HSC SOFTWARE DESIGN AND DEVELOPMENT

MAJOR PROJECT REVIEW

PROJECT OVERVIEW

Project Title:

Valorant Pro Stats

Project Description:

Project description: Valorant Pro Stats is a web application that allows users to access and explore various statistics and data related to the professional leagues for the popular first-person shooter game Valorant. Users can search for information about professional players, teams, events, and matches. The website is in an early state at present but future enhancements and new features may include a live feed of the latest news and updates from the official Valorant website and social media accounts.

Project technologies: The project uses the following technologies:

- **Python:** Python is a high-level, interpreted, general-purpose programming language that supports multiple programming paradigms. Python is used as the back-end language for the web application, handling the logic, data processing and communication with external APIs.

- **Jinja:** Jinja is a fast, expressive and modern templating engine for Python. Jinja is used to generate HTML pages dynamically based on the data and user input.

- **Flask:** Flask is a lightweight and flexible web framework for Python. Flask is used to create and run the web server, routing the requests and responses between the front-end and the back-end.

- **vlr.gg API:** vlr.gg API is an unofficial API that provides access to various data and statistics related to Valorant. Data from this API is sourced from the popular Valorant website vlr.gg. The project uses this API to fetch information about players, teams, events, matches and more.

- **Bootstrap:** Bootstrap is a popular front-end framework for developing responsive and mobile-friendly websites. Bootstrap is used to design and style the user interface of the website, using predefined components, layouts and themes.

Project Goals and Objectives List:

Goals and Objectives for this Project

- **User-friendly:** The web application should be intuitive and easy, i.e. the user should not need instructions to navigate to the content they need.
- **Dynamic UI:** The web application should be responsive and should be able to be used on mobile devices and computers with varying screen sizes.
- **Accurate and up-to-date:** No matter the amount of information the web application can serve to its users, the information should be up-to-date with whenever the user reloads or when the data changes, in order for the user to feel that they can rely on the information provided by the web application.
- **Smooth:** The web application should not have any major loading times and all animations should be non-intrusive and work on low-performance computer systems.

PROJECT STATUS UPDATE

Project Progress:

Update 1: Jul 30, 2023

- **Flask Development:** I've continued my exploration of Flask and have made significant headway. I've delved into creating routes, handling requests, and building dynamic web applications. Working with Flask has been quite enjoyable, and I'm starting to see how I can integrate it into my project effectively.
- **HTML Formatting:** My understanding of HTML formatting has deepened. I've learned about structuring web pages using HTML, creating forms, and styling elements with CSS. This knowledge is proving to be essential as I work on the frontend of my project.
- **Jinja Templating:** Jinja templating has been a bit challenging but rewarding. I've begun to implement templates in my Flask application, making it easier to maintain and update the HTML code. This has already saved me a lot of time in developing the user interface.
- **Project Progress:** I'm happy to report that my project overview is now complete, and I've made substantial progress on the actual implementation. With my knowledge of Flask, HTML, and Jinja, I'm confident in my ability to complete this project.
- **Adjusted Goals and Objectives:** I've continued to refine my goals and objectives based on my growing knowledge and skillset. This ensures that my project remains realistic and achievable while still challenging me to learn and improve.

Update 2: Aug 11, 2023

- **Flask Learning:** I've been diving deep into Flask, a web framework for Python. I've been working on understanding the basics like setting up routes and handling HTTP requests. It's been a bit challenging but also pretty rewarding to see how web applications can be built using Flask.
- **HTML Formatting:** I've been working on getting the hang of HTML formatting, mainly focusing on structuring web pages and creating forms. CSS has also been in the mix to add some styling. It's not glamorous work, but it's essential for creating user-friendly web interfaces.
- **Jinja Templating:** Jinja templating has been a part of my learning process. It's all about making templates for the web pages in my Flask application, which helps keep the code organized. It's been a useful tool in my web development toolkit.
- **Project Overview:** I've completed the initial planning phase of my project. It's not a masterpiece yet, but I have a clear direction to move forward with.
- **Adjusted Goals and Objectives:** Based on my initial assessment, I've made some changes to my project goals and objectives. It's all about keeping things realistic and achievable while ensuring I continue to learn and improve.

Issues and Problems Encountered

- One issue that I encountered in the development of this project was that while I was implementing the different features of this website, the api would not return any values for its regional ranking dict. I had already build a route in views.py and a dedicated html file for this feature. Unfortunately a fix for this never came about and i was forced to abandon the implementation of this feature.
- Another issue that I encountered during the implementation of my website was that even though my code was essentially perfect, I got errors that such as those called ‘ImportError’ that involved how python packages were installed. I managed to fix this by switching from Visual Studio Code to PyCharm for the implementation. The reason this switch allowed me to proceed without errors was because PyCharm allowed me to automatically develop my app inside a virtual environment. Using a virtual environment

Proposed Solutions

- A proper and working solution to the first problem described was not likely so I ended up removing the feature from the end product. However this problem could potentially be resolved by either hosting the API on a server of my own or using a different API. neither of these solutions ended up being practical so they were not implemented in the final version.
- The solution I utilised to solve this second problem was implementing my program using PyCharm instead of Visual Studio Code. This allowed me to easily set up a virtual environment to run my code in, because virtual environments are automated in PyCharm. Using a virtual environment in this scenario allowed for fresh installs of all the libraries needed for the program to run.

PROJECT PLAN UPDATE

Timeline Adjustments:

- In the original project plan that I submitted I stated that Implementation should commence midway through August. This ended up being spot on and the timeline has not shifted in any major way since I submitted Assessment 1 with the original plan.
- For a detailed timeline read the project progress section above with weekly updates on the different sections and tasks needed to complete the project.

Resource Adjustments:

Resources were not a major focus when I wrote up the original project plan. This was due to the project being entirely cloud based. This is still the case. for this project, no physical personnel or equipment is needed.

The resource planning has shifted somewhat although. In the original plan, the program would be run entirely run on the user's device with information downloaded on the user's device being used as input data for the algorithms. Resource planning has shifted and at this point more of the program will be cloud-based, meaning that load times may be lower and the system requirements of the user's device are less strict.

Budget Adjustments

No adjustments to the budget are necessary.

TESTING PLAN UPDATE

Testing Objectives:

- The objectives of testing this application are high functionality, simple but thorough user interface, high performance, high security, and high compatibility.

Testing Scope:

- The scope of the testing for this application involves primarily just the users machine and the api where they get information that is shown on the computer from.

Testing Methods:

- Testing methods for this application include both manual and automated methods that mimic a user's input with the page. This website will need minimal testing, as there are not many ways a user can manipulate the application. The methods that will most likely be used to test the application is running the app two times on one device to test the application's reliability under stress. Another method of testing is

sending many GET requests to measure how many users the application can handle at any given time.

Testing Timeline:

Rigorous testing will ideally begin towards the end of development, likely around the first of September, which gives 10 days of testing until the application must be completed. However, before this, minor testing and test-driven development will take place, throughout the development process.

IMPLEMENTATION AND TESTING DOCUMENTATION

IMPLEMENTATION

Implementation Overview:

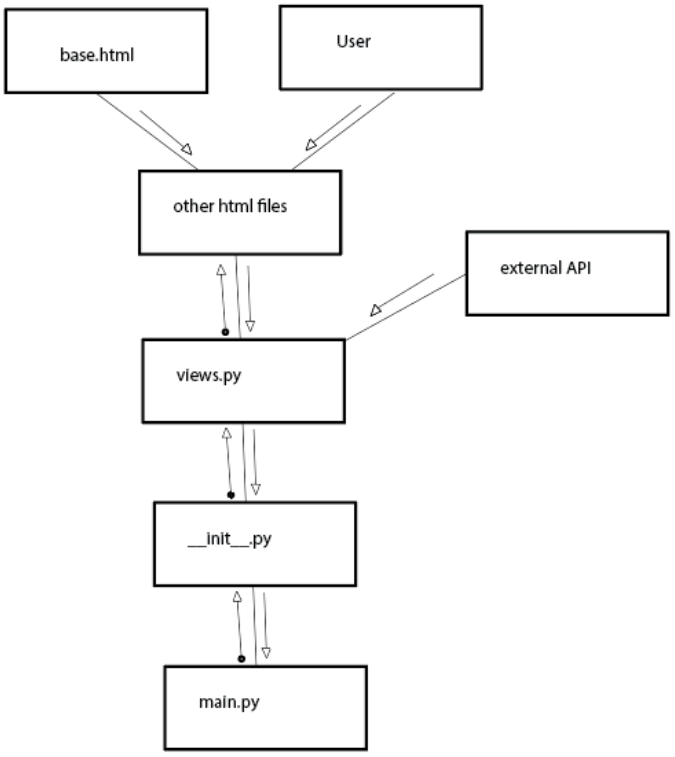
- Implementation of this project occurred over a period of over 20 days. The basis of the website implementation was inspired from a tutorial available on tutorial. After the basic framework of the website was written, I took the time to write out a route for each feature inside the views.py file, as well as a unique html file that corresponds to each feature targeted towards users of the website. An image of some of the routes inside views.py is available below.

- After this was completed, the main focus of the implementation process became the user interface experience of the application. The first step towards a stylish and intuitive user interface was the incorporation of the base.html file. By using this

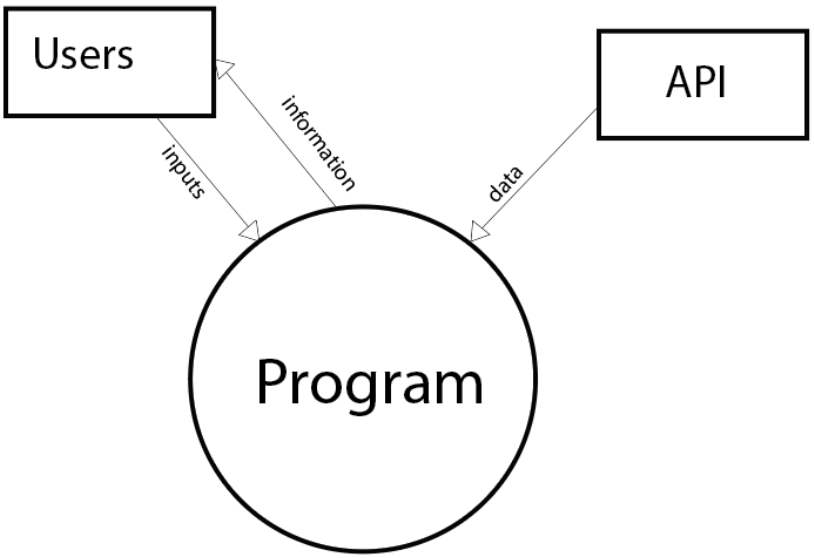
base.html file as a base for all the other html files to extend upon, I was able to incorporate a header and then navigation bar into the website, on all pages.

Technical Documentation:

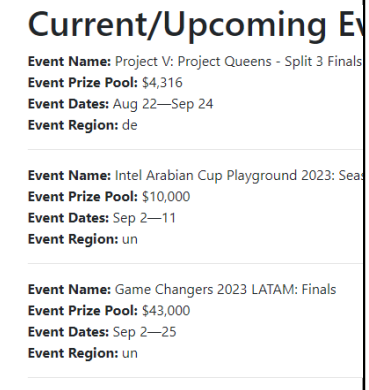
- Structure Chart



- Context Diagram



- IPO Diagram

INPUT	PROCESS	OUTPUT
content matches player events	variable is declared in views.py and is then looped through in an html file.	 (example)

User Documentation:

- A short description of the website is available on the home page and below. Due to the site being relatively simple, this is all the user documentation required to navigate through the website.

    Welcome to this Valorant esports Wiki
    Made by Caleb Henry
    This website currently has five working features:
        Players List
        Upcoming Events
        Upcoming Matches
        Completed Matches

Player Stats

Four of these features are avaliable as links in the navbar, and player stats are avaliable by clicking on one of the player names on the players search page or by specifying a specific player name in the url, for example:

127.0.0.1:5000/players/player_name

Deployment Documentation:

Deployement documentation is available in a README.md file in the github repository. This aids the user in copying the repository to their own machine and running the code.

Flask Web App

Setup & Installation

Make sure you have the latest version of Python installed.

```
git clone <repo-url>
```

```
pip install -r requirements.txt
```

Running The App

```
python main.py
```

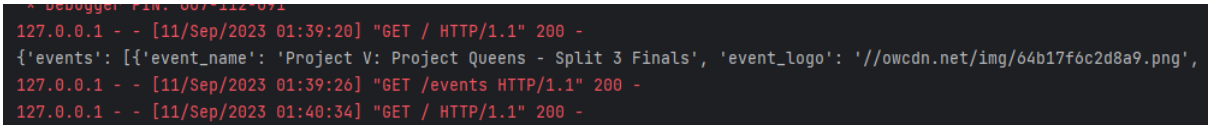
Viewing The App

Go to `http://127.0.0.1:5000`

Testing (TDD Method)

Testing Overview:

Test Driven Development played a large role in the mid and late stages of project development. Once the basic framework was developed, every consecutive feature added featured test driven development. For example, when I was implementing the html pages and views.py routes that housed the majority of the advanced code I utilised print statements. Print statements were utilised to ensure that variables were being passed through multiple different files, e.g. from views.py to events.html. Below is the result of one of the mentioned print statements.



Other testing that was conducted involved stress testing the website itself to see if I could trigger a crash or error by running multiple instances of the website or spamming the server with GET requests. Results were collected from this test only by human observance, not measuring values. This testing was also conducted manually, as automating it would not have much of an impact on the result of the testing.

Testing Results:

Based on the information provided, it appears that the results of the testing were largely positive. The use of Test Driven Development (TDD) and print statements helped to ensure that new features were integrated smoothly and that variables were being passed correctly between different files. Additionally, the stress testing performed on the website revealed that it was able to withstand heavy loads without crashing or producing errors. These results suggest that the testing strategies employed were effective in identifying and addressing potential issues, ultimately contributing to the development of a robust and reliable final product.

Bug Fixes and Enhancements

No bug fixes or enhancements were necessary.

Post-Implementation Review

Review of Project Success:

Were the project goals and objectives met?

Yes, the finished product, i.e. the website satisfies all the goals and objectives set at the conceptualization stage of this project development process.
The website is user friendly, it has dynamic user interface, the information found on it is almost always accurate and up-to-date

Lessons learned:

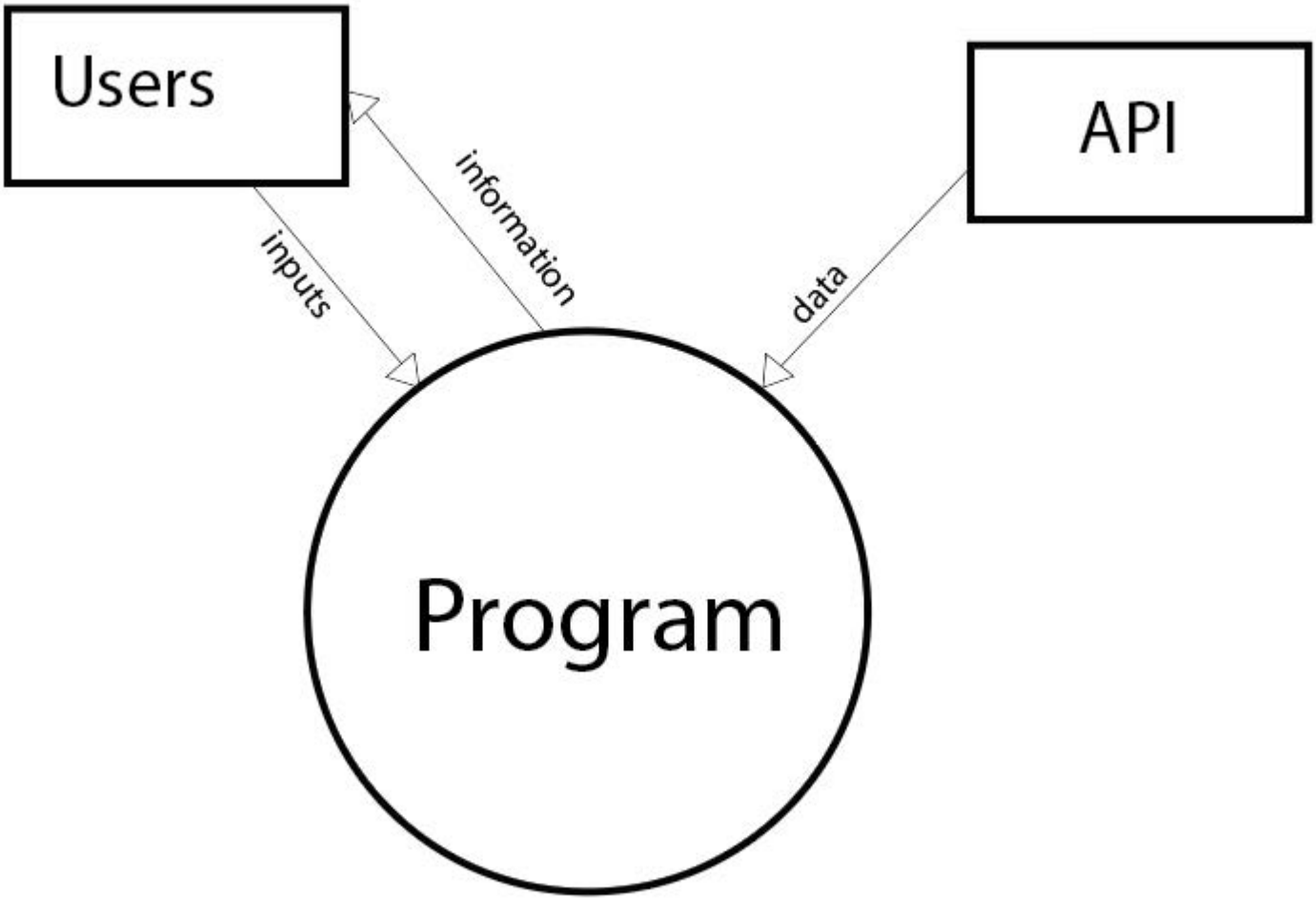
- Learned the importance of test-driven development during implementation.

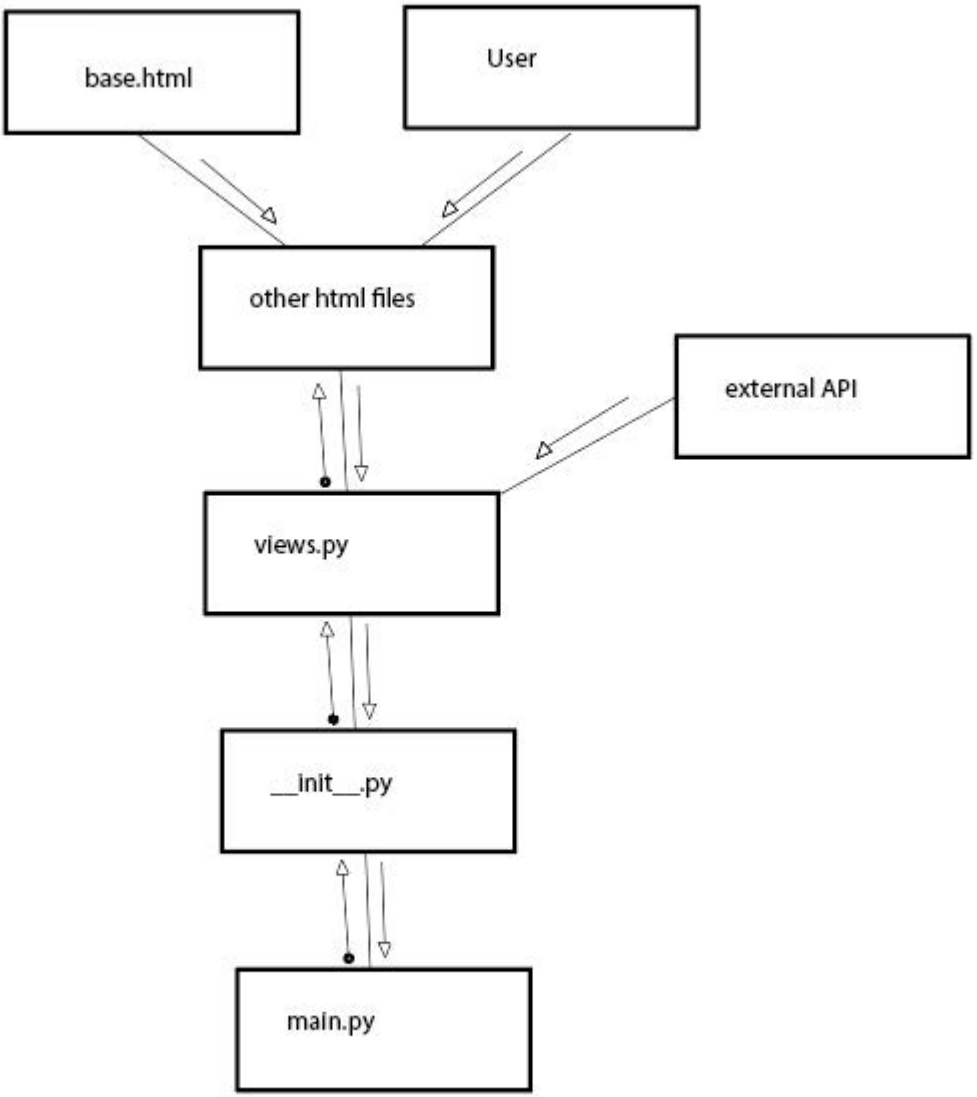
    - The distinct processes of test-driven development necessitate the cleanliness, functionality and efficiency of the code
- Learned a more appropriate timeline and order for what parts of the software development cycle should be done. For example, doing a context diagram before the implementation process is preferable so you have a clearer idea of what the system structure should be. However this is not the case with all technical documentation. It is worth it to make some of them after the implementation process

Next Steps:

The website is in an early state at present but future enhancements and new features may include a live feed of the latest news and updates from the official Valorant website and social media accounts. Ensuring that the ranking page works is also a major focus point in terms of future enhancements.

INPUT	PROCESS	OUTPUT
content matches player events	variable is declared in views.py and is then looped through in an html file.	 (example)


Users
API
information
inputs
data
Program


base.html
User
other html files
external API
views.py
__init__.py
main.py

sdd major gantt chart

- **Implementation**
  - Learning
  - Planning
  - Plan Execution
  - Further Software development
  - Testing
- **Documentation**
  - **Major Project REVIEW**
    - Project Overview
    - Project Status Update
    - Project Plan Update
    - Testing Plan Update
  - **Implementation and Testing Doc...**
    - Implementation Documentation
    - Testing
    - Post-Implementation Review