A System to Create Visual Novels from Film Scenes via Mechanical Turk

A dissertation submitted in partial fulfilment of

The requirement for the degree of

MASTER OF SCIENCE in Software Development

in

The Queen's University Belfast

By Aoibheann Maye

2nd October 2020

Acknowledgements

I firstly would like to thank my project supervisor Dr John Bustard. His guidance, support and everlasting positivity proved crucial to this time.

I would also like to thank my family, who constantly provided words of encouragement during hours of stress.

Abstract

This project revolves around investigating the answer behind one important, albeit broad question: what properties in film or television scenes are integral to conveying their meanings and emotions effectively? Beginning with gathering vast amounts of film/television scenes, this question diverts into two ambitious objectives: at a high level to solve and capture the multitude of states in the scene which are integral to conveying the meaning and emotion, and at a lower level, prioritising the specific labelling to try to replicate the end result in visual novel form. Amazon's Mechanical Turk was responsible for the labelling tasks intrinsic to the project.

Contents

List of Figures

List of Tables

0.1 Introduction

The following report documents the execution and completion of the CSC7058 Individual Software Development Project based on the creation of a visual novel game which can convey the same message and impact of that from a film or TV show. This section gives a brief introduction to the ensuing chapters and their contents.

Chapter One provides a overview of the original specifications of the project in addition to justification pertaining to any changes that were made over the course of the life cycle of the project. The current problem domain and the target audience are investigated and evaluated.

Chapter Two follows on from where Chapter One left off. The proposed solution is detailed along with the development strategy and its corresponding design details.

Chapter Three comprises of the analysis of requirements throughout the life cycle, beginning with requirement elicitation process which was completed retrospectively.

Chapter Four describes the design blueprint for the proposed system, highlighting the designs for the user interface and all components in the system.

Chapter Five outlines the main functions the software performs combined with methodology applied in the projects development. This chapter includes the specification and justification of the testing mechanisms utilised.

Finally, Chapter Six concludes the dissertation with a thorough evaluation of the project in its entirety. This includes the success of the project dependent of the criteria described in Chapter Three as well as an evaluation of any software and languages used to complete the project. The final part encompasses the opportunities for future work on the project and the identification of the most, and least, successful sections of the project.

Chapter 1

Problem Specification and Project Plan

1.1 Analysis of the Problem Domain

From as early as 1888, film and, from 1927, television has provided a cinematic experience where you can relate to characters and feel their plights and successes as if you were the characters themselves. The reiterating question throughout this project is by what methods do film/television reveal the emotions or meanings behind every scene so effectively. Incomplete answers have been acknowledged through focusing on one factor alone, such as fixating solely on the story writing like Save The Cat! books and website interpretations.[7] Save the Cat! by Blake Snyder is a popular book series focused on screenwriting, which formalises screenplays into 15 beats (and further into 40) which should be present in all screenplays regardless of theme. The title, Save the Cat, originated from the idea that the protagonist in a film tends to do a good deed the first time we meet him i.e. save a cat making the audience more likely to root for him. Save The Cat! creates beat sheets which are a form of outlining used to map out a story, often utilised by screen writers and authors alike and are made up of short bullet points, which represent the main pivotal or emotional elements in your story.[8] This concept that the construction of stories in film and television can be based on a particular structure or formula introduces the idea that this formula can be formalised.

Most stories in film follow a similar path: that a personality/world view/culture is validated through multiple testing situations provided by the threatening personality/world view/culture, and either ultimately triumphing or failing if a tragedy or presenting a socially unacceptable connotation which should not be glorified. However the story alone does not make up the whole experience in cinematography: the occupation, life situation, typecast actors, actors facial hair, actors resting facial expression, era, nationality, language, attire/style, locations, seasons, music, sound effects, lighting, props, camera angles, fonts, colour scheme all contribute to intensifying the subconscious associations of the two positions and are all intrinsic to conveying meaning and emotion. In particular, a character's personality proves to reveal their

status or legitimacy within the story and are usually unveiled through demonstrating situations in which they have important choices or obstacles to overcome. Depending on the outcome of these choices/obstacles, the film reveals how the creators wish the characters to be viewed in terms of their status and legitimacy: if the character falls at one obstacle but lifts themselves back up to ultimately triumph, that personality is validated and they are placed in the hero typecast as the most legitimate and highest status character.

Collections of personality data for fictional characters is fairly common on the internet, but none as varied as that of the Open-Source Psychometrics Project.[9] This website has a quiz entitled 'Which Character' Personality Quiz which asks the user thirty question where they rate themselves on a scale of personality traits: for example, sceptical or spiritual, where 100% sceptical means 0% spiritual. Combining the results, the quiz presents the user with their personality traits in graphic form along with a full personality match list of fictional characters. Each character has their own set of personality traits which can be accessed, given a full personality spectrum. Another important source of personality information for fictional characters is the Myers Briggs Type Indicator, which attempts to assign four categories: introversion or extraversion, sensing or intuition, thinking or feeling, judging or perceiving to every person. There are multiple sources where people have determined character's Myers Briggs types, and collated them in an easy to use manner. One of these is 'The Myers-Briggs Personality Types of 325 Fictional Characters' by Susan Storm.[10] This provides a crucial source of information which could answer how personalities are validated or not in film and television.

Since visual novels were first created, they have been adapted into anime and vice versa, but there are very few, if any, visual novels established from scenes from film or television. The appeal to create something of this manner is twofold: firstly, breaking down, labelling and parameterising the metadata from scenes would allow a dictionary or library of factors which could be accessed by filmmakers, writers and game developers alike and begins to answer the all-important question posed by this project. Secondly, successfully conveying the same underlying messages or feelings synonymous with a scene in a visual novel format highlights something that has not been done before.

The visual novel genre falls somewhere on the spectrum between interactive books and video games and does not involve the typical gameplay as other video games would, in fact the amount of gameplay is minimal. Instead they are based around lots of text, usually in the form of dialogue and clickable content, for example, choosing an option from a list.[1] In the majority of examples, there is often a set background image for each individual scene setting as they tend to focus primarily on the narrative itself. Originating in Japan, the graphics are generally anime characters which act as static or sprite-based visuals with the addition of background music and sound effects. [11] The prevailing theme for visual novel narratives is to play as the protagonist and go through the story as if you were in their place. Many visual

novels have multiple storylines and so multiple available endings which are dependent on the choices made throughout the narrative and the paths you choose to go down. According to Japanese nomenclature, there are two distinct categories of visual novels: first, the traditional NVL story (derived from "novel") which focuses on the narration of the story with very little opportunity for interactive decisions, and second, adventure games, or AVGs, which generally have much more interactivity including problem solving and choices.

Figure 1.1: Examples portraying the two types of visual novels, AVG and NVL respectively.[1][2]

1.2 Project Objectives & Specifications

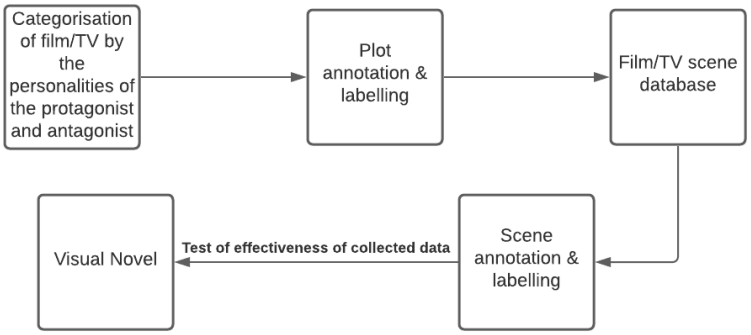


Figure 1.2: Visual summary of project objectives and structure.

Figure 1.2 provides a visual summary of the project objectives and the arrows portray how these parts could ultimately fit together to form a complete system. The scope of this project is fairly ambitious as it covers a lot of different factors and the goal is to produce the minimal viable product in order to see if the grand version is going to work or not. Keeping feasibility in mind at all times was important: for one person to answer the questions on how to replicate the emotions/meanings of a story in full is grandiose but focusing directly on the very literal representations and detail of a scene begins to provide some answers. There are four main objectives, which will be discussed in more detail later: the categorisation of film/TV by the personalities of the protagonist and antagonist, the annotation and labelling of the plot, the collection and creation of a film/TV scene database, and finally the annotation and labelling

of scenes. The final step is to evaluate the effectiveness of the collected data in a visual novel format. A more detailed visual summary is outlined in Figure 1.3. This project could revolutionise the future of game development and production.

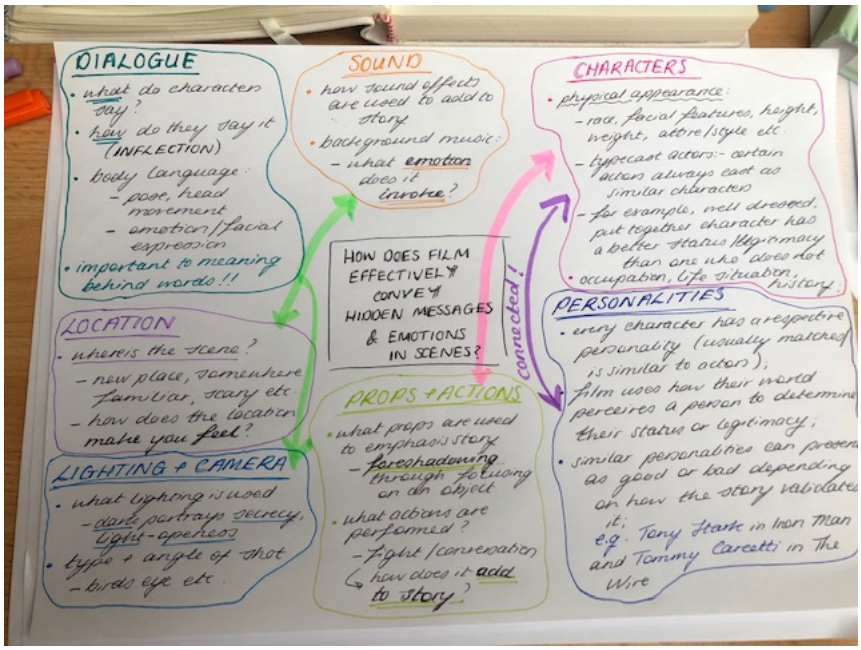


Figure 1.3: Hand drawn diagram with main points which, when combined, were thought to aid effective conveying of hidden messages and emotions in film. This diagram can be seen in larger form in Appendix A.1.

Any difficulty in this project arises from the set timeframe of the project: creating fully functional personality maps and labelling every property mentioned in Section 1.1 exhaustively within the timeframe was never the focus of the project. Instead, the project aims to answer the questions on how one would go about answering the question posed at the beginning. This includes assessing feasibility and prioritising tasks to obtain the best answers and results possible. The section which had the highest priority and was so focused on were creating a labelling tool to gain insights into the universe of film/television, as well as a secondary output of creating a simple visual novel representation to test the effectiveness of the labelled data. Treating this project as a set of experiments, the success is getting quality information, not necessarily getting a product to work. Amazon's Mechanical Turk is used as the labelling tool thus some trial and error will be involved in procuring quality metadata from the workers. The aim is to see if this process will be effective on a small scale, so in the future the project could be repeated for a less minimal project with the addition of some of the other elements mentioned earlier. The limitations MTurk presented made striking a balance between the amount of information labelled and its value to recreating what is important in the story a crucial factor to think about throughout the project.

1.2.1 Categorisation of Film/TV by the Personalities of the Protagonist and Antagonist

A particular focal point at the beginning of the project was breaking down popular films and television shows into all their protagonists and antagonists and analysing their respective personalities for trends. Working out the connections between a characters personality and how their status and legitimacy is represented within their world throughout a film would yield insights into how film writers can convey messages/emotions effectively, making viewers think and feel certain ways, and produce a source of information which could be utilised by any. This exploration could reveal the situations and arguments film writers use to show their legitimacy and status within their universe. In the future, this could be used to create an interactive and responsive game solely based on validating those personalities, while hopefully combining the secret of creating and prolonging mystery through a story.

A specific focus was placed on how a certain personality could raise the status and legitimacy of one character in one universe, but how the same personality could also work detrimentally for another in a separate universe. In particular, mapping heroes to their nearest villainous personality match and vice versa opens a door to analysing films with opposite ideologies where the hero/villain are people who you would not expect, for example, The Matrix trilogy has the hero as a nerdy character and the villain as basically a middle manager type rather than a super powerful boss. The idea was to create a large map of fictional character personalities, with links to show how different personalities could coexist, for example romantic and platonic relationships, as well as how physical appearances can validate certain personalities and this could even be expanding to include real-life people and their life events which led to their rise or fall in society. Associated with this are the key interactions and plot points which validate one personality while invalidating the other. Having the capability to identify the scenes that "make or break" a character would provide a crucial insight into what makes or breaks a scene. Combining all the metadata regarding characters and scene metadata begins to establish a library of available information which could be beneficial for a myriad of people/sources.

1.2.2 Film/TV Scene Database

Collecting vast amounts of film scenes was imperative and from there creating a tool which could be used for labelling was the next big step. The labelling involves capturing the metadata and the state of a scene which is important for the meaning/ emotion of the scene. Therefore, having a large collection of scenes from varying films and television is integral to performing the maximum amount of labelling and evaluating to aim to answer the question posed at the beginning of the project.

1.2.3 Scene Annotation & Labelling

Ideally, the stories from film and television can be decomposed into each of the elements which make up a result which creates the emotional impact and story structure via utilising Amazon's Mechanical Turk to ask Workers to label sections of stories with categorical and structural labels. As mentioned in Section 1.1, there are multiple areas of importance which would aid answering the question posed at the beginning of the project: by what methods do film/ television reveal the emotions or meanings behind every scene so effectively? There are multiple factors which are intrinsic to depicting the meaning or emotion, some of which would be more easily labelled than others. Part of the puzzle is working out what can be labelled easily and what takes definitive time and effort to do.

Attempting to define the parameters which make a scene effective or not would be an extremely useful source: labelling important props, sound effects, background music and location, and most importantly the dialogue of the scene itself, along with the other properties aforementioned. A definitive collection of labelled data was not the aim for this project, instead a more focused and prioritised analysis of the literal detail of what happens in the scene was the focus, along with justifying and proving that the labelling process could be performed effectively. Beginning to understand the impact certain properties have on making a scene effective in terms of conveying emotion and underlying messages is more important creating a fully functional and interactive visual novel.

1.2.4 Plot Annotation & Labelling

Possibly the most ambitious and least concrete idea, in terms of research present on the internet today, is attempting to replicate the mystery of the story that is so well done in film or television. The most popular and highly rated films/television shows are ones which are not predictable and keep the viewer guessing, so it would be ideal to replicate this in game form. Writers generally build up a story slowly and uncover small jigsaw pieces of information at a time through clever camera shots, acting and foreshadowing. The viewers learn something new until finally the way in which those parts of the storyline become important are revealed at the end much like a jigsaw puzzle coming together. Changes within the story serve both to intensify the theme of the story and to provide unexpected 'information that help piece together the jigsaw of the story. This helps to constrain the unknown and convey important answers to questions the viewer may have and ideally even cause new questions due to introducing or partially linking existing jigsaw pieces. The length of time and quantity of questions the viewer must keep in mind during the story affects how challenging and cerebral the story. Often, many films conclude with a kind of symmetry which both validates the viewers own ideology while validating the protagonist of the story.

1.2.5 Visual Novel

Assuming the labelled results are obtained, the next step is to test the suitability and effectiveness of said results by using them to create a visual novel. The visual novel will capture the products from the labelling tool and represent them in a visual form to see if the results are enough to convey the emotion and underlying messages similarly to the film scene. The goal is for the results of the labelling tool to effectively capture the essence of a scene so that the process can be replicated easily, only for a visual novel format. The high level ambition with the visual novel becomes apparent depending on the quantity and quality of labelled information: if every factor and property mentioned in Section 1.1 was labelled definitively, an interactive visual novel could be developed where the player would be able to choose and customise their character according to personality types (derived from characters from film/ television) and could progress in the game through a series of specially chosen game states which allow the player to validate the personality type chosen. Each scene would have dialogue which provides a series of choices the player could experience, each leading to the story progressing according to their decision.

1.2.6 Technical Description of Project Objectives

A combination of programming languages: Python, HTML, CSS and JavaScript, along with Amazon's Mechanical Turk were planned to be utilised for this project. In particular, Amazon's Mechanical Turk was projected to be employed for multiple labelling tasks including but not limited to the following:

- Labelling character's physical permanent appearances. This included skin colour, eye colour, hair style and colour etc.;
- Labelling character's emotions/facial expressions throughout a scene;
- Labelling actions character makes and their importance to the storyline;
- Labelling dialogue from characters, including the inflection used;
- Labelling important props, sound effects, background music;
- Labelling background locations;
- Recording plot information: what is being revealed, why the characters are doing what they do, importance of location/objects in the scene.
- Get Workers to use the Open-Source Psychometrics Project Statistical Which Character Personality Quiz to create personality clusters/maps.[9]
- Once the visual novel was created, getting Workers to rate how well the message of the scene was conveyed compared to the video clip itself.

Evaluating whether MTurk provided a valuable labelling tool by assessing the precision the labelling was performed at was crucial. If the results garnered from the MTurk tasks were significantly precise, they would provide valuable insights into what makes a scene effective and would allow the properties to be generalised. Through generalising scene properties, the process for creating effective scenes could be formalised, creating a useful output for the use of multiple sources. From the outset, it was obvious that certain tasks would be too complex for Mechanical Turk workers to complete as they involved high-level thought processes which would be difficult to, firstly, instruct clearly, and secondly pay correctly, so it was clear that some of the labelling would need to be done manually. However the initial visual novel goal was that the labelled results would be passed into a table structure where the variables could be altered interactively, and the corresponding visual novel representation would alter to portray said changes. HTML and JavaScript would be used to create an interactive version in which the frames are defined so that they can be altered and adjusted, and a scroll bar included so you can scroll left/right to watch the scene play out. SVG (Scalable Vector Graphics) elements will be used to draw characters as a set of layers that we can choose from to alter their appearance/expressions dependent on the frame.

1.3 Initial Experiment of Feasibility

As with any project of a more theoretical calibre, requirements and specifications change as time passes and so the project's scope begins to narrow. The purview of the project started to change when the focus became more fixated on answering the questions posed at the beginning of the project: such as the lower-level labelling which could be easily outsourced to MTurk which in turn could be used to create a visual novel rather than attempting to label the multiple intricacies that are involved in scenes. To fully answer how films convey the underlying messaging and emotions so effectively was simply not feasible within the timeframe given for this project. Instead, evaluating the success of how MTurk can label scenes with significant precision and attempting to create the minimal viable product which would answer the key questions were the main goals. The scope was simply too large for a single project and it was decided that creating a simpler version which has the potential for expansion at a later stage was ideal. As a result of this decision, it was realised that utilising and storing the information that was gathered at the beginning was imperative along with leaving the completed outcome with significant opportunity for continuation by others at a later date.

Initially, a MTurk task was created and published asking workers to label a video from a film scene, where they had to label high level information such as declaring the characters seen along with their corresponding actors, scene location, key props and more detailed information, for instance, detailing exactly what occurs in the scene along with a plot summary gathered from the film's IMDB page. These tasks required the workers to have an in-depth knowledge of the scene and the actors in it, along with asking them to source data from other pages.

No workers filled out the tasks and so the task received no results, thus focusing on the lower level features which could be reliably obtained was deemed the next step. In particular, this Amazon's Mechanical Turk task was significantly cut back to two separate but linked tasks: one to label the physical appearance of characters, such as complexion, hair colour and style etc. and the other to label frames taken from a scene for a specific character, which included tasks such as labelling what the character said and how they said it. The rest of the aforementioned tasks were deemed too high level for Workers to successfully label with enough precision and in lieu of this they changed into an opportunity to become an avenue for development of the project at a later stage. Provided the labelled results were significant, a dictionary or library of properties which add to the effect films have to conveying meaning and emotion could be made. This metadata would be used directly to produce the visual novel which is coded predominantly with a combination of JavaScript, HTML and CSS. The interactive SVG elements proved too complex for the timeframe given for this project, so the character creator customisation and subsequent avatar images used in the visual novel were sourced from The Character Creator.[12]

1.4 Useful Outputs and their Target Audience

There are various useful outputs that will arise from this project. The main benefit will be directed towards professional game developers or scene writers who would find the labelled data and insights accumulated over the course of the project a promising source of information. Being able to analyse the minutia of the film scenes would allow similar replications to be made before possibly adapting the information gained from this project to use in making new game mechanics. The effectiveness of the secondary output is solely dependent on the precision of the labelled metadata and how conclusively it answers how films convey emotion/hidden meanings. This output can test the precision of the labelled data: by inputting the properties into a visual novel and determining how well the emotion is conveyed merely from the inputted data. Creating a successful game from the aforementioned features and factors would establish a new format for more intricate games which take personality types fully into account and react flexibly depending on said personality type. This would be beneficial for not only new game developers etc. who want to dive into the world of visual novels but also game players who never have an emotional bond with the characters in a game. In fact, it would provide players the chance to validate their own personalities or a personality of their choice, leading to an emotional connection found nowhere else.

Other useful outputs and their respective target audiences include:

1. *How-to-guide on how to set up Python/Pip on slow computers:* As the owner of an old and slow laptop, it is reasonable to believe other people suffer from the same affliction. This guide solves the problem for people who cannot download/install Anaconda. This

could be useful in terms of being able to use Amazon's mechanical Turk without having to download Anaconda etc.

2. *How-to-guide on Amazon's Mechanical Turk and how to create Requester Tasks:* This guide would be helpful for anyone looking to start using Amazon's Mechanical Turk and would aid their journey from the very start of the process, throughout. This is combined with a guide on how to create a Requester task and publish a batch of HITs using the templates available.

3. *How-to-guide on GitHub Pages:* This guide could be helpful for anyone looking to start using GitHub Pages to host web pages.

4. *Storing Character Personality Traits:* Statistics from the Open Psychometric Personality Test Statistical "Which Character" Personality Quiz were taken and stored in the final web application of the project, providing a source of information of film writers or game character developer. This data can then be inputted back into the test to allow similar characters belonging to that personality type to appear: this creates a larger map of similar personalities for the aforementioned people to work off.

5. *Storing Scenes from Film/Television:* Scenes which represent important interactions between and about characters e.g. hero getting broke down, then rising back up were collected for use in manual and Mechanical Turk labelling tasks. Mechanical Turk workers gather metadata based on the scenes, recording properties such as background status, emotion of characters before and after dialogue, how the world changes respective to interactions, as well as recording physical (permanent and temporary) traits of the characters and breaking down the physical shots used in the cinematography.

6. *Formalisation of Scenes:* Connected to the output above, number 5, another useful output is the creation of a dedicated process on how to formalise scenes as it has not been done often, if ever. The successful MTurk labelling tasks provided a minimal version of this process and proved to effectively formalise scenes dependent on the factors labelled. The process can be replicated for a less minimal product during future development.

7. *Storing Important Dialogue from Film/Television:* Important from dialogues (lines and actions) from important scenes which influence the viewer/players perception of the characters status and legitimacy was collected. As with the two points above, this could be a useful archive of information for film/television writers/game developers when writing important scenes. Mechanical Turk workers labelled the emotional states of characters dependent on the words they say and their inflection. The MTurk workers also recorded key actions such as being given an object/attacking/moving to another location.

8. *Character Avatar Customisation:* A character customisation could be used by multiple people for example game designers etc. but is mainly used for the labelling of characters from scenes to ensure that the physical appearance factors which are integral to the storytelling are recorded. During the duration of this project, the Character Creator website was utilised for the customisation of avatars.[12]

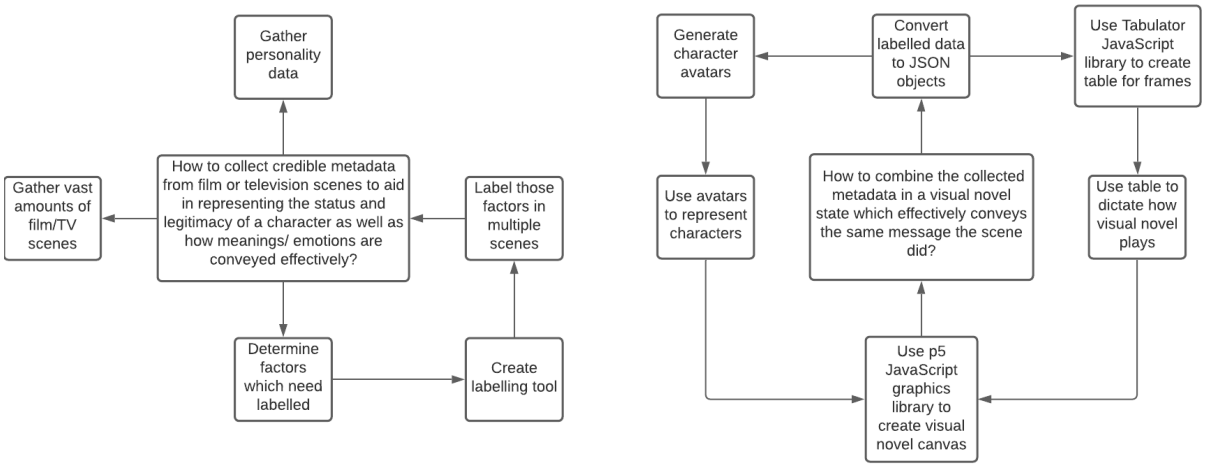
1.5 Interface Design

User interface design focuses on the users visual experience and determines how a user interacts with the interface, in this example, a web application. This project is made up of many sources and is the product of weeks of research and data mining, whether or not the data made it into the visual novel game or not. Consequently, it is imperative that all the research and data gathered is held in pages on the website which are easily navigated between and intuitive to use. The website itself will be hosted on GitHub Pages directly from a GitHub repository, and Notion, an all-in-one workspace, will be used to convert documents and Excel spreadsheets created over the duration of the project.[13] Having access to all the metadata from scene breakdowns and labelling, as well as personality investigations provide a source of information which start to formalise how film conveys hidden meanings and emotions effectively.

Chapter 2

Proposed Solution and Development Model

2.1 Proposed Solution



(a) How to gather information

(b) How to test data with visual novel.

Figure 2.1: Visual summary of proposed solution to the two main objectives: (a) displays the proposed solution on how to collect credible data from film/television, (b) displays the proposed solution on how to test the effectiveness of the collected data in visual novel form.

The integral aspect of this project is to, while considering the previously discussed factors in Chapter 1, answer how film and television convey hidden meanings/ emotions in scenes and thus systematically breakdown and label film and/or television scenes into their important metadata. As aforementioned, no software solution exists at this time which takes scenes and fully categorises their properties and tests their effectiveness by transforming them into a visual novel, therefore a degree of originality is guaranteed. This project aims to capture the state of a scene which is important for the underlying meaning/ emotion and to connect the world of film making and video games to build a visual novel, which has the potential for future greatness through increased interactivity and features, by attempting to replicate the

same process. Figure 2.1 displays a visual summary of the proposed solution for this project.

If successful, this project will address the questions on how to reverse engineer and so replicate the factors which allow film/television to convey meanings/emotions in scenes. There was a need to manage expectations: what is feasible and what is not? The visual novel will be able to read in JSON objects converted from the Mechanical Turk labelling batch results and subsequently draw call rendering a p5 canvas with the respective avatar and background images. Amazon's Mechanical Turk is a web service which provides a scalable, and most importantly, human workforce that can complete a variety of tasks, such as data categorisation, verification and tagging. It provides a marketplace for work with access to workers across the world (see Section 4.1.6 for more detail on MTurk).[14] The p5 graphics library is is popular with artists and designers for creative coding projects based on Processing, a creative coding environment and particularly useful for drawing and creating graphics.[15] JSON stands for JavaScript Object Notation, and is a lightweight format for storing and transporting data. The other aspects of the project come from the useful outputs mentioned in the last chapter: creating an application to store the data and information gathered either manually or by Turk workers.

2.2 The Development Strategy

There are multiple different development strategies and models utilised within the software industry today. Some follow a sequential approach such as the Waterfall or V model, whereas others follow a more iterative approach with Agile methods such as Scrum.[6]

2.2.1 Sequential Approach v. Iterative Approach

The first step using the sequential approach involves the gathering of requirements from the stakeholders, so this method proves advantageous when all the requirements are clear cut from the beginning. Because of this, this method does not suit projects where the requirements may be more ambiguous. There is also an explicit time scale for the development which allows continual validation and verification after each phase. Validation refers to guaranteeing the system aligns with the requirements, whereas verification guarantees the system itself is being built correctly. As mentioned earlier, the sequential approach can be narrowed down to two main models: the Waterfall Model and the V Model, both of which follow very similar processes during the development life cycle as shown in the figure below. [16] The main advantages to the sequential approach are that it is usually cost effective, efficient and that no pre-knowledge or training is required. However the disadvantages are how rigid it is and especially how hard it is to make any changes once the testing stage is reached. If any massive errors or bugs appear in the testing stage, it is expensive and takes a lot of time to rectify.[16]

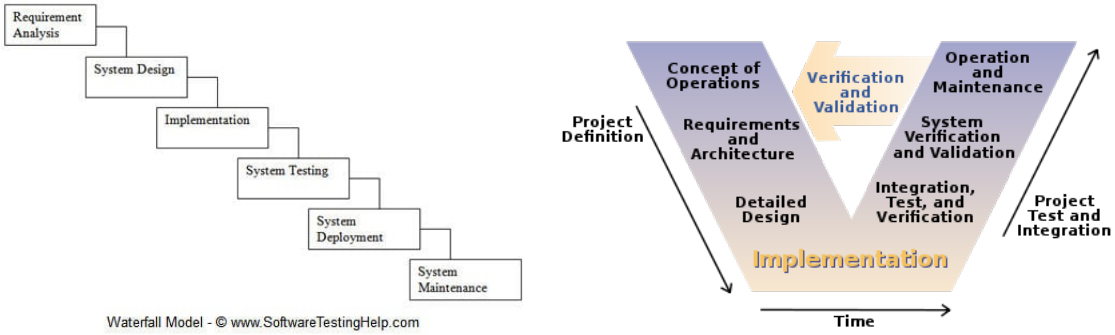


Figure 2.2: Images depicting the processes of the Waterfall Model and the V Model respectively.[3][4]

On the other hand, the Agile development model refers to an umbrella term for set procedures based on the Agile Manifesto. In fact, the creators of Agile Manifesto chose the name "Agile" because it was a "...word represented the adaptiveness and response to change which was so important to their approach".[17] A particular significance is placed on the delivery of less complex outcomes with frequent changes to design and regular feedback from the stakeholders. This approach is decidedly more suited to projects which have less concrete or ever-changing requirements as it is easy to make changes anywhere along the life cycle.[6] A main focus of the Agile method is collaboration: focusing on who is working and how they work together. The main advantages to Agile/Iterative are the ease of adaptability, frequent delivery and general reduction on total development time. Disadvantages include high costs relating to late-stage issues as well as the fact it becomes more difficult to establish an end date for the project's conclusion. An Agile approach like SCRUM was unsuitable for this project due to the project being completed by only one person compared to the usual multiple people who could have daily scrum meetings and hold each other accountable. Another model, like the Iterative model, had a much better likelihood of being suitable for this project. In this case, iterative refers to planning the task from one iteration to be revised and upgraded in any ensuing iterations, therefore with each iteration, greater detail is added and the software continually grows more superior, and incremental refers to ensuring each task is perfected and completed throughout the project.[18]

After concentrating directly on the project in question and considering the advantages and disadvantages of both approaches, it was determined that the best fitting model to follow would be the Iterative software development life-cycle approach. This project presents a more theoretical take on requirements, with less factors set in stone in the beginning, and considering the sequential approach focusing mainly on clear-cut requirements, it does not offer the appropriate adaptability needed. With the Iterative model, the development process requirements can be expanded at any point, and its repetitive nature allows new versions of the same product to be made for every iteration.

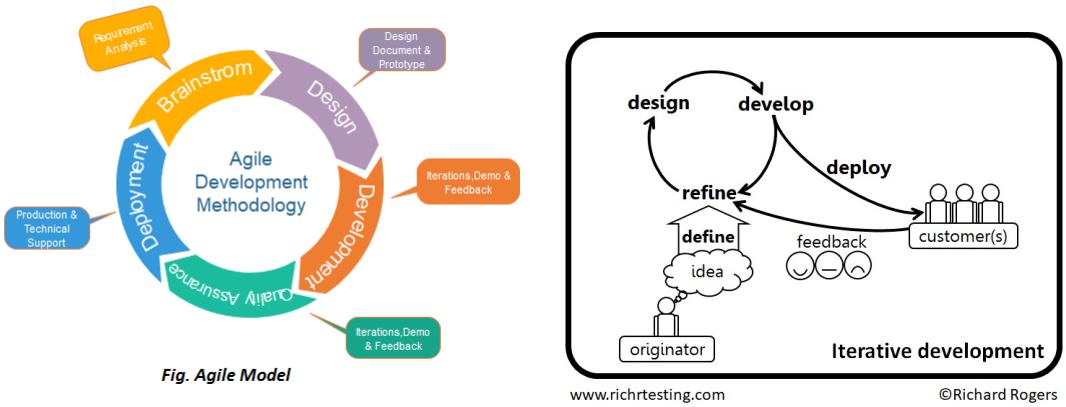


Figure 2.3: Images depicting the processes of the Agile Model and the Iterative Model respectively.[5][6]

2.3 Design Details

In the timeline of this project, the iterations coincided with weekly meetings with the projects supervisor, Dr John Bustard. Each meeting consisted of an evaluation of the work done prior along with new goals and deliverables for the next iteration. Despite weekly meetings, where learning logs helped keep accountable for the work completed, most iterations lasted for between two or three weeks.

The project will attempt to solve, or start to solve, two main questions:

1. How to collect credible metadata from film or television scenes to aid in representing the status and legitimacy of a character as well as how meanings/ emotions are conveyed effectively?

2. How to combine the collected metadata in a visual novel state which effectively conveys the same message the scene did?

Taking these questions into consideration, the project will commence with the accumulation of metadata, through manual means and then through Amazon's Mechanical Turk, then onto combining the metadata into a visual novel. The main challenges will arise in the gathering of metadata in terms of time limits, i.e. how much can get done in the time frame given, complexity, i.e. factors which are too difficult or high-level for Turk workers to be expected to label, or funds to pay the Turk workers. For the visual novel itself, it depends on getting the metadata in the first place to begin development.

Chapter 3

Requirements Analysis and Specification

3.1 Requirement Elicitation

Requirement elicitation refers to the process of gathering information from stakeholders to serve as the foundation of beginning recording requirements of a system. [19] Given the chosen development model, iterative, the functional and non-functional requirements as well as the error conditions for the software have been written retrospectively. In fact, in this particular project what was possible or what was best in terms of development only become apparent as the project progressed. Consequently, the requirements match the software and products created rather than dictating how the development started. This way, the requirements can be used as an opportunity to instead communicate the key important considerations for the different components of the project.

3.2 User Stories

As mentioned above, the requirements were determined by working backwards from the completed project, therefore the user stories could also be devised from the final results while the user needs were formulated from the start of development and throughout. User stories were developed along with their respective acceptance criteria to be tested and evaluated by users at a later stage. Table 3.1 contains the user stories that apply to all users who want/ need to access the user interface in its entirety, whether they are part of the general public, game developers, writers etc. The user should be able to access any stored data and download it in some form, as well as being able to adapt or add information at will. The applied Mechanical Turk task templates are hosted in the GitHub repository so future developers should be able to alter templates and follow a guide to creating their own batch. Users must also be able to interact with visual novel, even with reduced capabilities such as simply pressing play to begin.

ID	User	Acceptance Criteria
1	As a user, I want to be able to access personality data on fictional characters.	1. Can view all pages on website. 2. Can navigate to each page easily.
2	As a user, I want to be able to follow a guide to publish my own batch on Mechanical Turk.	1. Can access how-to guide for Mechanical Turk.
3	As a user, I want to be able to add to stored metadata from scenes/characters.	1. Can add new information to tables.
4	As a user, I want to be able to create/customise character avatars.	1. Can create new avatars using character creator. 2. Can download avatar as PNG or SVG files.
5	As a user, I want to be able to play a visual novel representation of a scene.	1. Can access visual novel page. 2. Can choose which scene to play. 3. Can press play to start the visual novel.

Table 3.1: Table displaying the user stories for UI users.

Table 3.2 contains the user stories specific to Amazon's Mechanical Turk workers when completing a Human Intelligence Task or HIT. Amazon's Requester takes care of most of the user stories itself, but some processes must be done manually as the developer.

ID	Mechanical Turk Worker	Acceptance Criteria
6	As a Mechanical Turk worker, I want to be able to access HITs.	1. Can access and view each HIT without any errors.
7	As a Mechanical Turk worker, I want to be able to view all media required in a task.	1. Can view any links/ videos/ images fully.
8	As a Mechanical Turk worker, I want to be able to follow clear instructions to complete a task.	1. Can view clear and full instructions with no ambiguity.
9	As a Mechanical Turk worker, I want to be able to submit a HIT.	1. Can submit a HIT successfully.
10	As a Mechanical Turk worker, I want my results to be approved and paid or rejected with explanation, in good time according to the creators views and paid.	1. Receives payment or explanation of rejection in good time as feedback.

Table 3.2: Table displaying the user stories specific to Mechanical Turk workers.

3.3 Requirement Specification

In Figure 3.1, the core requirements taken after the project was completed are displayed as *"High"* priority and are highlighted green, emphasising the requirements which were satisfied. These are the high-level objectives and are the things that need to be understood, where they all add up to begin to answer how film/television convey hidden messaging/ emotions effec-

tively. The high priority requirements are what was done to aid that understanding. These were the requirements which needed to be met in order to satisfy the project specifications. *"Future"* refers to some of the some, but not all, of the requirements which could be satisfied in future development; these could add to functionality but are not wholly necessary. As the aim of the project was to create a minimal viable product which can start to answer the question posed at the beginning of the project, and prove whether it is feasible to break down scenes to work out their inner mechanics in order to replicate the process, these requirements are opportunities for future development. Figure 3.1 refers to the things we want from the project, and the notes refer to the parts that were focused on due to feasibility.

ID	Task/Story	Priority	Notes
1	stories to discover how they convey hidden meanings/emotions	High	breakdown for labelling of properties.• Prioritise labelling to create minimal viable product to achieve blueprint for continuation.
2	Gather scenes from film/television.	High	• Utilising YouTube-DL to scrape scene videos.
3	Cut scenes into frames.	High	• Utilising Python.
4	Create a labelling tool to label scenes in terms of their metadata.	High	• Label frames from scenes and characters.
5	Create labelling tasks to gather metadata.	High	• Utilise labelling tool and/or manually label to compare results.
6	Create visual novel with data from Turk/manual labelling.	High	• Utilising p5.js, JavaScript, HTML, CSS.
7	Create character avatars for visual novel representation.	High	• Adapt charactercreator.org.
8	Add heightened interactive features to visual novel.	Future	• Opportunites for future development: add interactivity to visual novel.
9	Create unique character avatars for visual novel representation.	Future	• Opportunites for future development: create own avatar generator.
10	Breakdown and generalise scenes.	Future	• Label additional elements to create a more definitive collection of data.
11	Examine links between personality of characters and storyline.	Future	• Find link/connection between personalities and characters status and legitimacy. • Try to replicate validation/rejection process.

Figure 3.1: Table displaying the requirements including their priority, where High is what was done and Future refers to bonus points should the others be satisfied and/or future development opportunities.

3.3.1 Functional & Non-Functional Requirements

Functional requirements are defined as the characteristics which authorise a system to function in the way it was intended. If these requirements are not met then the system will simply not work the way it was expected to.[20] They refer to a description of the service or action the system must follow and can be anything from data manipulation to user interactions. In simple terms, these types of requirements define what a system must or must not do. The functional requirements can be seen in detail in Appendix B.1.

Conversely, non-functional requirements focus on the *how* rather than the what. As per the name, these requirements have no effect on the functionality of a system, instead they focus

on the usability of a system.[20] They refer to the property properties in place of the product features and defines the factors which affect a user's experience. This includes facts such as usability, performance and scalability. The non-functional requirements can be seen in detail in Appendix C.1. These were generated through general common sense when it comes to the properties one would like in a web application and focus on user experience.

3.3.2 User Journeys

ID	User
1	User accesses internet → accesses website → navigates through website → views stored data
2	User accesses website → navigates to stored data → adds new data
3	User accesses website → navigates to character avatar creator → creates existing or new character → downloads image files
4	User accesses website → navigates to visual novel → can click play to begin
5	Mechanical Turk worker accesses HIT → completes HIT and submits results → is approved or rejected accordingly
6	User (project author) accesses MTurk Batch results → examines results → approves or rejects accordingly

Table 3.3: Table displaying the user journeys to be demonstrated at project end.

To aid the demonstration of the final working products at the end of this project, user journeys were established to demonstrate system functionality.

3.4 Determination of Valuable Data to Label

This section contains the justification of the data chosen to be labelled in the MTurk tasks was determined. Determining the pieces of data which would be deemed the most valuable for the timeframe of the project was a high priority act. It is necessary to provide the minimal viable product to prove whether this project will be successful or not hence the very literal detail of a scene and a character's appearance was focused on. This literal detail in a scene consisted of anything which could affect the mood or underlying message: such as the sound effects and props utilised to get the story across, as well as the change in the character's emotion and dialogue, along with the inflection they use when speaking. These properties were all deemed important to how the story progresses. The literal detail involved in the character's physical appearance includes permanent physical features, such as hair style and colour, eye colour, the character's attire, their complexion and any distinctive features such as if they wear glasses or have any tattoos or scars. There are many more properties in terms of a character's appearance which could be deemed integral to the story, but they are harder to identify and label, for example determining the height of a character depends on what the character is doing in a certain scene: if they are seated then it is hard to ascertain. Thus, only the most easily identifiable factors and properties were utilised for both labelling tasks.

Chapter 4

Design

This chapter will describe and justify the design of the proposed system, including the user interface, architecture and system design for this project. This chapter is split into two main sections: the User Interface Design and the System Design. For the user interface, the visual novel and the MTurk tasks are the most technically challenging and time-consuming aspects of the project. Figure 4.1 portrays all the design factors which make up the entire project.

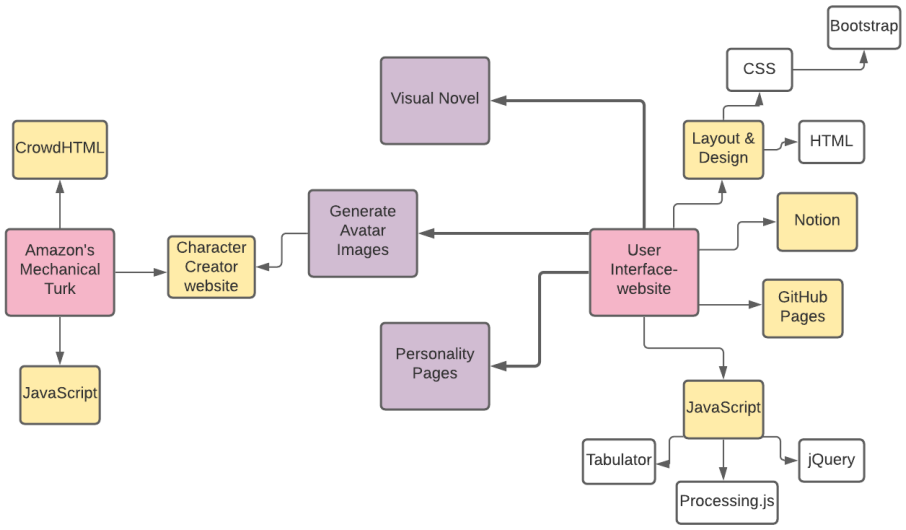


Figure 4.1: This figure shows a visual summary of the design details completed and shows at a glance how the different sub-sections contribute to the whole outcome.

The boxes in pink show the two main focuses, the web application created and the Amazon Mechanical Turk tasks. Each of these boxes have different boxes branching off. The purple boxes connected to the UI section detail the three main sections of the web application, the visual novel which proves the effectiveness of the labelled scene results from MTurk, the Generate Avatar Images page which proves the effectiveness of the labelled character results from MTurk, and the Personality Pages which hold the metadata sourced from the Open-Source Psychometrics Project. The yellow boxes detail the more technical software design utilised in the creation of the web application. The Amazon's Mechanical Turk box contains its own

technical design boxes which are largely unrelated to the UI design. Each of these boxes are covered in more detail in the sections below.

4.1 User Interface Design

This section details the decisions made around aesthetics, layouts and general functionality users will interact with in the user interface, which is in this case, a website. This section begins with an introduction to the languages, libraries and frameworks used, followed by an insight into original designs compared to the final outcome, journeying from the home page throughout highlighting the most important aspects a user would view. The front-end or user interface for this project's web application was developed mainly through HTML, JavaScript, CSS and Notion, and is hosted exclusively on GitHub Pages. The majority of following languages and libraries were chosen based on the huge wealth of resources each had available to assist throughout development. This allowed less time to go towards learning new languages and more time actually spent developing.

4.1.1 HTML and CSS

HTML or HyperText Mark-up Language is a standardised system which acts similarly to Microsoft Word in that it allows users to create and format structures such as headings, paragraphs, hyperlinks and media etc. on World Wide Web pages as well as edit and adapt fonts and colours. HTML provided the visible templates and structure for the rest of the website. A variation of ordinary HTML called Crowd HTML was utilised for the Amazon's Mechanical Turk tasks.[21] These will be covered in more detail in Section 4.1.6.

CSS or Cascading Style Sheets describe how HTML is to be displayed on screen and can control the layout and style of multiple web pages at a time. It's common to use only one CSS file for a full project. The main CSS framework used in this project was Bootstrap, a free and open-source framework widely used.[22]

4.1.2 Notion

One of the main prerequisites was that the research and data collection performed at the beginning of the project was present in the final product as it made the answer to the question on how film/television convey emotions in scenes. Notion is an all-in-one workspace which can easily convert Excel spreadsheets and Word documents into visually aesthetic HTML tables and pages, ready for quick export to add to the project's UI.[13] This made Notion is perfect application to utilise to ensure all of the hard work performed was displayed in satisfying way which was easy to use.

4.1.3 GitHub Pages

GitHub Page is a subset of GitHub, the world's leading software development platform, and it is a static website hosting service which runs directly from a GitHub repository. Thereby any HTML, CSS and JavaScript files can be pushed straight to a repository and GitHub Pages takes and applies them to publish a website.[23] The main benefits of using GitHub Pages are that it's completely free, it supports custom domain names (which makes it very attractive for blog creator) and it has support for HTTPS, or Hypertext Transfer Protocol Secure, which provides a layer of security and encryption. Another benefit is that the pages can made completely through Jekyll, which is basically a static website generator, if a person had little web development experience.

4.1.4 JavaScript

JavaScript refers to a text-based programming language which adds interactivity to web pages that HTML and CSS alone cannot provide. This interactivity includes examples such as changing colours of elements on click or hover, displaying animation, playing audio/video and even game development.[24] JavaScript is utilised both on client and server side and is the only language native to web browsers, making it the most popular coding language today. There is a myriad of JavaScript libraries available for public use, and all integrate easily with HTML, including the libraries utilised in this project:

- *jQuery:* This popular JavaScript library is fast, small, and feature-rich with excellent versatility and extensibility.[25] This is utilised alongside "vanilla" JavaScript throughout this project.
- *p5.js:* This JavaScript library is popular with artists and designers for creative coding projects based on Processing, a creative coding environment. p5.js is particularly useful for drawing and creating graphics.[15] This particular library is used in the creation and development of the visual novel.
- *Tabulator:* This JavaScript library creates interactive tables from almost any data source, including JSON formatted data and HTML tables.[26] This library is used in multiple places throughout this project, including storing data associated with the visual novel and labelled metadata from mechanical Turk tasks.

4.1.5 Layout Design and Use of Colour

As with any web application ease of use is paramount for navigating between pages, therefore a navigation bar was added to every page. The navigation options were as clear and obvious as possible, so users knew exactly what was in each section. Titles and headings were kept as concise and self-explanatory as feasible to continue to meet the requirements stated earlier.

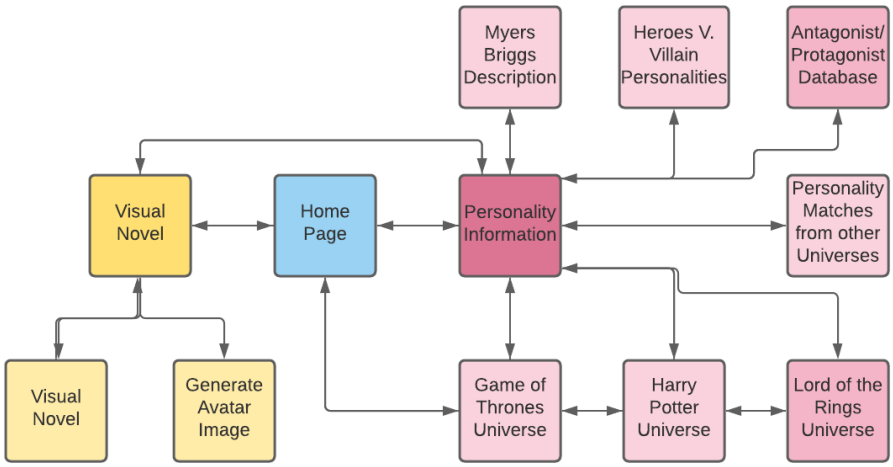


Figure 4.2: This figure shows a visual summary of all the main user interface screens and the arrows portray how they can be navigated between.

Figure 4.2 shows a visual summary of all the main user interface pages and how they can be navigated between. Every main web page can be navigated to through the navigation bar. An important aspect of any user interface is the use of colour, whether in the background, font, buttons or navigation bar. Any text needs to be strengthened with a colour to ensure it stands out and is clear for users. Most of the colour decisions were made through trial and error to find a colour scheme which suited the aesthetic of the entire project. A simple shade of black and white was used throughout the web application, with pops of blue used to highlight buttons and links.

4.1.6 Amazon Mechanical Turk Tasks

Amazon's Mechanical Turk is a web service which provides a scalable, and most importantly, human workforce that can complete a variety of tasks, such as data categorisation, verification and tagging. It provides a marketplace for work with access to workers across the world.[14] One of the main benefits as a task maker, or Requester, is you only have to pay for work deemed satisfactory: if a worker does not comply with the instructions given, there is no requirement to pay them for their time. This ensures quality over quantity. As a Requester, projects and tasks can be created via the Requester User Interface (RUI), as well as checking the status of published tasks, and accepting or rejecting work depending on their quality. Each task created has a set of HITs, or Human Intelligence Tasks, which are self-contained tasks, such as identify the animal in this image. These factors made Mechanical Turk the perfect labelling tool to begin parameterising scene detail to begin to answer how film and television convey emotions and underlying messages effectively.

With creating MTurk tasks, certain properties require declaring before publishing. These include factors such as assignment, approval and payment and qualification type. Assignment

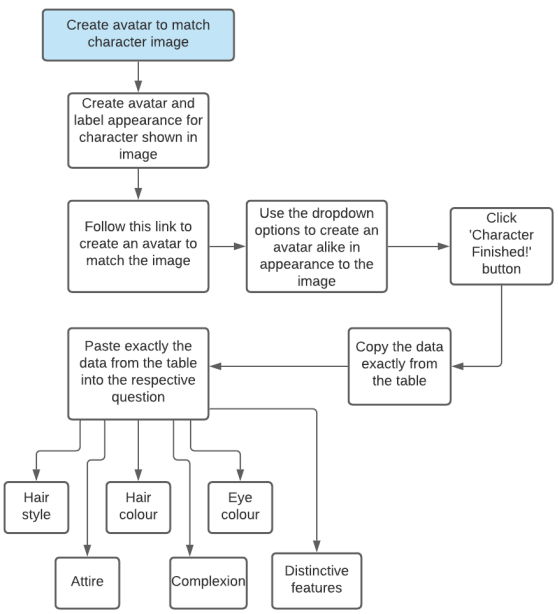
refers to how many workers are assigned to work on the same HIT: this ensures an effective way to test consensus on a subject if multiple workers provide the same answer. The approval and payment involve how much the Requester wishes to pay a worker per HIT, and usually depends on how much work they have to do, along with the estimated completion time. When a worker submits their response, the Requester must either approve or reject their submission depending on the quality of their answer. Once approval is performed, MTurk transfers the reward associated with the HIT into the workers account. The Requester must also set a deadline by which the HITs are approved or rejected before MTurk automatically approves any submitted assignments. The final factor is qualifications: this refers to the qualifications of the worker. To control who works on a HIT, the Requester can set specific restrictions such as region in the world the worker resides in, Approval Rate, which is the percentage of assignments a worker has been approved for and Assignments Approved, which is the number of approved assignments. Usually MTurk recommends "Workers to have a 95% Approval Rate and 1,000 Approved Assignments to work on your project" to ensure quality results, however it is not necessary to set any qualifications for a task.[27]

There is less of an emphasis on aesthetics and more purely on functionality with MTurk tasks: does media load correctly, are the questions clear, do the input fields or dropdowns work correctly? Amazon's MTurk Requester has multiple templates for creating new projects such as surveys, emotion detection, data collection etc. and all of them use Crowd HTML by utilising the Crowd HTML JavaScript file present in all templates.[21] To begin making a new project, the properties that are common for all of the tasks created using this project must be specified. These include a title and description, along with reward and time allocated per assignment. Next the HTML editor can be used to design the layout of a task for Workers to complete. Any HTML, CSS or JavaScript can be utilised to customise the layout design. Any variables added into the project, for example images which need labelling, must be defined as ${variable_name} and added to the CSV input file. MTurk automatically creates separate HITs for every row in the input file, thus enabling multiple Workers to work at the same time.[14] See figure in Appendix D.1 for an example of sections from a working MTurk task.

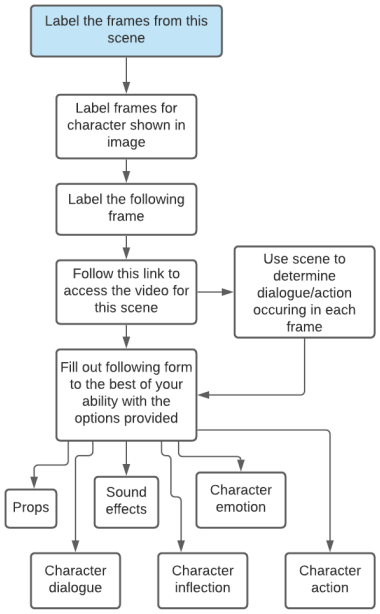
Figure 4.3 portray flowcharts detailing the general design of the character appearance and scene labelling tasks. These will be explained in more detail in Chapter 5.

4.1.7 Visual Novel

The visual novel makes up the largest visual test of the project's success, and ties together data collected manually and through MTurk Workers and can be accessed through the navigation bar on any of the pages. The visual novel page contains the same navigation bar and footer as the rest of the pages and follows the same colour theme. When the page is first loaded, a modal, or pop up will appear with instructions telling the user how to play the visual novel. Once this is closed, users will see a white background followed by a dropdown asking them to



(a) Flowchart for design of character appearance labelling task



(b) Flowchart for design of frames from scene labelling task

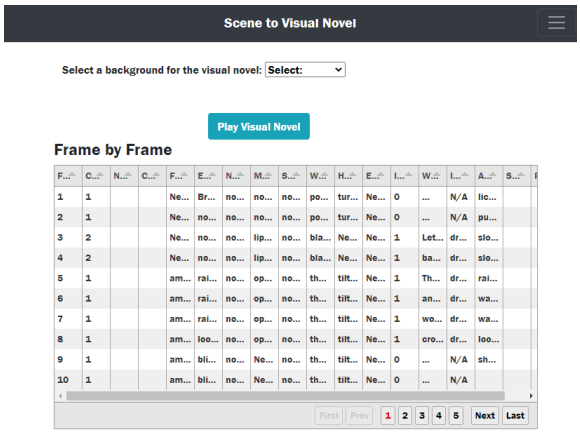
Figure 4.3: These figures display the flowchart design of the two main MTurk tasks performed: (a) details the process a worker follows when labelling the physical appearance of a given character (b) details the process a worker follows when labelling a frame for a given character from a scene.

select a background image from the list: this includes images such as a board room, beach, warehouse, prison etc. Beneath this dropdown is the 'Play Visual Novel' button followed by a table which is filled with a frame-to-frame breakdown of the visual novel's scene. These details are visible in Figure 4.4.

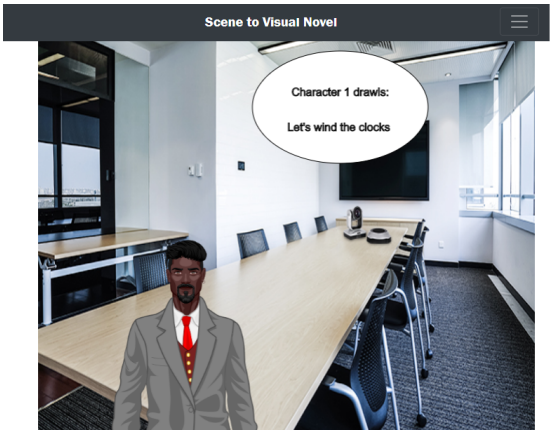
When the user clicks the 'Play Visual Novel' button, the visual novel will begin to play with a six second interval between each frame so the user can easily see the dialogue and character expression. The canvas loads the chosen background image and iterates through the frame table to display the corresponding character image with their corresponding emotion and dialogue. The dialogue is contained in a speech bubble and included the character number who is speaking, their inflection, and the words they say. These factors are sufficient to portray the essence of the scene effectively. The visual novel is as simple as possible to avoid incorrect user interactions, meaning the user only has to choose a background image and click play for the visual novel to work. More detail on the visual novel is explained in Chapter 5.

4.1.8 Generate Avatar Images

This page takes from the Character Creator website created by Frédéric Guimont and the labelled character appearance data from MTurk to allow users to pick one of the labelled characters and show them in avatar form.[12] Figure 4.5 shows the layout of the page remains sim-



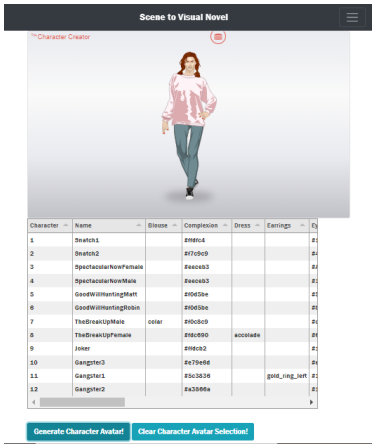
(a) Page entitled *Layout of page functions*.



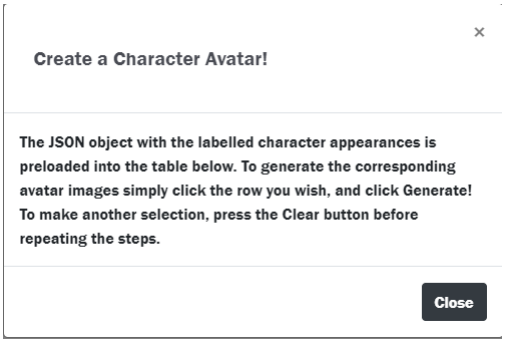
(b) Page entitled *Visual Novel*.

Figure 4.4: These figures display the visual novel: (a) displays the dropdown, play button and frame-to-frame table (b) displays the visual novel when played.

ilar to the rest of the web application, and portrays how the page looks when a character is selected.



(a) Page entitled *Generate Avatar Images*.



(b) Modal which appears to instruct users.

Figure 4.5: These figures display the page used to generate avatar images: (a) displays the page when a character has been chosen, (b) displays the modal used to instruct users how to use the page.

4.1.9 Web Application User Interface

The user interface for this web application needed to be kept simple and easy to use: there was no real priority for highly aesthetic user interfaces, as long as the whole website tied together with similar themes and layouts. It is simplistic in design and does not have many additional features bar what is required for the course of the project. The theme was kept consistent throughout through the use of imagery, templates, colours and fonts. The website was named 'Scene to Visual Novel' which is apt as the project's aim is to understand how scenes

can be broken down into factors which aid conveying hidden messages and emotions, the effectiveness of which is tested in a visual novel format. The colour scheme implemented was greys, black and white, which added a classic feel to the website and particular highlighting of links and/or buttons was implemented with a blue-teal colour. The home page serves as a simple introduction to the website, where the navigation bar at the top holds the different pages which can be accessed, and can be seen clearly in Figure 4.6.

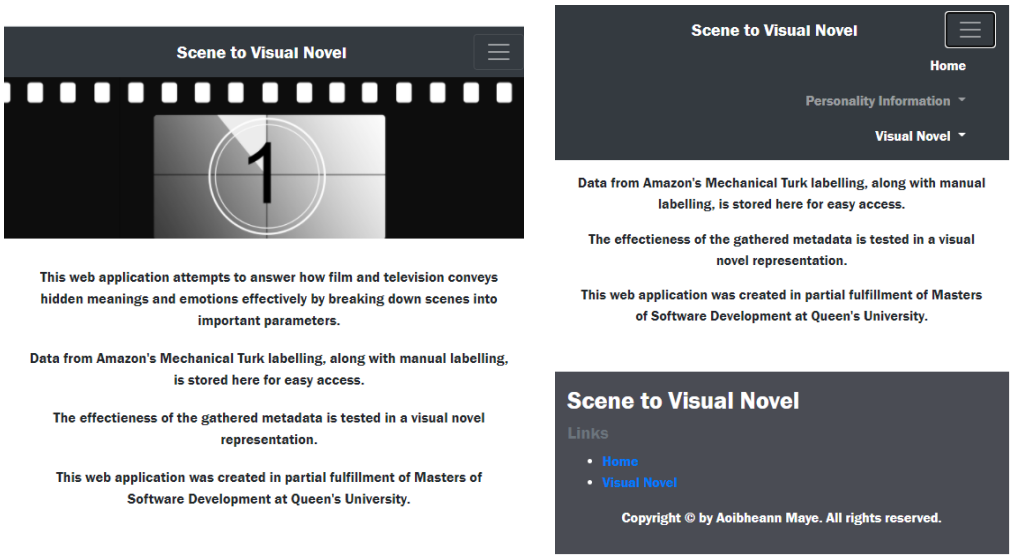


Figure 4.6: Images depicting the home page of the user interface. The first shows the index.html ordinarily, and the second shows the navigation bar dropped down, as well as displaying the footer.

The navigation bar can bring users to multiple pages: this includes the home page, the visual novel (see Section 4.1.7) and the character avatar creation page, and the personality information pages comprising of the following pages, each of which will be covered in more detail in their respective sections:

- Myers Briggs Description;
- Hero v. Villain Personalities;
- Antagonist/Protagonist Database;
- Personality Matches from other Universes;
- Game of Thrones Universe;
- Harry Potter Universe;
- Lord of the Rings Universe;

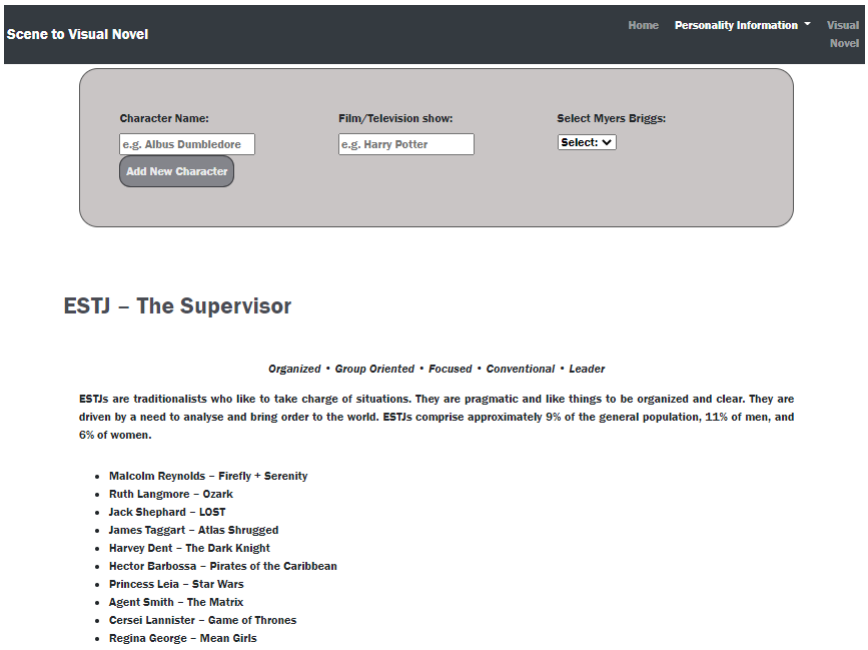


Figure 4.7: This figure shows the page entitled *Myers Briggs Description* and displays the user opportunity to add a new character as well as how each Myers Briggs Type is laid out.

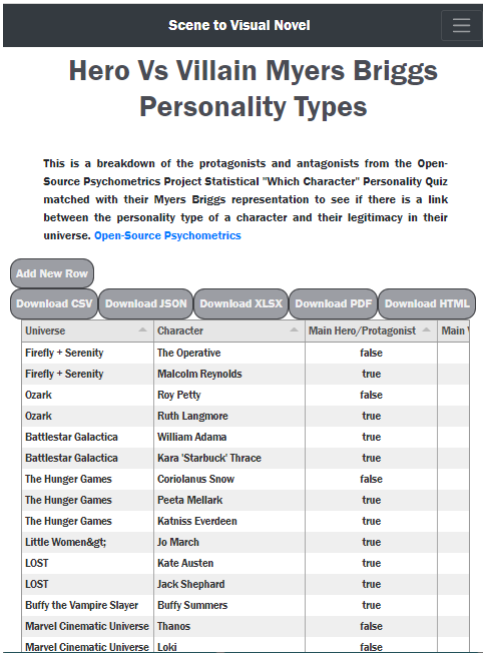
4.1.9.1 Myers Briggs Description

This web page describes the Myers Briggs typology system works, describes the sixteen possible personality types and presents the breakdown of all the characters present in the Open-Source Psychometrics "Which Character" Personality Quiz into their respective Myers Briggs types.[9] It includes a link to a table present on a web page described in the next section, Section 4.1.9.2. An image of a Myers Briggs personality key is utilised to give a brief insight into the eight letters which can be used in a Myers Briggs type. Beneath this is a simple prototype form in which users can add a new fictional character to their corresponding personality type in order to increase the amount of metadata stored. The more data recorded, the easier it becomes to find connections and links between characters and their personalities. This was removed as it was not feasible in the timeframe of the project. Figure 4.7 is a screenshot detailing the above content.

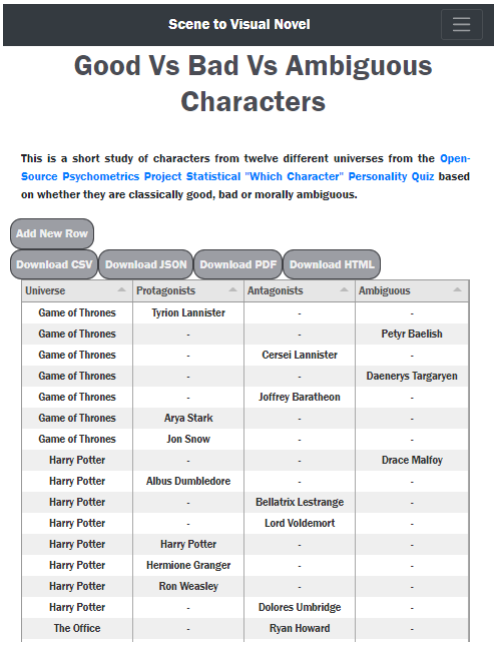
4.1.9.2 Hero V. Villain Myers Briggs Personality Types

This web page contains a breakdown of the protagonists and antagonists from the Open-Source Psychometrics Project Statistical "Which Character" Personality Quiz matched with their Myers Briggs representation in table form. The table portrays multiple characters from different universes, determines whether they are the main hero or villain in that universe, and their Myers Briggs type. The Main Hero/Protagonist and Main Villain/Antagonist columns use boolean strings, true or false, to determine who the character represents in that universe. Prototype buttons were added at the top of the table with user options to add a new row to the bottom of the table, so adding a new entry and increasing the legitimacy of the data collected in terms of answering the question posed at the beginning of the project, and buttons with options to download the table in a variety of forms. The button to add a new row was

removed but remains a good opportunity for future development of this page. This allows users to have easy access to the data for personal use. A screenshot of this page can be seen in *(a)* of Figure 4.8.



(a) Page entitled *Hero Vs Villain Myers Briggs Personality Types*



(b) page entitled *Good Vs Bad Vs Ambiguous Characters*

Figure 4.8: (a) displays the table of data along with buttons which users can utilise to add new data or download the table details, (b) displays the table of data along with buttons which users can utilise to add new data or download the table details.

4.1.9.3 Antagonist/Protagonist Database

This web page displays a short study of characters from twelve different universes from the Open-Source Psychometrics Project Statistical "Which Character" Personality Quiz based on whether they are classically good, bad or morally ambiguous. [9] The figure shows a prototype where this information can be added to as with the tables mentioned before, and the table data can be downloaded in different forms. This was removed as it was not feasible in the timeframe of the project. This information could be useful as a definitive library of the protagonists, antagonists and morally ambiguous characters in every universe, which could then be in turn combined with the characters personality information to see if there is a connection between their status and legitimacy in their universe and their personality traits. A screenshot of this page can be seen in *(b)* of Figure 4.8.

4.1.9.4 Personality Matches from other Universes

This web page matches corresponding heroes and villains to specific characters from film and television. This details a short study of characters from the Open-Source Psychometrics Project Statistical "Which Character" Personality Quiz.[9] The character's personality data was sourced

and put back into the test again to see which characters from other universes they would be most akin to. Finding the closest matches in both protagonists and antagonists provides a good insight to how similar personality types can lead to completely different world views, and how a film or television how can portray their status and legitimacy in different ways. This page consists of each characters name, along with a list of names and percentage matches in terms of antagonists and protagonists.

4.1.9.5 Character Personality Data according to their Universe

The pages entitled 'Game of Thrones Universe', 'Harry Potter Universe' and 'Lord of the Rings Universe' all follow a similar pattern.

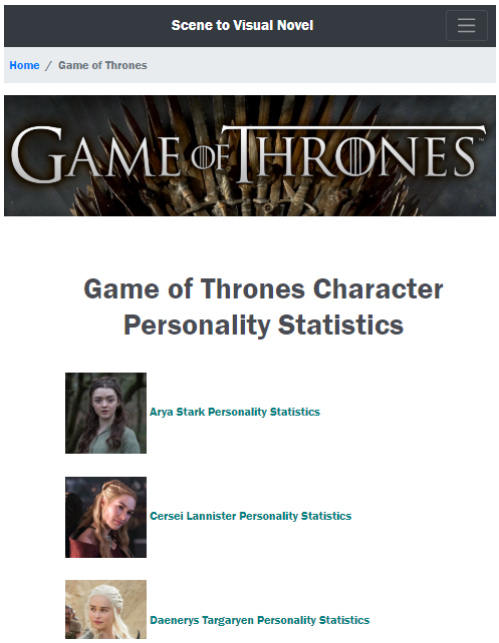
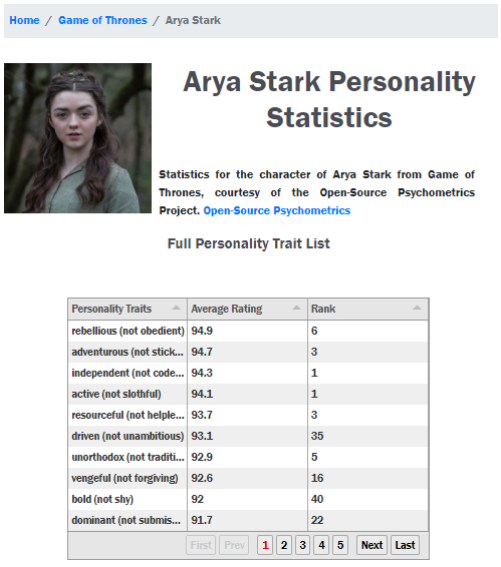


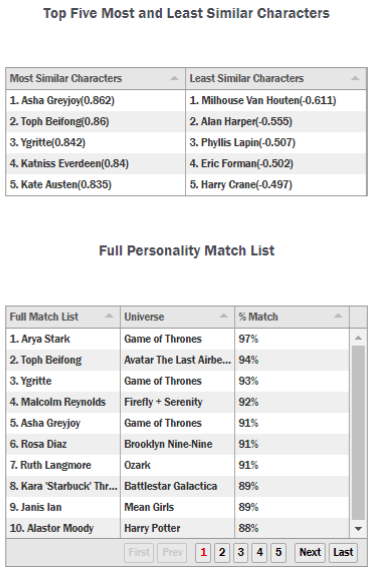
Figure 4.9: This figure shows the page entitled *Game of Thrones Character Personality Statistics* and displays some of the character personality statistics stored.

Each have a home page which has an image banner at the top of the page according to the universe, along with a breadcrumb: breadcrumbs are navigation elements which indicate a current location on the website and are particularly useful for these sections of the website. Each universe page has around five characters which are deemed to be the most integral to the story as well as portraying the major protagonists and antagonists in the universe. As the three universe pages follow the same layout, the Game of Thrones universe will be focused on to display the design details. The home page for the Game of Thrones universe is shown in Figure 4.9, and displays an icon of each character in question along with their name. Future development would allow an exhaustive collection of every character from each universe, but for the timeframe of the project, 5-7 characters were chosen for three different universes to give an insight to how the data could be collected and displayed.

When a user clicks into one of the character personality statistics from the universe home page, they will see something similar to that visible in Figure 4.10. As mentioned in the caption of Figure 4.10, the page begins with an updated breadcrumb to alert where the user is, and give them the opportunity to easily go back to the universe homepage to access other character's personality statistics. Each character page has an icon image of the character, as well as a brief introduction to the page. This is followed by three tables of data all taken from Open-Source Psychometrics: the first is the full personality trait list, followed by the top five most and least similar characters from all universes included in the quiz, and finally the full personality character match list. This final table was created by taking the first table and putting the results back into the Statistical 'Which Character' personality quiz to get a list of characters most similar to the character in question in terms of personalities. As mentioned earlier, this will hopefully allow a connection between how film/television portrays the status and legitimacy of a character and their personality traits: in one film a certain personality could be presented as a hero, whereas in another the same traits could portray a villainous character.



(a) Page entitled *Arya Stark Personality Statistics Part 1*



(b) page entitled *Arya Stark Personality Statistics Part 2*

Figure 4.10: These figures display the personality statistics for Arya Stark from Game of Thrones: (a) the breadcrumb at the top of the page alerts the user to where they are, followed by the full personality trait list. (b) displays the next two tables: the top five most and least similar characters from other universes, and the full personality character match list.

4.2 Software System Design

Software system design refers to the process of designing the elements of a system: this includes the architecture, modules and components, as well as the different interfaces of said components, and finally the data which travels through the system.[28] The main purpose of

system design is to ensure there is sufficient detailed information about the proposed system and its elements to enable implementation consistent with the proposed solution. Software architecture is defined by taking the software characteristics and converting them into a much more structured solution which fulfils your requirements and needs. These characteristics can include factors such as time, cost, maintainability, scalability, and many other traits. This section will describe how the software works at a high level.

In Chapter 2, the proposed solution was clear when it outlined a labelling tool was necessary to gather most, if not all of the metadata, and in turn a web application was needed which could store and display said data in a clear and concise manner. This consisted of a web application using Visual Studio with HTML, JavaScript and CSS. Tables were created via the Tabulator library for increased interactivity along with creating tidier structures to hold the metadata. The labelling tools itself was created via Amazon's Mechanical Turk via Amazon's Requester and used predominately Crowd HTML elements along with JavaScript. The MTurk tasks are published with a batch of HITS, in this case a set of images of scene frames which need labelled. Workers accept the HIT, fill out the form elements and submit their results. This then appears in the Results section of the Requester page, and from there the results can be approved or rejected depending on the quality of the answers. The approved results can then be downloaded in an Excel spreadsheet format for ease of future use.

The visual novel involved the majority of the difficult software design, despite the bulk of

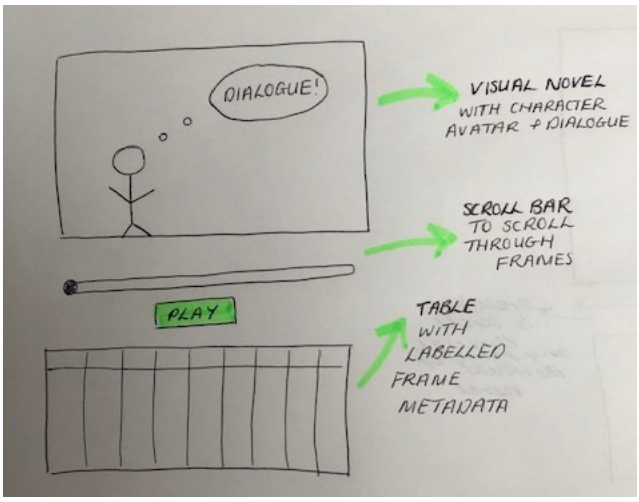


Figure 4.11: This figure shows a rough sketch of the proposed design of the visual novel at the planning stage.

the timeframe going towards gathering metadata. Figure 4.11 shows a first draft sketch detailing a visual novel structure which could be used for the final product. It includes a scroll bar function which can be used to scroll through the labelled frames and so the visual novel frames. The actual software design was more complex: the labelled data from either the MTurk tasks or from manual labelling needed to be in an appropriate format to be easily read into

the Tabulator table. The Excel spreadsheets were converted to JSON, standing for JavaScript Object Notation, which is a lightweight format for storing and transporting data. These meant that the data in the JSON object could be edited or added to easily, without having to redo code in the web page itself. The images for backgrounds and characters were also stored in JSON objects for reusability and maintainability, meaning new character/background images could be added at any point and the code itself would not need to be changed. The visual novel should work correctly regardless of the data used, providing it is in the correct JSON format, proving the visual novel an effective and reliable way to test how legitimate the results gathered are. The scroll bar was scrapped from the design due to feasibility, and for the purpose of demonstrating the effectiveness of the labelled results, the visual novel in its current form was deemed appropriate.

Tasks such as the gathering of vast amount of film and television scenes and converting the labelled data from Excel spreadsheets to JSON objects required the use of the Python language. The gathering of scenes was completed utilising the YouTube-DL library, which is a command-line program which is used to download videos from YouTube.com and other websites, with particular focus on the Fandango 'MOVIECLIPS' channel, which is the largest collection of licensed movie clips on the web.[29][30] Converting the labelled data to JSON objects utilised the 'pandas' and JSON libraries.[31][32]

Chapter 5

Implementation

5.1 Introduction

Given the Iterative development model style chosen for this project, the tasks were ordered in terms of their priority based on the requirements from Chapter 3 and are discussed in the order the functionality was implemented into the system. The integration and testing will be addressed at the end of the chapter, where the test cases are available in the appendices. Figure 5.1 displays the main technical standpoints used for the implementation of this project, and detailed explanations for each can be found in the sections below.

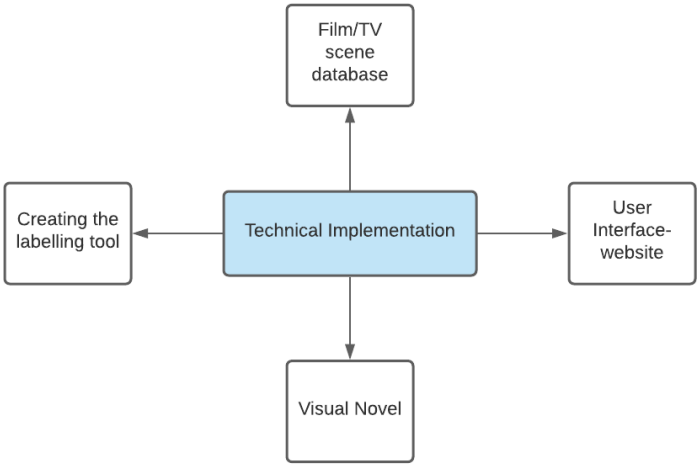


Figure 5.1: This figure shows the sections involving the maximum technical implementation in this project. The detailed explanation for both 'Film/TV scene database' and 'User Interface website' can be seen in Section 5.2.

Figure 5.2 portrays all the implementation factors which make up the entire project. The boxes in pink show the main focuses of this section, examining the higher level of stories, creating labelling tools, creating a visual novel, generating avatar images and finally testing. Each of these boxes have different boxes branching off that describe the further elements involved in each section, all of which are covered in more detail in the sections below.

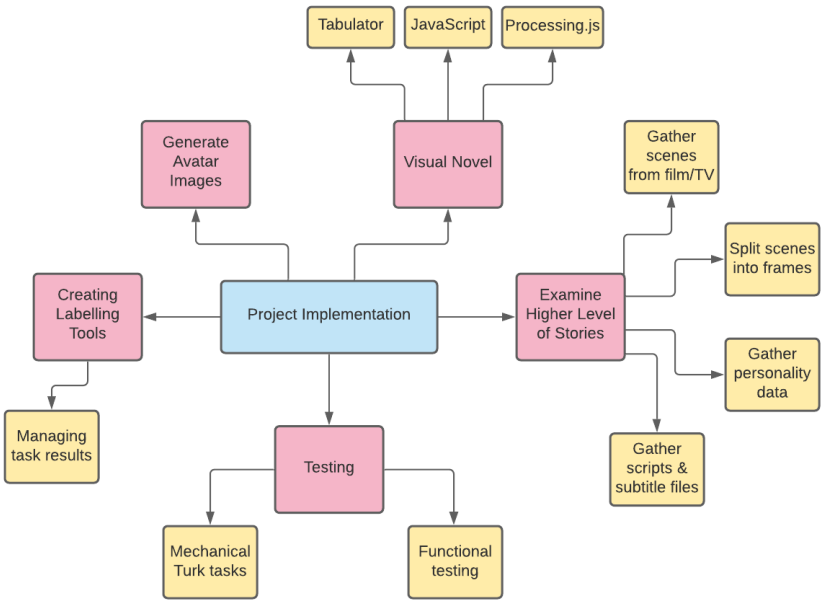


Figure 5.2: This figure shows a visual summary of the implementation details completed and shows at a glance how the different sub-sections contribute to the whole outcome.

5.2 Examine Higher Level of Stories

This requirement was deemed to have the highest priority for the project as it had the most impact in answering the question posed at the beginning: how film/ television convey hidden messages and emotions effectively, therefore it was implemented before the other functions. In fact, the other functions depended on accessing the information gathered due to this requirement. This required thinking about feasibility in terms of the project timeframe as well managing expectations: the aim is not to obtain a definitive collection of metadata, instead it is to demonstrate that from this minimal viable product that the some of the states of a scene can be gathered and proved to be important to its meaning and/or emotion. This requirement can be separated into a number of smaller requirements, the implementation of which is detailed below. All the Python scripts below were ran via a virtual environment set up at the beginning of the project involving Pip and Flask, where Pip is a package-management system which is utilised to install and manage software packages that are written in Python and Flask is a web framework, which provides users with tools, libraries and technologies to begin building web applications.[33][34]

5.2.1 Gathering Film/Television Scenes

The first main goal was to gather vast amounts of scenes from film and television so labelling could be performed on them. This was implemented through the use of a Python script which utilised the YouTube-DL library to get scenes from YouTube. [29] As mentioned in Chapter 4, a channel was found to have an extensive collection of scenes from a plethora of films, called 'MovieClips' on YouTube and this was used almost exclusively to download film scenes.[30]

The Python script used is shown in Figure 5.3, which was used to download videos from a specific YouTube playlist.

```
from __future__ import unicode_literals
import youtube_dl

def my_status(d):
    if d['status'] == 'finished':
        print('Finished!')
# checks status of download - particularly important if downloading a playlist
ydl_opts = {
    'progress_hooks': [my_status]
}
# This is where you input video or playlist link
with youtube_dl.YoutubeDL(ydl_opts) as ydl:
    ydl.download(['https://www.youtube.com/playlist?list=PL86SiVwkw_ofvEbosKlapvTrLs7geMKvp'])

```

Figure 5.3: This figure shows the Python script used to download video clips from the YouTube channel 'MovieClips'.

5.2.2 Splitting Scenes into Frames

Using a number of scenes downloaded via YouTube-DL, a Python script was employed to split the video into frames dependent on the set frames per second. The script to get the number of frames for a particular video can be seen in Figure 5.4, and the script for splitting videos into frames can be seen in Figure 5.5. The script iterated through all the videos in the folder and automatically saved frames as image files with the frame number attached. The frames were deemed necessary label scenes more accurately, rather than having MTurk workers keep pausing a video clip, leading to different workers labelling asynchronously.

```
import cv2

if __name__ == '__main__' :

    video = cv2.VideoCapture("FightClubIWantYoutoHitMe.mkv");

    # Find OpenCV version
    (major_ver, minor_ver, subminor_ver) = (cv2.__version__).split('.')
    # get number of frames
    if int(major_ver)  < 3 :
        fps = video.get(cv2.cv.CV_CAP_PROP_FPS)
        print "Frames per second using video.get(cv2.cv.CV_CAP_PROP_FPS): {0}".format(fps)
    else :
        fps = video.get(cv2.CAP_PROP_FPS)
        print "Frames per second using video.get(cv2.CAP_PROP_FPS) : {0}".format(fps)

    video.release();

```

Figure 5.4: This figure shows the Python script used to get the number of frames in a video.

5.2.3 Personality Data

At the beginning, the potential link between conveying hidden messages/emotions and personality of characters was undeniable. If certain personalities could be presented in different ways in terms of status and legitimacy within their universes, then it stood to reason that the personality of a character could then invoke similar effects within every scene. Exploring the highest level of stories could provide crucial answers. As mentioned earlier, the Open-Source Psychometrics Project, and in particular, their Statistical 'Which Character' Personality Quiz, was an integral source of information on fictional character personalities, the like of which has

```python
import os
import cv2
pathOut = r"C:/Users/amaye/Desktop/Youtube/frames/"
count = 0
counter = 1
listing = os.listdir(r'C:/Users/amaye/Desktop/Youtube/video')
# for video, get frames and create an image file for each
for vid in listing:
    vid = r"C:/Users/amaye/Desktop/Youtube/video/"+vid
    cap = cv2.VideoCapture(vid)
    count = 0
    counter += 1
    success = True
    while success:
        success,image = cap.read()
        print('read a new frame:',success)
        if count%30 == 0 :
            cv2.imwrite(pathOut + 'video%d'%counter + 'frame%d.jpg'%count ,image)
        count+=1
```

Figure 5.5: This figure shows the Python script used to split all videos in a file into their respective frames.

not been done before.[9] While perhaps limited to a number of popular films and television shows, the amount of data was still vast and would provide the perfect opportunity to use personality data in trying to answer how scenes can effectively convey status and legitimacy of a character, as well as conveying hidden meanings and emotions.

Some of the data was obtained through contacting the creators of the web application although most of the data was collected through manual means. Beginning by completing the quiz as a normal user, a full character match list was provided upon results, each of which was a link which held the full personality trait match for each character. As shown in Section 4.1.9.5, if the personality traits for one character were saved, the quiz could be redone using the traits of that character in order to find who they are most alike and different to from other universes. This provided a good insight into how similar personalities can be presented and validated, or not, in different manners. The results of the manual exploration of personality data is detailed in Section 4.1.9.4.

5.2.4 Scripts & Subtitle Files

Dialogue remains an integral factor which adds to how a scene can effectively convey messages and emotions, through the actual words spoken, the inflection the character's use, body language etc. Thus, scripts and subtitle files provided the perfect collection of dialogue, including acting cues which added another layer of information. Some scripts and subtitle files were manually searched for and downloaded but were deemed low priority compared to the rest of the components involved in creating a minimal viable product of the project.

5.3 Creating the Labelling Tools

As aforementioned, the labelling tools were created via Amazon's Mechanical Turk. Creating the labelling tasks was a challenge, and in particular, the first sprint of this task proved unsuccessful: in fact, no workers would work on the task. It was soon realised that it was impossible to ask any high-level questions which involved a significant amount of thought process,

instead it would be better to identify the smaller scale factors which could be easily labelled through a list of preordained options. When creating a new project, the properties common for all tasks created using it must be defined: this includes describing the task, a name, description and keywords, along with choosing the amount of time a worker has to work on one task, and how much they would be paid for each task. Throughout the labelling process, a total of £8.64 was spent paying Workers for the work they performed for the five tasks completed in full. Figure 5.6 shows the table of inputs wanted from the first MTurk task, and included expecting detailed paragraphs explaining the user's opinion on what occurs in the scene and getting the user to navigate to the film's IMDB page and copying and pasting the plot summary into the form. While this level of detail would be beneficial, taking the feasibility into account along with the disinterest MTurk workers had for this particular task proved this was the wrong plan of action.

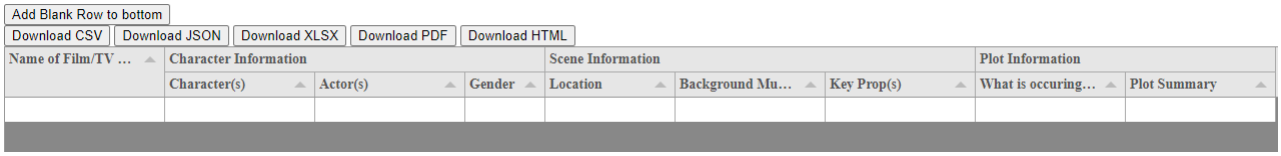


Figure 5.6: This figure shows the table detailing the user inputs wanted from the first Mechanical Turk task created.

Treating this as an experiment into the feasibility of obtaining the best answers possible, the next step was to create a labelling task which took the literal detail of the scene. This was separated into two main task templates: one to label character's physical appearances such as their hair, eyes, attire etc, and the other to label the detail of the scene dependent on one character, such as the character's actions and dialogues, as well as any sound effects or props used. There was a degree of trial and error required when publishing a batch of HITs for each task: the instructions for the tasks needed to be as clear as possible, even if they seemed like simple common sense. The best MTurk tasks appear to be one worker can do easily, quickly and without much thought.

Five scenes which involved a conversation ending in different manners, such as a kiss, hug, fight etc. were used for the tasks. The labelling task for character appearances was performed for the main characters (had screen time and said words) from the scenes: the worker was provided with a screenshot/ frame of the scene which contained a clear image of the character and were asked to label it. This labelling task involved following a link to a page which much like that of that mentioned in Section 4.1.8, where the user could choose the properties of the character using an edited Character Creator version (with reduced options), before pressing the 'Character Finished' button and copying and pasting the values which appeared in the table below to the MTurk form for submission. This task had a couple of iterations before successful quality results were obtained. The labelling task for the scene frames were performed on one of the videos which contained two characters. The scene was split into frames and split

into two tasks, so the task size was more manageable. The workers then had to label for the female character and the male character separately. The form had an image of the character needing labelled at the start, followed by the image of the scene frame. There was also a link to the scene video, which played with a significant pause after each of the frames so the workers could easily distinguish what occurred in which frame. Again, a few iterations were performed before the results were significant. See Appendix D.1 for a screenshot of two of the tasks created.

5.3.1 Managing Task Results

Once a worker had submitted a HIT, their results would appear on the Amazon Requester page ready for approval or rejection by the task creator. An example of this can be seen in Figure 5.7. These results needed to be examined and evaluated to ensure accurate/significant

	Blouse	Complexion	Dress	Earrings	Eye Colour	Facial Hair	Gender	Glasses	Hair	Hair Colour	Hat	Jacket	Makeup	Pants	Scar	Shirt	Shoes	Skirt	Tattoo	Tie	Vest
annamy/...	no	white	t-shirt	10	black	white	male	good	boy cut	merun	nice	no	100	no	no	merun	no	no	n/a	no	no
annamy/...		e5c298			e5c298		m		e5c298	e5c298						e5c298					
annamy/...		#f0d5be			#a6bc51		m		gelled	#c4a61e				jeans		tshirt	hightops				
annamy/...		#e79e6d			#e79e6d		m		down	#1a1a1a		suit		jeans		tshirt	hightops				
annamy/...		#e1b899			#2c2b2b		m		gelled	#1a1a1a		suit		jeans		tshirt	hightops				
annamy/...		e5c298			e5c298		m		e5c298	e5c298		e5c298		e5c298		e5c298					
annamy/...		#a3866a			#1a1818		m		wavy	#504d4d		suit		suit		colar	leather			neck	
annamy/...	na	brown	suit	na	black	na	male	na	casual	brown	na	na	na	na	na	shirt	yes	ma	na	yes	yes
annamy/...		#eeceb3			#83571a		m		gelled	#514f4f		suit		suit		colar	leather			neck	
annamy/...	blue	#8564	accolate	earring	#553220	black	mail	cooler	down	#1h1h1h	ring		no	suit	scar	white	higheels	school	tatto	black	vest
annamy/...	white	talked with lady	black		brown	kurly	Male	napkin		dark		black		black		white	black	black		pink	black
annamy/...	n/a	#704139	n/a	Yes, small ----	#552200	goatee	m	n/a	crewcut	#1a1a1a	n/a	suit	n/a	suit	n/a	colar	n/a	n/a	n/a	neck	n/a

Figure 5.7: This figure shows a table of results for MTurk task.

results were being provided by the workers, for example, often a worker would answer one input on the form and submit, hoping to get paid despite not labelling anything. The figures in Figure 5.8 show examples of the feedback provided for incorrect results.

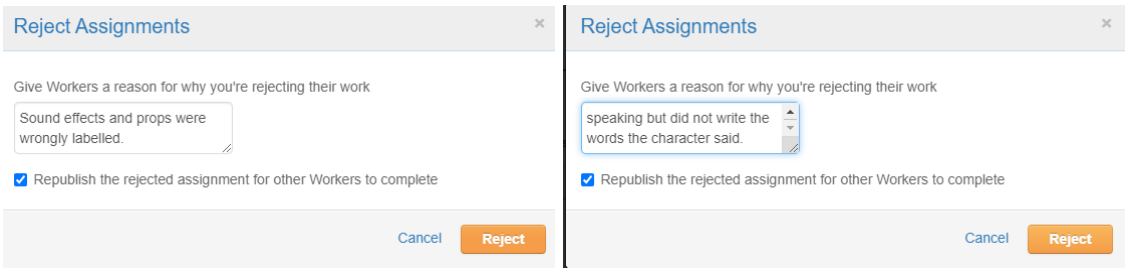


Figure 5.8: This figure shows two examples of feedback provided when rejecting a workers results.

Once all the HITs had been completed and approved, an Excel spreadsheet of the results could be downloaded from the Requester site. Figure 5.9 how the spreadsheet is laid out, and it includes the properties set for the project, i.e. title, time allowed, amount to be paid etc. As mentioned in Section 4.2, the result spreadsheets needed to be converted into JSON objects for ease of future use, this was the same for the manual labelling performed. The Python

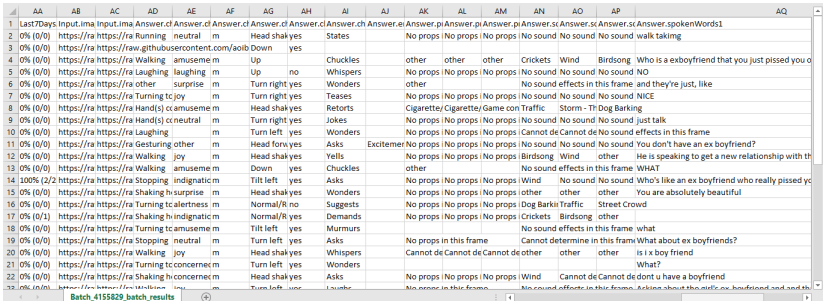


Figure 5.9: This figure shows a screenshot of the spreadsheet of results for MTurk task.

script used to do this is shown in Figure 5.10. Due to the spreadsheet having multiple columns which did not contain the labelled results, it was better to ask the script to find the columns which contained the answers needed and use that to create the JSON object. As most of the tasks performed had at least three different workers employed, duplicate results were obtained. It is up to the discretion of the project creator how to sort through these, in this project, the most accurate labelling for each image was selected based on the results that were manually labelled.

```
import pandas as pd
import json
# read in excel spreadsheet
df = pd.read_excel(r'C:/Users/amaye/Desktop/aoibheannamy.github.io/turkResults/CharacterLabelledResults.xls')
# read in the data specifically from the columns listed
df1 = pd.DataFrame(df, columns=['Names','Answer.blouse','Answer.complexion','Answer.dress','Answer.earrings','Answer.eyeCol

#print to ensure they are correct
print(df1)
#convert spreadsheet to json object, oriented by row
json_str = df1.to_json(orient='records')
print('Excel sheet to JSON:\n',json_str)

# here we write the data from the Excel file to a json object
with open('turkResults/CharactersLabelledDetails.json', "w") as file_write:
    # write json data into file
    json.dump(json_str, file_write)
```

Figure 5.10: This figure shows a screenshot of the spreadsheet of results for MTurk task.

Once the results had been narrowed down and converted to JSON notation, they were ready to be utilised.

5.4 Visual Novel

The visual novel utilises the JSON objects obtained from labelling as well as JSON objects created with image links (which allows the opportunity for new images to be added in at any stage without changing the code). The visual novel itself depends on a range of JavaScript libraries and frameworks including Tabulator, Processing.js and vanilla JavaScript.

5.4.1 Tabulator

Tabulator is used to create the table which hold the labelled frame metadata in a similar way to the rest of the Tabulator tables mentioned in this project. The table column row editors

are set to 'input' so in the future, the visual novel could become interactive where the user could input whatever they would like, and it would appear in the visual novel. The rowIndex is defined in the code for the table, as shown in Figure 5.11 and it is used to iterate through the frames.

```
rowClick: function (e, row) {
    rowIndex = row.getIndex();
    // displays index
    console.log("Row index: " + rowIndex);
}
```

Figure 5.11: This figure shows the function to get the rowIndex from the table.

5.4.2 Processing.js

Processing.js creates the canvas of the visual novel and makes a draw call which in turn renders the images for the characters and the backgrounds, as well as any dialogue taken from the table. As seen in the figures in Appendix E.1, the p5 JavaScript graphics library was involved in the *setup* function of the canvas, the *draw* call function, and the *playVisualNovel* function.

1. *Setup:* This function is required in every p5 script, and is called once when the program begins. It is used to define initial properties such as screen size and background colour and to load media such as images as the program starts. As seen in Figure E.1, the canvas size and position is defined, as well as loading the background and image as a placeholder image. Finally the container which holds the canvas is set.

2. *Draw Call:* This function is also required in every p5 script and is called directly after setup function. This function continuously executes any lines of code within its block. As seen in Figure E.2 the canvas is cleared at the beginning of each draw call to reset the images, then the background image is loaded. If there is a background image present, the draw call draws a speech bubble shape, followed by declaring the image (which is set in the *playVisualNovel()* function). Finally the text fields are declared along with their colour and size.

3. *Play Visual Novel:* As seen in Figure E.3, the play button below the canvas has an onclick event which calls this function. It calls all the functions required to set the image files and dialogue (these are explained in Section 5.4.3). Finally the rowIndex is incremented by one in order to move to the next frame in the table. The if statement at the end of the function checks if the rowIndex is less than the number of rows in the table, and if so, the function is called recursively with a six second delay. This ensures the user can observe the differences made between each frame before it moves to the next one. If the rowIndex is greater than the number of rows, then the visual novel has finished and an alert signifying the end appears.

5.4.3 JavaScript

This section focuses on the JavaScript used outside of the aforementioned. There are three main script used in the visual novel: *getBackgrounds*, *getCharacterImage* and *getDialogue*. The scripts for these are available in Appendix F.1.

1. *getBackgrounds.js:* Figure F.1 shows the script responsible for setting the background image of the visual novel. First the JSON object containing the links to each background image is stringified before being parsed. Then dependent on the dropdown list on the main HTML page, the user's selection will be used to match the entry in the JSON object, resulting in the matching background image being loaded. This has good opportunity for development: if other background options wish to be included, the selection dropdown and the if statement need only be updated, along with adding a new image link to the JSON file. This function is called before the *setup function* to ensure the correct background is loaded.

2. *getCharacterImage.js:* Figure F.2 shows the script responsible for loading the image of the character in the frame. As with the previous script, a JSON object of character image links is stringified and parsed. Before this, the frame table data is accessed, and the character number and their emotion are defined according to the current rowIndex. These two variables are then used in a switch statement to match the entry in the table to an entry in the JSON object, resulting in the correct image being loaded for each frame. This function is called in the *playVisualNovel() function* and the iteration of rowIndex ensures the image changes respective to the frame data.

3. *getDialogue.js:* Figure F.3 shows the script responsible for loading the dialogue of the character speaking in the frame. Similar to above, the frame table data is accessed and the character number, the words spoken, and the character's inflection are set as variables dependent on the rowIndex. The words the character speaks is set as *textWords* which is called in the *draw call* function. The if statement checks if the character number is not zero (signalling no character is speaking at that point), and then sets the descriptive text which will appear in every frame, else it will display no character is speaking in said frame. This function is also called in the *playVisualNovel() function* and the iteration of rowIndex ensures the dialogue changes respective to the frame data.

5.5 Generate Character Avatars

As introduced in Section 4.1.8, this page edited the web application Character Creator to utilise for this project, as was done with the MTurk labelling task for labelling character appearances. [12] This page reads in a JSON object containing the character appearances from the MTurk task into a Tabulator table. Above the table is an iframe which contains the code from the Character Creator, which had small edits to remove all dropdown options. A modal

was introduced to give users instructions on how to utilise the page effectively: the user must click on the row they wish to generate, then click the 'Generate Character Avatar' button. This loads the corresponding avatar image. To make a new selection, the user must click the 'Clear Character Avatar' button, then repeat the process. The labelled parameters are taken from the selected row and pushed into the iframe's URL to produce the avatar image. The code used for this page is show in Appendix G.1, and a couple of errors appear due to some of the changes made with the charactercreator.org code, however this was deemed acceptable for the timeframe of the project. In reality, a project specific avatar creator would need to be written.

5.6 Testing

Testing in a software development project is imperative to ensure the system is suitably validated and verified by the requirements. Thorough testing also highlights defects or bugs in the software code, so ensuring the developer becomes aware of them and in turn can rectify the errors. Testing was performed throughout the project's lifecycle ensure reliability and high performance. Software development testing can take on two forms: black box or white box. Black box testing is defined as testing in which the internal structure and implementation of the item being tested are not known to the tester and involves providing inputs to verify the outputs against what the expected outcome is.[35] Whereas white box testing is the opposite: it involves a method where the internal structure and implementation of the product being tested is known.[36] For this project, the user experience remains the highest priority, thus testing will focus solely on black box testing. Throughout the development cycle regular testing was operated, predominately on the visual novel, through the use of try-catch statements to make the developer aware of exceptions appearing in the code, therefore increasing the efficiency of fixes.

5.6.1 Mechanical Turk Testing

As mentioned in Section 5.3.1, the MTurk tasks required some trial and error experimentation: testing the effectiveness of the instruction given to workers led to adapting the tasks until the expected outcome was achieved. Also the quality and significance of the results obtained in the MTurk tasks were evaluated based on manual labelling, and the results were approved or rejected respectively.

5.6.2 Functional Testing

Functional testing is a form of black box testing which comprises of using the requirements as test basis and are generally written as if from a user's point of view. This method of testing is a technique to ensure the system operates as expected, and usually involves entering input data to check the output against the expected outcome. Each section of the web application

created for this project was tested through the creation of test cases. Test cases are made up of a test case ID, requirement or objective, input and/or action needed, the expected results and the status of the test. As functionality evolved during the lifecycle, testing evolved alongside to reflect the changes made. No stress or load tests were performed as it was assumed that GitHub Pages could withstand large numbers of users at one time. The test cases performed can be seen below in Figures 5.12 and 5.13. These tests were chosen to be performed to test the mandatory and high priority requirements set in Chapter 3 to ensure the project met its set specifications.

Test Case ID	Requirement	Action	Expected Result	Result Status
1	The application must be publicly accessible on the world wide web.	Navigate to https://aoibheannamy.github.io/	The website should load correctly.	Pass
2	The front end must have a consistent and user friendly user interface.	N/A	The color scheme, layout and general design should remain consistent throughout the site, with no issues with UI being reported during user testing.	Pass
3	The application must be easy to use for newcomers.	N/A	Users should report no issues with on to use the application.	Pass
4	The visual novel must match the video representation of the scene.	N/A	The visual novel characters and dialogue should match that of the video scene.	Pass

Figure 5.12: Table detailing the test cases and results for the non-functional requirements.

The non-functional requirements proved simple to implement as they revolved on general user experience, whereas the functional requirements required more in-depth testing. The layout of the web application was ensured throughout through the use of a single CSS stylesheet, and this was verified by careful visual combing of the website, and the use of simple navigation bars and breadcrumb formats allows users to navigate quickly and easily. The user-friendly requirement prompted the use of modals as popups to provide users with instructions on both the visual novel and the page to generate a character avatar. This ensured that users would explicitly know what to do and ensure that they did not feel frustrated.
The functional requirements were tested via user acceptance testing, checking if the user use the software, if they have any trouble using it and does the software behave exactly as it is anticipated. This ensured that the user could access all the functions promised by the web application with no issues.

Testing is beneficial at all stages and allows key issues to be resolved which may have been missed otherwise. The visual novel encountered issues caught by testing, which was expected due to the extra complexity involved with real-time applications like visual novels. One such issue was the dialogue aspect of the visual novel was displaying the wrong information despite the code appearing correct. This allowed the code to be altered so that the character image and dialogue were only loaded when the row changed, rather than every time the frame was

Test Case ID	Requirement	Action	Expected Result	Result Status
1	The system must allow user access to all webpages.	Navigate to every web page on the website	Users should have no issues visiting and viewing every webpage.	Pass
2	The system must allow users to play the visual novel.	Press "play Visual Novel" button.	The visual novel should begin to play on button press, and continue until the scene ends.	Pass
3	The system must allow users to pick a background for the visual novel.	Select background from dropdown list.	The visual novel should load the corresponding image to the users choice.	Pass
4	The system must allow users to save images of labelled character avatars.	Click row in table, then click "Generate new avatar" button and choose to download image.	Avatar image can be saved as a png or svg file.	Pass

Figure 5.13: Table detailing the test cases and results for the functional requirements.

drawn. The error resulted in the code also being updated to increase reusability for future development, making the final product much more significant.

Chapter 6

Evaluation and Conclusion

6.1 Introduction

This chapter provides a detailed evaluation and conclusion on the success of this project compared to the criteria mentioned in Chapter 1 and Chapter 2. A critical evaluation of the technology utilised in implementation is provided along with a discussion of potential areas for future development, including those which could be improved upon as well as concepts for new features for future iterations. Finally, a conclusion will be drawn on the success of the processes followed in the project.

6.2 Evaluation of Success

The aim and narrative of this project was reiterated throughout this dissertation: the goal was to take a very grand ambitious project and create the minimal viable product to prove if the grand version could become reality in the future. Attempting to parameterise film or television scenes in order to understand what factors go towards being able to effectively convey the hidden messages and emotions of the story is a huge step in beginning the steps of formalisation. Formalising everything from dialogue, character appearances, personalities, body language, backgrounds and many more, allow an insight on understanding what combination of factors takes a story from baseline to exceptional. However, it was never the goal to fully answer how film does this, only to prove that the formalisation can be done, even at a small scale.

6.2.1 Mechanical Turk

The goal of this project was to gather multiple scenes from film/television, create a labelling tool to determine the properties which effect how emotions and underlying meanings are conveyed, and to create a visual novel to test the effectiveness of the labelled data in terms of a small scale viable product. The project successfully gathered vast numbers of film scenes and demonstrated that a successful labelling tool could be created which would get workers to label the very literal detail of a scene or character. The trial and error involved with the MTurk tasks was discussed in Chapter 5, but it heightened the learning curve and demonstrated that

asking workers to label intensively was a challenge, and it would be more suitable as well as time and cost effective to narrow the labelling scope. Evaluation of the labelled results obtained from each MTurk task proved that around 80% of the gathered results captured the frame in the scene effectively. This confirmed that the workers on MTurk could label with significant precision, and so it stands to reason that it remains a viable application to develop this project further.

6.2.2 Web Application & Visual Novel

The web application stores most of the personality information which was manually gathered from the Open-Source Psychometrics Statistical 'Which Character' Personality Quiz.[9] Although the scope and timeframe of the project did not allow for any real data analysis to create personality maps and connections, the presence of the data itself in an aesthetic and easy to use layout was a good beginning. The visual novel was implemented as a test of the significance and effectiveness of the results from manual and MTurk labelling. The coding practices used, along with the libraries and frameworks allow simple reusability for future development. A simple minimal version of a visual novel was developed to test the effectiveness of the labelled data and successfully portrayed this. Future development could allow the visual novel to become a more interactive version in which users could input their own frame information, and even have their personality validated through options that would have been generated from in depth information and analysis from multiple film scenes. The developed visual novel plays through the labelled frames and represents the changes in character, character emotion and dialogue effectively. Implementing additional features from the labelled table such as props, actions and sound effects would definitely make the visual novel mirror the scene itself more accurately and would add to proving the effectiveness of the labelling completed and remain a good opportunity for future development. As the Character Creator was utilised to produce the character avatars employed in the visual novel, that part of the project relies on the coding of others¿ This could be a factor for future development however, where the web application had its own avatar creator where a user can create multiple characters.

6.2.3 Findings

A number of findings were uncovered over the course of this project, and some are listed below.

- The difficulty involved in asking people to label elements, no matter the category.
- The challenge involved in writing coherent and definitive instructions for users to follow which have no room for misunderstanding.
- Ensuring labelling tasks are simple enough that any person at any level could record: it is easy to overcomplicate because it can be done manually.

- It is better to load data into a draw call only when the row changes, rather than having it render multiple times in every frame.
- Creating customisable avatars is a challenging feat and requires in depth knowledge of SVG elements, hence why the Character Creator was utilised for demonstration purposes in this project.

6.2.4 Reviewing Useful Outputs

Reviewing the potential useful outputs declared in Chapter 1 brings the project to a full circle by evaluating the amount of work accomplished.

1. *How-to Guides:* Three separate how-to guides were mentioned, one for the set-up of Python/Pip on slow computer systems, one for Amazon's Mechanical Turk and the last for GitHub Pages. These guides were completed from the experiences during the project and provide a handbook for any interested parties in the future.
2. *Storing Character Personality Traits:* As mentioned throughout the project, personality data was taken from the Open-Source Psychometrics Project and Myers Briggs pages to store on the created web application. The website does not hold a definitive collection of data but instead shows how the data would be displayed for users to utilise. Future development would grant further data scraping and input from Mechanical Turk tasks to produce an exhaustive collection. Taking the statistics from the Open Psychometric Personality Test and storing them in a easy-access structure such as using JSON objects and storing them in a JSON library, where anyone could access would allow people to attain full character spectra.[9] As an opportunity for later stage development, Mechanical Turk workers could take the stored characters and use the traits and process that was used manually to add to the list, adding new fictional characters from new worlds and universes to create a more detailed representation of how personality effects characters and stories. In this section, the visual characteristics of the characters could be recorded as well as providing a more thorough overview of them respective to their physical appearances, job occupation etc. Combining this begins to address the question posed at the beginning of the project at a deeper level.
3. *Personality Map:* Creating a personality map of characters could allow others, such as game developers or writers, to immediately utilise a set of personality types and therefore the information to create characters and their respective interactions accordingly. It would allow the easy portrayal of the sequences of actions/choices that are necessary to make or break a character (become good or become bad). If the player can feel like their own personality is validated through the game and its corresponding choices, then games will be able to have the same emotional connection that films provide, leading to

more successful games. Mechanical Turk workers could take this map and add new characters to it as mentioned above. This output requires curation by more expert minds to discover the intricacies and bonds between different personalities in order to begin generalisation to create a useful personality map, and remains a significant opportunity for future development.

4. *Storing Scenes from Film/Television:* The collection of multiple scenes from film and television was successful through the use of YouTube-DL and Python. The methods used to gather these scenes can be easily adapted to continue efforts to produce a database collection of scenes from film and television. Collecting more scenes and storing them in a dedicated YouTube channel would allow easy access for film/ television writers or game developers. Organising scenes together in specialised sections, e.g. love/ romance development, hero realises something about themselves, villain backstory etc. would allow an archive of specialised information to be available, dependent on what the person is looking for. Storing this information in a YouTube channel would allow anyone to access it, and the structured layout in terms of the theme of the scenes would aid quick searches.

5. *Formalisation of Scenes:* The formalisation of scenes was shown to successfully work for a minimal viable product during the course of this project. The methods utilised in the project, including manual and Mechanical Turk labelling are easily adapted for continual labelling of scenes to begin formalisation. If this project is reiterated for a less minimal product, the labelling forms can be easily updated to include new properties or factors, creating a taxonomy of what makes a scene effective.

6. *Storing Important Dialogue from Film/Television:* This output was deemed a lower priority in terms of collecting scripts and subtitle files as they too require a more expert curation, however the successful labelling of dialogue within scenes was proved through MTurk labelling tasks. This output has significant potential for future development. Combining this with the YouTube channel of scenes, this would allow users to see how film/television writers create the clever riposte and the emotional impact behind characters words and actions. This could also be used to simplify dialogue to the most important parts and then a simple visual novel with a small number of scenes and corresponding dialogue could be created to see if the correct/same impact is received by the player as would be expected from the dialogue if it were in a film/show instead. Mechanical Turk workers could label how the other character receives the dialogue and their emotional responses.

7. *Character Avatar Customisation:* As aforementioned, the character avatar creation was significantly more complex than expected, therefore the code from the Character Creator was edited and used to create the avatars used in the visual novel. Future develop-

ment of a project specific avatar customisation application would be ideal so that a user could create their own character avatar and use it within their interactive visual novel.

Overall the project is deemed a success in how it proved the parameterisation of film scenes could be achieved on a small scale in order to answer how those scenes effectively convey hidden messages and emotions, and to gather labelled information which held significance.

6.3 Evaluation of Testing

The testing process mentioned in Chapter 5 involved mainly functional testing due to the emphasis delivering the highest priority functional requirements as well as on general user experience. On the occasion that more time was allocated to the testing process, the black box testing could have been more thorough, as well as implementing more white box testing to potentially find other issues from within the system. However, at the moment, the completed testing meets the requirements and has been carried out to an acceptable level for this project.

6.4 Technologies

A range of different languages, libraries and frameworks were utilised in completing this project, most of which were new concepts and required a learning curve. However, they were all chosen for their broad range of documentation which aided the learning process as well as provided the all-important ability to increase reusability of features for future development. Learning Python, albeit for small specific scripts, was perhaps the most difficult, but was worth the ease at which scenes could be downloaded and split into frames along with conversion of spreadsheets to JSON objects. Overall the technologies used were effective and would be recommended for future development.

6.5 Future Development

This project has vast potential for future development, with many of the features having been mentioned in this dissertation in passing already. There are two main opportunities which are suitable for expansion: the visual novel and the labelling and parameterising of scenes.

1. *Repeating the project for a less minimal product:* This project focused on producing the minimal viable product by labelling the literal details in a scene. The personality data gathered could be developed further by allowing users to add in new fictional characters in order to get a huge and diverse taxonomy. Future development could be advanced through one of two broad ways: repeating the labelling tasks produced for this project but for multiple film scenes and/or create more focused labelling tasks which concentrate on higher level details. The first involves a lot more time and money being involved: taking the tasks performed in this project totalling approximately £6.50 for one scene with two characters (each HIT (Human Intelligence Task) was priced at $0.02, the

end cost would depend on the number of scenes labelled, as well as the number of frames and main characters needed labelling within them. However this would allow scenes to be generalised and multiple taxonomies to be created, each of which would be useful for a myriad of people, not to mention getting more accurate answers to how films convey hidden message and meanings. The second has the same time and cost effects but involves splitting the projects into more focused tasks. These could include, but are not limited to, labelling ways in which opposing personalities can be shown as illegitimate, the themes such as love, family, power etc.) that most strongly reveal the strengths of the personality, the personality or world view that film presents as most legitimate, the career or life situation at corresponds to feelings associated with a situation and the dialogue and general body language and the escalation they are part of. While these are almost infinitely more complex labelling tasks, the potential they represent for creating useful and informative taxonomies is astronomical.

2. *Making the visual novel interactive:* One of the low priority requirements of this project was to create an interactive version of a visual novel, however this was not feasible. This presents itself as an opportunity for future development. As aforementioned, the visual novel could be adapted so that a user could choose the data which makes up each frame of the scene, including the character's appearance, occupation, personality etc. and so be able to play through the scene, only to be presented with a choice at the end of each scene. his choice would be generated from the parameterisation and analysis for the plethora of information gathered in the above point to generalise scenes and personalities in order to know how to validate certain world views and choices. This would be the perfect test for the effectiveness and significance of the research performed in the future, and to test how well the hidden meanings and emotions are invoked and conveyed. Aside from this, as the avatars in the visual novel were created via the labelled character results put into the Character Creator website, a potential task for the future would be to create the project's very own avatar creator so that the images are tailored especially for what the user wants, and these can be utilised in the visual novel.

6.6 Conclusion

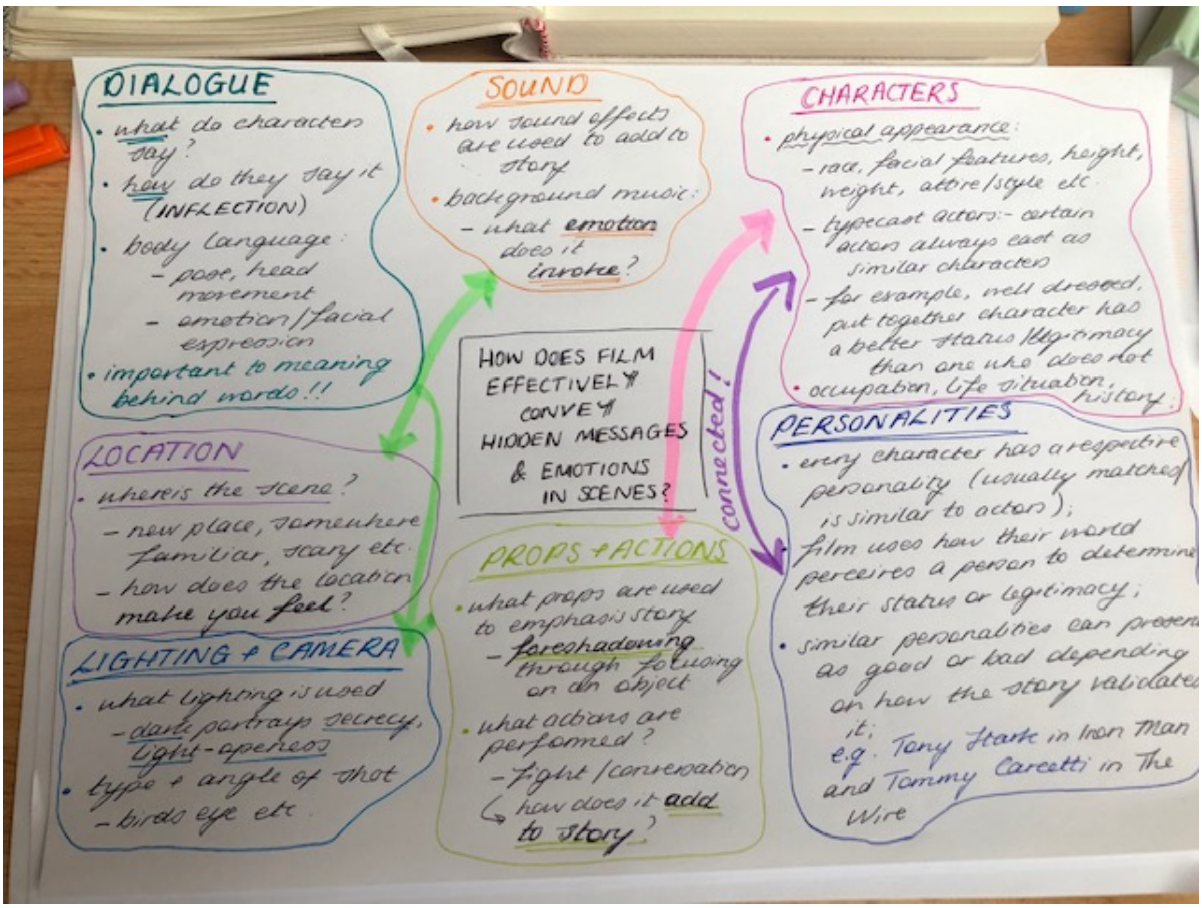
To conclude, the labelling tool created via Amazon's Mechanical Turk provided a valuable insight into the type of data which is feasible to gather and what is more likely to require curation by an expert. The successful MTurk tasks proved that the very literal data of a scene and character appearance was the most MTurk workers could provide: this includes data which is distinctly visible in the scene, such as props, character emotion, hair colour etc., or clearly heard such as any sound effects utilised and the character's dialogue and inflection. Anything which required a more detailed thought process was either too in-depth for MTurk workers or would require the rewards set to be much higher. This includes labelling of a character's

personality, how the story shows the personality as legitimate or illegitimate, the choices and problems that enable characters to show their strengths and weaknesses etc. The visual novel was produced to become a test of the significance of the gathered data, as well as showing how effective the data could convey the same emotions and/or underlying messages the scene itself conveyed. Based on the character avatars and dialogue alone, the visual novel proved it could portray this to an acceptable degree. This proves that MTurk is a useful tool to use for formalisation of scenes, and the process and methods utilised in this project only require a small number of alterations to replicate the project's results for a less minimal product. The potential this project created for future development is both undeniable and impressive.

Appendices

Appendix A

A.1 Diagram outlining Thought Process



Appendix B

B.1 Functional Requirements

ID	Requirements	Priority
	User	
1	The system must allow users access to all webpages.	Mandatory
2	The system must allow users to play the visual novel.	Mandatory
3	The system must allow users to add new information to tables.	Inessential/ Desirable
4	The system must allow users to save images of labelled character avatars.	Mandatory
5	The system must allow users to adapt/change the data producing the visual novel and process these changes.	Inessential /Desirable
6	The system must allow users to pick and choose their characters: physical appearances, occupation and personality.	Inessential /Desirable
7	The system must allow users to have their own or chosen personality validated.	Inessential /Desirable
8	The system must allow users to pick and choose scenes and what happens in each.	Inessential /Desirable
9	The system must allow users to access choices after each scene which will continue the storyline dependent on their decision.	Inessential /Desirable
	Mechanical Turk Worker	
10	The system must allow Workers to access each HIT.	Mandatory
11	The system must allow Workers to view all media included in a HIT.	Mandatory
12	The system must allow Workers to complete and submit each HIT.	Mandatory
13	The system must provide Workers with clear instructions to carry out each HIT.	Mandatory

Appendix C

C.1 Non-Functional Requirements

ID	Requirements	Priority
1	The application must be publicly accessible on the world wide web.	Mandatory
2	The front-end must have a consistent and user friendly user interface.	Desirable
3	The application must be easy to use for newcomers.	Desirable
4	The visual novel must play at the same time as the video representation of the scene.	Desirable

Appendix D

D.1 MTurk Task Example 1

Instructions: Click the link below to review the website and collect the following information if it's available.

We are conducting an academic study to parameterise film and tv characters physical appearances for the creation of visual novels.

We would like you to look at the following image of a film/tv character and use the link below to create a corresponding avatar to represent them.

Select the link below to complete the study.

- Don't worry about changing the colours of the attire/accessories.
- **Please choose the closest approximation with the options provided:** e.g. if the specific clothing option is not available, choose something that would portray the same effect i.e. casual wear would be a sweatshirt.
- When you are happy with the representation of the character you have made, click the **"Character Finished!"** button and data will appear in the table below.
- Copy **exactly what is in the table, e.g. for eye colour it will appear like #552200.**
- Paste entries into the form in the corresponding places.
- If any column in the table is blank, **leave that part of the form blank.**
- Do **not** fill out the form with anything bar the data from the table.

Link to the Website:

https://aoibheannamy.github.io/

e.g. accolade

Tie:

e.g. bow

Vest:

e.g. vest

Jacket:

e.g. suit

Trousers:

e.g. suit

Glasses

Hat

Earrings

Makeup

Tattoo

Scar

Submit

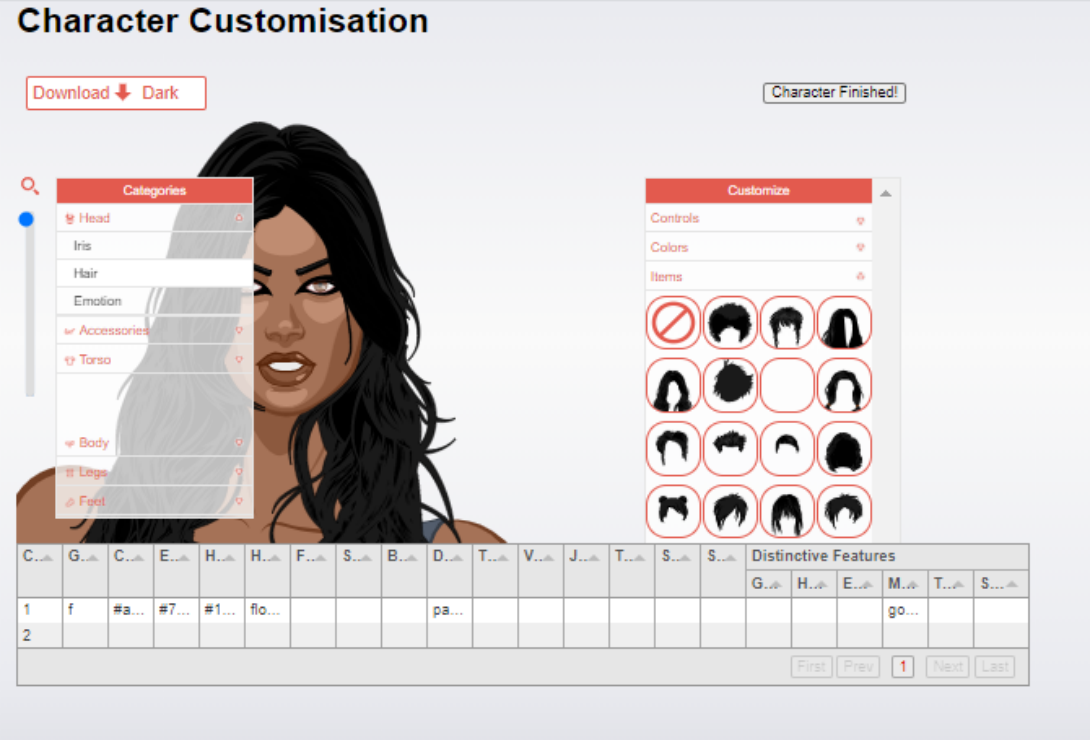


Figure D.1: Screenshot showing the layout of the character appearance labelling task.

Appendix E

E.1 Processing.js Screenshots

```
/**
 * This sets up the canvas for drawing
 */
function setup() {
    // create the canvas including the size, dependent on the size of the window
    var canvas = createCanvas(5 * windowWidth / 6, windowHeight);
    // position canvas
    canvas.position(50, 100);
    // display as block
    canvas.style('display', 'block');

    // set background as placeholder image to begin with
    bg = loadImage('../Images/placeholderImage.jpg');
    // set img as placeholder image to begin with
    img = loadImage('../Images/transparentImage.png');

    // set container to hold the canvas
    canvas.parent('sketch-holder');
    textAlign(CENTER, CENTER);
}
```

Figure E.1: Function to setup canvas for visual novel.

```
/**
 * function to render the draw call of canvas elements
 */
function draw() {
    // clear canvas after every draw call
    clear();
    // make sure the background image has been loaded before continuing
    if (background(bg)) {
        // draw a speech bubble
        fill(290);
        stroke(0);
        ellipse(430, 150, 250, 160);
        // declare image as loaded img
        image(img, 0, height / 2, 3 * img.width / 4, 3 * img.height / 4);
        // declare text fields and set colour and size
        fill(0, 0, 0);
        textSize(20);
        text(textWords, 330, 80, 200, 200);
        fill(0, 0, 0);
        textSize(16);
        text(charSpeaking, 430, 130);
    }
}
```

Figure E.2: Draw call function for visual novel.

```
// call in function to iterate through the table when play button is clicked
document.getElementById("play-button").onclick = function () { playVisualNovel() };
/**
 * This function calls the functions to get the character images dependent on emotion
 * as well as to get the character's dialogue for rowIndex before slowly iterating
 * throughout all the table rows, with a 6s delay between each frame
 *  When play button is clicked, the visual novel will play - iterating through every row in the table
 */
function playVisualNovel() {
    var rowCount = table.getDataCount();
    // load and insert image
    // call in function to set and load the correct image for character dependent on rowIndex
    setCharacterImage();
    // call function to get dialogue from tabulator table for rowIndex
    // - contains the data required for the drawWords() function
    readDialogue();
    // iterate through the frames
    rowIndex++;
    if (rowIndex < rowCount) {
        setTimeout(playVisualNovel, 6000);
    } else {
        alert("Visual Novel finished for this scene!");
    }
}
```

Figure E.3: Function for playing the visual novel.

Appendix F

F.1 JavaScript for Visual Novel

```
/**
 * Function to read in JSON file of background images,
 * checks which has been selected in the dropdown,
 * and loads the corresponding image
 */
function setBackgroundImage() {

    // create variable to hold the parsed JSON file
    let locationBack = null;

    try {
        // use the JSON.stringify() method to convert the data into a string before attempting to parse it
        var stringified = JSON.stringify(backgrounds);
        // Now locationBack is the parsed result
        locationBack = JSON.parse(stringified);
    } catch (e) {
        // display error if the JSON file does not parse correctly
        console.log("There was an error in parsing the JSON file!");
    }

        /**
         * Checks which background is selected from the dropdown
         * and loads respective image dependent on the result
         */
        document.getElementById('backgrounds').addEventListener('change', function () {
            if (this.value == "BoardRoom") {
                id = locationBack.data[0].BoardRoom;
                bg = loadImage(id);
            } else if (this.value == "Beach") {
                id = locationBack.data[0].Beach;
                bg = loadImage(id);
            } else if (this.value == "Forest") {
                id = locationBack.data[0].Forest;
                bg = loadImage(id);
            } else if (this.value == "Bedroom") {
                id = locationBack.data[0].Bedroom;
                bg = loadImage(id);
            } else if (this.value == "ClubBar") {
                id = locationBack.data[0].ClubBar;
                bg = loadImage(id);
            } else if (this.value == "Office") {
                var id = locationBack.data[0].Office;
                bg = loadImage(id);
            } else if (this.value == "Prison") {
                var id = locationBack.data[0].Prison;
                bg = loadImage(id);
            } else if (this.value == "Street") {
                var id = locationBack.data[0].Street;
                bg = loadImage(id);
            } else if (this.value == "Supermarket") {
                var id = locationBack.data[0].Supermarket;
                bg = loadImage(id);
            } else if (this.value == "Warehouse") {
                var id = locationBack.data[0].Warehouse;
                bg = loadImage(id);
            } else if (this.value == "") {
                bg = loadImage('../Images/placeholderImage.jpg');
            }
        });
```

Figure F.1: Script to set the background image of the visual novel from an JSON object.

```
/**
 * Function to read in JSON file of character images, get table data for specific rowIndex,
 * check which character is in the selected row and what their emotion is,
 * then load the corresponding image
 */
function setCharacterImage() {

    // create variable to hold the parsed JSON file
    let characterImage = null;
    // get data from tabulator table to use to access the correct images
    myTable = Tabulator.prototype.findTable('#example-table')[0];
    var data = myTable.getData();
    // get the character number and respective emotion for that rowIndex
    var charNo = (data[rowIndex].CharacterNumber);
    var charEmotion = data[rowIndex].Emotion;
    console.log("Char no: "+charNo+", emotion: "+charEmotion);

    // read in JSON file of character images
    try {
        // use the JSON.stringify() method to convert the data into a string before attempting to parse it
        var stringified = JSON.stringify(characters);
        // Now characterImage is the parsed result
        characterImage = JSON.parse(stringified);

    } catch (e) {
        // display error message in console if the JSON file does not parse correctly
        console.log("There was an error in parsing the JSON file!!");
    }
/**
 * This switch function take the variable charEmotion which finds the emotion stored in the table
 * for a particular rowIndex, and matches it to the corresponding image stored in JSON object
 * charNo-1 takes the character number from the table, and subtracts 1 to get the index of the data
 * in the JSON object
 */
switch (charEmotion) {
    case "Neutral":
        img = loadImage(characterImage.emotions[charNo - 1].Neutral);
        console.log("Image loaded successfully!");
        break;
    case "Angry":
        img = loadImage(characterImage.emotions[charNo - 1].Angry);
        break;
    case "SinisterGrin":
        img = loadImage(characterImage.emotions[charNo - 1].SinisterGrin);
        break;
    case "Smile":
        img = loadImage(characterImage.emotions[charNo - 1].Smile);
        break;
    case "Laughing":
        img = loadImage(characterImage.emotions[charNo - 1].Laughing);
        break;
    case "Sneer":
        img = loadImage(characterImage.emotions[charNo - 1].Sneer);
        break;
    case "Stern":
        img = loadImage(characterImage.emotions[charNo - 1].Stern);
        break;
    case "Grin":
        img = loadImage(characterImage.emotions[charNo - 1].Grin);
        break;
    case "Shock":
        img = loadImage(characterImage.emotions[charNo - 1].Shock);
        break;
    case "Sad":
        img = loadImage(characterImage.emotions[charNo - 1].Sad);
        break;
    case "Scared":
        img = loadImage(characterImage.emotions[charNo - 1].Scared);
        break;
    default:
        alert("No image file found");
}
```

Figure F.2: Script to set the character image in the visual novel according to the character number and emotion present in the table of frame data.

```
/**
 * Function to read dialogue, including who speaks and their inflection,
 * from the tabulator table of frames
 * to then load the corresponding text in the speech bubble
 */
function readDialogue() {

    myTable = Tabulator.prototype.findTable('#example-table')[0];
    var data = myTable.getData();
    // find dialogue for rowIndex
    // is the character speaking and who is it?
    var charSpeak = data[rowIndex].CharacterSpeaking;
    // what do they say? and set it to variable used in drawWords() function
    textWords = data[rowIndex].WhatDoTheySay;
    // get their inflection
    var inflection = data[rowIndex].Inflection;

    if (charSpeak != 0) {
        charSpeaking = "Character " + charSpeak +" "+ inflection + ":";
    } else {
        charSpeaking = "No Character is speaking.";
    }
}
```

Figure F.3: Script to read the dialogue from the table of frame data and set as text.

Appendix G

G.1 Generate Character Avatar Code

```
<!-- Modal to give user instructions-->
<div class="modal hide fade" id="myModal" tabindex="-1" role="dialog" aria-labelledby="myModal" aria-hidden="true">
    <div class="modal-dialog modal-dialog-centered" role="document">
        <div class="modal-content">
            <div class="modal-header">
                <h5 class="modal-title mytitles" id="">Create a Character Avatar!</h5>
                <button type="button" class="close" data-dismiss="modal" aria-label="Close">
                    <span aria-hidden="true">&times;</span>
                </button>
            </div>
            <div class="modal-body">
                The JSON object with the labelled character appearances is preloaded into the table below. To
                generate the corresponding avatar images simply click the row you wish, and click Generate! Then all
                you have to do is click the hamburger icon to the right of the avatar, and choose Download! To make
                another selection, press the Clear button before repeating the steps.
            </div>
            <div class="modal-footer">
                <button type="button" class="btn btn-dark" data-dismiss="modal">Close</button>
            </div>
        </div>
    </div>
</div>

<!--use container to apply styling to page-->
<div class="container mycontainer">

    <!--Create iframe container for edited character creator -->
    <div class="embed-responsive embed-responsive-16by9"
        style="border: 1px solid rgba(0,0,0,.1); border-radius: 0.25rem; min-height: 400px;">
        <iframe id="myiframe" class="embed-responsive-item" src="" style="width: 100%; height: 400px;">
        </iframe>
    </div>

<script type="text/javascript">
    //create table to hold the character appearances that were labelled via Amazon's mechanical Turk
    var local_data = charDetails;
    var rowIndex;
    console.log("table data: " + local_data);
    var table = new Tabulator("#example-table", {
        index: "numberOfCharacters",
        data: local_data,
        placeholder: "No Data Available",
        columns: [
            { title: "Character", field: "numberOfCharacters", editor: "input" },
            { title: "Name", field: "names", editor: "input" },
            { title: "Blouse", field: "blouse", editor: "input", },
            { title: "Complexion", field: "complexion", editor: "input", },
            { title: "Dress", field: "dress", editor: "input", },
            { title: "Earrings", field: "earrings", hozAlign: "center", editor: "input", },
            { title: "Eye Colour", field: "eyeColour", editor: "input", },
            { title: "Facial Hair", field: "facialHair", editor: "input", },
            { title: "Gender", field: "gender", editor: "input", },
            { title: "Glasses", field: "glasses", hozAlign: "right", editor: "input", },
            { title: "Hair Style", field: "hair", editor: "input", },
            { title: "Hair Colour", field: "hairColour", editor: "input", },
            { title: "Hat", field: "hat", hozAlign: "center", editor: "input", },
            { title: "Jacket", field: "jacket", editor: "input", },
            { title: "Makeup", field: "makeup", hozAlign: "center", editor: "input", },
            { title: "Trousers", field: "pants", editor: "input", },
            { title: "Scars", field: "scars", hozAlign: "center", editor: "input", },
            { title: "Shirt", field: "shirt", editor: "input", },
            { title: "Shoes", field: "shoes", editor: "input", },
            { title: "Skirt", field: "skirt", editor: "input", },
            { title: "Tattoos", field: "tattoos", hozAlign: "center", editor: "input", },
            { title: "Tie", field: "tie", editor: "input", },
            { title: "Vest", field: "vest", editor: "input", },
        ],

        // this function accesses the row which has been selected, or clicked

        rowClick: function (e, row) {
            // get index of row selected
            rowIndex = row.getIndex();
            // print index to console
            console.log("Row index: " + rowIndex);
            // get data for the selected row
            var rowData = row.getData();
            // print row data to console
            console.log(rowData);
```

```
// split up row data into separate variables
// the following variables should always have a value
// need to add the prefixes before so that charactercreator.org can read them correctly!!
rowGender = "sex=" + row.getData().gender;
rowComplexion = "skinColor=" + row.getData().complexion;
rowEyeColour = "irisColor=" + row.getData().eyeColour;
rowHair = "hair=" + row.getData().hair;
rowHairColour = "hairColor=" + row.getData().hairColour;

// these variables are optional, so need the check the value is not null before setting to the corresponding variable

rowFacialHair = "facialhair=" + row.getData().facialHair;
rowShirt = "shirt=" + row.getData().shirt;
rowBlouse = "blouse=" + row.getData().blouse;
rowDress = "dress=" + row.getData().dress;
rowSkirt = "skirt=" + row.getData().skirt;
rowPants = "pants=" + row.getData().pants;
rowJacket = "jacket=" + row.getData().jacket;
rowTie = "tie=" + row.getData().tie;
rowVest = "vest=" + row.getData().vest;
rowGlasses = "glasses=" + row.getData().glasses;
rowShoes = "shoes=" + row.getData().shoes;
rowTattoos = "tatoo=" + row.getData().tattoos;
rowScars = "scar=" + row.getData().scars;
rowHat = "hat=" + row.getData().hat;
rowEarrings = "earings=" + row.getData().earrings;
rowMakeup = "makeup=" + row.getData().makeup;


// pass parameters into current parent URL to be transferred into iframe URL
var parameterArray = [rowGender, rowComplexion, rowEyeColour, rowHair, rowHairColour, rowFacialHair, rowShirt, rowBlouse,

// remove commas from array and replace with & instead through the join() method
parameterArray = parameterArray.join('&');

// create onclick event functions to generate and clear avatar creation
document.getElementById("generateAvatar").onclick = function () { generateAvatar() };
document.getElementById("clearAvatar").onclick = function () { clearAvatar() };

        /**
         * This function will, when the Generate Avatar button is clicked, reload the iframe to include the parameters
         */
        function generateAvatar() {
            // set main url as that which is in the iframe
            var mainUrl = "https://aoibheannamy.github.io/getCharacterAvatar/charactercreator.html";

            // turn the parameter array into a string
            urlParams = parameterArray.toString();

            // put together mainURL, followed by a hash, with the urlParams
            var iframeUrl = mainUrl + '#' + urlParams;
            console.log(iframeUrl);
            // set the iframe's src as the iframeURL
            document.getElementById('myiframe').src = iframeUrl;
        }

        /**
         * This function clears the iframe src so that another selection can be made.
         */
        function clearAvatar() {
            document.getElementById('myiframe').src = "";
        }
    }
});
// replace the data to ensure the table is loaded correctly
table.replaceData(local_data);
```

Bibliography

[1] Emma Roth. Are visual novels video games? well, what exactly is a video game?, 2019. https://whatnerd.com/visual-novels-video-games/, Last accessed 24 August 2020.

[2] Afterthought Studios. A more beautiful world - a kinetic visual novel, 2018. https://store.steampowered.com/app/436680/A_More_Beautiful_World__A_Kinetic_Visual_Novel/, Last accessed 24 August 2020.

[3] Software Testing Help. What is sdlc waterfall model? https://www.softwaretestinghelp.com/what-is-sdlc-waterfall-model/ Last accessed: 31st August 2020.

[4] Airbrake. V-model: What is it and how do you use it? https://airbrake.io/blog/sdlc/v-model Last accessed: 31st August 2020.

[5] javaTpoint. Agile model. https://www.javatpoint.com/software-engineering-agile-model.

[6] [Richard Rogers. A simple comparison of sequential and iterative software development methods. https://richrtesting.com/a-simple-comparison-of-sequential-and-iterative-software-development-methods/.

[7] Blake Synder Enterprises. Save the cat! the language of storytelling. https://savethecat.com/.

[8] Industrial Scripts. How to write an awesome beat sheet for your screenplay. https://industrialscripts.com/beat-sheet/.

[9] Open-Source Psychometrics Project. Statistical "which character" personality quiz, 2020. https://openpsychometrics.org/tests/characters/, Last accessed 30 August 2020.

[10] Susan Storm. The myers-briggs personality types of 325 fictional characters. https://www.psychologyjunkie.com/2020/07/29/myers-briggs-325-fictional-characters/.

[11] D. Cavallaro. *Anime and the Visual Novel: Narrative Structure, Design and Play at the Crossroads of Animation and Computer Games*. EBL-Schweitzer. McFarland, Incorporated, Publishers, 2014.

[12] Frédéric Guimont. The character creator, 2014-2020. https://charactercreator.org/.

[13] Ivan Zhao. Notion. https://www.notion.so/about/.

[14] Amazon Mechanical Turk. Using csv files to create multiple hits in the requester ui. https://blog.mturk.com/using-csv-files-to-create-multiple-hits-in-the-requester-ui-22a25ec563dc.

[15] Lauren McCarthy. Hello! https://p5js.org/.

[16] Web Application & Data Integration Company. The linear sequential model of software development. https://wadic.net/software-development-linear-sequential-model/.

[17] Agile Alliance. What is agile? https://www.agilealliance.org/agile101/.

[18] Mountain Goat Software - Mike Cohn. Agile needs to be both iterative and incremental. https://www.mountaingoatsoftware.com/blog/agile-needs-to-be-both-iterative-and-incremental.

[19] Medium - DLT Labs. Requirement elicitation techniques for business analysis. https://medium.com/@dltlabs/requirement-elicitation-techniques-for-business-analysis-ed0a3d405910.

[20] QRA Team. Functional vs non-functional requirements: The definitive guide. https://qracorp.com/functional-vs-non-functional-requirements/.

[21] Amazon Web Services. Crowd html elements. https://docs.aws.amazon.com/AWSMechTurk/latest/AWSMturkAPI/ApiReference_HTMLCustomElementsArticle.html.

[22] Bootstrap. Introduction. https://getbootstrap.com/docs/4.5/getting-started/introduction/.

[23] GiHub Docs. About github pages. https://docs.github.com/en/github/working-with-github-pages/about-github-pages.

[24] What is javascript used for? https://www.hackreactor.com/blog/what-is-javascript-used-for.

[25] The jQuery Foundation. What is jquery? https://jquery.com/.

[26] Oli Folkerd. Complex tables, simple code. http://tabulator.info/.

[27] Amazon Web Services. Mechanical turk concepts. https://docs.aws.amazon.com/AWSMechTurk/latest/RequesterUI/mechanical-turk-concepts.html.

[28] Didacus Odhiambo . System design in software development. https://medium.com/the-andela-way/system-design-in-software-development-f360ce6fcbb9#:~:text=System%20design%20is%20the%20process,that%20goes%20through%20that%20system.

[29] Daniel Bolton. Youtube-dl. https://github.com/ytdl-org/youtube-dl.

[30] Fandango. Movieclips. https://www.youtube.com/channel/UC3gNmTGu-TTbFPpfSs5kNkg.

[31] NUMFocus. pandas. https://pandas.pydata.org/.

[32] Douglas Crockford . Introducing json. https://www.json.org/json-en.html.

[33] Pallets. Installation. https://flask.palletsprojects.com/en/1.1.x/installation/.

[34] David Richards. How to install pip on windows. https://www.liquidweb.com/kb/install-pip-windows/.

[35] Software Testing Fundamentals (STF). Black box testing. https://softwaretestingfundamentals.com/black-box-testing/.

[36] Software Testing Fundamentals (STF). White box testing. https://softwaretestingfundamentals.com/white-box-testing/.