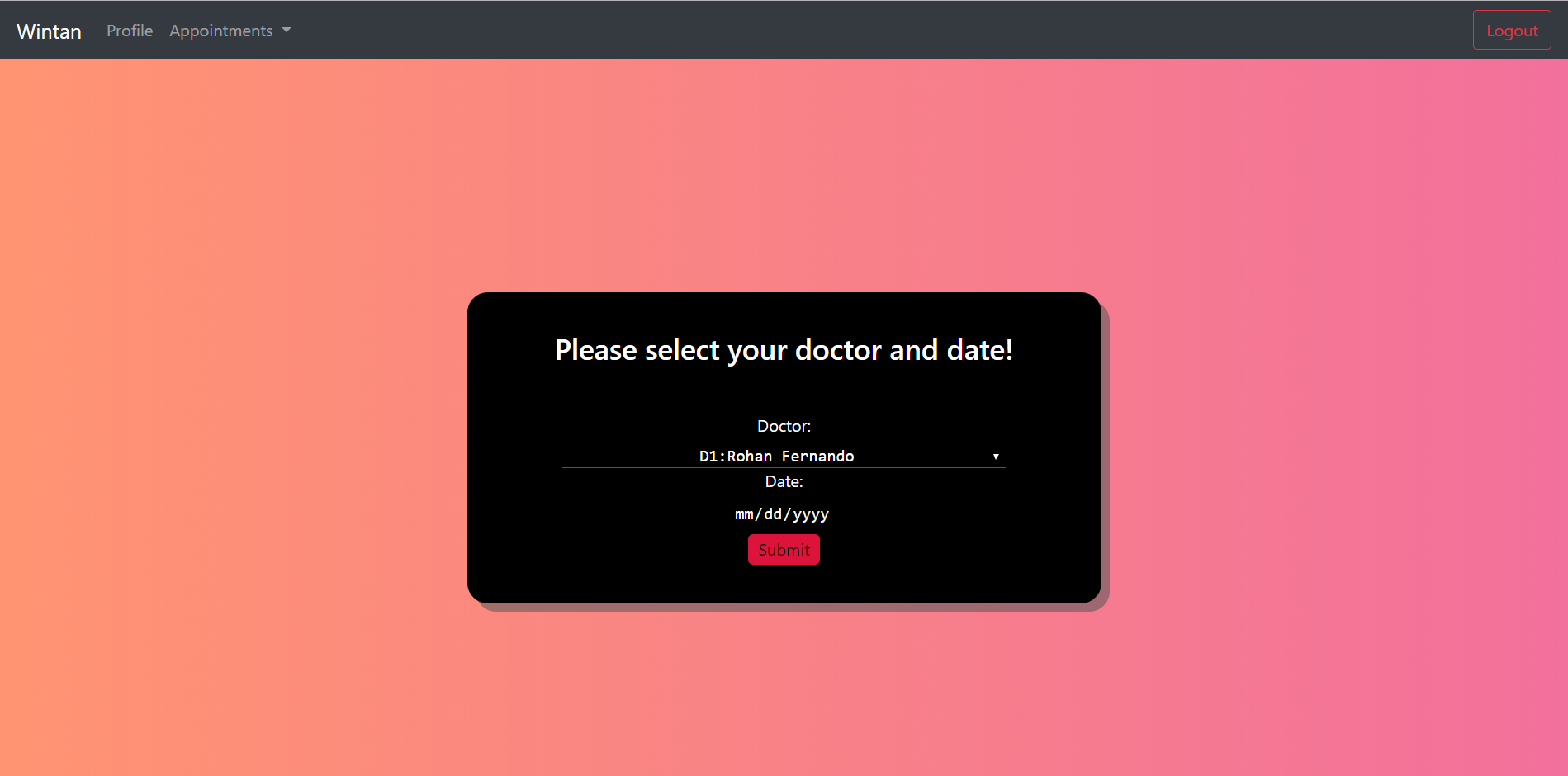


Figure 2.1, 09/06/2019

The heart of the patient appointment system is the Appoint a Doctor page, this boasts a simple to use and less effort-based web application.

### Compare, contrast and evaluation of design to development

The wireframe of the Appoint a doctor-page is entirely identical to the developed page, the only difference is that there are no 2 colored boxes. The 3 main components, the header text, the form and the submit button are all in place.

The dropdown in this form dynamically loads the doctors from the database that stores doctors information, the ID and the doctor name are combined here for processing easiness in other stages, so for example a doctor with an ID of 123 and name of John Doe will appear as “123:John Doe”.

## Your appointments-page

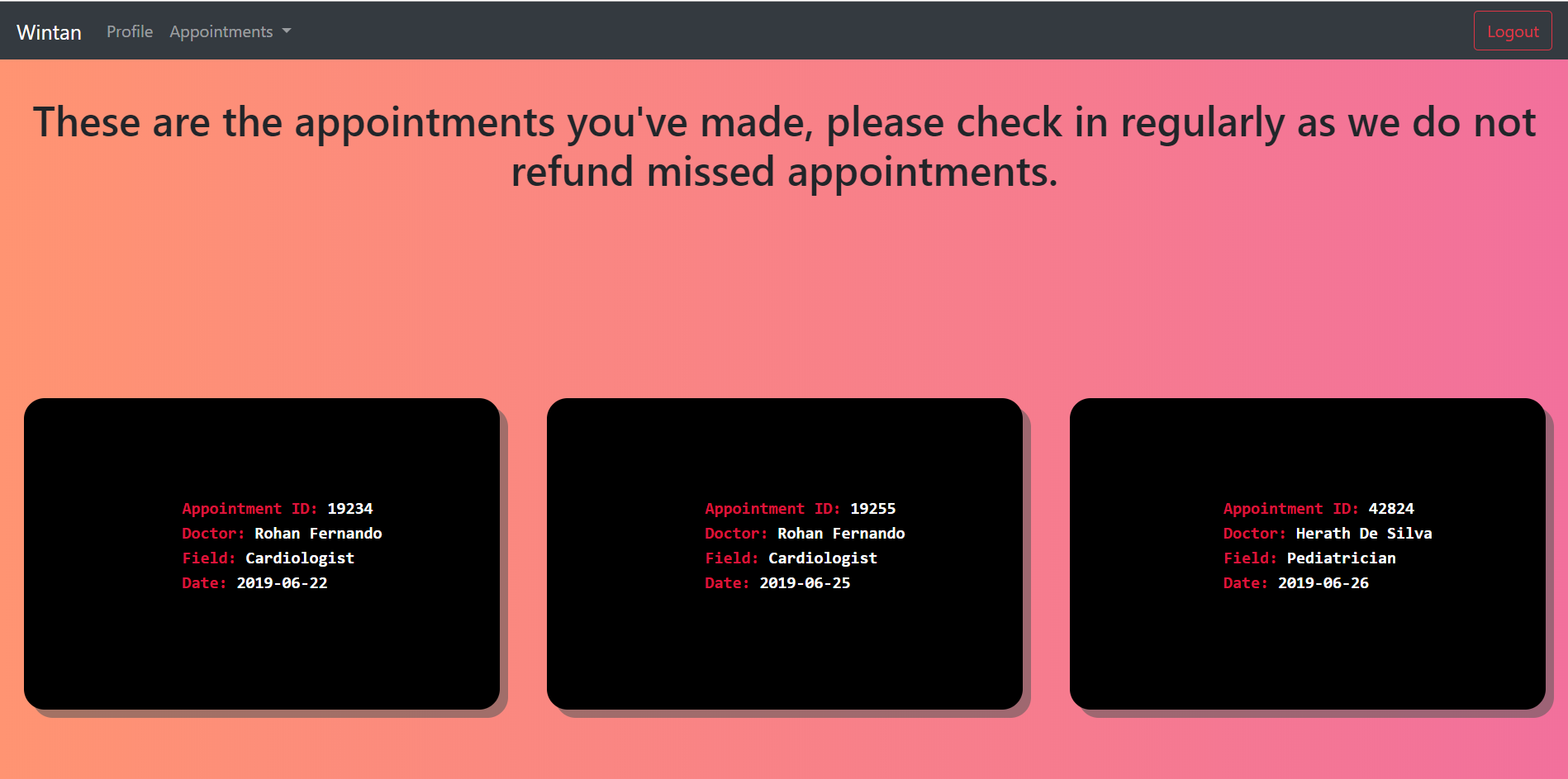


Figure 2.2, 09/09/2019

This page will show all the appointments that have been made, declaring verbosely the details of each appointment.

### Compare, contrast and evaluation of design to development

This page was not wireframed during the design stage and was added in the development stage in order to improve user experience and ease of use. It dynamically loads all the appointments made of a patient from the database. The Appointment IDs are dynamically created during the phase of creating an appointment, the doctor’s information are taken from the database as well.

## Doctors Schedule-page

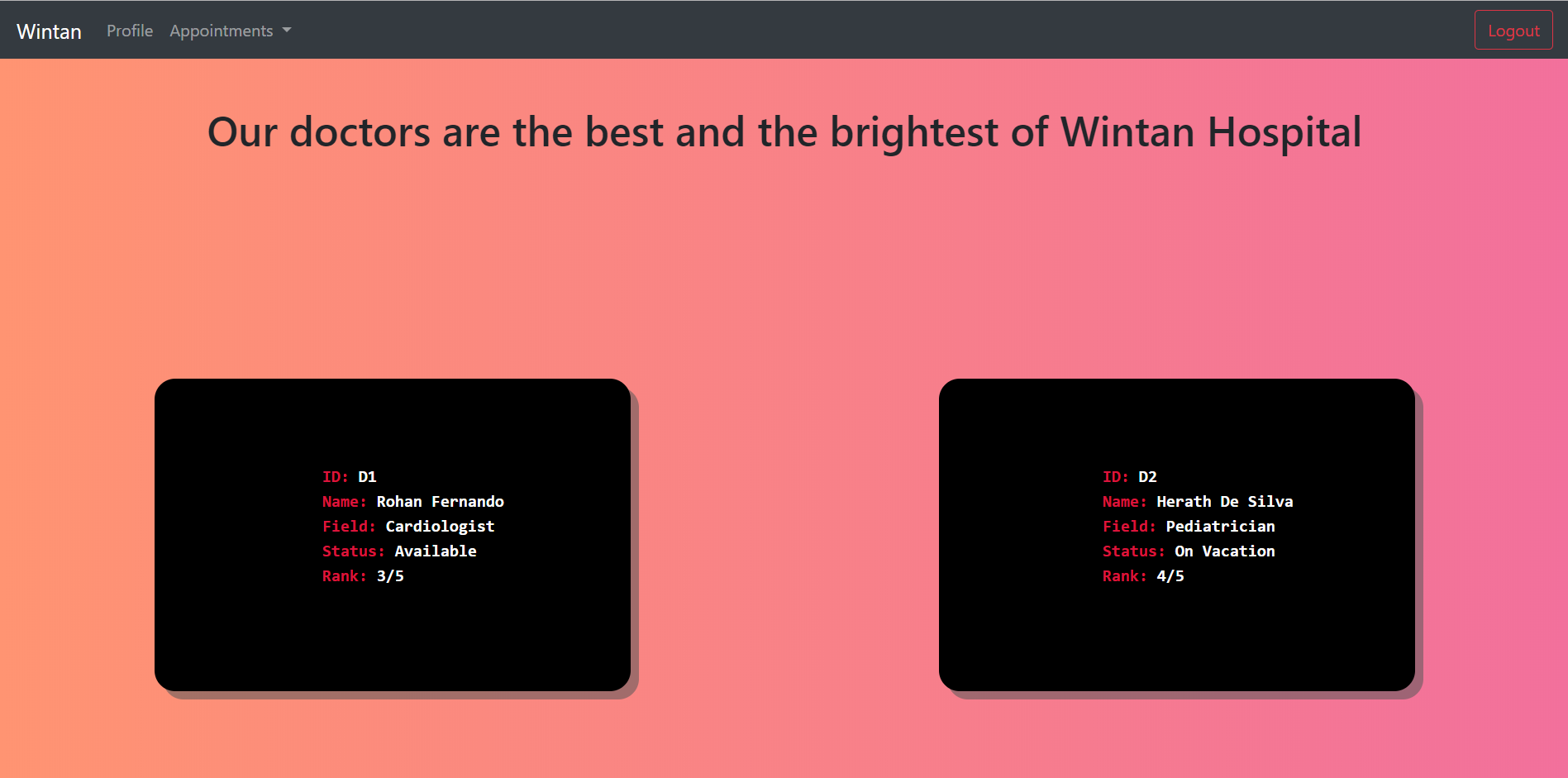


Figure 2.3, 09/09/2019

This page will be accessible from the Appointments dropdown and is the last of the dropdown, it shows all the doctors available for appointment in the wintan hospital.

### Compare, contrast and evaluation of design to development

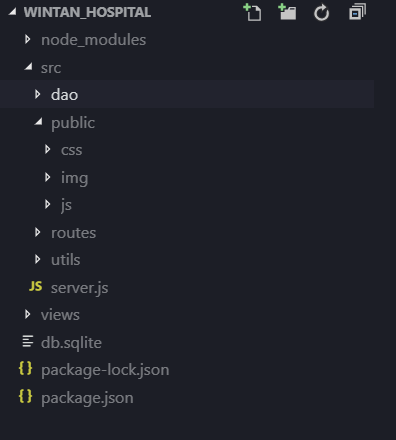
This page was not wireframed during the design stage and was added in order to boost user experience and ease of use. All the data that is displayed is dynamically loaded from the database, adding/editing/updating/deleting doctor information is not possible as it is not a user nor system requirement. All the values are loaded directly from the database and as such can only be added to the database manually or through the doctors DAO(data access object).

# Evidence for LO3 and LO4

## The project-tree overview

The DAOs are the modals, the views are the views and the routes are the controllers within the web application. The public folder contains all the static content like the css, images, and the javascript used. The db.sqlite is the database file that contains all the data of the web application.

The utils folder contains some utility files that aren’t directly used for the project. The server.js is the main file that runs the ExpressJS server that is written in NodeJS.



## Example of a view file (using PHP, HTML and Javascript)

The following image is one of the many view components written using PHP, HTML and Javascript within the web application. The following is the Login-Form’s code. The Javascript is used for form validation, to check if the fields are empty and then submit the form data.

## *Example of a Data Access Object file that acts like a modal(written in ES6 classes of Javascript, that uses NodeJS promise async/await features)*

DAOs are ES6 classes that opens the connection to the database and then provides functions that allow to manipulate the database. DAOs are used to encapsulate the logic required to insert data to the database. Promise based asynchronous coding has been used in order to boost the performance of the application. The DAOs use advanced javascript concepts like function composition, currying etc.

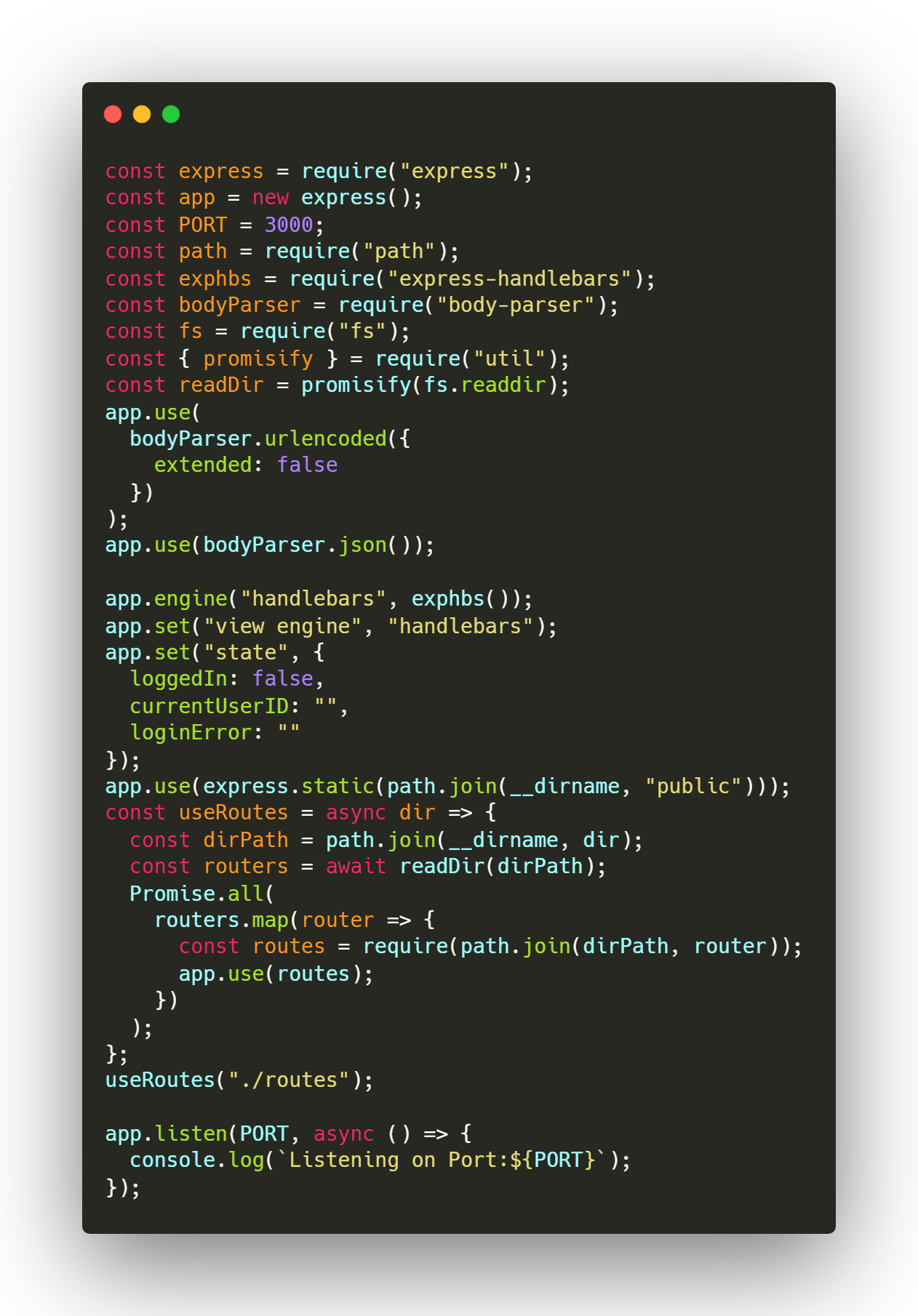
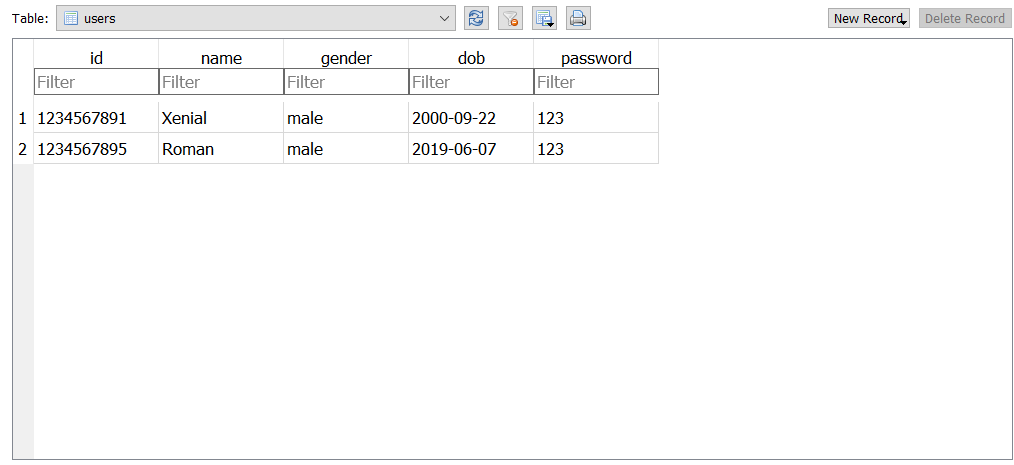
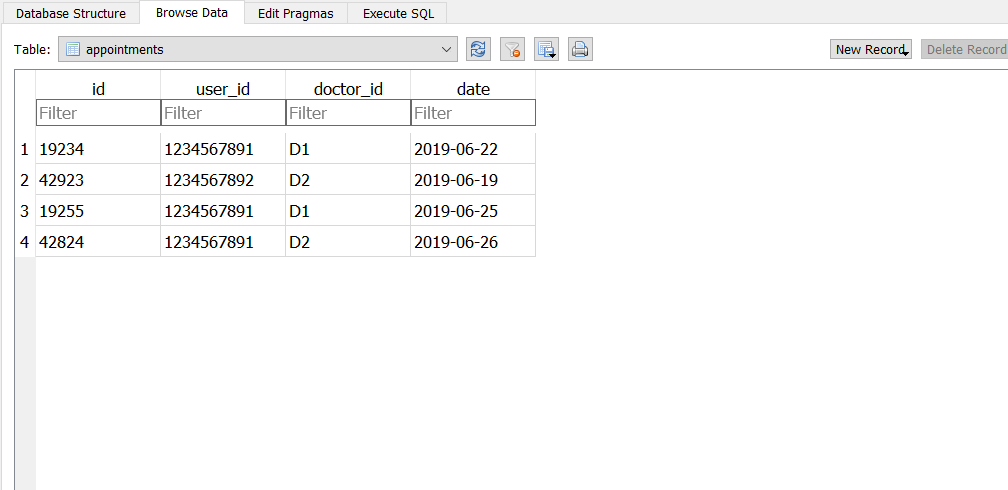
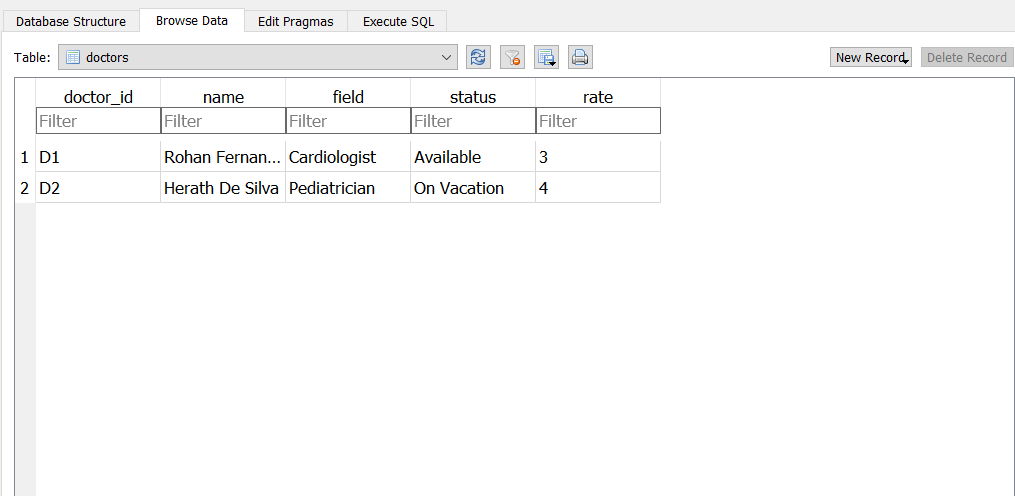
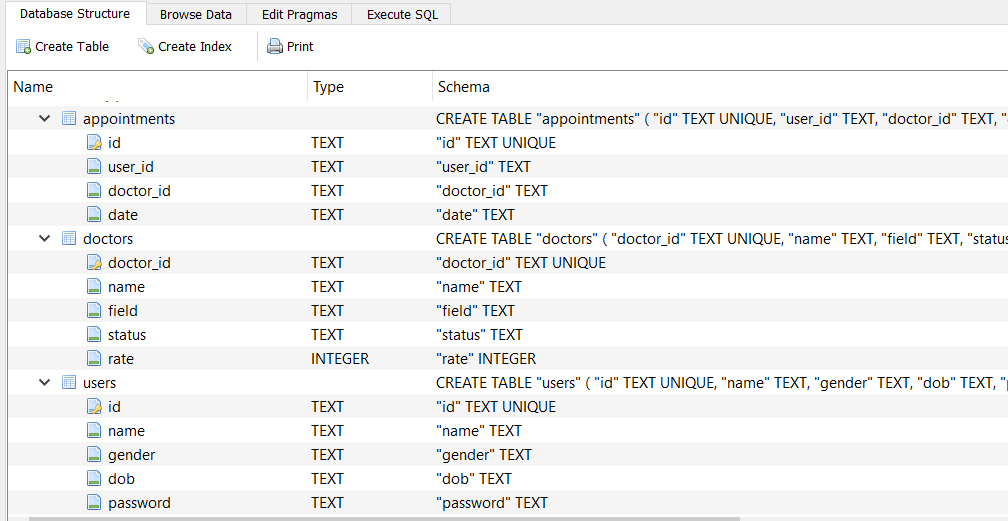


## Example of a router which acts as controller in the web application (uses ExpressJS router module, Javascript and NodeJS promise async/await features)

The routers handle the PHP templates and are mounted to the application each time a page is requested. All the controllers are modularized and completely isolated from one another so each router does not conflict with another. The following router is for the “/profile” route, it uses the ExpressJS Router function in order to achieve the modularization feature. This too like most other components of the web application uses asynchronous code in order to boost the performance of the application.



## Other required evidences



# Technical Difficulties faced during the development

1. Managing the state of the web application, in order to identify which user is logged in
2. Performance considerations, so that the UI does not look bloated or does not take too long to load
3. Mobile-first design, due to using cards it was difficult to make them look good on mobile screens, however it was achieved using CSS3 media queries and CSS3 flexboxes.
4. Overall memory efficiency, Javascript is known to be sloppy when it comes to maintaining the memory, there were a few memory leaks that were created during the development.
5. Troubles with walking the routes directory to mount the each router controller to the web application.
6. Enforcing the MVC design pattern, it is quite difficult at times to isolate each component of the web application separately.
7. Loading data from the database without making the frontend lag
8. Using PHP with ExpressJS for templating purposes
9. Creating a Navbar that is actually responsive

# Test Suite



Figure 2.4, 22/06/2019

Figure 2.4, shows the overview of the test suite that will be used to test the full stack application. During this test suite, the following things will be done:

* Chrome DevTools performance profiler will be used to analyze the performance of the web application.
* Chrome DevTools memory snapshot system will be used to analyze how much memory is being consumed and provide deeper analytics on where memory is being used.
* Expected results will be matched to the actual results.

|  |  |  |  |
| --- | --- | --- | --- |
| Test Name | Expected | Actual | Comments |
| Register a patient | Patient is added to the database | No response | DAO has not been properly mounted |
| Register a patient x2 | Patient is added to the database | Patient is added to the database | The test has been passed |
| Login as a patient | Navigation bar must be updated | Navgiation bar is updated | The test has been passed |
| Add an appointment | Your appointments page should show it | No response | Error saying mismatch of schema of the appointments table and data sent |
| Add an appointment x2 | Your appointments page should show it | No response | Application stuck in an infinite loop due to sending empty field values |
| Add an appointment x3 | Your appointments page should show it | Appointments page shows it | The test has been passed |
| Logout | User should be redirected to home page | User gets redirected to the home page | The test has been passed |

## Performance Test

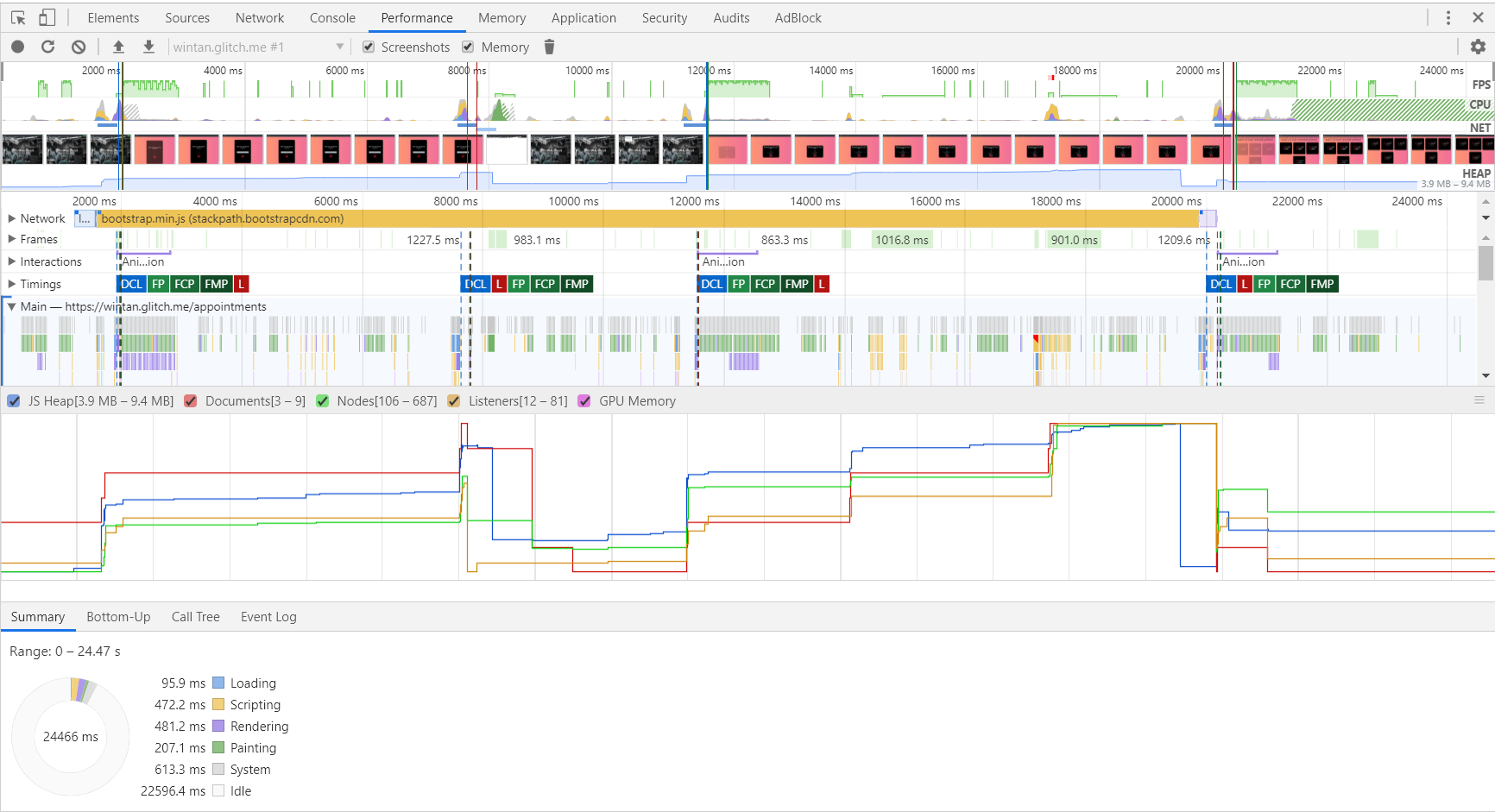


Figure 2.5, 22/05/2019

The performance test shows that 24466 miliseconds were taken for the entire test suite demonstrated in Figure 2.4 to be performed, which involved:

registering a patient -> logging in -> making an appointment -> viewing it -> logging out.

24466 miliseconds translates to 24.5 seconds approximately, for 5 actions performed it is fair to say that the wintan hospital appointment system is very fast.

As you see, most of the time is spent idle which means the time I spent on inputting data which accounts for roughly 22.6 seconds. As such during the whole test suite, the time it took for loading, scripting, rendering and painting approximately took only 1.9 seconds!

This is blazing fast for a full stack application and I’d say that this project is a complete success, in terms of performance

## Memory Test



Figure 2.6, 22/09/2019

There were 3 snapshot rounds that were taken as the V8 Engine that NodeJS uses for handling server side logic uses what is known as a Just-In-Time compiler, as such on the 1st snapshot it requires 5.3 MB as all assets are loaded and then frequently used components are cached.

Which is why during the 2nd and 3rd snapshots, you can see only 3.8 MB is used. For a fullstack application that loads content dynamically, has static content, is coupled with a database, 3.8 MB is a very low amount of memory used.

As you see, over 50% of the memory is allocated for the Javascript’s internals and managing the DOM. So you can conclude that the Wintan Hospital online patient appointment system uses somewhere around 20% to 30% of the allocated memory which when calculated is some where around 1.16MB to 1.74 MB in the 1st snapshot and 0.76 MB to 1.14MB in the 2nd and 3rd snapshots.

Which means that even on a mobile that is atleast 4 or 5 years old, this web application will use extremely small amounts of memory, as such the application will be blazingly fast.

# Quality Assurance

There were 4 main features that allowed for maintaining and ensuring that quality assurance was a large factor that played in both the design stages and the development stages which are:

* **Stress/Load based testing** – The performance test and the memory snapshots were taken to ensure that the web application will run fast and smoothly on all devices across any browser.
* **Simple testing** – The test suite that checked things like registering an account, logging in and logging out were to ensure that all the features that cover the user requirements were functioning to the highest levels and there is no problem at all in each feature.
* **Usage of a mature design pattern like MVC** – This ensures that logic and purpose of each element/component within the application is isolated and encapsulated properly in such a way they do not conflict with each other and create unnecessary errors that will damage the user experience and reduce the quality of the application.
* **Wireframing during design stage –** This allowed me to have a rough idea of how the backend architecture would be structured and as such resulted in a neat project tree.
* **Using technologies that have survived the test of time –** Using technologies like bootstrap, PHP, NodeJS etc. that have long withstood the pressure of industrial demands and the challenges that each technology had to face through time made it very extensively suitable for this application, using them kept me from falling to common pitfalls that would’ve been met if I implemented certain things from scratch.
* **Using mobile**-**first design and development –** This ensures that across all platforms and devices, this application will be usable and would not look unresponsive or difficult to interact with.

# Overall success review and possible improvements

I would say that the Wintan Hospital Online Patient Appointment system came out extremely well and exceeded both my expectations and the expectations of my test suite.

The reasons I say the project was successful are:

* All user and system requirements were met ideally
* Performance of the web application is perfect
* Memory used is very less
* The codebase is very friendly due to the use of a design pattern like MVC
* Looks great on all devices from mobiles to tablets to laptops to 4K screens
* There are no notable errors or missing functionality that may annoy a user
* The application is very secure and handles security neatly
* The test suite came out saying that the application works perfectly

However, though there are some considerable improvements that can be made to the existing online appointments system, which are:

* Dropping PHP from the backend stack as PHP is obsolete, legacy and uses synchronous programming, which is quite slow
* Using a matured frontend framework that would better handle view components like ReactJS, as reactJS is the recommended Javascript frontend framework that emulates the functionality of View in MVC
* Use sessions instead of having a global application state which would allow for easier maintainenance of the state as a global state is susceptible to change from anywhere
* Using media queries to load smaller sizes of the image for smaller screens, which would improve load time
* Have Search Engine Optimization so the website is possible to be found from search engines
* Compress all images used so it would improve the load time of the web application
* Use a bundler like Webpack which would allow to create minified Javascript files which would improve load time
* Use Koa instead of ExpressJS as Koa is based on NodeJS asynchronous event driven nature more, and it has significantly higher performance
* Use a CSS preprocessor to reduce the number of styling rules in the stylesheet which would result in faster loading