COMSM0104 Web Technology Report

Qingqi Shi and Muyu Yang

Estimate of Marks

- Overall, we aim to achieve 70+% for our work
- A for HTML
- A for CSS
- A for JS
- A for PNG
- A for SVG
- A for Server
- A for Database
- A for Dynamic pages
- 28+ for Depth (out of 40)

HTML

- We wrote HTML within handlebars template files for ‘home’, ‘game’, ‘browse’ and ‘about’ pages.
- We used standard HTML 5, all pages are validated using an automated validation tool (written as a grunt task).
- We aimed to use semantic tags such as ‘article’, ‘nav’ and ‘footer’ to make our website potentially more search friendly.
- Some ‘div’ elements were necessary for styling purposes.
- We feel completely confident in HTML.

CSS

- We did not use any frameworks. We wrote standard pure CSS to style all pages.
- We aimed to reduce duplicated styles by grouping styles into meaningful classes.
- We investigated multiple issues to achieve the style that we wanted. For example, while a pop-up window is displaying, a backdrop element darkens the page which had to be styled

to be semi-transparent and covers the entire viewport; we investigated different position modes such as the 'fixed' mode for the navigation bar; we also used box-shadows, text-shadows and transitions to enrich the page styles.

- We feel completely confident in CSS.

JAVASCRIPT

- We did not use frameworks for the client-side Javascript.
- We implemented the 'game' using client-side Javascript.
    - We implemented the 'game' using the Model-View-Controller pattern.
    - We manipulated HTML canvas for display of the 'game'.
    - We implemented features such as play, pause, clear canvas, zoom in/out, and speed up/down.
- We implemented the hide and show of pop-up panels by 1) listening for click events and 2) adding and removing classes from elements.
- We've done a lot of string construction and parsing, such as for saving and loading game data.
- We've used javascript to select and manipulate DOM elements (by id, by class name, and by css selector).
- We've done client-side form validation using Javascript.
- We investigated and used browser local storage features through Javascript.
- We also put a lot of time and effort into organising and structuring large scripts.

PNG

- We investigated the use of favicons, we used an online tool to generate png formatted images of our website logo of different sizes, and linked them in HTML.
- The design of our website did not include any bitmap graphics, so we created two images for the about page to show our ability in producing bitmap graphics using Gimp. Only the first one was used in the about page, but we also included it here as it shows that we have used some of the Gimp tools.

- 

    - We took images for the two of us, removed the background for each one by using the free select tool and layer masks.
    - We used thresholding to convert both ‘heads’ into binary images of black and white. Then, we removed all white pixels.
    - We merged the two heads into a single layer.
    - We also imported our website logo and placed it as another layer in front of the head layers.
    - We added radio gradient to both heads and the logo.
    - Finally, we added drop shadow effects to give the image some depth.

- 

        - We used two photographs, each containing one of us, and removed the background using the scissors select tool and layer masks.
        - We created a new document in Gimp and place the two cropped images as layers.
        - We also imported our website logo and placed it as another layer behind everything else.
        - We added drop shadow effects to give the image some depth.

SVG

- We created out site logo using Inkscape
    - 
    - We used rectangle tool to create the square ‘cells’.
    - We used ellipse tool to create the circles to represent a head.
    - Bezier tool was used to create paths. Then we used power stroke to create tapered strokes to make the path look like ‘mustache’. Finally we duplicated and mirrored the path to complete both sides of the ‘mustache’.
    - We used the difference tool to cut out the circles.
    - We used inset and outset tools to thicken some parts of the logo.
- Home page background using Inkscape

    - We created square ‘cells’ and grouped them into different patterns.
    - We spread the patterns around in a way to reduce seams when repeated.
    - We lowered the transparency as it would be used as background.
- Dynamically generated thumbnails in the browse page
    - We also investigated the fundamental code of SVG as we dynamically generate thumbnail images.
    - We made use of SVG lines and rectangles for these thumbnails.

Server

- We created our NodeJS server using Express
- We gained experience dealing with routing requests and obtaining ‘get’ and ‘post’ data
- For form submissions, we redirected the user back to the original page after the request has been processed.
- We have also set up HTTPS and SSL certificate for the server to encrypt all the page. Currently the certificate is self-signed.
- We used sessions and cookies to store user login states.
- As mentioned above, we created server-side scripts to generate SVG thumbnails.

Database

- We used SQLite through the sqlite3 module of NodeJS.

- We paid attention to details such as using prepared statements (data binding) to avoid SQL injection attacks.
- We were specific with table creating, using correct data types and setting ‘not null’, ‘unique’ flags for correct fields.
- We have set up linked tables using foreign key fields
    - We have two tables: Users and Game_States
    - One to many relationship: One user can publish multiple game states
- We also investigated callback and command chaining - submitting a command based on the result of another.

Dynamic Pages

- We delivered dynamic pages by inserting data into handlebars templates.
- We used handlebars helpers, both built-in and custom wrote
    - Block and extend to insert page specific style/script
    - Each loop to insert the list of games
    - If with equal tests to determine if the delete button should be displayed
- We used handlebars partials, which help us organise HTML code into modules so it can be reused.
- Depending on the login state, either sign in/up button or username and sign out button will be displayed in the navigation bar
- If a user loads a game from the browse page, the game page will be initialised with the selected game, otherwise the game page initialises an empty game.
- The delete button will only be displayed on the browse page if the game belongs to the current user.
- Form validation results will also be displayed if the user sends invalid data to the server (if client-side validation is by-passed).

Depth

This project was created by Qingqi Shi and Muyu Yang. We began by setting up a repository on Github that makes it easy for use to work cooperatively. The project was setup in such a way that at anytime anyone can clone it from Github and follow a detailed list of instructions to install the necessary node modules and run the server. We think this effort was worthwhile as we did not encounter any set-up related problems during this project.

Our idea was to create a website for John Conway's Game of Life. We imagined features such as the game simulator itself, a user login system, a publish feature for users to share their games, and a browse feature to see all published games. We have successfully implemented all those features.

We avoided using large frameworks such as Bootstrap and JQuery to demonstrate our understanding of the fundamental principles of pure HTML, CSS and Javascript.

We adopted a modern and elegant design language that uses mostly black and white with some blue as a highlighting colour. Following this theme of design throughout all pages, we manually created page structures and styled every element. We also used Google web fonts to enrich the reading experience of our users. Not to mention the animation library that we used to give an elegant zoom-in animation as the user scrolls the page. We have put a lot of thoughts into the details and we are very proud of our design.

We spent a lot of effort and time into programming the game simulation itself. It included things like using HTML canvas, capturing mouse events, simulate the evolution of cells, adjustment of game state, speed and zoom. We spent a lot of time into structuring the code into a Model-View-Controller architecture. We also investigated local storages so that the user does not lose a cool game just by signing in or publishing.

We also feel quite proud of our website’s security. We followed good practices such as encrypted user credentials using hash with salt, not including usernames or passwords in cookies, and using prepared statements to prevent SQL injection. Moreover, we encrypted all traffic to and from the website by using SSL and HTTPS. We spent a lot of time figuring out how to self sign the SSL certificate with maximum security (setting up encryption methods).

Github

- https://github.com/QingqiShi/Game-of-Life-Website

Used Frameworks and Libraries

- NodeJS modules
    - Express
    - Grunt
    - Body-parser
    - Cookie-parser
    - Express-session
    - Hbs
    - Morgan
    - Serve-favicon
    - Sqlite3
- Google web fonts
- AOS - Animate on scroll
- Font Awesome icons