**INFS3202 Group Project Report**

Harry Hoyland - 43545544

Cameron Weber - 43212789

Chris Wilkinson - 43942004

**Project Aims**

* To develop a website based on booking doctors appointments
* To further develop each group member’s understanding of website development and creation

**Project Outcomes**

* Our website : [www.shopadoc.me](http://www.shopadoc.me)
* We developed a cloud hosted website with the following features
  + Users can search for doctors
  + Users can login with their google accounts
  + Users can book doctors appointments and a confirmation email will be sent
  + Users can leave reviews about each doctor
  + Users can save their preferred doctors
* We all learnt many new techniques and approaches including
  + Cloud deployment
  + Responsive webpages
  + User interaction
  + Implementing a 3 their WIS architecture

**Design**

**Design Approach**

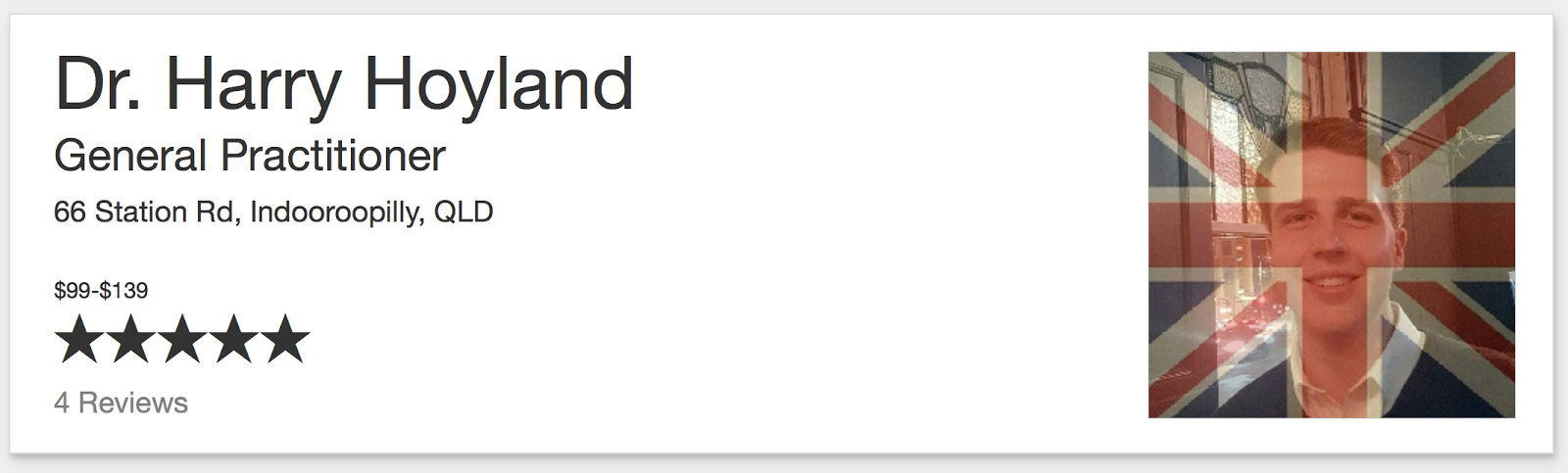
The design of the website was achieved using HTML and CSS technologies. Twitter’s Bootstrap 3 framework formed the basis for some elements, such as grid alignments in the index page, and the booking times in the listings page.

**Best Design Functionalities**

Functionalities I am most proud of is the responsive design of the website. As the user resizes their browser window, the layout of the page rearranges such that all functionality is retained. This required considerable work in CSS using ‘media’ tags. Manual definition of elements were required to account for each screen size using ‘@media’ tags.

|  |  |
| --- | --- |
| creen Shot 2017-05-30 at 8.38.09 am.png | creen Shot 2017-05-30 at 8.38.02 am.png |

I am also proud of the work done for each doctor listing. Each element has clarity in their purpose, and is usable in their function. Users can see vital information at a glance, without having to click on the listing to find out more.



**Problems Encountered in Design**

Problems were encountered along the way. This included accounting for changes from other group members in the design. At various points, elements would break because someone else changed functionality somewhere else in the code. This was solved by having regular group meetings and asking frequently what they were doing at that moment. This meant that I could help them with functionality, as well as prepare for any changes in design that might occur.

Another problem was the organisation of CSS elements. It was difficult to lay out style definitions in such a way that they were logical, and structured, especially as a single file. The solution was to organise the elements by page. Divide each page by a simplt comment, and place the most important elements near the top. This solution proved to be effective in keeping track of style elements.

**Server Side Functionalities - Cameron Weber**

**Key technologies used:**

* Database creation and management (Mongodb)
* Server creation and management (Node.js)
* Client, server and database communications (sockets)
* Cloud deployment (Amazon Web Services)

**Notable approaches and implementation**

Scalability:

* The server itself is behind a load balancer which redirects traffic from the web to a specific server instance
  + Allows for multiple servers behind a single domain name
  + Easy to scale up the number of servers required on demand
  + If one server is broken / unreachable the balancer will not use that server
  + Allows for servers from multiple regions around the world

Security:

* SSL and HTTPS are implemented
* Ip whitelisting so only the server can talk to the database and only developers can SSH into the server
* Communication between the server and client is filtered so only expected values and requests are acted upon
* The server then controls what information is sent and received from the database
* The server controls what html pages and content is visible to the public

Client independence, the requests from one client won’t stop another client from requesting from the server

* Used Sockets so multiple clients can talk to the one server asynchronously
* Node.js is an asynchronous system meaning it runs multiple threads and

Url processing for user simplicity

* Used url routing so users don’t see any‘.html’ file extensions

Emailing

* Sends an email to the client after they confirm a booking

**Harry**

**Key Technologies Used:**

* Node.js Client/Server Architecture
* Google Platform Library API
* google-auth-library node.js package
* Google OAuth 2.0
* jQuery

**Notable Approaches and Implementation:**

User Account / Login System:

* Utilising the Google Platform Library API to offload secure storing of user passwords.
* Rather than passing plain user IDs between the client and server which would allow modified clients to send arbitrary user IDs to the server to impersonate a client, verificable user ID tokens are used instead.
* User ID tokens are passed securely to the server which then verifies the token, obtains user information and confirms the login on the client side.
* For security reasons, requests to the server for verification and login purposes are only accepted from valid domains (ie. https://shopadoc.me)

Modularity:

* Due to the recurrent nature of the login and checks to verify if the user is logged in, these functions were placed inside a separate js file (util.js)
* Jquery’s .getScript function was used to load this script from within each page’s js file. Another jQuery function was also utilised to ensure a page’s js was only run once this util.js file had been executed.

jQuery:

* jQuery selectors were used to easily retrieve and filter through DOM elements on the page. jQuery was then used to register and respond to events such as click and keyDown.