# Slide 1

Hello my name is Gavin Hanna, this is my presentation on my final project application which is called “My Recipe Book”. A social network web application based around creating and sharing personal recipes, and curating your own personal recipe book, using you own and other users’ recipes.

# Slide 2

Some background on the application.

Recipe based websites are an extremely popular corner of the internet, with the top 15 recipe based websites ranging from 2 million unique monthly visitors for KraftRecipes.com, to 25 million visitors to AllRecipes.com.

The word “food” has an average monthly search count of 823,000, with the word “recipes” having 450,000 and “recipe” with 90,500 (**searchvolume.io, 2018**).

The popularity of recipe searches on the internet, and the overwhelming popularity of social networks (Facebook alone has over 2,196,000,000 users (**statista.com, 2018**)), lead naturally to the combination of the two, of which there are many such websites currently available online. Two of the most popular are bakespace.com and cucumbertown.com. While these are technically social networks, they either focus more on simply being a recipe search engine, or acting as a blogging platform focused on recipes.

I believe there is a space for a fully realised recipe based social network web application in which people can follow their friends and share their particular cooking styles while interacting with and sampling/critiquing styles of their friends and family.

# Slide 3

This knowledge, coupled with the statistics that suggest that mobile browsing has now taken over desktop browsing (**Eric Enge at stonetemple.com, 2018**), suggests there is a possibility for a successful recipe based social network application. I believe by building the application with it also being a Progressive Web App so that users can easily use the sites functionality on their phones in the same way as a native app, and also on the browser, a large userbase could be obtained.

# Slide 4

The diagram outlines a high level view of the overall system architecture.

In this representation, the client represents a user’s browser, on whichever device they are using e.g. tablet, phone, desktop etc. When the client navigates to the web application’s URL, the React application loads in their browser and displays the application to the user. The user can navigate to different “pages” in the application, which will be handles by React using the React Router. The application itself will be a single page application, but will function as a multi-page website. The React Router will handle all routing in the application. Page change renders will happen extremely quickly as the application will not have to request an entirely new document on each page change, but will simply load the specific component which is related to the current URL (e.g. when the user navigates to /login, the React application will load the Login Component).

The server is built using node.js and the express framework to create a RESTful API. The server will store data in a MongoDB instance hosted at **mlab.com**. The server will interact with the database using the Mongoose JavaScript library for easily querying and parsing responses form the database. Authentication will be handled with JSON Web Tokens using the jwtwebtoken npm package, which will enable a user to securely communicate with the server from the client-side React application.

# Slide 5

The user interface was designed to be “familiar” to a user that has experience with social networks.

The landing page displays the app logo, following by register and login buttons. Underneath this information is a series of demonstration gifs showing some basic usage of the application to give a potential user a brief taste of the app. The navigation bar remains in the top corner throughout the app.

# Slide 6

The image on the current slide shows the “feed” page, this page displays the most recent recipes posted to the app. The left highlighted card shows information about the currently logged in user, clicking on this card takes the user to their profile page. The main body of content on this page is the actual recipe card as shown in the centre of the page, the small heart in the corner displays the number of likes this particular recipe has. Clicking on this heart icon will pop up a modal box displaying the names of the user that liked the recipe, with links to their profile.

# Slide 7

The users profile page contains a top section with the users profile pic, name, username and short “blurb” or self-description. A logged in user, while visiting their own profile, will see an options button beside their username, represented by a cogwheel icon, which allows them to edit some basic profile information including profile pic, name, username and blurb.

If a user visits another users page, the options cogwheel is replaced with a “follow” button, which becomes an “unfollow” button if the user already follows that user.

The following section underneath contains 3 user related sub routes. The first being the users personally posted recipes, the second is their personal recipe book, which can contain other users recipes and acts as a sort of favourites list. And finally the social section which shows who the user follows, and who follows them.

# Slide 8

The create recipe page, which has full validation, including a check for an associated image. The gif gives a quick example of the usage of some of the inputs including numeric inputs and dropdown selections. The user can also add as many ingredients or method steps as they would like, and can easily remove them from the list. The user can also change the image to something else before submitting the form.

# Slide 9

The page for each individual recipe is laid out as shown in the current slide. The image and basic information is displayed at the top of the page. Underneath the image are several buttons. The firsts two of which are the like button, and the add to recipebook button respectively. When clicked they have an immediate affect on the state of the app and the database. The edit and delete buttons are only available to users who have created the recipe, or to admin users.

# Slide 10

For a logged in user, each recipes ingredients can be immediately added to the users shopping list.

Here the user can click on items as they pick them up, remove them entirely from the list or add new items using the input box. When all items have been selected as picked up, the option to clear the list is shown. The shopping list is and all actions taken are saved to the database, so that if you add an item on your desktop, it will be there on your mobile app etc.

# Slide 11

Development of the application began with the server, and as there was no client, or “visual” aspect, of the application, testing of the API took place using the Postman tool.

Postman is used to test API endpoints by making HTTP requests and displaying the returned data. It is by this method that each of the server’s endpoints was tested.

The figure shows an example of testing the login POST route. The email and password fields have been added, with an intentionally incorrect email address. The response is shown at the bottom of the image as “email: User not found”. Thus the application correctly detects that the email does not connect to a known user in the database.

# Slide 12

Once he correct email and password have been submitted, the server returns a success message, along with the JSON Web Token which will be used by the client to confirm a logged in user when connecting to the server side application.

All other routes are tested in exactly this fashion, testing the bearer token, all form input data and requests for information.

# Slide 13

Testing of the client-side React/Redux application takes place using three key technologies. The developer console, the React Devtools and the Redux Devtools.

The image shows the React dev tools in action. The “Hero” component is selected in the top half of the screenshot, and both the components props and state are shown in the bottom half. The props and state and even be modified in real time using these tools to test the effects on the application directly. As can be seen by the highlighted “isAuthenticated” prop. This is a Boolean value which can be set to true or false simply by clicking the checkbox in the devtools. This is extremely useful for testing the effects of data on your application.

# Slide 14

The image shows the Redux dev tools in action. The left section of the image shows the actions that are being sent to the reducers along with some timing information related to each action. The right side shows the applications current application-wide state.

The recipes object has been expanded to give an example of the nested nature of the state. The recipes object contains recipe array, of which the 0th element is expanded to show the information contained in that recipe.

# Slide 15

In developing a Progressive Web App using modern JavaScript technologies, several key advantages and disadvantages have become apparent.

Advantages

* Once a web site is downloaded, subsequent visits to the page can be lightning fast as cached data can be served before going to the network.
* PWAs offer the ability to provide push and desktop notifications to the client which has the possibility of improving the user experience.
* A PWA can be saved to the home screen of mobile devices and, by opening the app this way, can take on the appearance and behavior of a native application.
* Developing a PWA can save a lot of money and time as the web and mobile versions of the app are actually the same code base and do not require the time and money needed to develop a native iOS and Android application.
* PWAs are very quick and easy to install, as by simply visiting the URL, the user is prompted to add the application to the home screen, removing the necessity of visiting the app store.

Disadvantages

* Push notifications can become annoying to general users once many websites begin to implement the technology and request the users permission to send notifications.
* The notification to alert the user to the fact that they can save the application to their home screen might also annoy some users that may simply want to browse the mobile version of the site.
* PWAs cannot access all functionality of a device that a native application might be able to access, e.g. PWAs cannot access the devices Bluetooth, ambient lighting settings, advanced camera controls, contact, proximity sensors among other features (Khan, 2018).

# Slide 16

Due to the modular nature of the development of the application, editing existing, or adding new functionality to the application becomes far easier. Adding an entirely new aspect to the application might require adding new routes to the API, creating new React components and new Redux actions/reducers.

The application itself could be expanded to include data from external APIs, for example a food information API could be brought it to give information about different ingredients for users that are unfamiliar with certain foods.

A private messaging service could be implemented to allow users to privately communicate with one another.

A system of alerting the user to new or updated site content could be implemented, perhaps with email notifications.

The application currently uses Bootstrap to provide the CSS and basic layout functionality (dropdowns, modals etc), however given more time- a fully application specific design could be implemented which could improve the overall UI/UX, visual appearance and reduce the “generic” feeling associated with the Bootstrap framework.