CS345 Lab 4 – Xamarin Chess Application

The purpose of this lab is to allow you to familiarize yourself with some of the features of the Xamarin development tools.

The application is a designed to allow two people to play a game of chess without a chess set. The basic application will look like this:



Players take turns moving the chess pieces. To move a piece, the player touches the piece they want to move, then touches the destination square. The color or the squares should change to indicate that the square has been selected. The software will not enforce any rules. Just as with an actual chessboard, players can move pieces anywhere they like. After a players make their moves they touch the large button that is closest to them to stop their clock (which serves as a count-down timer starting at 5 minutes), and start their opponents clock.

Recommended Approach

You are welcome to implement this application differently than described here, as long as you use only your own code. You may take as much or as little as you like from the recommended approach. The recommended approach comes with a recommended order of completion as described in the following steps.

**Implementing the board and the clocks**

Use XAML to define a Grid layout. The outermost layout will have 3 rows – one for each of the clock/buttons, and one for the board. The board should occupy 3 times as much screen height as the clock/buttons. The board itself can be implemented using another Grid layout. That layout should be 8x8 with all rows and columns having the same width and height. You should name the Grid containing the board so that you can gain access to the grid using

Grid myBoardGrid = this.FindByName<Grid>(“nameOfBoardGrid”).

You can use the Image class to render each of the chessboard squares and to display pieces on those squares. It is easier to create these images in C#, rather than using XAMAL.

To get a single image:

Image newImage = new Image();

newImage.Margin = 0;

boardGrid.Children.Add(newImage, column, row);

To create an image with a particular image use something like:

Image pieceImage = new Image { Source = “white\_rook.png”};

Attach a TapGestureRecognizer to the image so you can handle the event when a player selects the square.

Create an instance of TapGestureRecognizer – probably as a data member of MainPage.

TapGestureRecgonizer recognizer = new TapGestureRecognizer();

Add an event handler to recognizer

recognizer.Tapped += Recognizer\_Tapped;

Attach the recognizer to your images

myImage.GestureRecognizers.Add(recognizer);

If your images are in a grid, you can recover their rows an columns using properties like this:

Private void Recognizer\_Tapped(object sender, EventArgs e)

{

Image image = (Image) sender;

int row = (int) image.GetValue(Grid.RowProperty);

int column = (int) image.GetValue(Grid.ColumnProperty);

}

The two clocks can be implemented using instances of the Stopwatch class. You will need to add using System.Diagnostics. Use Device.StartTimer() to cause the time displayed on the button to be updated every second.