Proposed Game:

Tic-tac-toe

Game Title:

4Square

**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

Technologies

Technical architecture 4

Storyboard interface representation 5

Object diagram (UML) 7

Code Outline

Visual Studios XML file 8

Visual Studios Java files

foursquare Class 10

gamelogic Class 14

gamedisplay Class 15

**Overview**

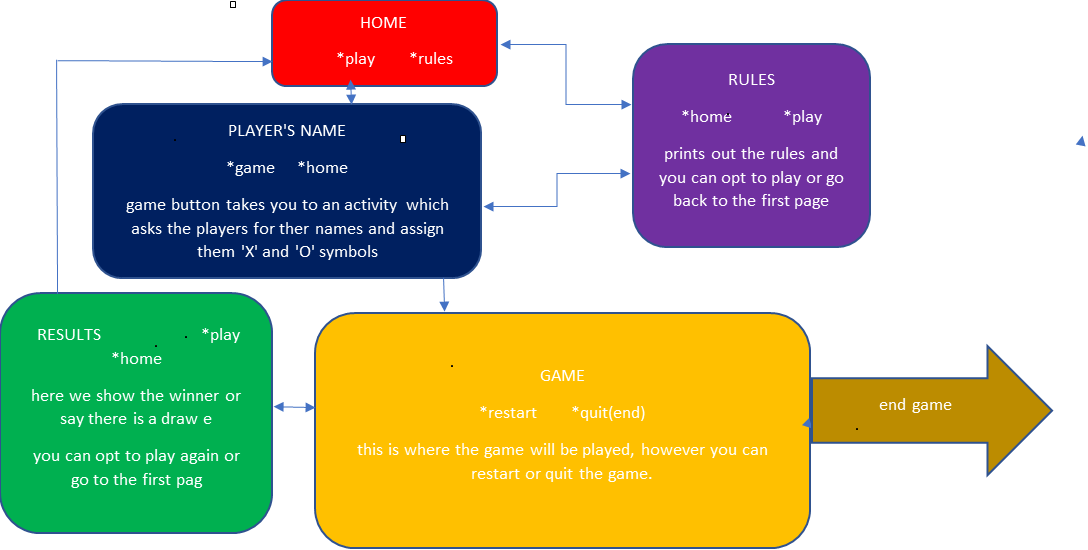
On opening the app the user will see the home activity, and from there they can click either the play button or the rules button, both will change the user’s view to suit what button they clicked. Each activity will have two buttons the user can click. One will allow them to always return to the home 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. 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 and the board will be reset to blank.

**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**



Once someone won the game the player will immediately be taken to the results activity.

Figure 1: Games Technical Architecture.

Key:

\* = buttons player will be able to click on the screen

Users View

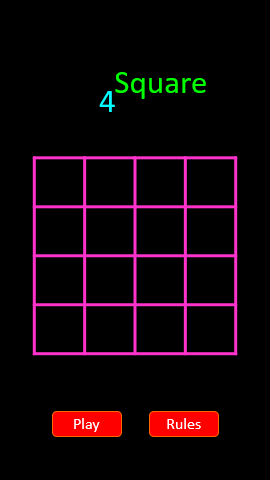
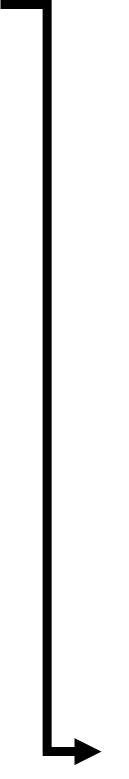
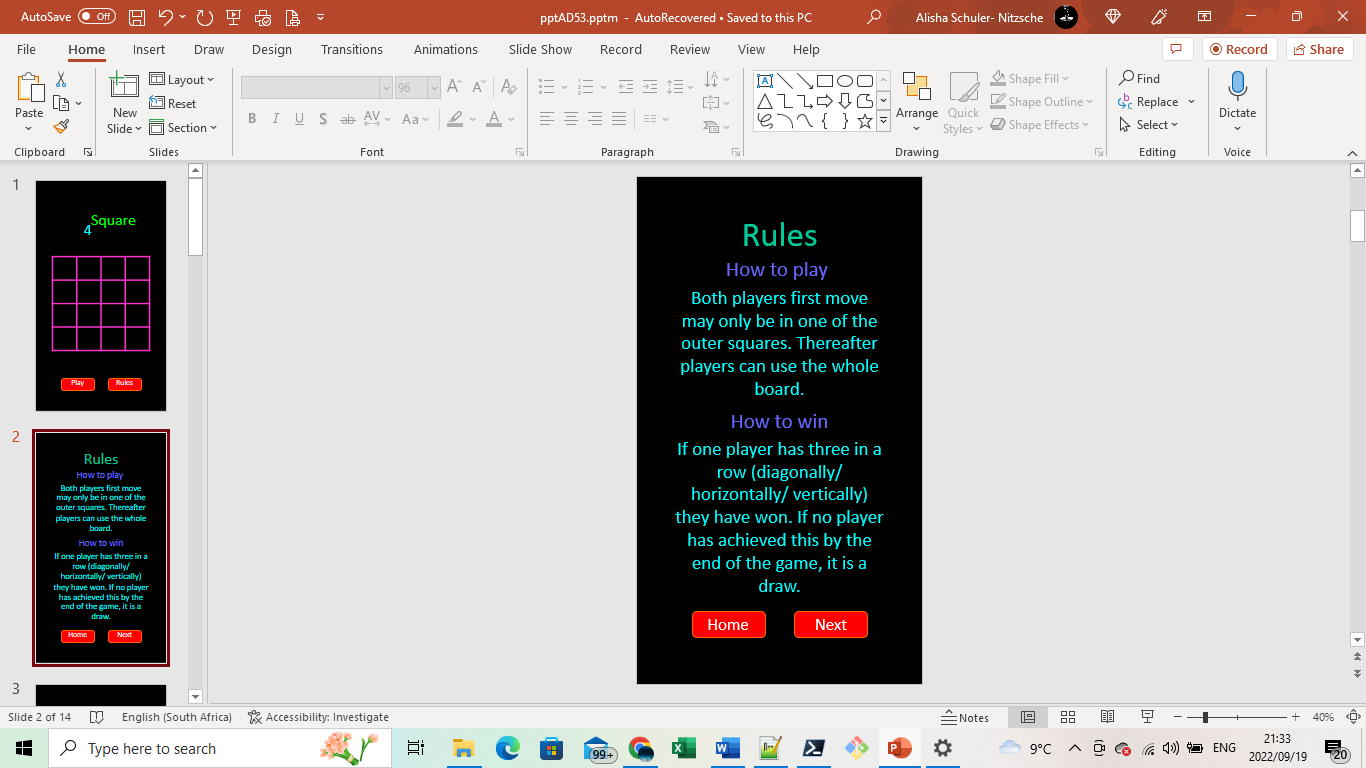
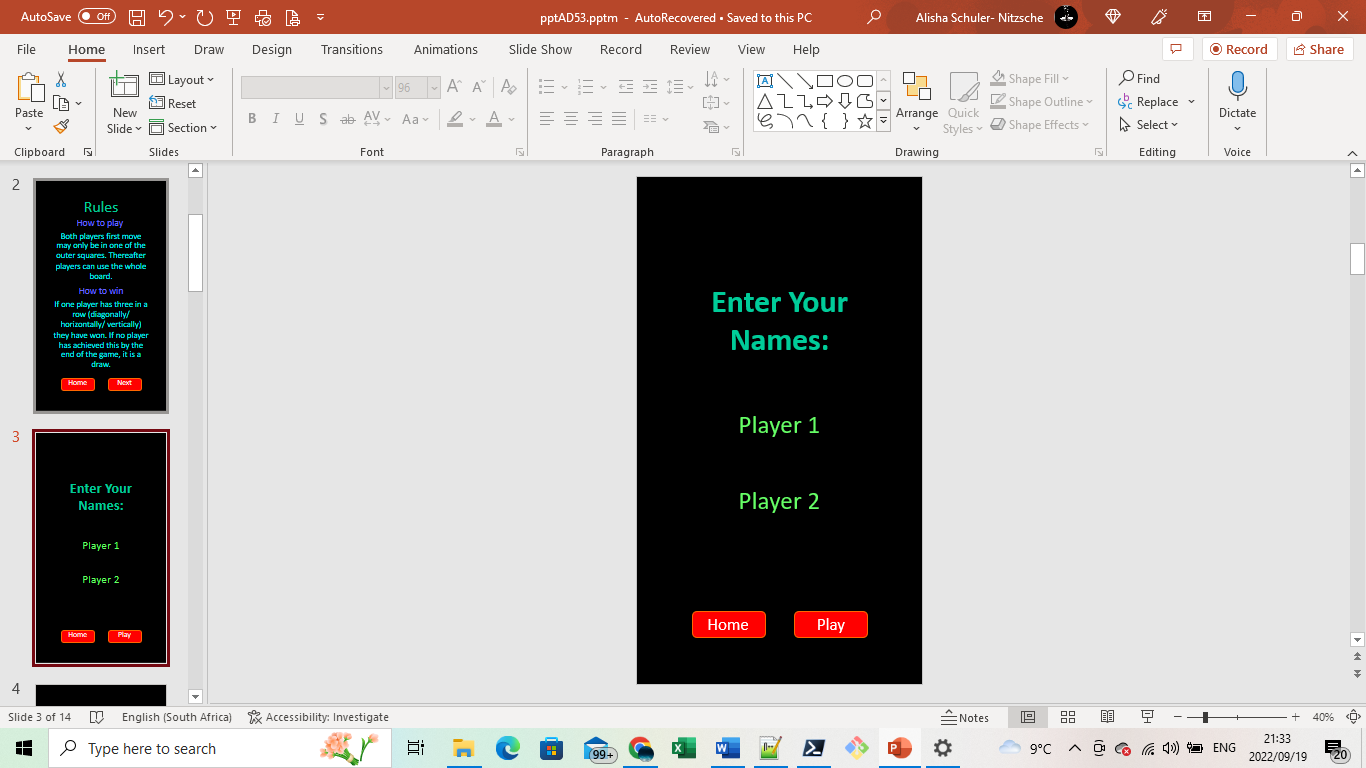
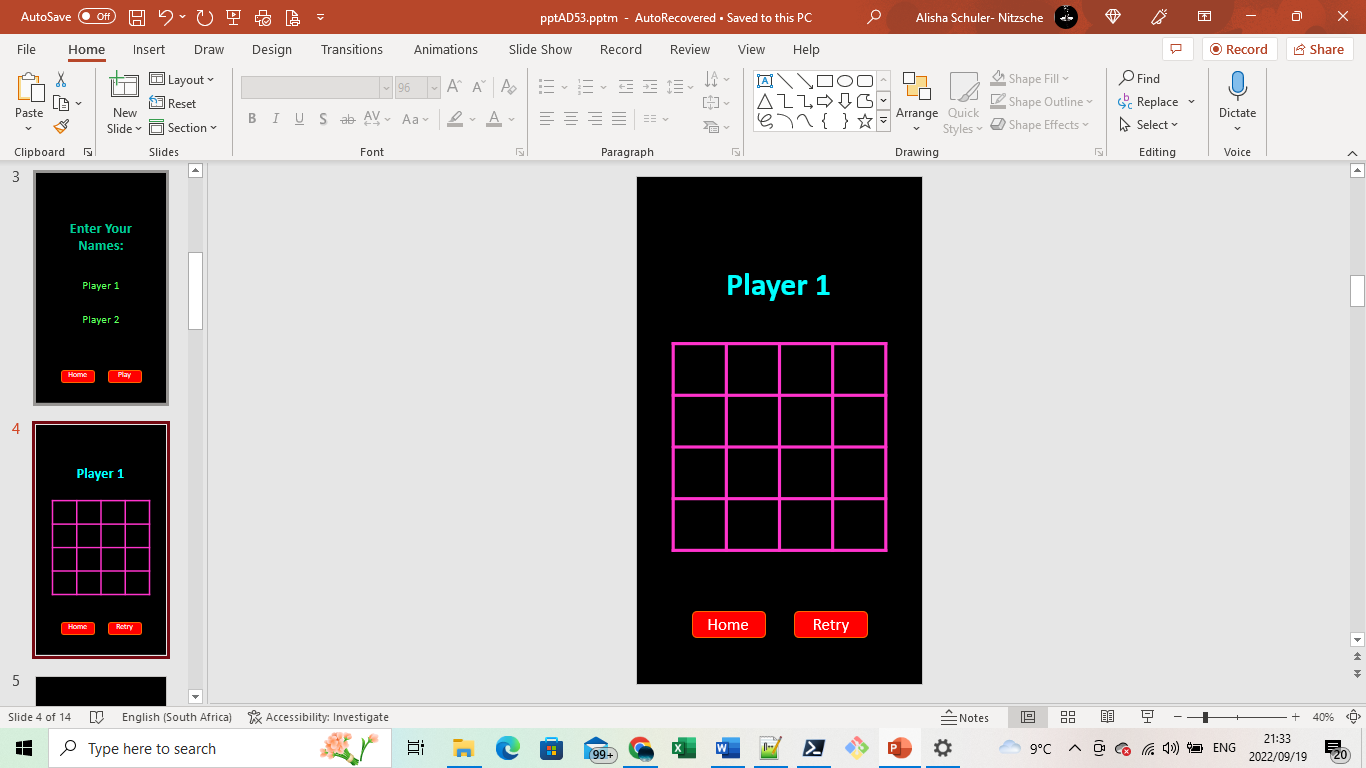
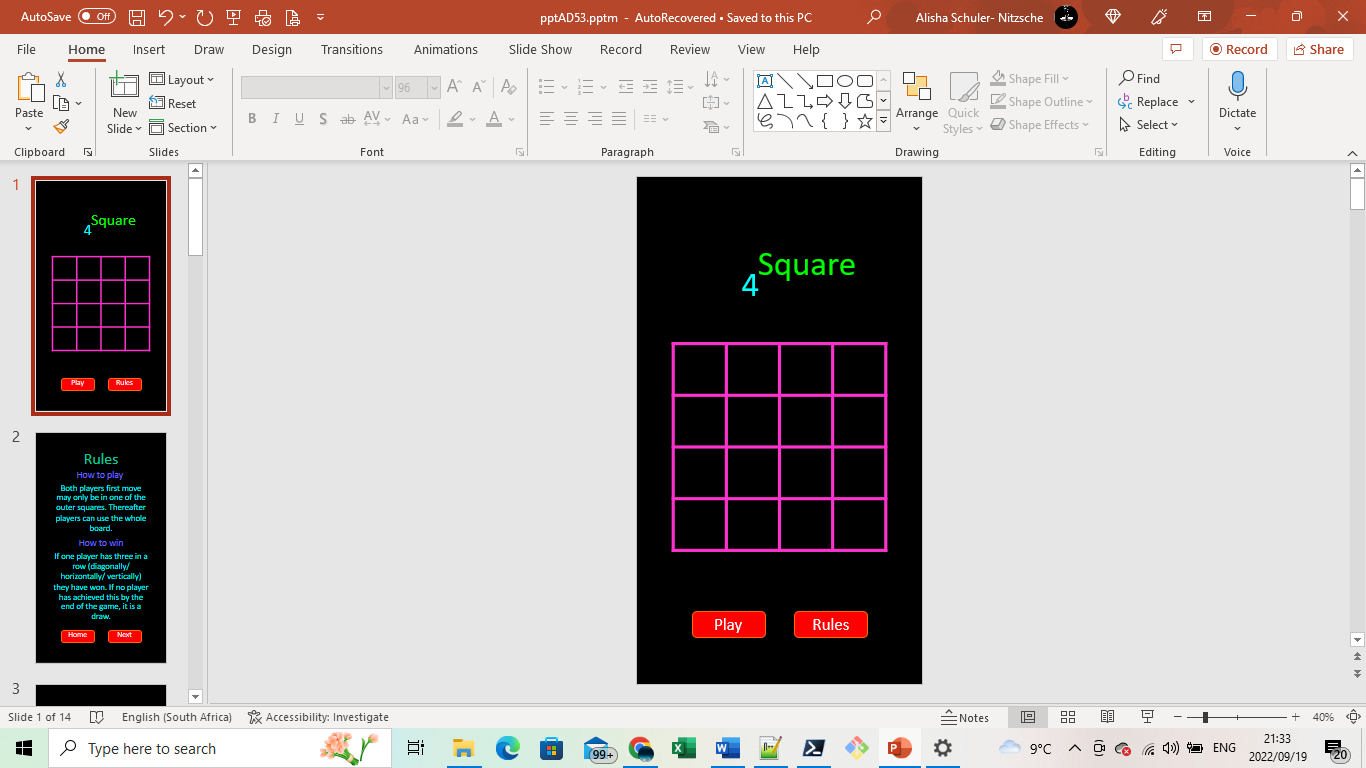
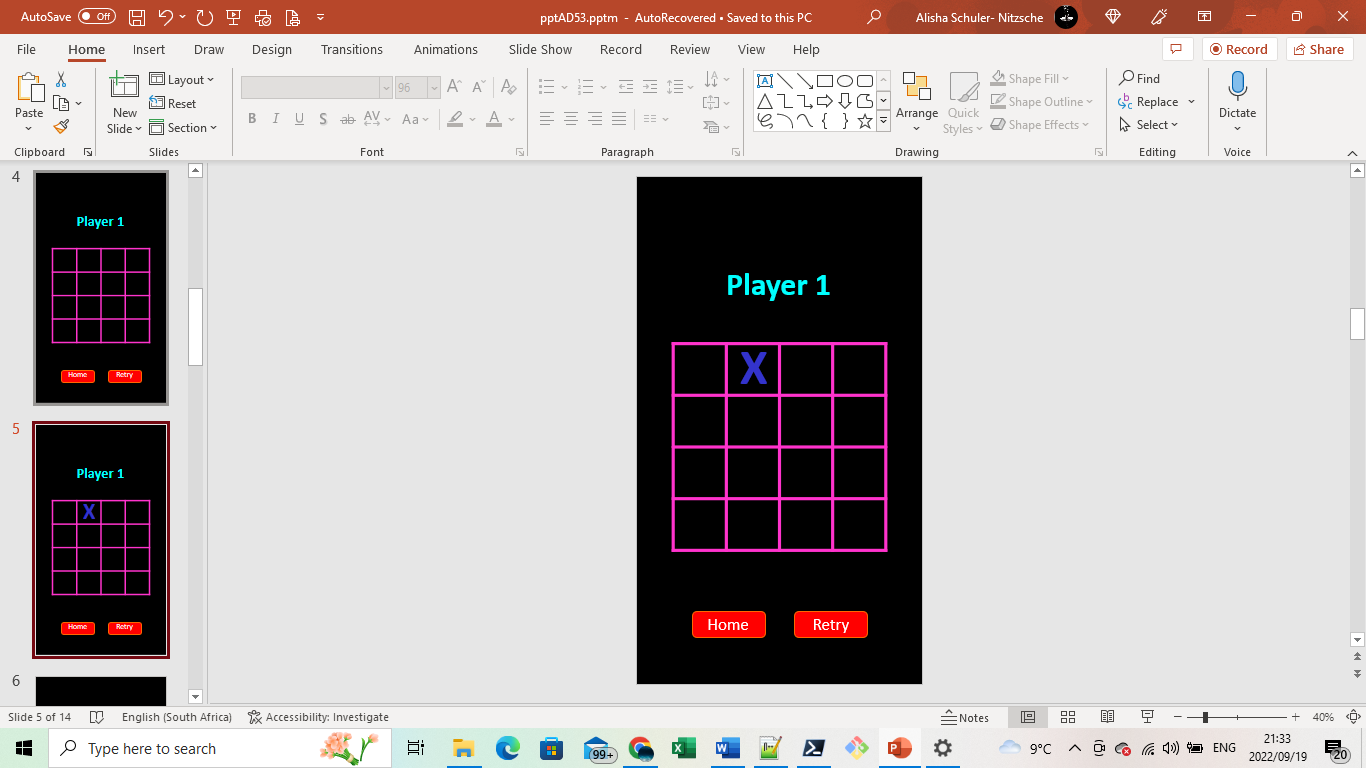
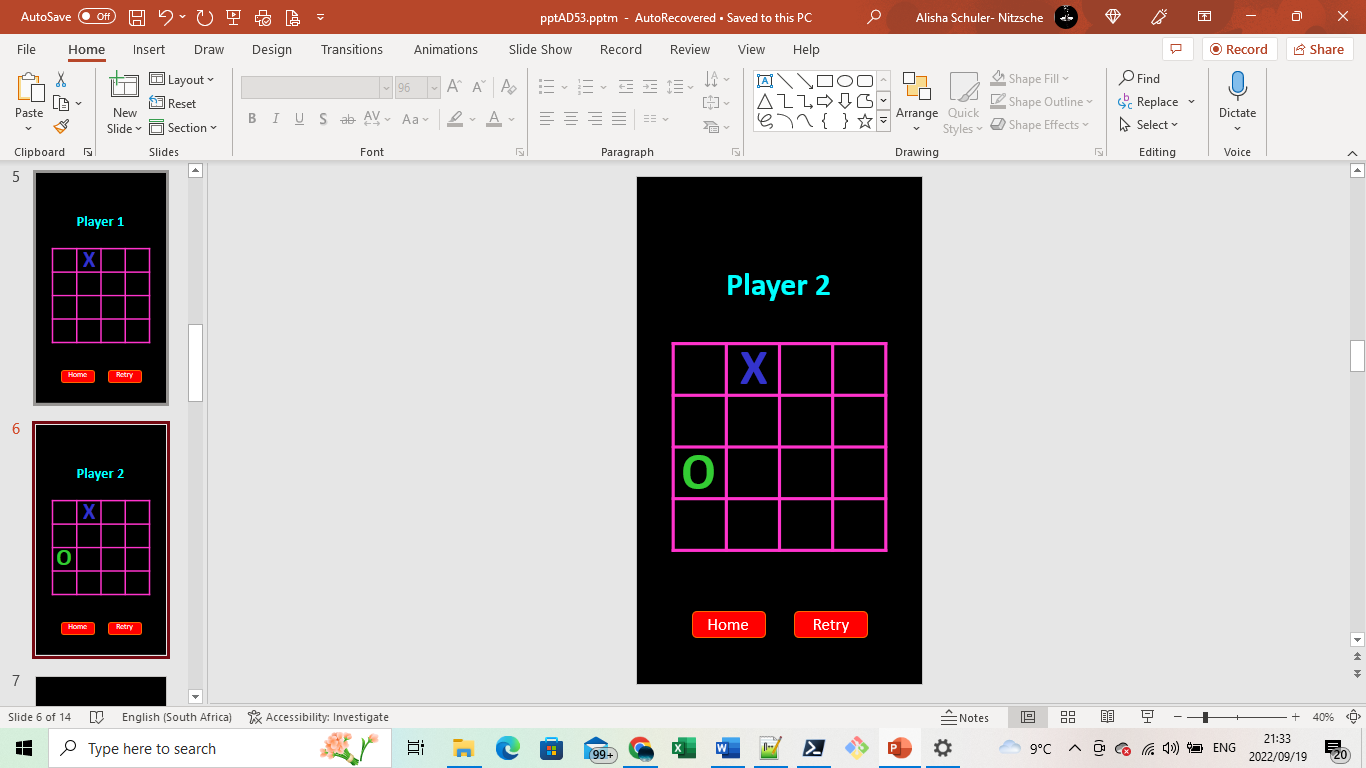
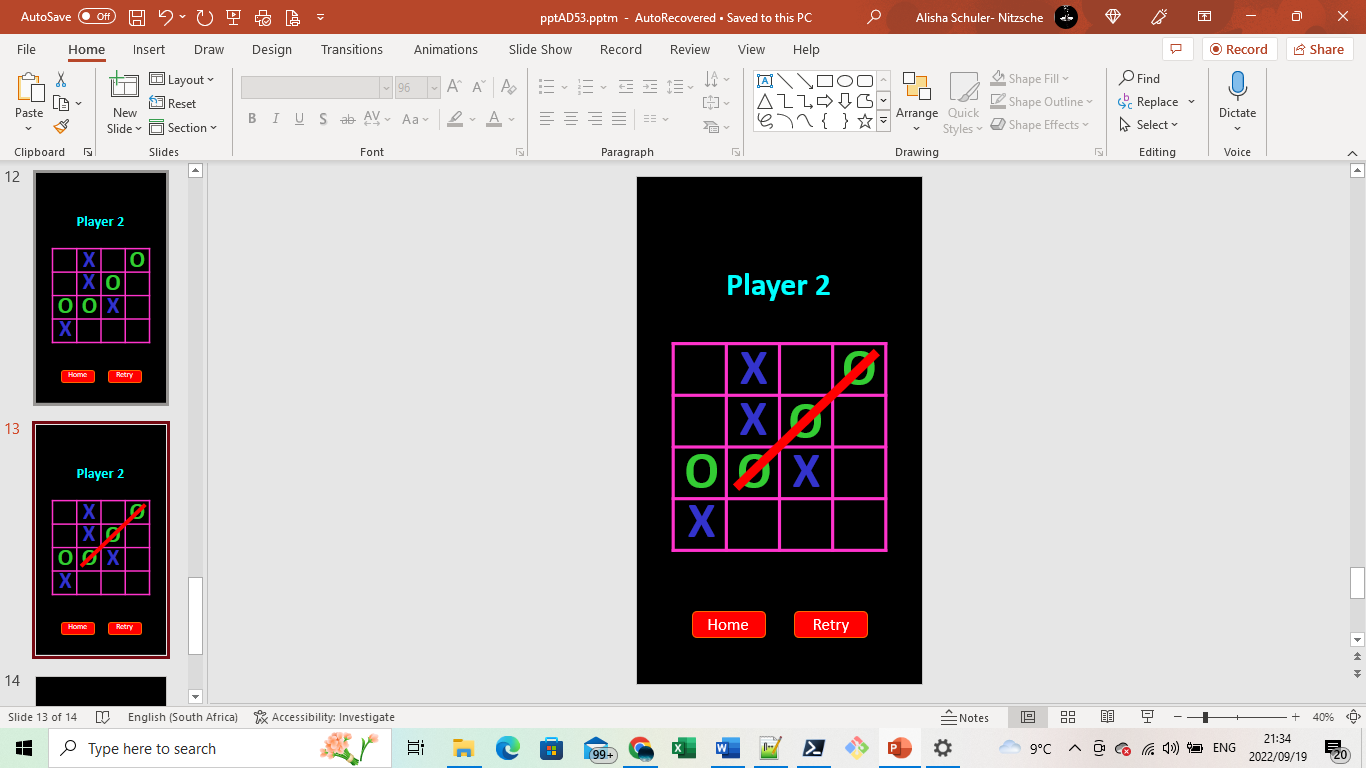
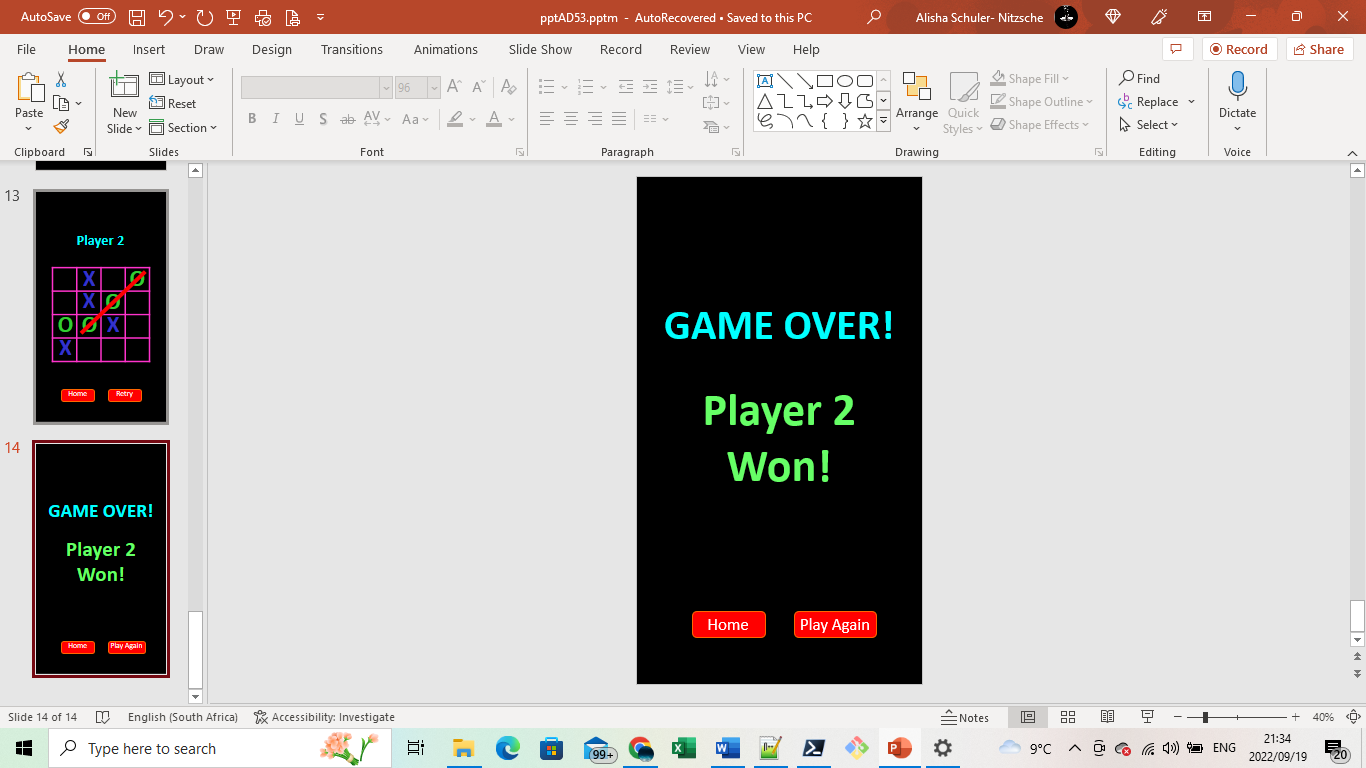
****

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

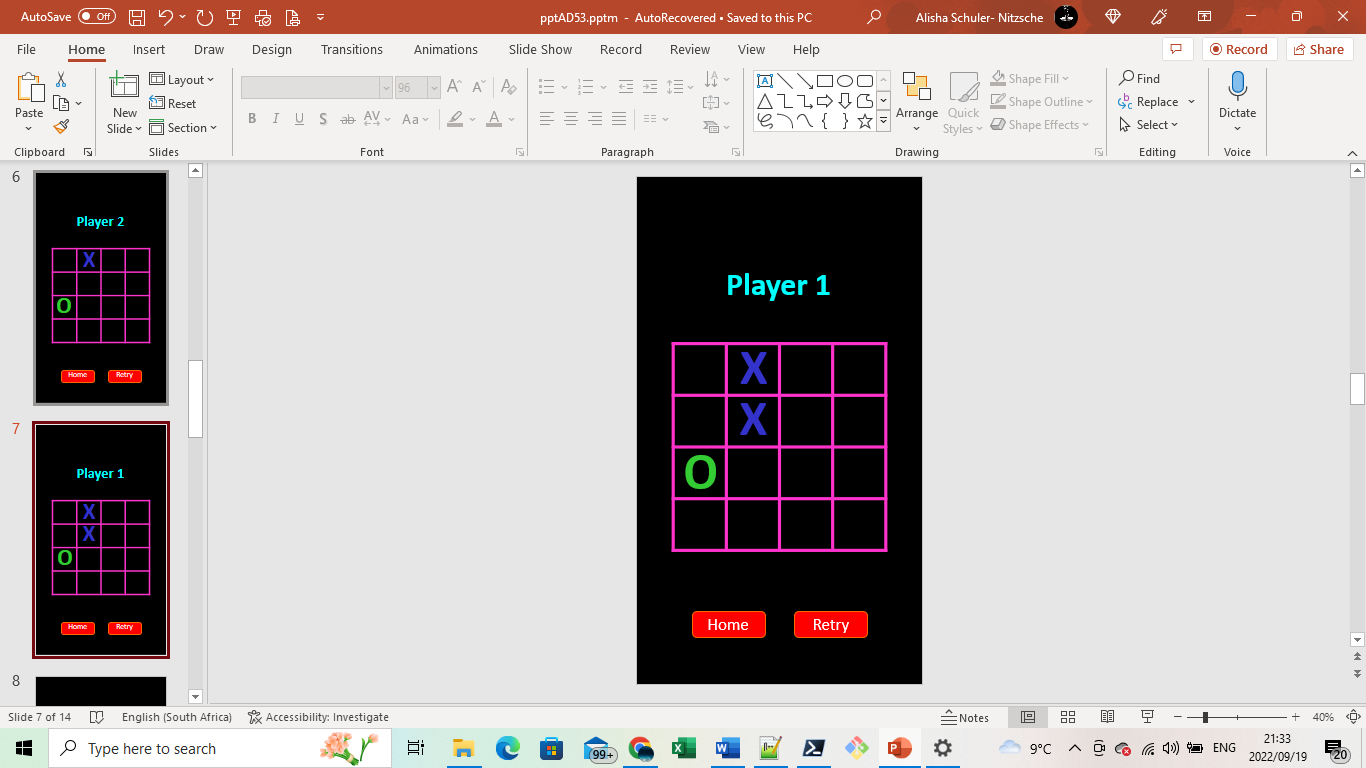
**Shape

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

**Diagram

Description automatically generated**

Figure 3: Object diagram (UML)

**Code summary**

**Visual Studios XML file:**

**Home Activity:**

Clicking on the design tab we will drag in 3 views, one text view and 2 button views.

The text view will be to show our title (4Square) and the buttons will take us either to the rules activity or the enter names activity.

We will constrain all three views to the top of the page and the left and right side of the page, the two buttons will be constrained with each other as well as with the title text. We will then move each view around to suit our predesigned layout.

Then looking at the xml code with out designed home page we will change the text within our views by opening the resource folder, expanding the values folder and opening the strings.xml file. In this file we will add three new strings that we want our views to be.

i.e.

<string name=”homeTitle”>4Square</string>

         <string name=”playButton”>Play</string>

Once this is setup we will go back to our activity main.xml file where we will reference those strings, allowing us to change the text of our views.

i.e

         android:text=”@string/hometitle”

Clicking the text we will edit the color and size of the text in the attributes panel under the common attributes section.

To edit the button we will need to create a new the drawable resource file in the drawable resource folder. In the new buttons xml file we will introduce a new item, Shape. This shape will be attributed and defined as a rectangle. Within the shape tag we will define the background color of the button and round the edges of the button.

i.e.

<item>

                     <shape android:shape=”rectangle”>

                                 <solid android:color = “#ff0000”/>

                                 <corners android:radius=”100dp”/>

                     </shape>

         </item>

To apply the new button we will need to change the parent of the theme to “Theme.AppCompat.Light.NoActionBar”  and then are we able to reference the new button in the activities main .xml under the buttons tag..

i.e.

         android:background-“@drawable/button”

Now we will change the background colour to black. We will do this by defining a background to it in the constrain layout tab

Here we will also call the board draw method to add visual stimuli to the opening home screen of the app.

**Name Activity:**

We will create a new activity in our games file where user can input their names or use the pre existing names (i.e. Player 1).

We will add a text view to notify what the user has to do in this activity and we will drag in two Plain Text as well as two buttons. We will then constraint all the views and edit them to suit our design idea the same way we did for our Home activity.

We will carry over our theme from the last home activity to the names activity by copy pasting the background.

**Linking Buttons**:

In the main activity.java file we will create a new public method under the one create method. In this method we will be using the intent class by creating a new intent that calls the players name page.

i.e.

         Intent intent= new Intent(this, Name.class);

         startActivity(intent);

now we need to link this method to our play button. This is done by clicking the play button and in its attributes panel selects our public method that we made for the button on the on click dropdown.

These steps will be repeated until all five activities have been made and changed to suit our design idea and they all have functional buttons.

**Visual Studios Java files:**

|  |  |
| --- | --- |
| **fourSquare Class** |  |
| **Objects ->** | boardcolour(int)  Xcolour(int)  Ocolour(int)  Winlinecolour(int)  Winline (Boolean)  Painit (new Paint())  Game  cellsize(int) |
| **Method** |  |
| fourSquare(Context contex, @Nullable AttributeSet  attrs): main | Gets empty board from other class  Has exception clause  Calls empty board from other class( game = new GameLogic(); ) |
| @Override  onDraw(Canvas canvas):void | We will make a method that calls the draw board function and the draw X or O.  In this method, we will also set the style of drawing  i.e.           paint.setStyle(Paint.style.STROKE); |
| @Override  onTouch(MotionEvent event):boolean | Takes one parameter in, the spot that players clicked on screen (MotionEvent event)  It get the x and y value of the position the user clicked and checks if user clicked on the game board.  It will find the cell that the user tapped on  i.e.           int row = (int) Math.ceil(y/cellSize); //find the cell of the tap           int col = (int) Math.ceil(x/cellSize);  Will be in charge of printing the updated board and alternating between the players.  i.e.           if (game.getPlayer%2==0)//determining if its even or odd number{                       //if even it was player 2 turn therefore change to player 1 now                                                                         game.setPlayer(game.getPlayer()-1);           }else{                                                                   game.setPlayer(game.getPlayer()+1);//changes it to player 2           }  All this will be in a an if statement that quecks weather someone has won yet |
| @Override  measure(int width, int height): void | before we can start with the draw board method we need to see how large the screen is so that we can allocate a cell size for the grid so that everything stays on the screen.  This method will take in two parameters, width and height.  To get dimensions of the screen and find the smallest length we will use the math.min value. To convert this to our cell size we will divide it by 4.  i.e.           cellsize =( Math.min(getMeasureWidth(),  getMeasureHeight()))/4; |
| draw\_board(Vanvas canvas): void | in this method we will have pull the colour we set for the board from the xml file and will tell the stroke width.  In this method there will be a for loop that will draw the column lines and the row lines of boar  i.e.           canvas.drawLine(cellSize\*c, 0,  cellSise\*c,canvas.getWidth(), paint);  //draws columns |
| draw\_x(Canvas canvas, int row, int col): void | will take in three parameters the Canvas,  row, column that we want to draw in.  the x will be drawn in two lines by taking the starting x and y value and the ending x and y values and paint.  i.e.  canvas.drawLine((col+1)\*cellsize - cellSize\*0.2                                   row\*cellSize + cellSize\*0.2,                                   col\*cellSize + cellSize\*0.2,                                   (row+1)\*cellSize - cellSize\*0.2,                                   paint);  //the + or – at the end of each x or value is so that the symbol wont get drawn on the grid itself but rather in the grid |
| draw\_o(Canvas canvas, int row, int col): void | canvas.drawOval(col\*cellSize + cellSize\*0.2,                              row\*cellSize + cellSize\*0.2,                                 (col\*cellSize+cellSize) - cellSize\*0.2,                                 (row\*cellSize+cellSize) - cellSize\*0.2,                               paint); |
| Draw\_symbole(Canvas canvas):void | Will go through board and see if we need to draw x or o.  With a nested for loop and an if statements we can check if that position the user clicked and has been taken or if it is still 0, if it is 0 then we can fill it with the correct users symbol by checking which users turn it is and then calling their symbols draw function. |
| gamesetup(TextView playerdisplay, String[] name):void | Will set all the values the game logic class will need to set certain hidden and seen views. It will allows gam logic class to see whos turn it is and allow it to change attributes of the text view. |
| resetgame(): void | Calls method from game logic class that will reset the game |

Here after we will make attributes customizable by adding:

xmlns:custom=<http://schemas.android.com/apk/res-auto>

then we will rebuild the project allowing us to drag our board our of our palate and adding it to our activity where we can now add its constraints and place it where we want it.

Then by making a new java class we can start coding the game logistics, class name: game logic.

|  |  |
| --- | --- |
| **gamelogic class** |  |
| **Objects ->** | gameboard (int)  player (int)  playernames (String[])  playersturn(TextView) |
| **Method** |  |
| Gamelogic(): void | Will initialize the private gameboards 2d array that will hold the values for the board.  i.e.         gameboard = new int [4][4];  it will also have a nested for loop allowing us to fill each spot n the array with a 0, thus telling the game this slot is available. |
| updateboard((int row, int col): boolean | Takes in two parameters, row and column of the click.  Here it will check if position in the array has been taken or not. If it hasn’t then it will replace the 0 with the players number, Player X will be 1 and player O will be 2.  i.e.                       if(gameBoard[row-1][col-1]==0                       {                           gameBoard[row-1][col-1]=player;                             return true;                             }  Will also print whose turn it is at the top of the screen. |
| win\_check(): boolean | Through different loops going through the gameboard array, it will check if someone has won or if all the spaces have been taken and it is a draw. |
| restartgame():void | Refilled the board game array with 0 so that the game can start again. |
| Getgameboard(): int[][] | Getter method to retrieve the game board so we can fill it in |
| setPlayer(int player): void | Takes one parameter, the players number (int) and sets the players number. |
| getPlayer(): int | Getter method to retrieve player number |
| setplayerturn(TextView playerturn): void | Sets the players turn |
| setplayernames(String[] playernames):void | Sets the players names |

|  |  |
| --- | --- |
| **GameDisplay class** |  |
| **Objects ->** | Foursquare(foursquare) |
| **Methods** |  |
| @override  onCreate(Bundle savedInstanceState): void |  |
| retryButtonClick(View view):void | Calls the reset game method to update the old array and updates the visuals of the board. |
| homeButtonClick(View view):void | It will take the user back to the home page every time |