Technical Lab Senior Mobile Developer

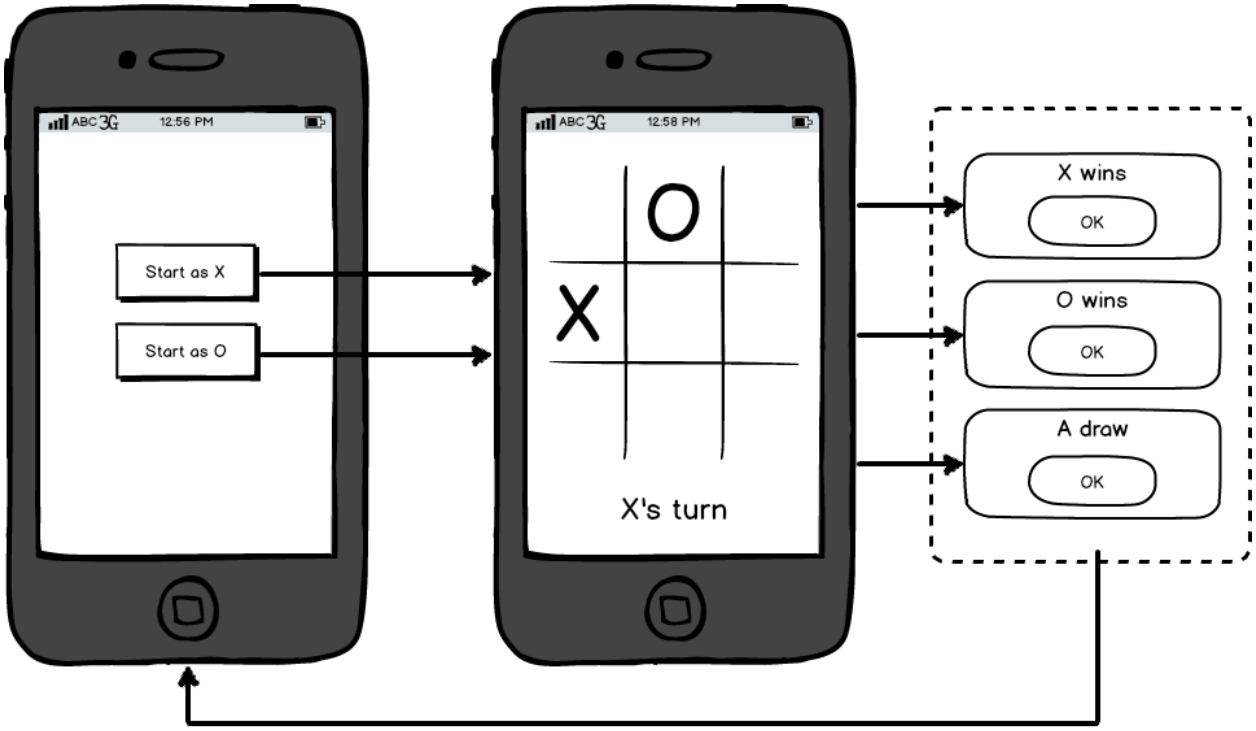
The tech lab is a collaborative coding exercise meant to demonstrate the types of development tasks that a senior mobile developer should be capable of handling, and the candidate's ability to handle them.

You will be developing a simple mobile application targeting your preferred mobile platform. You may use the vendor's preferred native development environment (Xcode, Eclipse / Android Studio, Visual Studio), Xamarin Studio, or the Xamarin plugin for Visual Studio.

Please refrain from writing any code until the lab begins.

Task Overview

The objective is to implement a simple tic tac toe* game using the native UI controls. The game should have a menu screen with two buttons: "Start as X" and "Start as O". Tapping one of these buttons will present the game screen, and give either X or O the first move based on which start button was pressed. The game should respond to user taps on the game board, placing X's and O's alternately until a win or draw occurs. The game will then present a simple alert with the message "X wins", "O wins", or "Draw". Dismissing the alert should take the user back to the main menu. Both X and O will be played by humans, AI or computer players are out of scope.



* Tic tac toe is a game played on a 3x3 grid. Two players, X and O, take turns placing their mark on an unmarked grid square of their choosing. The first player to place three of their marks in a row, horizontally, vertically, or diagonally wins the game. A draw, in which no player wins, is common.

Instructions

1. Ensure you have your mobile development environment installed and configured.

2. Download the image assets for the X and O marks from https://www.dropbox.com/sh/gau8ly51aw2yjtd/FMo57qHa_T
3. Join the Google Hangout and share your screen.
4. Create a new project file and import the image assets.
5. Implement the menu screen with the two start buttons.
6. Implement the game screen, ensuring that it responds to touches on the game board.
7. Implement the win or draw condition checking and UI alerts.
8. Optional: Implement one or more of the bonus features.
9. Push the finished source code to the provided git repository.

Requirements

- Use the vendor's preferred development environment and language (Java, Objective-C/Swift, C#) or Xamarin. Do not use javascript or HTML.
- The app must compile with the most modern SDK and run on modern devices, backwards compatibility is not required.
- The app must display properly in portrait orientation. Landscape orientation must either be disabled or function properly as well.
- Load and utilize the X and O images provided.
- Correct functionality is more important than aesthetics.
- The cleaner and more readable your code is, the better.

Bonus

Animation / Transitions

Add some fade in/out or other animations to the gameplay or view transitions. Use your creativity.

Social sharing

Add an extra button to the win alert. Tapping this button will allow the user to boast about their mild accomplishment on social media networks. Use the OS provided sharing mechanism to do this (i.e. intents on Android, action provider on Windows Phone, activity controller on iOS)

Persist game state

If you background the app during gameplay, then force quit the application, it will forget its gameplay state. Persist and resume the game state during app and/or view lifecycle events to allow the gameplay to resume properly instead.

Integrate Ads

Add a banner ad to one of the screens using:

- iOS: iAds, using the framework included in the base iOS SDK.
- Windows Phone: Microsoft Advertising SDK available here: http://www.microsoft.com/en-us/download/details.aspx?id=8729
- Android: Google Mobile Ads (aka AdMob), which is part of Google Play Services

No ad IDs are provided, so no actual ads will appear. That is ok for this exercise. Enable a test mode if available.