Architecture

Contents

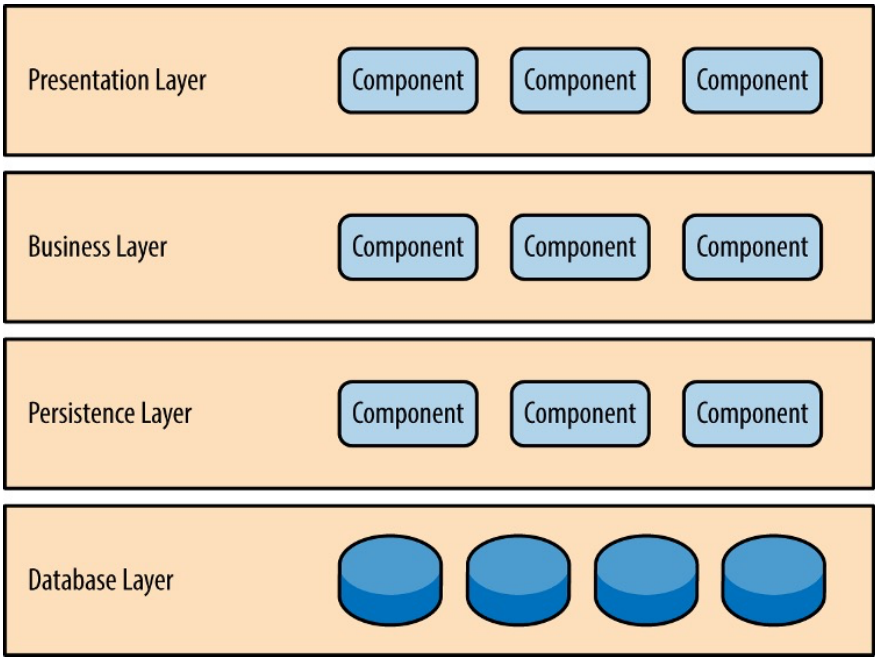
The architecture design must include the following documents:

- The analysis of the whole system
- Component Diagram
- Sequence Diagram
- Activity Diagram
- Class Diagram
- Entity-Relationship (ER) Diagram
- Deployment Diagram
- Technology
- Interface documentation
- High Level Diagram

Other documentations could be supplements to the architecture design.

Architecture Design Draft

Architecture:



Presentation layer: retrieve and display the data for users
Controller: cooperate the business logic and front data
Page:

- Main page (create or join)
- Create a new game (input name)
- Join an existed game(input roomID, name)
- Present the whole story background
- Role responsibility introduction
- Choose role Page
- Each round information
- Each round decision
- Each round outcome
- Final game outcome
- Present all the choices

Service:*? Is it necessary in this project*
Business logic

- User: login, choose role,
- Role: start the round, choose decision, discuss,
- Round: show the infomation, present the decision and choices, set up the timer, calculate decision and generate a final decision,
- Room: generate the final outcome, present all the choices, create a new room, set the room name

Domain model:

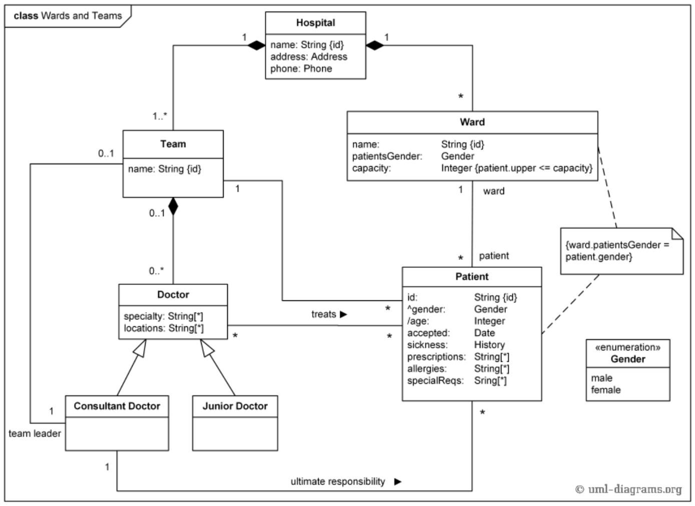
- Story (-> n round -> 5 Role): background(string), outcome
- Round(-> 1 question): roundID, roundBackground, roundOutcome, method , nextRoundID
- Role: roleID(int), roleName (string), roleBackground
- RoleQuestion: roleID, questionID, weight, individualInfo, answer
- Question (-> n Decision): questionID(int), decision, questionText
- Decision: decisionID, decisionText, madeByCharacter
- Room (-> n User) : roomID(string) , playerNum (int), timeLimit
- User: userID, roleID

1. Enter the room,
2. User choose roles,
3. Present story:

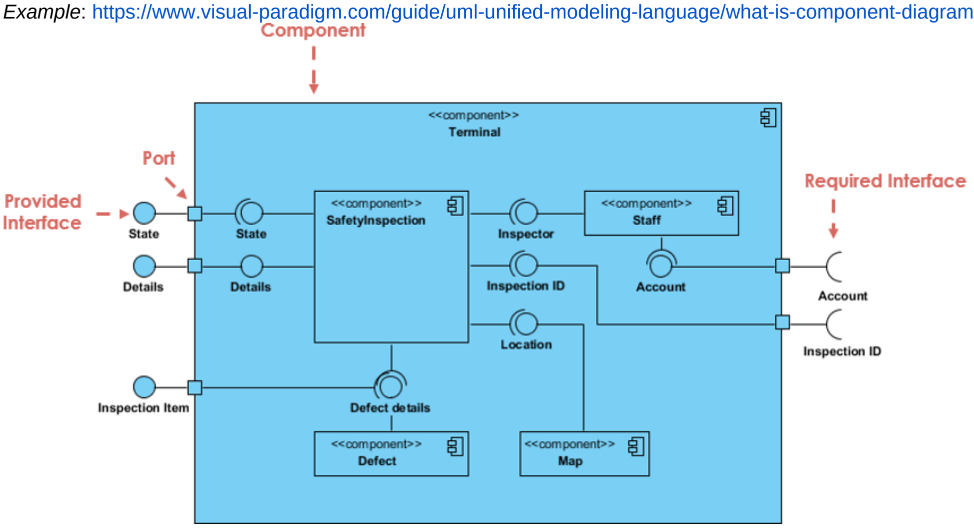
Data source layer: SQL
Database

Diagrams:

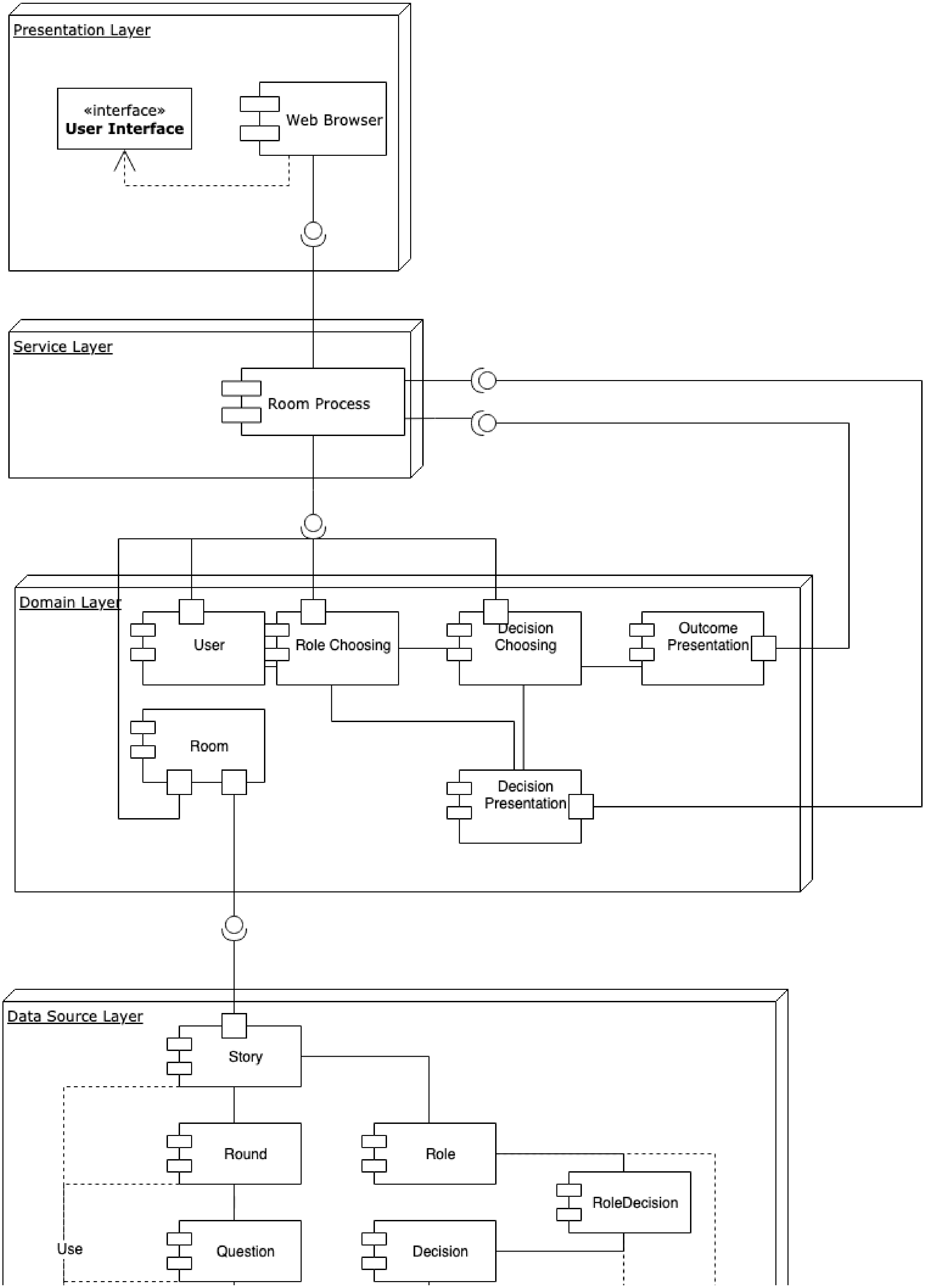
High-level architecture Diagram: Yuhen
Class diagram:
Presentation (controller?) / domain model front (patterns?)
business logic / data source back (patterns?)
ER diagram (**database**): backend : Lu (patterns?)
Domain diagram: class diagram without method-->front
Example: https://www.uml-diagrams.org/examples/hospital-domain-diagram.html

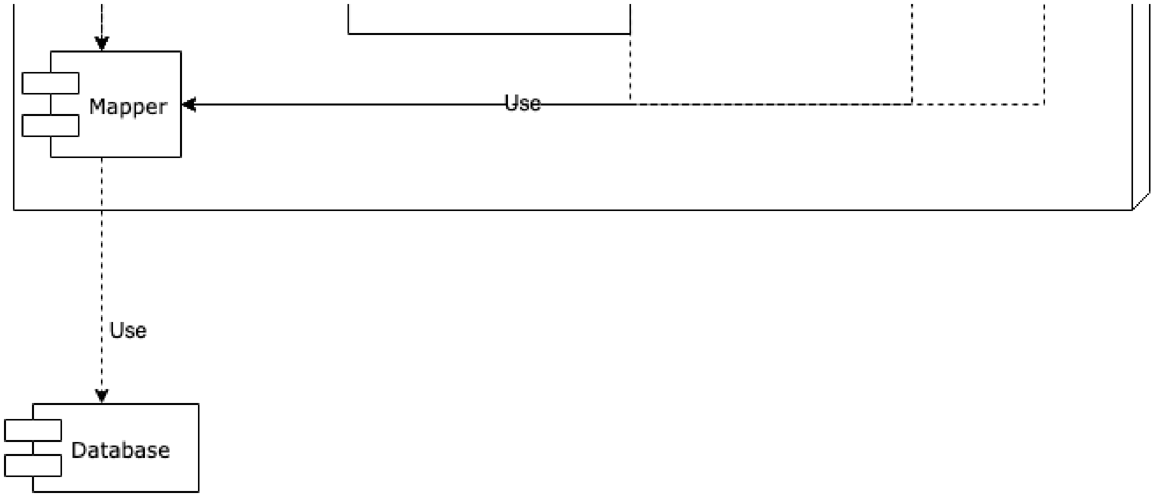


Activity diagram/sequence diagram:--> user stories (for the whole team)
Component diagram: Chenling
Example: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-component-diagram/



Component Diagram


Presentation Layer
«interface»
User Interface
Web Browser
Service Layer
Room Process
Domain Layer
User
Role Choosing
Decision Choosing
Outcome Presentation
Room
Decision Presentation
Data Source Layer
Story
Round
Role
RoleDecision
Question
Decision
Use


Mapper
Use
Use
Database

ER Diagram (Database)

Updated on Aug. 12:

games:

```
    {"_id":{"$oid":"5edb3b7a45996d0a69c4ca78"},
    "players":[],
    "roles":[{"roleID":{"$numberInt":"0"},"role":"Boeing Executive","isAvailable":true,"player":null},
        {"roleID":{"$numberInt":"1"},"role":"Aeronautical Engineer","isAvailable":true,"player":null},
        {"roleID":{"$numberInt":"2"},"role":"Software Developer","isAvailable":true,"player":null},
        {"roleID":{"$numberInt":"3"},"role":"FAA Official","isAvailable":true,"player":null},
        {"roleID":{"$numberInt":"4"},"role":"Boeing Pilot","isAvailable":true,"player":null}],
    "duration":{"$numberInt":"0"},"playerCount":{"$numberInt":"1"},
    "status":"waiting","__v":{"$numberInt":"0"}}
```

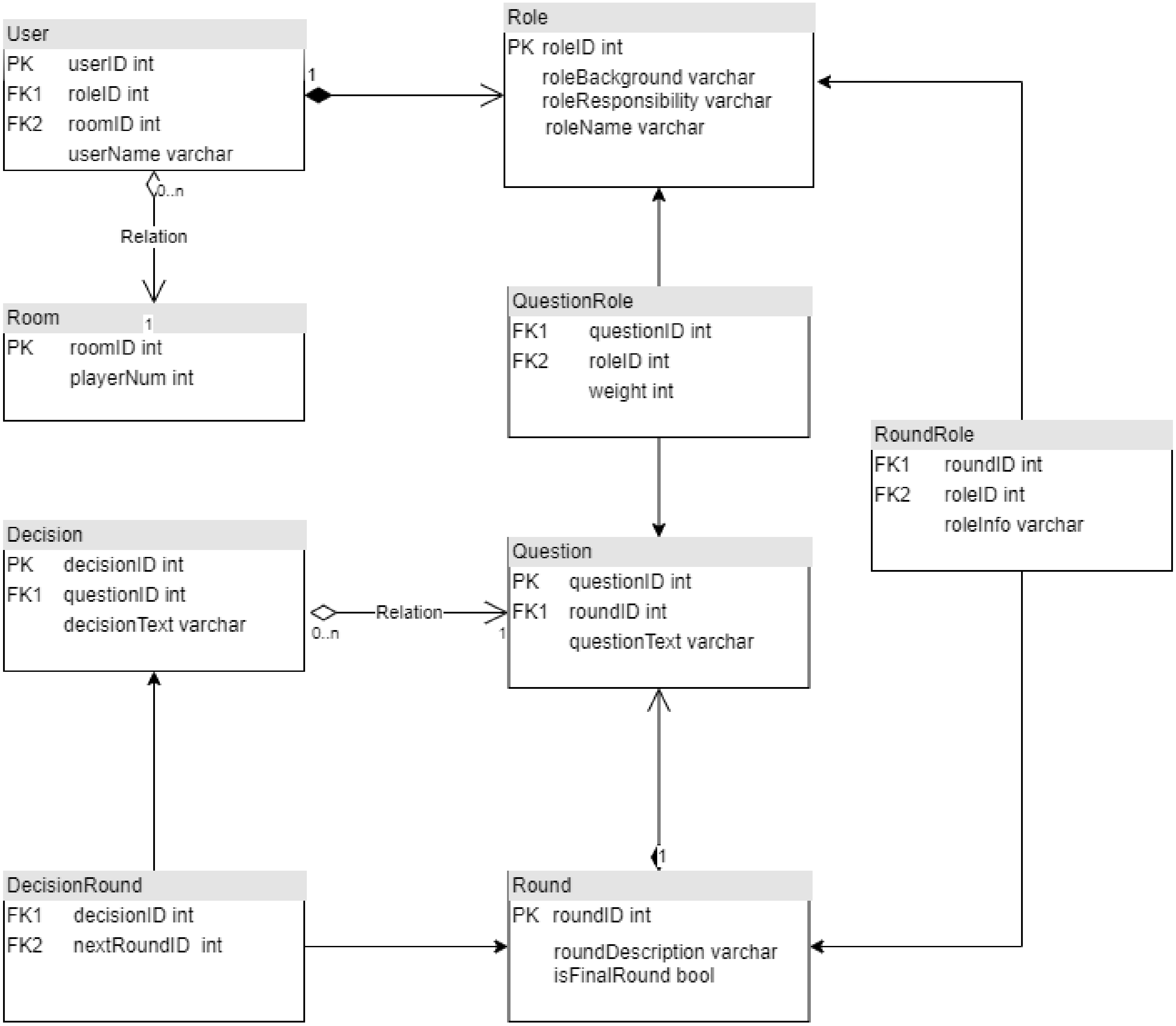
roles:

```
    {"_id":{"$oid":"5edc6a4b75432464a82b9255"},
    "name":"jest",
    "__v":{"$numberInt":"0"}}
```

users:

```
    {"_id":{"$oid":"5edc6a4a16736071ac6d878f"},
    "name":"jest",
    "university":"jest",
    "studentid":{"$numberInt":"1"},
    "__v":{"$numberInt":"0"}}
```

Version1. Based on the domain model in the draft (Architecture Design Draft)



User

PK	userID int
FK1	roleID int
FK2	roomID int
	userName varchar

Role

PK	roleID int
	roleBackground varchar
	roleResponsibility varchar
	roleName varchar

Room

PK	roomID int
	playerNum int

QuestionRole

FK1	questionID int
FK2	roleID int
	weight int

RoundRole

FK1	roundID int
FK2	roleID int
	roleInfo varchar

Decision

PK	decisionID int
FK1	questionID int
	decisionText varchar

Question

PK	questionID int
FK1	roundID int
	questionText varchar

DecisionRound

FK1	decisionID int
FK2	nextRoundID int

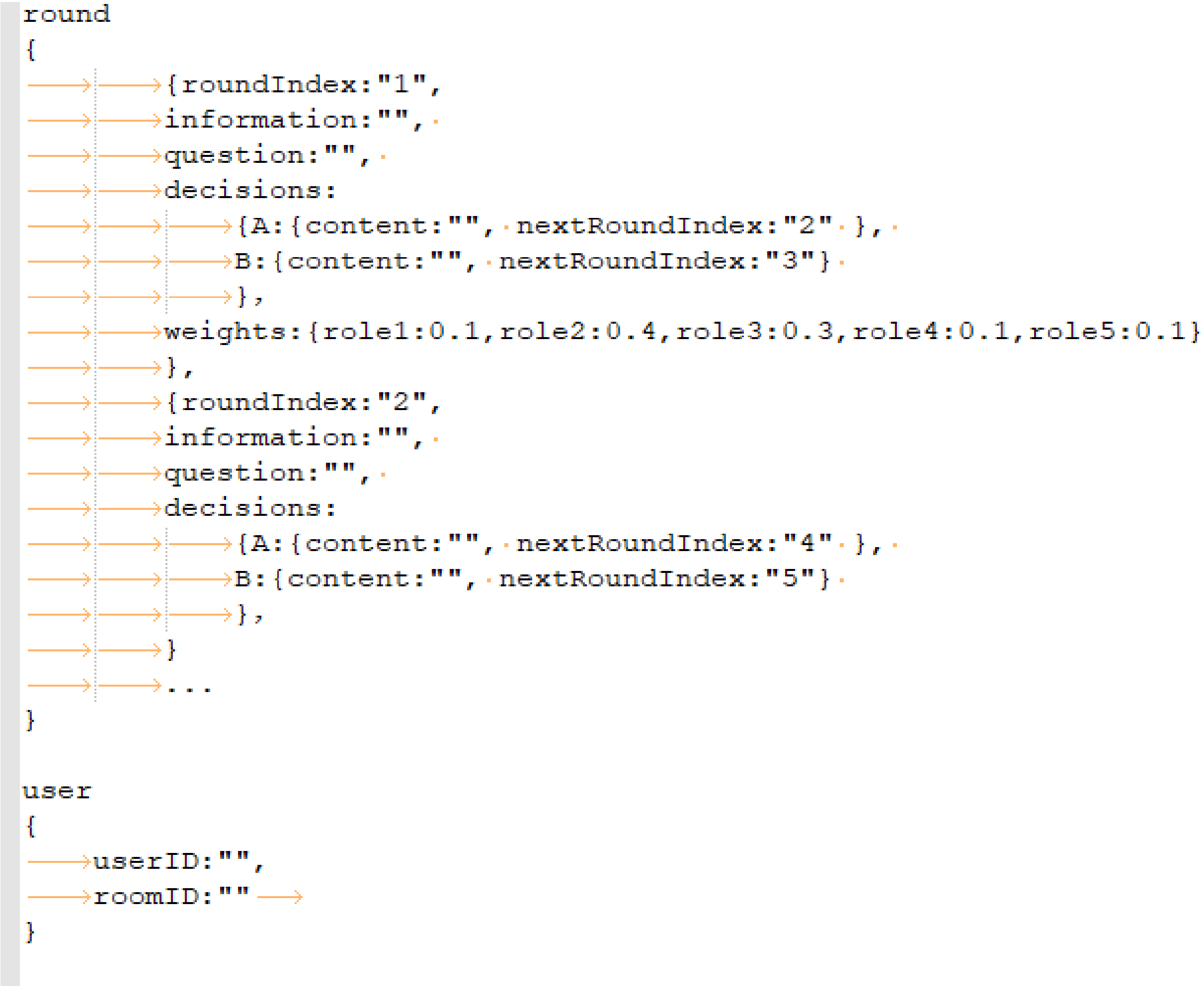
Round

PK	roundID int
	roundDescription varchar
	isFinalRound bool

*RoundDescription will be the outcome of the story when it is marked as the final round.

Version2: non-relational database design in the format of JSON datasample

An example:



```
round
{
        {roundIndex:"1",
        information:"", 
        question:"", 
        decisions:
            {A:{content:"", nextRoundIndex:"2" }, 
            B:{content:"", nextRoundIndex:"3"} 
            },
        weights:{role1:0.1,role2:0.4,role3:0.3,role4:0.1,role5:0.1}
        },
        {roundIndex:"2",
        information:"", 
        question:"", 
        decisions:
            {A:{content:"", nextRoundIndex:"4" }, 
            B:{content:"", nextRoundIndex:"5"} 
            },
        }
        ...
}

user
{
    userID:"",
    roomID:""
}
```

Decision-Tree Design

Key Features

- As per a standard tree design, the tree will feature a singular root node that corresponds to the starting state of the game
- Throughout a game session, a singular 'company' object will be persisted, which will represent the full state of information about the status of the company. It is a subset of this 'perfect' information set that will be delivered to each of the player roles where appropriate
- A question will list the options the player can take, where each option itself includes a 'consequence' object that describes how choosing that option influences the 'company' information, as well as the next decision node to move to in the tree

Logical Flow

1. Question is moved to, the Question sends player-specific information from the 'company' object to all players, and the decision question to the necessary player/s, with all of the corresponding options
2. The player/s make the decision, based on the 'consequence' object for that decision, the 'company' object is updated, the decision history is updated
3. Repeat!

Objects

<u>Game</u>

- Question currQuestion: A Question from a singleton 'Decision Tree' object persisted in back-end, representing the current Question the game is at
- Company companyInformation: An object storing all information about the company, which is updated based on decisions made during the game
    - Financial information, environmental information, productivity information, PR information etc.

<u>Question</u>

- int id: The identification number for the node
- string question: The decision question
- List<Option>: The options available for the decision, minimum 1, maximum 4
- int duration: The amount of time allocated for discussion/decision-making for this node
- List<int>: The weights for each role for the decision, should sum to 100 (corresponding to 100%)

Option

- string description: The description for the particular option
- Consequence consequences: An object storing the deltas for values in the Company object
    - This will be an identical looking object to the Company object
    - e.g. Company has a value netWorth = 100, Consequence has a value netWorth = -10, final netWorth value = 100 - 10 = 90
- Question next: The node to move to if this decision is chosen

Company

- Financials financialInfo
    - int contracts: Company contracts with airlines (Initial: 80)
    - int compContracts: Competitor contracts with airline (Initial: 20)
    - int shares: Company share value (Initial: 40)
- Reputations reputationInfo
    - int publicRep (0-100): Company reputation with the general public (Initial: 50)
    - int regulatorRep (0-100): Company reputation with the FAA (Initial: 50)
    - int internalRep (0-100): Company reputation with internal employees/managers (Initial: 50)
    - int boardRep (0-100): CEO reputation with board of directors (Initial: 50)
    - int enviroRep (0-100): Company reputation with environmental organizations (Initial: 50)
- Project projectInfo
    - int expectedProg (0-100): The expected level of progress of project according to board of directors (Initial: 0)
    - int actualProg (0-100): The actual level of progress of project (Initial: 0)
    - int actualCost: The actual level of project cost (Initial: 0)
    - int expectedCost (0-100): The expected level of project cost according to board of directors (Initial: 0)
    - int costDelay (0-100): The delay of the cost

We will need to update the Role object to include information about what values in the Company object that particular role can observe.

This model will require 1 unique Company object per game.

Advantages

- This design supports the presentation of role-specific information to the five different supported roles in the game
- Only one decision tree object is required to be persisted in back-end for ALL GAMES, no need for redundant copying of data
- Supports granular control of decision consequences on the company, as well as the complex structure of a real-life decision tree
- Supports individual player decisions, group decisions, and whole game decisions
- Easy to make changes, fairly flexible structure

Extensions

Feedback

- There needs to be an assurance on the decision tree that there is no cyclic behaviour, for example, if a child of a parent node directed back to the parent node, we could have an infinite loop, no node should be visited more than once

Decision History

Decision-Tree Design

Solution: Basic ArrayList/Linked List Implementation.

Objects

List

- Decision head: The first decision in the list
- Decision last: The last decision in the list
- int size: The length of the decision list

Decision

- Question question: A Question object corresponding to the tree node visited in the game
- Option chosen: The Option object chosen
- Decision next: The next Decision object in the list

We will make one of these lists per game.

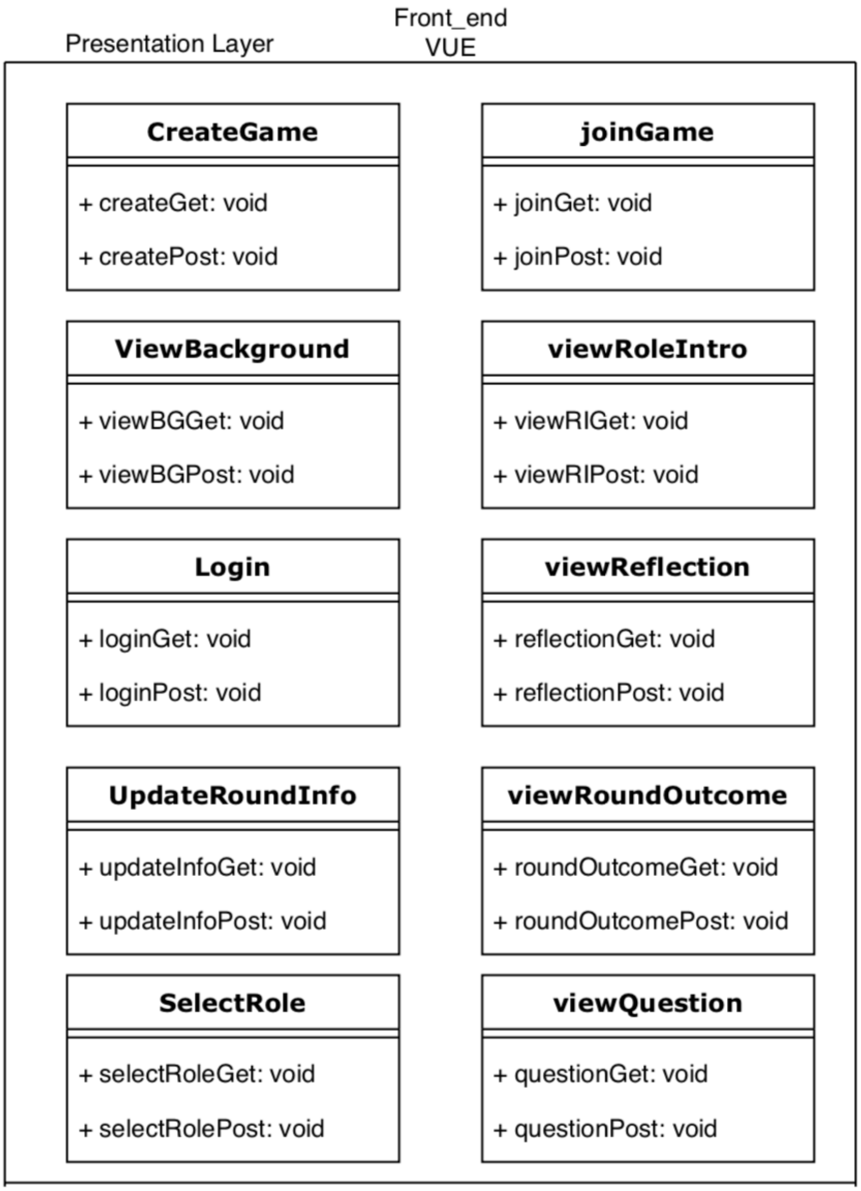
Advantages

- Easy to iterate through this structure and present a decision history
- Can be naturally added to as the game progresses, uses objects from decision-tree implementation
- Again, no need to make any copies of Question themselves
- We get a natural ordering that can be accessed e.g. 1->5->10->15...

Class Diagram

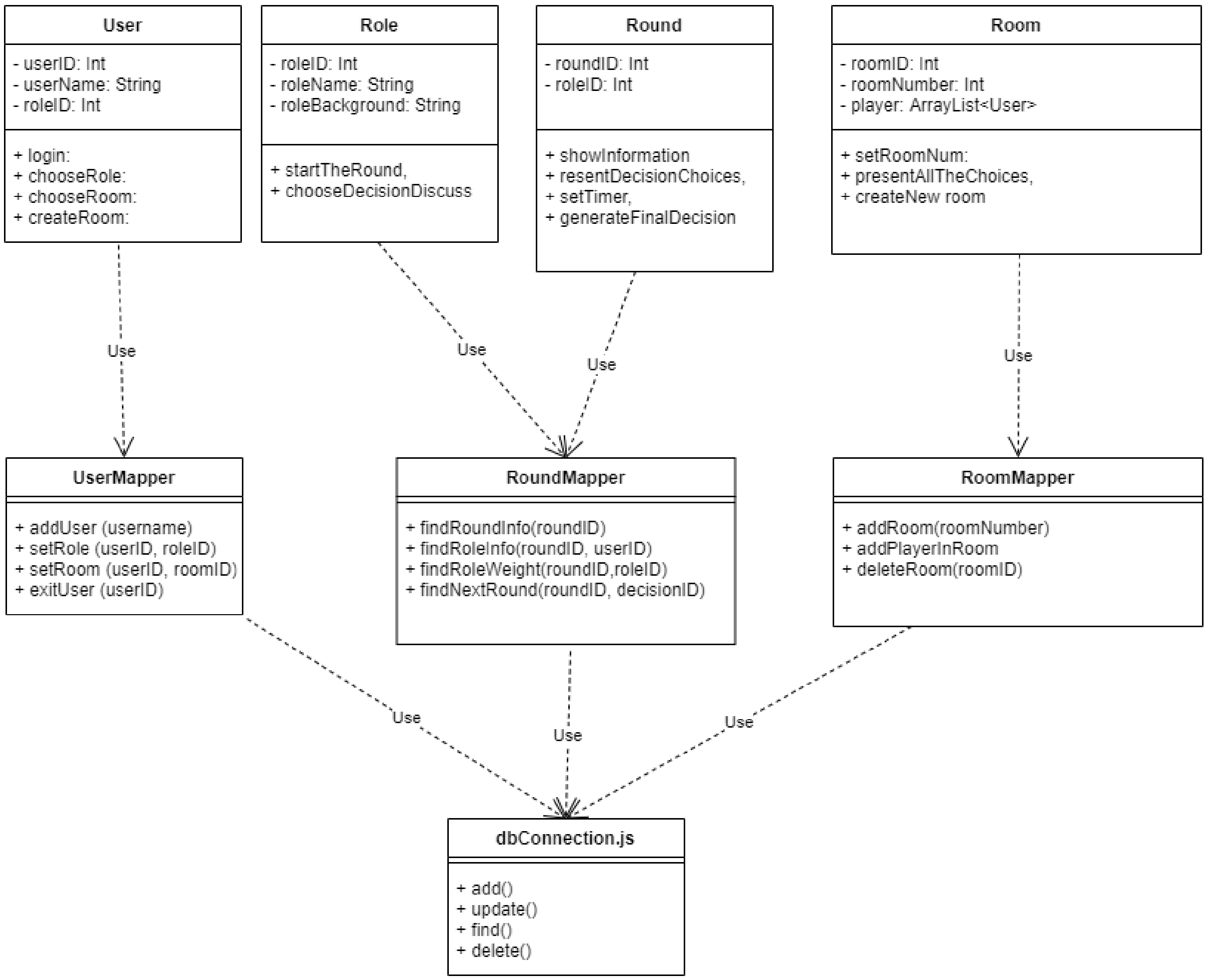
Class DiagramFrontend

https://app.diagrams.net/#G1Cpp6TvBSFy5qT_-uKMNGiL0YfBOg4i0P



Class DiagramBackend

https://app.diagrams.net/#G1IpqUjYMsOAupooXi3CctkYS-jFGC5Tjn


User
- userID: Int
- userName: String
- roleID: Int
+ login:
+ chooseRole:
+ chooseRoom:
+ createRoom:
Role
- roleID: Int
- roleName: String
- roleBackground: String
+ startTheRound,
+ chooseDecisionDiscuss
Round
- roundID: Int
- roleID: Int
+ showInformation
+ resentDecisionChoices,
+ setTimer,
+ generateFinalDecision
Room
- roomID: Int
- roomNumber: Int
- player: ArrayList<User>
+ setRoomNum:
+ presentAllTheChoices,
+ createNew room
Use
Use
Use
Use
UserMapper
+ addUser (username)
+ setRole (userID, roleID)
+ setRoom (userID, roomID)
+ exitUser (userID)
RoundMapper
+ findRoundInfo(roundID)
+ findRoleInfo(roundID, userID)
+ findRoleWeight(roundID,roleID)
+ findNextRound(roundID, decisionID)
RoomMapper
+ addRoom(roomNumber)
+ addPlayerInRoom
+ deleteRoom(roomID)
Use
Use
Use
dbConnection.js
+ add()
+ update()
+ find()
+ delete()

Sequence Diagram

Tool: https://sequencediagram.org/

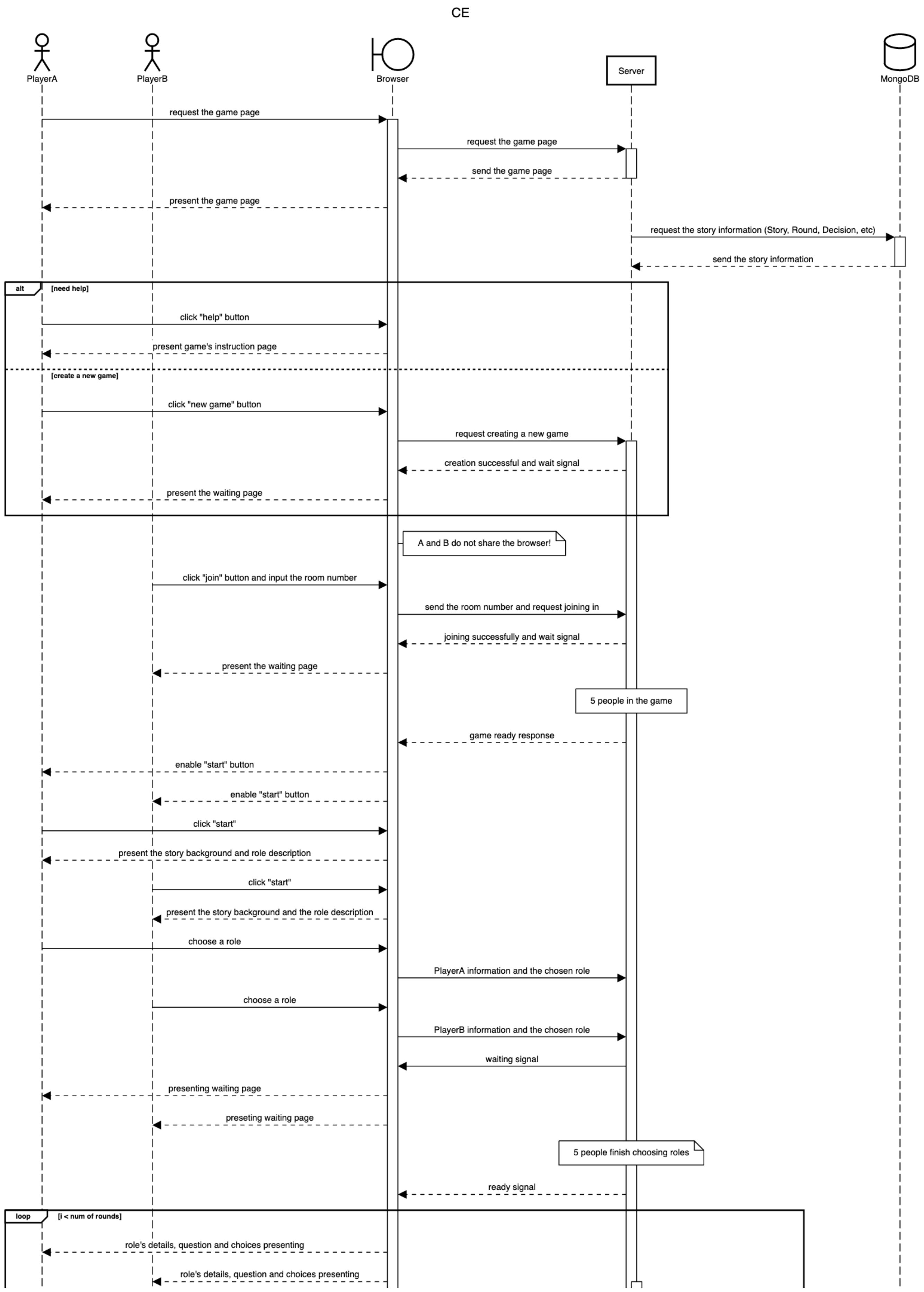
Link of the current diagram and code: CE Sequence Diagram

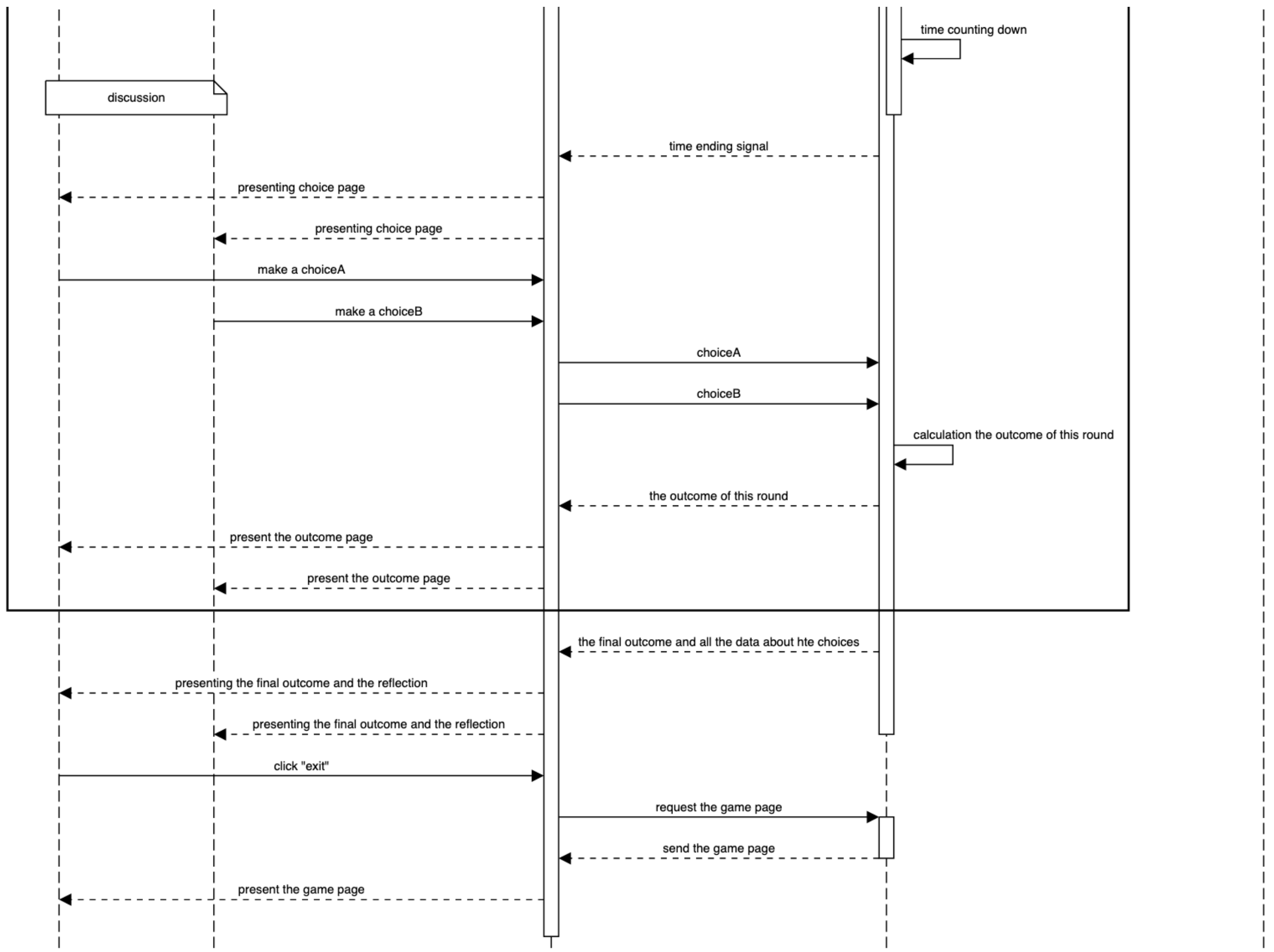
Note:

It is important to know that **some details of actions are not shown in this diagram**, otherwise this diagram will become really huge.

For example, the role description, the background information for each round and the decision for each round should be in different pages; the player can only see the next page when they click "next".

However, the assuption is that **all the data and resources should be transmitted back to the browser within one request and one response between the server and the browser (especially for playing in round part)**, so that the page forwarding can be done by the browser and no longer depend on the server.


CE
PlayerA
PlayerB
Browser
Server
MongoDB
request the game page
request the game page
send the game page
present the game page
request the story information (Story, Round, Decision, etc)
send the story information
alt
[need help]
click "help" button
present game's instruction page
[create a new game]
click "new game" button
request creating a new game
creation successful and wait signal
present the waiting page
A and B do not share the browser!
click "join" button and input the room number
send the room number and request joining in
joining successfully and wait signal
present the waiting page
5 people in the game
game ready response
enable "start" button
enable "start" button
click "start"
present the story background and role description
click "start"
present the story background and the role description
choose a role
PlayerA information and the chosen role
choose a role
PlayerB information and the chosen role
waiting signal
presenting waiting page
preseting waiting page
5 people finish choosing roles
ready signal
loop
[i < num of rounds]
role's details, question and choices presenting
role's details, question and choices presenting


time counting down
discussion
time ending signal
presenting choice page
presenting choice page
make a choiceA
make a choiceB
choiceA
choiceB
calculation the outcome of this round
the outcome of this round
present the outcome page
present the outcome page
the final outcome and all the data about hte choices
presenting the final outcome and the reflection
presenting the final outcome and the reflection
click "exit"
request the game page
send the game page
present the game page

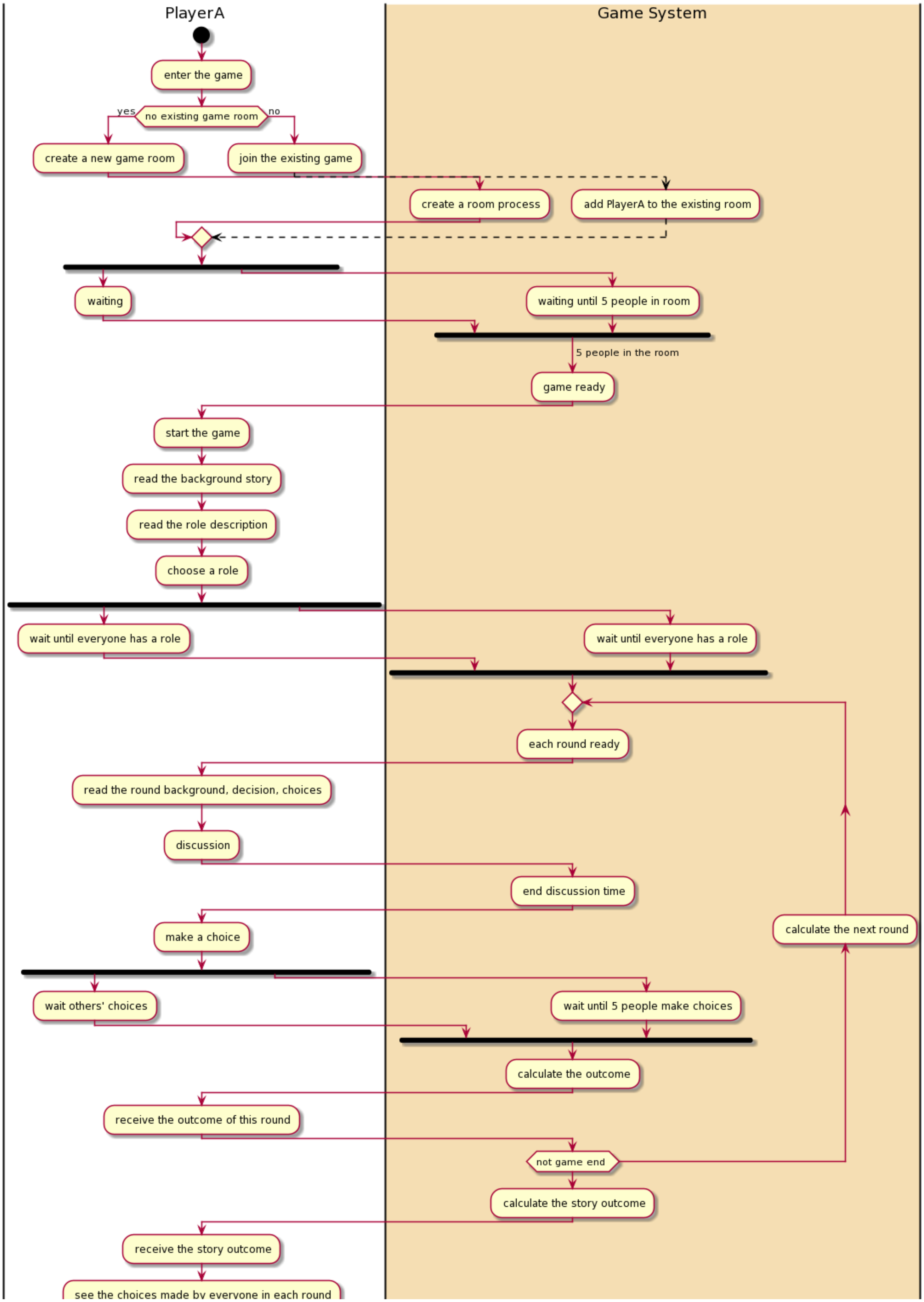
Activity Diagram

Description:

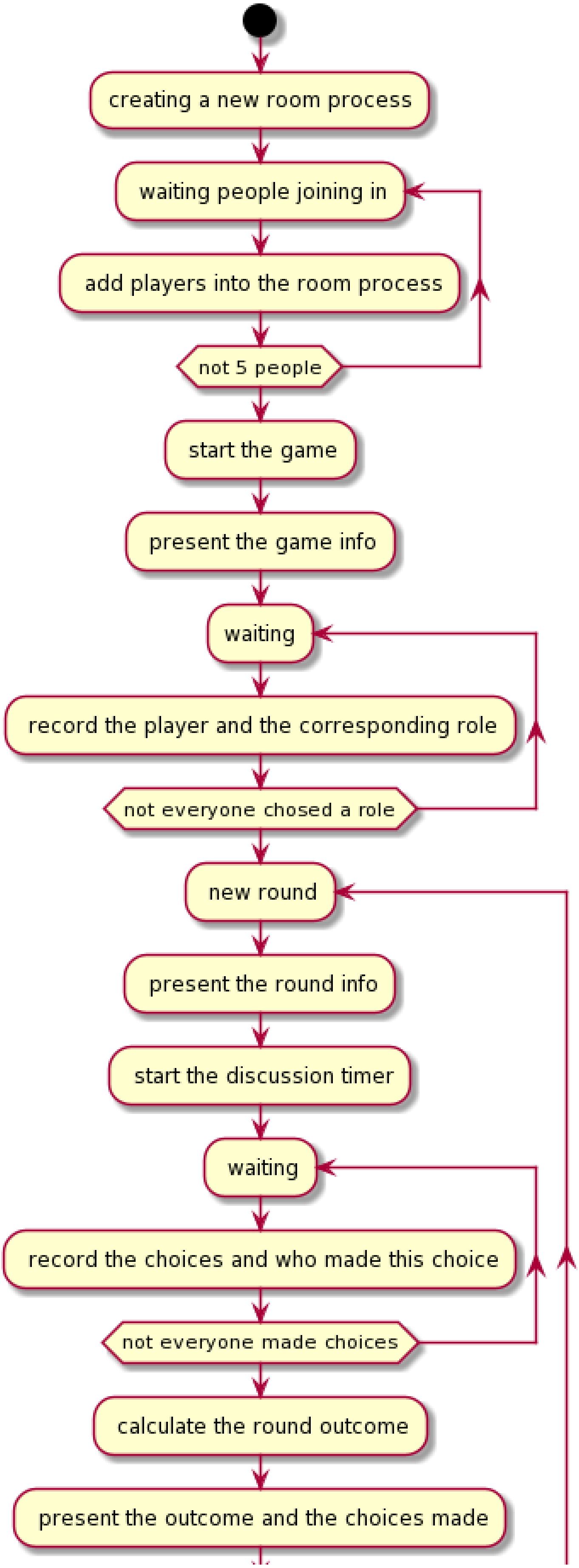
The activity diagram is used to describe the activities that will be performed by the user (player) and the system (the game). It also describe the functionality that the system provides to the users.

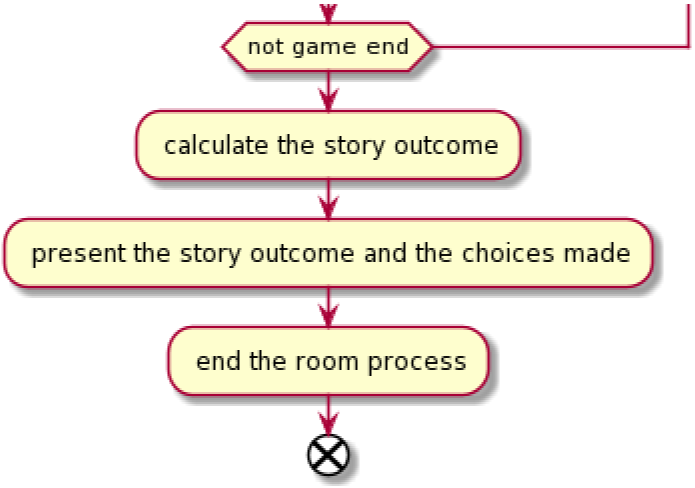
The activity is an abstraction of the behaviour excluding the details of interations. For the interation details between the frondend, the backend server and the database, please refer to Sequence Diagram.

Player and Game System Activity Diagram


PlayerA
Game System
enter the game
no existing game room
yes
no
create a new game room
join the existing game
create a room process
add PlayerA to the existing room
waiting
waiting until 5 people in room
5 people in the room
game ready
start the game
read the background story
read the role description
choose a role
wait until everyone has a role
wait until everyone has a role
each round ready
read the round background, decision, choices
discussion
end discussion time
make a choice
calculate the next round
wait others' choices
wait until 5 people make choices
calculate the outcome
receive the outcome of this round
not game end
calculate the story outcome
receive the story outcome
see the choices made by everyone in each round

Game System Activity Diagram


creating a new room process
waiting people joining in
add players into the room process
not 5 people
start the game
present the game info
waiting
record the player and the corresponding role
not everyone chosed a role
new round
present the round info
start the discussion timer
waiting
record the choices and who made this choice
not everyone made choices
calculate the round outcome
present the outcome and the choices made


not game end
calculate the story outcome
present the story outcome and the choices made
end the room process

Technology

Frontend

Vue.js

For more information, please refer to Front-End Tools

Backend

Node.js

For more information, please refer to Back-end Tools

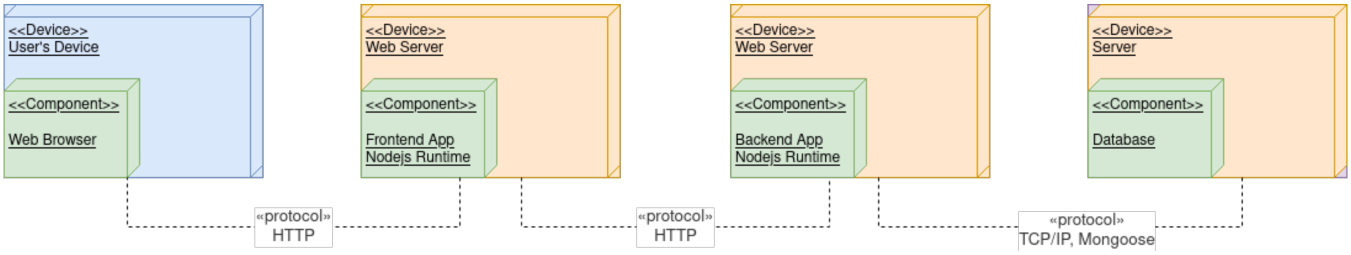
Deployment

Docker: For setting up local environments

Gitlab: For building CI/CD pipelines

For more information, please refer to Dev-Ops Tools

Deployment Diagram



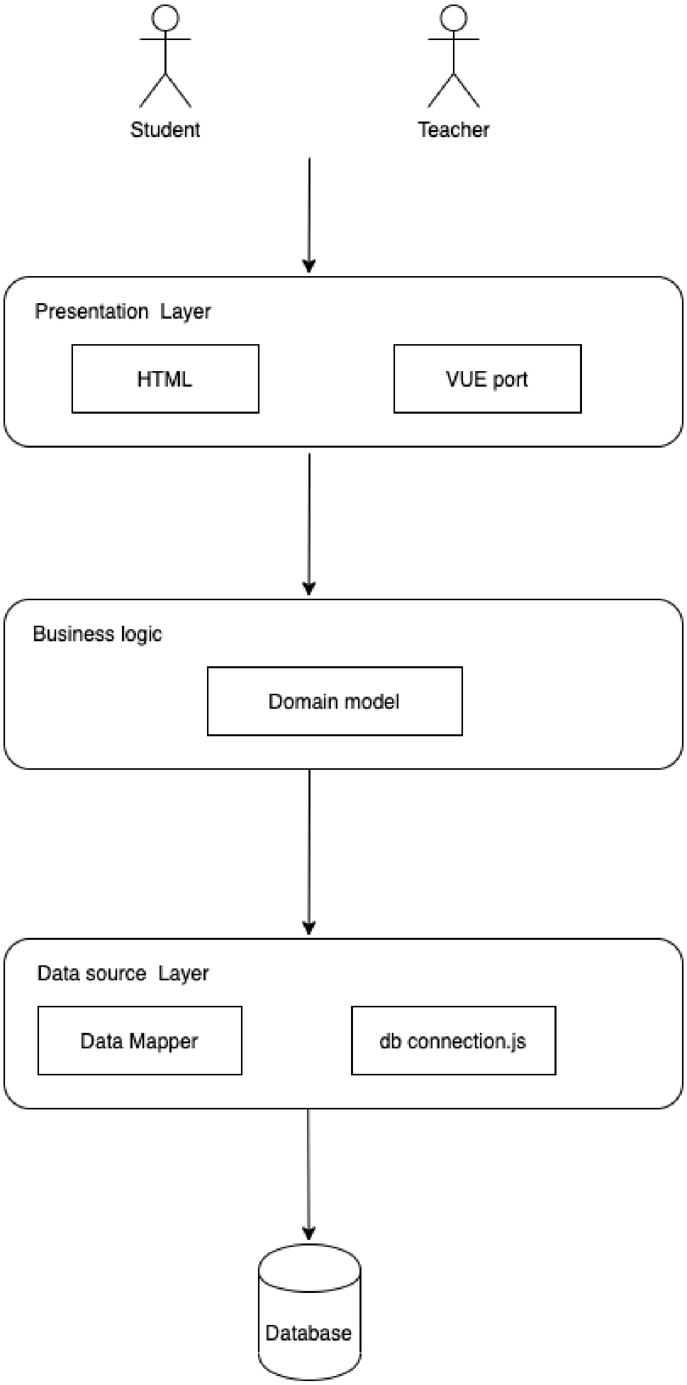
Notes

- In our current iteration The Front-end and Back-end Applications reside in separate servers. This decoupling can make scaling the system easier in the future.

High Level Diagram

This page is the high level diagram for the project.

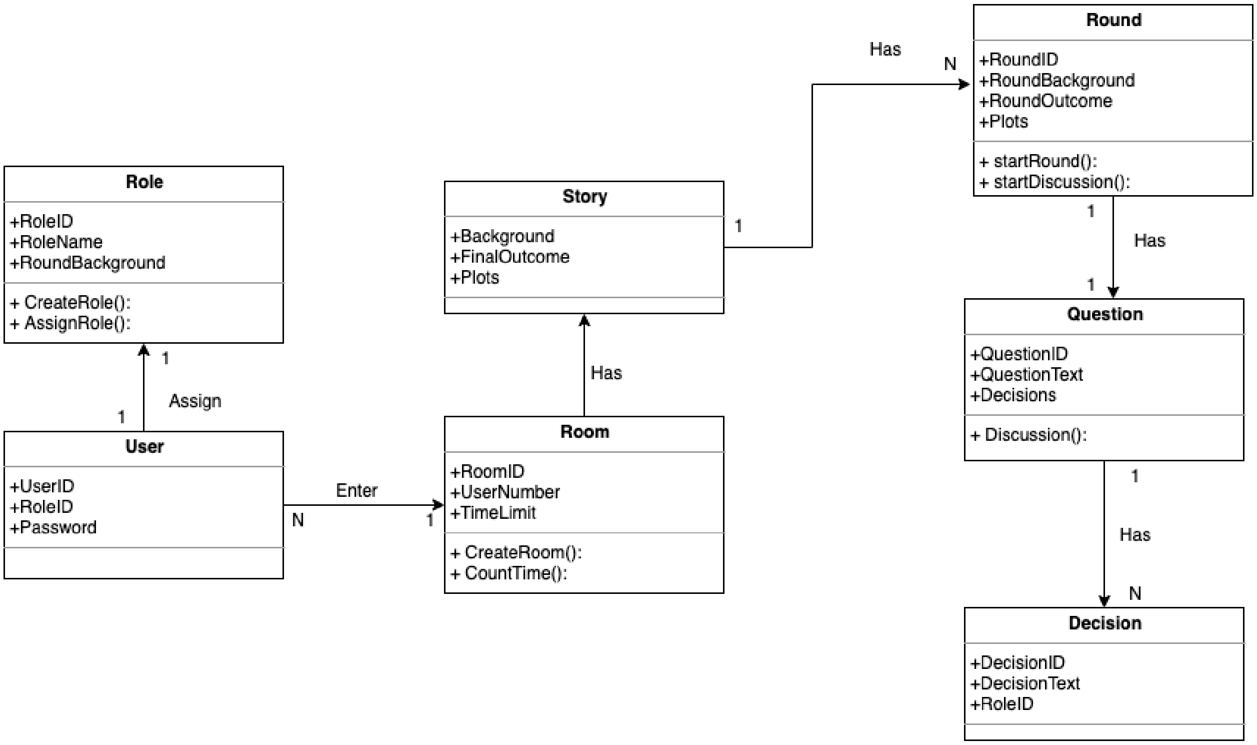
https://app.diagrams.net/#G1ty3dOC0gLK_KkIfQ2W1Ah9oSHx3FZ0a9



Domain Model

This page is Domain model for the project

https://app.diagrams.net/?state=%7B%22folderId%22:%2216hModzuCo_4oxqC1zJFYqjpUR_nlupZ6%22,%22action%22:%22create%22,%22userId%22:%22109999630773790380818%22%7D#G1B2Dn9RzMut6Gwm_7l-nIBh1hFpn8ghu0



API Specification

This page documents the API used to connect the front- and back-ends.

ID	Implemented at Back-End	Implemented at Front-End	Method	Endpoint	Description	Parameters	Responses
1	✓	✓	POST	/play/new	A user wants to start a new game	- body - `{"name": "John",` `"duration": 30}` - Note: if user does **not** select that they want a timed game, duration = 0	- 200 - 400
2	✓	✓	POST	/play/{gameID}/join	A user wants to join an existing game	- gameID - (path, string) - body - `{"name": "John"}`	- 200 - 400 - e.g. invalid Game ID supplied

3	✓	✓	GET	/play/{gameID}/wait	Player is waiting for all players to join the game	- gameID
						  - (path, string)
							- 200
							  - `{"players":`
							`[`
							`{"id": 0, "name": "John"},`
							`{"id": 1, "name": "Georgia"},`
							`{"id": 2, "name": "Emily"}`
							`]`
							`"ready": false}`
							  - ready = true if all 5 players are in and the game is ready to start
							- 400
							  - e.g. invalid Game ID

4	✓	✓	GET	/play/{gameID}/status	Retrieve information about the current game e.g. when viewing the in-game menu	- gameID - (path, string)	- 200 - {"status": "inProgress", "players": [{"id": 0, "name": "John", "role": "Aeronautical Engineer"}, {"id": 1, "name": "Georgia", "role": "Boeing Executive"}, {"id": 2, "name": "Emily", "role": "Pilot"}, {"id": 3, "name": "Ming", "role": "FAA Official"}, {"id": 4, "name": "Steve", "role": "Software Developer"},] } - subject to change - status of the game can be: - waiting - waiting for all players to join the game - ready - all 5 players have joined, ready for players to press "Start game" - inProgress - completed - 400 - e.g. invalid Game ID
5	✓	✓	GET	/play/{gameID}/background	Get the "Background" information shown to all players	- gameID - (path, string)	- 200 - the content (perhaps as plain text including <p> tags? or maybe as JSON with .) - 400
6	✓	✓	GET	/play/{gameID}/roles	Get the "role" information shown to all players	- gameID - (path, string)	- 200 - {"roles": [{"title": "Boeing Executive", "description": "Oversees the whole 737 MAX project, as well as the other operations occurring within the Boeing company."}, {"title": "Aeronautical Engineer", "description": "Acts as the project lead for the 737 MAX project, and is responsible for the continuous and stable operation of the 737 MAX in the future."} etc] }
7	✓	✓	GET	/play/{gameID}/choose-role	Players select their roles. Gets all the roles, whether the role is available to be selected (and if not, who is it assigned to)	- gameID - (path, string)	- 200 - {"roles": [{"roleID": 0, "role": "Boeing Executive", "isAvailable": true}, {"roleID": 1, "role": "Aeronautical Engineer", "isAvailable": true}, {"roleID": 2, "role": "Software Developer", "isAvailable": true}, {"roleID": 3, "role": "Pilot", "isAvailable": false, "player": "Ming"}, {"roleID": 4, "role": "FAA Official", "isAvailable": false, "player": "Georgia"}] } - 400

8	✓	✓	POST	/play/{gameID}/choose-role	Player selects their role	- gameID - (path, string) - roleID - role ID (int between 0 and 4, as specified in the previous request)	- 200 - player successfully assigned to role - 400 - e.g. invalid game ID - 403 - player is not assigned to the role e.g. the role is no longer available
9	✓ update	✓ now have a bug	GET	/role/introduction/	Get the "Role Introduction" content, i. e. a page of text showed to the player at the start of the game, based on the role they chose	- body - `{"roleID":0}`	- 200 - the content - 400

10		✓	GET	/play/make-decision/:gameID	Get all the decisions during the game in reflection phase.	- gameID - (path, string)	- 200 - {"reflectionList": [{"questionID": 1, "questionText": "Do we need to halt development in order to further investigate the AoA sensor?" "options": [{"id": 1, "text": "Halt development", "users": [{"name": "hh", "role": "Aeronautical Engineer"},{"name": "Emily", "role": "Software Developer"}] }, {"id": 2, "text": "Continue Development", "users": [{"name": "Steve", "role": "Executive"}, {"name": "Ming", "role": "Pilot"},{"name": "Georgia", "role": "FAA Official"}] }], "result": 1 }, {"questionID": 1, "questionText": "should we build an expensive new software module to help avoid potential issues with the sensor?" "options": [{"id": 1, "text": "Develop new module", "users": [{"name": "hh", "role": "Aeronautical Engineer"}] }, {"id": 2, "text": "Do not develop new module", "users": [{"name": "Steve", "role": "Executive"}, {"name": "Ming", "role": "Pilot"},{"name": "Georgia", "role": "FAA Official"}, {"name": "Emily", "role": "Software Developer"}] }], "result": 2 },] } - 400 - e.g. invalid game ID

11	✓		GET	/play/{gameID}/round	Get all the information that the player needs to see for the current round, i.e. the information shown under "Chapter One" in the prototype: "Background", "Role Background" and "Decision"	• gameID • (path, string) • body • `{"roleID": 1}`	• 200 • `{"questionID": 1, "background": ["This is the first paragraph of text. Work is progressing on-time for the design of the Boeing 737 MAX; however it is critical to keep up the current pace to meet the deadline.", "This is another paragraph of text. The design team are currently focussed on perfecting the MCAS system to help manoeuvre the stabiliser nose of the aeroplane to reduce the risk of stalling."], "roleinfo": ["You discover that there may be an issue with the angle-of-attack (AoA) sensor used in the aeroplane design.", "The sensor usually is reliable, but in certain conditions during testing it appeared to report erroneous data.", "Further investigation could be performed in order to figure out the cause and severity of this issue."], "question": "Should we build an expensive new software module to help avoid potential issues with the sensor?" "options": [{"id": 1, "text": "Develop new module"}, {"id": 2, "text", "Do not develop new module"}] }` • 400 • e.g. invalid game ID
12	✓		POST	/play/make-decision/:gameID	Player enters their selected choice for the question of the round	• gameID • (path, string) • body • `{"questionID": 1, "optionID": 2, "roleID": 1}`	• 200 - choice was valid and recorded correctly • `{"questionID": 1, "optionID": 2, "roleID": 1}` • 400 • e.g. invalid input, round has expired, etc.

13	✓		GET	/play/{gameID}/round-status	Get information about the current round. The purpose of this is so the front-end can frequently check the state of the round, and know when to move on to the outcome.	- gameID - (path, string)	- 200 - {"questionID": 1, "status": "pre" "voted": 2 "remaining": 3 } - roundID is the current round in play for this game - perhaps this should be 0 if we are not actually in a round (e.g. we are still in the role selection phase). - status of the **round** can be: - **pre** - pre-decision. round is in progress, timer is running at back-end, players should be reading the information /discussing/entering their choice - **post** - post-decision. all players have voted and/or the round timer has expired. Players can no longer enter a choice, and are now shown to the outcome of the decision. - voted: number of players that have entered their decision - remaining: number of players who still need to enter their decision - 400 - e.g. invalid game ID
14	✓		GET	/play/{gameID}/round-outcome	Get the outcome and conclusion for the round i.e. "Result" page in the prototype	- gameID - (path, string)	- 200 - {"questionID": 1, "question": "Should we build an expensive new software module to help avoid potential issues with the sensor?" "options": [{"id": 1, "text": "Develop new module"}, {"id": 2, "text", "Do not develop new module"}] "result": 2 "outcome": ["First paragraph. The decision has been made not to develop this new module.", "Another paragraph. While the Aeronautical Engineer wanted this module to be developed, the other four decided this was not necessary."] } - 400 - e.g. invalid game ID
15			GET	/play/{gameID}/game-outcome	Get the final game outcome of game i.e. "Outcome" page in the prototype	- gameID - (path, string)	- 200 - The game outcome - 400 - e.g. invalid game ID

Decision Tree

For the structure and content underlying the decision tree, refer to Decision Tree (Final plot)