GIT

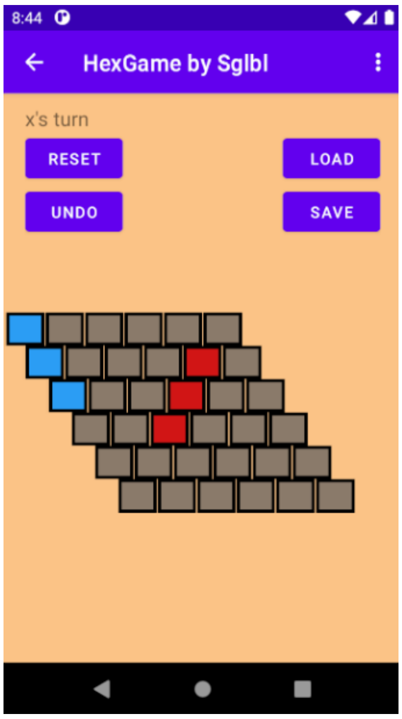
DEPARTMENT OF
COMPUTER ENGINEERING

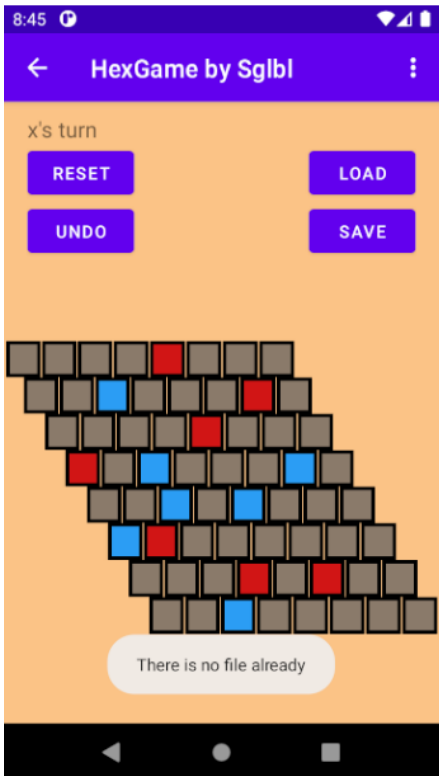
CSE 222/505 – Spring 2021

REPORT FOR WINTER PROJECT

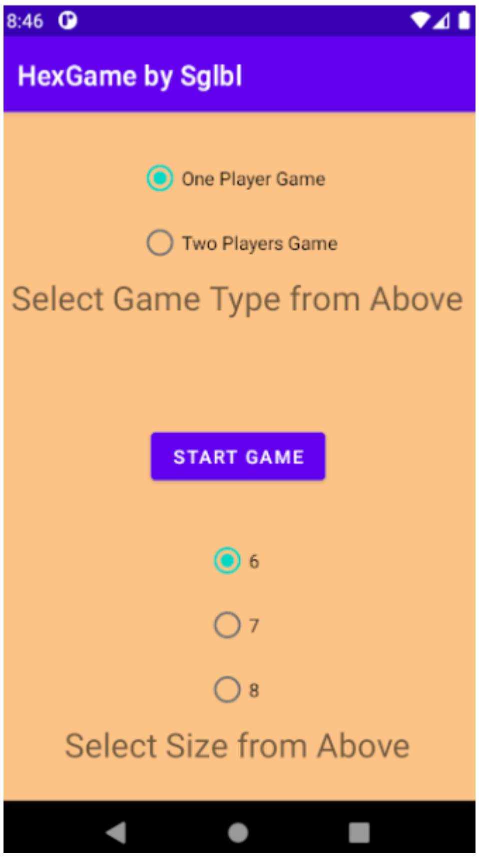
SÜLEYMAN GÖLBOL
1801042656

SCREENSHOTS





There is no file already



[✓ One Player Game]

Two Players Game

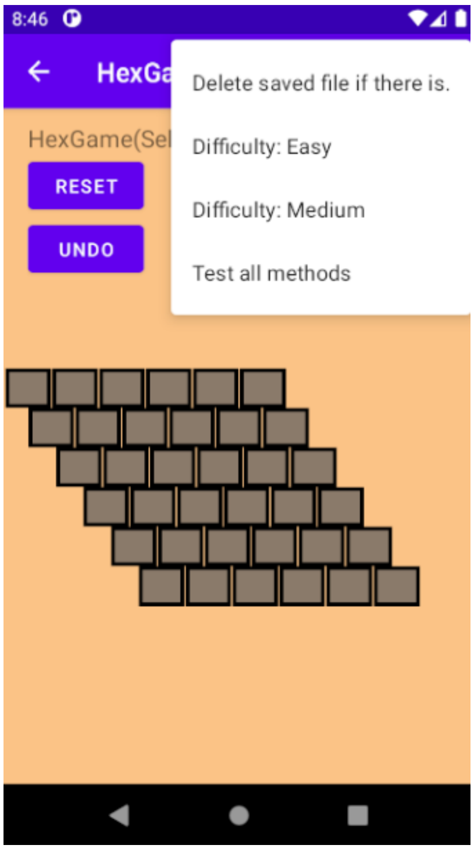
Select Game Type from Above

[✓ 6]

7

8

Select Size from Above



PSUEDOCODE AND ALGORITHMS OF HEURISTIC FUNCTION

```
START Class Main
    Create Button for starting game
    Create radio buttons to select size
    Create radio buttons to select how many player

    function() CheckHowManyPlayer
        if button clicked
            show toast as "Selected Player Type: "

    function() CheckSize
        if button clicked
            show toast as "Selected Size: "

    function OpenSecondPage(size, player type)
        if button clicked
            Start game with selected size and player type
END CLASS

START Class MainActivity
    Create variables for size,turn, player type, label
    Create Button variables for reset,undo,load,save and all board

    function onCreate()
        set player type and set label
        call addIDs()

    function onCreateOptionsMenu()
        Add 3 dots menu at the top of screen

    function addIDs()
        Make all buttons connected with buttons in Xml file
        Set text and color for all buttons
        call fourButtonClick()

    function fourButtonClick()
        Make reset,load,save,undo buttons coonected with buttons in Xml file
        if load button clicked call readFromFile()
        if save button clicked call writeToFile()


    function onClick()
        if board buttons clicked look whose turn and make move with play() and play(i,j)

    function play(row, column) #move of user
        check whose turn and make move with that row and column in board
        check if game ended
        if game ended
            change label and return
        else
            change turn

    function play() #move of computer
        set variable maxVal to -10000
        Create GradientDrawable for button background without border
        for i=0 until size of board
            for j=0 until size of board
                if button i row and j column is empty
                    set it 'o'
                    if isEnd() equals 3 #o(blue) won the game.
                        set button color to blue
                        set last move
                        change turn
                        end game
                    set minimax(level, isMaximizing) to moveValue
                    set it '.'
                    if moveValue > maxVal
                        set maxValto moveValue
                        set best move to i row and j column

        change button color to blue
        change turn
        set oldMove if user clicks undo
        if computer won
            change label to "O won"

    function change_turn()
        if turn equals x
            turn = o
        else
            turn = x

    function getBoardSize()
        return size of board

    function isEnd()
        create 2d array in size of board for checking if recursively visited
            make visiteds every element to 0
        for m=0 until size of board
            if m row of boards first column='x'
                call check_full() method
                if check_full() equals 2
                    return 2 #x won

            if m column of boards first row='o'
                call check_full() method
                if check_full() equals 3
                    return 3 #o won

            return 0; #no one won yet
```

```
function minimax(level , isMaximizing)
    set score to 0
    if level equals 0 or someone won
        return minimaxScore()
    if board is full
        return 0
    if maximizing
        set bestScore to -10000
        for i=0 until size of board
            for j=0 until size of board
                if button in i row j column equals '.'
                    set it to 'o'
                    if minimax score > bestScore
                        set bestScore to minimax score
        return bestScore
    else
        set worstScore to 10000
        for i=0 until size of board
            for j=0 until size of board
                if button in i row j column equals '.'
                    set it to 'x'
                    if minimax score < worstScore
                        set worstScore to minimax score
        return worstScore

function isInInterval(i,j)
    if i and j bigger than -1 and smaller than size
        return true
    else
        return false
function min (s1, s2)
    return smaller variable
function max (s1, s2)
    return bigger variable

function minimaxScore(isMaximizing)
    set variable score power to 3 and score1,score2, score3 to 0
    for i=0 until size of board
        for j=0 until size of board
            check interval and button status and if valid
                increase score1

            if button status equals to 'o'
                check status of next row and if is valid
                    increase score2

            if o won
                return 250
            if x won
                return -250
```

```
function check_full(column, row, visited, checking player)
    if player equals 'x' and column equals size-1
        for m=0 until size of board
            if button of row m and column size-1 equals 'x'
                return 2 #because x won

    if player equals 'o' and row equals size-1
        for m=0 until size of board
            if button of row size-1 and column m equals 'o'
                return 3 #because o won

    else #recursively find if someone wins
        if right top is valid to move and not visited
            make visited = 1
            call check_full() and if true
                return 2

        if right is valid to move and not visited
            make visited = 1
            call check_full() and if true
                return 2

        if bottom is valid to move and not visited
            make visited = 1
            call check_full() and if true
                return 2

        if left bottom is valid to move and not visited
            make visited = 1
            call check_full() and if true
                return 2

        if left is valid to move and not visited
            make visited = 1
            call check_full() and if true
                return 2

        if top is valid to move and not visited
            make visited = 1
            call check_full() and if true
                return 2

        make column and row of visited = 0 # recursively make it 0

    return 0

function writeToFile()
    Create FileWriter inside Buffered Writer and create text file.
    write size to file as string
    write player type to file as string
    write whose turn to file as string

    for i=0 until size of board
        for j=0 until size of board
            write to file status of buttons i row and j column
        add new line '\n'

    write lastMove of user
    if player type equals one player game
        write lastMove2 of computer

    if process is successful
        show toast as "File saved to device "
    if process is not successful
        show toast as "Error "

function readFromFile()
    Create GradientDrawable for button background without border
    Create FileReader inside Buffered Reader and select text file.

    read size from file and convert to integer
    raad player type from file
    read whose turn from file

    for i=0 until size of board
        for j=0 until size of board
            read from file status of buttons i row and j column


    read last move
    if player type equals one player game
        read last move 2

    if process is successful
        show toast as "File readed to device "
    if process is not successful
        show toast as "Error "

function isValidMove(row,column, letter)
    check if x and y is bigger or equals than 0 and smaller than size
        check if status of button i row and j column equals to letter #turn
            return true

    if false
        return false

function changeColor(row, column, color)
    Create GradientDrawable for button background without border
    get background color of button i row and j column
        if color equals 1 #gray
            set color of that button to gray
```

```
                  if color equals 2 #red
                        set color of that button to red
                  if color equals 3 # blue
                        set color of that button to blue

END CLASS

START CLASS Move
   private variable x,y,status
   function setAll()
         set x,y,status to objects variables
   function getRow()
         return y
   function getColumn()
         return x
END CLASS
```