**SDD HSC major work Documentation**

**1.1 Problem statement**

I realised that during every gamer experience that at one point in time they will face gamer

block, a phase where there is no good game out or don’t know what to play. The solution GamersGateway, using flask and visual studio code I will create a web app that will randomly select a game from a list of thousands on steam using request to API that returns the appid and name, then it will go and use the steampowered API to send a request out for the information and return it in JSON format. Additionally, the app will use a web scraper that will search on Bing for the image relating to the game and then return the first image result that pops up. The Applications features will involve the user generating a game at random. The information about the game being returned will consist of an image the name and developer, trailer, description of what it is and what platforms it is playable on. However, there are some limitations due to time constraints like I was planning to add a feature of saving the games they like but with time it was too difficult to implement, and an additional feature was the ability to view how many people enjoy the game but I with in time frame it would be too hard to implement. An additional limitation is that

**1.2 Legal and Ethical issues**

**1.2.1 Legal**

There are two main legal issues within the app GamersGateway are the use of intellectual property such as copyright law, and when planning to monetize the app. The use of intellectual property is to tie pieces of software to a person and stop it from being copyrighted, in my app the game description, trailer, or images might be copyrighted. And if I display these assets it could infringe on the rights of the content owner, an example in my app is that using steampowered API has links to videos and images relating to the specified game. Where developers may have copyrighted the file and only given access to steam. Additionally, if I plan to monazite my app and make advertising, I will have to make sure that the owner of the external data accepts the use of data on commercials and that the APIs and content providers complies with the commercial use policies.

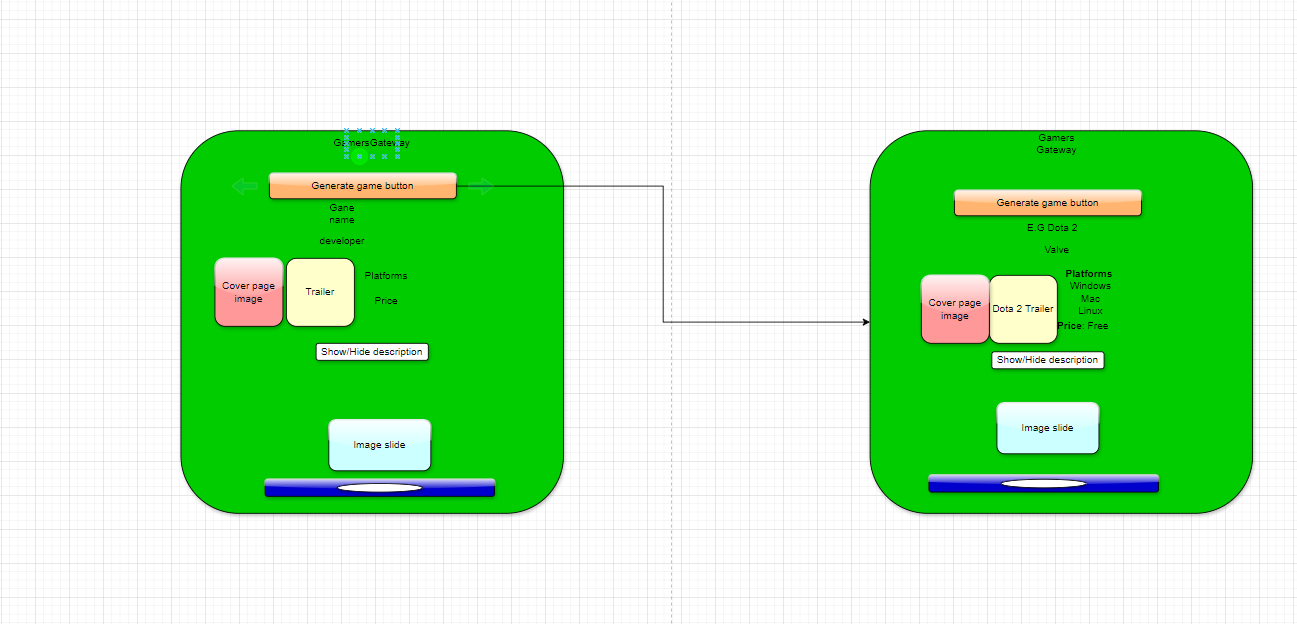
**1.2.2 Ethical**

There are 2 main ethical issues within the app GamersGateway and that is firstly an algorithmic bias opinion, secondly Content moderation through relevant information. Algorithmic bias is a key ethical issue as within the app as a developer I could show favouritism towards specific game developers, meaning the idea of randomly generating games would be tarnished through prompting more frequent displays of certain developers. Furthermore, the use of content moderation as the API being sent data request could contain false information about the developer or game to deter the user from buying the game, which would ruin the reliability of the app and potentially lose user interest.

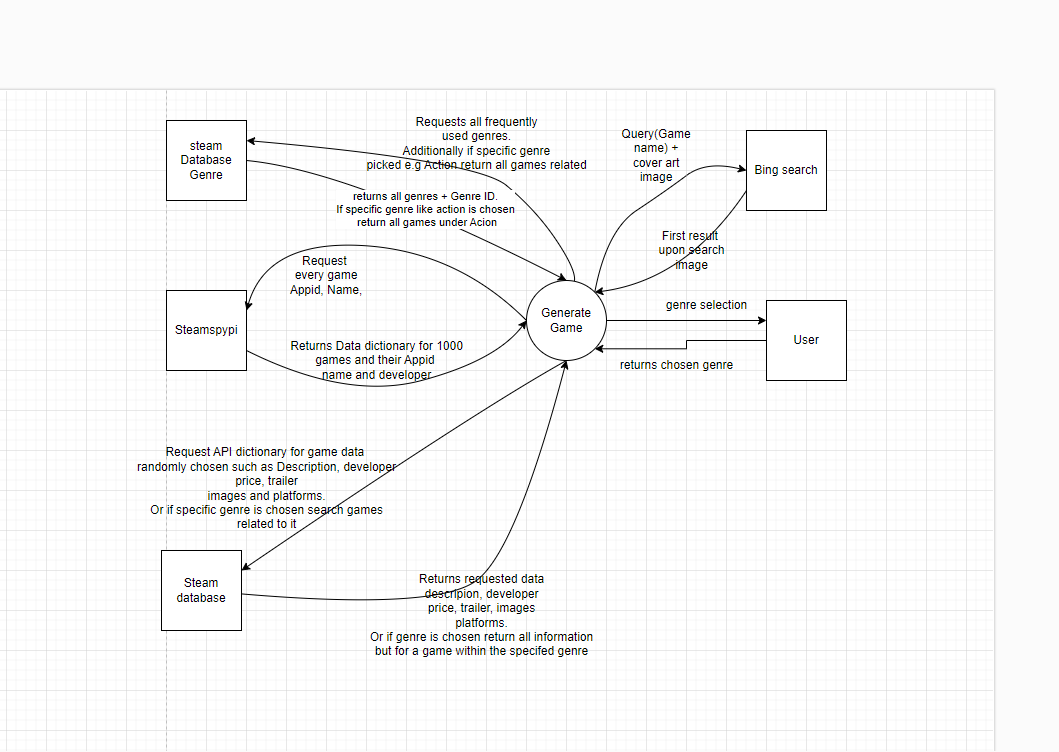
|  |  |
| --- | --- |
| Functional | Non-functional |
| The ability to re generate a random game | A simplistic User interface that doesn’t overload the user with information |
| The ability to watch and see the trailer of the game | The ability to hide/ show the game description |
| The user can view a description | A button to enter Fullscreen to have a better experience when viewing the trailer |
| A way to view images of in game things | Viewed through a carousel on an automatic timer to swap between or the user can interact. |
| A view of what platforms the game can be played on | The view being in nice text not a list of text |

**2.1 Storyboard**

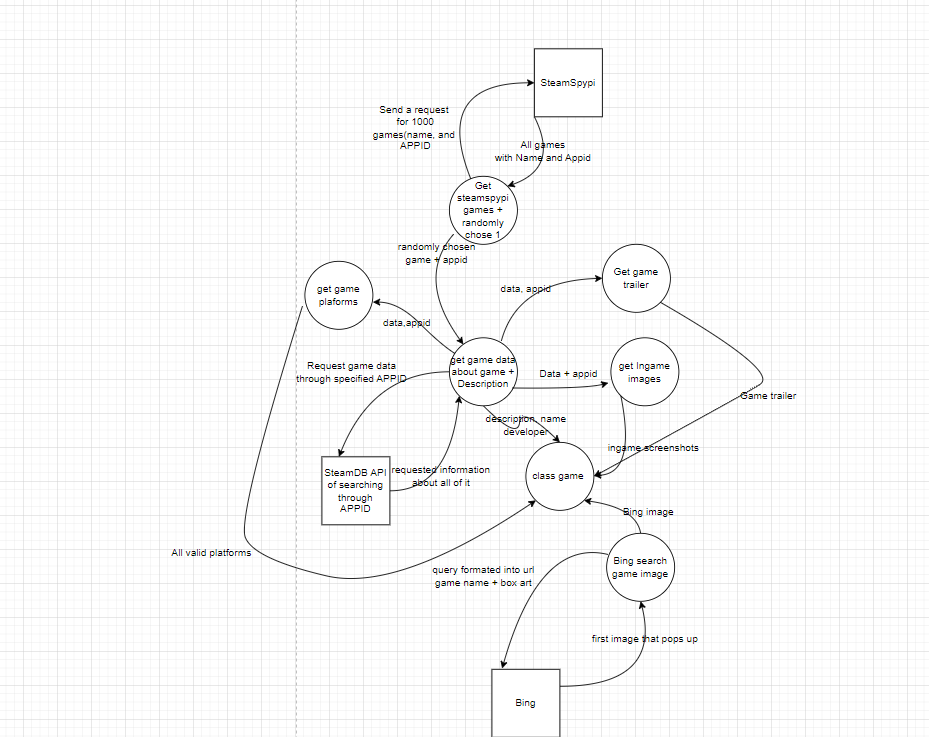
**Every diagram has its file within the zip file call diagram folder for better look**

****

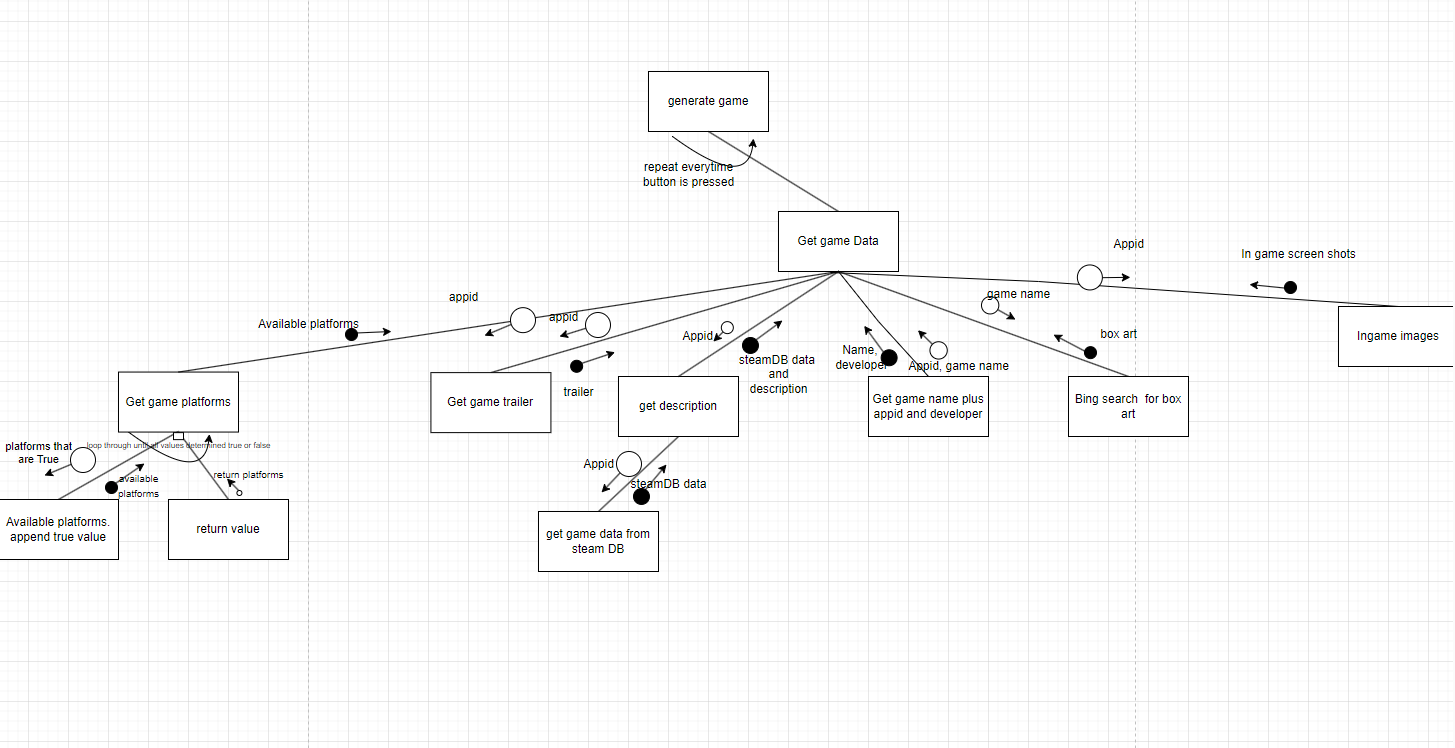
**2.2 Context diagram**

****

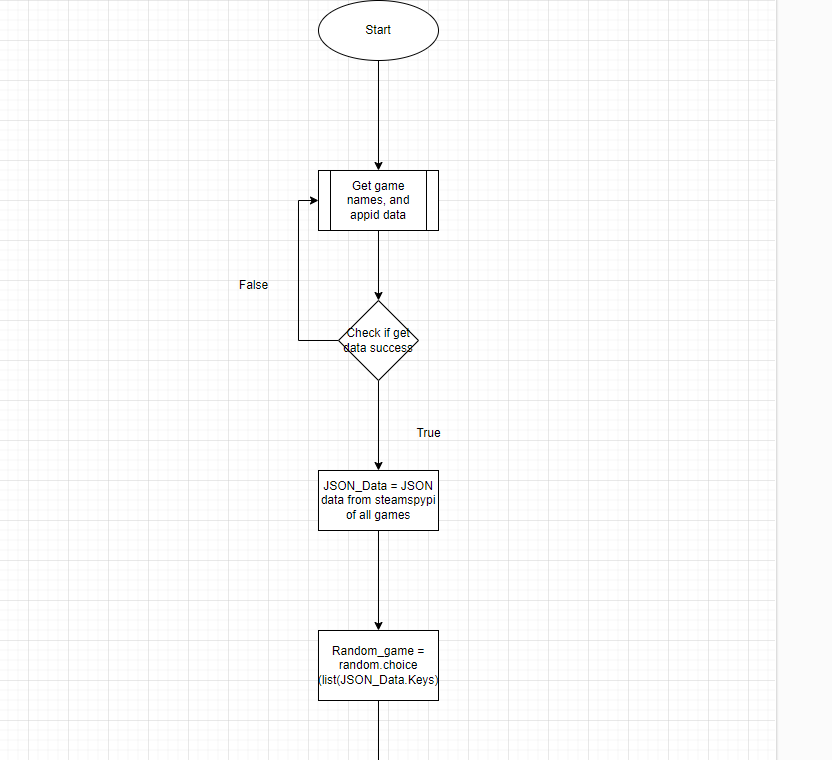
**2.3 Dataflow diagram**

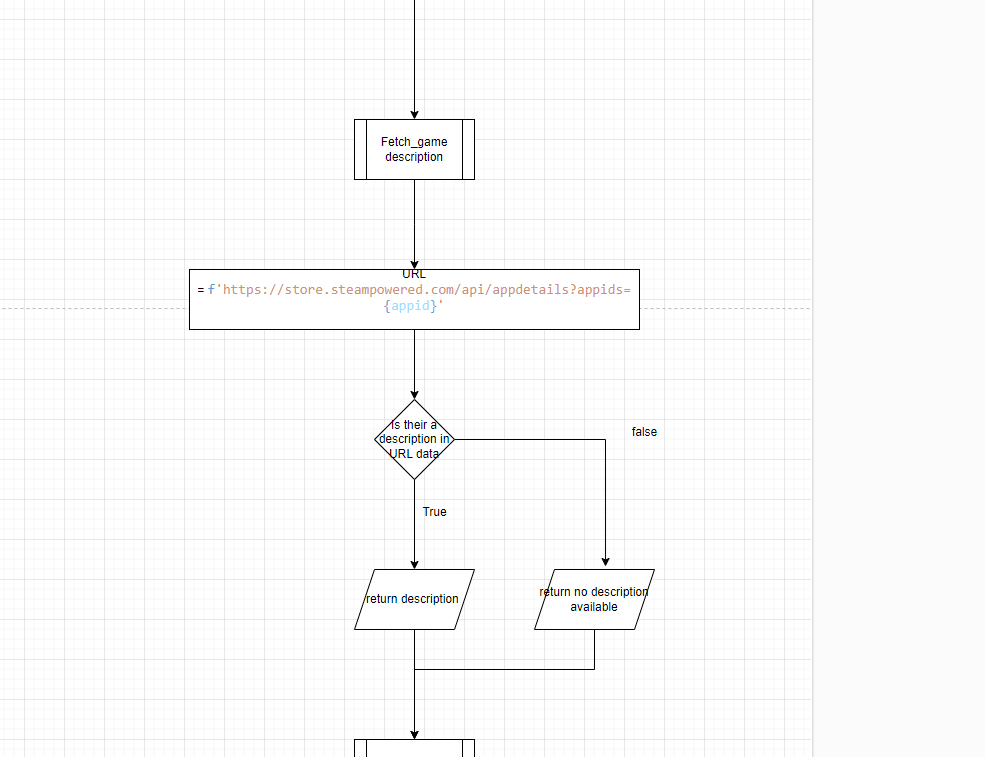
****

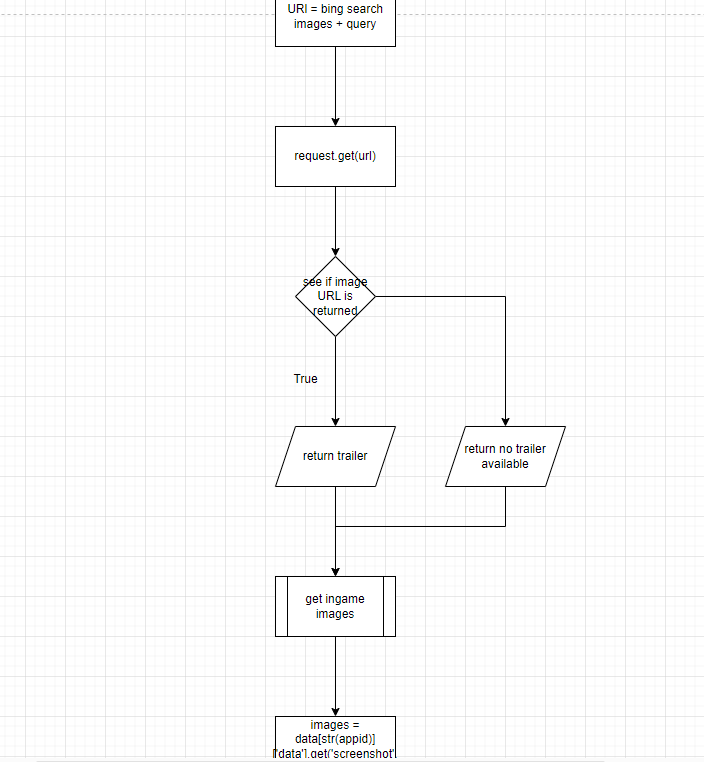
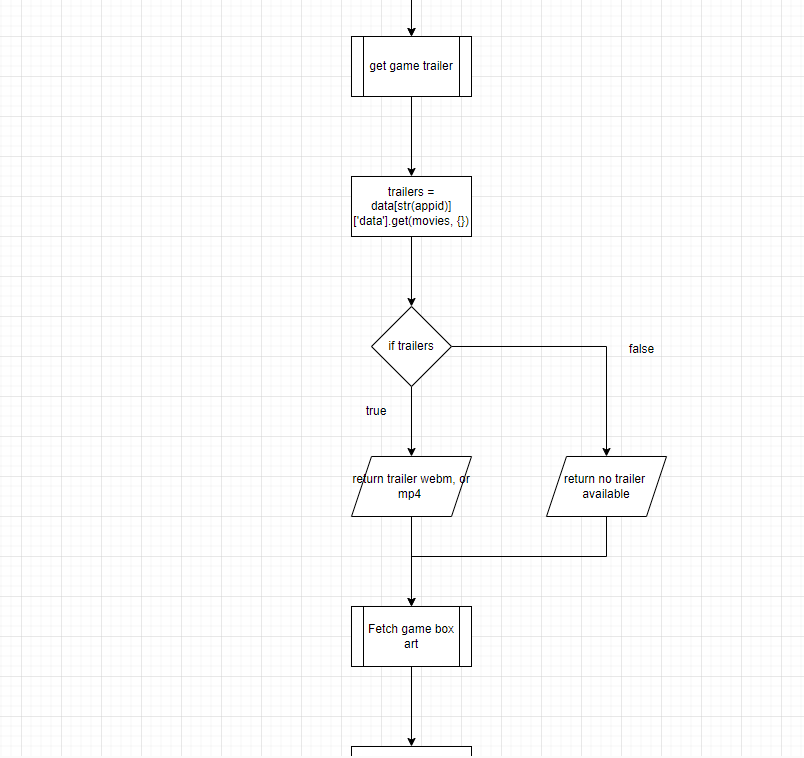
**2.4 Structure Chart**

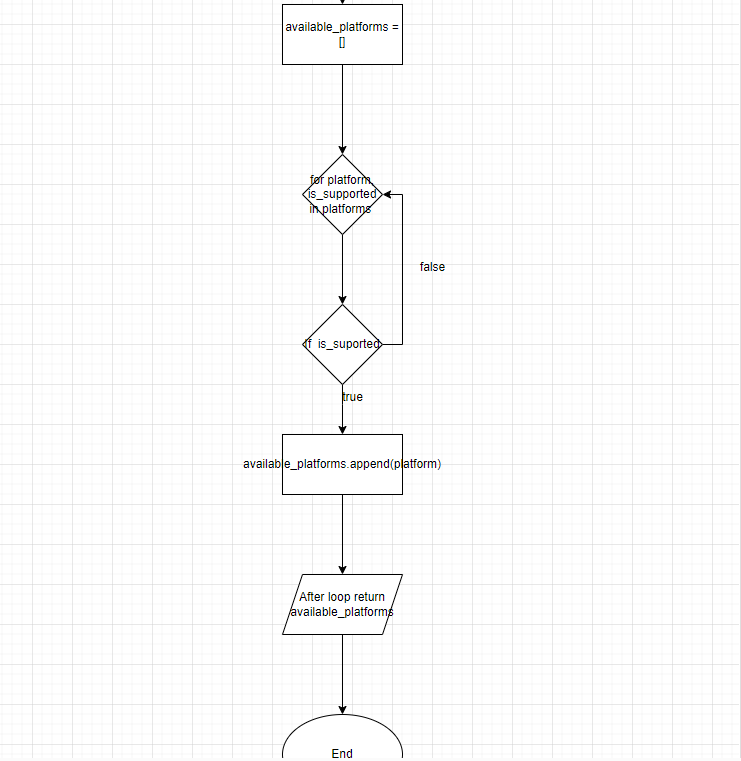
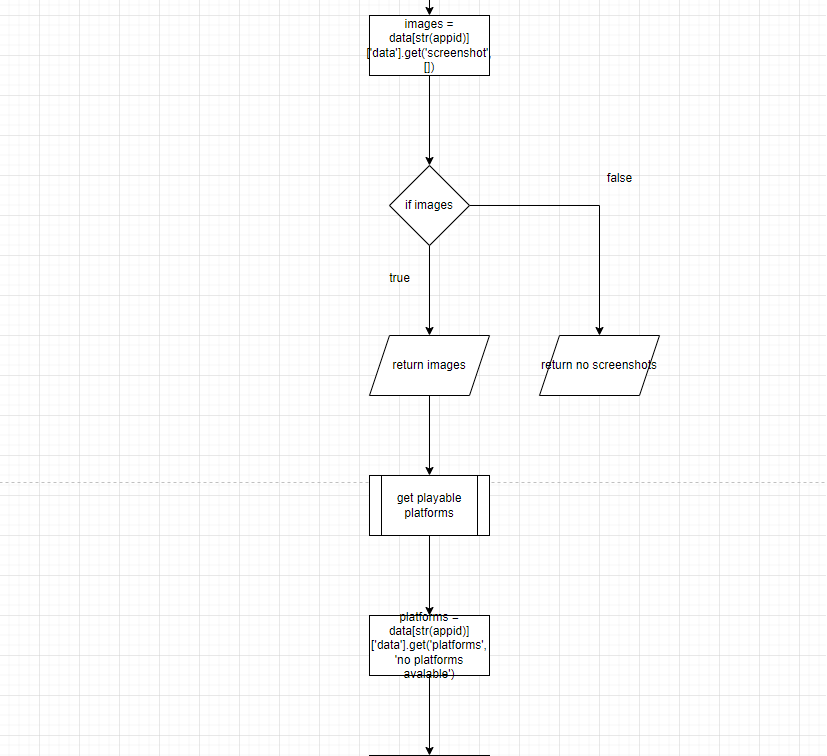
****

**2.5 Algorithms  
2.5.1 flow chart**









**2.5.2 Algorithmic description**

**Fetch\_Game\_Trailer(appid) function:**

1: creates variable trailers to store the specified area movies within the data dictionary of the random game

2: uses a if statement that says if the data is in there proceed

3: if the variable has data then set a new variable trailer and take trailers[0]

4: create trailer\_url variable and asign it a WEBM video or a MP4 video URL

5 return trailer\_url

6: IF trailers don’t have anything then return No trailer available

**Fetch\_game\_Image(appid) function:**

**1:** create variable query with game\_name + “box art image”

**2:** set up search URL through creating URL variable and formatting the variable query to the Bing search

3: use a variable header to disguise the request URL to breach the blocker to say the request is form either chrome, Mozilla, or safari.

4: If statement to see if the request connected

5: parse the text from HTML

6: compile all the Text with ‘class=’iusc’’ to image\_results

7: return the first image if there is something with image\_results

8: return first\_image url

Fetch\_game\_source\_data() function

1: x = request.get(<https://steamspy.com/api.php?request=all>) – url to collect game name, appid, developer

2: jsaon\_data = json.load(x.text) converting te data to json creating dictionaries for the data inside one big dictionary

3: random\_game = random.choice(list(json\_data.keys())) using this to chose the random game I plan to find information about

Fetch\_game\_description(appid): function

1: set variable url to be formatted with the appid and steamDB

2: response = request.get(url) sending a request out to get the data

3: if response.status\_code == 200 checking if the request got connected successfully

4: data = response.json() changing the format of the data to json

5: if data[str(appid)][‘success’] checking the data actually came back within the request

6: required\_description = data[str(appid)][‘data’].get(‘detailed description’, ‘no description available’)

7: soup = beautifulsoup(required\_description, ‘HTML parser’) pare the html to extract the text content

8: clean\_description = soup.get\_text(separator=' ').strip() removing excessive space between words

9: return clean\_description

Find\_platforms(appid) function:

1: platforms = data[str(appid)][‘data’].get(‘platforms’, {}) getting each possible platform with the true or false value

2: supported platforms = [] creating an empty list for all the true values

3: for platform, is\_supported in platforms.item() used to iterate over the platforms

4: if is\_supported checking if the variable has an attribute of true

5: supported\_platforms.append(platform) appending all true values to the list

6: return ‘,’.join(supported\_plaforms) sending the list back as a string of text

**2.6 Gantt charts**

**3.1 Development Log**

|  |
| --- |
| **Development entry 1** |
| Date 9th May |
| Week 2 |
|  |
| **Summary of work done;** I have found an achievable idea of solving gamer block through GamersGateway. What has been achieved is that I was able to figure out how I was going to find the games through an API called Steam spy which enables me to run line of code to randomly select a game. |
| **Challenges And Solutions:** The challenges I faced was that when I found the problem I was confused on how I was going to find APIs for the game but that was easily solved through steamspy API that took a couple days to find. Additionally, the way to return the text I didn’t know what JSON was or request so with YouTube videos I learnt how the request function work and how JSON works. Lastly another challenge is that I was originally going to import the API and use its functions, but I found that using the website URL I can just model my code to return the information myself and it helps through not being limited to the Steam spy API functions. It also enables the information to be passed around the code more easily as some functions might not interlink between python and cause future errors |
| **Milestones achieved:** found the API that will assist in getting the valuable information related to the app that will be persistently used in the future as it is easily usable and manoeuvrable |
|  |
| **Development entry 2** |
| Date 13th May |
| Week 3 |
|  |
| **Summary of work done:** Setting up a website template using Flask that in future will display all of the information and functions that GamersGateway will show to the user, such as search the specific game you want, search through genres or just clicking a button and returning a random game. |
| **Challenges and solutions:** A major challenge that I had is that I didn’t know how to code using HTML as I hadn’t of used it in a long time which ment I had to go and watch videos on how HTML works. Furthermore, as I learnt what HTML does, I didn’t know how to interpret it to my code so I kept getting errors as I couldn’t get the buttons to work. The solution was that I never created a route in my main code which is key as when the button is pressed it needs a function in the main file to call to as any accessibility features are to be defined within it. |
| **Milestones Achieved:** I have now been able to start displaying the information regarding the Games been generated which was key as I now have learnt how to code in HTML making development later a lot easier when using HTML within the project. |
|  |
| Development entry 3 |
| Date 15th May |
| Week 3 |
|  |
| **Summary:** set up a function to return the first image that comes up through a Bing search request. I have additionally made it so the user can see the image the game name, and developer. Additionally, as stated in the last development log I was going to implement a search function for searching for games through their name, but I decided against it as it was too similar toz steam. |
| **Challenges and solutions:** I had faced many challenges that I was initially trying to connect to steam DB and brake through their web scraping protection through masking where request was coming from. But I then altered the search to be through Bing and masked the search as a google chrome to successfully break through. Additionally, the challenge of what image arose because of the thousands of results that would pop up I decided to go through and just chose the first image. Lastly the challenge of the image being reliable and if it is about the game, I couldn’t solve the problem, but it was just a possibility as a few times the image can be something different. |
| **Milestones Achieve:** The milestones achieved are that I have successfully returned the cover art of the game through a Bing search request and pull method. I have also managed to display the image at a reasonable size along with the developer and game name. |
|  |
| Development entry 4 |
| Date 29th May |
| Week 7 |
|  |
| **Summary:** I have created a function that returns the trailer and fixed the image positioning on what size it is on the screen I have also figured out how to return the description. I have also found the steamDB API that allows me to return all specified information on the app that I need |
| **Challenges and solutions**: the challenges that I faced were finding the areas that I can get all this information from and how to access it for each game, solved through researching the steam DB and finding API documents of HTML. Furthermore, an additional challenge I faced is the |
| **Milestones:** I have found the access point for the trailer and additionally I have managed to send a request and return the trailer and display on the website. Additionally, I have gotten the game description to be displayed. Lastly, I have found a website that will be used throughout the whole development for getting all the information in the future about the app. |
|  |
| Development entry 5 |
| Date June 18th |
| Week 8 |
|  |
| Summary: Improving quality of through fixing the video sizing, additionally I have added a show/ hide description feature. I have also fixed the description showing all the HTML code. |
| Challenges and solutions: I faced challenges of not knowing how to get the button to make the text be hidden or reappear and additional challenge was I struggled to implement beautiful soup to remove the HTML from the description. |
| Milestones: nothing really to big was achieved it was just the problem of having HTML in the description took to much time than it should. |
|  |
| Development entry 6 |
| Date 25th June |
| Week 9 |
|  |
| Summary: I have added a carousel that loops through showing pictures of what the gameplay is. Additionally, I have got it to only display the platforms the game is on. I have fixed a couple of bugs and tiddy up the website to look a bit nicer and more appeasing |
| Challenges and solutions: Challenges I faced where that I couldn’t get the trailer to view within the carousel, so I had to just place it outside and additionally the viewing of the carousel is a bit bugging when it changes slide it glitches with the screen for a second. [I was additionally going to include the price, but it all displayed differently for each app I couldn’t get it working so I had to scratch the idea. I additionally realised a genre search feature was working partially but I couldn’t get the end point URL for each genre where all its games are stored and because the way attain each bit of data makes it difficult to implement a genre search because I don’t pull all the data about every game and store it I only request when needed. |
| Milestones achieved: I created a high functioning carousel that displays relevant pictures about the game I have also fixed up the platforms. Additionally, I finished finalizing some of the features |

**4.1 Testing table**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test ID** | **Category** | **Test case description** | **Input to provide** | **Expected output** | **Actual output** | **Pass/Fail** | **Action taken** |
| Test 1 | Data check | Testing if the steam DB connection was successful | APPID | If successful Data, such as the description and all the data Steam has about a game | All data about the game seam has and the description | Pass | N/A |
| Test 2 | Logic check | Testing to see if the genre input requires specific wording | Search genre such as Action, action aCtIoN | Display a game with related genre | Error as action, aCtIoN didn’t work because of capital letters | Fail | I created a drop-down menu selection that will allow the user chose genre so specific spelling is overwritten and eliminates logic errors |
| Test 3 | Request success check | Testing to see if connection Bing search connection is successful | Query + URL | Successfully connected | Failed to connect | Fail | Solution was to create a header to mask the requesting point as Mozilla, Chrome or safari |
| Test 4 | Data check | Testing if steam SPypi returned the data | Request all data | Success | Success | Pass | N/A |
| Test 5 | Testing data return | Testing if the Bing search returned a URL | Query + URL | A long url in the terminal + Successful | A long URL + Successful in terminal | Pass | N/A |
| Test 6 | Data flow check | Testing to see if each image goes into the carousel | Images URL | All images displayed on the website | All images displayed on the website | pass | N/A |
| Test 7 | Boolean check | Testing to see which platforms the game be played on and to see if it only returns the TRUE value | The Array of all platforms steam provide and the Boolean values | All the platforms with the value true | Every platform that steam can be played on | fail | I had created an array to append all the true values into it through using a for loop to check each value that is true then append it to the empty array |
| Test 8 | Data check | Checking if the trailer returned a URL to be displayed | The APPID | A URL that is either for an MP4 or webm | A URL for a webm or MP4 | Pass | N/A |
| Test 9 | Abnormal data | Checking if the game data is accurate | The game data and steam data | Successful return | Success | Pass | N/A |
| Test 10 | Faulty data | Checking if all the within the steamdb is right that has a value like Boolean or numerical value | Game data, and steam data | Return a success | Success | Pass | N/A |

**5.1 Git hub repository**

**6.1 Project reflection**

the program I have developed is called Gamers gateway it is a small app that allows users to generate a game and it will return the name, developer, platforms available, trailer, description, in game images, and a box art of the image. The aim of this project is to help users that suffer from gamer block and looking for new games to play but don’t know what to play. I intended to achieve a fully functional app that allowed the user to use genres to search or no genre to search for the games and that came back with the price, developer, trailer, images, name, in game images, description, and platforms.

The app was built with mostly success as I was able to build a full functional image slider that can automate the transition between in game images or the user can manually do it. Additionally, the use of getting a trailer video to work and display, the ability to scrape Bing for a box art image and pull it and the ability to hide or show the description.

However, there were challenges that I faced and managed to overcome most of them but due to limited resources some difficulties forced me to rethink the features I was going to implement into the app. These difficulties such as not even knowing how to code in HTML as at the beginning I didn’t understand any of it and how structuring the HTML page worked which led to me in future now being able to confidently use HTML to construct a website through flask. Additionally, not being able to resize trailer borders, removing HTML from a string of text, caused huge time loss as I spent numerous days trying to figure out a solution, causing to me stop developing for a period. Led to me learning how HTML works properly and understanding it more in depth. Through these setbacks led to me having to stop and think about what features I should implement due to limited resources though I was struggling to implement the genre search function and had to compromise with time to instead implement an image slider that automatically works but still has manual function.

The project I have worked on has various flaws that I know could be fixed with a bit of extra time. However, these flaws consist of the overall UI9 I personally am not happy with it as I could have put more effort into making it look good. Features I wished to implement were the genre search as I believed it would be nice to search in a finer area, however due to the resources available it constricted me to being able to do so. The project can be refined through the image slider as the images randomly go back to their original size and it just doesn’t look as flowy or seamless, the layout as well it could be improved, the addition to search with or without genres. Lastly in future these enhancements that I wish to add will make the better through more user accessibility in what games they can see, and additionally just for ease of use.

To sum everything up this project has had a successful development of the term. From this I have learnt how to use request and pull things from web browsers, changing the format of data through JSON, it has further deepened my understanding of how HTML functions and coding using it. In future this will help me through any website I may develop that require me to use HTML.