Project report: Pinn Booranamaitree

Outline at a high level what your project is designed to accomplish.

This should be understandable by a layperson (a grandparent for example). Try to avoid using any especially technical words.

# Part 1: Planning

## 1.1 Planning tools & techniques

Discuss the tools and techniques you used. Why did you choose them? Include things like a Gantt chart if you did one. Show you know what tools are available and why they are useful.

Git hub

Trello

Kanban

Burndown chart

### 1.1.1 Actual use of the planning tools

Give examples of how you used the tools and techniques you selected. What was the impact of using them? How did they contribute to the success (or otherwise) of your project?

Show that you not only know **how** to plan a project, but that you actually **did** plan a project. If you wrote out a project plan, include that.

It is fine to write about things that didn’t work as well as things that did.



### 1.1.2 Version management

Discuss version management and version control tools. Which techniques or tools did you choose to use? Why? How did your tool impact on your project for good or bad?

## 1.2 Project components

Go into more detail about what the project was about. You can be a little more technical here, but try to avoid too much detail.

### 1.2.1 Problem decomposition

How did you break it up into manageable chunks? How did you break them up further? What was the rationale behind the way you broke the project up into smaller pieces? When you talk about the various components you ended up with you can go into detail if you think it helpful. The person reading this may not necessarily be a coder though. They are interested in why you broke things up into parts, not how the parts work internally.

Trello/comment on the program

What parts are necessary to make your program work? How did you come to the conclusion about the makeup? Remember any problem can be solved in an almost infinite variety of ways, so why did you pick your way? Tell a story that helps the reader to understand your decision making process.

### 1.2.2 Component trialing

For each of the components, discuss how it evolved over the course of the project. What changed? What other options did you try to solve the mini-problem that was encapsulated by the particular component? Show why you are confident that the approach and components you have chosen are the right ones for the job.

### 1.2.2 Trialling of multiple components

Did you try different combinations of components together? If so, what did you discover?

Did you try multiple different ways to address one or more components? Which were best, and why?

### 1.2.3 Results of trialling

What changes did you make as a result of your trialling? How did these changes make your project better? Show how your program has evolved over time. Provide evidence of the change (e.g. screen shots, or Github timelines)

## 

## 

## 1.3 Relevant implications

Your project has an impact on people. These are encapsulated in the relevant implications.

### 1.3.1 Relevant implication #1 (give the name)

Describe the relevant implication in general terms

Explain how it impacts on your project.

Discuss what you have done to address this implication.

### 1.3.2 Relevant implication #2 (give the name)

Describe the relevant implication in general terms

Explain how it impacts on your project.

Discuss what you have done to address this implication.

### 1.3.3 Relevant implication #3 (give the name)

Describe the relevant implication in general terms

Explain how it impacts on your project.

Discuss what you have done to address this implication.

|  |  |
| --- | --- |
| **Implication** | **Why is it relevant?** |
| social | It should be a fun game that you can play with your friends and family. The game should have time limit or best score for more fun. In my game It might be too hard for an eight   year old to play  because it requires lots of logical thinking. I can  made accescable to any age changing a different level in the game. |
| usability | The game should be able to play with minor errors and be easy to understand with clear instruction. Also it shoul be easy to use for everyone(e.g color blind). |
| sustainability and future proofing | A programe that easy to maintain and update overtime, programe that canbe scale and extend in the future time. Programe with enough information on the code good structure that easy to understand for future development. |
| end-user considerations | The end-user should have a positive experience with the game. The game should run smoothly without any bugs or errors. An easy to use and understandable game user interface and game control e.g. how players interact with the game (keyboard, mouse). How the game communicates with the user e.g. wrong input, win or lose. |

# Part 2: Testing

Describe your testing regime. Include your test plans and test history.

## 

## 2.1 Expected conditions

## 2.2 Boundary events

## 2.3 Invalid situations



## 2.4 Changes resulting from testing

Did you make any major changes as a result of testing? If so, what, and why?

# 3. Summary

All the planning, trialing and testing has led to a great final program. Talk about that, summarise how each of these (both individually and in combination) have contributed to that excellent final product that you have produced.

Provide evidence from your project to back up your discussion.