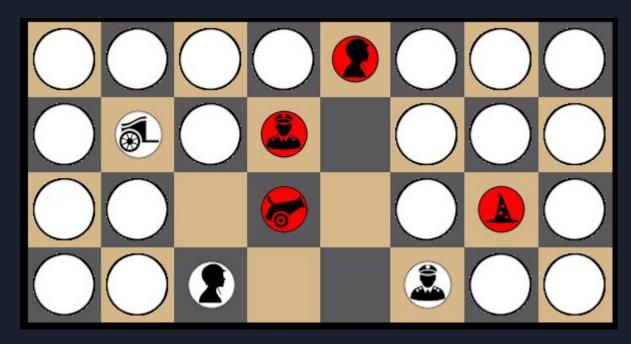# Hackers Anonymous

Sprint 3 :
Evan Lavertu, Cody Coleman, Kareem Youssef, Patrick Keleher, Jack Fitzgerald

Banqi

- Banqi is a variant of the popular Chinese game called Xiangqi.
- It is played on half of a Xiangqi board
- All of the pieces are flipped over in the beginning of the game

Pieces per team:
- x1 General
- x2 Chariot
- x2 Horse
- x2 Cannon
- x2 Advisor
- x2 Elephant
- x5 Soldier

Development process

- We used the Vue JavaScript Framework for our frontend and Java for our backend.

- As a team we would meet once a week to discuss what we each were working on and then work together as a team.

- We assigned tasks by what each team member felt was their strengths.
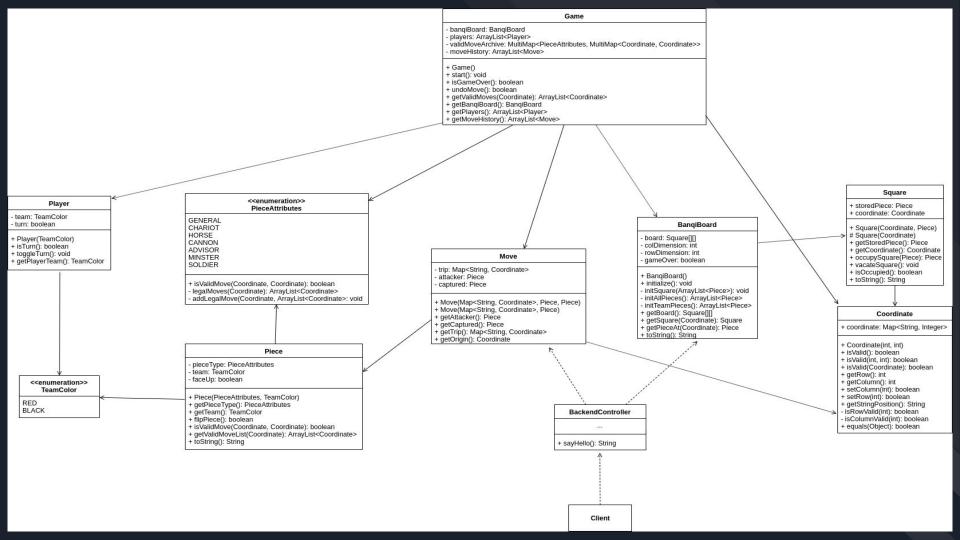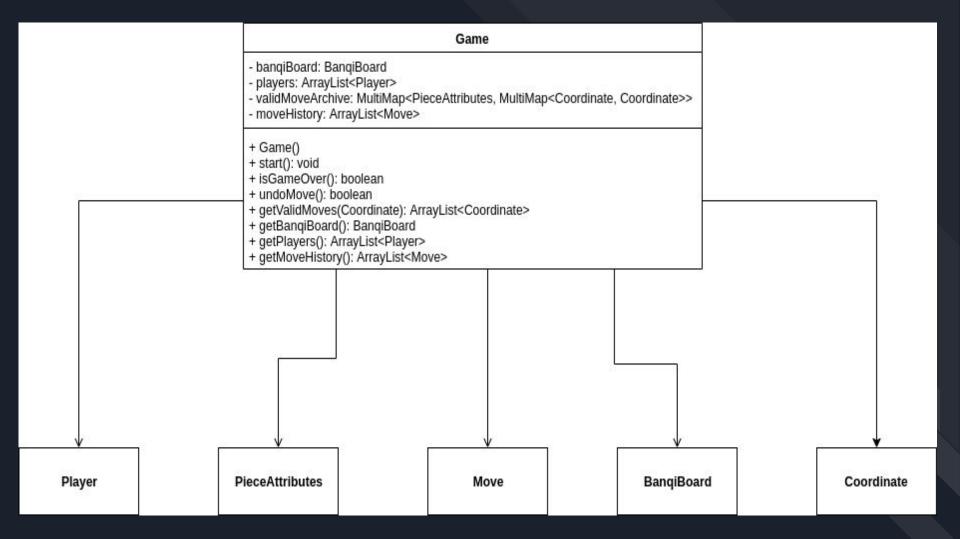
Design Decisions

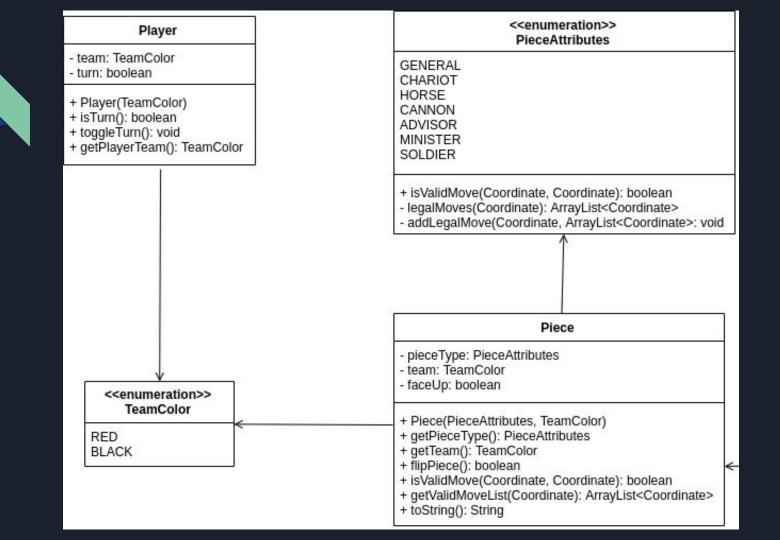- We hosted our database on Microsoft Azure for ease of access to both clients.

- Using classes based on the Adaptor design pattern we configured our backend to send information from our client to the database.

- For the pieces we used aspects of the Decorator design pattern as what the backend will return changes based on the input of the client.
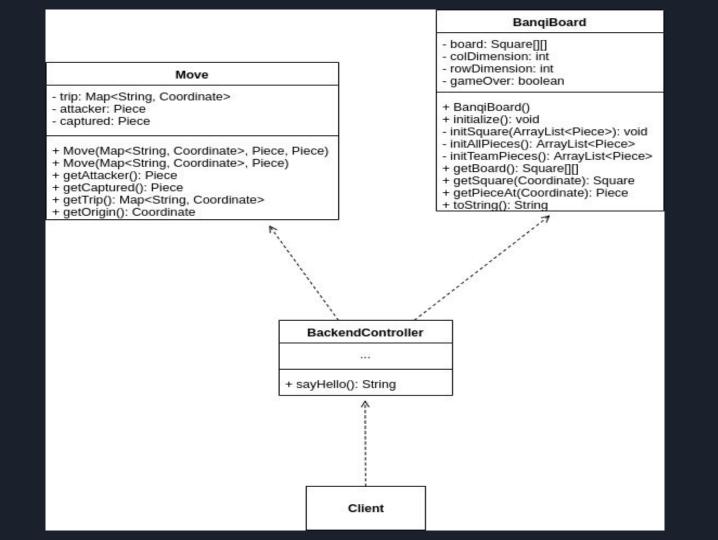
Class Diagrams

Game

- banqiBoard: BanqiBoard
- players: ArrayList<Player>
- validMoveArchive: MultiMap<PieceAttributes, MultiMap<Coordinate, Coordinate>>
- moveHistory: ArrayList<Move>

+ Game()
+ start(): void
+ isGameOver(): boolean
+ undoMove(): boolean
+ getValidMoves(Coordinate): ArrayList<Coordinate>
+ getBanqiBoard(): BanqiBoard
+ getPlayers(): ArrayList<Player>
+ getMoveHistory(): ArrayList<Move>

Player

- team: TeamColor
- turn: boolean

+ Player(TeamColor)
+ isTurn(): boolean
+ toggleTurn(): void
+ getPlayerTeam(): TeamColor

<<enumeration>> PieceAttributes

GENERAL
CHARIOT
HORSE
CANNON
ADVISOR
MINSTER
SOLDIER

+ isValidMove(Coordinate, Coordinate): boolean
- legalMoves(Coordinate): ArrayList<Coordinate>
- addLegalMove(Coordinate, ArrayList<Coordinate>: void

Move

- trip: Map<String, Coordinate>
- attacker: Piece
- captured: Piece

+ Move(Map<String, Coordinate>, Piece, Piece)
+ Move(Map<String, Coordinate>, Piece)
+ getAttacker(): Piece
+ getCaptured(): Piece
+ getTrip(): Map<String, Coordinate>
+ getOrigin(): Coordinate

BanqiBoard

- board: Square[][]
- colDimension: int
- rowDimension: int
- gameOver: boolean

+ BanqiBoard()
+ initialize(): void
- initSquare(ArrayList<Piece>): void
- initAllPieces(): ArrayList<Piece>
- initTeamPieces(): ArrayList<Piece>
+ getBoard(): Square[][]
+ getSquare(Coordinate): Square
+ getPieceAt(Coordinate): Piece
+ toString(): String

Square

+ storedPiece: Piece
+ coordinate: Coordinate

+ Square(Coordinate, Piece)
# Square(Coordinate)
+ getStoredPiece(): Piece
+ getCoordinate(): Coordinate
+ occupySquare(Piece): Piece
+ vacateSquare(): void
+ isOccupied(): boolean
+ toString(): String

Coordinate

+ coordinate: Map<String, Integer>

+ Coordinate(int, int)
+ isValid(): boolean
+ isValid(int, int): boolean
+ isValid(Coordinate): boolean
+ getRow(): int
+ getColumn(): int
+ setColumn(int): boolean
+ setRow(int): boolean
+ getStringPosition(): String
- isRowValid(int): boolean
- isColumnValid(int): boolean
+ equals(Object): boolean

<<enumeration>> TeamColor

RED
BLACK

Piece

- pieceType: PieceAttributes
- team: TeamColor
- faceUp: boolean

+ Piece(PieceAttributes, TeamColor)
+ getPieceType(): PieceAttributes
+ getTeam(): TeamColor
+ flipPiece(): boolean
+ isValidMove(Coordinate, Coordinate): boolean
+ getValidMoveList(Coordinate): ArrayList<Coordinate>
+ toString(): String

BackendController

...

+ sayHello(): String

Client

Game

- banqiBoard: BanqiBoard
- players: ArrayList<Player>
- validMoveArchive: MultiMap<PieceAttributes, MultiMap<Coordinate, Coordinate>>
- moveHistory: ArrayList<Move>

+ Game()
+ start(): void
+ isGameOver(): boolean
+ undoMove(): boolean
+ getValidMoves(Coordinate): ArrayList<Coordinate>
+ getBanqiBoard(): BanqiBoard
+ getPlayers(): ArrayList<Player>
+ getMoveHistory(): ArrayList<Move>

Player

PieceAttributes

Move

BanqiBoard

Coordinate

Player

- team: TeamColor
- turn: boolean

+ Player(TeamColor)
+ isTurn(): boolean
+ toggleTurn(): void
+ getPlayerTeam(): TeamColor

<<enumeration>> PieceAttributes

GENERAL
CHARIOT
HORSE
CANNON
ADVISOR
MINISTER
SOLDIER

+ isValidMove(Coordinate, Coordinate): boolean
- legalMoves(Coordinate): ArrayList<Coordinate>
- addLegalMove(Coordinate, ArrayList<Coordinate>: void

<<enumeration>> TeamColor

RED
BLACK

Piece

- pieceType: PieceAttributes
- team: TeamColor
- faceUp: boolean

+ Piece(PieceAttributes, TeamColor)
+ getPieceType(): PieceAttributes
+ getTeam(): TeamColor
+ flipPiece(): boolean
+ isValidMove(Coordinate, Coordinate): boolean
+ getValidMoveList(Coordinate): ArrayList<Coordinate>
+ toString(): String

BanqiBoard

- board: Square[][]
- colDimension: int
- rowDimension: int
- gameOver: boolean

+ BanqiBoard()
+ initialize(): void
- initSquare(ArrayList<Piece>): void
- initAllPieces(): ArrayList<Piece>
- initTeamPieces(): ArrayList<Piece>
+ getBoard(): Square[][]
+ getSquare(Coordinate): Square
+ getPieceAt(Coordinate): Piece
+ toString(): String

Square

+ storedPiece: Piece
+ coordinate: Coordinate

+ Square(Coordinate, Piece)
# Square(Coordinate)
+ getStoredPiece(): Piece
+ getCoordinate(): Coordinate
+ occupySquare(Piece): Piece
+ vacateSquare(): void
+ isOccupied(): boolean
+ toString(): String

Coordinate

+ coordinate: Map<String, Integer>

+ Coordinate(int, int)
+ isValid(): boolean
+ isValid(int, int): boolean
+ isValid(Coordinate): boolean
+ getRow(): int
+ getColumn(): int
+ setColumn(int): boolean
+ setRow(int): boolean
+ getStringPosition(): String
- isRowValid(int): boolean
- isColumnValid(int): boolean
+ equals(Object): boolean

BanqiBoard

- board: Square[][]
- colDimension: int
- rowDimension: int
- gameOver: boolean

+ BanqiBoard()
+ initialize(): void
- initSquare(ArrayList<Piece>): void
- initAllPieces(): ArrayList<Piece>
- initTeamPieces(): ArrayList<Piece>
+ getBoard(): Square[][]
+ getSquare(Coordinate): Square
+ getPieceAt(Coordinate): Piece
+ toString(): String

Move

- trip: Map<String, Coordinate>
- attacker: Piece
- captured: Piece

+ Move(Map<String, Coordinate>, Piece, Piece)
+ Move(Map<String, Coordinate>, Piece)
+ getAttacker(): Piece
+ getCaptured(): Piece
+ getTrip(): Map<String, Coordinate>
+ getOrigin(): Coordinate

BackendController

...

+ sayHello(): String

Client

Traceability link matrix

	GameService	UserService	BackendController	GameFunctions	MoveFunctions	UserDAO	GameDAO	Board	User	Coordinate	Game	Move	Piece	Square
US-2: I would like to play more than one game at a time														
US-3: I would like to leave games at any time	X													
US-4: I would like to create a match														
US-5: I would like to be able to receive invitations to games	X													
US-6: I would like to be able to send more than one invitation	X													
US-7: I would like to be able to register to the platform		X												
US-8: I would like to be able to unregister from the platform														
US-9: I would like pieces to disappear when they are captured				X	X		X							
US-10: I would like to be able to play Banqi online														
US-14: I would like to know if I won or lost the game														

Traceability link matrix continued

	GameService	UserService	BackendController	GameFunctions	MoveFunctions	UserDAO	GameDAO	Board	User	Coordinate	Game	Move	Piece	Square
US-15: I would like to know when a game is over	X													
US-16: I would like to see a record of all the games I've played														
US-17: I would like to be able to save my current game state	X		X											
US-18: I would like to be able to reload saved games	X		X											
US-26: I woud like to be able to flip a piece over for my turn	X		X					X		X	X	X	X	X
US-27: I would like to be able to move my pieces	X		X					X		X	X	X	X	X
US-28: I would like to be able to capture enemy pieces	X		X					X		X	X	X	X	X
US-36: I would like to be able to login to my account that I registered with		X	X			X			X					
US-40: I would like to be able to view my profile for the account I created		X	X						X					
US-42: I would like to be able to play Banqi with friends.	X		X											
US-43: I would like to be able to come back to my game	X		X											
US-44: I would like to be able to view my game statistics on my profile page	X		X											

Lessons learned

- Working as a team to build a large project from scratch is very different and much more complex than individual projects.

- Meeting with the product owner regularly helped us know what to prioritize during our sprints.

- We now have a better understanding of object oriented design.

Demo