**Artefact 1) IPO Chart**

*The IPO chart serves as a reference to the software developers while developing this project.*

|  |  |  |
| --- | --- | --- |
| **INPUT** | **PROCESS** | **OUTPUT** |
| Search Term  Map Search Button | Reads the database to find matching results  Plots results on map. | List of features relevant to the user  Map displaying features in the user’s area |
| Search Term  Feature search button | Reads the database to find matching results.  Compiles a list of links of results. | List of features relevant to the user  Links to page of features. |
| City Name Button | Fetches map of city from the database | Displays map of city to user |
| Register menu Button | Fetches the “register” page from the controller. | Displays the “register” page. |
| Username  Password  Confirm password  Confirm account registration button | Checks whether the username is available  Checks whether the username is valid (i.e. no characters such as ‘$’ or ‘&’)  Checks whether password is secure | If the above criteria are met, the details are stored in the database, creating the new account.  If the criteria is not met, an error message is display, prompting the user to fix the relevant fields.  The new user is then redirected to the home page. |
| Log out menu button | The current user session is terminated by the controller. | The user is logged out. |
| Log in menu button | The controller fetches the “log in” page. | Redirects the user to the log in page. |
| Username  password  Log in button | The username and password entered are compared to the database | If the username and password match, the user is logged in and redirected to the home page. |

**Artefact 2) Coding Standards**

*When collaborating in a group, coding standards are necessary in keeping the code clean and understandable for all group members. Without proper coding standards, the repository can become an inconsistent mess with poorly named variables and parts of the code that make sense to some team members but not to others.*

As coding standards for our project, we are using PEP 8 for Python code. The official PEP 8 styling guide can be found here: <https://www.python.org/dev/peps/pep-0008/>

As for html, team members should generally follow the style set out by w3schools: <https://www.w3schools.com/html/html5_syntax.asp>

On top of that: Keep things like argument ordering and casing consistent.

Lower case is preferred over upper case.

Using forward slashes for clarity when possible. For example:

Yes: <link *href*="{% static 'favicon.ico' %}" *rel*="shortcut icon"/>

No: <link *href*="{% static 'favicon.ico' %}" *rel*="shortcut icon">

Avoid useless whitespace/Line-breaks

Inline code should be padded. For example:

Yes: {% if user.is\_authenticated %}

No: {%if user.is\_authenticated%}

Other than that, just keep things tidy

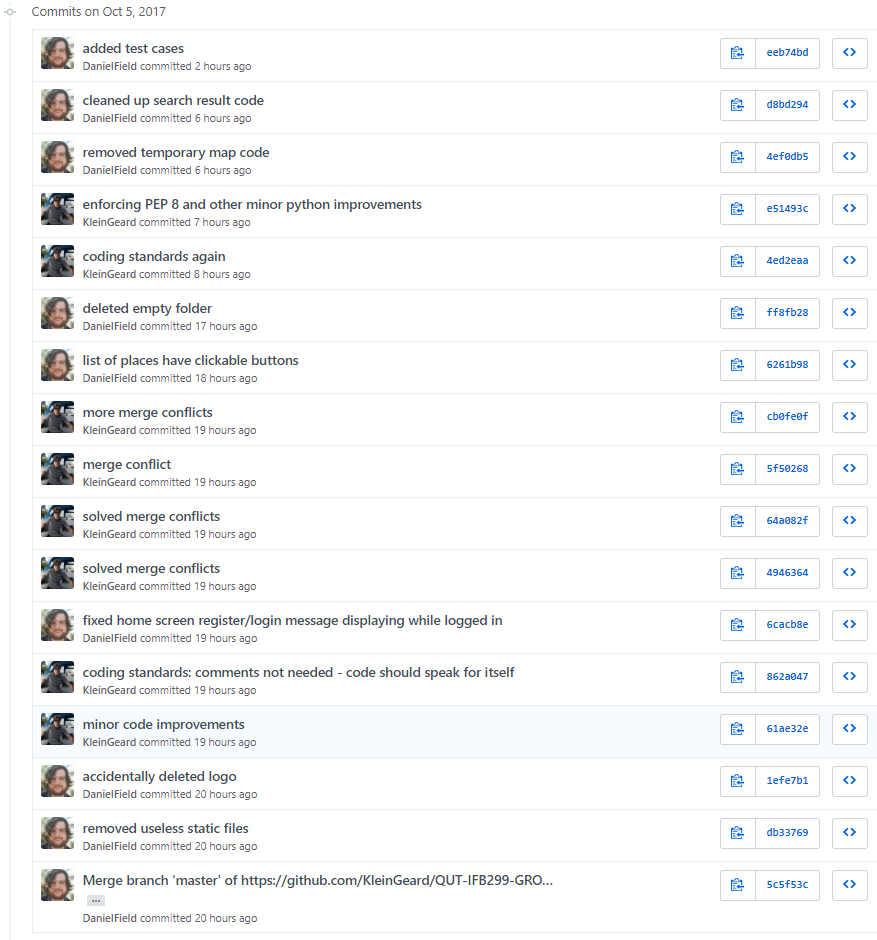
**Enforcing coding standards:**

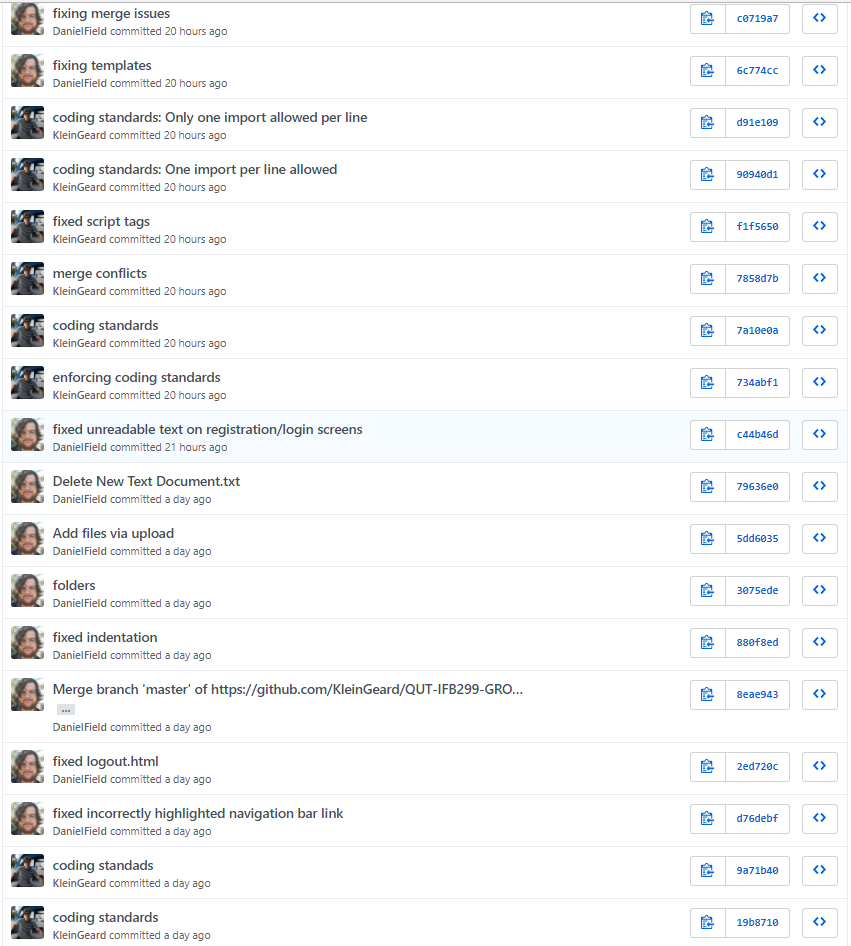
We kept these coding standards in mind from the beginning of the project so **it went well to begin with** so there were no giant messes in the first place. There were, however, a few minor cases where I had to enforce our coding standards.

1. <https://github.com/KleinGeard/QUT-IFB299-GROUP-94/commit/4ed2eaaadfc48d1b29acecf7942f1485cbad0327>
2. <https://github.com/KleinGeard/QUT-IFB299-GROUP-94/commit/e51493cb6c75eb3e30d4346cad01a7de8f8ea9e5>
3. <https://github.com/KleinGeard/QUT-IFB299-GROUP-94/commit/f4b25c2e11d1511e91b05cbf67461806e8ae6f28>

My full commit log showing all enforcements of coding standards, as well the solving of merge conflict issues and code optimisations, can be found here: <https://github.com/KleinGeard/QUT-IFB299-GROUP-94/commits?author=KleinGeard&since=2017-09-30T14:00:00Z&until=2017-10-05T14:00:00Z>

As can be seen, I have been working on enforcing coding standards and generally increasing the overall quality of the code.





**Artefact 3) Justification for architecture**

*It's important to carefully plan out the structure of your web application rather than just jumping into things. By justifying the architecture, we are showing that our decision is well thought out valid and not rushed.*

Django uses what they call “model-template-view” architecture, which is pretty much the exact same thing as model-view-controller but with a different name.

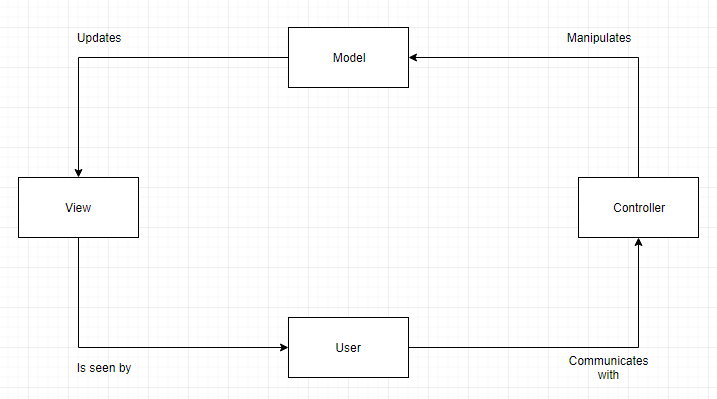
MVCs strength is primarily in preventing repetition and creating a proper clean and safe structure for web applications. Benefits include:

* **Very** loose coupling. With the server, client, and database completely separated, parts of the program may be modified without affecting other parts of the program.
* High cohesion. Django’s MVC architecture logically groups related elements together within each component, making Django programs extremely navigable.
* Efficiently separates the client side from the server and database making for a clean and safely structured web application.
* Good security. With MVC based frameworks, the backend (controller and model) is completely hidden from the client, making Django web applications less likely to be exploited.

Django also features a rich template system which basically does most of the work for us. The templates allow us to quickly implement common features such as “log in/log out”, “add thing to database”, “remove thing to database”, “modify thing in database”, and more.

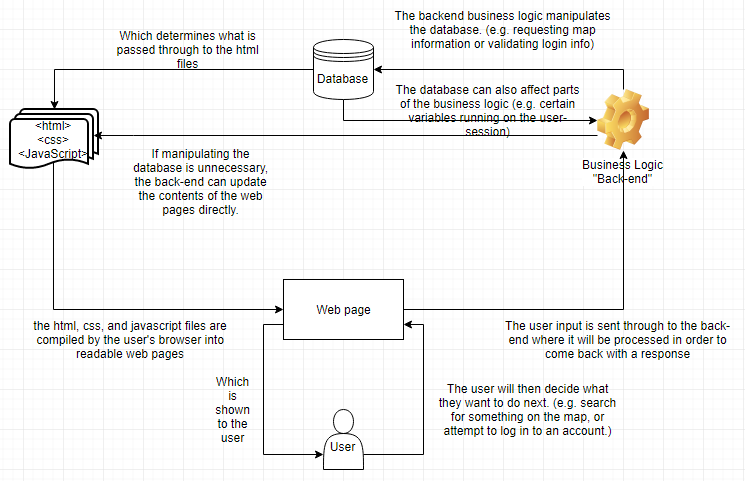
**Artefact 4) Logical and physical diagrams of the chosen architecture**

*It's important for developers to have a clear understanding of the architecture of the web application they are developing. Visual diagrams assist with this and help reduce confusion.*



As the above logical diagram shows, MVC is split into 3 components; the model (database), view (front-end/displays/templates), and the controller (business logic).

A physical representation of this architecture is as follows:



**Artefact 5) Testing**

*The testing was completed to ensure that the web-application runs the way we want to it. It lets the developers know of any issues with the program and gives them general feedback on the quality of the application.*

See test-case file in the .zip folder I uploaded