SENG2021 – Software Engineering Workshop 2B: Testing report

Authors: Group 10 - Hariharen Veerakumar, Yuwei Chen, Ahsan Rahman, Saishav Agarwal

# latest version of software architecture

**Web Server – Controller**

* Processes request from “controller” (client)
* Sends request to application server for necessary data to modify view

**Client – Controller**

* Used to enable interactivity with the user
* Interactions occur via displayed buttons in the view
* Examples include:
  + Buttons to view seating statistics in different scenarios
  + Access to different infographics representing seating statistics

**Client – View**

* Displays infographics in a web browser
* Infographic is dynamic, and includes graphical representations of the controller

Send request to update view

Send request for modification

**Application Server - Model**

* Sends requests to different external servers to obtain necessary data for modifying view
  + Application will be developed for this instance
* Examples include:
  + Retrieving statistical data regarding seating
  + Retrieving major concerns and policies of a party

Send data

Send request

Send user event

The design is based on the original design from design report 1, based around the client-server Web architecture. This was deemed the most convenient for our purpose, since there will be a variety of back-end needs required to produce the desired result, especially considering its dynamic nature.

Model-View-Controller proxy (MVC) pattern has been used to help organise the various components of the system, such that each component can be dedicated to a simple task in itself. The MVC pattern serves to be very useful for web-based applications.

We sent a request to external source from official Australian elections website to return the number of seats each party holds for all time frames and Houses, in this stage the controller will send a request to the model to update the view.

The front-end system for the client is essentially for viewing purposes only, with elements of the controller integrated within to help with the function of the web system. We are aiming to display all information on a minimal number of pages (at most one, currently). We aim to do so in a feasible, aesthetically pleasing manner. All of this will be downloaded from the server to the client, whereby it will be opened by the client to display the required information.

The view is finally updated, and the web app shows the data fetched from another website. Although we tried implementing most of the MVC model as discussed, some features could not be incorporated due to time constraints.

Components

The main necessary component, given a client, is a variety of server components and systems within a computer itself. Provided this server can run indefinitely, the design will work such that the client can access the information at any time necessary. The client view is generated as a web application for dynamic use by the user. The client will also be able to send requests to the server.

A web server essentially creates the website and sends the information to the client. Various templates, and other data, will be accessed by this server and transmitted to the client upon request from the “controller”. The web server will send requests to the application server to retrieve information by contacting other servers, such as from www.aec.gov.au (Australian Electoral Commission Official Site) to collect details regarding seating positions in Houses, for example. Considering some of this information can change extremely readily, we will have to collect this dynamically. The web server then receives this data and uses it to update the client.

Relationships between components, and deployment

The web server will require applications for their respective tasks, mainly concerning the sending and receiving of data to and from other components. In particular, the web server will have to retrieve data from other servers, as well as receive requests and send replies back to the client. Hence, it acts as a ‘model’, in our MVC framework, while the client incorporates the ‘view’ and ‘controller’. This relationship is more so evident in the figure above.

Lastly, the type of browsers to use was chosen based on a universal scoring system used to judge the rendering ability of HTML5 and CSS3 in particular. From experience with all browsers throughout time, we decided that the fairest minimal score, considering the available upgrades for each browser, would be 300. This was also a key determining factor for which OS compatibilities we will be catering for, as seen in the next section.

The browsers chosen include Internet Explorer 10 (scoring 322), Chrome 7 (308), Firefox 6 (333), Safari 5.1 (319) and Opera 11.10 (301).