|  |
| --- |
| BHASVIC Computer Science |
| **Animalopoly** |
|  |

|  |
| --- |
| Taran Singh  [Date] |

Contents

[Analysis 2](#_Toc196816466)

[What goes in this section? 2](#_Toc196816467)

[Success Criteria 3](#_Toc196816468)

[Design 7](#_Toc196816469)

[What goes in this section? 7](#_Toc196816470)

[Checklist 8](#_Toc196816471)

[Development 9](#_Toc196816472)

[What goes in this section? 9](#_Toc196816473)

[Checklist 10](#_Toc196816474)

[Evaluation 11](#_Toc196816475)

[What goes in this section? 11](#_Toc196816476)

[Checklist 12](#_Toc196816477)

[References 13](#_Toc196816478)

[References Explanation 13](#_Toc196816479)

[Reference List 15](#_Toc196816480)

[AI References 16](#_Toc196816481)

[AI References Explanation 16](#_Toc196816482)

[AI Reference List 16](#_Toc196816483)

[Referencing FAQs 17](#_Toc196816484)

[What happens if I don’t cite a source I’ve used? 17](#_Toc196816485)

[What if I don’t realise, I’ve plagiarized? 17](#_Toc196816486)

[What are the potential consequences of plagiarism? 17](#_Toc196816487)

[If AI generated work isn’t marked, then why use it? 18](#_Toc196816488)

# Analysis

## What goes in this section?

In your actual programming project, the analysis section is a significant piece of work in which you justify why the program/product you have decided to create should be created. You’ll look at existing solutions and competitors to examine what their strengths and weaknesses are and what opportunities and threats this creates for your project. You’ll also use questionnaires and interviews with your potential stakeholders/target audience to understand their needs.

Once you’ve gathered this information, you’ll create success criteria that will inform your development and design tests that will ensure that you can evidence your progress towards meeting these success criteria.

You’ll complete the Analysis Phase of your programming project before the Summer Break so that you can then focus on design in September.

For Animalopoly, we’ve given you the success criteria to save time! The only thing you need to do is to prioritize them based on their importance:

* Green = Most Important, do this first!
* Yellow
* Red = Least Important, do this last!

## Success Criteria

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Success Criteria** | **Justification** | **Importance** |
| 1. Dice | | |  |
| 1.1 | Two Dice Implementation: Implement two dice that the player can roll, with each die generating a random number between 1 and 6. | Implementing two dice is fundamental to the game mechanics, as it determines how far players move on the board. This randomness adds excitement and variability to each turn, making the game more engaging. | Yellow |
| 1.2 | Dice Roll Outcome: Display the total number of spaces the player should move based on the sum of the dice roll. | Displaying the number of spaces the player should move based on the dice roll ensures players understand their movement clearly. This transparency is crucial for maintaining the flow of the game and preventing confusion. | Green |
| 1.3 | Double Roll Detection: Detect and confirm when two of the same dice have been rolled, triggering a special event. | Confirming when two of the same dice have been rolled is important because it triggers special events, such as drawing a card. This feature adds an extra layer of strategy and unpredictability, enhancing the overall gameplay experience. | Red |
| 2. Board | | |  |
| 2.1 | Board Display: Create and display a text-based board with 26 spaces to the user. | Creating a text-based board with 26 spaces and displaying it to the user provides a visual representation of the game state. This helps players keep track of their positions and the overall progress of the game, making it more immersive. | Green |
| 2.2 | Player Location Storage: Store and update the location of each player on the board after every move. | Storing the location of each player on the board is necessary for tracking player positions and game progress. It ensures that the game can accurately reflect each player's movements and interactions with the board spaces. | Green |
| 2.3 | Current Player Position: Display the current position of each player on the board at all times. | Showing on the board where each player currently is helps players understand their current status in the game. This visibility is essential for strategic planning and decision-making during their turns. | Yellow |
| 2.4 | Space Instructions: Provide specific instructions to the player based on the space they land on. | Informing the player what to do when they land on a specific space ensures players know the rules and actions required for each space. This clarity is vital for smooth gameplay and helps prevent misunderstandings. | Green |
| 2.5 | Start Space Rewards: Automatically give the player £500 when they pass start and £1000 when they land on start. | Giving the player money when they pass (£500) or land on (£1000) start adds strategic elements and rewards for player movement. These incentives encourage players to keep moving and add excitement to reaching key points on the board. | Yellow |
| 2.6 | Miss a Turn: Have the player miss their next turn when they land on the "miss a turn" space. | Having the player miss a turn when they land on the appropriate square introduces penalties and adds complexity to the game. This feature can change the dynamics of the game and requires players to adapt their strategies. | Red |
| 3. Player | | |  |
| 3.1 | Playing Piece Selection: Allow players to select a text-based playing piece (e.g., \*, @, ?). | Allowing players to pick a text-based playing piece personalizes the game experience for each player. This customization makes the game more enjoyable and helps players feel more connected to their in-game character. | Red |
| 3.2 | Player Name Storage: Store and display the player's name at the start of their turn. | Storing and announcing the player's name on their turn enhances player identification and engagement. It adds a personal touch to the game, making each turn feel more significant and interactive. | Yellow |
| 3.3 | Money Management: Track and update the amount of money each player has throughout the game. | Storing and updating the amount of money each player has is central to the game mechanics and determining the winner. This feature is crucial for tracking financial transactions and ensuring the game progresses correctly. | Green |
| 3.4 | Bankruptcy Notification: Notify players when they run out of money and are eliminated from the game. | Informing players when they run out of money indicates game status and player elimination. This notification is essential for maintaining the competitive aspect of the game and ensuring players are aware of their standing. | Green |
| 3.5 | Winning Condition: Announce the winner when they are the last player remaining with money. | Announcing the winner when they are the last player with money defines the end goal and victory condition of the game. This clear objective helps players understand what they are striving for and adds excitement to the competition. | Green |
| 4. Animals | | |  |
| 4.1 | Animal Assignment: Assign a unique animal to each space on the board. | Assigning an animal to each space on the board adds thematic elements and variety to the game. This feature enriches the gameplay experience by introducing unique attributes and interactions for each space. | Green |
| 4.2 | Animal Information Storage: Store detailed information for each animal, including name/species, level, cost to stop/visit, cost to buy, and owner. | Storing information for each animal (name/species, level, cost to stop/visit, cost to buy, owner) provides detailed attributes for game interactions. This data is essential for managing ownership, costs, and upgrades within the game. | Green |

|  |  |  |  |
| --- | --- | --- | --- |
| 4.3 | Animal Purchase: Allow players to purchase animals at their set cost when they land on an unowned animal space. | Allowing players to purchase animals at their set cost when they have no owner introduces ownership and investment mechanics. This feature adds strategic depth, as players must decide when to invest in animals to gain advantages. | Green |
| 4.4 | Animal Upgrade: Enable players to upgrade animals to increase their level at the same cost as the purchase price. | Allowing players to upgrade animals to increase their level at the same cost as purchase adds strategic depth and progression. This feature encourages players to enhance their assets, making the game more dynamic and competitive. | Green |
| 4.5 | Stop Cost Notification: Inform players of the cost to stop on an owned animal space. | Informing players of the cost to stop on an owned animal space ensures players are aware of financial penalties. This transparency is crucial for strategic planning and helps players make informed decisions during their turns. | Green |
| 5. Cards | | |  |
| 5.1 | Card Pack Creation: Create a pack of 20 cards with various scenarios where players gain or lose money. | Creating a pack of 20 cards with various scenarios where players gain/lose money adds random events and unpredictability to the game. This feature keeps the gameplay exciting and introduces new challenges and opportunities. | Yellow |
| 5.2 | Random Card Draw: Implement a feature to randomly draw a card from the pack when needed. | Implementing the ability for a random card to be drawn when needed enhances game dynamics and player decision-making. This randomness adds an element of surprise and requires players to adapt to changing circumstances. | Yellow |

# Design

## What goes in this section?

You need to include any whole project design/planning that your group did as well as the design/planning that you did for your objects – you do not need to include the design/planning that other members of your group did for their objects.

This should include:

* class diagrams
* flowcharts
* pseudocode
* data dictionaries

It should also include your test planning – remember that you need to have iterative testing, system testing and useability testing.

For each of these you need to plan **exactly** what test data should be used and **exactly** what output/result you would expect if that data were input.

## Checklist

This is the top band marking criteria for the design section of your programming project – you should aim to tick off every statement on this list. You should also be clear about what evidence you have included for each statement and what page of your document this evidence is on.

|  |  |  |  |
| --- | --- | --- | --- |
| **I have…** | **y/n** | **As evidence, I have included…** | **On page…** |
| Broken the problem down systematically into a series of smaller problems suitable for computational solutions, explaining and justifying the process |  |  |  |
| Defined in detail the structure of the solution to be developed. |  |  |  |
| Described the solution fully using appropriate and accurate algorithms justifying how these algorithms form a complete solution to the problem. |  |  |  |
| Described, justifying choices made, the usability features to be included in the solution. |  |  |  |
| Identified and justified the key variables / data structures / classes (as appropriate to the proposed solution) justifying and explaining any necessary validation. |  |  |  |
| Identified and justified the test data to be used during the iterative development of the solution. |  |  |  |
| Identified and justified any further data to be used in the post development phase. |  |  |  |

# Development

## What goes in this section?

This section is about explaining the process you went through to develop the program – you should be taking regular screenshots of your **fully annotated** code and then adding this to your development section with information about what you did and why you did it.

You also need to show evidence that you were regularly running and testing your programming – you should have an iterative test plan to follow but this will/should change and evolve as you are programming, and you start to encounter issues you didn’t predict.

It is **super** important that you include evidence of any failed tests, what you changed to make your code work and **why** these changes were important/the best solution for that situation. If you try to run your code and it doesn’t run then take a screenshot, add it to your development section, make the change, explain how you solved the problem, then add a screenshot of the fixed code.

## Checklist

This is the top band marking criteria for the development section of your programming project – you should aim to tick off every statement on this list. You should also be clear about what evidence you have included for each statement and what page of your document this evidence is on.

|  |  |  |  |
| --- | --- | --- | --- |
| **I have…** | **y/n** | **As evidence, I have included…** | **On page…** |
| Provided evidence of each stage of the iterative development process for a coded solution, relating this to the breakdown of the problem from the analysis stage and explaining what I did and justifying why |  |  |  |
| Provided evidence of prototype versions of my solution for each stage of the process. |  |  |  |
| Ensured the solution is well structured and modular in nature. |  |  |  |
| Annotated the code to aid future maintenance of the system. |  |  |  |
| Ensured all variables and structures are appropriately named. |  |  |  |
| Included evidence of validation for all key elements of the solution. |  |  |  |
| Shown my review at all key stages in the process. |  |  |  |
| Provided evidence of testing at each stage of the iterative development process. |  |  |  |
| Provided evidence of any failed tests and the remedial actions taken with full justification for any actions taken. |  |  |  |

# Evaluation

## What goes in this section?

This section is about showcasing your system and usability testing results – it’s ok if your program doesn’t fully work (it probably won’t!).

Wherever possible the evidence in your testing table should be a screen recording of the program in action. The videos should be stored in the same folder as this document.

If it doesn’t fully work, then you actually have more to write about and show your evaluation skills – make sure that you comment on why you were unable to get the program fully working and how you could meet the missing criteria through further development.

Make sure to also think about whether the code you have created would be easy, difficult or somewhere in the middle for somebody else to maintain – have you coded in a weird way? Could you have been more modular? Used more inheritance?

## Checklist

This is the top band marking criteria for the evaluation section of your programming project – you should aim to tick off every statement on this list. You should also be clear about what evidence you have included for each statement and what page of your document this evidence is on.

|  |  |  |  |
| --- | --- | --- | --- |
| **I have…** | **y/n** | **As evidence, I have included…** | **On page…** |
| Provided annotated evidence of post development testing for function and robustness |  |  |  |
| Provided annotated evidence for usability testing. |  |  |  |
| Used the test evidence to cross reference with the success criteria to evaluate the solution explain how the evidence shows that the criteria has been fully, partially or not met in each case. |  |  |  |
| Provided comments on how any partially or unmet criteria could be addressed in further development. |  |  |  |
| Provided evidence of the usability features justifying their success, partial success or failure as effective usability features. |  |  |  |
| Provided comments on how any issues with partially or unmet usability features could be addressed in further development. |  |  |  |
| Considered maintenance issues and limitations of the solution. |  |  |  |
| Described how the program could be developed to deal with limitations and potential improvements / changes. |  |  |  |
| There is a well-developed line of reasoning which is clear and logically structured. The information presented is relevant and substantiated. |  |  |  |

# References

## References Explanation

You **must not** take anything you find online and include it (even with changes) directly into your project – you should be reading through the sources you use and using the knowledge gained to solve the problems.

The penalties for plagiarism can range from zero marks for your coursework to disqualification from **all** your A Levels (and everything in between) depending on the severity of the plagiarism.

To clear, it’s ok (encouraged even) for you to access websites books, and other resources (as long as they **aren’t** AI bots), when you get stuck as long as you use the knowledge acquired to then independently solve your problem.

You should try figuring out the problem first but if there is a pesky syntax error then you can’t solve then try and find a solution elsewhere. You just need to reference **every** source you look at.

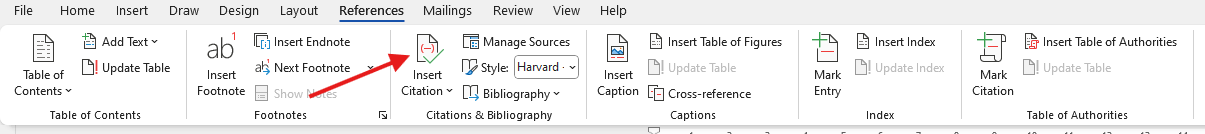
As you are writing this documentation you should include details about **every** website you access, **every** book that you read, and **every** other source that you access that can’t be so easily categorised.

Even if you don’t feel like you learned anything from the source, you **must** mention that you looked at it while trying to find the answer to something.

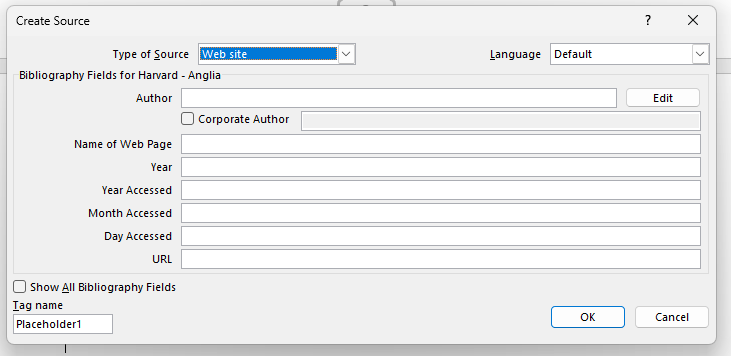
The easiest way to track this is to use the inbuilt References tab in Word – when you ever you need to cite that you’ve accessed a website you click the icon shown on the next page, select the type of source and the fill in the form – this will cause something like this: (Zucker, 2025) to appear in your workand the References section (below) to automatically update with details of the source.

If you reuse a source, you’ve already added then it appears on a drop-down menu to save time.

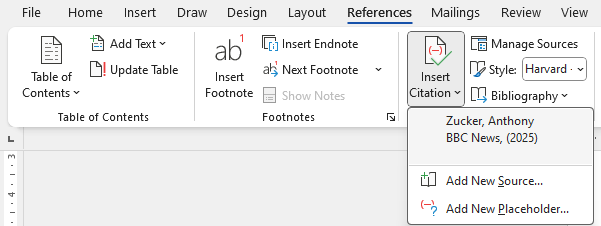
**References Section**



**Source Form**



**Reusing a Source**



# Reference List

Zucker, A., 2025. *BBC News.* [Online]   
Available at: https://www.bbc.co.uk/news/articles/c5ypz7yx73wo  
[Accessed 29 April 2025].

# AI References

## AI References Explanation

**AI should be your absolute last resort.**

As with other sources, any part of your work that comes directly from AI **will not be marked**, the big difference is that any work that is improved by AI **will also not be marked.**

This is not limited to code generation, if you use AI to help with any section of your project (Analysis, Design, Development, Evaluation) then you **need** to cite it and make it clear what was your work and what work was create via assistance from AI so that only **your** work is marked.

Examples:

* Using AI to reword your work – we will mark the original work shown in the prompt, not what the AI has generated.
* Generating code using AI – this should not even be considered as code generated via AI will not be marked
* Debugging code using AI – we will mark the original work shown in the prompt, not what the AI has debugged

As a result, citing AI sources is a little bit harder than a “normal” reference.

You need to include in your write up something like: “I was struggling to get the dice rolling mechanic to work and had to use ChatGPT to debug the code I had written.”

(ChatGPT 3.5 (https://openai.com/blog/chatgpt/), 25/01/2024.)

They key information here is

* The name of the AI bot
* The date the content was generated
* How you used the response

You should then include in your Refence Section screenshots of your **full** conversation/interactions with the AI.

* The prompt used
* The AI response

## AI Reference List

# Referencing FAQs

## What happens if I don’t cite a source I’ve used?

That won’t happen… you are going to be screenshotting your project so frequently that you’ll always have your documentation open, so you’ll just record that you’ve accessed the site as accessing the site.

With non-AI sources, if you’ve got dozens of sources and one of them is accidentally missing, then you’ve clearly shown understanding of the importance of referencing and that you’ve taken it seriously – that’s what matters, you won’t be penalised.

If you deliberately don’t cite a source, then it will be treated as plagiarism and investigated.

With AI sources, if you don’t cite AI usage then we will notice (it’s more obvious than you think) and we will have to investigate.

## What if I don’t realise, I’ve plagiarized?

What a stupid question!

If you can answer no to **all** these questions, then you haven’t plagiarized.

* Have you directly copied from a source?
* Have you copied from a source and changed the words?
* Have you deliberately not included a source for any reason?
* Have you used AI and not sourced it?

## What are the potential consequences of plagiarism?

All suspected plagiarism is fully investigated and will involve meeting with your class teacher and the Head of Department.

If you are found to have plagiarised, then the consequences will depend on the severity of the plagiarism but could include:

* Receiving zero marks for a section of your work
* Receiving zero marks for the entire coursework unit
* Disqualification from your OCR A Level Computer Science course
* Disqualification from **all** OCR qualifications – including in the future
* Disqualification from **all** your A Levels (extremely rare).

## If AI generated work isn’t marked, then why use it?

That’s sort of the point… but seriously there might be a very specific issue that you cannot solve that is preventing you from moving forward and accessing marks in other sections of the mark scheme. In this situation you should make sure that you have exhausted **literally every other option** before using AI.

# Allocation (vaguely by class):

Charlie – Player, cards

Solomon - Board, main

Taran – Animals, dice