**README**

This is the readme file for the Firmstep Develop Test, here I will explain the steps I took to create the solution.

I started out by making a decision whether to user JavaScript of PHP. I chose Javascript over PHP as I have more experience in this language and felt that this would work to my advantage. When it came down to the layout of the page I decided that the most important aspect of the layout was to get all the information displayed as the sample layout suggested, so I decided to stick as closely to it as possible, with a few tweaks.

I chose to the use Google’s Materialize CSS framework as I have used this on a couple of projects in my current job and find it very clean and modern. On top of this it’s responsive so the solution would be adaptable to not just desktop PCs but tablets and PCs.

I started out by making columns on for the user to enter customer data and the other to display the queue. The materialise css works in a similar way to Bootstrap by using a grid system consisting of 12 columns. To make his responsive I used the **col** class and set it to **s12** and **l6,** this translate to 12 columns for small screen and 6 columns for a large screen (desktop or larger).

I started out by creating the radio button services as per the example. As the fields for each of the types on offer were different, I decided to implement a tabbed section where each tab would be named as a service and would show the fields associated with it below. Each tab had a div with a similar name and this is where the fields would be held.

To store the customer information I needed a database to store the information. After doing some research I decided to use Taffy DB. I had not heard of this database before doing this assessment but as it was light weight it seemed like a sensible solution. In order to submit entries to the database a table had to be created, this done by following the ‘getting started’ instructions on the Taffy website, I created a database called customerDB.

**Inserting records into the database**

Inserting records into the database required the data to be converted into JSON. I started off by creating a customer object in JavaScript and adding the information from the fields to it.

Although TaffyDB creates a unique ID for each entry I decided to make my own by counting the number of records in the database and adding 1 to it. This would make it unique and more meaningful.

To get the type of customer I created a variable called *CurrentTab* which would hold the value of the selected tab.

To get the combined the name of a citizen I created a function called getFullName(), this used the string concat method to combine the title, first name and last name.

The service was taken from the radio buttons. When the user clicks on a service its value gets stored in the *currentService* variable.

By using the *JSON.stringify* command I was able to convert the object to a JSON string and insert it to the database.

**Retrieving the results and displaying them in a table**

To display the information in the queue table I needed to extract it from the database. The stringify function can also be used for this, but only when used with the command JSON.parse, this converted the JSON text back into a JavaScript object. If there was more than 1 record it would create multiple JavaScript objects.

As the number of records returned can’t be predetermined I decided to use a for..var..in loop which would iterate through all the records and a second for..var..in loop to iterate over the properties of each record to display the values and insert them into the table.

To separate the table structure from the table content a create 2 separate function. The first function is createQueueTbl(), which creates the table and its headers. The second function is populateQueueTbl(), which pulls the data from the database and creates the rows and contents in the table body. So that the whole table doesn’t need to be created everytime a record is created the createQueueTbl() function is run when the page is loaded and the populateQueueTbl() function is run whenever a new record is inserted.

**Validation**

I have used a very simple validation method to check whether a service has been selected by checking the *checked* property. If all services return *false* then the customer cannot be inserted into the queue.