**Ai Dungeon Game**

**Kamil Marszalek(25794511)**

**Alex Clarke(25662485)**

**Craig Potter(25785913)**

**Jamie Grant(25750898)**

**Lukas Sobolevas(25728866)**

**Alfie Atkinson(25715017)**

**Team Software Engineering**

**18th May 2023**

**Contents Page**

[**Introduction**](#_6sc97gaoebbn) **3**

[**Software Engineering**](#_ykhs4lm9mxo9) **3**

[**Implementation**](#_ettoimxxg3sd) **5**

[**Testing Strategy**](#_vruhfnh2s7w2) **6**

**Release****11**

[**Evaluation**](#_9l9xcqvdjr7b) **11**

[**Conclusion**](#_thakotjaf2sh) **12**

[**Group Work Conclusion**](#_v8lyiq2r4eiw) **12**

[**Artefact and Media Materials**](#_w1981b9xds5s) **14**

# **Introduction**

Our group for this project has decided to make a game that creates an AI-generated story that the player can continue by typing whatever they want within the input box. This would progress the story in many ways depending on what the user has inputted.

In terms of functionality, this game is meant to be similar in concept to a single-player Dungeons & Dragons game where the player decides all their actions that could have many different consequences.

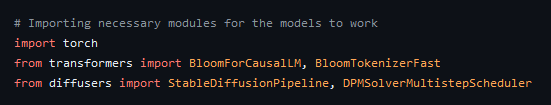
Throughout the report, we will be discussing the ways that we have implemented the game's main functions, as well as the overall process of how we got from the start to the end product.

# **Software Engineering**

The main development methodology used within our group project took an Agile approach; Agile methodology promotes collaboration and flexibility through its iterative development cycles, which makes it an effective method for projects involving multiple people. Agile makes possible faster feedback loops between customer and stakeholder and improves project management.

Agile methodologies employ shorter, more frequent, development cycles known as ‘sprints’ which allow for regular adaptations to the code dependent on customer satisfaction and feedback. This allowed us to meet up often and share with each other what work we had produced and either redirect our workload focus to pressing matters or draw focus towards someone that is in more need to help with their assignment - enhanced collaboration is paramount to Agile methodologies and members of groups that employ it.

For our specific project we were using Slack, which is a website that uses the Scrum Agile Framework. Scrum divided our work into time-restricted iterations to make sure that we were consistently improving on our code and outputting working solutions. Scrum emphasises teamwork, singular accountability and iterative progress, enabling efficient coordination and ensuring that everyone remains aligned and focused on their goals.

The main dependencies in our code come from the implementation of both AI text generation and text-to-image generation. We use Bloom (HuggingFace, n.d., para. “BLOOM”) for text generation and Stable Diffusion (HuggingFace, n.d.2) for image generation. 

We have employed modular, object oriented code to maximise code reusability by encapsulating various core functions into self contained modules that are called at various points throughout the code. In doing so we have improved our time efficiency by not having to rewrite code, and by calling these functions it also saves on space complexity. For a software engineering project, testing code is one of the most important things someone could do; with modular code we can test any singular module at a time by only calling that independently. We are also passing by reference, meaning that the code directly accesses the original data through its encapsulation, resulting in more efficient memory usage than passing by value. In the case of future improvements to the system, modular code and OOP both emphasise reusability through inheritance, which reduces code duplication, and composition which allows objects to be composed of other objects, allowing for flexible design.

As a group, we formed a joint GitHub project that we were included in. GitHub is a web-based platform that uses Git, which ‘provides distributed version control’ (Wikipedia, 2023). The GitHub collaborative project acted as our repository, a database for holding all historical changes and commits to the system, and each individual’s personal computer possesses the role of being a ‘working copy’, or where the updates to the project are first made. GitHub is an excellent form of version control as it is distributed, making it faster to run and less prone to errors. Furthermore, with distributed version control, every user has their own locally stored repository that they can commit to before pushing towards the joint repository. This provides data redundancy and offline access which is useful if GitHub is not working or there are personal issues regarding internet access.

As we are working from GitHub, we are using GitHub Action to publish our artefact.

# **Implementation**

# As a group, we have created a robust AI-driven text-based dungeon crawling game which implements both AI text and image generation, using Bloom and Stable Diffusion (HuggingFace, n.d.3) respectively. We have used CustomTkinter as a frame to communicate our program with users of the system, improving the visual clarity of our system and making it straightforward and intuitive to use. In our coding approach, we decomposed our proposed project into its core features and divided the workload among our peers to work on separately; this reduced the overall workload and stress for each individual member and also helped speed up the rate of produced development cycles as multiple sections can be worked on simultaneously, in parallel to one another.

# As a home for all of our coding efforts, we used a joint GitHub repository, known as a project, where we can all access the current model of our code and also submit our own improvements and additions for everyone to see online. GitHub was the best choice to use for a group project of this size as Git provides version control, allowing users to access historical versions of their code if needed as well as uploading their new code to the repository quickly and in an easy fashion. This distributed version control also accounts for data redundancy and offline access which are important in case any of our work becomes corrupted and prevents any unforeseen setbacks from occurring to our group work.

# We have previously mentioned the benefits of our Agile development methodology, but the main reason we have used this comes into play when in relation to the key libraries and toolsets we used in our time coding. Both Bloom and Stable Diffusion were completely foreign application software to all of our members when we first embarked on this project. The best method of learning any new software is to use it for yourself and to explore the limits of what it can do and what is possible. With Agile Sprints, we can quickly update our previous best practices with the software so that everyone can see the current condition of the software in relation to its working operations. Comparing historical versions of code to the current working model, with help from reading tool-tips and user guides, provided us with the guidance needed to better understand and develop our knowledge of the libraries.

# One of the bigger issues we encountered was to do with a proposed image generation model, known as Stable Diffusion. As the hardware we were trying to run this model on was not particularly strong, we could not provide the needed resources for its generation that uses multi-GPUs/fp16. (PyPi, 2023), therefore we had to install the dependency ‘accelerate’ to abstract the boilerplate code so that it would run. However, even with this fix, StableDiffusion was still having trouble generating images without running into memory issues on the user’s PC. To combat this, we explored multiple solutions, one of which was the use of another image model called Openjourney, however after a lot of testing we ultimately decided to continue with the use of Stable Diffusion while lowering the image size to 256x256, this makes it run on our home hardware while also producing a more retro look to the game.

One of the main programming techniques we have used in development was object oriented programming.The program uses objects with encapsulation, abstraction and overridden functions. In our program, we implemented objects and functions to group data together to provide a more effective way of understanding and reading our code. This is important within early stages of development as not using these would mean repeating or copying unnecessary code, which would only set us back with time and otherwise would be more difficult for group members to understand and build off of each other’s code.

Another feature of object oriented programming we have implemented was abstraction which is the technique of removing unnecessary data or characters to produce working code showing only the essential information of an object. Abstraction provides a means of viewing functions or algorithms easier within our program and strays away from repeating code but rather reusing code. This technique has helped optimise our program and made it more efficient, needing less memory to run our code.

The second main programming technique we have used in development is modular design. Modular design is the programming concept for separating data into functions. We have integrated functions when defining and creating all of our classes, giving each class. Modular programming was an essential part of our development as it provided with many advantages, such as the ability to test different functions outputs for errors and not having the hassle of sifting through large amounts of data that isn’t categorised into functions. When programming it was also helpful with searching for specific sections of code later on and saved a lot of time.

# **Testing Strategy**

One of the major testing strategies used within the development of our artefact was performing Unit Tests. Unit testing is a process where the developer ensures that their individual additions within the software are working as they’re intended to, which also helps with the development in the future as it prevents many major issues developing and taking less time to fix if they do. Within our project we carried out unit testing manually as this made more sense to us because the software itself doesn’t contain too much code where automatic unit testing would be necessary.

As a team we split up the workload by working on different sections of the software individually. This meant that we couldn’t test the whole software at once until we put it all together, which gave us the only option of performing unit tests within our own sections until we did put it all together. We do believe that unit testing is a useful method of testing as it doesn’t take much time for the tests to be carried out and lets us prevent and overcome issues that do arise a lot quicker than without unit testing. It also made putting our sections together quicker which meant another strategy of testing could be carried out as soon as it was put together.

| ID | Test | Expected Outcome | Actual Outcome | Screenshot |
| --- | --- | --- | --- | --- |
| 1 | Testing message input and seeing if output box works | The area where a generated message should be shown, it shows user input instead to see if user input is being taken correctly. | User inputted message is shown in the output area correctly. |  |
| 2 | Testing if the input is edited correctly | The user input should be edited, and a full stop should be added if no punctuation is detected. | The user message is corrected by adding punctuation. |  |
| 3 | Testing if the input is edited correctly | The user input should not be edited, and no punctuation mark should be added. | No punctuation mark is added. |  |
| 4 | Testing the text output box | Testing that the new input by the user replaces the previous message on the text display window. | The text is replaced by the new user input. |  |
| 5 | Testing the user input box | The user input box gets cleared every time the user inputs a new message | User input box gets cleared every time the user ‘sends’ the message (by clicking enter or arrow in the text box). |  |

Our next stage of testing was doing Integration Testing. Once our unit tests were complete, that meant we could move on to start implementing our different sections into one and start testing if our sections work and operate as they should once put all together. One of the expected outcomes we looked at is if the UI elements work correctly with the AI text generation and if it is shown within the UI. This is one of the more important aspects that we needed to get right as the user will look and use those elements every time they want to continue with the generated story.

Our integration testing mostly worked by testing the software as in the eyes of a user and what they could and would possibly do while using our AI dungeon software. An example of a test we carried out would be typing out a prompt and seeing if the generated AI text is being outputted to the screen correctly and if there is interference with the UI where some of the generated text may not be visible and hidden by some of the UI elements. This was an issue we discovered while testing which proves that integration testing is important and something that may not get discovered while carrying out unit tests on their own.

| ID | Test | Expected Outcome | Actual Outcome | Screenshot |
| --- | --- | --- | --- | --- |
| 1 | Testing if input box works with text generation | The text input box works with AI generation and input is used for generating a story. | The text input box works with the text AI generation and is given as a prompt to the AI. |  |
| 2 | Testing if the input box works with the image generation. | The text given by the user should affect how the image is generated. | The image is generated from the user input and displays a correct image most of the time. |  |
| 2.1 | Testing if a placeholder image is shown the first time the program is turned on. | Every time you boot up the program, a placeholder image should be shown and not the previous generated images. | A placeholder image is shown every time the program is turned on. |  |
| 2.2 | Seeing if the previous generated image is replaced by the new generated image | Every time you write in a new prompt, the previous generated image gets replaced by the new generated image. | The previous generated image is replaced by the new generated image. | Before After |
| 3 | Testing if the output box can be read by the text to speech function | The text to speech reads the output box and can be used every time a new message is generated | The text can be read by the tts every time a new message is generated. |  |

Once Integration Testing finished and all the added sections were tested, we moved onto some non-functional testing starting with performance testing. We decided to do performance testing as it is important to see if generating AI text for a long time hinders the performance of the software as well as if any unexpected crashes or errors occur within a prolonged usage of the software. Testing the performance of the software is important and shouldn’t be left untested as in a scenario where a user is generating a story that they are into and the program crashes, they may get unmotivated to use the software as they know something like that could happen again. This could then make the user leave a bad review or tell people around them that the program is not worth using leading to less people using our software, and therefore we chose to do some performance testing.

As our software doesn’t involve any online interaction and doesn’t make the user give away any personal information or make the user create an account and log in, we decided on skipping security testing. Our software also doesn’t save any user prompts or generation done for those prompts so even information like that cannot be retrieved, meaning the software is quite secure and wouldn’t involve much or even any testing to begin with if we did decide to do it.

We did decide to do accessibility testing however, and although there isn’t much we would need to test as the UI is quite minimal, we thought it is still quite important as many different users with unique needs could use this software and may not be accessible if testing isn’t done to cater towards that group of people. Accessibility testing consists of checking if different colours clash for different colour blind types and if the text displayed on the screen is readable without having to make different options for the specific colour blind types.

| ID | Test | Expected Outcome | Actual Outcome | Screenshot |
| --- | --- | --- | --- | --- |
| 1 | Looking if different colour blind types can use the program properly without struggling to see text or other UI elements. | Users have no problem identifying different UI elements and the text presented to them. | The UI itself seems to be usable with different colour blind types, according to a colour blind simulator If the text isn’t readable then the user can use the TTS function. Although different images will look different depending on the colour they are generated in. |  |
| 1.1 | Testing Green-Weak/Deuteranomaly colour blindness | Users with Deuteranomaly colour blindness can still use the program without issue. | Based on the colour blindness simulator, the program seems to still be usable, although images look different as expected. |  |
| 1.2 | Testing Red-Weak/Protanomaly colour blindness | Users with Protanomaly colour blindness can still use the program with no issue. | Based on the colour blindness simulator, the program seems to still be usable, images look different as expected. |  |
| 1.3 | Testing Blue-Weak/Tritanomaly | Users with Tritanomaly colour blindness can still use the program with no issue. | Based on the colour blindness simulator, the program seems to still be usable, images look different as expected. |  |
| 1.4 | Testing Monochromacy/Achromatopsia | Users with Achromatopsia colour blindness can still use the program with no issue. | Based on the colour blindness simulator, the program seems to still be usable, images look different as expected. |  |
| 2 | Testing Performance | Testing if the program can be run multiple times without an effect in performance as well as how the program runs during runtime. | The program can be run multiple times with no noticeable issues, the performance of the program and how fast it loads images is impacted by how powerful your NVIDIA GPU is. |  |

The last stage of testing was doing some acceptance testing, where we looked at the finished software to check if it is ready for release at its current stage. Our acceptance testing consisted of checking if the code is working as it is supposed to with the different modules in our software and if it all computes logically. We once again did some similar testing as we did in the previous stages, one of the tests being if the generated story and images by the AI are still outputted to the user correctly and if the images resemble in any way to the generated story and making necessary tweaks to the final product for release.

**Release**

We released the first version (Version 1.1) of our videogame on itch.io which is quite a large platform for indie and tabletop rpg games. Itch.io allows users to publish, sell and buy indie video games with revenue splits from 0% to 30% off of each purchase. It’s a great venue for new and small groups of developers like us to share and publish our products for free as there was no developer fee when putting it on the Itch.io page. Putting our product onto a platform like steam would be good but it has flaws for small groups of developers, such as our videogame might be shadowed by larger triple A games making it harder to find an audience for us. Although, we could say the same for Itch.io and it being shadowed by larger indie games on their page so there are pros and cons for both. Another reason why itch.io was a great venue for our product, was their “bundles” feature for releasing a variety of games with different developers at a discounted price. These bundles can help promote our game to a larger audience and provide support among game developers.

One of the precautions we put in-place to lower the number of issues on release was quality assurance testing before release. This was a crucial part of the development process as we needed to focus on removing errors in our code before our game was launched with serious bugs. We did this by using static analysis tools such as compilers to check for syntax errors during the process of our development stage. It also helped with optimising and making our code run more efficiently.

After the release of our product we planned on communicating with our users to help retrieve feedback for future updates to fix any issues with our game or to change any algorithms. This was needed as compilers only detect errors when not inspecting the code, and so it wont detect dynamic semantic errors when the code is executed. So consistent updates to our game will be needed.

After we received some feedback, we decided to work on a second release (Version 2.0) of the game, this game includes features asked for by the players such as an interactive UI rather than a console app and TTS for the visually impaired. We implemented these features and released the second version of the game shortly after.

# **Evaluation**

The team has worked hard to produce a quality game that would give players an accessible story driven experience and meet the brief requirements of utilising existing Ai models. A fair attempt has been made at this and many positives can be drawn from our released product however so can many improvements be made.

From our initial project proposal, we have deviated very little. Some design choices were made to better suit the program we were making that were not mentioned in the proposal, such as the Ui which functions very well. Given more time we could have altered the Ui to be more immersive and designed something more aesthetically pleasing but that is not to say it was left unfinished, it is clean and concise and works very well. Furthermore, we had originally stated that the program would use Tortoise TTS to facilitate text to speech, this however is no longer the case. We faced many difficulties Tortoise to work and then later abandoned incorporating it into our program all together. If we had more time to work on the project, I would use the time to research alternatives and work on integrating one of them into our current build. The most likely text to speech model that the program would adopt in future would be Google Text-to-Speech API. This would add another dimension to our game, drawing in our player into a deeper story experience. Utilising text to speech would also satisfy our brief of interactive storytelling using off the shelf models as well as making the game more accessible. Expanding on accessibility more, we would have liked for the program to be able to recognise speech and use this as the player’s input. This would have made the game completely accessible to those who are visually impaired and made a brilliant addition to our project.

The current state of the game is stable and functional, though not to the standard we would have liked when planning this project earlier in the year. The code we have produced is to an acceptable standard and shows off the expertise of the team. Despite this the program itself is slow and clunky and should the deadline not be fast approaching there are touch ups that should be made to the code. More footnotes and comments would have also made the readability of the code we’ve produced much better. This would have been especially beneficial to the team considering that we all worked on different portions of the code and “stitched” it together after all the segments were finished. A lot of time could have been saved had the group written more readable code and noted important requirements such as the imports that must be used for the code to work. Being a team, we could share this information as and when needed but to a peer reviewer this caused a headache. Not all is bad however, largely the code holds up very well and individuals in the group have produced some very robust code.

# **Conclusion**

To conclude we believe that our game takes a very interesting approach in the world of storytelling games and creates a very dynamic and constantly evolving new plot, with the use of Ai technology, that will keep players captivated.

# **Group Work Conclusion**

These last weeks of the project’s development have seen the team working very hard to complete the program we dictated in our proposal. Despite the challenges faced it has been delightful to work on this project with my peers and the expertise they have brought to the project has made the work all the more enjoyable. I am proud of the work we have accomplished although it is clear to me now that the group would have benefited from some improved teamwork skills. Most notable would be our communication skills. Without a doubt had the team communicated its ideas more freely and openly work would have run much smoother. A dozen instances where members were unsure as to what work had been done or what work was to be done and by whom would likely not have slowed us down had before jumping in we planned the workload. This is something that, if working on a similar project in future, I would suggest happening in weekly meetings. Paired with a schedule, the project could have been broken down into more manageable chunks and assigned a team member and duration. On the other hand, the team was very well suited to the more flexible style we adopted when working on the project and managed extremely well, even if we might just have made everything harder for ourselves.

An example of where a more structured approach may have made lighter work for us came very early in the project’s development. We held a meeting to discuss the existing interactive models we wanted to incorporate into our program. Unfortunately, some members of the team could not attend the meeting and were unaware of the models we had chosen going forward. Some work was already done unbeknown to those who had attended that utilised a model we had disregarded. Fortunately, the model was bloom, a model for image generation, and caused the team no trouble adopting over the alternatives we had deliberated. The possibility for this to create more work for us however was worrying and would not have happened should we have had better communication and organisation.

Though our group elected to have no project leader, it is worth mentioning the leadership qualities that Alex gave us. He was a constant source of optimism and stepped up to offer guidance on numerous occasions. I like to think he filled the role a leader would have in motivating our group and managing our project. Likewise, Alife and Lukas provided a great deal of knowledge of Python and their patience in combination with their skill has made them a valuable asset to the team. The cohesion of the team was second to none and has made the work we shared very rewarding. Despite our shortcomings, there doesn’t strike me one moment where the team was not on the same page throughout the development process.

| Name | StudentID | Percentage |
| --- | --- | --- |
| Kamil Marszalek | 25794511 | 16.6% |
| Alex Clarke | 25662485 | 16.6% |
| Craig Potter | 25785913 | 16.6% |
| Jamie Grant | 25750898 | 16.6% |
| Lukas Sobolevas | 25728866 | 16.6% |
| Alfie Atkinson | 25715017 | 16.6% |

# **Artefact and Media Materials (Links)**

# References (no limit)

HuggingFace. (n.d.) *Transformers* [online] Available at: <https://huggingface.co/docs/transformers/v4.29.1/en/model_doc/bloom#transformers.BloomForCausalLM> [Accessed 12 May. 2023]

HuggingFace. (n.d.2) *Stable diffusion pipelines* [online] Available at: <https://huggingface.co/docs/diffusers/api/pipelines/stable_diffusion/overview> [Accessed 12 May. 2023]

HuggingFace. (n.d.3) *Openjourney* [online] Available at:<https://huggingface.co/prompthero/openjourney> [Accessed 13 May. 2023]

PyPi. (2023). *Accelerate* [online] Available at:<https://pypi.org/project/accelerate/> [Accessed 13 May. 2023]

Wikipedia. (2023) *GitHub* [online] Available at: <https://en.wikipedia.org/wiki/GitHub> [Accessed 12 May. 2023]