Team: Jacob Brauchler
Alex Tzinov
Kyle Wiese

Title: Checkers!

Project Summary:
We are making an interactive, gui based checkers game against an AI that will have varying levels of difficulty.

Requirements:

No Business requirements

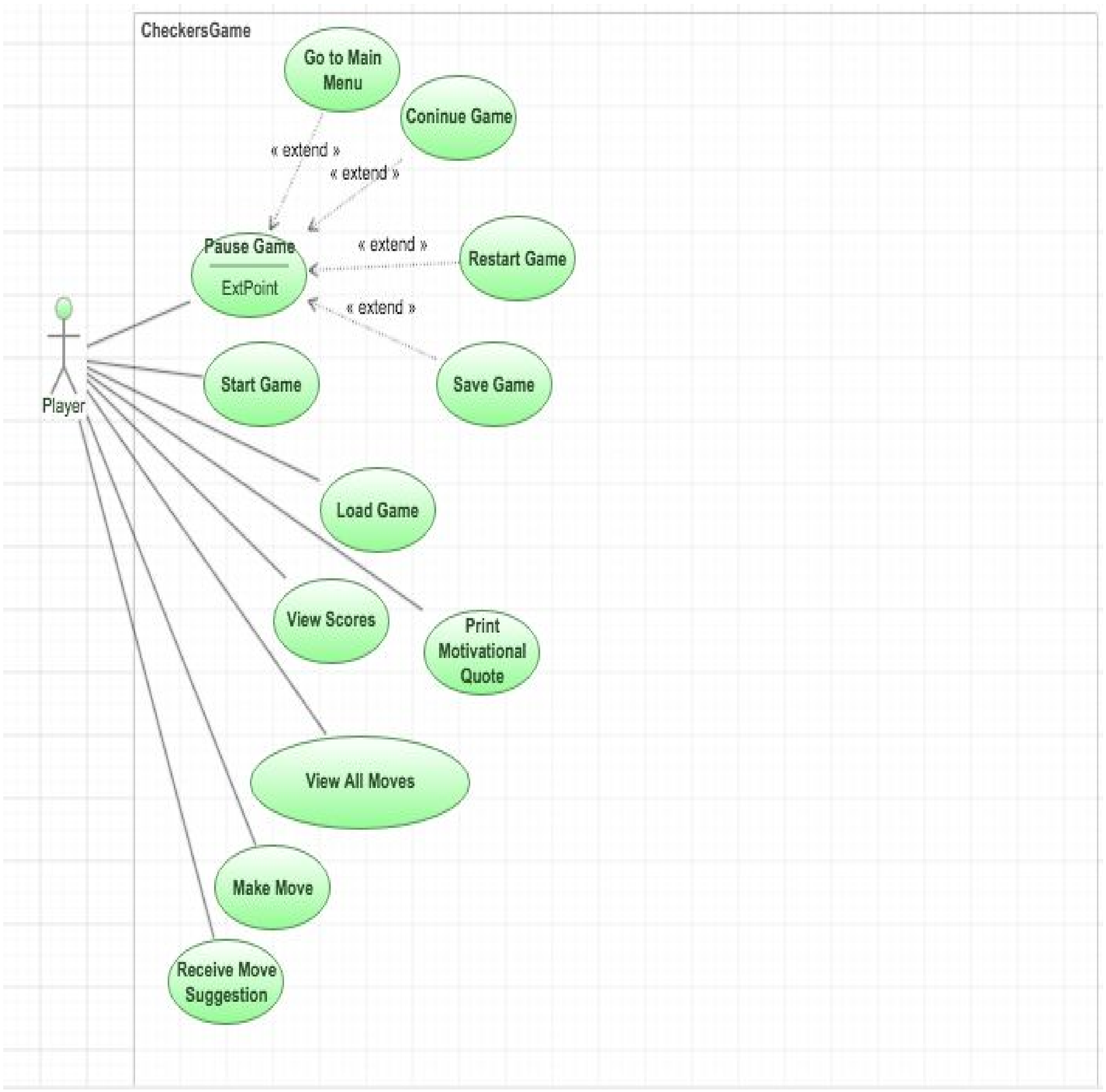
User Requirements		
ID	Description	Priority
US-01	As a player, I can make moves	High
US-02	As a player, I can see my score and game time	Medium
US-03	As a player, I can interact with game menus	High
US-04	As a player, I can save and come back to a game	Medium
US-05	As a player, I can pause a game	Low
US-06	As a player, I can leave a game	Low
US-07	As a player, I can play against an AI	High
US-08	As a player I can view what moves I can make	Medium

Functional Requirements		
ID	Description	Priority
FR-01	When saving a game, a SQL database is overwritten with the current game state	Medium
FR-02	When restarting a game, the game board is	High

	reset to the default state	
FR-03	When pause menu is displayed, normal game functions (i.e moving) are disabled	Medium
FR-04	When loading a game, the SQL database is read and game state is set based on information in database	Medium
FR-05	When AI must make move, all possible moves are calculated and move is chosen based on difficulty	High
FR-06	After player makes move, all game functions (i.e move) are disabled for that player until the other player makes a move	High
FR-07	When “go to main menu” is selected, game board is deleted and main menu is displayed	Medium

Non-Functional Requirements		
ID	Description	Priority
NFR-01	AI makes a move in less than 2 seconds	High
NFR-02	Game should be playable on all platforms (Linux, Windows, OSX)	High
NFR-03	Database should be replaceable with another relational database	Medium
NFR-04	When starting a game, the game board should be set in less than a second	High
NFR-05	When loading a game, the game board should be set in less than a second	High

Use Case Diagram:



<u>Use Cases:</u>

Actors:

Player

Kyle Wiese Use Cases:

Use Case ID	UC-01		
Use Case Name	Pause Game		
Description	Allows the player to pause the game in its current state		
Actors	Player		
Pre-Conditions	Game is currently in play		
Post-Conditions	Game is paused and menu options are displayed (neither player has moved)		
Frequency of Use	User discretion		
Flow Of Events	Event #	Actor Action	System Response
	1	Player presses “Pause”	Game timer is paused and menu options appear
	2	Player sees pause menu	
Variations	1. (user) Player presses ‘p’ to pause game 1. (system) Game timer is paused and menu options appear		
Notes and Issues			
Developer Notes			

Use Case ID	UC-02		
Use Case Name	Save Game		
Description	Allows the player to save the game in its current state		
Actors	Player		
Pre-Conditions	Game is started		
Post-Conditions	Game is saved in permanent storage (neither player has moved)		
Frequency of Use	User discretion		
Flow Of Events	Event #	Actor Action	System Response
	1	Player enters pause menu	Pause menu is displayed to user
	2	Player clicks “Save Game” from the pause menu	The game board state, time, and turn are stored into permanent storage and user is notified
	3	Player sees notification that game is saved	
Variations	1. (user) Player presses ‘s’ 1. (system) The game board state, time, and turn are stored into permanent storage and user is notified when game is saved 2. (user) N/A 2. (system) N/A 3. (user) Player sees notification that game didn’t save correctly 3. (system) Game is ended		
Notes and Issues			
Developer Notes	Only the latest save should be stored		

Use Case ID	UC-03		
Use Case Name	Restart Game		
Description	Allows user to completely restart game		
Actors	Player		
Pre-Conditions	Game is started		
Post-Conditions	A completely new game board is present		
Frequency of Use	User discretion		
Flow Of Events	Event #	Actor Action	System Response
	1	Player enters pause menu	Pause menu is displayed to user
	2	Player selects “Restart Game”	“Are you sure?” prompt is displayed
	3	Player selects “Yes”	Game board, time, score, and turn is reset
	4	Player can make moves	
Variations	3. (user) Player selects “No” 3. (system) User is returned to pause menu		
Notes and Issues			
Developer Notes			

Use Case ID	UC-04		
Use Case Name	Continue Game		
Description	Allows player to continue where he/she left off after pause		
Actors	Player		
Pre-Conditions	Game is started and player is in pause menu		
Post-Conditions	Game is in same state as before when the user paused game (neither player has moved)		
Frequency of Use	User discretion		
Flow Of Events	Event #	Actor Action	System Response
	1	Player selects “Continue” from pause menu options	Pause menu is no longer displayed and game board, time, turn, and score are returned to what they were before
	2	Player sees full game board again and can make moves	
Variations			
Notes and Issues			
Developer Notes			

Jacob Brauchler Use Cases:

Use Case ID	UC-05		
Use Case Name	Go To Main Menu		
Description	Allows the player to leave the game and view the Main Menu		
Actors	Player		
Pre-Conditions	Game is paused		
Post-Conditions	Player can see the main menu		
Frequency of Use	Whenever you want to quit a game or save a game and come back to it.		
Flow Of Events	Event #	Actor Action	System Response
	1	Player clicks pause Button	System shows Pause menu
	2	Player clicks Main Menu button	Takes player to the main menu screen.
	3	Player sees Main Menu	
Variations	2. Press esc to go to main menu		
Notes and Issues			
Developer Notes			

Use Case ID	UC-06		
Use Case Name	Start Game		
Description	Player can start a game from the main menu		
Actors	Player		
Pre-Conditions	Must be at the main menu		
Post-Conditions	The player can see game is started with the board set.		
Frequency of Use	Every Time a new game is started		
Flow Of Events	Event #	Actor Action	System Response
	1	Click Start Game button	Start a new game and set the board.
	2	Player will see new game board	
Variations			
Notes and Issues			
Developer Notes			

Use Case ID	UC-07		
Use Case Name	View Scores		
Description	Allows the player to view their past scores playing checkers.		
Actors	Player		
Pre-Conditions	Player must be at Main Menu		
Post-Conditions	Player sees past scores		
Frequency of Use	Not frequently when player wants to check scores		
Flow Of Events	Event #	Actor Action	System Response
	1	Player Clicks View Scores	System accesses permanent storage and displays it to the player
	2	Player will see past scores	
Variations			
Notes and Issues			
Developer Notes			

Use Case ID	UC-08		
Use Case Name	Load Game		
Description	Player can load a saved game from the Main Menu		
Actors	Player		
Pre-Conditions	Player must be at Main Menu and have a saved game		
Post-Conditions	Player must see loaded game board and be able to continue the game		
Frequency of Use	Whenever the player wants to resume a game		
Flow Of Events	Event #	Actor Action	System Response
	1	Player clicks load game button	System will load saved board state and display it for user.
	2	Player will see loaded game	
Variations			
Notes and Issues			
Developer Notes			

Alex Tzinov Use Cases:

Use Case ID	UC-09		
Use Case Name	Make Move		
Description	Player picks a piece on the board and makes a move with it		
Actors	Player		
Pre-Conditions	> Both player's still have pieces on the board (neither player has lost the game yet) > It is the player's turn to make a move		
Post-Conditions	> Board is in a valid state (pieces only found on dark colored squares, no two pieces exist on the same square, etc) > Either both players still have pieces (neither has lost) OR one of the player's has lost all of their pieces and the other player has won > It is the AIs turn to make a move		
Frequency of Use	Every other turn in the game (every single time it's the player's turn to make a move)		
Flow Of Events	Event #	Actor Action	System Response
	1	Player identifies which piece they want to move (either by mouse or by keyboard entry of coordinates)	The system will have the board highlight available squares where this piece can move to
	2	Player then identifies where they want to move that piece	System validates the move and updates the state of the board, redraws the board for the player with the piece moved
	3	Player sees their piece move and has their turn completed	
Variations	1 (system) The system will warn the player that this piece cannot possibly be moved legally because either the piece can't move anywhere or there is a jump move available 2 (actor) Player decides to make a jump move 2. (system) System fails to validate move, prompts player to try a different, legal move 3. (actor) Player gets told their previous move was illegal, gets asked to make a move again		
Notes and Issues			
Developer Notes			

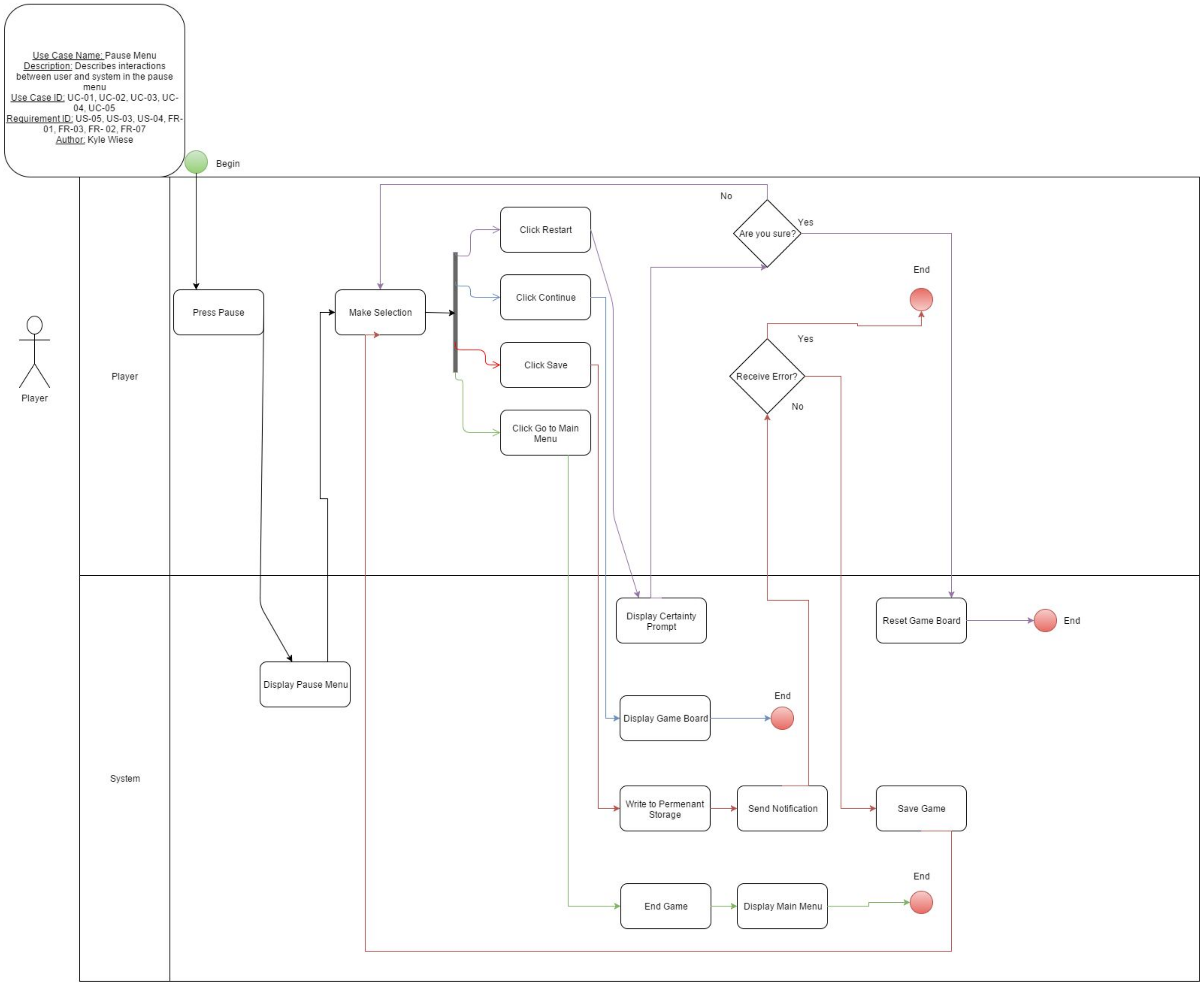
Use Case ID	UC-10		
Use Case Name	View All Moves		
Description	Player can view all of their possible moves on the board		
Actors	Player		
Pre-Conditions	> Both players still have pieces on the board (neither player has lost the game yet) > It is the player's turn to make a move		
Post-Conditions	> Board is in a valid state (pieces only found on dark colored squares, no two pieces exist on the same square, etc) > Both player's still have pieces on the board (neither player has lost the game yet) > It is **still** the player's turn		
Frequency of Use	Whenever the player decides to view his possible moves		
Flow Of Events	Event #	Actor Action	System Response
	1	Player clicks on the **View All Possible Moves** button	The system scans all of the player's pieces and for each piece that has a valid move, board will highlight that particular move
Variations	1 (actor) Player triggers this action using a keyboard shortcut 1 (system) User has a single available jump move, system will just highlight this		
Notes and Issues			
Developer Notes			

Use Case ID	UC-11		
Use Case Name	Receive Move Suggestion		
Description	Player can be provided with a hint / intelligent move that they can play		
Actors	Player		
Pre-Conditions	> Both player's still have pieces on the board (neither player has lost the game yet) > It is the player's turn to make a move		
Post-Conditions	> Board is in a valid state (pieces only found on dark colored squares, no two pieces exist on the same square, etc) > Either both players still have pieces (neither has lost) OR one of the player's has lost all of their pieces and the other player has won > It is **still** the player's turn		
Frequency of Use	Whenever the player decides to be given a hint		
Flow Of Events	Event #	Actor Action	System Response
	1	Player selects the **Get Smart Move** button	The system will look at all of the users possible moves (most likely using the logic used in the UC-10) and score each one of them based on the outcome of the board afterwards. The system will then provide the user (via highlighting) with a randomly selected, highest scoring move.
Variations	1 (user) Player triggers this action		
Notes and Issues			
Developer Notes			

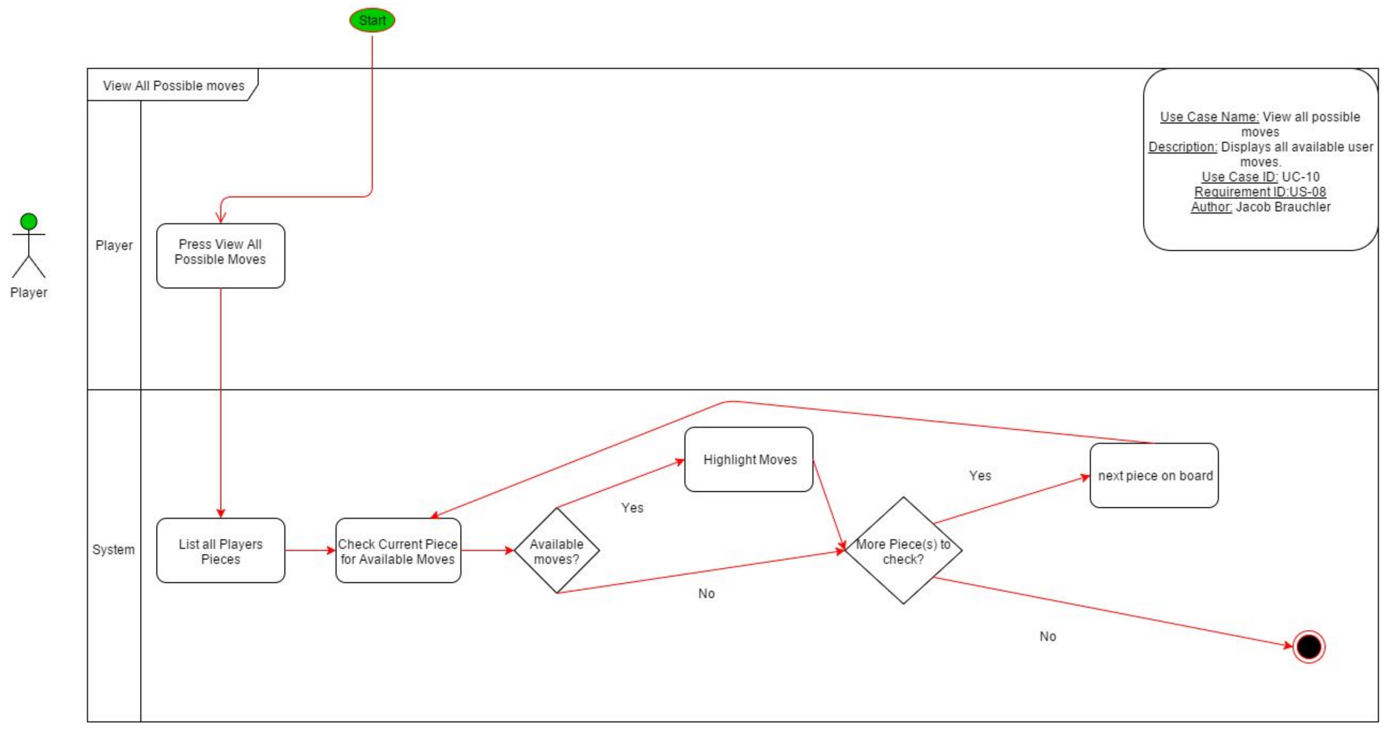
Use Case ID	UC-12		
Use Case Name	Print Motivational Quote		
Description	Player will be given an uplifting motivational message		
Actors	Player		
Pre-Conditions	> Both players still have pieces on the board (neither player has lost the game yet) > It is the player's turn to make a move		
Post-Conditions	> Board is in a valid state (pieces only found on dark colored squares, no two pieces exist on the same square, etc) > Both player's still have pieces on the board (neither player has lost the game yet) > It is **still** the player's turn		
Frequency of Use	Whenever the player decides they need motivation		
Flow Of Events	Event #	Actor Action	System Response
	1	Player clicks on the **Motivate Me** button	The system will select a random message from an array of potential motivational strings and print it to the user
Variations			
Notes and Issues			
Developer Notes			

Activity Diagrams:

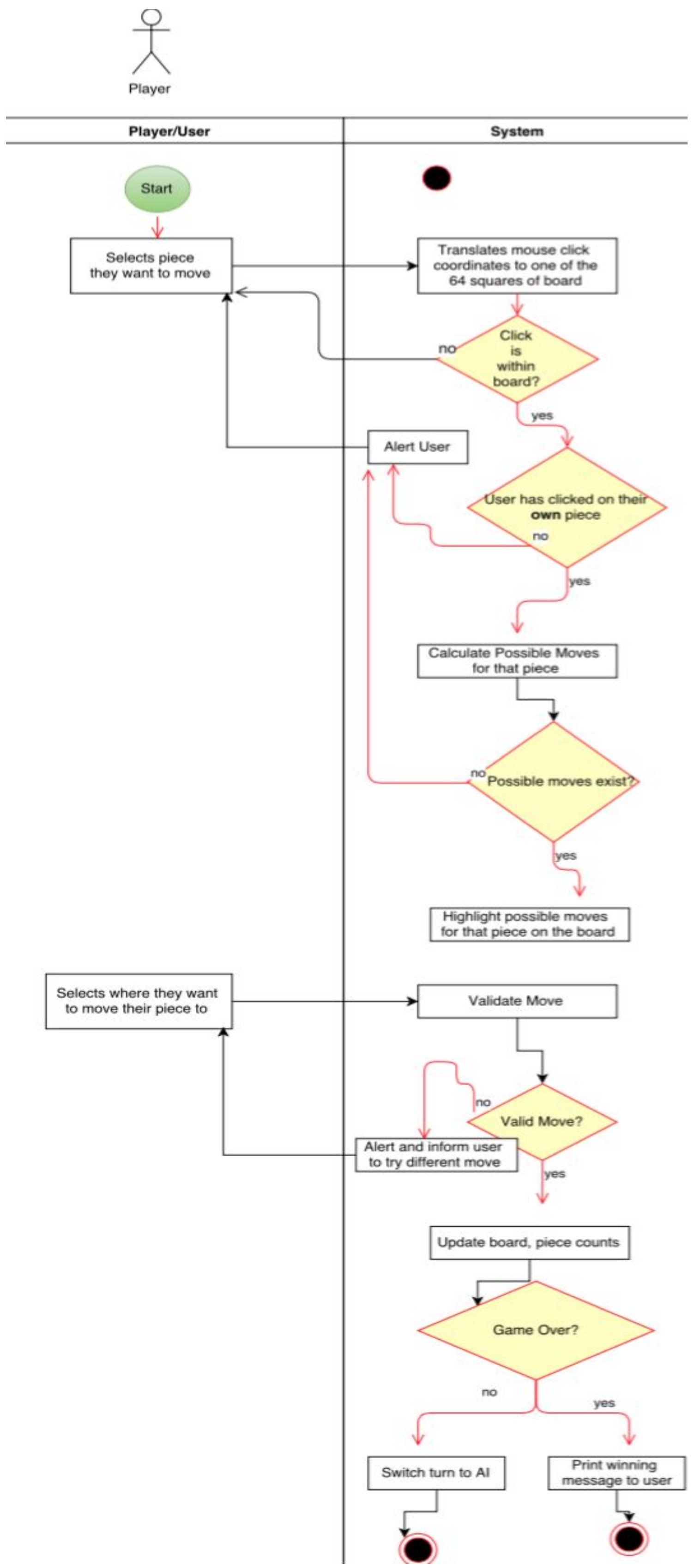
Kyle Wiese Activity Diagram:



Jacob Brauchler Activity Diagram:



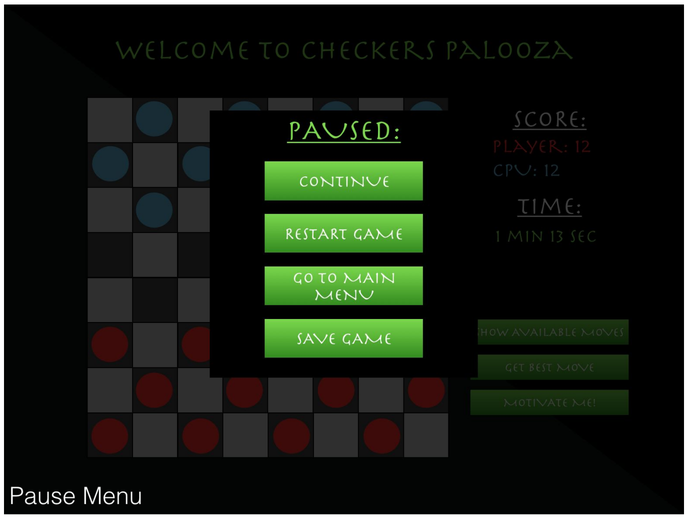
Alex Tzinov Activity Diagram(name: Make Move, case_id: UC-09, req_id: US-01. how user performs a move)


Player
Player/User
System
Start
Selects piece they want to move
Translates mouse click coordinates to one of the 64 squares of board
no
Click is within board?
yes
Alert User
User has clicked on their own piece
no
yes
Calculate Possible Moves for that piece
no
Possible moves exist?
yes
Highlight possible moves for that piece on the board
Selects where they want to move their piece to
Validate Move
no
Valid Move?
Alert and inform user to try different move
yes
Update board, piece counts
Game Over?
no
yes
Switch turn to AI
Print winning message to user

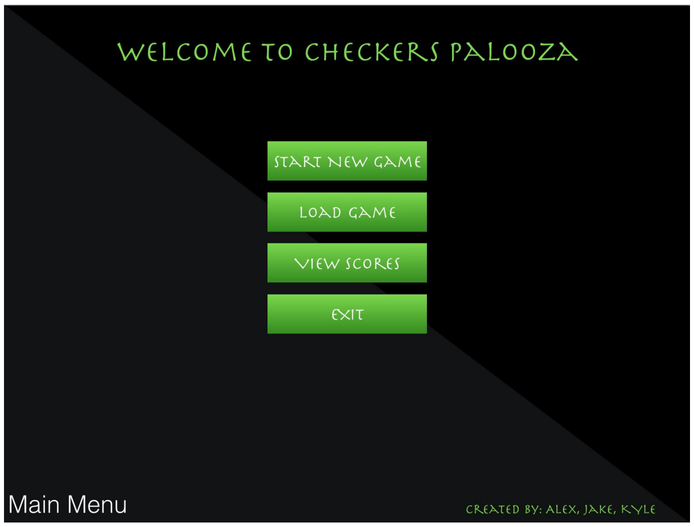
Data Storage:

- Type: SQL
- Classes:
  - DatabaseModel
    - Save game from pause menu
    - Load game

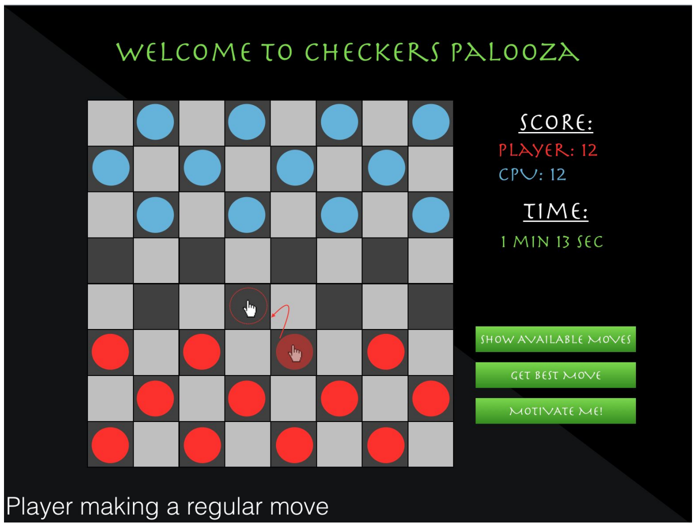
UI MOCK UPS:



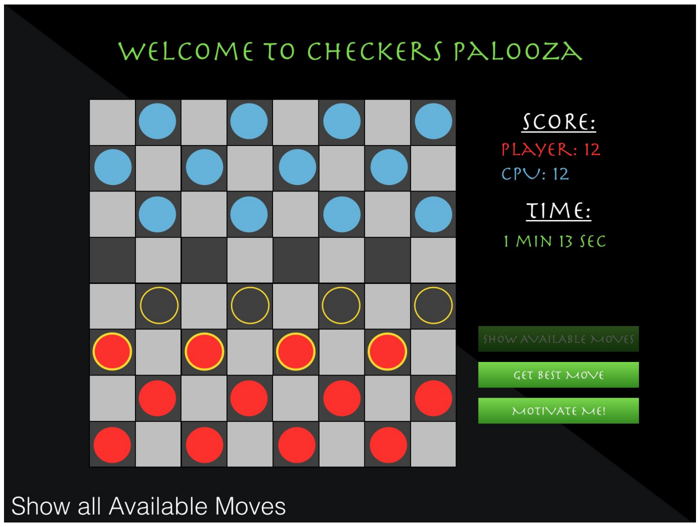
Pause Menu



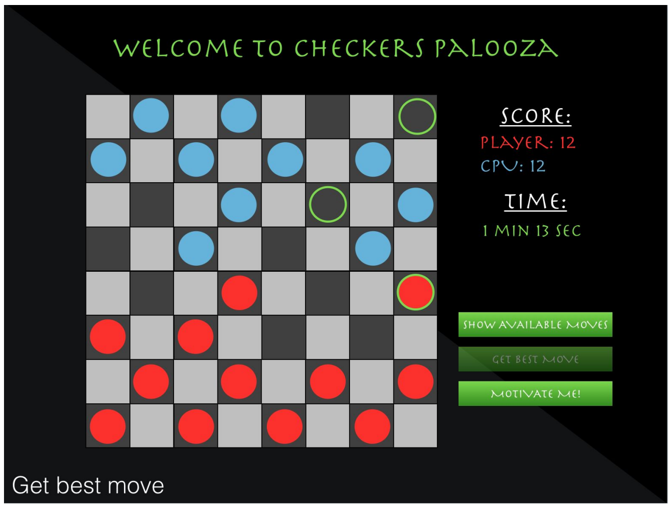
Main Menu



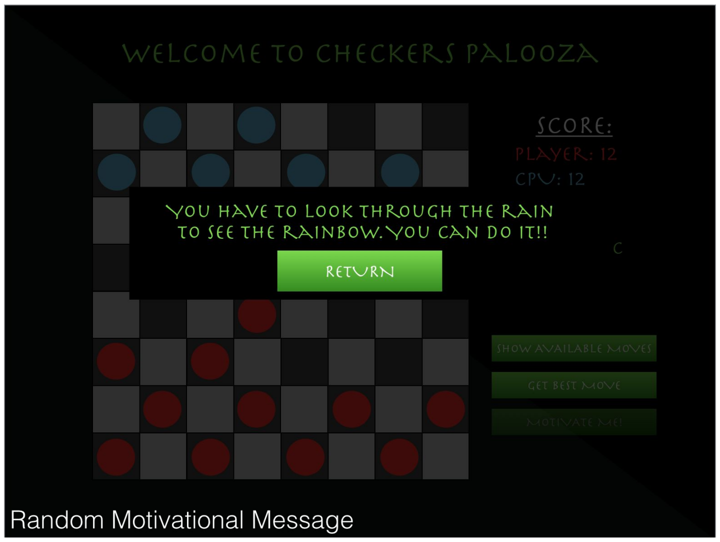
Player making a regular move



Show all Available Moves

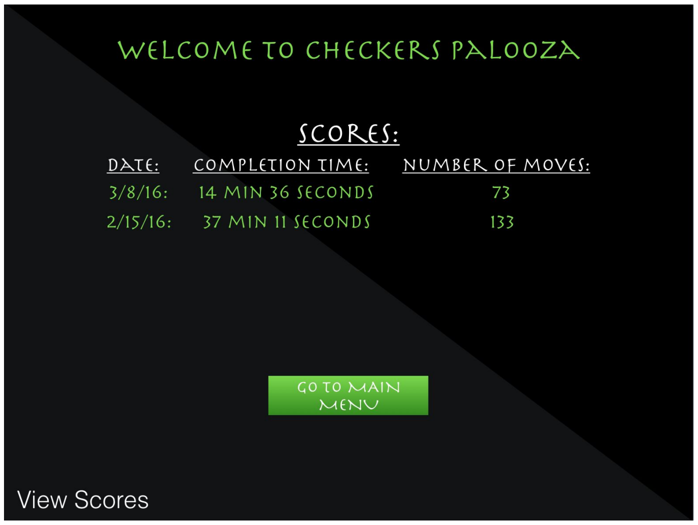


Get best move



YOU HAVE TO LOOK THROUGH THE RAIN TO SEE THE RAINBOW. YOU CAN DO IT!!

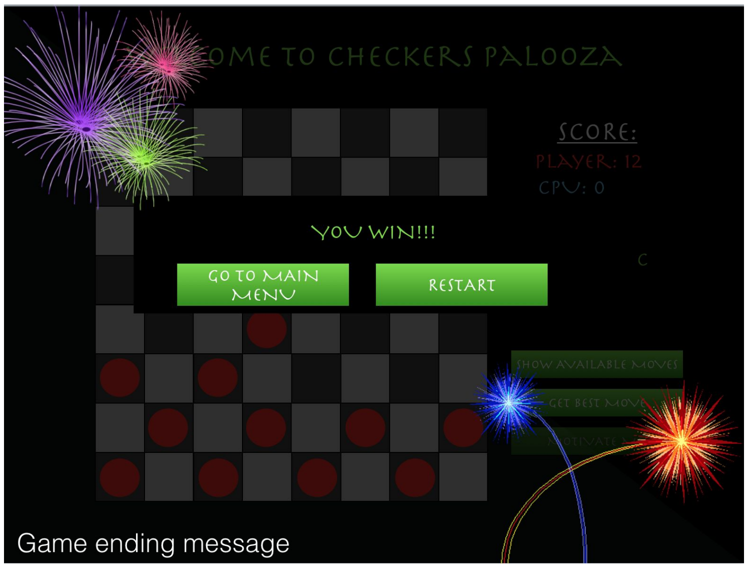
Random Motivational Message



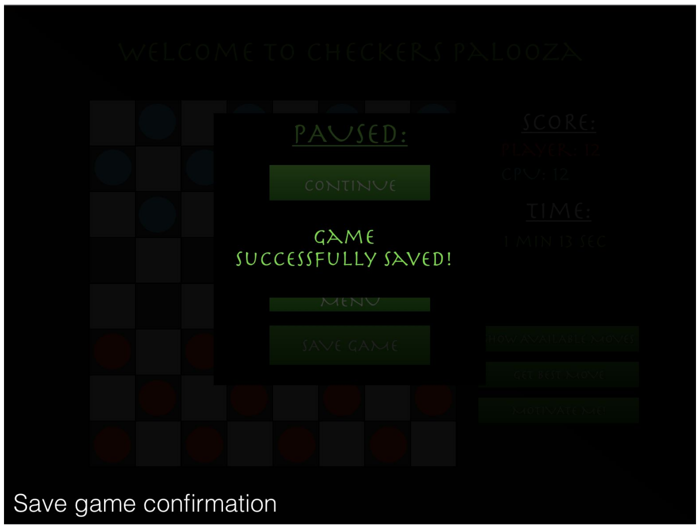
SCORES:

DATE:	COMPLETION TIME:	NUMBER OF MOVES:
3/8/16:	14 MIN 36 SECONDS	73
2/15/16:	37 MIN 11 SECONDS	133

View Scores



Game ending message



GAME SUCCESSFULLY SAVED!

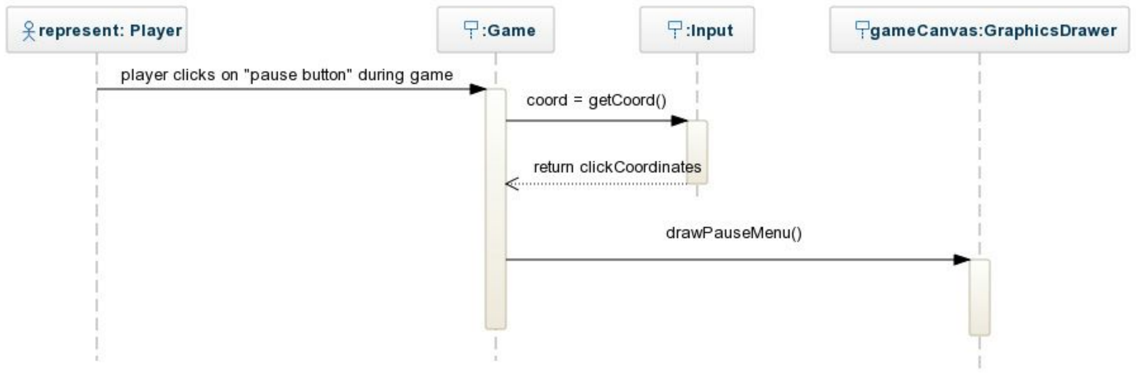
Save game confirmation

User Interactions (Sequence Diagrams):

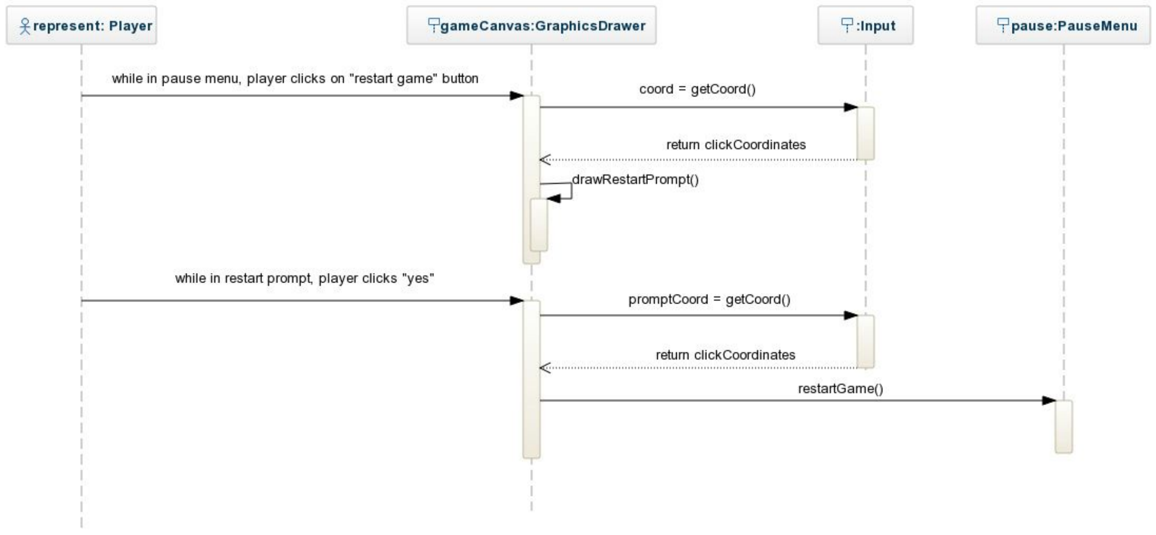
Kyle Wiese Diagrams:

Sequence Diagram Name: Pause Menu
Description: Describes interactions between user and the system in the pause menu
Use Case ID: UC-01, UC-02, UC-03, UC-04, UC-05
Requirement ID: US-05, US-03, US-04, FR-01, FR-03, FR- 02, FR-07

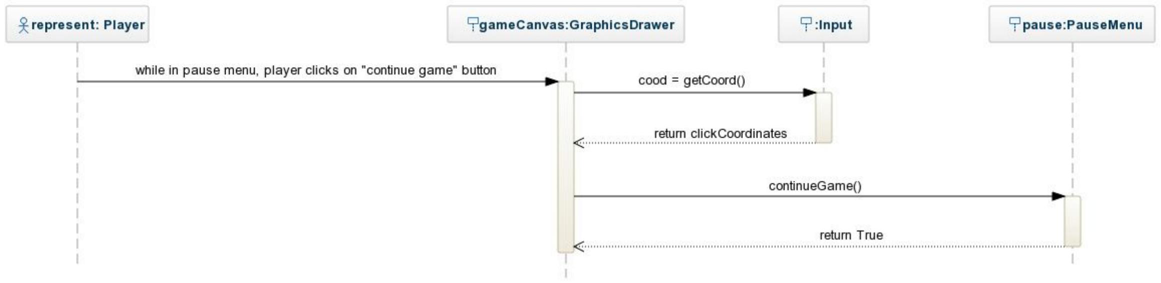
Draw Pause Menu:



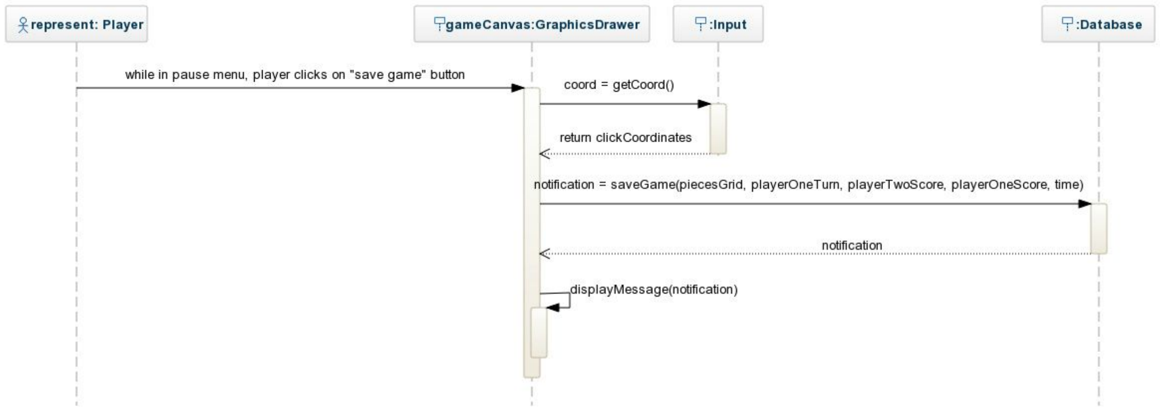
Restart Game (Already in Pause Menu):



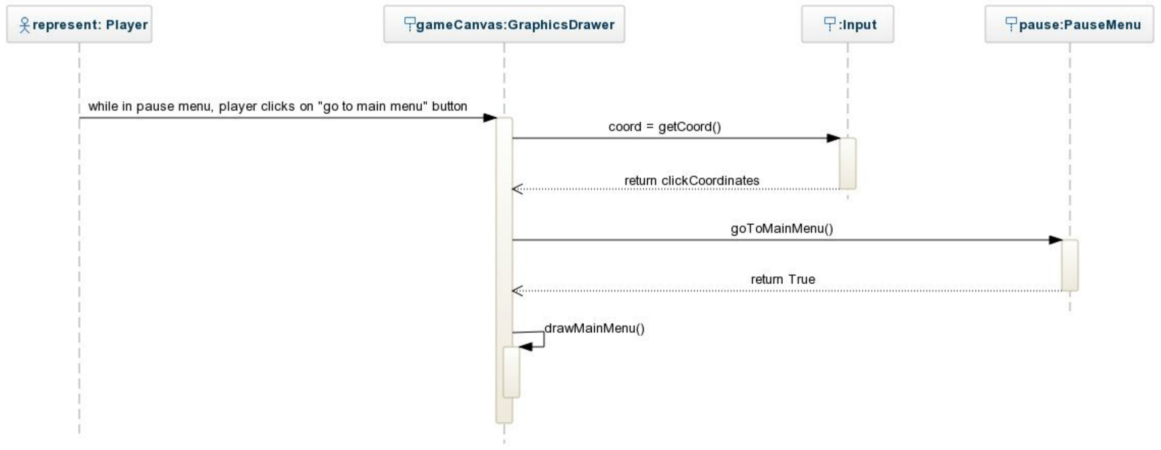
Continue Game (Already in Pause Menu):



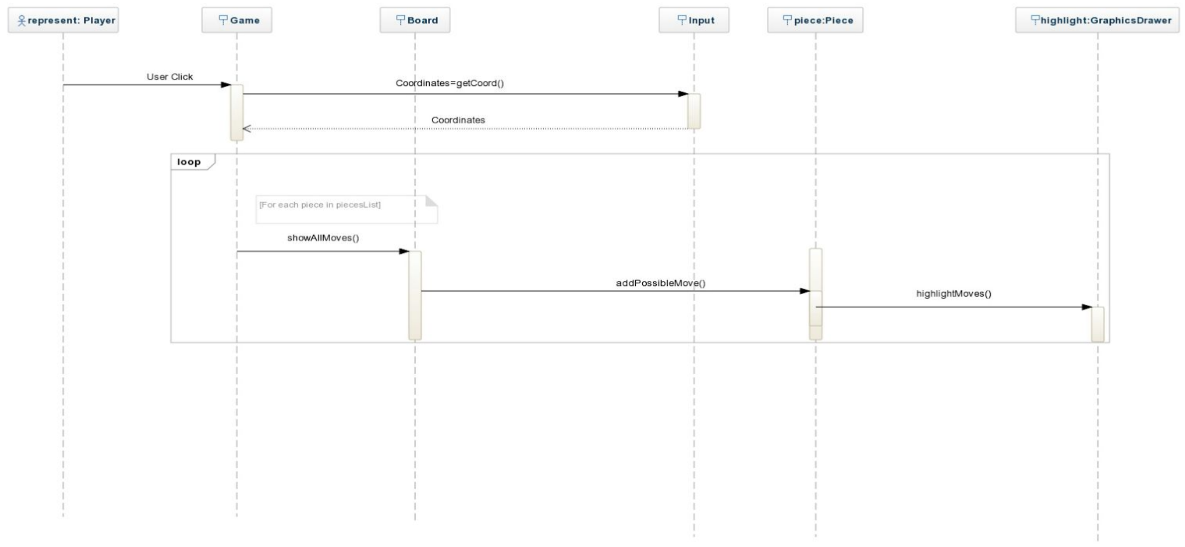
Save Game (Already in Pause Menu):


represent: Player
gameCanvas:GraphicsDrawer
:Input
:Database
while in pause menu, player clicks on "save game" button
coord = getCoord()
return clickCoordinates
notification = saveGame(piecesGrid, playerOneTurn, playerTwoScore, playerOneScore, time)
notification
displayMessage(notification)

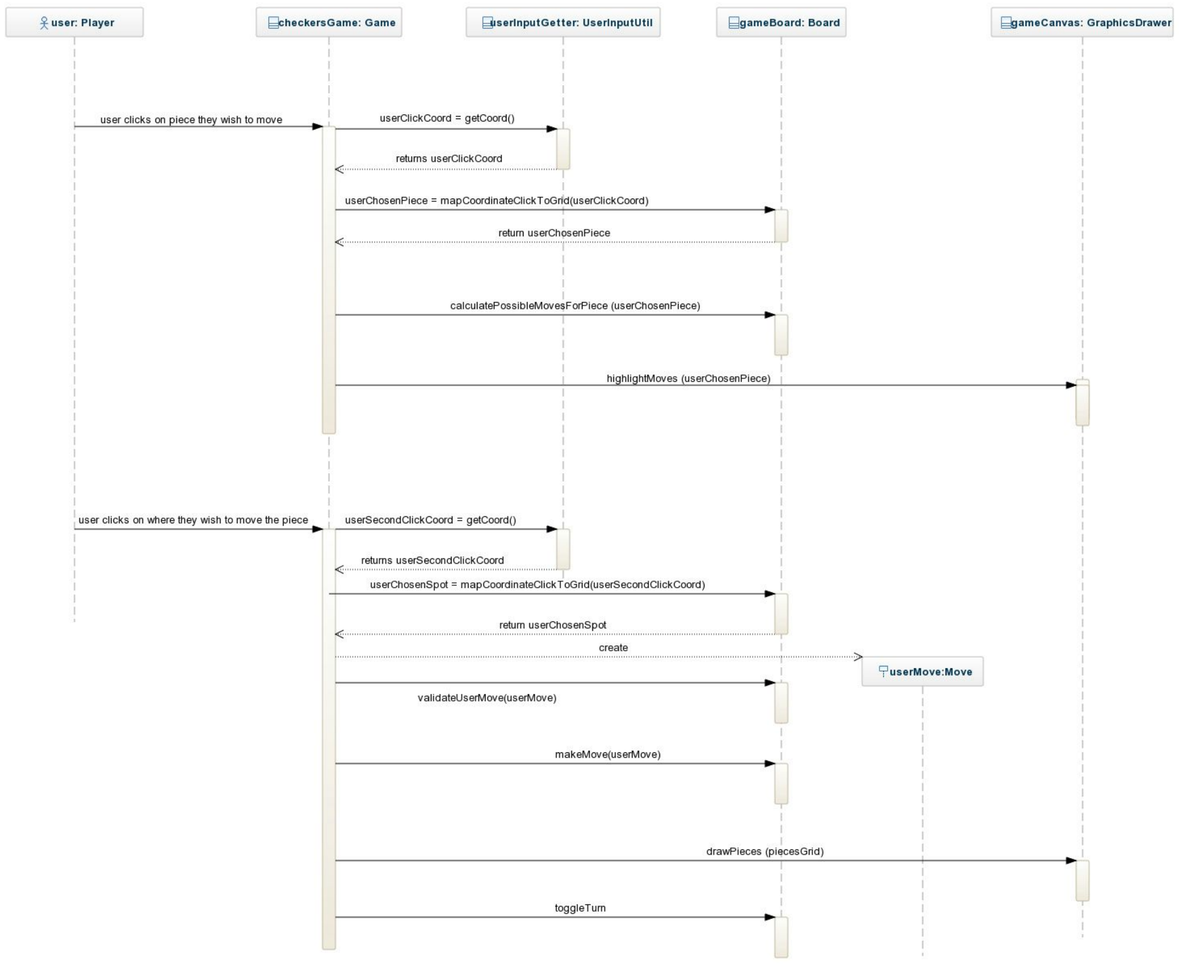
Go to Main Menu (Already in Pause Menu):



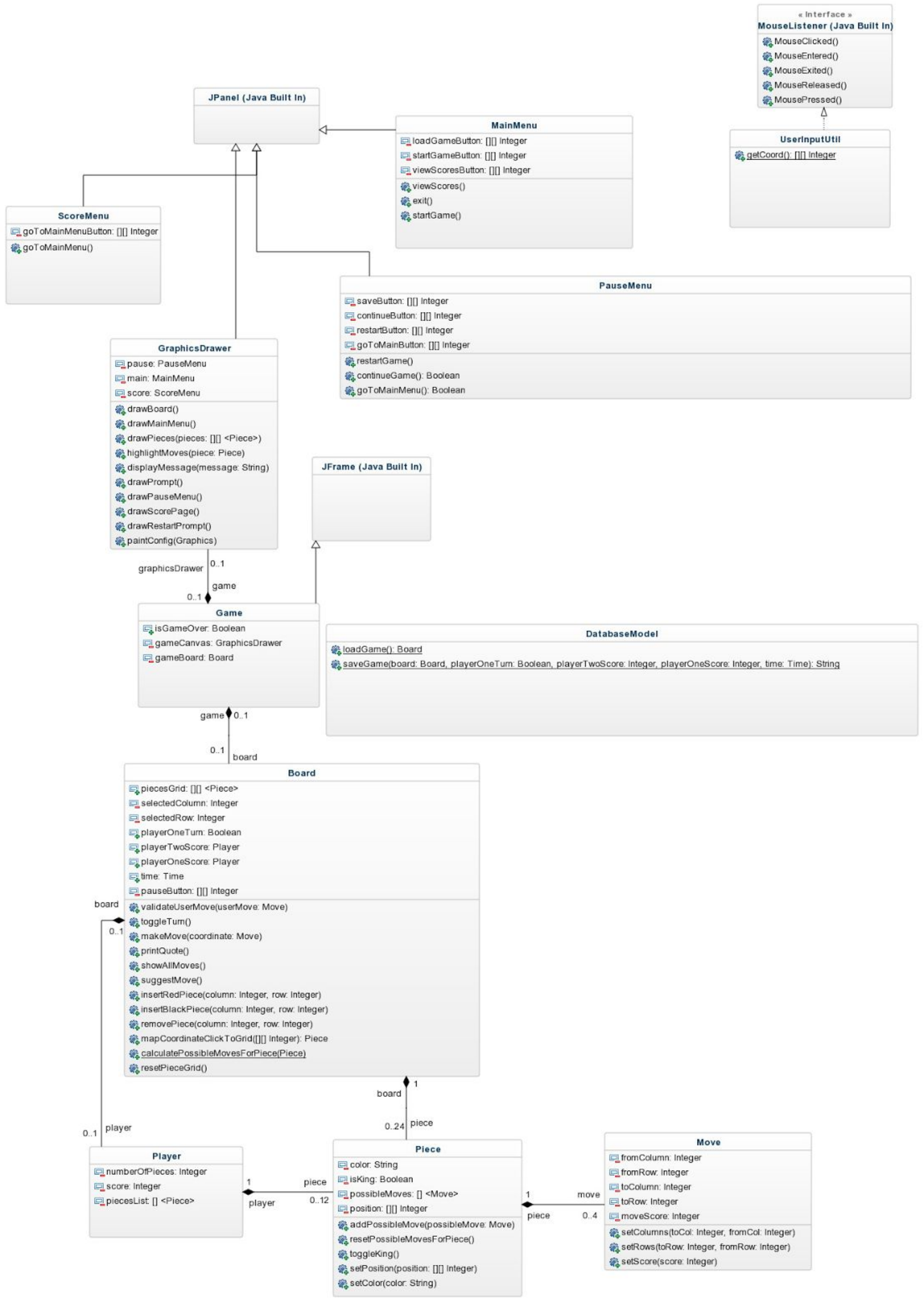
Jacob Brauchler Diagram:



Alex Tzinov Diagram: **(Case_id: UC-09, req_id: US-01. how user performs a move on the board**


user: Player
checkersGame: Game
userInputGetter: UserInputUtil
gameBoard: Board
gameCanvas: GraphicsDrawer
user clicks on piece they wish to move
userClickCoord = getCoord()
returns userClickCoord
userChosenPiece = mapCoordinateClickToGrid(userClickCoord)
return userChosenPiece
calculatePossibleMovesForPiece (userChosenPiece)
highlightMoves (userChosenPiece)
user clicks on where they wish to move the piece
userSecondClickCoord = getCoord()
returns userSecondClickCoord
userChosenSpot = mapCoordinateClickToGrid(userSecondClickCoord)
return userChosenSpot
create
userMove:Move
validateUserMove(userMove)
makeMove(userMove)
drawPieces (piecesGrid)
toggleTurn

Class Diagram:


« Interface »
MouseListener (Java Built In)
MouseClicked()
MouseEntered()
MouseExited()
MouseReleased()
MousePressed()
UserInputUtil
getCoord(): [][] Integer
JPanel (Java Built In)
MainMenu
loadGameButton: [][] Integer
startGameButton: [][] Integer
viewScoresButton: [][] Integer
viewScores()
exit()
startGame()
ScoreMenu
goToMainMenuButton: [][] Integer
goToMainMenu()
PauseMenu
saveButton: [][] Integer
continueButton: [][] Integer
restartButton: [][] Integer
goToMainButton: [][] Integer
restartGame()
continueGame(): Boolean
goToMainMenu(): Boolean
GraphicsDrawer
pause: PauseMenu
main: MainMenu
score: ScoreMenu
drawBoard()
drawMainMenu()
drawPieces(pieces: [][] <Piece>)
highlightMoves(piece: Piece)
displayMessage(message: String)
drawPrompt()
drawPauseMenu()
drawScorePage()
drawRestartPrompt()
paintConfig(Graphics)
JFrame (Java Built In)
graphicsDrawer
0..1
game
0..1
Game
isGameOver: Boolean
gameCanvas: GraphicsDrawer
gameBoard: Board
DatabaseModel
loadGame(): Board
saveGame(board: Board, playerOneTurn: Boolean, playerTwoScore: Integer, playerOneScore: Integer, time: Time): String
game
0..1
0..1
board
Board
piecesGrid: [][] <Piece>
selectedColumn: Integer
selectedRow: Integer
playerOneTurn: Boolean
playerTwoScore: Player
playerOneScore: Player
time: Time
pauseButton: [][] Integer
validateUserMove(userMove: Move)
toggleTurn()
makeMove(coordinate: Move)
printQuote()
showAllMoves()
suggestMove()
insertRedPiece(column: Integer, row: Integer)
insertBlackPiece(column: Integer, row: Integer)
removePiece(column: Integer, row: Integer)
mapCoordinateClickToGrid([][] Integer): Piece
calculatePossibleMovesForPiece(Piece)
resetPieceGrid()
board
0..1
board
1
0..24
piece
0..1
player
Player
numberOfPieces: Integer
score: Integer
piecesList: [] <Piece>
1
player
piece
0..12
Piece
color: String
isKing: Boolean
possibleMoves: [] <Move>
position: [][] Integer
addPossibleMove(possibleMove: Move)
resetPossibleMovesForPiece()
toggleKing()
setPosition(position: [][] Integer)
setColor(color: String)
1
piece
move
0..4
Move
fromColumn: Integer
fromRow: Integer
toColumn: Integer
toRow: Integer
moveScore: Integer
setColumns(toCol: Integer, fromCol: Integer)
setRows(toRow: Integer, fromRow: Integer)
setScore(score: Integer)