**Personal Portfolio – Release 2**

1. **Implemented uploading of media**

For Release one, we had failed to deliver uploading of user data to our website. I therefore looked into assisting Jae in solving this problem, which was taking longer than expected. I created an upload script known *as upload\_Ross\_Two.php*, which was used to upload image files into a directory on the server known as ‘uploads’. The upload script validates each image file, for file type and size, and then saves the file to the ‘uploads’ folder on the server. As I was not aware of the server permissions, it took me a while to get this script to successfully upload the first file, once I figured out how to change the permissions. This was a very important script as it was used by the other team members in creating upload scripts for the other media types, and allowed us to begin working on the downloading and streaming features.

1. **Improved ‘uploads’ directory for users**

When we first started uploading files to the server, our initial directory was to an ‘uploads’ folder, which contained all the users’ files. This was a problem, as all our users’ files were mixed together in the same directory, making them vulnerable to being overwritten or hacked. I therefore went about designing a storage system, where each user had their own uploads folder. Therefore, I changed the *register\_val.php* script to automatically create a new personal upload folder for when a new user registered successfully. This proved quite time consuming as I had to figure out how to change the server privileges to allow for a new directory to be created each time a user registered. Once completed, the new directory system proved very effective in only accessing the data of the user logged in and therefore reducing the database queries response time.

1. **Created Image Gallery Script**

We received feedback from our client team that they would like to see the users’ images displayed in a gallery format. At this stage we had created a list view of the image files and I took on the task of creating a gallery view. I created a script called *image\_gallery\_script.php* which grabs the user’s images and creates a thumbnail copy in another directory called ‘thumbnails’. These thumbnails are then displayed on the image page in a gallery format, where the user can click to enlarge each image and then scroll through them. I then created a toggle button, which changes between the list view and gallery view as required by the client.

1. **Created Video Gallery Script**

Once I had competed the image gallery, I looked into replicating something similar with the video page, which also had a list view of video files. I was able to implement a HTML5 video player which could scroll through a gallery of video thumbnails and play whichever video was selected by the user. This proved more difficult to implement than the image gallery in terms of dynamically creating image thumbnails for the video files. Once complete, I added a toggle button to switch between list view and gallery view to meet the client’s expectations, and provide a consistent format between the different media pages. The implementation can be found in *video\_gallery\_test.php*.

1. **Implemented client side upload validation**

Our initial upload scripts included server side validation of file types and size, however our client team noticed that the client side validation had not been implemented. I therefore created a JavaScript validation script called *upload\_val.js*, which contained a validation function for each media type. This checked the extension and the size of the user’s file and if correct, would allow the user to submit the upload form, to be tested by the server-side validation. This feature is important for security, however it also minimises redirecting the user to the PHP validation page, therefore making the application more user-friendly.