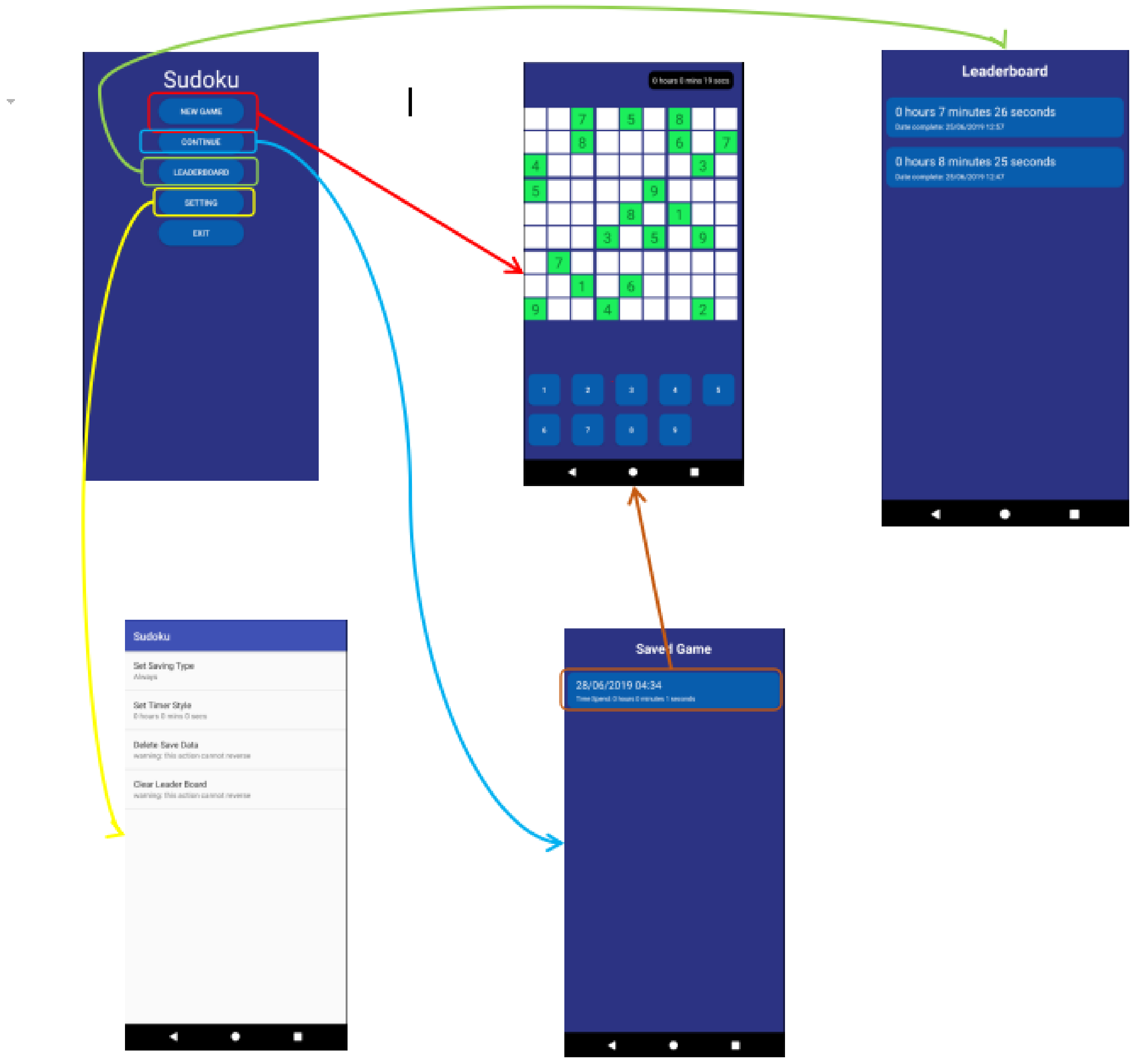
Contents

Purpose

The purpose of sudoku app is to allow user play sudoku on their phone, this sudoku app can random generate sudoku to let user play anytime and save the game whenever they want. When user complete a sudoku their spent time will record in the leader board. The leader board data and the saved game data can be clear through the preference page.

Navigation Graph/ Design



0 hours 7 minutes 26 seconds
Date complete: 25/06/2019 12:57

0 hours 8 minutes 25 seconds
Date complete: 25/06/2019 12:47

Sudoku

Set Saving Type
Always

Set Timer Style
0 hours 0 mins 0 secs

Delete Save Data
warning: this action cannot reverse

Clear Leader Board
warning: this action cannot reverse

28/06/2019 04:34
Time Spend: 0 hours 0 minutes 1 seconds

Documentation

public class AllData

Class contain all static method allow app to interact with database, the database being used in this class is Couchbase lite

Field:

1. **private static Database saveData**
   saved data database

2. **private static Database leaderBoard**
   leader board database

Constructor:

1. **AllData(Context context)**

   Initialize (or get if exist) all database

Static methods:

1. **void saveGame(MutableDocument mutableDocument)**
   Save current game to database

2. **void saveLeaderBoard(MutableDocument mutableDocument)**
   save current spent time to database when user finish the game

3. **ResultSet getAllSavedGame()**
   return all user saved game data

4. **ResultSet getLeaderBoard()**
   return leader board data

5. **Result getSaveByID(String id)**
   return single save data by id

6. **void deleteSaveByID(String id)**
   delete single save data by id

7. **void deleteAllSave(Context context)**
   delete all save data, basically delete the save data database and recreate

8. **void deleteLeaderBoard(Context context)**
   delete all leader board data, basically delete the leader board database and recreate

public class Game extends AppCompatActivity

Class for game activity, which initiate the game layout and other component in the game layout

Field:

1. **private GameBoard gameboard**
   Relative layout which display the sudoku board

2. **private TextView timer**
   Text view which display spent time

3. **private Timer timerS**
   The timer used in game class

4. **private int time**
   Time spent in second

Overridden methods:

1. **protected void onCreate(@Nullable Bundle savedInstanceState)**
   initiate all element in game activity, such as timer textview, game board, and setOnClickListener to relevant element

2. **public void onBackPressed()**
   on back button pressed, will save the game depend on user preference

class NumGroupAdapter extends RecyclerView.Adapter<NumGroupAdapter.NumGroupHolder>

Recycler view adapter for number buttons at the bottom of game layout

Field:

1. **private Context context**
   context from parent activity

2. **private int size**
   size of the sudoku

3. **private OnNumClick onNumClick**
   interface which use to handle number button clicked event

Constructor:

1. **NumGroupAdapter(Context context,int size)**
   Initiate the context and the size

Overridden methods:

1. **public NumGroupHolder onCreateViewHolder (ViewGroup parent, int viewType)**

2. **public void onBindViewHolder(NumGroupHolder holder, int position)**

3. **public int getItemCount()**

Public method:

1. **void setOnNumClick(OnNumClick onNumClick)**
   set the event execute when the number button is clicked

Sub class:

1. **NumGroupHolder extends RecyclerView.ViewHolder**
   view holder class for this recycler view

Sub interface:

1. **OnNumClick**
   interface which handle the click event on the number button

   methods:
   a. void onClick(String s)

public class GameBoard extends RelativeLayout

Main game board which contain 81 number space form a sudoku board, with editable and non-editable space

Field:

1. **private List<List<GameNumSpace>> gameNumSpaces**
   All number space on the board, including editable (which is let user input) and non-editable number space.

2. **private int selectedX**
   selected number space position x value, -1 mean no number space is selected

3. **private int selected**
   selected number space position y value, -1 mean no number space is selected

4. **private int size**
   size of the board

5. **private int time**
   initial time, can be load from save data

6. **private int[] grid**
   grid size of the sudoku board (for example 9*9 sudoku grid size would be 3*3)

7. **private String[][] board**
   current board value, “x” mean no number input

8. **private String[][] initBoard**
   represent how does the board looks like before input any number

9. **private Context context**
   parent context

10. **private String id**
    save data id, empty string mean current game is new game

11. **private TimerAction timerAction**
    use to handle the event each second when timer start

12. **private TimerAction completeAction**
    use to handle the event each second when timer cancel

Constructor

1. **public GameBoard(Context context,String id,TimerAction timerAction)**
   initiate the context, id, and timerAction

2. **public GameBoard(Context context, AttributeSet attrs)**
   default constructor from parent class

3. **public GameBoard(Context context, AttributeSet attrs, int defStyleAttr)**
   default constructor from parent class

Private methods:

1. **private void initSave(Context context)**
   initialize the board and the game with saved game data based on id

2. **private void init(Context context)**
   initialize the board and the game with new game data

Public methods:

1. **public void setCompleteAction (TimerAction completeAction)**
   completeAction setter

2. **public void setNum(String s)**
   set the number to the board based on selected index (x and y)

3. **public boolean validate(String s,int x,int y,Boolean first,Boolean updateElement)**
   validate the board, including row, column and grid

4. **public boolean checkRow(String s,int x,int y,Boolean first,Boolean updateElement)**
   validate the board, only the row

5. **public boolean checkCol(String s,int x,int y,Boolean first,Boolean updateElement)**
   validate the board, only the column

6. **public boolean checkGrid(String s,int x,int y,Boolean first,Boolean updateElement)**
   validate the board, only the grid area

7. **public void saveGame(int time)**
   save the game, if current game is loaded from save data, overwrite the save data

8. **public void saveLeaderBoard(int time)**
   save the result to the leader board when user finish the game

9. **public MutableDictionary getCellDetail(int x, int y)**
   get cell detail based on index and return as Couchbase Mutable Dictionary

Sub initerface

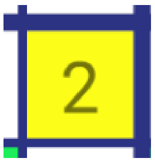
1. **TimerAction**
   Action trigger by timer

   method:
   a. **void action(int i)**

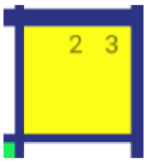
public class GameNumSpace extends ConstraintLayout

Number space in the game board, which allow user input the number, and also allow multiple value input, if multiple value in the same number space, it will show as “pencil mark”

Normal case:



Pencil mark:



Field:

1. **private TextView mainNum**
   textview show when only one number input

2. **private Map<String,TextView> markNums**
   HashMap of all pencil mark textview

3. **private int x**
   current number space position x

4. **private int y**
   current number space position y

5. **private boolean editable**
   if this number space is editable

6. **private boolean warning**
   if this number space is in warning state

7. **private boolean selected**
   if this number space is selected

8. **private Map<String,Boolean> enabledNum**
   check user have input which number in this number space, allow multiple input

9. **private GameNumOnClick gameNumOnClick**
   Event execute when this game number space is clicked

10. **private int markCount**
    Number of pencil mark in this number space

11. **private String selectedNum**
    current selected number (default is “”, keep default value when number of pencil mark is greater than 1)

Constructor

1. **public GameNumSpace(Context context,int x,int y)**
   initialize context and position

2. **public GameNumSpace(Context context, AttributeSet attrs)**
   default constructor from parent

3. **public GameNumSpace(Context context, AttributeSet attrs, int defStyleAttr)**
   default constructor from parent

Private methods

1. **private void init(Context context)**
   initialize all component and set onclick listener

Public methods

1. **public void setEditable(boolean editable)**
   editable setter, also modify the background base on editable value

2. **public boolean getEditable()**
   editable getter

3. **public void setGameNumOnClick(GameNumOnClick g)**
   gameNumOnClick setter

4. **public void setNum(String s)**
   set/remove the number from number space

5. **public int getPosX()**
   position x getter

6. **public int getPosY()**
   position y getter

7. **public int getMarkCount()**
   markCount getter

8. **public void setWarning()**
   warning setter

9. **public String getSelectedNum()**
   selectedNum getter

10. **public boolean getWarning()**
    warning getter

11. **public void setWarningBack()**
    set the background to warning color

12. **public void select()**
    action perform when this number space is selected

13. **public void deselect()**
    action perform when this number space is deselected

14. **public void setToDefault()**
    set background color to default

15. **public Map<String, Boolean> getEnabledNum()**
    enabbledNum getter

Sub interface

1. **GameNumOnClick**
   Allow user set action performed while this number space is on click

   methods:
   a. **void onClick(GameNumSpace g,int x, int y)**

public class LeaderBoard extends AppCompatActivity

Class for Leader board activity, which initiate the recycler view and other component in the Leader board layout

Overridden method:

1. **protected void onCreate(@Nullable Bundle savedInstanceState)**
   initialize all the element

class LeaderBoardAdapter extends RecyclerView.Adapter<LeaderBoardAdapter.LeaderBoardViewHolder>

Adapter for recycler view in leader board activity

Field:

1. **private AppCompatActivity activity**
   parent activity

2. **private List<Result> results**
   List of results to show in recycler view

Overridden method:

1. **public LeaderBoardViewHolder onCreateViewHolder(ViewGroup parent, int viewType)**

2. **public void onBindViewHolder(LeaderBoardViewHolder holder, int position)**

3. **public int getItemCount()**

Sub class:

1. **static class LeaderBoardViewHolder extends RecyclerView.ViewHolder**
   view holder for this adapter

public class MainActivity extends AppCompatActivity

Class for main activity

Overridden method:

1. **protected void onCreate(@Nullable Bundle savedInstanceState)**
   initialize all the element

public class NumButton extends ConstraintLayout

Class for number button in game recycler view

Field:

1. **private Button btnNum**
   main button

Constructor (all is default parent constructor):

1. **public NumButton(Context context)**

2. **public NumButton(Context context, AttributeSet attrs)**

3. **public NumButton(Context context, AttributeSet attrs, int defStyleAttr)**

Private method:

1. **private void init(Context context)**
   initialize all component

Public method:

1. **public void setNum(int i)**
   set the display number

public class SavedGame extends AppCompatActivity

Class for saved game activity, which initiate the recycler view and other component in the saved game layout

Overridden method:

1. **protected void onCreate(@Nullable Bundle savedInstanceState)**
   initialize all the element

class SavedGame Adapter extends RecyclerView.Adapter<SavedGameAdapter. SavedGameViewHolder>

Adapter for recycler view in saved game activity

Field:

1. **private AppCompatActivity activity**
   parent activity

2. **private List<Result> results**
   List of results to show in recycler view

Overridden method:

1. **public SavedGameViewHolder onCreateViewHolder(ViewGroup parent, int viewType)**

2. **public void onBindViewHolder(SavedGameViewHolder holder, int position)**

3. **public int getItemCount()**

Sub class:

2. **static class SavedGameViewHolder extends RecyclerView.ViewHolder**
   view holder for this adapter

public class Setting extends AppCompatActivity

Class for setting activity

Overridden method:

1. **protected void onCreate(@Nullable Bundle savedInstanceState)**
   initialize all the element

public class SettingPreference extends PreferenceFragmentCompat

Class for setup the preference fragment in setting activity

Overridden method:

1. **public void onCreatePreferences(Bundle savedInstanceState, String rootKey)**
   setup the preference

public class Sudoku

Class for generate random sudoku

Field:

1. **private static int size**
   size of the board

2. **private static int[] gridSize**
   grid size of the board (example: 9*9 board should have 3*3 grid)

Static method:

1. **static String[][] getBoard()**
   get the generated board

2. **private static String[][] randBoard()**
   randomize whole board

3. **private static String[][] randBoardNum(String[][] board)**
   randomize number in board

4. **private static String[][] randBoardGrid(String[][] board)**
   randomize position of grid of the board

5. **private static String[][] randBoardRowCol(String[][] board)**
   randomize position of row and column in board

6. **private static String[][] copyArray(String[][] arr)**
   copy 2d array

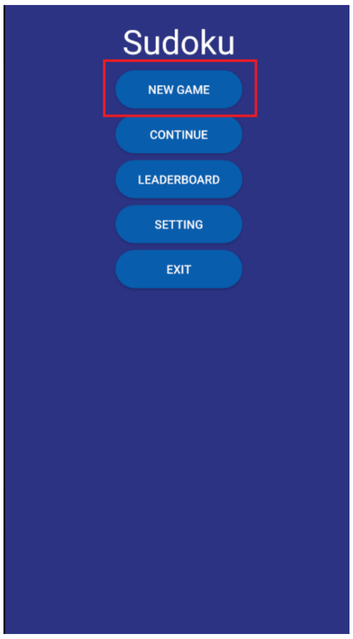
7. **private static int[] getGridSize(int size)**
   get grid size base on size

Tutorial

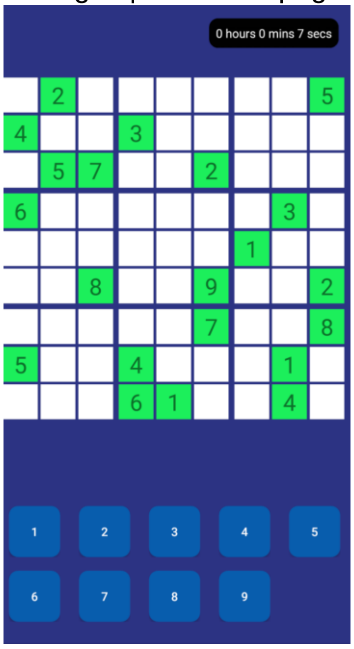
1. Game play

This game is design to allow people play sudoku at any place

User can press new game to play a new board

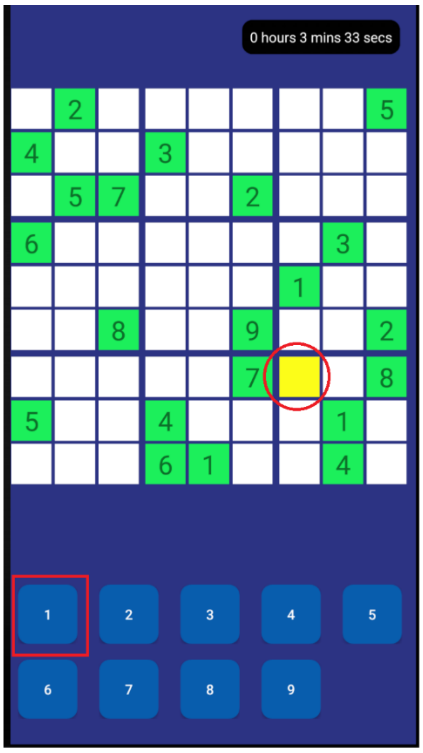


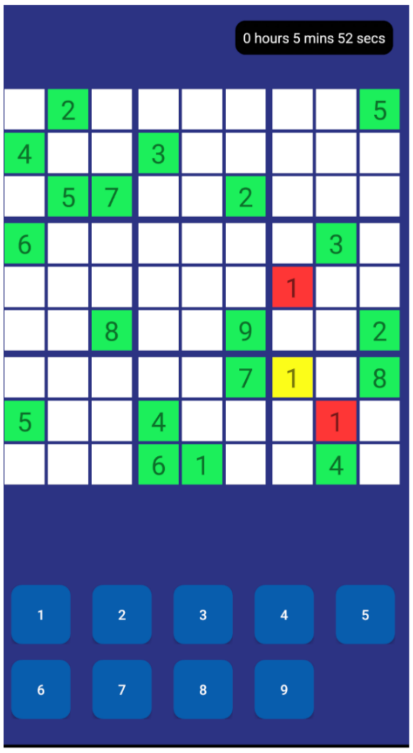
Then, user should be redirect to here, and the timer start running, timer style is base on user setting in preference page



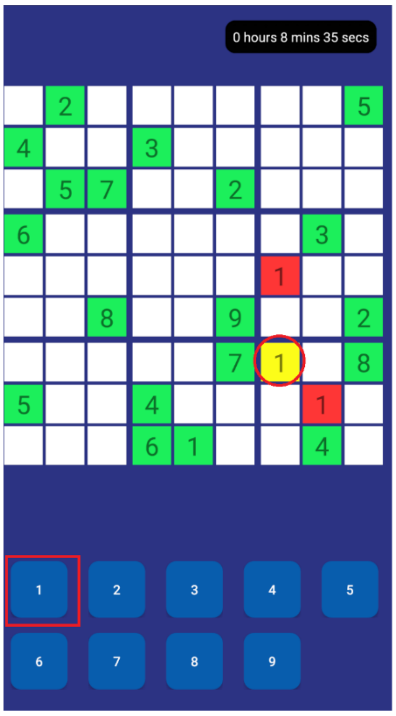
	2							5
4			3					
	5	7			2			
6							3	
						1		
		8			9			2
					7			8
5			4				1	
			6	1			4	

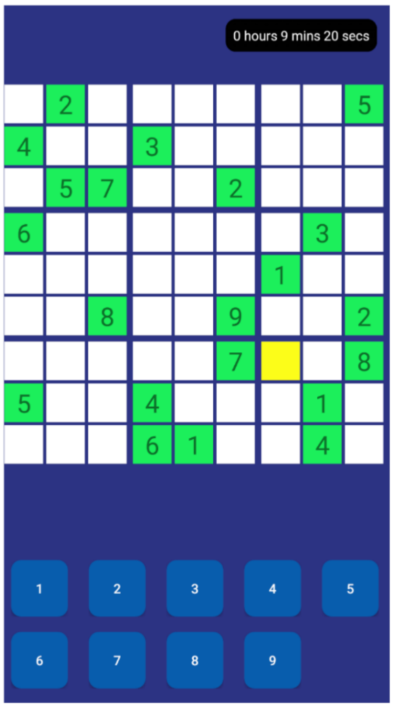
To put a number in to the board, user need to select a space, and press the number button below, if the number is invalid, the same number in grid, row and column will be highlighted





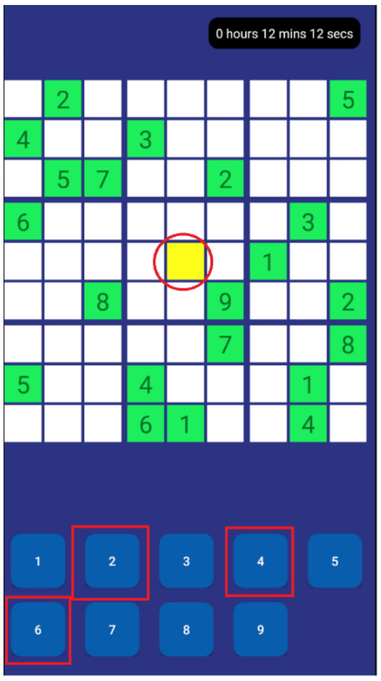
If user want to deselect a number, just select the grid, and press the number button

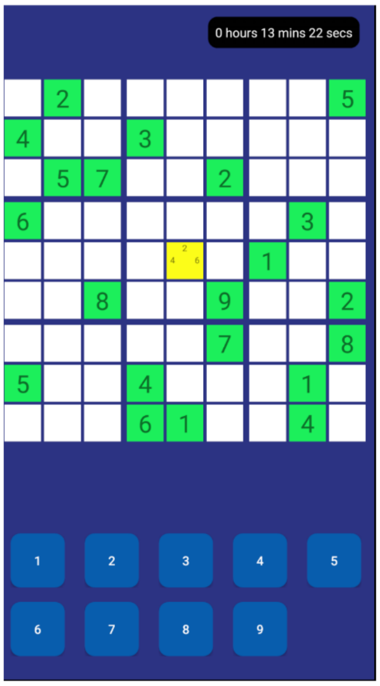




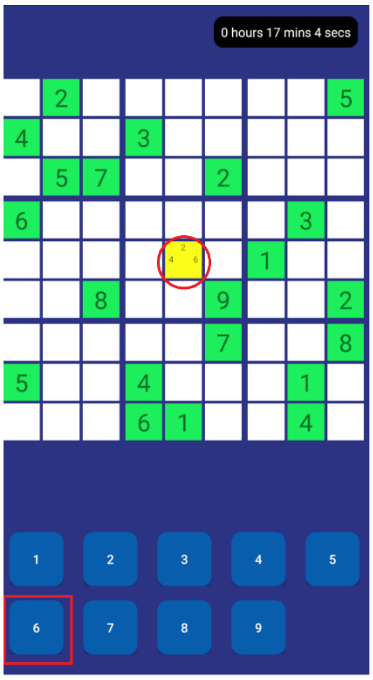
If user want make a pencil mark, just select the space and press the numbers they wish to put,

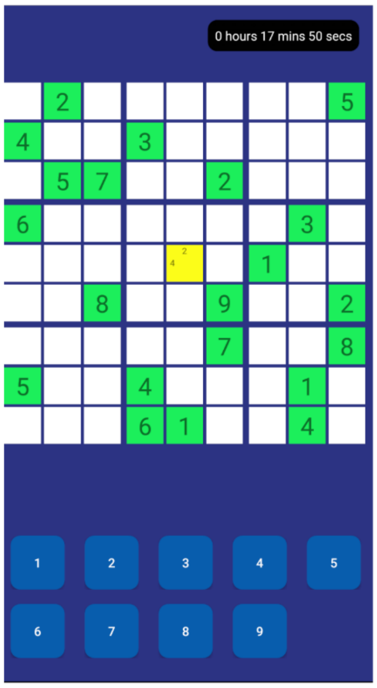
Let’s say user want to put 2, 4, 6, just select the grid and press 2, 4, 6



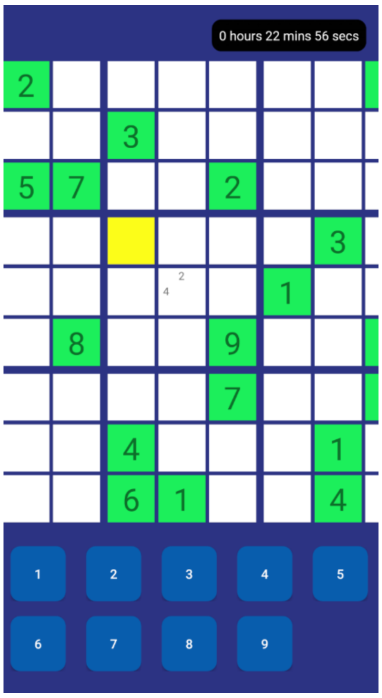


Remove 6 from pencil mark can done by select the space and click on 6

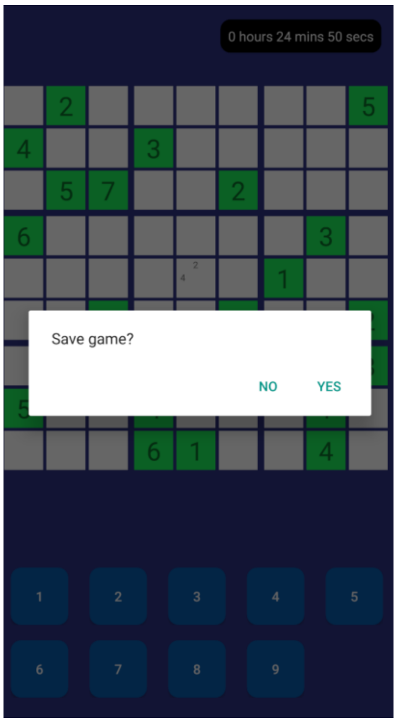




User can zoom in/out the board by using two finger



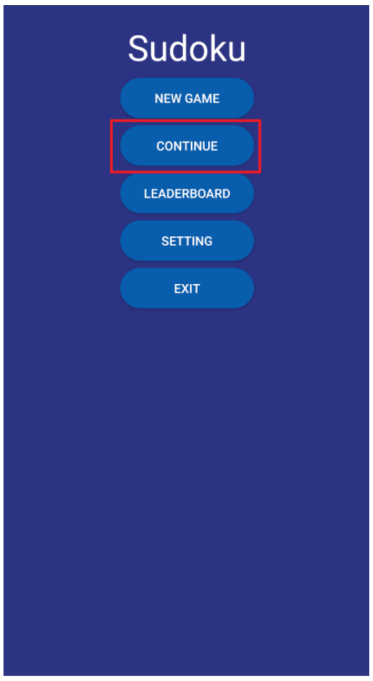
When user press back, the game will save depend on user setting (current setting is ask)



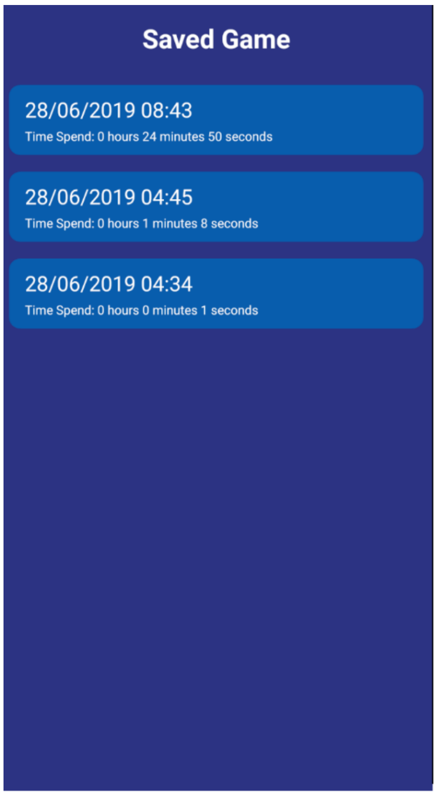
Save game?

2. Continue game

User can select continue when wish to continue previous saved game



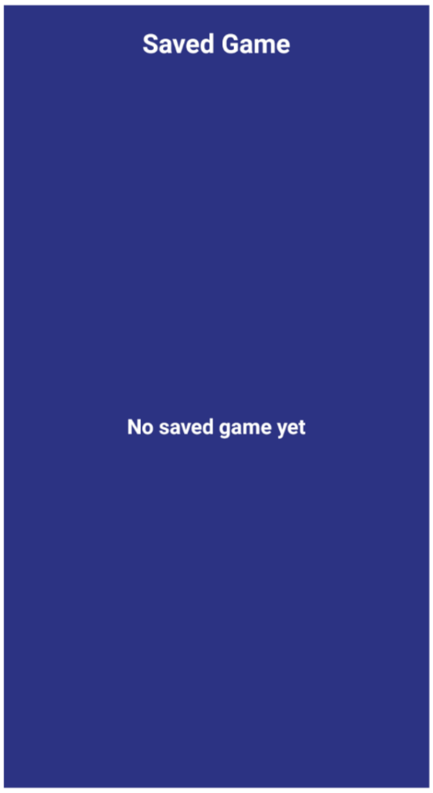
If there’s a saved data, user can select and continue, otherwise no data will be show



28/06/2019 08:43
Time Spend: 0 hours 24 minutes 50 seconds

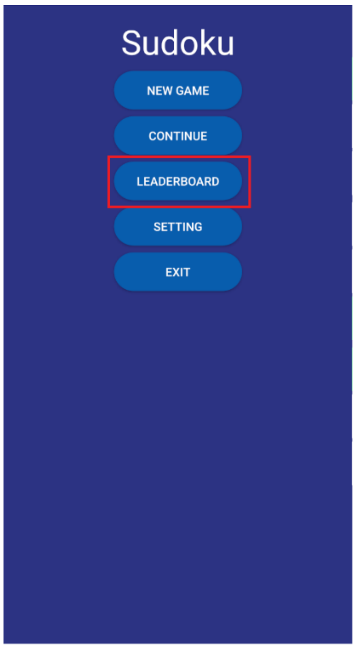
28/06/2019 04:45
Time Spend: 0 hours 1 minutes 8 seconds

28/06/2019 04:34
Time Spend: 0 hours 0 minutes 1 seconds

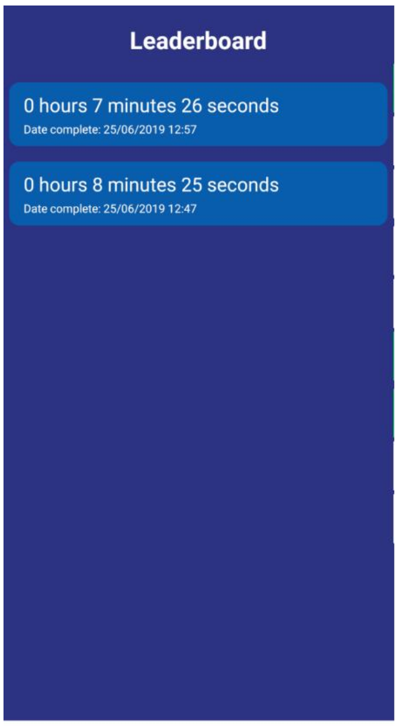


3. Check Leader Board

User can check leader board by click on leader board button

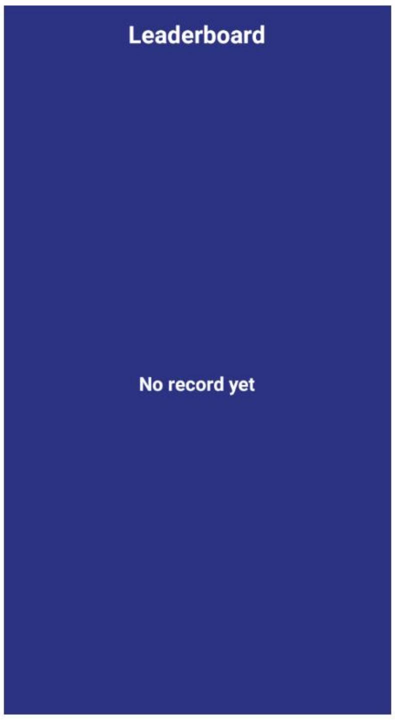


If user have complete a game, spent time will record in leader board, otherwise no data will be show


Leaderboard

0 hours 7 minutes 26 seconds
Date complete: 25/06/2019 12:57

0 hours 8 minutes 25 seconds
Date complete: 25/06/2019 12:47


Leaderboard

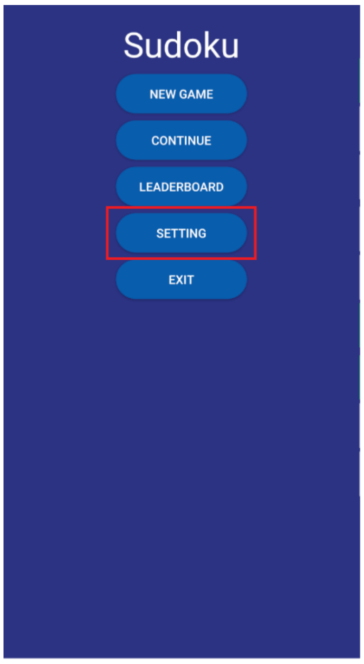
No record yet

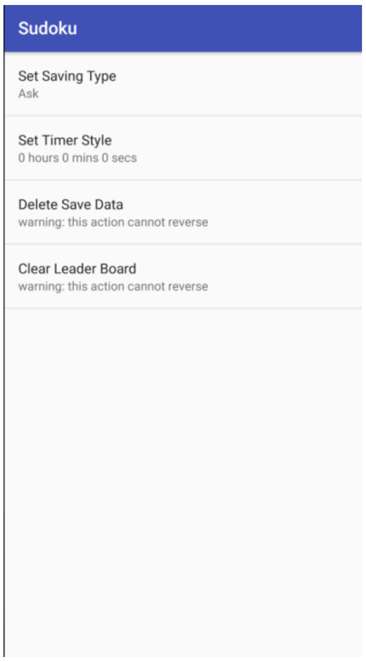
4. Setting

User can customize the game in setting page,

In setting page, user can setup the style of the timer, saving type

They can even delete their save data and clear the leader board




Sudoku

Set Saving Type
Ask

Set Timer Style
0 hours 0 mins 0 secs

Delete Save Data
warning: this action cannot reverse

Clear Leader Board
warning: this action cannot reverse