WEB TECH Report: Mihai-Alexandru Cristache(mc14468) & Tudor Petrescu(tp15462)

HTML (MARK: **A)**:

- In order to generate html code we used the Pug template engine. This provides a cleaner approach to writing html as well as various neat features such as variable injection, conditionals, loops and template inheritance. Thus, each view is built on top of the layout template by **extending** it. This helps with consistency across the website as well as ensuring the code is DRY(do not repeat yourself). Inside the ***layout.pug*** the **block content** statement is replaced in each of the views accordingly, while inheriting the style of the parent template.
- Validation: done using https://html5.validator.nu
- Testing for different devices and older browser versions using the inbuilt features of mozilla firefox, internet explorer, opera and google chrome. Some small css problems with window resizing were found on the older versions, but the functionality is still the same.
- All the files related to this section can be found inside the **views/** folder

CSS (MARK: **A**):

- Almost each page has its separate stylesheet file, so that we keep the declaration coherent.
- All the elements and separate graphics that we generated with the css were meant to fit our website theme, not very complex and sophisticated, but a clean and simple one. Different styles were added using: **background editing, shading for boxes, sizing, margins, paddings, font editing, cursor editing, display settings** etc.
- Special features: besides the basic css mentioned above, transitions were added to make things more appealing(e.g. intro page animation is made purely in css; fading animation for the image slideshow on homepage).
- Last, but not least, all the dimensions settings were changed to be a percentage of the window width or height in order to help the page fit perfectly in any screen format. Some elements such as the menu bar converts into a hamburger menu when the window is too small for the buttons to be visible enough.
- All the stylesheets can be found in the **/public/stylesheets** folder

JS(Mark: A):

- Client side javascript was used to interact or animate different parts of the website when needed. All these files can be found inside **public/javascripts**.
- The blur effect of the home page when first entering the website only lasts 2 seconds. This was done by using the **setTimeout** function and providing a callback which removes the **blur** class from the body element after the specified time.
- The slideshow of images present on the home page is achieved via Javascript as well. This time, the **setInterval** function is called in order to loop through a given set of images.

SVG (MARK: **A**):

- Line drawing style used.
- All svg files used were integrally drawn and animated by us, using Inkscape. (images in **about**, **events**, **forum**, animation- on the loading screen <tachometer>). The tachometer was animated using the svg animation, not javascript.
- For the svg creation we used: **edit path, bezier curves, brushes, text insertion, path algebras(difference, union, intersection), multiple layers, measurement tools, path effects, cut path, etc**.

PNG (MARK: **A**):

- All the images used were either drawn from scratch or edited.
- Images such as the footer logos were redesigned completely using **gimp**(**brushes, selection tools, layers, blur, color selector etc.**).
- The society logo was created from scratch starting from the university's original logo. The small images that were added on top were first edited to fit our logo both in terms of size and also colours and quality(as much as possible). As for the text, an inbuilt font was used that was edited afterwards to be unique for our application.
- All the images used throughout the website are found in the **/public/images** folder

SERVER(MARK: **A**):

- For the server we used ExpressJS framework. Thus, it is organised in three parts. In ***bin/www*** the server is created, ***app.js*** handles the loading of requested middlewares as well as page delivery and inside the ***routes/*** folder are the routes that deliver the appropriate response for each request.
- The uploads in the gallery are handles by the **multer** package and stored inside the /**public/uploads** folder. Upon deletion, the file system package is used in order to remove the files from disk as well.
- We also secured the communication between the server and the client using HTTPS by generating a certificate and a key using OpenSSL and passing them as parameters to the https.createServer function located in the ***www*** file located in the ***bin/*** folder.
- Furthermore, all pages that modify data in any way have CSRF protection in order to be protected against this type of attack. This was achieved by using the ***csurf*** package and generating a token which is then passed to the view.
- In order to prevent XSS vulnerabilities we ensured the cookies are only accessed by the server by using the ***express-session*** and setting the cookie setting for httpOnly and secure to true.
- To protect against Clickjacking attacks, we disabled the ability to embed our website into a frame or iframe tag by disabling the X-Frame-Options header for all responses via the ***helmet*** package
- Another precaution we took was against Brute Force and DDOS attacks. Thus, we limited the number of requests for each IP using the ***express-rate-limit*** package
- Last, but not least we ensured that the sensitive information in all forms has the autocomplete feature off, thus ensuring that passwords or emails are not saved. There

are pages that are only available to admins and we put a lot of work in making sure that regular users can not perform admin tasks.

Database(Mark: **A)**:

- Each requests is either a GET or a POST on the server which then makes the appropriate query to the database which is found in the root folder under the name **data.db**.
- For the database we used sqlite with the sqlite3 node module to deal with the queries made to the database.
- There are different actions implemented such as getting rows from the database, deleting rows and updating rows in the database. Any possible error that might occur from the interaction with the database is handled to throw an error.
- The database design is quite complex. There are multiple tables such as users, events, forum_posts, etc. These are also linked. For example, there is a **Many to Many** relationship between the users and the events, which means that a user can follow multiple events and an event can be followed by multiple users. Moreover, each forum post reply is connected to a forum post row in the forum_posts table. By specifying the **ON DELETE CASCADE** action when creating the database the database will automatically delete any rows in the child table that are connected to the deleted rows in the parent table.
- The interaction with the database uses prepared statements, thus defending ourselves against SQL injection.
- A seed file found in the root folder called **seed.js** creates all the required tables and populates the website with some dummy data

Dynamic Pages(Mark: **A):**

- We used Pug templating engine to render our views. This supports variable injection which provides us with the means to dynamically display content on our pages. On top of variable injection, Pug provides other neat features such as conditionals and for each loops. One example of a dynamic page is the **events** page where all users can follow events, but only the admins can create/delete/edit events.
- Another similar example is the side bar menu which is displayed only if a user is logged in. This is done by using the **user** object as a flag to hide or show that component