# FIT3140 Advanced Programming, Assignment 1

Abigail Lewis

August 13, 2017

This report will highlight the main design choices, testing methods, added features and how to use the Online Voting System. An Agile approach was taken to build a system that allows users to log in (using an email address, password and voting ID), cast a vote (by choosing one of three options) and view results. If users are still able to vote on a topic, these results are displayed in real time.

## Project Design

For the development of the Online Voting System, Node.js was chosen as a backend language and a combination of HTML, CSS and Bootstrap were used to generate the front end.

### Technologies

Node.js was chosen over other alternatives such as Python and PHP for several reasons. Firstly, it allows simple implementation of the JavaScript library Socket.io which not only is vital in achieving communication between the client and server but is also a requirement in the assignment specification. This use of Socket.io could also have been achieved using PHP. However, this would require the use of Websockets which would result in a more complex implementation. Due to having no experience in any language other than PHP, Node.js was a more natural choice than languages such as Python because Node.js is just JavaScript. Having used JavaScript before, less time would need to be spent learning the syntax of a new language and more time spent on development of the system. Furthermore, attributes of Node.js, such as its speed, thriving open source community (which produced the Socket.io module) and how it allows developers to write JavaScript server side and client side, contributed to this decision.

Several Node.js modules were also used, including express, body-parser, fs, http and socket.io. express was used to set up middleware to respond to HTTP requests. The middleware used is body-parser, this reads form input and stores it in a JavaScript object allowing the server to access the form input. For this to happen, the http module was required to transfer data over the Hypertext Transfer Protocol (HTTP). Additionally, the Node.js file system module, fs, was needed to read the data from the JSON file.

Bootstrap was used to style the HTML pages and achieve an attractive and consistent design for the application. Bootstrap allows fast development which was a necessity when building a full application in a short amount of time. It is also easy to integrate and has a big support community. Although it is a very customisable language, another style sheet was used to make small changes to Bootstrap elements and to add simple styling that didnt require Bootstrap. For example, changing the colour of paragraph text.

JavaScript was used to manipulate and keep the page content up to date. On the voting page it was used to specify functions such as sendVote(), which sent a message containing the option the user voted for over socket.io and changeSelection() which was called whenever there was a change in selection of the radio buttons. With regards to the results page it was used to update the information displayed by the paragraph tags that output the result percentages.

In terms of data storage, a combination of JSON and JavaScript data structures were implemented. JSON was chosen over other data-interchange formats, such as XML, due to it being so lightweight. XML items are required to be wrapped in open and close tags whereas in JSON the tag is just named once. By wrapping the data in fewer tags the result is faster data transfer and consequently a faster application. JSON also has other benefits such as being easy to read and is compatible with JavaScript in that functions such as JSON.parse() allow simple conversion between JSON and JavaScript objects.

## Data Structures

The Online Voting System makes use of appropriate data structures to aid efficiency. For example, the login details of users (email and password) are stored in a dictionary like structure rather than an array. JavaScript does not have a built-in dictionary data structure, but a similar structure was created by using JavaScript objects as associative arrays. This means that when checking if a user is authorized, it is only necessary to iterate through the key to see if the email address exists and if it does, only then check the password.

Due to the iterative approach to development when using an Agile methodology, developing a system that can easily change is a necessity. The other main data structure used was a JSON file to store the question that users were voting on and the three options they could vote for. This was chosen more for extendibility than efficiency. In the future, the application may have more than one question that users can vote for. The use of a JSON file would mean that these questions and answers are able to be easily maintained. Also in terms of extendibility, the login page asks for a Voting ID which represents the question that the user wishes to vote for. Although currently there is only one question, so a voting ID of 1 is the only valid input, it means that this extension could easily be implemented in the future.

## Testing

Unit testing was carried out at each increment of development to ensure that each part that was implemented in the system worked as it should before it was integrated with other components. For example, the login page was fully tested before development of the voting section of the application. This testing integrated into development is an important part of an Agile approach. Once the application was completed System testing was used to ensure all the implemented components worked together. For example, testing that the user could log into the system and then vote and then log out again.

## Added Features

Several additional features were added to the application that were beyond the original specification. Some of which have already been mentioned, for example the use of the JSON file and Voting ID. In addition to this extensive error and information messages were provided using JavaScript. These included not only navigating to an error page if the user entered an incorrect email address, password and voting ID combination when logging in but also messages, on the voting and results pages, informing the user if voting had closed. This added a great deal of clarity to the application. For example, when on the results page the heading in the text box either read Live voting results when voting is still open or Final voting results when voting has closed. This was achieved by using JavaScript to read from the JSON file and checking if the end date of voting had been reached. Then depending on this outcome, different messages where appended to elements from the HTML document. Another example is, in addition to disabling the buttons on the voting page when voting has closed, a message, Voting has closed is displayed to the user so they understand why the buttons on the form have been disabled.

## Running the application

The application can be run by navigating to localhost:8080 which will automatically display the login page. There are two login credentials that can be used to access the application, user1@email.com and password1 or user2@email.com and password2. A voting ID of 1 is needed in both cases, as the application currently only supports one question. After gaining access to the system, the user can navigate to three pages. If voting has closed, they are able to view the voting page (but not vote, as the buttons which do this will have been disabled) and view the final results on the results page. If voting is still open, they can cast a vote and then are redirected to the results page to view the live results. Alternatively, they can access the about page.

There are also several commands that may be necessary to run to install modules:

- npm install express - -save
- npm install socket.io - -save
- npm install body-parser - -save
- npm install fs - -save
- npm install http - -save