**Design:**

In designing my solution for this assignment, I focused on ensuring the game was easy to play and had some nice features (replay with option to jump in, randomiser for board and disc colours etc.). It was also important to me that the game be stable so I spent much time testing various bugs i.e. tapping the board like crazy at particular points in time like when the replay animation started. It is, as such, a fairly stable game.

I kept the simplistic feel shown in the assignment brief as I felt the board should be the centre of attention with no fluff around to distract. Below are some screenshots of the game at different points that will help me describe my design process.

A screenshot of a cell phone

Description automatically generated

Figure 2 shows the game in play

A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generated

Figure 3 shows a completed game

Figure 1 shows the opening

A screenshot of a cell phone

Description automatically generatedA picture containing computer

Description automatically generatedA screenshot of a cell phone

Description automatically generatedscreen of Connect4

Figure 6 shows the restart prompt in a game with randomised colours

Figure 5 shows the Replay screen

Figure 4 shows the History tab where previous games can be selected

**Board:**

In drawing the board I wanted to use all available space so I measure the width of the screen, subtracted a margin of 5 either side, then divided the remaining space by 7 to get the disc size. This would ensure a square board with evenly spaced slots.

A screenshot of a cell phone

Description automatically generatedI initially, added the discs to the board using the UISnapBehavior method, but this made emptying the board problematic. To remedy this, I added barriers for the frame of the board (bottom and between slots). Then as each disc is dropped, I add a barrier above the previous disc to make sure all disc line up correctly. This makes the animation of the discs dropping out the bottom of the board simple:

A screenshot of a cell phone

Description automatically generated

To make the board more interesting, I added a button (Shuffle button on top right) to allow the user to shuffle the board and disc colours to random colours. To make my life easier, I implemented a UIColor extension that returns a colour from three randomly generated values for RGB. I added a highlight pieces feature and would have probably changed it to add a glow around the four winning discs but did not have time.

**GameVC:**

The main purpose of GameVC is to control the Board View and give the game the correct dynamics. It uses the Alpha0Connect4 framework, in particular the GameSession class, to model the actual game play. This VC has gesture recognisers for both tapping on the board when dropping a disc as well as when the board is double tapped during a Action Replay. The touchesBegan function deals with selection of columns for discsA screenshot of a cell phone

Description automatically generated

as well as the tap to show the order of the game (sequence of moves).

When conditions are correct, i.e. it’s the users turn and the board is finished animating, then this function calls . This calls the corresponding function to drop a disc for either a user or the bot.

A picture containing drawing, food

Description automatically generated

During animation of the Action Replay, the function sets up the labels and repeatedly calls the function every 2 seconds via the Timer class. When the user wants to jump back in, they can double tap the screen to invalidate this timer and stop play. The tap must happen during the users turn, not the bot

**CoreDataTVC:**

This class inherits UITableViewController and NSFetchedResultsControllerDelegate and updates the tableview of GameHistoryVC using the NSFetchedResults.

**GameHistoryVC:**

This class deals with the History tab, displaying details of previously completed games. It inherits CoreDataTVC so that historic games can be added from the CoreData model. This class uses CoreDataTVC to set the labels in Figure 4. A segue from this View brings the user back to the game screen and starts replaying the selected game. I made use of the String method with Regular Expressions so I could save the moves in the GameSequence class by String.

This class also has the **GameHistoryTVCCell** class which has the cell labels as attributes.

**DiscBehaviour:**

This class controls the UIDynamicBehavior of the board and disc. Here I add gravity, collision and item behaviours so the discs can appear as they’re dropping due to gravity. I also add and remove barriers using this class that make the board act as if it has structure.

**GameSequence:**

A picture containing table

Description automatically generatedScreen of a cell phone

Description automatically generatedThis NSManagedObject class deals with the CoreData model entity GameSequence allowing the game to be saved and displayed in the GameHistoryVC. The variables contained in this class are used to set the gameID, winner and sequence of moves in the History tab.

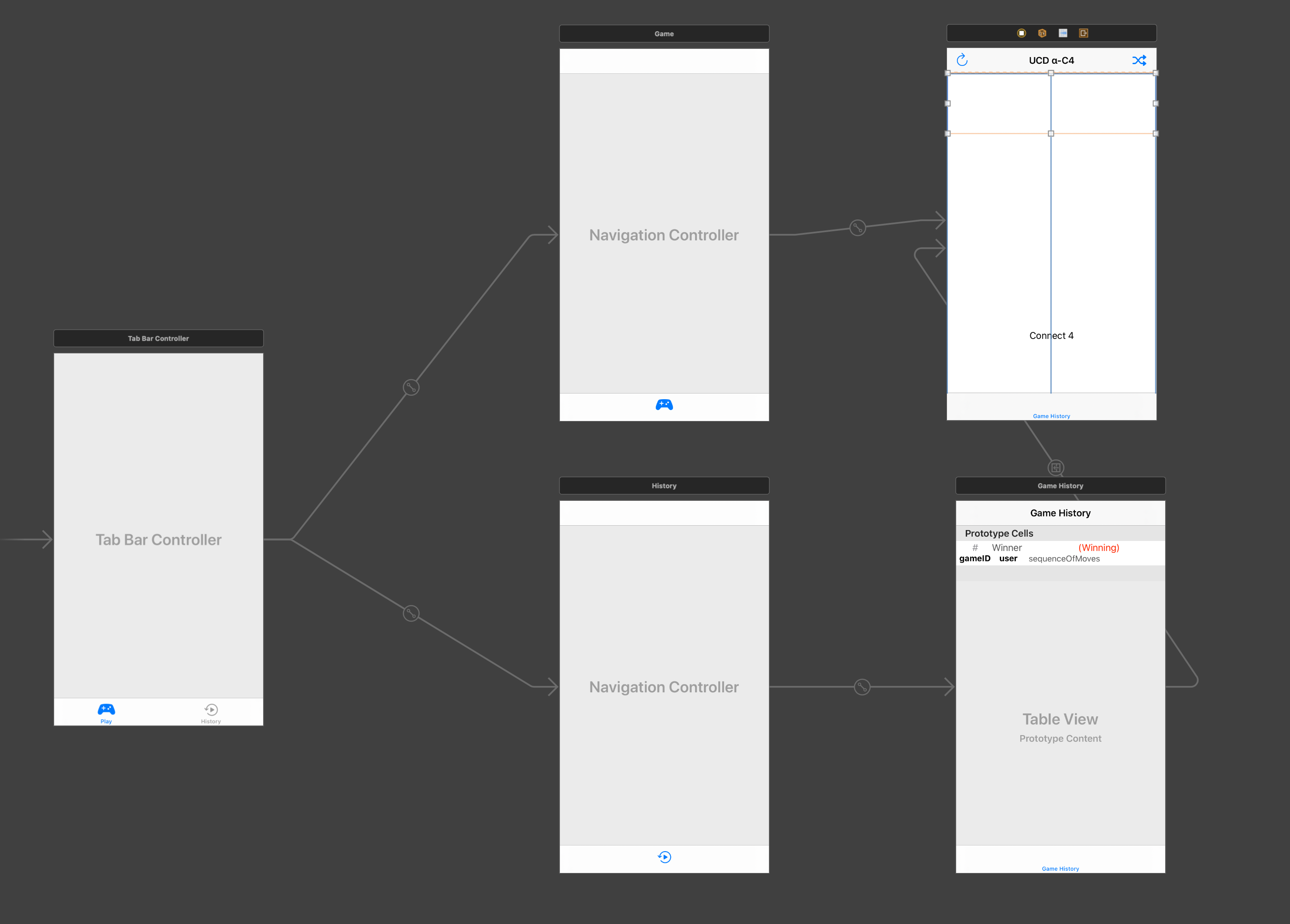


Figure 7 Storyboard