Sprint 3
Rollerball

Blueberries!

COLORADO STATE UNIVERSITY

Department of Computer Science

Jared Ham, Alex Enthoven, Daniel Chavez,
Sharon Zhu, Robbie Weinel

Rollerball Refresher

- 7x7 square grid with middle 3x3 grid missing
- 4 Unique Pieces
  - 2x Rook
  - 2x Pawn
  - 1x Bishop
  - 1x King
- White always moves first
- Two Ways to Win
  - Checkmate Enemy King
  - Move king to enemy king starting location through clockwise movement of king

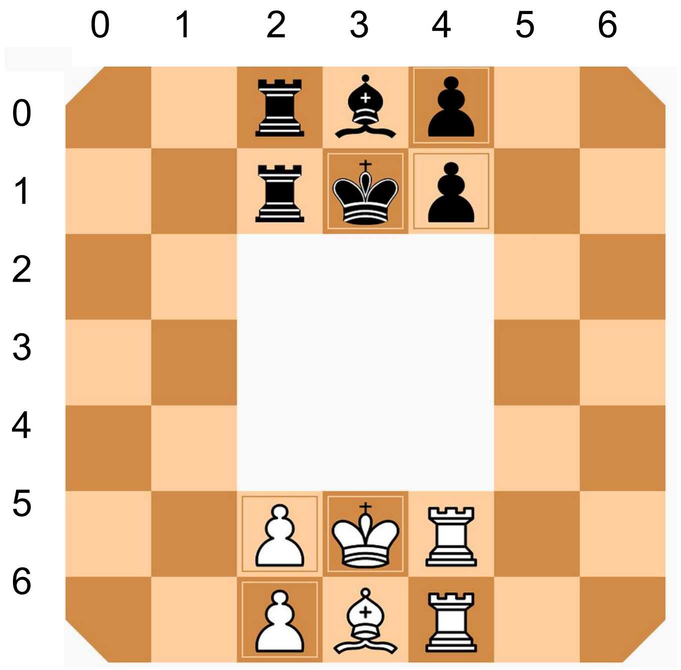


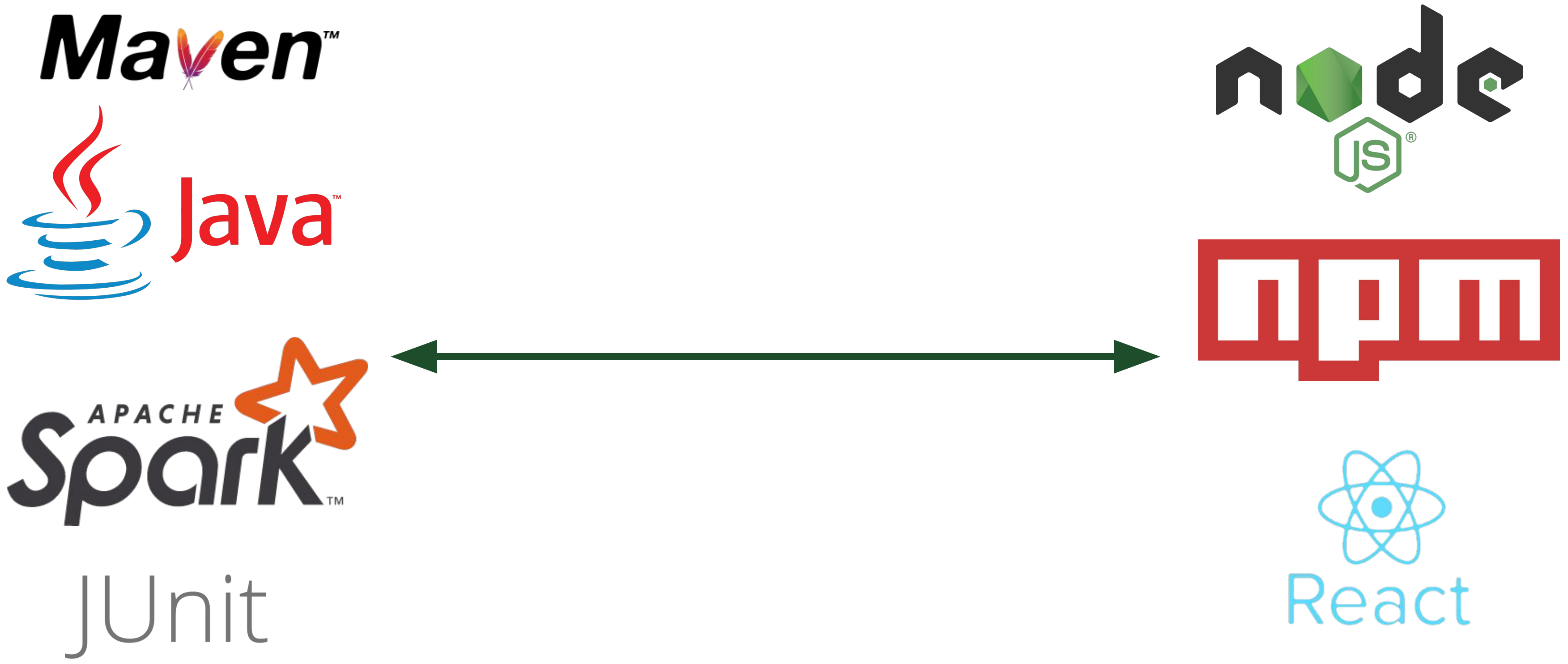
Figure 1: Rollerball Board
https://en.wikipedia.org/wiki/Rollerball_(chess_variant)

Process/Design Decisions

COLORADO STATE UNIVERSITY

Department of Computer Science

Project Structure (Technology)

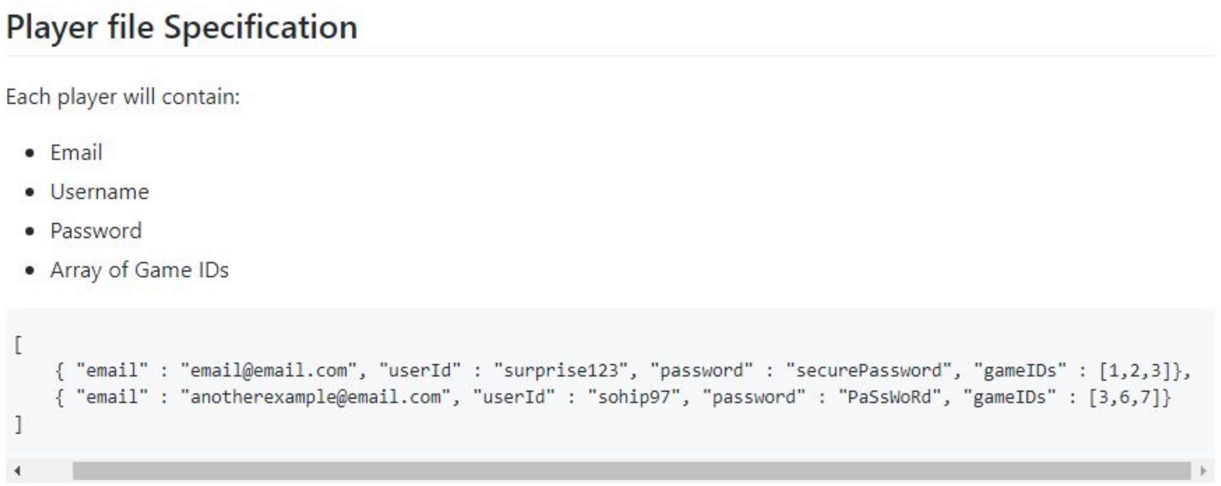


How to Handle Networking?

- **Client Server** or Peer to Peer
- Where to Host the server?
  - Hosted at team members house.
  - Router is set up to forward the client and server ports
  - No domain name because well… **we're cheap**
- Some network related downfalls to our program:
  - The board does not automatically update for the non moving player
  - This set up would not scale
  - No locking systems in place to prevent data corruption

How to Store Data?

- JSON Files saved on server over a database for simplicity
  - Save a player file and a game file
- Would need to convert to database for scalability

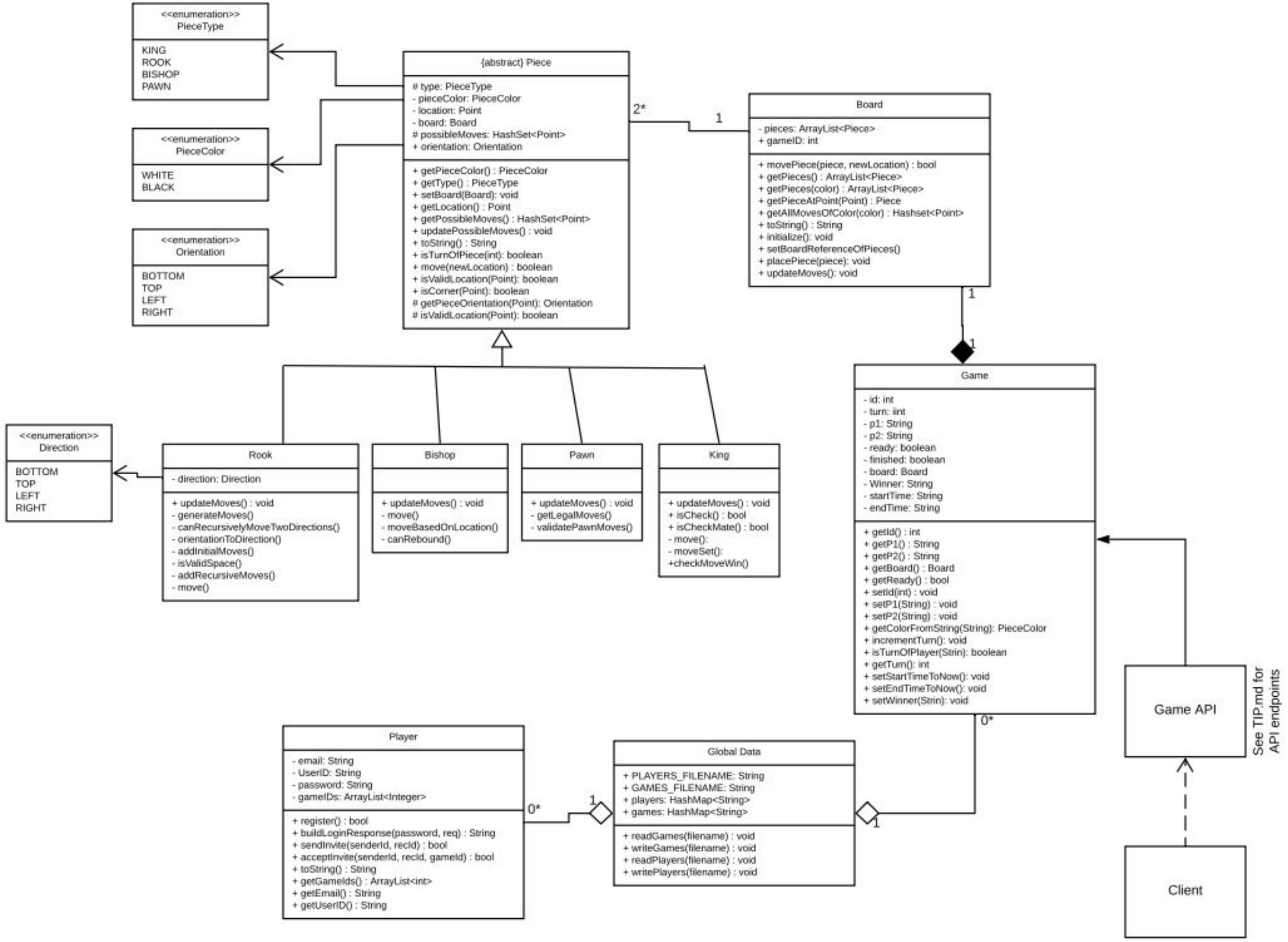


Player file Specification

Each player will contain:

- Email
- Username
- Password
- Array of Game IDs

```
[
    { "email" : "email@email.com", "userId" : "surprise123", "password" : "securePassword", "gameIDs" : [1,2,3]},
    { "email" : "anotherexample@email.com", "userId" : "sohip97", "password" : "PaSsWoRd", "gameIDs" : [3,6,7]}
]
```


<<enumeration>>
PieceType
KING
ROOK
BISHOP
PAWN
<<enumeration>>
PieceColor
WHITE
BLACK
<<enumeration>>
Orientation
BOTTOM
TOP
LEFT
RIGHT
{abstract} Piece
# type: PieceType
- pieceColor: PieceColor
- location: Point
- board: Board
# possibleMoves: HashSet<Point>
+ orientation: Orientation
+ getPieceColor() : PieceColor
+ getType() : PieceType
+ setBoard(Board): void
+ getLocation() : Point
+ getPossibleMoves() : HashSet<Point>
+ updatePossibleMoves() : void
+ toString() : String
+ isTurnOfPiece(int): boolean
+ move(newLocation) : boolean
+ isValidLocation(Point): boolean
+ isCorner(Point): boolean
# getPieceOrientation(Point): Orientation
# isValidLocation(Point): boolean
2*
1
Board
- pieces: ArrayList<Piece>
+ gameID: int
+ movePiece(piece, newLocation) : bool
+ getPieces() : ArrayList<Piece>
+ getPieces(color) : ArrayList<Piece>
+ getPieceAtPoint(Point) : Piece
+ getAllMovesOfColor(color) : Hashset<Point>
+ toString() : String
+ initialize(): void
+ setBoardReferenceOfPieces()
+ placePiece(piece): void
+ updateMoves(): void
1
1
<<enumeration>>
Direction
BOTTOM
TOP
LEFT
RIGHT
Rook
- direction: Direction
+ updateMoves() : void
- generateMoves()
- canRecursivelyMoveTwoDirections()
- orientationToDirection()
- addInitialMoves()
- isValidSpace()
- addRecursiveMoves()
- move()
Bishop
+ updateMoves() : void
- move()
- moveBasedOnLocation()
- canRebound()
Pawn
+ updateMoves() : void
- getLegalMoves()
- validatePawnMoves()
King
+ updateMoves() : void
+ isCheck() : bool
+ isCheckMate() : bool
- move():
- moveSet():
+checkMoveWin()
Game
- id: int
- turn: iint
- p1: String
- p2: String
- ready: boolean
- finished: boolean
- board: Board
- Winner: String
- startTime: String
- endTime: String
+ getId() : int
+ getP1() : String
+ getP2() : String
+ getBoard() : Board
+ getReady() : bool
+ setId(int) : void
+ setP1(String) : void
+ setP2(String) : void
+ getColorFromString(String): PieceColor
+ incrementTurn(): void
+ isTurnOfPlayer(Strin): boolean
+ getTurn(): int
+ setStartTimeToNow(): void
+ setEndTimeToNow(): void
+ setWinner(Strin): void
0*
Game API
See TIP.md for API endpoints
Client
Player
- email: String
- UserID: String
- password: String
- gameIDs: ArrayList<Integer>
+ register() : bool
+ buildLoginResponse(password, req) : String
+ sendInvite(senderId, recId) : bool
+ acceptInvite(senderId, recId, gameId) : bool
+ toString() : String
+ getGameIds() : ArrayList<int>
+ getEmail() : String
+ getUserID() : String
0*
1
Global Data
+ PLAYERS_FILENAME: String
+ GAMES_FILENAME: String
+ players: HashMap<String>
+ games: HashMap<String>
+ readGames(filename) : void
+ writeGames(filename) : void
+ readPlayers(filename) : void
+ writePlayers(filename) : void
1

Design Pattern - Proxy

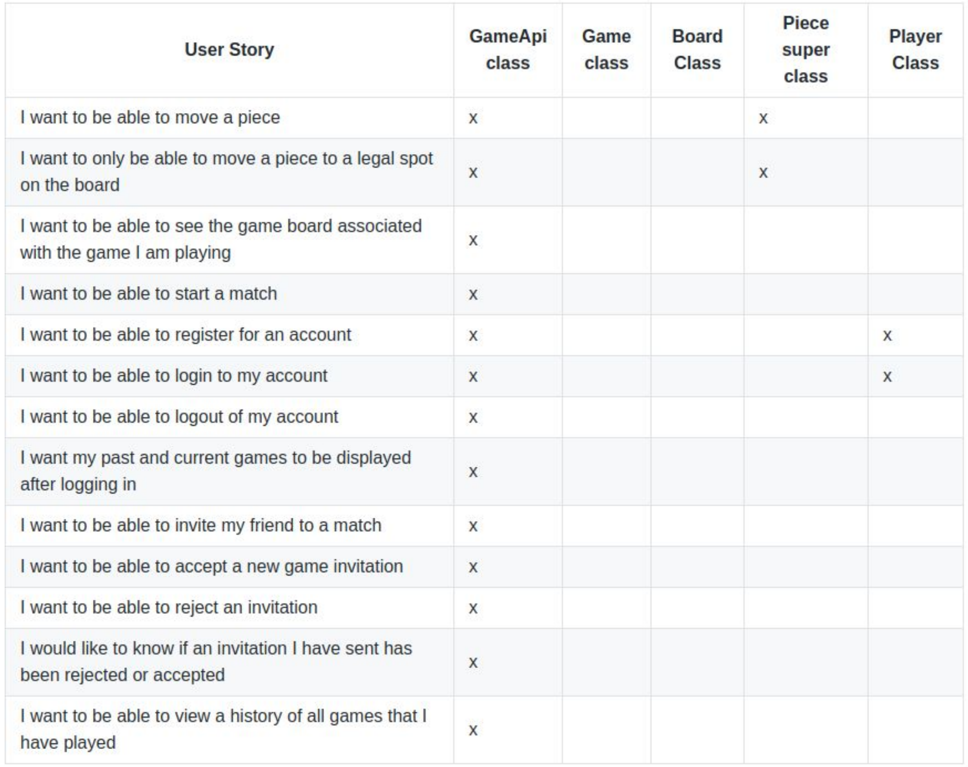
- GameApi provides access to the Rollerball functionality through the following Api endpoints:
  - Register
  - Login
  - Unregister
  - SendInvite
  - AcceptInvite
  - Move
- Obfuscates move logic from front-end development

Design Pattern - Factory Method

- Frontend code decides which piece image to show a user based off of the supplied string
- White and black pieces have separate images for each of the 4 pieces in the game
    - WhitePiece
        - If (type.equals(king, rook, bishop, pawn) …
    - BlackPiece
        - If (type.equals(king, rook, bishop, pawn) …

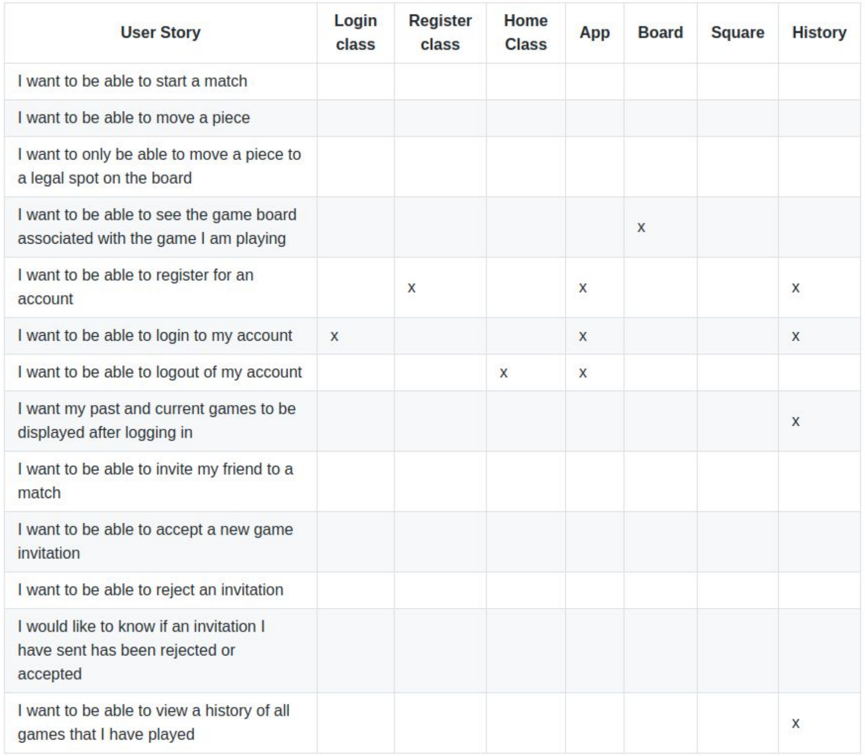
Traceability Link Matrix - Server Side

- GameApi is involved in every user story since every user story involves an interaction between client and server.



User Story	GameApi class	Game class	Board Class	Piece super class	Player Class
I want to be able to move a piece	x			x	
I want to only be able to move a piece to a legal spot on the board	x			x	
I want to be able to see the game board associated with the game I am playing	x				
I want to be able to start a match	x				
I want to be able to register for an account	x				x
I want to be able to login to my account	x				x
I want to be able to logout of my account	x				
I want my past and current games to be displayed after logging in	x				
I want to be able to invite my friend to a match	x				
I want to be able to accept a new game invitation	x				
I want to be able to reject an invitation	x				
I would like to know if an invitation I have sent has been rejected or accepted	x				
I want to be able to view a history of all games that I have played	x				

Traceability Link Matrix - Client Side



User Story	Login class	Register class	Home Class	App	Board	Square	History
I want to be able to start a match							
I want to be able to move a piece							
I want to only be able to move a piece to a legal spot on the board							
I want to be able to see the game board associated with the game I am playing					x		
I want to be able to register for an account		x		x			x
I want to be able to login to my account	x			x			x
I want to be able to logout of my account			x	x			
I want my past and current games to be displayed after logging in							x
I want to be able to invite my friend to a match							
I want to be able to accept a new game invitation							
I want to be able to reject an invitation							
I would like to know if an invitation I have sent has been rejected or accepted							
I want to be able to view a history of all games that I have played							x

Demo Time

Demo

- Attempt to register new user with an invalid email address
- Register new user
- Demonstrate login with wrong password
- Login with correct password
- Unregister User
- Attempt login with unregistered users account info
- Create 2 new users and login
    - User 1:
    - User 2:

Demo

- User 1 invite User 2 to a new game
- Accept invite from User 1 home page
- Demonstrate that when it is not your turn you cannot play
- Demonstrate you cannot move your opponent's piece
- Demonstrate a few moves, (Bounce off walls, not being able to move king into check)
- Demonstrate a checkmate
- Game is over so show the game history