Tic Tac Toe Features:

Tic tac toe is one of the other two games we decided to implement. There are game modes to choose from in Tic Tac toe; player vs player, player vs random AI, and player vs a somewhat strategic AI. In this game, the user always plays first, and marks their circle with a red marker, and player 2 with a blue marker. There is a countDown timer that allows for us to keep score in the leaderboard. The score is calculated and kept by the shortest time it takes to beat the AI. There is no score kept when playing player vs player mode. The timer starts at 100 seconds, and whoever wins first, ties or if the timer runs out decides how the game ends.

- What is your unit test coverage?

  Our unit tests cover any controller code. This includes these classes from Sequencer:

  Sliding Tiles:

  - BoardandTile
  - BoardManager
  - Board
  - MoveStack
  - Tile

  TicTacToe:

  - TicTacBoard
  - TicTacBoardManager
  - TicTacMinimaxStrategy
  - TicTacMarker
  - TIcTacRandomStrategy

  ScoreBoard:

  - LeaderBoardController
  - LeaderBoardReader
  - Scores
  - UserScores

  This total code coverage excluding the model and view classes is :

  The classes excluded from testing are all model and view classes:

  These include classes from :

Sequencer:

Sliding Tiles:

Check this

Board

- CustomAdapter
- GameActivity
- GestureDetectGridView
- MoveStack
- SettingsActivity
- SlidingMovementController
- StartingActivity

Tile

-GameActivity

TicTacToe:

- TicTacCustomAdapter
- EmptyStrategy
- TicTacGameActivity
- TicTacGridView
- TicTacStrategy and TicTacEmptyStrategy
- Timer

ScoreBoard:

- LeaderBoardCustomListAdapter
- LeaderBoardFrontEnd
- swipeTest
- swipeViewAdapter
- DemoFragment

LeaderBoardFrontEnd is responsible for reading and writing to firebase,

LeaderBoardCustomListAdpater takes in a list and and uses row.xml to generate the view that user see's of a scrollable list of highscores.

swipeTest and swipeViewAdapter and DemoFragment create the tab view of highscores that user see when they press Leaderboard on the page navigation.

- What are the most important classes in your program?

The most important classes in our program are leaderboardfrontend, leaderboard controller, GameActivityView

- What design patterns did you use? What problems do each of them solve?

For the leaderboard we reduced the number of classes, and the navigation for the user. We did this by using a tabview which allowed the user to swipe through the various highscores. Additionally, we implemented a factory design pattern initially for the various leaderboards, but realized that it could be simplified to one class which is responsible for reading and writing to our database. And one class for sorting and adding new highscores to a local list.

- How did you design your scoreboard? Where are high scores stored? How do they get displayed?

In each game activity we create an instance of a controller, in the controller we create an instance of LeaderFrontEnd, this class has a method called saveScoreToLeaderBoard. This method is called whenever data is changed in the firebase database using an event listener. Once called it saves the firebase data to leaderBoardController which adds the new high score using certain game logic conditions. This process is used for all games. The highscores are stored in firebase in a branch called Games. We display the data using a tab view. This works by first extracting the highscores of each from firebase, and storing them in leaderBoardReader, from there using fragments we display certain scores on the corresponding page.

The scores are displayed from lowest to highest to signify rank, and since rank is always lowest to highest we followed this structure for all our games.