Michael Rooplall
CS389D Mobile Web Development

Sudoku in React Native

Project Writeup

Introduction

My project is a build of Sudoku, a logic-based number-placement puzzle, implemented in React Native. The objective of Sudoku is to fill a 9x9 grid with digits, so that each column, each row, and each 3x3 subgrid (or “block”) contain all the digits from 1 to 9, with no repetitions. My application builds a home screen, level selector, designs and implements the gameplay functionality alongside additional features for user experience. I use a variety of tools including React Native, some open-source Javascript libraries, Expo, and Android Studio.

Development Process

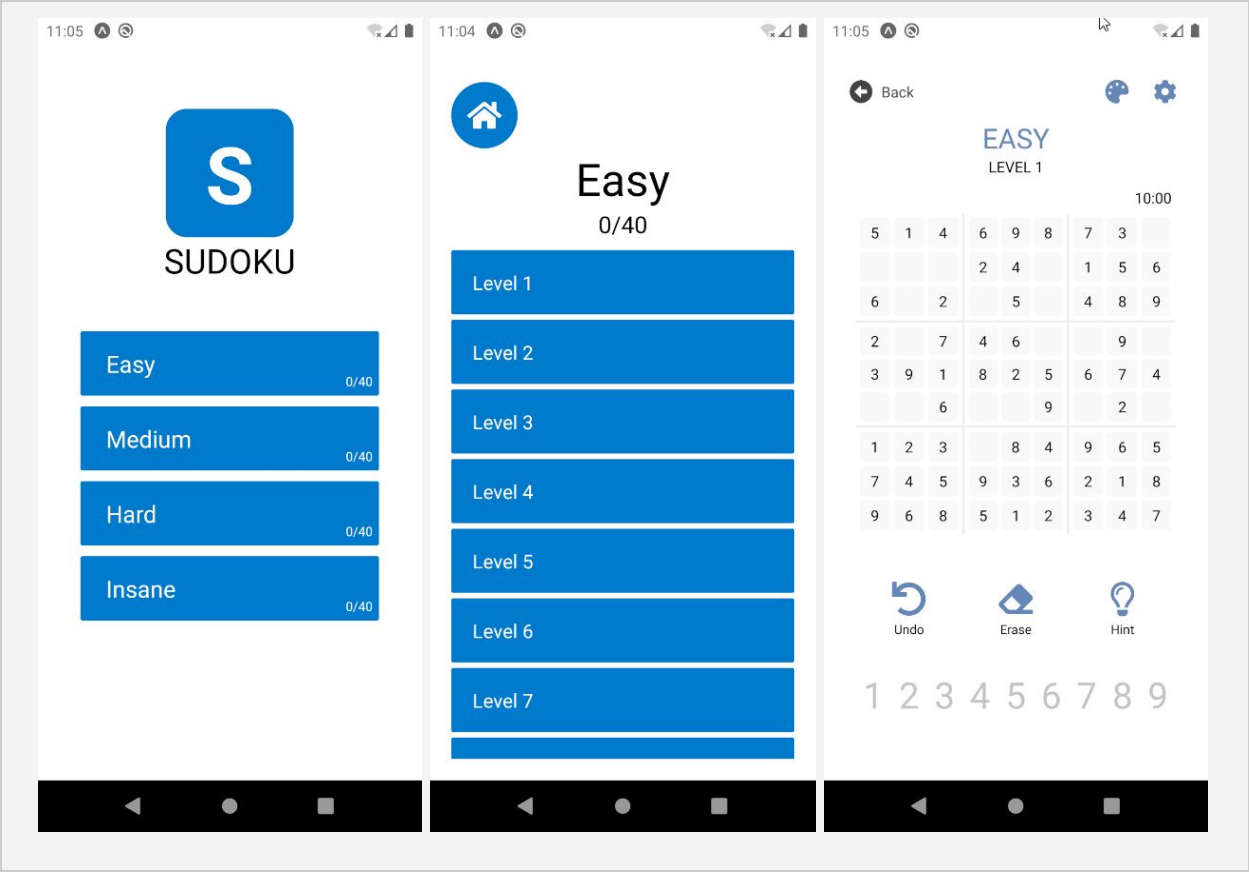
React Native

React Native is a framework for creating native applications for Android and iOS using React. It supports many platforms and has a write-once use-anywhere methodology. Using the vast and robust feature set of React Native and my own experiences with building applications, I was able to build a fully functioning Sudoku game.

Designing the Layout

I first worked on the layout, creating a home screen with a minimalist logo and options for different difficulties (Easy, Medium, Hard, and Insane). The difficulty buttons lead to a levels screen, which implements a ScrollView to display each available level for that difficulty. I precompiled 40 random and unique levels for each difficulty, for a total of 160 available levels. The ScrollView ensures that no matter what the resolution size is or the number of levels that are available, they will always fit on the screen. Selecting a level takes you to the Game screen.

The game screen contained a top bar of buttons, the game grid itself, a row of image buttons, and another row of the number buttons. The game grid view was tough to implement as React Native does not have a simple Grid View like CSS on the web. Each section was split off into 9 subgrid components. Each subgrid then contained 9 “item” cells. Using percentage widths and heights and a combination of flex layouts and aligning, I was able to get a nicely centered grid. The state of the difficulty and level number are remembered from the previous pages. These screens are all stored as individual jsx files in a screens directory, with initial navigation provided by App.js in the root directory.



Level Generation and Implementation

To generate the sudoku levels and ensure they met the requirements for valid unique sudoku levels, I used a project called sudoku.js, an open-source Sudoku puzzle generator and solver JavaScript library. This library provided a single 81 character string (9x9 grid) to represent a sudoku puzzle. These pre-generated levels can be found in a game_levels.json file at the root

directory. I had to implement what I call the indexTracker (found in indexTracker.js), which was responsible for converting the character indices to grid-item indices (eg. top-left grid, second cell) and row-column indices (eg. row 4, column 8) for use in the UI.

Reactive Layouts

The UI is “smart” in that if you select an immutable cell, the numbers bar disables itself. If there are duplicate values in the same row or column, it highlights them in red. The entire board also boldens similar numbers for better readability. The board also includes other various forms of UI features, like slightly highlighting the row and column of the selected cell for aesthetics and better user experience. These fast UI changes are made possible through ReactNative’s ‘useState’ and ‘useEffect’ features, essentially creating hooks on certain variables and firing off events for whenever they change- eg. if the current selected cell is changed (using ‘useState’), the grid cells in the UI are listening for this change and update their appearance accordingly (using ‘useEffects’).

Additional Features

Alongside the core gameplay, there are a few features to improve the user experience like an undo button, erase button, and hint button. All the player's changes are added to a stack, and the undo button simply pops the last event off the stack and reverts the change. The hint button looks for any empty cell, and retrieves that cell's correct value from the solution string-generated from the sudoku.js library mentioned earlier.

Additional Tools

Testing this application was made simple using Expo, a framework and a platform for universal React applications. It is a set of tools and services built around React Native and native platforms that help you develop, build, deploy, and quickly iterate on iOS, Android, and web apps from the same JavaScript/TypeScript codebase. I also used Android Studio to spin up different Android emulator devices, and when paired with expo, allowed me to check the applications scaling and usability on a variety of devices.

Resources

GitHub:
https://github.com/DeveloperBlue/SudokuReactNative

Submission Guidelines:

Project Writeup:

Demo Video:

Code Documentation:

Powerpoint Presentation: