CFG Degree 2022
Project - Pick-A-Movie
(fka "Decision making app")

Team: Helen, Phoenix, Vania, Ziling

Introduction

Our aim from the very beginning of this project, was to provide a simple, random choice of a film for those of us who suffer from annoying indecisiveness when it comes to what we watch on a movie night, faced with the ever-growing libraries of streaming services, tv channels and the internet at large.

Opting for this fun and time-saving idea, we attempted a project that we would all be engaged in, and see the potential for, from both a programming/creator's mindset as well as putting ourselves in the role of a user.

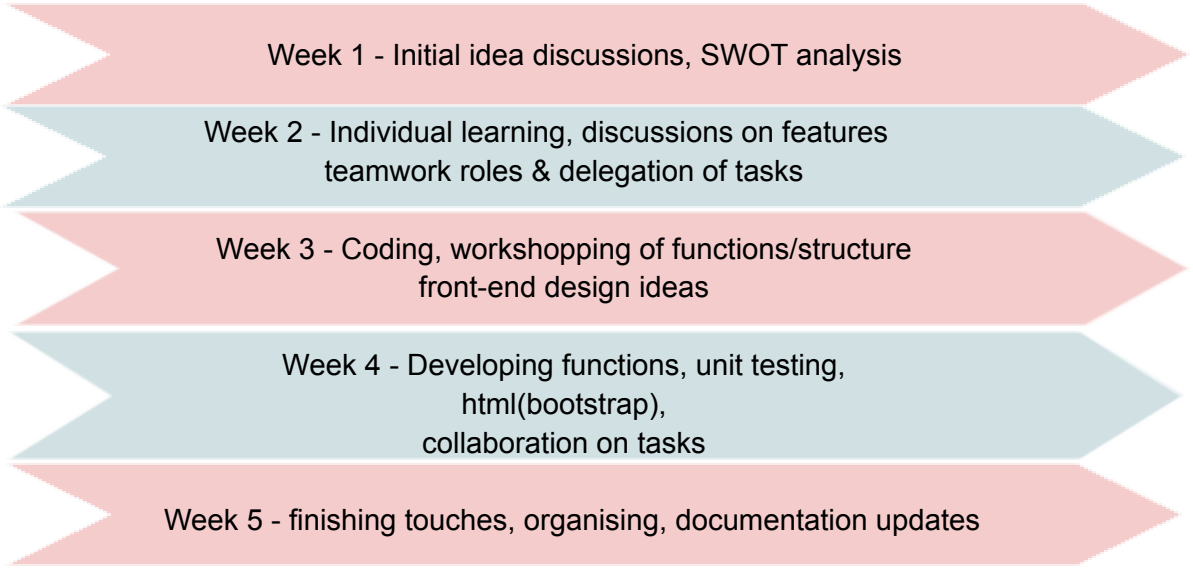
We envisioned that the specific aim of an inherently random choice and simple features of our web-based project would lend itself to being user friendly and offer quite a unique aspect to recommending something that users might not have prior knowledge of or otherwise encounter in their day to day. With this in mind we chose a database of movies that encourages user-input and offers more selection than any one single streaming service might. The Movie Database is a community built and open source database of movies and tv-shows that is continuously updated and offers a huge selection of media that includes, but is not limited to, the popular and trending "current" movies.

Our Objectives

- A random movie offered to users
- Clear, simple user interface
- Unique use of chosen API
- Integration of back-end code to front-end web pages

For a flexible roadmap, as a team we decided early on to set ourselves goals to achieve, discussed each week at our weekly meetings, and to track issues once we began coding, via githubs "issue" and "tasks" features. This helped us to see what had been assigned and to go some way to providing a running record of what was accomplished each week. This was not without some issues admittedly - as we discussed during our first few meetings, between the four of us we each had a different level of understanding and prior use of Github and some of us had to work on our familiarity with it.
The team were very upfront with what we considered our own strengths and weaknesses in the different areas of python coding and project work, as well as the parts of our project that were obviously not covered by the course or things that we had to work on or bring our own outside knowledge to.

Roadmap



STRENGTHS	**WEAKNESSES**
As a team we have the following experience: - Project Management - Project Coordination - Python - Working within team structures - Product Management - Front End Development, including html and CSS - Familiarity with GitHub.	Some members are unfamiliar with aspects of GitHub. Knowledge gaps with Python, Flask, html, testing.
OPPORTUNITIES	**THREATS**
To learn from each other's experience and skills. To expand on existing Python knowledge. To experience all aspects of planning and implementing a project through each stage of development.	Only 4 weeks to finish the project, with limited time to spend on it on top of existing commitments. Different availability to work online at the same time. Distribution of responsibilities.

User experience

When the user navigates to our application they should encounter a simple web page with a single button, offering to pick a random film.

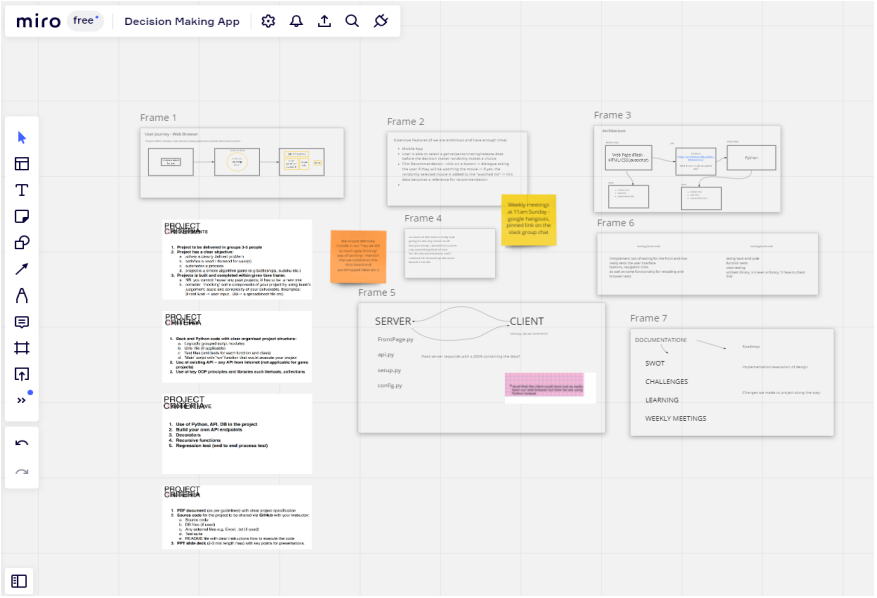
Upon button press, the user is navigated to another page, displaying a result. Their random film will be shown - title at the top, a link for more information (to TheMovieDB themselves, where synopsis, a rating, cast, genre and a lot more information is available) and the movie's poster will be shown - if no poster is available a placeholder picture with "not available" is displayed. Underneath this a button will offer a "Try for another film" option to the user if they wish to opt for another random film. Upon pressing a different film will be displayed, and this can be repeated any number of times.
A navigational "home" link is available on the page for a quick return to the main index page.

Should an error occur, as a function of the api call (movie id returns a missing movie, or the api is not available) the page will show an error message. Should the user mistype a webpage, or try to access a html page not intended, a 404 error will show.

Specifications and Design

As we decided upon our basic design, we used a Miro board to keep track of our project's framework, and how each component would interact. Initially we knew that Flask would be the framework and that at least two web pages would be required for our user to interact with.
So a back end, - python code - interacting with an api, creating a flask app, and setup files.
And a front end, - html, css, and the use of templates where useful.

We did consider, if there was any need for a database element to this design, but discarded the idea as unnecessary. If there were a good argument to be made for keeping track of the users "watched movies" or a list of rejected movies, or even a user rating for example - a database would be ill-suited in this type of app/web based application and instead a link to the actual TheMovieDataBase's own login and user functions would serve those purposes.



Our roadmap was initially realised via our two core ideas to feature 1) Randomness and 2) A simple project that could hold scope to be extended, or even implemented multiple times for other forms of media or beyond. These ideals formed the basis for choosing everything from our specific API (themoviedb.org) and the front-end html/css templates, as well as the python libraries, and adhering to the pythonic best practices of keeping our back-end code easy to read and understand.

We began with a lot of discussions! Inevitably there were lots of ideas for functionality and "possibilities" that could easily lead to more work than our original approach anticipated so we dedicated part of our

workshop discussions to “functional” and “future” (which became also known as the “if we had the time” pile of ideas).
Throughout the process, we tried to keep an even weight of work, with regular check in’s and communication of when team members would be working on specific features, with estimations.

We discussed the Django framework, and after some reading, decided that the lightweight Flask would actually be what we needed, especially in our “keep it simple stupid” approach to the project.

The libraries, modules and packages used in this project were:

- ★ API: TheMovieDatabase.org

- ★ Python:
  - Flask - lightweight web application framework
  - Random - pseudo random number gen module
  - Requests - http request library
  - Unittest - unit testing framework

- ★ Selenium: Open source tools & libraries for web-testing

Our challenges and achievements

Towards the end of our project we made a concerted effort to reflect on what we found challenging and what we learned specifically for this assignment.

“I sometimes found timekeeping challenging when it came to making sure we could work on problems and not have to wait for something to be finished to start another task.”
“I learned a lot more about Flask and how frameworks can work with python, as well as how important good API documentation is!”

“I found working with GitHub and Pycharm challenging at first, in particular working on different branches, creating tickets, and merging them back together. It was difficult to divide the tasks up to know what to work on.”

“I personally found it super challenging to write unit tests for front-end, also found difficult when we were planning the sprint and raising the tickets before understanding exactly what we would need to do.”

“I found identifying, planning and breaking down the tasks difficult, especially in the beginning as we started the project from scratch.”
“I learned how the project came together, with API responding to the BE and the FE is able to render the information retrieved from the database.”

Teamwork and communication

Our team used a flexible, agile approach to team working with very frequent check-ins over Slack and a weekly Google hangout on Sundays. We spent time workshopping ideas on how we would implement the basic idea, and what features we would work on using a collaborative Miro board and Google Docs.

Once we had assigned individual learning and the tasks that we had identified, we used issues on github to review each other's code and approve merges. Weekly meetings were held throughout to ensure we

were all on the same page and team members were very proactive in ensuring that all members were aware of the current tasks and progress.

We scheduled a Sprint Retrospective after all tasks were completed and our app was functioning. This was to allocate time for us to learn and reflect on what went well, what didn't, and how we could improve for next time. Ziling led this meeting via Google Meet and we used EasyRetro to collaborate and share our thoughts and ideas. More specifics on this can be found in our conclusion.

Implementation challenges

Some of the challenges found during the process necessitated changes in our code. Though none of our changes affected the user experience or any large scale overhauling of any planned features or design.

During the API coding and gathering relevant information on how we would make the endpoints we needed, it became apparent that the functions we were defining did not allow for further parameters to be passed, and the way we planned to call directly a movie by it's identification number meant if we wanted to later implement a "select genre" function (for example) would cause us to have to think again on the random movie functionality. The only change this caused in our existing code was to alter the response code so that instead of trying to pass a parameter to filter out adult films, we instead had to re-run the code if an adult film was fetched by our query.

The biggest challenge for our team was the rapid learning of how to manage creating a front-end to the project and have it interact with our functions and display what we needed, with our limited experience in specific front-end coding. When planning we knew that the Flask framework took functioning python code and could make simple web based apps, but we admittedly did not have the full understanding of exactly how something more than a text output could be accomplished until we devoted time to more research.

Testing and Evaluation

The team was happy to look over code that had been written initially on a case by case basis as Phoenix and Vania started uploading the first pieces of python to github. Manual testing was done by each team member ensuring we all had access to what the code should be able to do and could see the progress.

As Helen began uploading front-end html and we saw the structure of how everything needed to interact, we rolled out the need for functional tests and could see the elements of our project that would benefit and require testing.

After researching the many different kinds of testing, we planned some Unittests and Ziling brilliantly identified that Selenium would be a great addition for the front-end, specifically to test our browser-based functionality.

Our tests are planned to cover the functionality of components, accessibility and server-side validations. Having the tests implemented helped us to identify issues throughout the development cycle and thereby improving our design and performance.

Unit tests - Back End:
As our core feature interacts with an API, we wrote unit tests to focus on the HTTP status responses for each method and data retrieved from the resources are in the expected format. On top of this, as a

preventive measure in case the type of the data changes from the API, we have added tests to ensure the expected data types were returned to precisely identify errors in the future.

To ensure individual components of the code were functioning as expected we created unit tests for each method. This would ensure we picked up any bugs or errors in the code early on. We also made tests to check the returned data type from the API is as expected. Because Python is a dynamically typed language, any data type could be received from the API, we wanted to ensure the expected data types were returned to prevent any errors in the future. Upon further reading, it was noted that in many developer opinions, some of these type tests may be superfluous.

Selenium WebDriver - Smoke Test:
The framework was chosen as it is used for automating web-based applications and its fair amount of documentation. We have utilised Selenium to automate a smoke test to test our main features to ensure the quality of our application, and from the results, we were quicker at identifying issues and became more confident in the user experience. Another benefit from the automation is that we can easily run the smoke test after a code release to ensure thorough testing. However, the limitation is the test script is currently running on Google Chrome browser and to extensively test the application's responsiveness, we can implement the test scripts on multiple browsers in the future.

Selenium Unit Testing - Front End:
To test the UI elements (client-side), we have implemented unit testing with Selenium. This allows us to access each element on the page and test if they are as expected. During the development process, we have maintained our tests and added as more elements were introduced. As our project is relatively small and tests required are lean, we have found unit testing and Selenium are well suited frameworks to use.

CONCLUSION

We completed the specifications of our initial ideas outlined in our aims and objectives, resulting in a functional app which achieves our goal: to select a random film for the user and display that film, via title and image - as well as the option to try again.

Our only constraint in this project has been time and managing that limitation against our various ideas for what would make this even better as a small project.

Due to limitations in time, our focus was on creating a simple but functional app which selects a film, however the end result has potential to develop more extensive features, including the ability for the user to select a genre of the film they want to watch, whilst keeping it random. For recommendations of "similar films" to pop up below the random film displayed, and to enhance the front-end with a more interactive loading page or spinning wheel, as a visual display of the application. We considered keeping track of film id's that weren't suitable, or were rejected by the user in a callable array and to have the random id's filtered through that before calling the api. We looked at the sheer amount of parameters and sections that TheMovieDatabase has and had to stop wanting to add more things to the to-do pile!

Ultimately we prioritised a complete project, where we all learned aspects of development, of planning, time management, and working within a team under pressure, vs something half-done or that would leave us feeling less-than accomplished.

During our retrospective, we all agreed that the team worked well together, and complemented each other's strengths and weaknesses. We identified what each member thought went well, and how we should improve if the project were to continue;

The team were relieved to have a functioning app, and that our planning from the offset was rewarded. We stayed true to our simple approach and because of this we completed a project in time, without added stress or last minute panic.

We acknowledged that we had all learnt from each other and could rely on each other as a very productive and communicative team. We enjoyed our time together and the ease of which we worked together and helped each other at times. Every member mentioned that we got on well and there were no problems with sharing workloads, egos or any negativity, only compliments!

What we would change if the sprint was to continue is simply a clearer use of tracking issues and tasks on github and to ensure clarity on delegation, and to use pull request comments more to keep track of uploads & reviews.
Though we have all learned a lot during this project inevitably we will always have different ways of working and time constraints in future. This process has been a lot of fun and a great introduction to what to expect from software development and aspects of agile management.