How to Use this Template

1. Create a new document, and copy and paste the text from this template into your new document [Select All → Copy → Paste into new document]
2. Name your document file: “**Capstone_Stage1**”
3. Replace the text in green

GitHub Username: sobelman666

25 Squares

Description

25 Squares is a game in which the objective is to touch all the squares in a 5x5 grid. You may start in any square. After that you may only move to a square that is either 1) 3 squares away horizontally or vertically, or 2) 2 squares away diagonally. You may not touch the same square twice. If you can visit all 25 squares, you win!

Compete with others by signing in and posting your score to the leaderboards. Challenge your friends to beat your best score!

Intended User

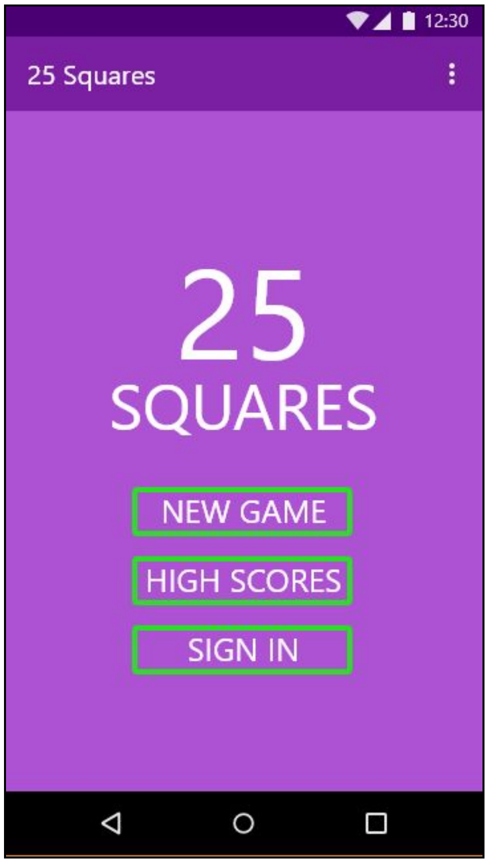
25 Squares is for anyone of any age who enjoys a good logic game.

Features

- 5x5 grid for game play
- High scores saved locally
- Optional sign-in for posting scores to leaderboards
- Sharing scores with friends
- Widget for starting a new game or resuming a game that is in progress

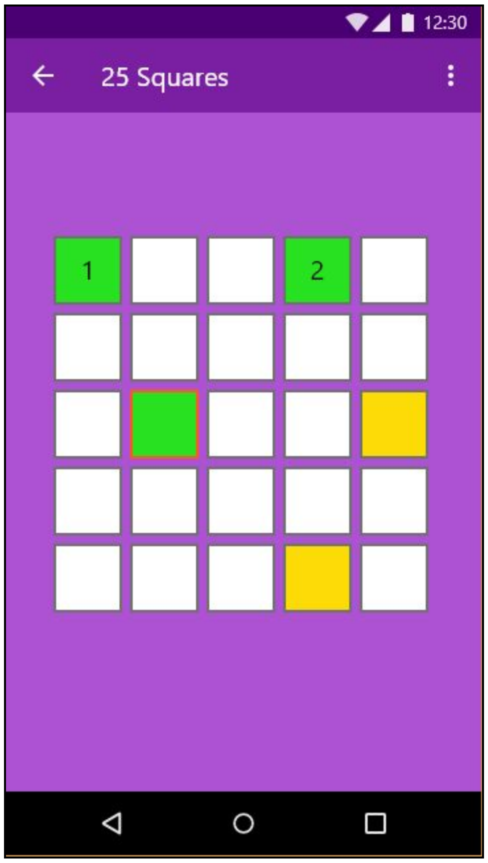
User Interface Mocks

Screen 1



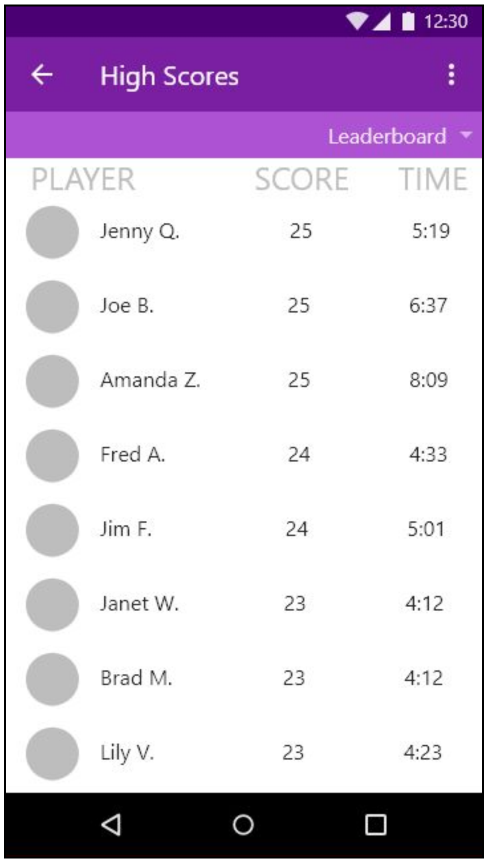
Main app screen featuring app title [will be replaced by a logo, if there is time], and buttons for starting a new game, viewing the list of high scores, and signing in. A button for resuming an in-progress game will be shown if there is a game in progress. The “Sign In” button will not be shown if the user is signed in. Overflow menu contains items for displaying instructions and, if applicable, signing out.

Screen 2



Game play screen, featuring the game board. Last move is highlighted with a colored border, previous moves are numbered in the order that they were clicked. Potential next moves are highlighted in yellow. Overflow menu contains items for displaying instructions and, if applicable, signing out.

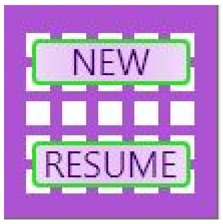
Screen 3



PLAYER	SCORE	TIME
Jenny Q.	25	5:19
Joe B.	25	6:37
Amanda Z.	25	8:09
Fred A.	24	4:33
Jim F.	24	5:01
Janet W.	23	4:12
Brad M.	23	4:12
Lily V.	23	4:23

High score list. If the user has signed in, the spinner is shown allowing the user to switch between viewing local high scores and the network leaderboard. Overflow menu contains items for displaying instructions and, if applicable, signing out.

Widget



Home screen widget allowing user to start a new game or resume an in-progress game.

Key Considerations

How will your app handle data persistence?

Locally stored data, such as previous game times and scores, will be managed via Room. Remote data, such as leaderboards, will be managed via Firebase Realtime Database.

Describe any edge or corner cases in the UX.

Game timer will automatically pause if a user leaves the game screen. The elapsed time of the current game will be saved in shared preferences until the user either resumes the game or starts a new game. App widget will allow the user to resume an in-progress game or start a new game.

Describe any libraries you'll be using and share your reasoning for including them.

Room will be used for local data persistence. Firebase AuthUI will be used to automatically generate the sign-in workflow. Butterknife will be used for data binding.

Describe how you will implement Google Play Services or other external services.

Firebase Auth will be used to handle authentication for users who choose to sign in. Firebase Realtime Database will be used to store remote data.

Describe other considerations.

App is written solely in the Java programming language. App utilizes stable release versions of all libraries, Gradle, and Android Studio. App includes support for accessibility, including content descriptions and navigation using a D-pad. App keeps all strings in a strings.xml file and enables RTL layout switching on all layouts.

App theme extends AppCompat. App uses an app bar and associated toolbars. App uses standard and simple transitions between activities.

App builds from a clean repository checkout with no additional configuration. App builds and deploys using the installRelease Gradle task. App is equipped with a signing configuration, and the keystore and passwords are included in the repository. Keystore is referred to by a relative path. All app dependencies are managed by Gradle.

Next Steps: Required Tasks

This is the section where you can take the main features of your app (declared above) and break them down into tangible technical tasks that you can complete one at a time until you have a finished app.

Task 1: Project Setup

Set up Firebase using Firebase Assistant or manually by following the instructions at https://firebase.google.com/docs/android/setup.

- Add Firebase to project
- Add library dependencies in app-level build.gradle

Add additional library dependencies in app-level build.gradle.

Configure Firebase Realtime database with preliminary permissions and data structure.

Under Authorization, allow Email and Google authentication.

Task 2: Implement UI for Each Activity and Fragment

- Build UI for MainActivity
- Build UI for GameActivity
- Build UI for HighScoresActivity

Task 3: Implement Game Logic

Implement the logic for game play. Game will be represented internally by an 2-dimensional array of Integers.

- Start timer when the GameActivity starts
- Change background of most recently touched square to green highlight (not applicable for first move)
- Change color of squares that are available moves to yellow
- Label squares for previous moves in numerical order
- If last move was winning move, show dialog with “You Win” message and elapsed time

- If there are no more moves available, show dialog with “Game Over” message, game score, and elapsed time
- Share button on end-of-game dialog launches share intent to share score and time

Task 4: Implement High Scores Database

A high scores database will contain local history of scores and times. The HighScoresActivity will show a list of local high scores when the user is not signed in, or when the user is signed in and has selected local high scores to be displayed.

- Implement Room database with a high scores table
- Create a DAO

Task 5: Implement Sign In Using Firebase AuthUI

Firebase AuthUI will be used to generate a sign in workflow.

- In the MainActivity, when the user clicks the Sign In button, launch the AuthUI workflow
- Create AuthStateListeners in each activity to detect whether user is signed in
- Update UI to reflect sign in state

Task 6: Implement High Scores List

The HighScoresActivity will show a list of local high scores is the user is not signed in, or give the user the option to show either local high scores or the network leaderboard if the user is signed in.

- A ViewModel will hold the high scores data, which will update based on sign-in state and user’s selection
- Data will be loaded into the ViewModel from the Room database or from a Firebase Realtime Database reference as appropriate

Task 7: Implement Instructions Dialog

An item in the overflow menu will allow the user to view the game play instructions.

- Instructions will be downloaded from a remote server using an AsyncTask
- Instructions will be displayed in a Dialog
- If the network is unavailable, a Dialog containing an error message will be displayed

Task 8: Implement Home Screen Widget

The home screen widget will be simple 1x1 widget. It will have a button to start a new game. If there is a game in progress, it will also have a button to resume that game.

- Create widget layout and AppWidgetProvider class
- Clicking one of the buttons in the widget launches the GameActivity directly
- Update widget UI to show Resume button in GameActivity's onPause() method
- Update widget UI to remove Resume button in GameActivity's onResume() method
- If there is a game in progress, show a dialog with an "Are you sure?" message to confirm that user wants to discard their progress

---

Submission Instructions

- After you've completed all the sections, download this document as a PDF [File → Download as PDF]
    - Make sure the PDF is named "**Capstone_Stage1.pdf**"
- Submit the PDF as a zip or in a GitHub project repo using the project submission portal

If using GitHub:

- Create a new GitHub repo for the capstone. Name it "**Capstone Project**"
- Add this document to your repo. Make sure it's named "**Capstone_Stage1.pdf**"