**Mission Sheet for KF5012**

Please create one of these sheets **for each mission you have done,** and please include these sheets **in your project submission**.

The point of this form is to give you a chance to direct us to the work you have done, so that marks can be given fairly, and without missing any of your work. We want you to draw attention to which work meets the criteria for the specified elements in the mission brief. Also this gives you a chance to draw attention to any work you have done that goes beyond expectations

|  |  |
| --- | --- |
| Mission name | Application Programming Interface (API) |
| Team name | KeepingUpAppearances |
| Student responsible for mission (it team write “team”) | Dawid Michniuk |
| For team missions: should this mission be marked? | N/A |
| In the project brief, the details for this mission outlined several **elements** that should be created for the mission. Please identify clearly which work meets which elements. | * **API Design** – the main API file (api.py) loads in the index page at the root of the localhost, followed by /showcase/, /showcase\_testing/ and /showcase\_training/ routes to showcase specific pictures. Finally, the two main routes, /train/ and /colorize/ are tasked with doing their respective roles. Flask was used for many reasons, firstly, it offered minimalistic set-up, meaning it was easy to add components (easy when compared to Django, where everything is set up from the start). The main page offers all the options of the api, allowing for easy transition between parts of the program * **Model Preparation** – Train/ and Test/ folders had to be moved into “static/” folder to ensure it complies with how Flask works with files. I also had to create templates for different website parts the API would go to. * **Server Setup** – Flask was used. It creates a localhost server at port number 5000, with several routes, as previously mentioned in API Design point. * **User Testing** – I provided try-catch statements around the baseline and then did my best to test both, the API and the model and underlying classes. From my tests, it turned out that all folders that I wanted to access should be in “static/” folder, unless the file was a template of a html file. This testing allowed me to produce proper sub-pages of the API. |
| If you have work that you think means you have **gone beyond expectations** for the mission, please give details here. Note: this includes sensible work you have done to compensate for team members dropping out or team missions not being completed. | The API uses good software engineering practices. I aimed to use novel design choices.  Mainly, webpages use templates, and extend from a main html file, rather than being copy and paste code. This means that a potential bug needs to be fixed in one place to affect all pages, rather than having to manually fix it, page by page.  Furthermore, training loop was conducted in a novel way, where the script was imported in. This proved to be very good at avoiding code errors when copying the code over but disabled the possibility of adding specific requirements from command like (like requesting a different Train folder pathway for example). |
| Any other **notes** you want to make about this mission. | * With no classes on API (from university), and no previous knowledge of REST API, I had to develop my understanding, as well as produce code by using many different resources. This leads to me not really knowing what can be counted as “going beyond expectations”. * Due to Keras/Tensorflow version issues, the debug and multithreading options of the API have to be disabled for the program to run correctly. |