- A for HTML
- A for CSS
- A for JS
- A for PNG
- A for SVG
- A for Server
- B for Database
- A for Dynamic pages
- ? for Depth (out of 40)

HTML (Grade A)

Through creating a number of HTML pages, we have become confident with HTML structure. We started with delivering our pages as XHTML, but ran into an issue with the google maps API due to this. We therefore instead passed our HTML through a validator to ensure they are correct, and stopped delivering our pages as XHTML.

CSS (Grade A)

We created a CSS file for each HTML page, to avoid the use of inline CSS. We made extensive use of flexboxes to create interesting page layouts with both vertical and horizontal flow. We spent a lot of time using media queries to ensure that our pages display correctly on phones, tablets and PCs; for example, we change the top menu from the PC page into a hamburger menu when the site is viewed on a phone, and we rearrange image layouts to fit the screen in the best manner.

JS (Grade A)

Using JS event listeners we created an on-hover animation for the links to the location pages, which gradually blurs the image and increases the font size. We also used JS to implement an animation for the opening and closing of the hamburger menu, and implemented an accordion for the “all locations” page, to make it easier to navigate a large list of locations. We used the Google Maps API to place a map within our site and display markers at specific locations, with info text and a custom icon. When a user clicks a link which redirects to an item on the current page (for example clicking “Map” on the home page) we smoothly scroll to this item, so its is clear they have remained on the same page.

PNG (Grade A)

We cropped and resized many of the images on our page before converting them to PNG. We also superimposed our character into some images. To make the character blend into the scene we used gaussian blur (to simulate depth of field), motion blur (to simulate motion of the character), and transparency cutouts (so that the character was occluded correctly by aspects of the image they are behind). We also used the histogram tool to modify the color of the character slightly so that they blended in with the lighting of the image.

SVG (Grade A)

We created SVG social media icons, an SVG for our character, and an SVG for the circular character icon used in the “about” section. To create the social media icons we used path editing, transformations and gradients. To create the character SVG, we also used some freehand drawing, and for the circular avtar we used a pattern to create the background.

Server (Grade A)

When starting to develop the server we opted to use Express. We quickly realised however that we did not have a good understanding of what the code we were writing was doing, and decided to instead write a pure Node server using await/async.

After implementing delivery of static pages, we worked on implementing a user account system. When a user creates an account by entering a username,email and password, we store this information as an entry in the database. The password is stored securely by first appending a random salt (which is stored in the database) and then hashing it using the SHA-256 algorithm. When a user attempts to login, we check their password by appending the appropriate salt from the database, hash the result and compare to the stored hash. Once a user has logged in successfully, we use a cookie to allow them to remain logged in.

We then implemented the comments system. This involved rendering dynamic pages using EJS templates. Once a logged in user posts a comment, the server reads from the body of the request and stores this comment in the database along with the name of the commenter. Whenever the page is loaded, the comments from the database are inserted into an EJS template which is then rendered and delivered.

We also implemented simple URL validation to prevent users from accessing files they should not have access to. We also used redirection: if the user enters an invalid URL they are redirected to a dynamic error page, and when a user logs in successfully they are redirected to the home page. We finally created a key pair and certificate using OpenSSL and used this to switch the server from HTTP to HTTPS, in order to improve the security of our website.

Database (Grade B)

Using SQLite we created a database to store comments and user accounts. When a user creates an account or submits a comment we store this information in the database. This information is accessed by the server when rendering dynamic pages using EJS. We used prepared statements for all of our SQLite queries, to eliminate the possibility of SQL injection.

Dynamic Pages (Grade A)

We experimented with both EJS and Pug. We decided to use EJS, as we had already created all the HTML files needed for our site, and less modification of these files was needed when using EJS. When a page with comments on is requested, all comments for this page are read from the database and inserted into the EJS template, which is then rendered and delivered. Furthermore, if the user is not logged in, the page is delivered such that the comments box is disabled and they are prompted to login. Similarly, we use EJS templates to notify users whether they are logged in: if a user is logged in, the "*signup/login"* button on the header is replaced with "*logged in as X"* using EJS templates. A dynamic page is also used to display error messages when an invalid URL is entered, or other errors occur. Finally, we also use dynamic pages to notify the user that they have successfully created an account or entered invalid login details.