Proposed Game:

Tic-tac-toe

Game Title:

4Square

**GitHub Link:**

**Mentor:**

Kudzai Maziriri

By

Nyiko Ashanti Ngobeni

G22N3906

Christopher McCune

G22M5298

Alisha Schuler-Nitzsche

G22S7874

Mihlali Xhala

G22X7152

Mthobisi Latha

G22L5863

**Content page**

Overview 3

Goals and risks 3

Gnatt Chart

Technologies

Technical architecture 4

Storyboard interface representation 5

Object diagram (UML) 7

Code Outline

How we did it

Difficulties

**Overview**

On opening the app, the user will see the home activity, and from there they can click either the play button or the rules button, both will change the user’s view to suit what button they clicked. The rules activity gives an explanation of the rules. Each activity will have two buttons the user can click, one will allow them to always return to the home activity and the others will allow them to continue the game.

To code the game to work we will make a 2D array that stores mutable values that will change as the game goes on and players choose the move they want to make. To simplify the coding of board we implemented buttons across the actual game board in the squares that the player clicks on to place a symbol. As part of our rules, each players first move is required to be on one of the 12 outer blocks. After each move is made the code will check if someone has won or if there is a draw. If this occurs the view for the users will change to the results activity where, the board will be reset to blank and the user can click on the home button or the play again button which are both displayed as options.

**Goals and Risks**

We want to create a working android app that is our own creation in that it is a unique version of the game tic tac toe. Our goals, hence are centered around the keywords; “working” and “unique”. For the sake of completing this project, we as a group have identified that a desirable takeaway is to understand how the creation of a functioning android app happens and we would rather stray away from overcomplicating our unique game. The uniqueness of our version of tic tac toe is that we apply a four-by-four grid with the requirement of the first move to be on one of the outer blocks. This variation is enough for our game to be original and only poses the challenge of figuring out how to encode our first move requirement. The risks we may be undertaking due to our inexperience would be categorized into what we can and can’t identify. Categorically we are aware that our variation poses that challenge and creating an android app that fits onto a phone screen, in the required time, without overcomplicating our task.

**Technologies**

USER’S POINT OF VIEW

HOME

play

rules

**PLAY**

rules

Player’s Names

play

home

Game button takes you HERE, where player’s names are added and symbols are assigned

**HOME**

**PLAY**

Rules

home

play

Rules are displayed HERE. You can return home or play game.

**HOME**

**PLAY**

RESULTS

home

play again

HERE the result is displayed. You can then play again or return home.

PLAY AGAIN

**HOME**

Figure 1: Games Technical Architecture.

Game

restart

home

HERE the game is played. You can quit and restart. Once someone won the game the player will immediately be taken to the results activity.

**RESTART**

**HOME**

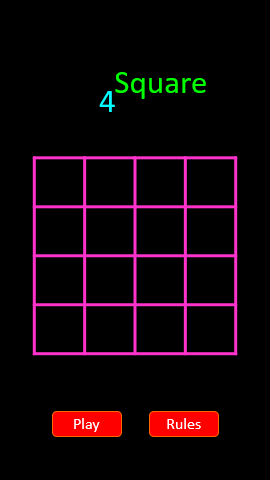
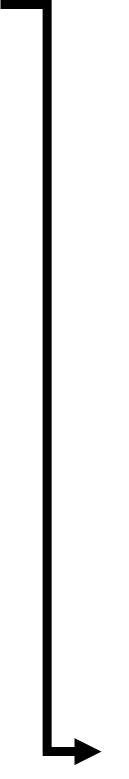
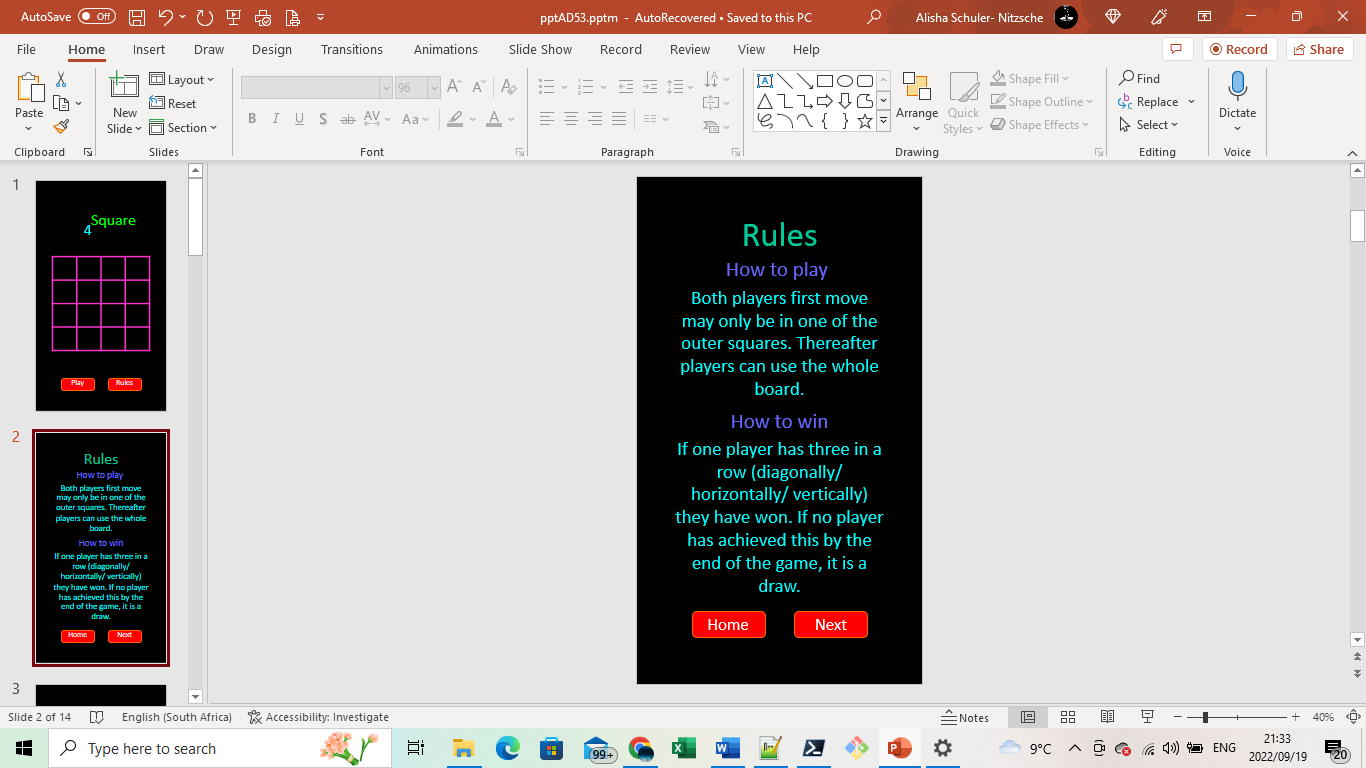
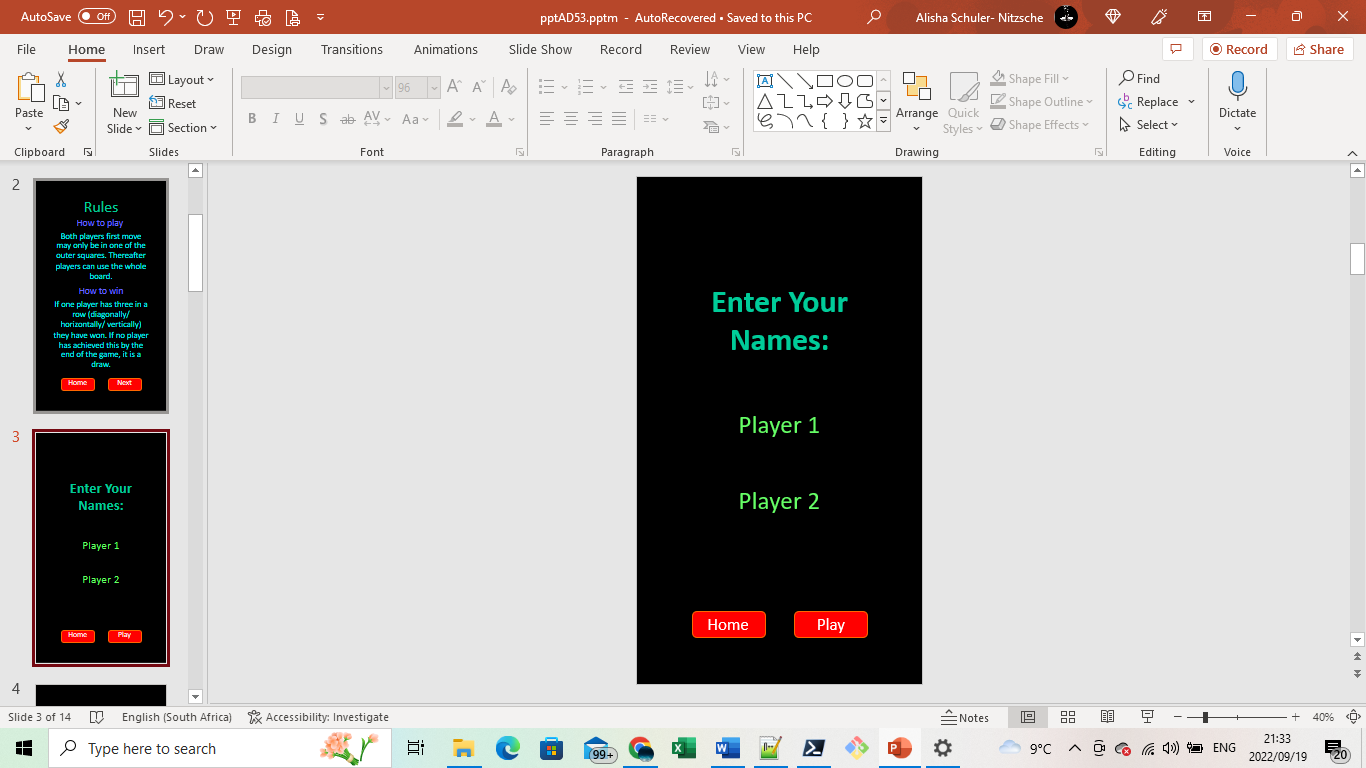
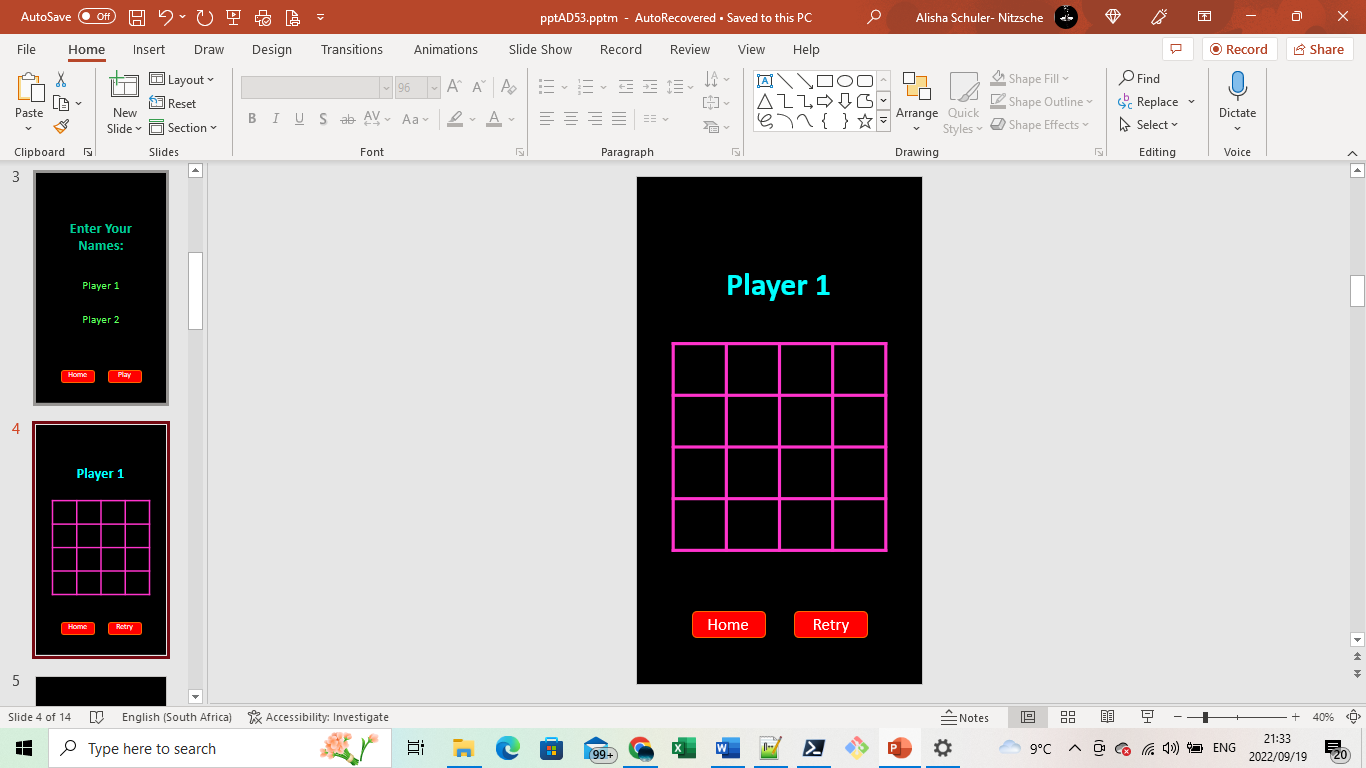
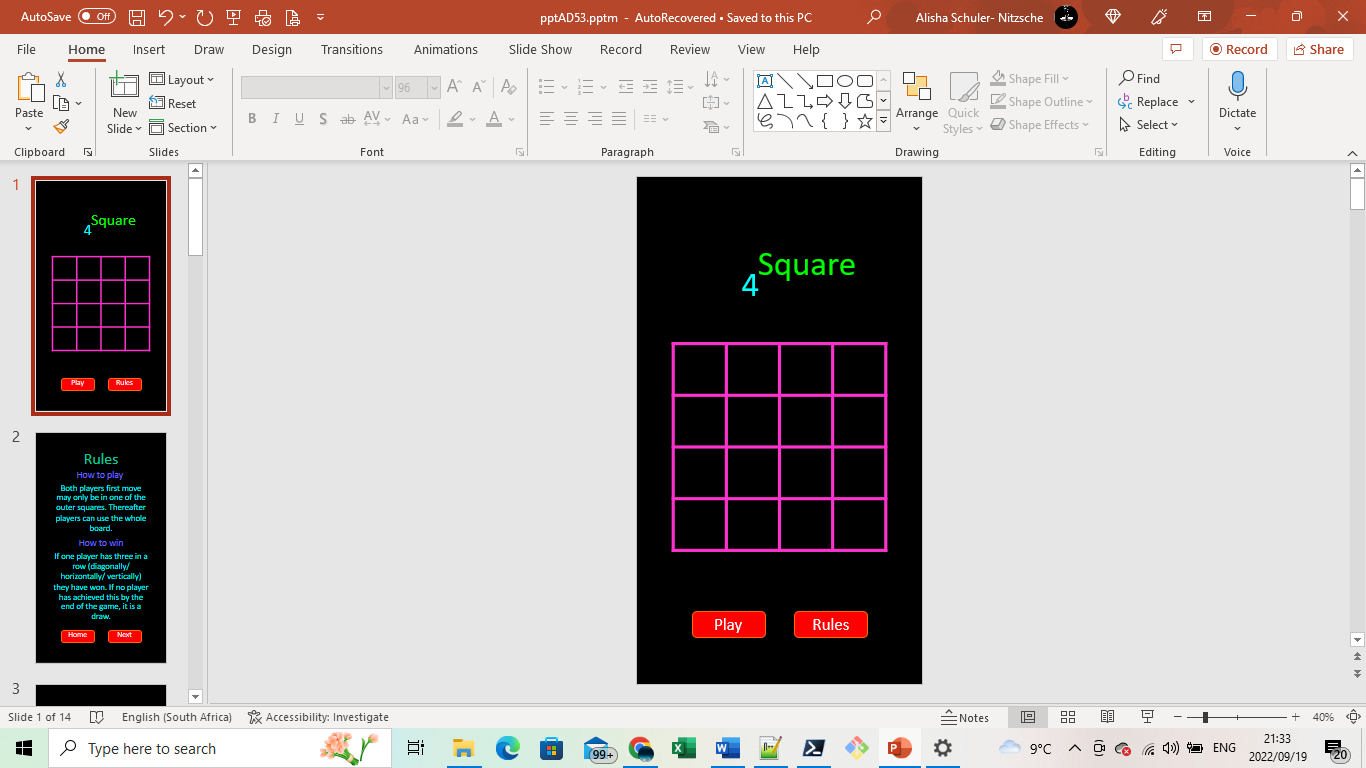
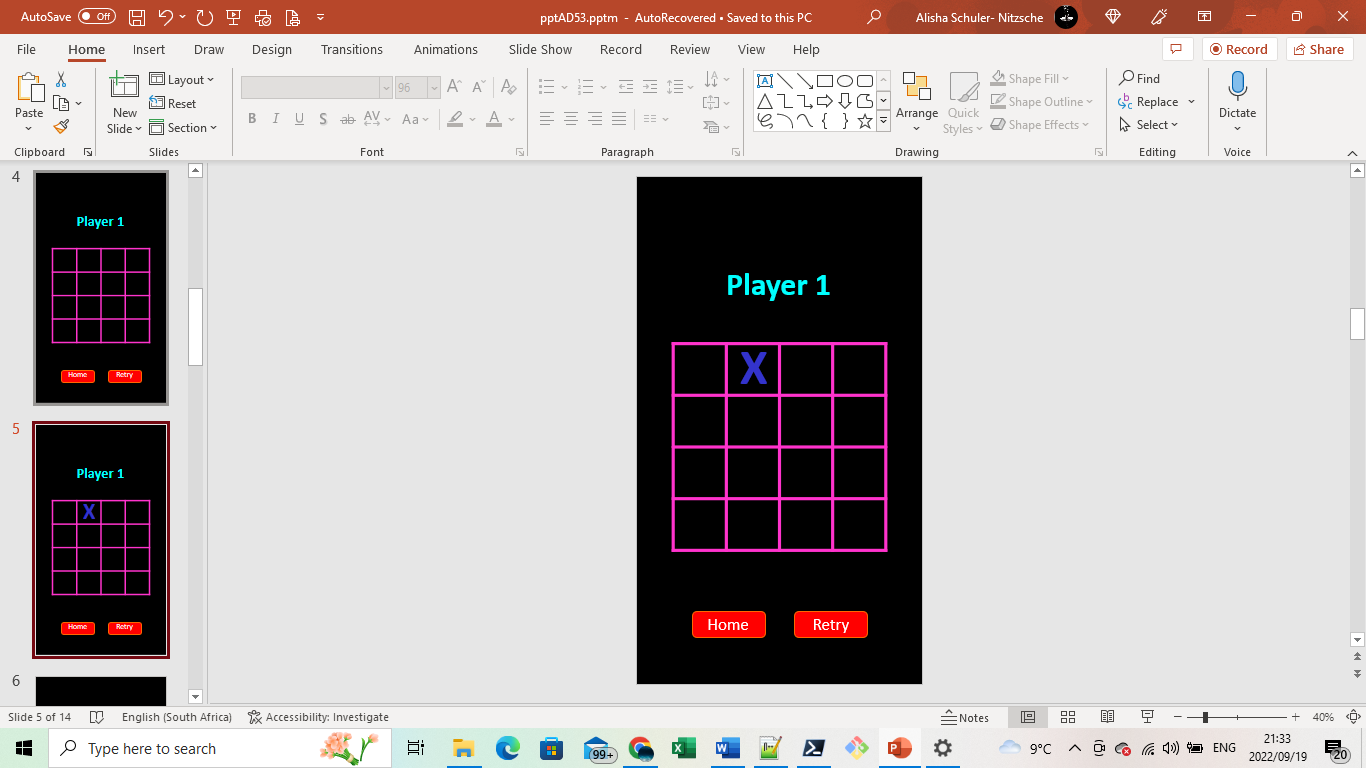
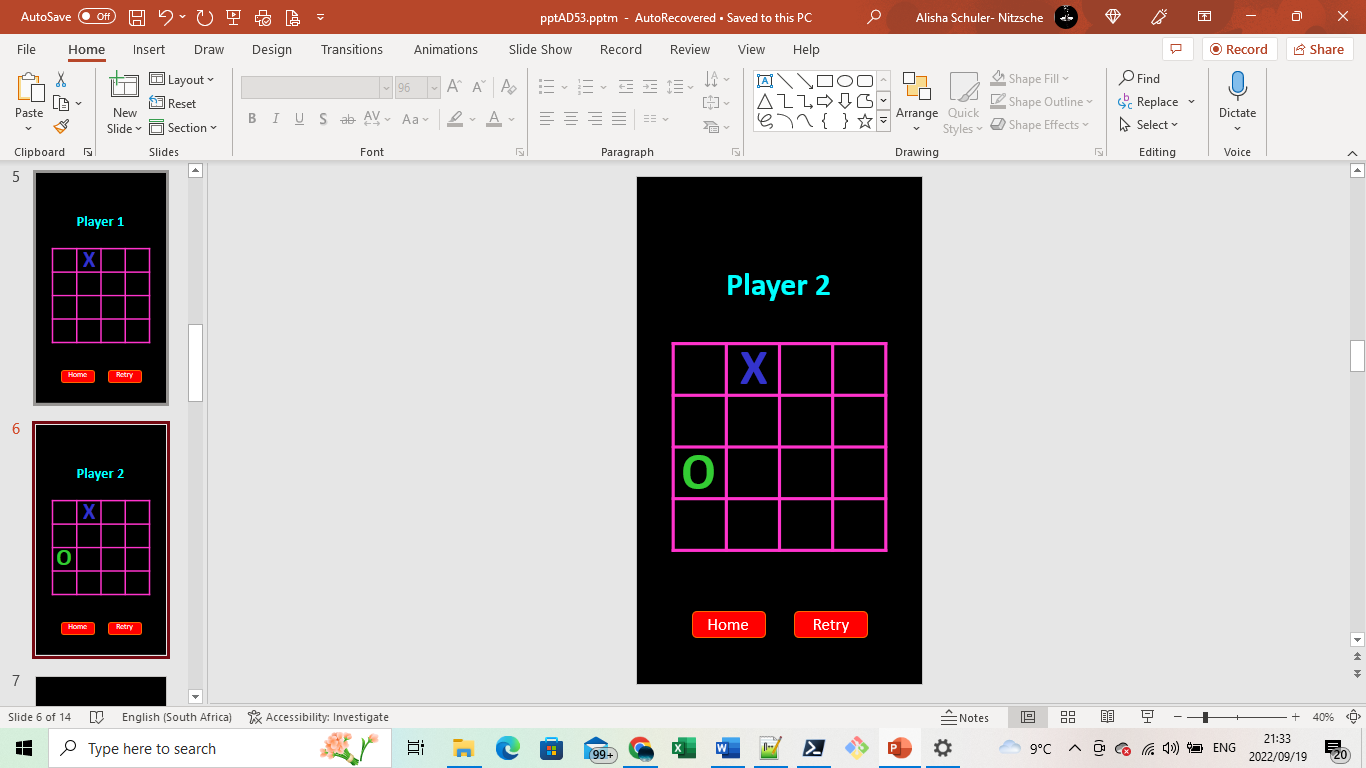
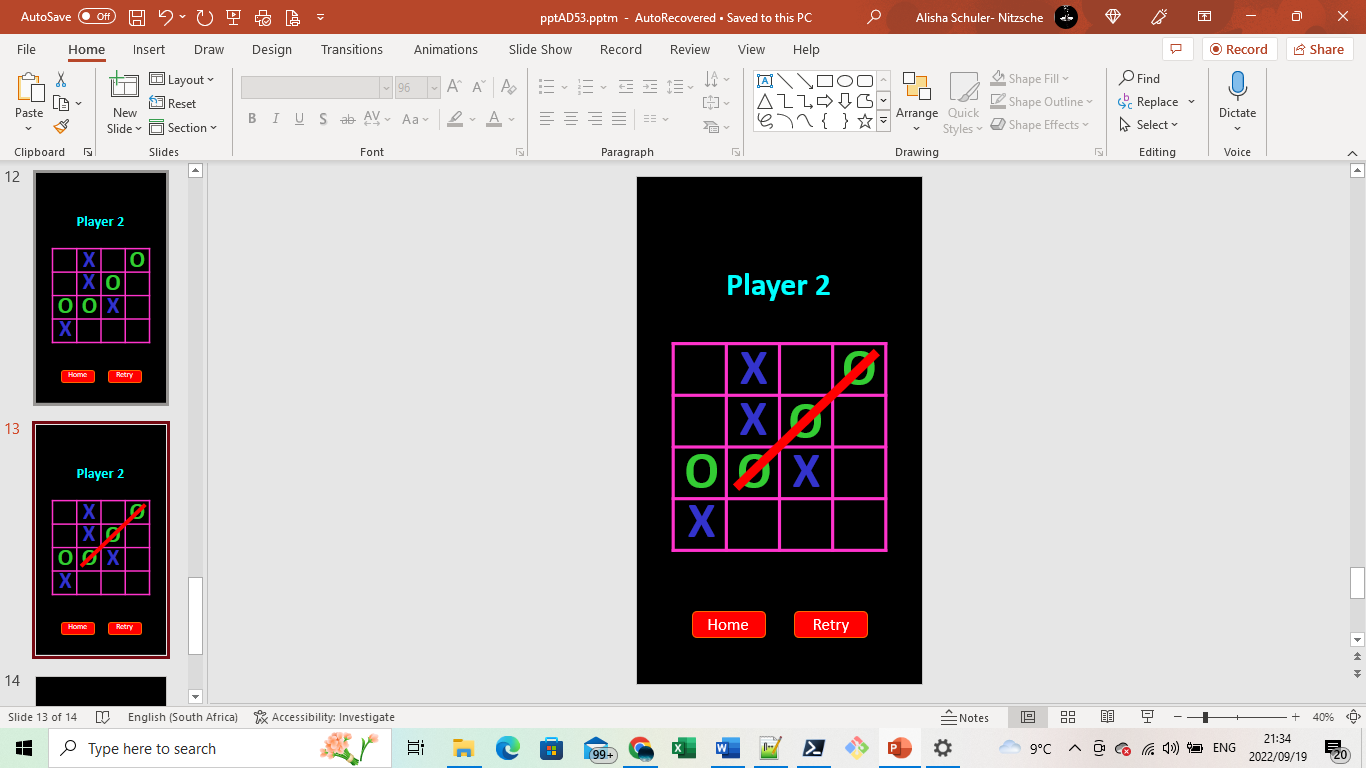
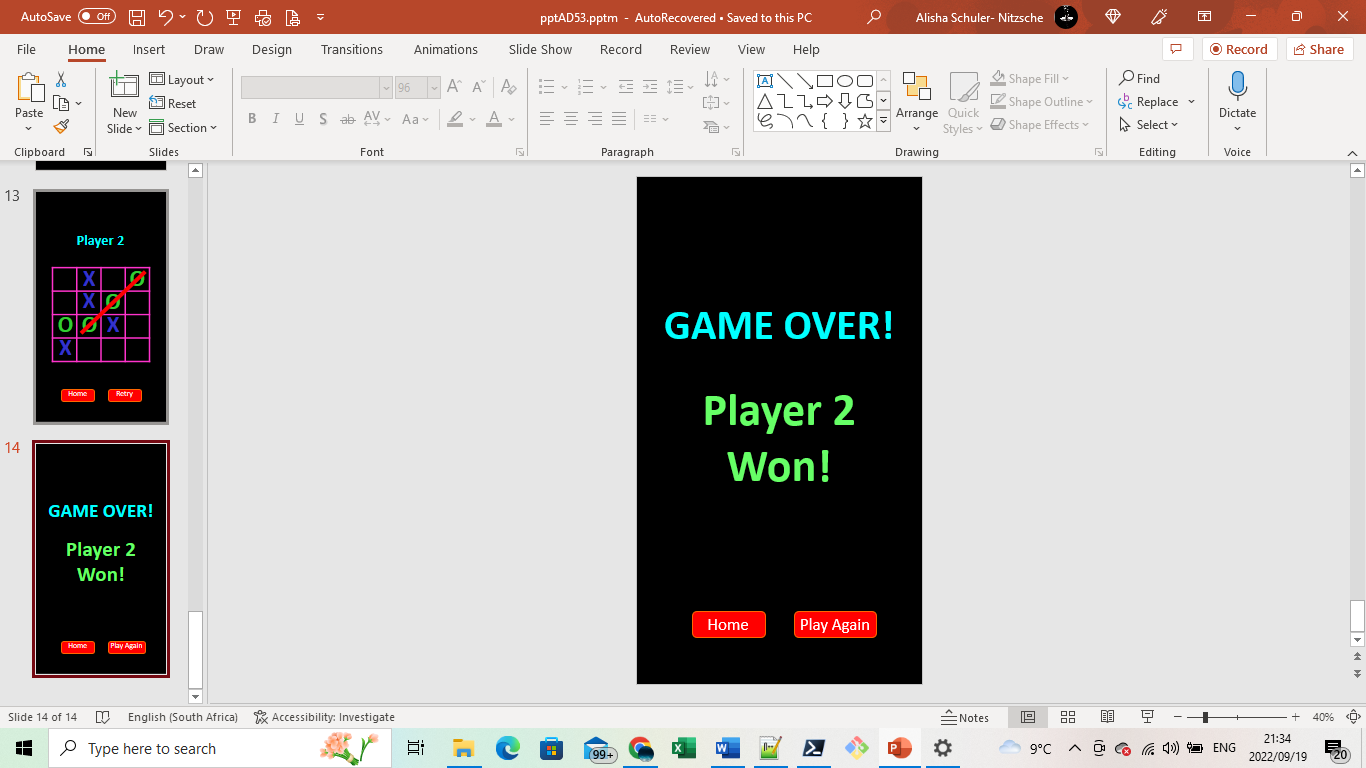
****

Figure 2: How players could potentially use the app.

**Story Interface Representation**

**![Shape

Description automatically generated with medium confidence](data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAAYMAAAA3CAMAAADZltb3AAAAAXNSR0IArs4c6QAAAARnQU1BAACxjwv8YQUAAAAVUExURQAAAAAAAAAAAAAAAAAAAAAAAAAAABIBAKQAAAAGdFJOUwAgYICf35XBnYYAAAAJcEhZcwAAFxEAABcRAcom8z8AAACuSURBVHhe7dK7EoJAEEVBBOH/P9kXKqDlbpUsN+mOJplkznQAAAAAANBK388DMcM0qhA2TJMKYbcGKmQ9GqiQ9GygQs67gQopywYqZKwb1FfY7rGnugoatFVTQYPWyhU0aK9UQYMjjKf53F9pcIDzzwQatFcooEFzxQIaNFZRgD19/LMCh9s0UCBg1UCBiEUDBUJeDRSImRsoEHRvoEDUtYECYYMCAAAAAAD/67oLUQZOlNvEeRUAAAAASUVORK5CYII=)**Home Activity Rules Activity Names Activity

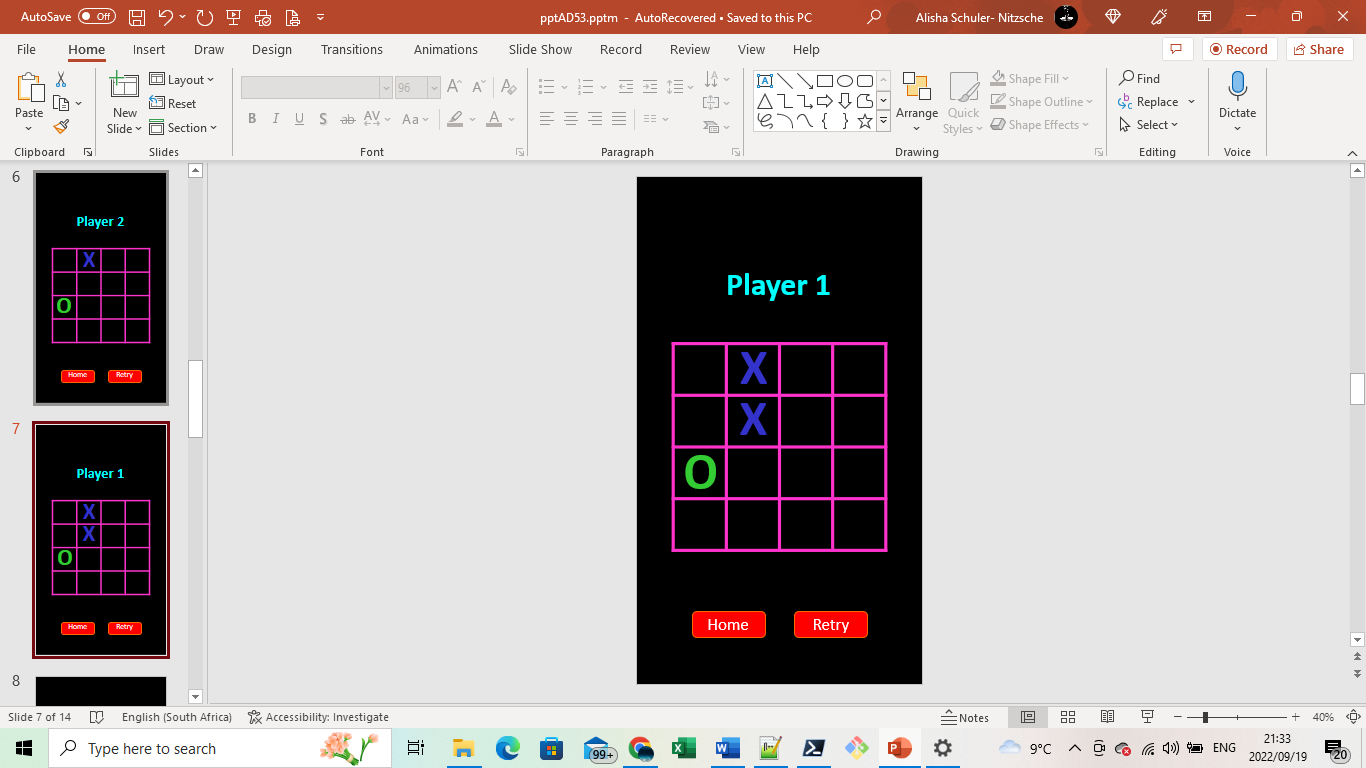
**Shape

Description automatically generated with low confidence**

Game  Activity

********![Shape

Description automatically generated with medium confidence](data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAAYMAAAA3CAMAAADZltb3AAAAAXNSR0IArs4c6QAAAARnQU1BAACxjwv8YQUAAAAVUExURQAAAAAAAAAAAAAAAAAAAAAAAAAAABIBAKQAAAAGdFJOUwAgYICf35XBnYYAAAAJcEhZcwAAFxEAABcRAcom8z8AAACuSURBVHhe7dK7EoJAEEVBBOH/P9kXKqDlbpUsN+mOJplkznQAAAAAANBK388DMcM0qhA2TJMKYbcGKmQ9GqiQ9GygQs67gQopywYqZKwb1FfY7rGnugoatFVTQYPWyhU0aK9UQYMjjKf53F9pcIDzzwQatFcooEFzxQIaNFZRgD19/LMCh9s0UCBg1UCBiEUDBUJeDRSImRsoEHRvoEDUtYECYYMCAAAAAAD/67oLUQZOlNvEeRUAAAAASUVORK5CYII=)

****

Results Activity

|  |
| --- |
| **<<Java Class>>**  **MainActivity** |
| No objects |
| * playbuttontap(View scr): void * rulebuttontap(View scr): void |
| @Override   * onCreate(Bundle savedInstanceState):void |

|  |
| --- |
| **<<Java Class>>**  **Name** |
| No objects |
|  |
| @Override   * onCreate(Bundle savedInstanceState):void |

|  |
| --- |
| **<<Java Class>>**  **Rules** |
| No objects |
|  |
| @Override   * onCreate(Bundle savedInstanceState):void |

|  |
| --- |
| **<<Java Class>>**  **Game** |
| * MoveCnt: int * GameBoard: Button[][] * Move: String[][] |
| * win\_checker(): Boolean * ResetButton(): void |
| @Override   * onCreate(Bundle savedInstanceState): void * onClick(View v): void * onClick(View scr): void |

|  |
| --- |
| **<<Java Class>>**  **Winner** |
| No objects |
|  |
| @Override   * onCreate(Bundle savedInstanceState):void |

Figure 3: Object diagram (UML)

**Code Overview**

**How we did it**

* started with designing activitys
  + Coded button format
  + Coded @String resources
  + Added TextViews and called hardcoded string resources
  + Added buttons
  + Constrained TextViews and Buttons
  + Changes text coulours, formats
  + Changed background of all activities
* Coded the button board and its input array
* Coded to check if the button clicked was empty or not
* Coded a move counter so that we can see whos turn it was
* Coded to print correct symbol into button when clicked
* Coded win checker by making a string array where each added symbol to button gets added to string[][], then for loop to check horizontally and vertical wins, manually check diagonals, call this method after each move
* Coded what happens when someone wins or there is a draw(ie gets taken to last activity)
* Coded to change the names ontop of the game board as players are playing
* Coded to bring winners name across to winner activity, if no one won and there was a draw it was hardcoded that defalt screen for last activity is “Game over! It was a Draw!”
* Coded working buttons
* Tested Game

**What we struggled with**

* Attempted hardcoings visual board drawing but haven’t got experience with drawing using code thus failed and used buttons instead for grid
* Bringiing names across boards and changing names on top was hard
* Getting the game to run on the emulator as we did not have a private laptop/computer to use and uni computers did not allow us to install HAXM extension needed to use emulator
* Uploading code to an already existing GitHub group, ended up having 5 different Github groups for the code, hence why its not visible that everyone contributed
* Getting the game to run in the end as we used coded for the game class first without coding each functunanilty button so we could try out if the game class worked until the end, made it difficult to see where we went wrong, should worked on the code for first activity firsts and moved from one activity to the next and not code middle activity first with no ways of accessing it to see if it worked.