Explanation of architecture:

The game server uses the ASP.NET Core platform (the C# equivalent of Swing for Java) and can run on Windows, Linux, and MacOS. The game client uses unity and can be build on the mentioned platforms as a desktop application, and it can also be built using WebGL to run in a web browser application served by the game server.

The server and clients communicate through web sockets. Upon receival of a web socket request, the server's and client's message handlers will parse the event name and data and call the appropriate event handler. The result is an event driven approach

The architecture of the game uses the Model View Controller (MVC) principle to handle the decoupling of the game logic and the Unity UI. The server game controller registers different events, manipulates the quest game and players models, and sends the required updates to the view to the Unity game client. The sole responsibility of the client is to update the view when the server sends out any events, and to notify the server when a user has interacted with the UI. In this implementation of MVC, the server's game controller's event handlers serve as the controller, the game logic that the game controller manipulates serves as the model, and the Unity client serves as the view.

The main advantages of the game's architecture is that it is portable, with the server being able to run on most platforms, as well as the client being able to run on most platforms as well as web browsers. The UI and the actual game code are completely decoupled. The server side game logic and the client side UI have no awareness of each other, and the UI can be easily swapped out with any other platform supporting the web socket protocol, provided it respects the contracts for handling events sent to and received from the server.

Class diagrams and interactions:

The game controller class manipulates the game model (quest game class) and updates the view by sending relevant information and prompts to the game client. The game controller acts as a **mediator**, listening to messages from the game client and sending the appropriate message to the game. For example, when the controller receives a discard event from a player, it will call the relevant players discard method in the game model**.**

The server class diagram in the file **class_diagram.png** with CRT cards explains the interactions, patterns, and abstractions used for the server side game logic.

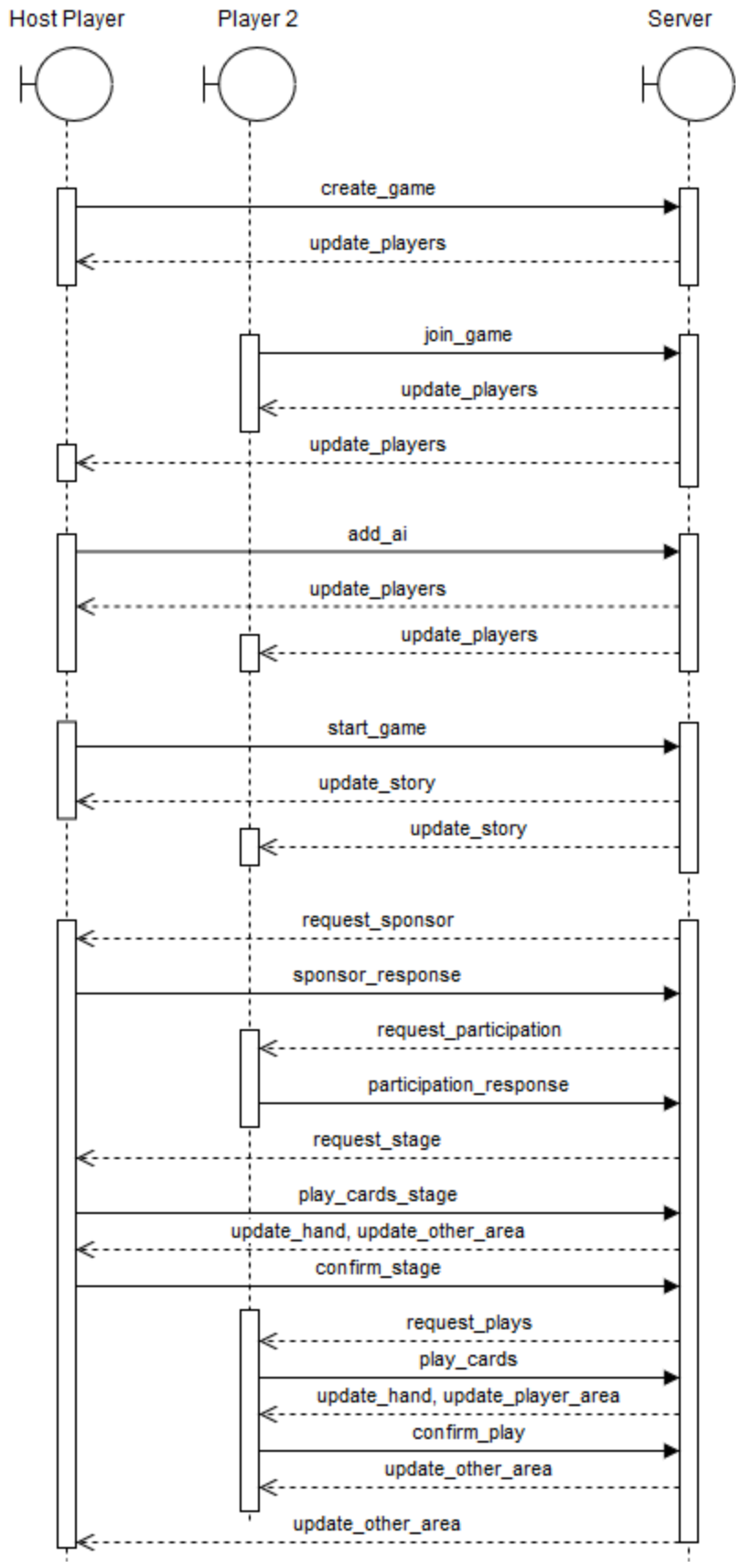
Since the client has very minimal responsibilities, a client class diagram is not included, but the interactions for handling sending and receiving messages is similar to the server side.

Client and server interaction:

There is a contract in place for handling socket messages. The message body must be a UTF8 encoded JSON object with the field “event” containing an event name string and a field “data” containing any arbitrary JSON object. This contract allow for the client and server to register a dictionary of event handlers using an abstraction function .On(“event_name”, JsonObject data) to register handlers.

On the server side each socket is mapped to a player and each player is mapped to a game as well, allowing the event handlers to know the player sending the event. This also allows events to either be broadcasted privately to a single player or to be broadcasted to an entire game.

The following client server interaction diagram explains the process of a host player creating a game, adding an AI, and having another networked player join, and running a single stage quest.


Host Player
Player 2
Server
create_game
update_players
join_game
update_players
update_players
add_ai
update_players
update_players
start_game
update_story
update_story
request_sponsor
sponsor_response
request_participation
participation_response
request_stage
play_cards_stage
update_hand, update_other_area
confirm_stage
request_plays
play_cards
update_hand, update_player_area
confirm_play
update_other_area
update_other_area

Use cases:

A use case diagram can be found in the file use_case_diagram.png

Use Case: UC-1 Create game
Description: Describes the steps of creating a new game room for players to join.
External Actors: player, game system
Precondition: Game system is waiting for requests from player.
Triggering Event: Player requests to host a new game room.
1. Player requests to host a new game.
2. Game system creates a new game lobby and adds the requesting player to it.
3. Player becomes the lobby host and is added to the list of players.
4. Player waits for other players to join the lobby.
Resulting Events: The host of the lobby now waits for other players to join the lobby.
Post Condition: A new game lobby is created.
Alternatives:

-Player selects a pre set game scenario from a list of scenarios and requests to create a game in that state.

-Player is loaded into the pre set game state according to the scenario he selected.

-A Quest match begins in that scenario state. The game continues to follow a "script" until a certain point is reached

---

Use Case: UC-2 Join game
Description: Describes the steps of a human player joining a match.
External Actors: player, game system
Related Use Cases: UC-3 Add AI Player
Precondition: Pre existing game lobby must exist, along with the host/game creator
Triggering Event: player requests to join the match.
1. Player requests to join the match.
2. Game system verifies that adding the human player won't exceed the maximum number of players allowed in a match.
3. Game system adds the requesting player to the list of players in the match.
Resulting Events: A new player is added to the match.
Post Condition: Game system waiting for request from player(s).
Alternatives:

-If adding the AI player exceeds the maximum number of players allowed in a match, the request is denied.

Use Case: UC-3 Add AI Player
Description: Describes the steps of adding an AI Player to the match.

External Actors: player, game system
Related Use Cases: UC-2 Join game
Precondition: Players in lobby, game system waiting for request from player(s).
Triggering Event: player requests to add an AI to the match
1. Player requests to add an AI player to the match
2. Game system asks player which strategy the AI should use. Player selects a strategy.
3. Game system verifies that adding the AI player won't exceed the maximum number of players allowed in a match.
4. Game system adds the corresponding AI player to the list of players in the match.
Resulting events: An AI player is added to the match.
Post condition: Game system waiting for request from player(s).
Alternatives:
-If adding the AI player exceeds the maximum number of players allowed in a match, the request is denied.

Use Case: UC-4 Start game
Description: Describes the steps of starting a new Quest game.
External Actors: players, game system
Precondition: Players in lobby and ready, game system waiting for request from host player.
Triggering Event: Game host requests to start a new Quest match.
1. Host player requests to start a new Quest match.
2. Game system verifies that there is a sufficient number of participating players in the lobby.
3. Game system creates and runs a new game.
4. Game system shuffles the deck of cards and deals each player 12 cards.
5. Host player draws the first story card and game system sets current story to be the drawn card.
Resulting Events: A new match is created and participating players are dealt 12 cards each.
Post Condition: The match begins and players start drawing the story cards.

Use Case: UC-5 Round end
Description: Describes the steps of a player ending their turn.
External Actors: player, game system
Precondition: Quest or tournament has been completed.
Triggering Event: Current player (player that drew the story card) ends his turn.
1. Player ends their turn, resolving the current story card.
2. Game system verifies that the number of cards the player is holding is not over the limit.
3. Game system ends the current players turn. All participating players are rewarded accordingly.
4. Game system verifies that none of the players is of rank Knight Of The Round Table.
5. Game system proceeds to the next player's turn.
Resulting Events: The current quest/tournament is resolved and the next player's turn begins.
Post Condition: The next story card is drawn and current story is updated.

Alternatives:

-If a player ends their turn while their number of cards in hand is over the limit, they will be prompted to play or discard cards.

-If a player is promoted to a Knight Of The Round Table after the quest/tournament, he wins the game.

Use Case: UC-6 Play cards
Description: Describes the steps of a player playing their cards for a quest or tournament phase.
External Actors: player, game system
Related Use Cases: UC-9 Participation response
Precondition: Ongoing quest/tournament. Participants have either just started the quest/tournament or moved on to the next stage.
Triggering Event: Player is deciding on the cards he wants to play for the current phase.
1. Player selects cards from his hand that he wishes to use in the current phase.
2. Game system verifies that the selected cards are currently playable.
3. Game system verifies that the number of cards in the player's hand is not over the limit.
4. Game system uses the selected cards for the current phase. The selected cards are moved from the hand to the board.
5. Player ends turn for the current phase and waits for the rest of the participating players to select their cards.
Resulting Events: The other players select their cards and the current quest/tournament phase is resolved. Winning players are rewarded accordingly.
Post Condition: The next quest/tournament phase begins and winning players select which cards to play for that phase.
Alternatives:

-If a player ends their turn while their number of cards in hand is over the limit, they will be prompted to play or discard cards until under the limit.

Use Case: UC-7 Play cards stage
Description: Describes the steps of a quest sponsoring player selecting the cards to use for the quest stages.
External Actors: Sponsor player, game system
Related Use cases: UC-10 Quest sponsor response
Precondition: A story card has just been resolved, the next story card has not yet been drawn.
Triggering Event: A Quest card was drawn from the Story deck and a player decides to sponsor it.
1. Sponsor player selects cards from his hand that he wishes to use for setting up a stage of a quest.
2. Game system verifies that the selected cards are usable in the quest stage.

3. Game system uses the selected cards for the current phase. The selected cards are moved from the hand to the board.
4. Game system repeats the last four steps until each stage is properly set up.
5. Game system invites other players to join the quest. Game system starts the quest once all players have responded to the invitation.
Resulting Events: The quest is properly set up. Players may choose to participate or not.
Post Condition: The quest begins.
Alternatives:

-If no one joins the quest, the quest is resolved and the sponsor draws a number of cards equal to the number used to set up the quest.

Use Case: UC-8 Discard
Description: Describes the steps of a player discarding card.
External Actors: Discarding player, game system
Precondition: Player draws one or more cards
Triggering Event: Player's number of cards in hand is over the limit.
1. Player selects cards from his hand to discard.
2. Game system moves the selected cards from hand to discard pile.
3. Game system verifies that the number of cards in the player's hand is not over the limit.
Resulting Events: Player now has 12 or less cards in their hand.
Post Condition: The match resumes.
Alternatives:

-If the player still has over 12 cards after discarding, he will be prompted to discard more cards.

Use Case: UC-9 Participation response
Description: Describes the steps of a player responding to a quest or tournament invitation.
External Actors: player, game system
Related Use Cases: UC-6 Play cards
Precondition: Quest or tournament card has been drawn. Sponsor for quest has been chosen.
Triggering event: Game system sends an invitation to a non-sponsor player.
1. Game system sends an invitation to the player and waits for their response.
2. Player chooses to participate and is dealt an adventure card by the game system.
3. Player waits for the other players to respond to the invitation.
Resulting Events: The tournament begins or the quest sponsor begins setting up the quest.
Post condition: All players have sent a response and the accepting players are added to the list of participants for the current story.
Alternatives:

-The player does not choose to participate and is not dealt any adventure cards.

Use Case: UC-10 Quest sponsor response
Description: Describes the steps of a player responding to a quest sponsor request.
External Actors: player, game system
Related Use Cases: UC-7 Play cards stage
Precondition: Quest card has just been drawn, sponsor has not yet been chosen.
Triggering Event: Game system asks player to sponsor the quest.
1. Game system sends an invitation to the player and waits for their response.
2. Player chooses to sponsor the quest and is set as the quest's sponsor.
3. Other players are notified that the quest has been sponsored.
4. Game system begins requesting quest participation from the other players.
Resulting Events: Other players receive an invitation to join the quest.
Post Condition: Sponsor waits for the other players to respond to the quest invitation.
Alternatives:

-If the player does not choose to sponsor, the next available player receives a request to sponsor.

-If no player sponsors, the quest is abandoned and the next story card is drawn.