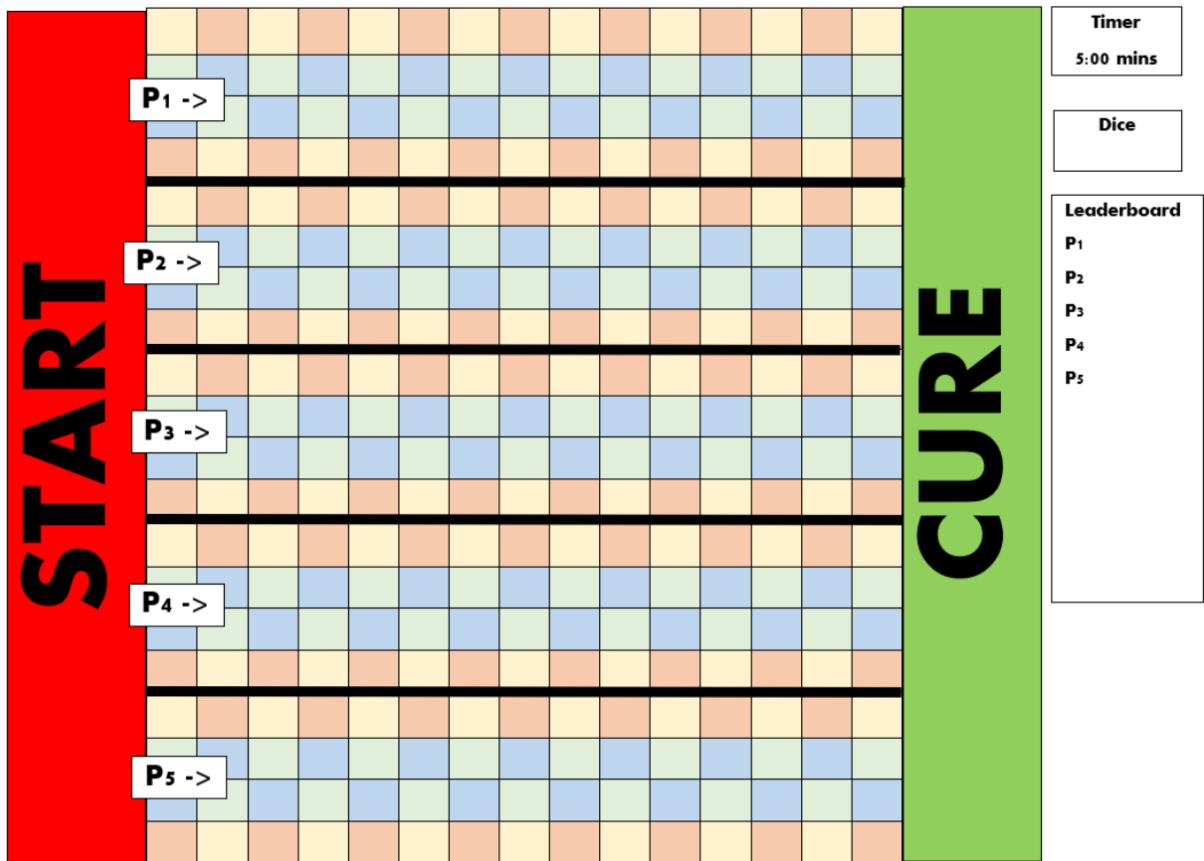


CMPT 276 - Phase 2

Build Automation: python3 "pandemic.py"

Prototypes:

Game Board Prototypes:



Niroshan's Prototype Discussion Pros and Cons:

Pros:

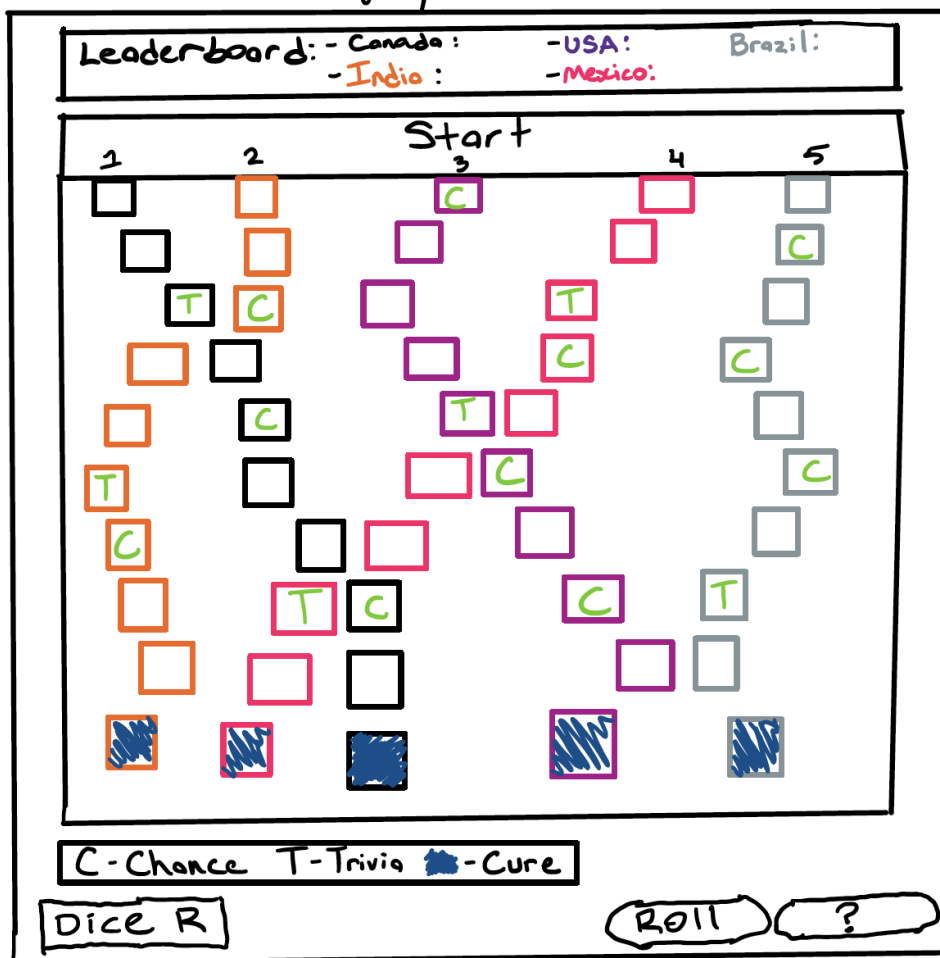
- Each player has their own lane; this removes the possibility of crossing paths. Additionally, it can lower the cognitive load on new players.
- Bold colours on START and CURE improve readability

- Bold separating lines are used to display each lane separate from each other player; it is consistent throughout the board and is easily recognized
- Consistency/Similarity. Board cells are structured very consistently throughout, with no obscure changes or sudden misalignments.

Cons:

- Lack of graphics on UI features inhibits recognition of what each button does
- More interesting UI features that accelerate or slow down gameplay could also be added.

Main Gameplay



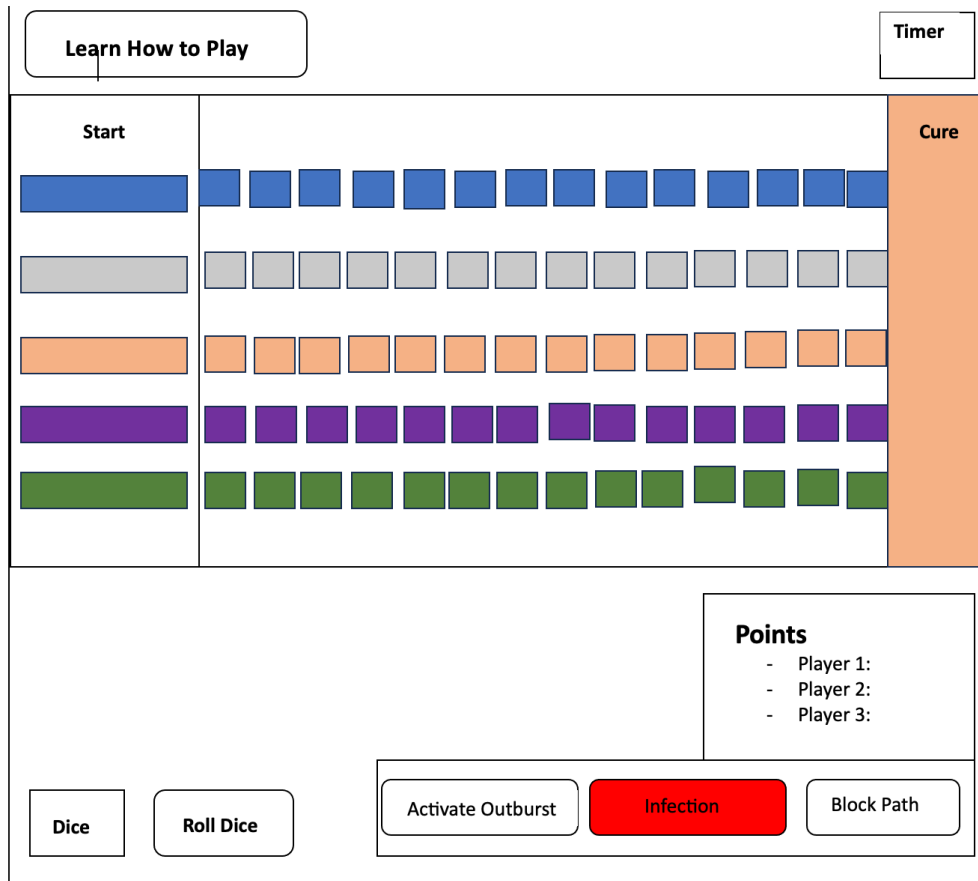
Gavins Prototype Discussion Pros and Cons:

Pros:

- Each player is given a colour; their path and country are shown in this colour for overall consistency
- User is able to get help by clicking the “?” button in case they are confused about how to make their move
- Recognition > Recall, the “?” symbol is the universal help symbol
- Each player’s coloured path allows for a more readable board
- All info shown is easy to understand, decreasing the cognitive load

Cons:

- Overlapping paths makes the board difficult to follow, decreasing the clarity
- The use of the blue-green colours could be difficult for vision-impaired users
- The state of the game is not communicated clearly. No indicator showing which players turn it is



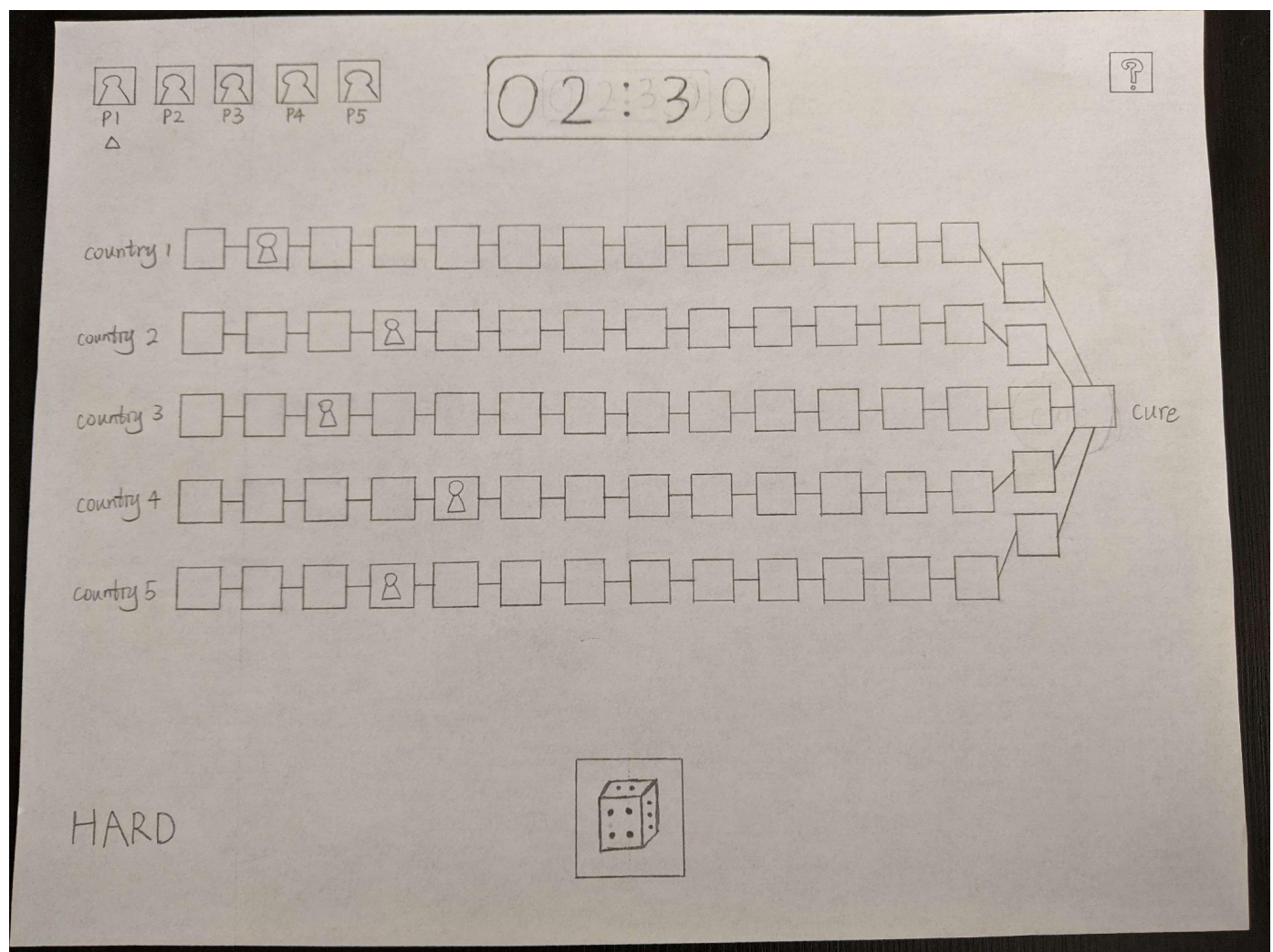
Abdullah's Prototype Discussion Pros and Cons:

Pros:

- The use of different colours for each path allows for readability and clarity.
- Each path is of a fixed length and direction, allowing for consistency among paths
- Users can learn how to play the game through a guided tutorial button, which decreases the cognitive load on players.

Cons:

- Button colours are inconsistent, the red infection button is used to show dangerous/strong upgrades but no colours are given to other buttons
- Bolder colours on some buttons could inhibit colour blindness accessibility as colour is the only thing that displays information to players.
- Lack of graphical images in the UI does not encourage recognition.



Kilin's Prototype Discussion Pros and Cons:

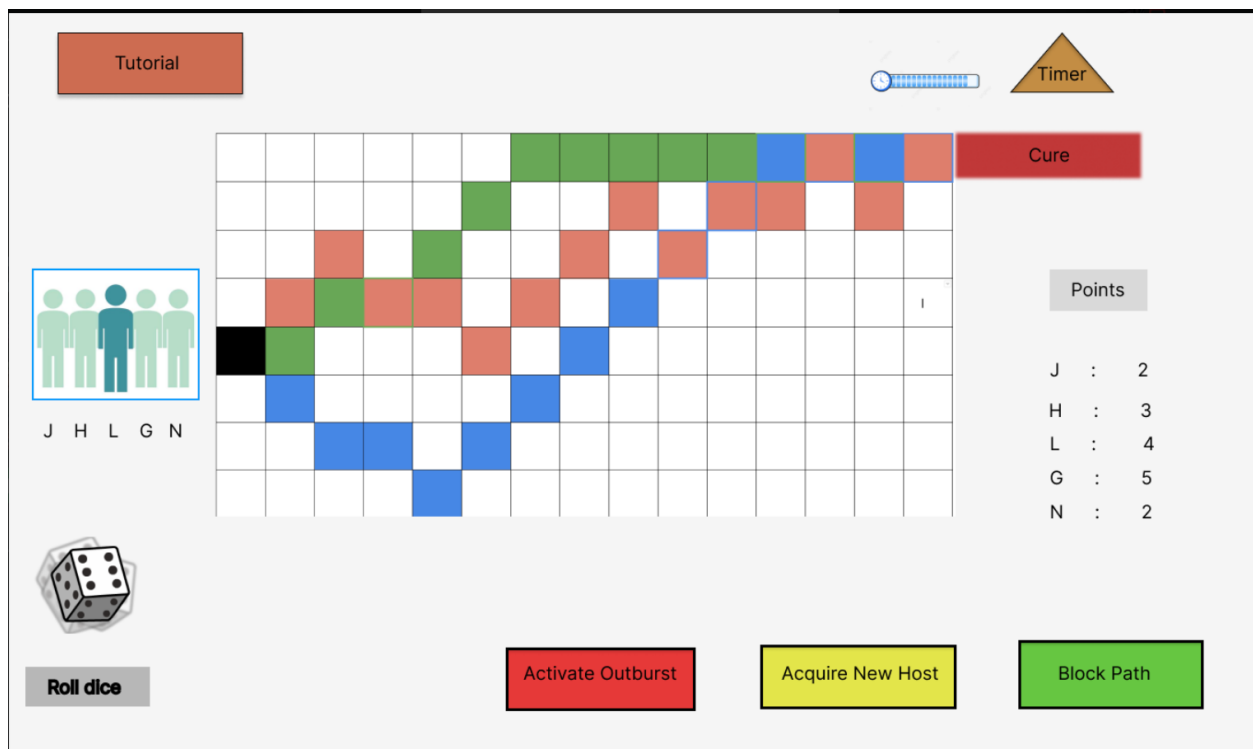
Pros:

- Icon on the straight path allows players to understand their current rank easily.
- Name of the country is shown on the left-hand side of each path; players do not need to recall.
- Indicating whose turn it is, even an AI bot.(top left)
- UI Indicates how many players and how many AI bots are in the current match.

- Use the icon, for example, question mark and dice, to indicate the help and roll dice buttons.

Cons:

- Player has minimum control: only the roll dice button on the gameplay interface.
- Lack of contrasting colours to help people recognize different paths instantly.
- Lack of text information to guide new players.
- Players have no idea what exactly happened after their actions; they only know the consequence.



Jaideep's Prototype Discussion Pros and Cons :

Pros:

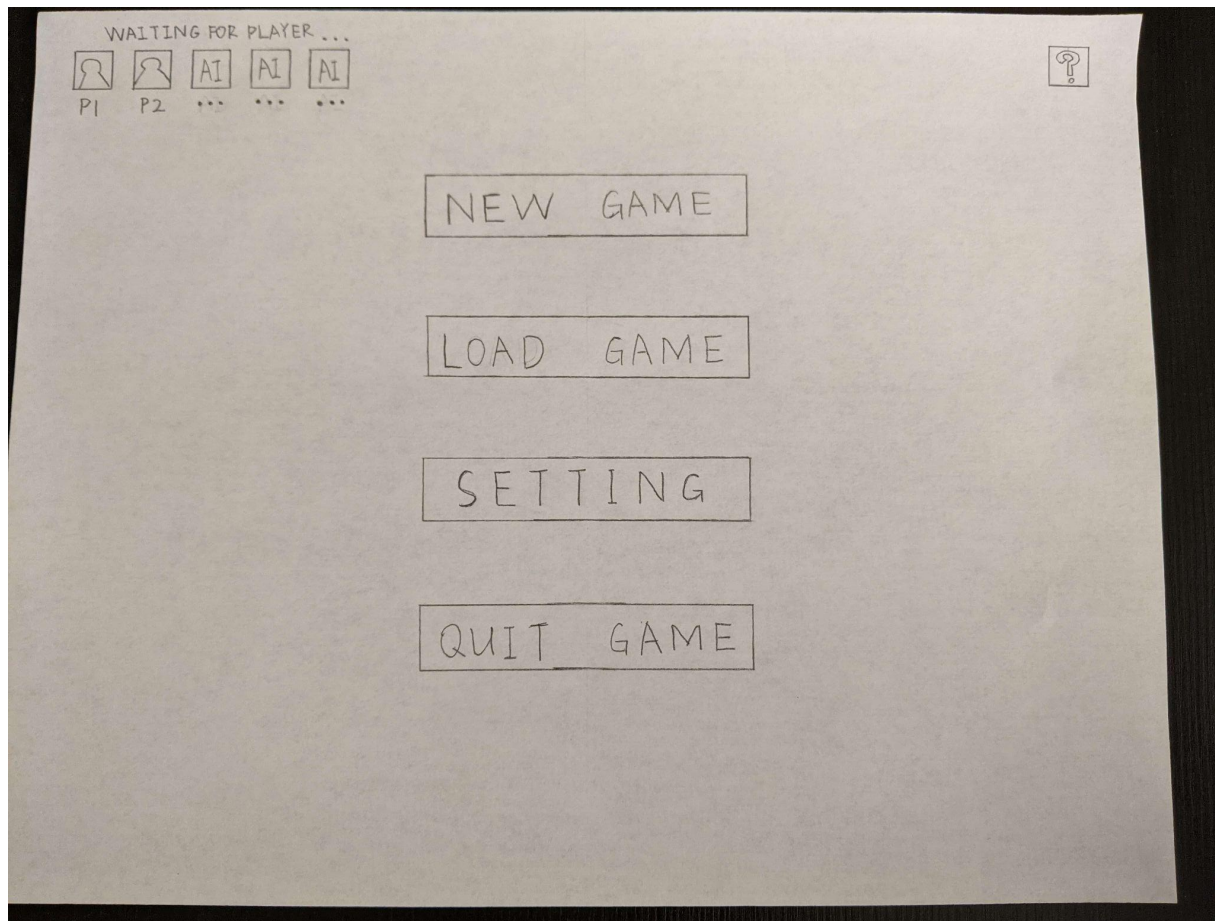
- Used bolder colours for interactive user buttons and softer colours for non-interactive buttons

- Tutorial for decreasing cognitive load and user help
- Contrasting colours (3 steps away) for action buttons
- Buttons similar in action grouped together abiding proximity (Gestalt principle)
- Each player has different coloured paths for similarity (gestalt principle)

Cons:

- The board will violate the similarity Gestalt principle because, at some point, multiple paths will intersect
- No screen magnifiers or screen readers reduce accessibility
- UI doesn't communicate the state of the game clearly: it doesn't show the position of players on the board, nor does it show whose turn it is
-

Main Menu Prototypes:



Kilin's Prototype Discussion Pros and Cons:

Pros:

- Minimum design, simple and easy to understand.
- Consistent font choice, font size, and button size.
- Indicator of player list (human player or AI), including the order of players.

Cons:

- Miss of colour indication.
- The upper right question mark button kind of broke the consistency and generally players probably don't need specific help operating the main menu. I will consider to add a "tutorial" button instead to include the help.



Niroshan's Main menu Prototype

Niroshan's Prototype Discussion Pros and Cons

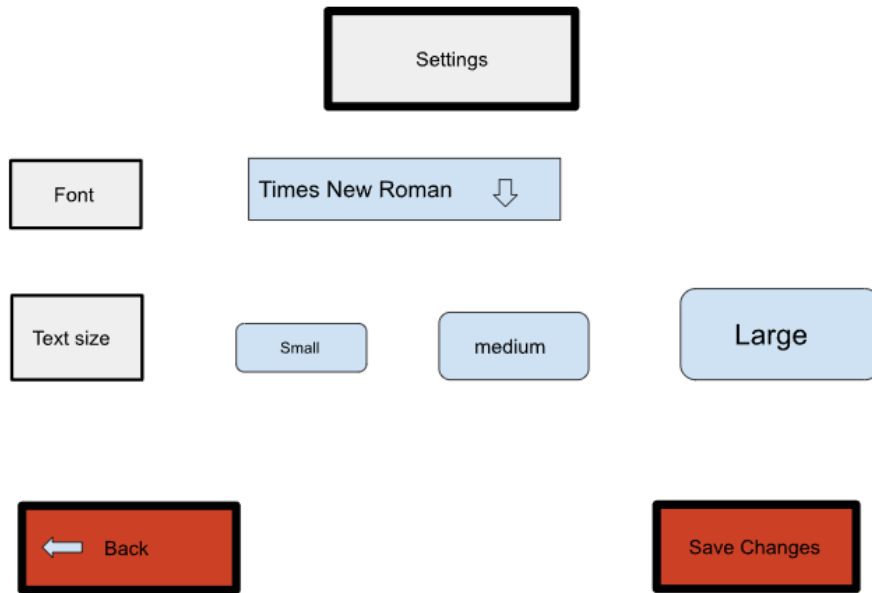
Pros:

- Includes a guide which will allow the user to learn the game's functionality
- Uses a coloured outline for each button with contrasting colours for "start game" and "quit"

Cons:

- Buttons sizes are inconsistent.
- Used the colours red, green, and blue, which should be avoided to help colour blindness accessibility

Settings Prototypes:



Jaideep's Prototype Discussion Pros and Cons

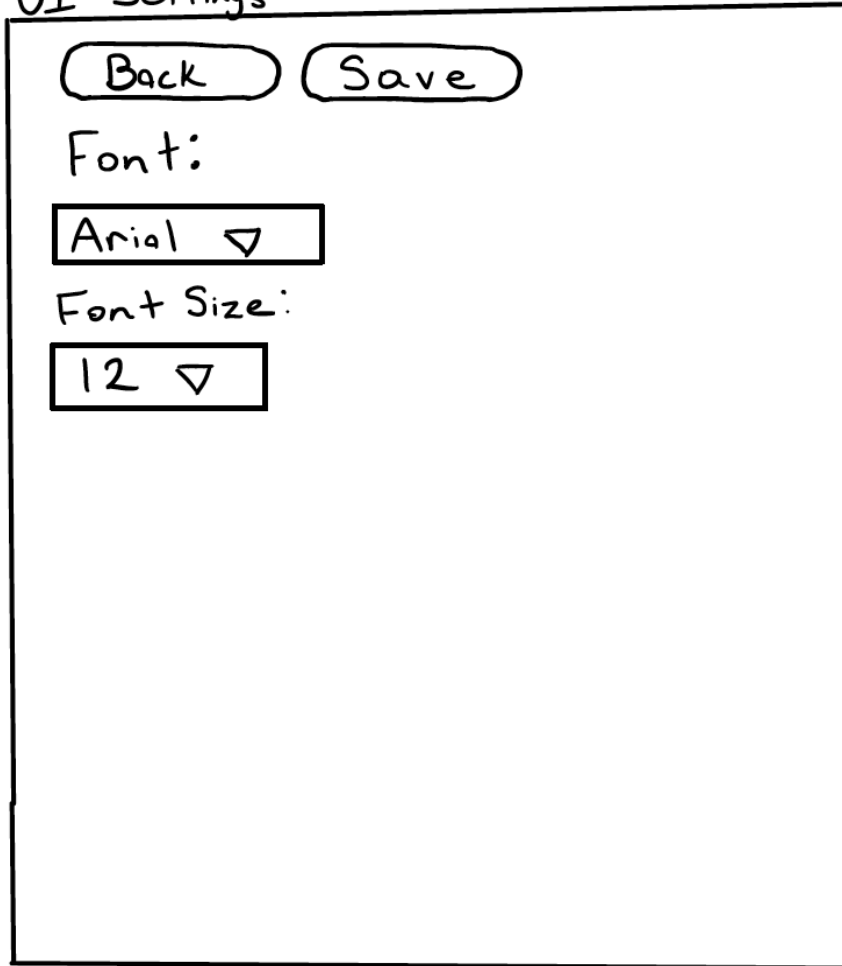
Pros:

- The font option has a drop-down menu for Recognition > Recall
- The text size options are grouped together yet separated by sizes taking advantage of Similarity and proximity (Gestalt principles)

Cons:

- Only uses bright colours for the back and saves options
- Even though different text sizes are used for better recognition, they could be causing inconsistency in the UI

UI Settings



A hand-drawn sketch of a 'UI Settings' window. The window is a large rectangle. At the top left, the title 'UI Settings' is written. Below the title, there are two rounded rectangular buttons labeled 'Back' and 'Save'. Below these buttons, the text 'Font:' is followed by a rectangular box containing 'Arial' and a downward-pointing triangle. Below that, the text 'Font Size:' is followed by a rectangular box containing '12' and a downward-pointing triangle.

Gavin's Prototype Discussion Pros and Cons:

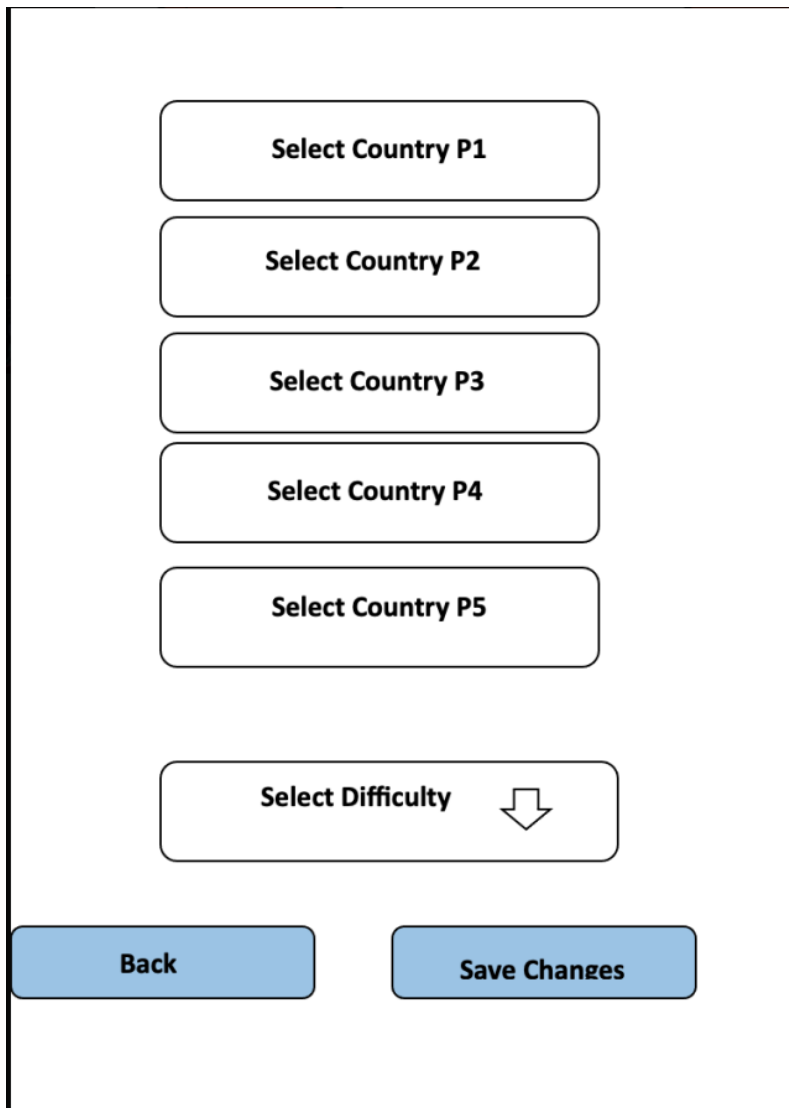
Pros:

- Recognition > Recall. The font type is shown in the drop-down box, making it easier for the user to choose which font they want
- Clear and simple design for ease of use

Cons:

- Recognition > Recall. The font size number is shown rather than the size, which would be more helpful
- UI lacks the use of colours to help with clarity

New Game Menu Prototypes:



The diagram illustrates a 'New Game Menu' prototype. It features a vertical stack of five white rectangular buttons with rounded corners and black borders. The first four buttons are labeled 'Select Country P1', 'Select Country P2', 'Select Country P3', and 'Select Country P4' from top to bottom. The fifth button is labeled 'Select Country P5'. Below these is a sixth white button with rounded corners and a black border, labeled 'Select Difficulty' followed by a downward-pointing arrow icon. At the bottom of the menu are two blue rectangular buttons with rounded corners and black borders, labeled 'Back' and 'Save Changes'.

Abdullah's Prototype Discussion Pros and Cons:

Pros:

- Button location similarity and proximity allow for easy differentiation between button functionality.
- The colours of “back” and “save changes” give boldness to movement between different user interfaces.

- Players can choose which country they want to select, which allows for usability and customization.

Cons:

- Buttons are very similar in function; extension of this interface to more than five players would be cumbersome.
- The select country buttons for each player have no way of distinguishing themselves from others other than by text, do not encourage recognition and could be difficult to use for visually impaired players.

Choice of Overall Prototype Design and Why

Game Board:

We will be choosing Niroshan's Game Board prototype.

How did our final board prototype evolve from Niroshan's Game Board prototype?

- From Niroshan's design, each player follows their own lane toward the cure, having space for movement without any overlap with another player, which makes the game board clearer and much more readable. This improvement was developed from looking at other members' boards and noticing that board movement was hard to read due to paths overlapping.

- We decided to incorporate the following UI features from Abdullah's/Jaideep's/Gavin's prototype into Niroshan's prototype:
 - Dice graphic/Roll dice
 - Guided tutorial feature.
 - Leaderboards feature
 - Point system
 - Upgrades (bought with points)
- One new design element we discussed for the final prototype, that our initial designs lacked is conveying the state of the game clearly. This was done by adding a turn indicator and a win indicator.
- We changed the board layout from Niroshan's board prototype slightly. The board now displays purple columns of trivia cards with each cell labelled with the letter "T"; this addition, along with a trivia popup feature, clearly displays the game's state. (for both the player and the AI).

Main Menu:

We will be using Kilin's main menu prototype

- This main menu prototype includes all the necessary buttons that we decided to include on the main menu
- There is a feature that portrays functionality similar to the "party room" feature present in most modern multiplayer games. This is another example of displaying game state clearly as it shows which players have joined and whether an AI is active.

Settings:

We will be using Jaideep's settings prototype

- The use of colour emphasis on the “back” and “save changes” buttons is useful to have as they differentiate functionality from other dropdown menus or buttons.
- This prototype also uses recognition when having players choose their font size and go back. (buttons for font size different sizes for “small,” “medium,” and “large”).

Design Patterns Used:

1.) Abstract Factory

We implemented Abstract Factory in player.py.

The Abstract Base Class is named **CharacterFactory**, and it contains two abstract methods:

1. **update_position()**: this abstract method provides the basic movement logic of each character in the game.
2. **move()**: this abstract method defines a random movement of the character based on the outcome of a dice roll

There are two concrete factories: **Player** and **AI**

1. Player :

a) **update_position()**: This abstract method is different from the base class method as it defines boundaries for character movement on the board. (rows and columns)

b) **move()**: This method just passes in the player subclass as the user will control the player and wouldn't need random movement.

2) AI :

a) **update_position()**: This method is different from the base class as it defines boundaries for AI movement (rows and columns)

b) **move()**: This method defines specific wall conditions for the AI. The AI can't move up when it's at the starting row and down when it's at the bottom row. This method also defines how the AI moves randomly across the board depending on the dice roll.

- The constructor of both the Player and AI subclasses is inherited from the Abstract Base Class using `super.__init__()`.

We chose to implement abstract factory because we recognized that the two main entities that are moving on the board in our game (player and AI) extend from a general player character. This allowed us to achieve parallelism and readability in our code as we have now implemented classes that have specific functionality tended towards a particular type of player character.

2.) Prototype

We implemented a Prototype in `triviacard.py` to assist with object creation. For our final game, we will need to create 60 **TriviaCard()** objects, each requiring an expensive database call (Trivia Card questions and answers will be stored in a database). To solve this problem, we have implemented a `clone()` function in **TriviaCard()**, which creates a shallow copy of a trivia card. Using this function, we only need to create 12 unique `TriviaCard()` objects (1 per trivia card spot along a single player's path), which can then be cloned onto other players' paths. This avoids a large sum of expensive object creation/database calls.

3.) Observer

We implemented the Observer design pattern to notify the players anytime the state of whose turn is changed on the board. Inside the **Board()** class, we implemented several functions, `attach()`, `detach()`, `turn_setter()`, and `notify()`, and inside the **Player()** class, we implemented an `update()` function. Players are attached to the board as a list of observers, and upon the use of `turn_setter()`, all observers are notified of this turn change. This will be useful for future UI features we decide to implement because all players are notified of their turn at run time.