1. **Analysis**

1.1. Introduction

1.1.1. What is the problem

1.2. Features of the problem

1.2.1. Why it is solvable by computational methods

1.3. Stakeholders

1.4. Research the problem

1.4.1. Evidence of research

1.4.2. Why are suitable for my Solution

1.4.3. Limitations of the example solution

1.5. Essential Features

1.6. Limitations

1.7. Hardware and software Requirements

1.8. Measurable success criteria

1. **Design**

2.1. Breaking the solution into smaller sections

2.1.1. Explain and justify the decisions made

2.1.2. Explain and justify the structure of the solution

2.1.3. Flow Charts

2.1.4. Pseudocode

2.2. Describing the Solution using algorithms

2.2.1. Example Algorithms - Pseudocode

2.2.2. Example Algorithms - Flowcharts

        2.2.2. Justify how the algorithms form a solution

2.3. Modelling the Problem

2.4. Usability Features

2.4.1. Annotated Interface Design

2.4.2. Justify the included usability features

2.4.3. Justify what makes them user friendly

2.5. Permanent data storage of user information

2.5.1 How the data will be stored

2.6. Key variables, data structures and classes

2.6.1. Justify the choices made

2.6.2. Any necessary validation

2.7. Test data for use during iterative and post development

2.7.1. Identified test data - Normal, Boundary, Erroneous

2.7.2. Destructive testing of inputs and outputs

2.7.3. Justify chosen test data

2.7.4. Annotated examples of test data

1. **Implementation**

Analysis

**Introduction:**

The solution I am proposing is an adventure-rpg style game that gets increasingly harder as the player progresses through it.

**Features of the problem:**

One feature of the program that is solvable by computational methods is that it will allow me to generate different looking environments for the player to progress through. By using a programming language (Python 3) i am able to build the graphical room layouts for the environment of my program and make some unique to others, e.g. the layout of the floors; the size of the floors and also the colours and textures.

Another feature of my program that makes it solvable by computational methods is that I can create enemies. By using python I am able to create quite unique enemies that appear all throughout my game; I can easily choose a list of premade skins to then use in python that will allow each “room” of my game to have a different enemy than the last. I will be able to randomly generate the values of each enemy as well and I will also be able to increase the values of the enemy at a constant rate as the player progresses and the game increases in difficulty. It also allows me to add multiple different entities that behave differently; for example one entity is faster than the others but does less damage, and another entity might have a large amount of enemies compared to the others but have reduced movement speed allowing the player to manipulate the movements of that enemy more effectively compared to others. This also allows me to add multiple different textures/models for different enemies.

One main feature of my program is a database system so players can save their high scores and continue from where they left off, if they haven’t completed the game yet. This feature will be done graphically using python’s pygame module, by allowing players to create a login (username and password) to link their score to an account, which will be stored in a database and before exiting the game the user will have the option to save their high score with their account details. This could allow the player to start the game from where they left off so they don't have to start from the beginning (a save game/progress feature); players also have the option to play as guest - this means login isn’t required but any progress the player makes will not be saved and will only be relevant for the current session.

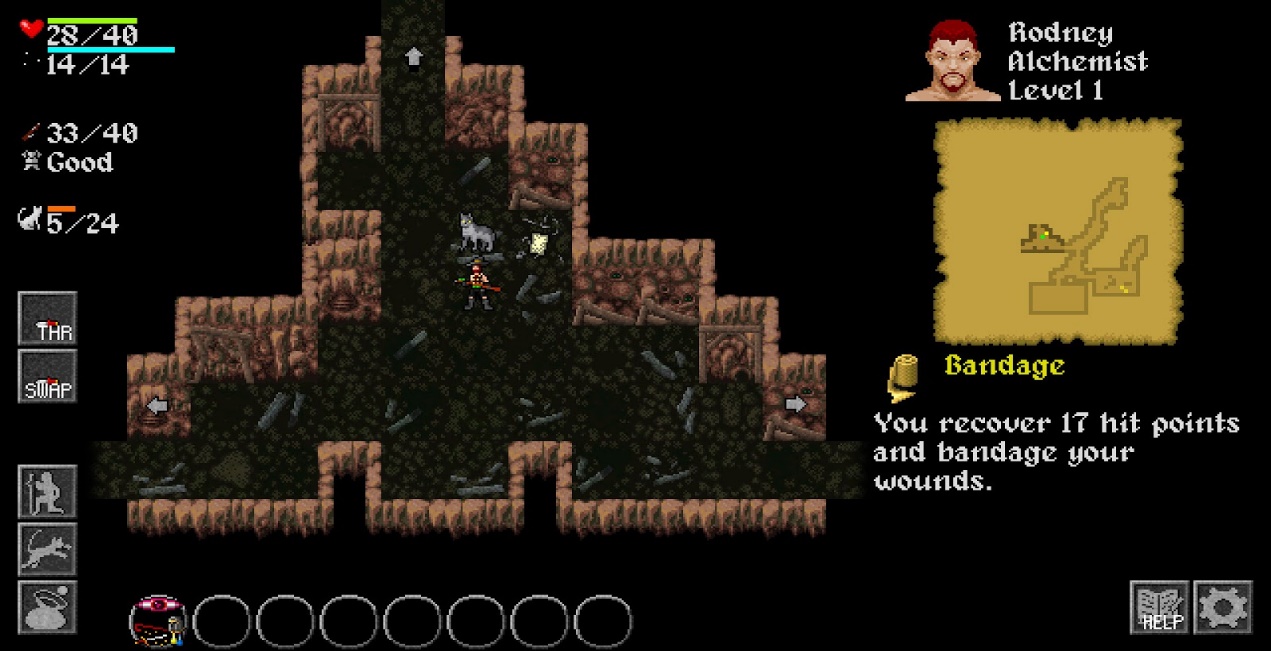
Another feature of my program would be how the player interacts with the environment; for example interacting with enemies, interacting with randomly spawning power-ups to help the player and increases their power; also the way the player interacts with the environment; e.g. not being able to go past the set borders of the environment or continue to the next section without clearing the previous one.

Another feature of the program is giving the user the option to change the keybinds for each different action. For example, by default WASD will be used to control the player. The user will have the ability to change the keybinds in order to suit their preferred layout. For example they could change the movement keys to the arrow keys.

**Suitable Stakeholders:**

A Suitable Stakeholder for this solution would be the retro games club at All saints school; which suits my target audience of teenagers; that get together once a week and play video games from the the 90’s to the early 2000’s and i think that this club could make use of my program as classic RPG style games, that i am taking inspiration from were made  in the 90’s and 2000’s which suits the retro games club. Because of my chosen stakeholder, it will also allow me to get consistent feedback from the people that are part of the club, during the development of my program; which will allow me to be made aware of errors or certain systems that don’t work entirely, or as intended. This will be good for the development of my program because different people may find errors that I have overlooked and will allow me to fix them as I develop my program; rather than find all of the errors at the end of development and have to change them all at once.

**Research Problem in Depth: 2/3**

This example of a rogue-like adventure game is a suitable example of what my solution will look like at the end of development; in particular the graphical user-interface layout. However, I would include a settings option, allowing the user to save the game and access a help section if needed.One weakness of this GUI is that it is cluttered and has a few UI elements that don’t need to be constantly displayed to the user and can be confusing to a new user if they aren’t given clear instructions on what each UI element does. Also the borders/”walls” of this environment are messy and visually unappealing; in particular the textures that stick out and look out of place and almost look like it has been generated incorrectly.

Unlike the previous example the User Interface for this example game is very clear and simple for the user to understand, even without the annotated text, it is clear to the user what each element of this UI does - for example the button label Map, clearly opens the map for the current environment. However, one weakness of this UI is that the elements are quite large and take up a lot of space on the screen compared to the size of the environment the user is controlling their character in; with this example i would reduce the size of the surrounding UI elements in order to make the environment, that the user controls their character in, the most dominant aspect of the UI as it is the most important. Another strength of this example is the textures and layout of the environment. The textures of the “walls” look clean and suit the style of game i am going for and the on screen annotations give me give me the opportunity to design an informative help section/screen for user who require it; whilst also make it concise and not just a wall of text for users to read through.

*Take 3 examples from other solutions similar to my own and take sections from it that i will want to take inspiration from in my program and analyse them - use SWOT analysis (strengths, weaknesses, opportunities, threats) to make it effective and be able to justify my own approach to this problem*

**Essential Features of the program:**

An essential feature of my program is that the program will be displayed in its entirety through a GUI. This is an important feature as it would be difficult and quite boring to play as a text-based program and by using a GUI it opens up many different options, in terms of how far i can go with this solution.

Another essential feature is a database-like system where users can record their username and highscore, whether they have completed the game or not they will be given the option to save their score to the database at the end of the game. This is an essential feature from a users perspective as it ensures that the progress they have made in the game isn’t wasted and they can even come back to it when they want.

Another essential feature is that I will be consistently increasing the values of each enemy (speed, health and damage dealt) as the player progresses. This ensures that the game increases in difficulty and that the game poses a certain level of challenge to the player.

**Limitations and solutions to those limitations:**

A potential limitation of my project is that I need to improve my knowledge of the pygame module in Python3 to a high standard in order to create all of the graphical elements of my program; I could resolve this by... Another potential limitation I may face when developing my proposed solution is the time I have in order to complete it; I can overcome this legislation by managing my time effectively and by setting frequent deadlines for different actions of my project.

**Hardware and Software Requirements:**

A machine with Python3 and Pygame installed and running correctly.

**Success Criteria: (*10+)***

1. Users must be given the option to create a username and password.

* The username must be between 3 and 12 characters.
* The password must be at least 6 characters long.
* The password must be re-entered by the user in order to make sure it matches and is the desired password.
* The user must also be given the option to change their password.
* Users must be given the option to play as a guest where login isn’t required but whatever score/progress the player makes will not be stored.

1. Users must be 100% satisfied with the performance and response times of the solution.
2. Users must be able to pause the game at any point during their current session regardless of their progress through the game.

* The game must pause every time the user enters the settings or help section and unpause once the user exits these sections.

1. The entire solution must be displayed to the user through a GUI.

* The GUI must have clear usability features.
* The GUI must keep UI elements small, but understandable, in order to keep it clean and uncluttered.
* The GUI should have links to other applications in order to make it understandable by everyone - e.g. using the disc icon as a save button from microsoft word

1. Users must be able to access a help section at any time during their current session.
2. The player must only be able to advance to the next level, once all enemies have been defeated
3. The sprite animations must be smooth and fluid
4. The level layouts must be randomly generated, including the starting position(x,y) of each sprite
5. Must be able to collect the x and y coordinates of all drawn sprites, including the mouse cursor, and store them in variables

**Design**

**Breaking the solution into smaller sections**

**Explain and justify the decisions made**

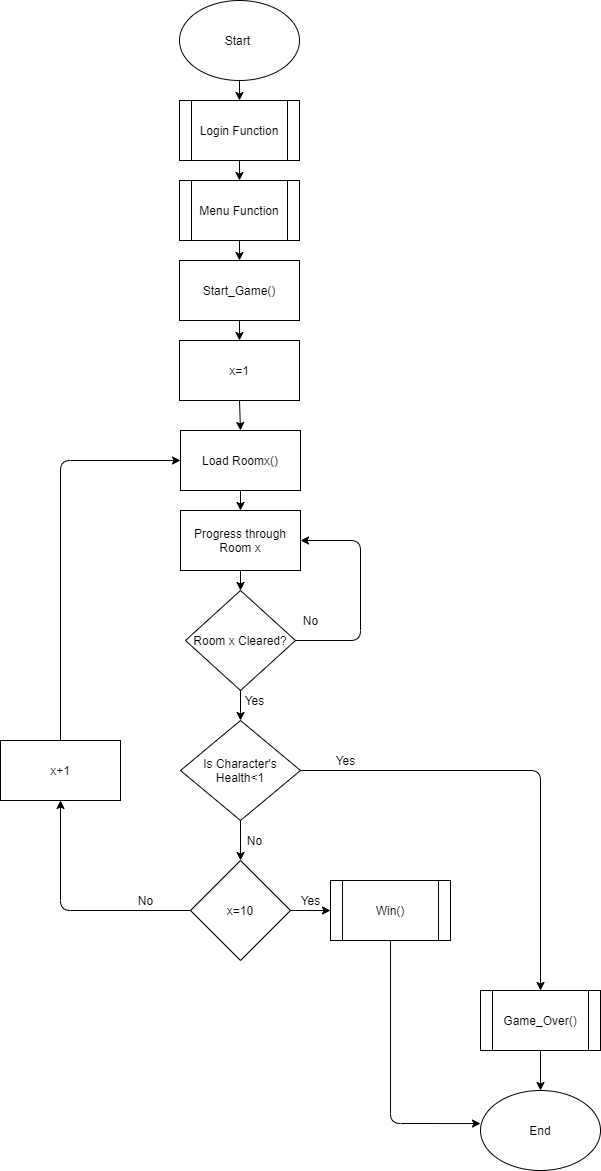
**Explain and justify the structure of the solution**

**Describing the Solution using algorithms**

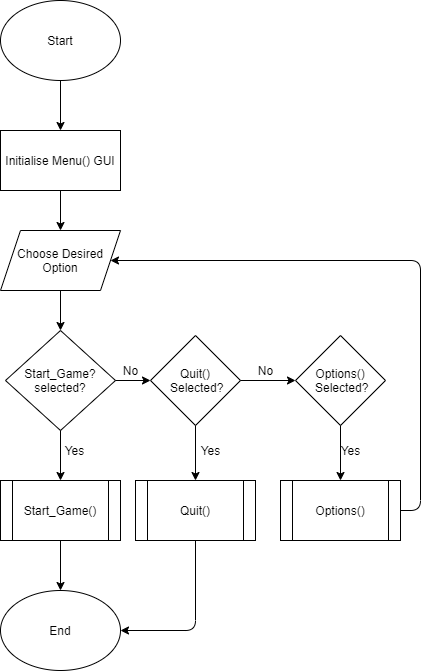
**Example Algorithms - Pseudocode**

**Example Algorithms - Flowcharts -**

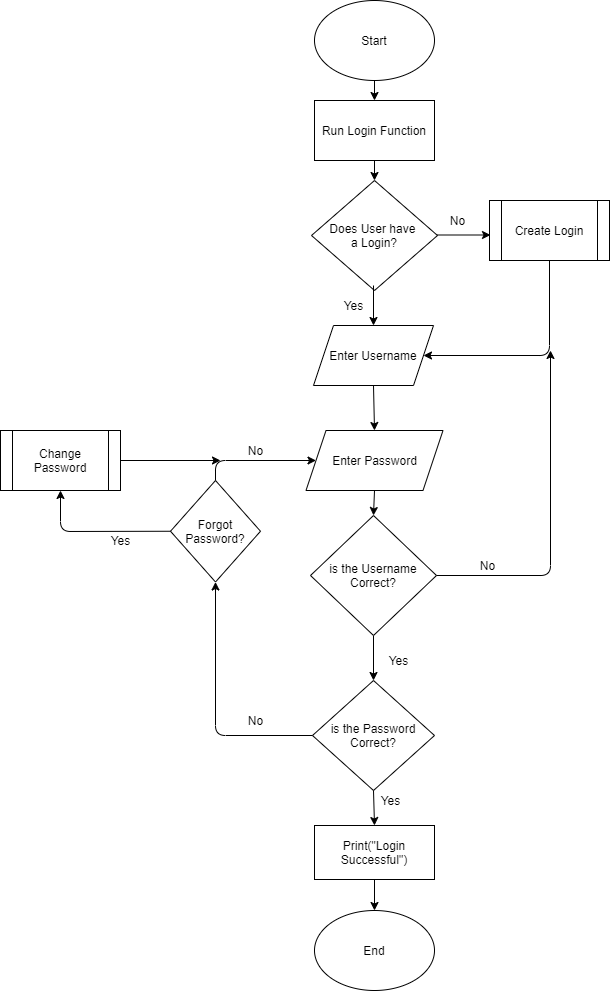
**Overall Program:**



**Menu Function:**

****

**Login Function:**



Justify how the algorithms form a solution

Modelling the Problem

Usability Features + Justification

Annotated Interface Design

Permanent data storage of user information

How the user data will be stored

Key variables, data structures and classes - (consistent variable names!)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Name | Data Type | Scope | Starting Value | Range of values | Validation | Notes |

**Login Function:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Username | String | Global | “” | Between 3 and 12 Characters | Check whether the entered string is between the specified range |  |
| Password | String | Global | “” | Between 6 - 16 Characters | Check whether the entered string is between the range that is specified in the validation |  |

**Main Game Loop:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| run | Bool | Global | True | N/A |  | Used to break out of the main game loop; to do this the variable is set to False |
| idle.left | Bool | Global | False | True or False |  | If the a key is pressed man.left is set to true and the x value of the character is decreased. When the character is drawn the program knows to use the moving left animations |
| player.idle | Bool | Global | True | True or False |  | If no keys are pressed player.idle is set to true so the image for the character standing still is displayed. If one of the movement keys is pressed the variable is set to False. |
| screen\_x | integer | Global | 1000 | N/A |  | This variable is used to set the width of the pygame window (x-axis) |
| keys |  | Global | pygame.key.get\_pressed() | N/A |  | Collects all of the keys pressed by the user and stores them in this variable |
| screen\_y | integer | Global | 600 | N/A |  | This variable is used to set the height of the pygame window (y-axis) |
| win |  | Global | pygame.display.set\_mode((screen\_x, screen\_y)) | N/A |  | This variable sets the dimensions of the pygame window and establishes the surface, win, for images to be drawn on |

**Player() Class:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Vel | integer | Local | 5 | N/A |  | This variable controls the speed at which the images are cycled through (controls the speed of the character)9 |
| x | integer | Local | screen\_x/2 | N/A |  | This variable is used to set the x coordinate of the sprite and by dividing the width of the screen by 2 (and the y) the sprite will appear in the middle of the screen |
| walk\_count | integer | Local | 0 | 0-27 |  | This variable is used to to cycle through each different sprite animation - each walk\_count value relates to a different image in the list of images. |
| y | Integer | Local | screen\_y/2 | N/A |  | This variables is used to set and control the y coordinate of the player |

**Menu Function:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Mx, my |  | Local | pygame.mouse.get\_pos() | N/A |  | Gets the x and y coordinates(current position) of the mouse cursor and stores them as this variable |
| click | Bool | Local | False | True or False |  | This variable is used to check whether or not the mouse has clicked by the user - if the mouse has been clicked the value of this variable changes to True |
|  |  |  |  |  |  |  |

**Justify the choices made**

**Test data for use during iterative development**

**Login Function:**

**Save/Export Function:**

I will use a temporary file, in order to test whether or not the program can successfully open and read from a file.

I will use test data within this temporary file in order to make sure that the program can open a file and then read information from it to be used in the program.

I will also use a set of test data within the program to export to the temporary file in order to make sure that data can be saved to an external file.

**Test data for post development**

**Identified test data - Normal, Boundary, Erroneous**

**Destructive testing of inputs and outputs**

**Justify chosen test data**

**Annotated examples of test data**

**Implementation**

*From the A-level specification: Mark band 3; 9 - 12 Marks*

*• Provided evidence of each stage of*

*the iterative development process for*

*a coded solution relating this to the*

*break down of the problem from the*

*analysis stage and explaining what they*

*did at each stage.*

*• Provided evidence of some prototype*

*versions of their solution.*

*• The solution will be modular in nature.*

*• Code will be annotated to explain all*

*key components.*

*• Most variables and structures will be*

*appropriately named.*

*• There will be evidence of validation for*

*most key elements of the solution.*

*• The development will show review at*

*most key stages in the process*