Software Analysis & Design Part 2 CA

Continuous Assessment -Connect 4 Game

SA52 Team 2

Name	Student ID
Fun Weng Lup, Ronnie	A0230477E
Aye Phyu Sin	A0230446M
Changying Shao	A0230469A
Willard Toh Hui Kian	A0226755X
Chen Wenquan	A0230451W
Chia Yan Long, Brandon	A0097266R
Hou Lu Chiok Weh Alejandro	A0230492L
Zhao Qi	A0230464M

Contents

1. Background

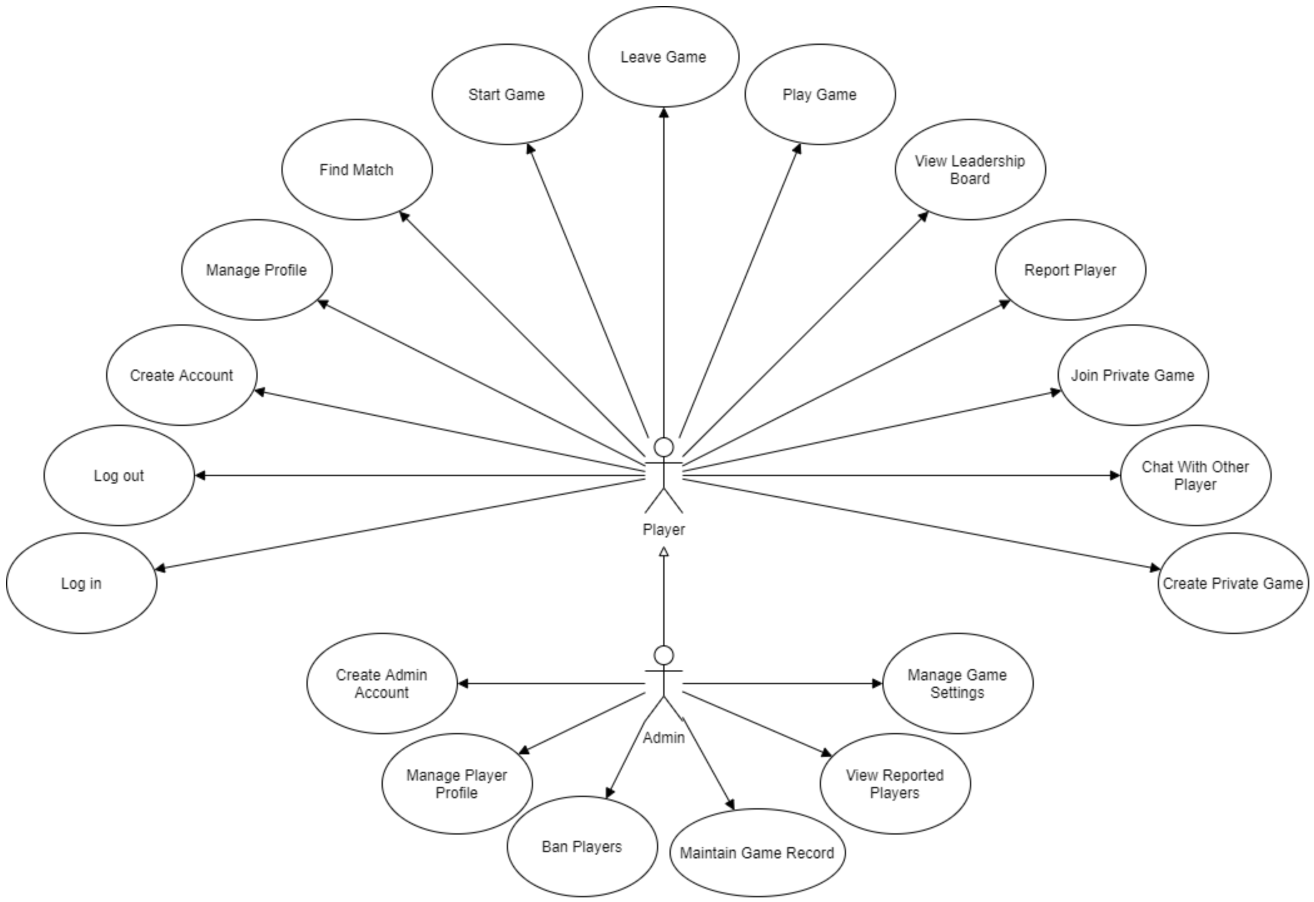
The aim of this project is to develop a simple game of “Connect 4” where players can play against each other.

The objective of connect 4 is to connect 4 tokens of the same colour in a 6 by 7 grid vertically, horizontally, or diagonally. The Tokens must be stacked upwards starting from the bottom of the board. Players will take turns placing their tokens of assigned colour until the win condition has been met.

Players will be “match-made” with other players of similar skill levels for “ranked” play or play in a casual private game between their friends.

The results for each “ranked” match will be recorded and the leader board updated. Player will also be able to chat to one another during the match so that there is an option for social interaction. The program will also allow players to create, view and manage their own profile including their win/loss records.

2. Use Case Diagram



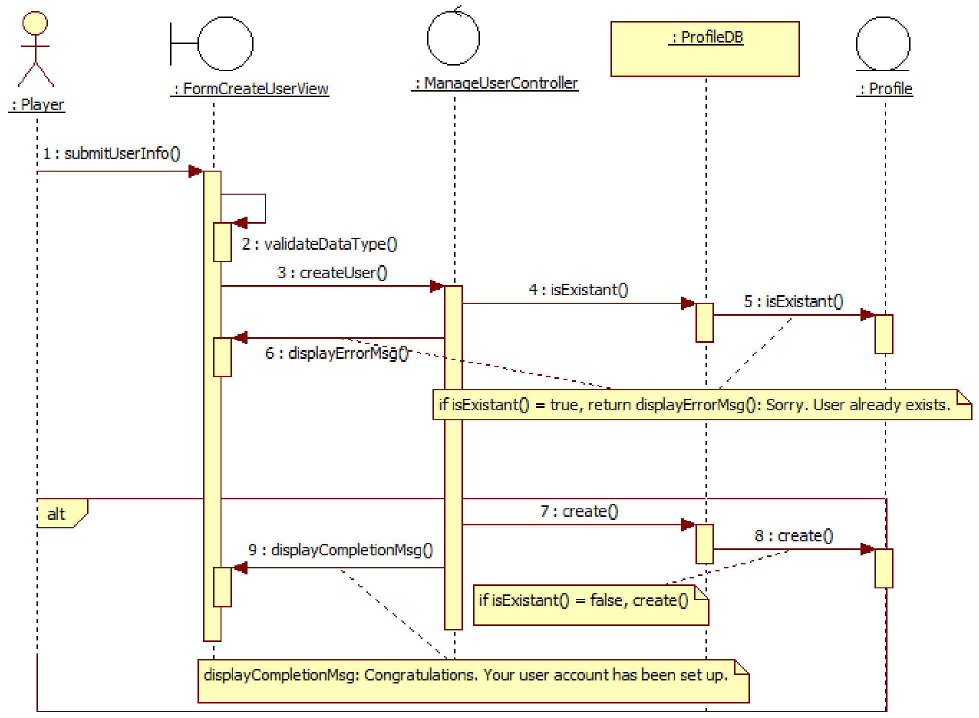
3. Analysis Model

3.1 Player→Create Account

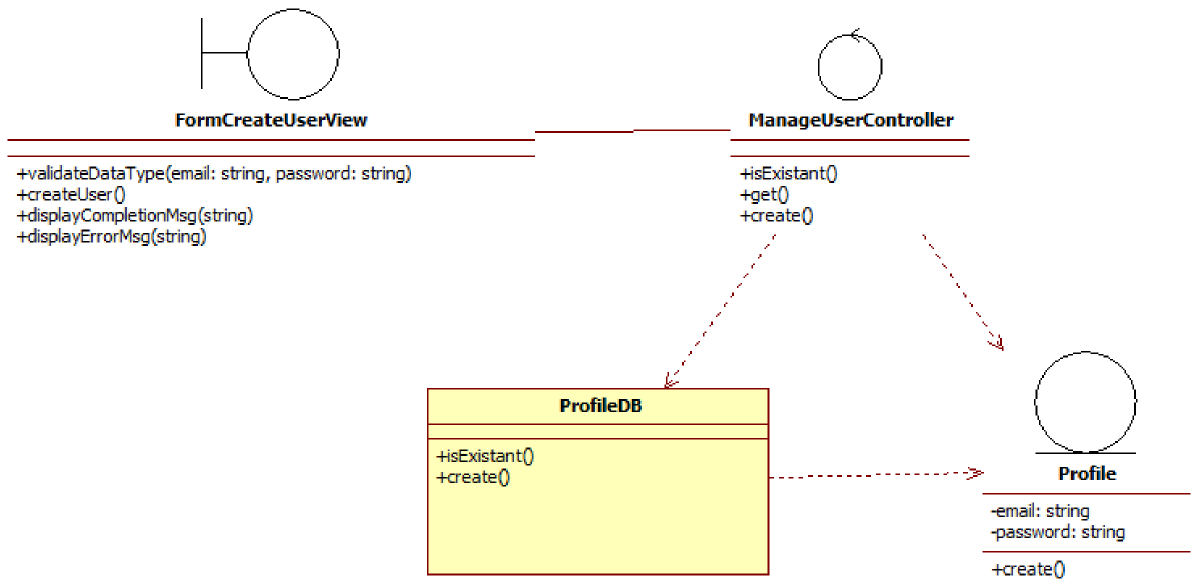
3.1.1 Description

As a pre-requisite for all prospective players to start playing our rendition of "Connect 4", each would require their own respective user accounts. The required fields for each player account set-up would be their first name, their preferred email account and their password. Our form requires each field to input based on an appropriate data type (including a requirement for a special character, lowercase, uppercase, digit and special character, and an email consisting an @). Our form will conduct a self-validation before submission. Following which, our profile database will conduct a quick sieving through all pre-existing accounts to find users with similar details. An error message will be returned to the form view if there was a pre-existing user with a similar email address / username pair, with an indication inscribed, "Sorry. User already exists." As exhibited in the alternative route, should there be no matches, the account will be successfully created, with a completion message inscribed, "Congratulations. Your user account has been set up."

3.1.2 Sequence Diagram


: Player
: FormCreateUserView
: ManageUserController
: ProfileDB
: Profile
1 : submitUserInfo()
2 : validateDataType()
3 : createUser()
4 : isExistant()
5 : isExistant()
6 : displayErrorMsg()
if isExistant() = true, return displayErrorMsg(): Sorry. User already exists.
alt
7 : create()
8 : create()
9 : displayCompletionMsg()
if isExistant() = false, create()
displayCompletionMsg: Congratulations. Your user account has been set up.

3.1.3 Class Diagram

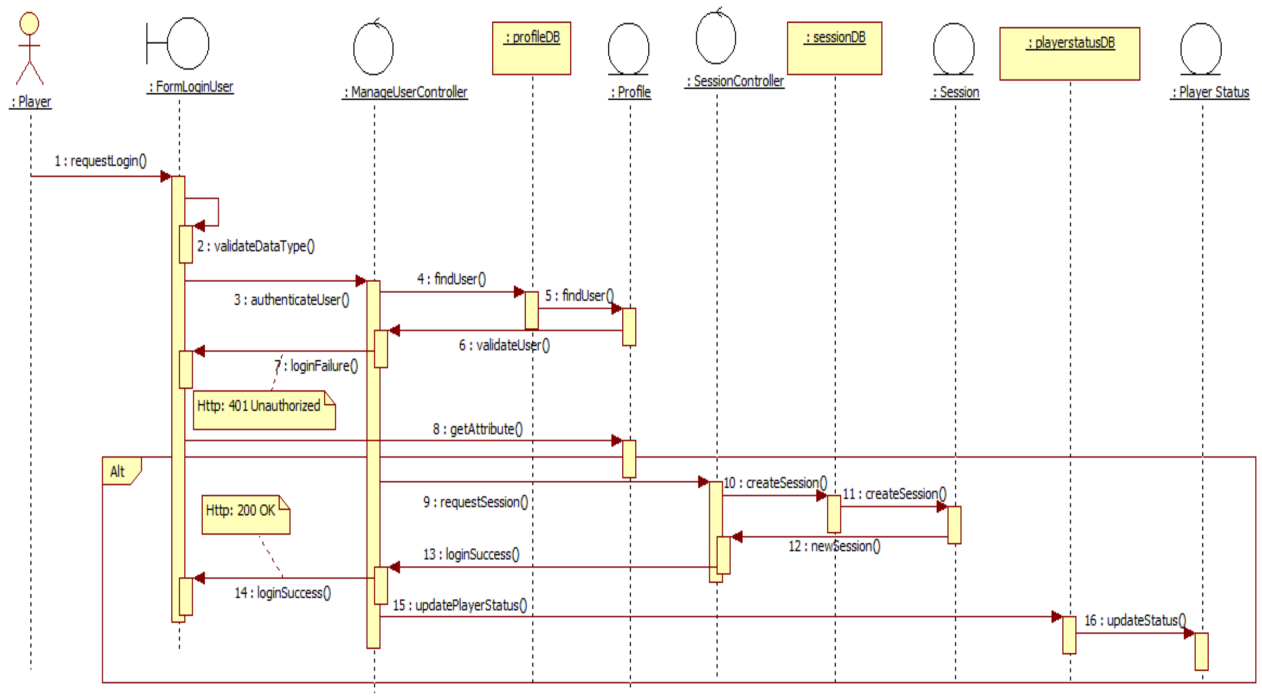


3.2 Player→Log In

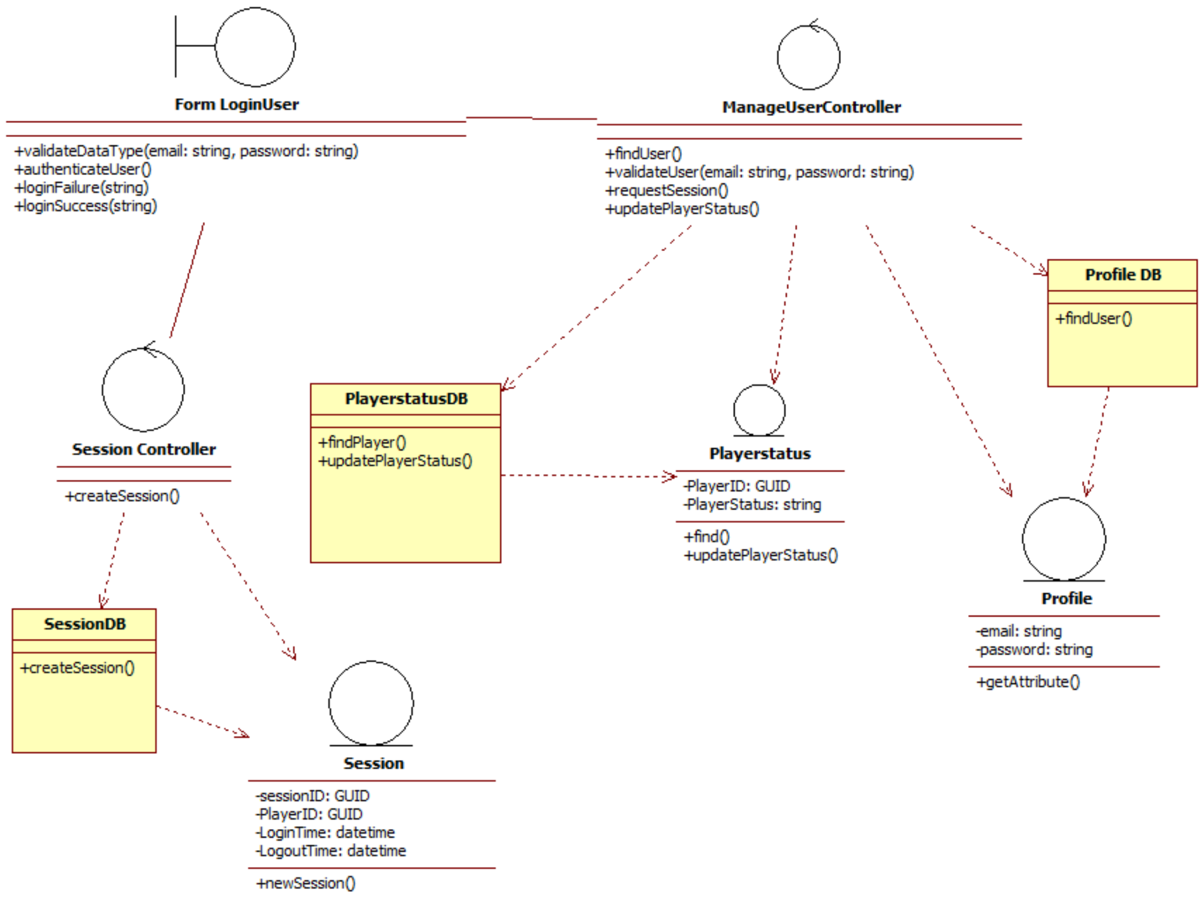
3.2.1 Description

For every instance that the player would like to join a game (be it – ranked or unranked), each player will have to log in to their activated user accounts. They can commence this process by requesting a login at the login graphical user interface (GUI) called FormLoginUser. The GUI will conduct a self-validation of each input and their data types. Following which, the system will scan through the profile database for the matching user inputs. Upon validation attempt, if the system could not find a user to match, it will return a 401: Unauthorized error message. Alternatively, should the system find a matching user, it will request a new session from the session database and assign it to the user from the time their login was authenticated to the time they decide to end their session. The session will then be rendered a success with a 200: Ok message. For every user-tagged session that is ongoing, a player status will be assigned. So long as the player has their mouse hovering around the screen, they will be tagged as "active". If they are within a game, they will be tagged as "ingame". If the player is idling for more than three minutes, they will be tagged as "idling".

3.2.2 Sequence Diagram



3.2.3 Class Diagram

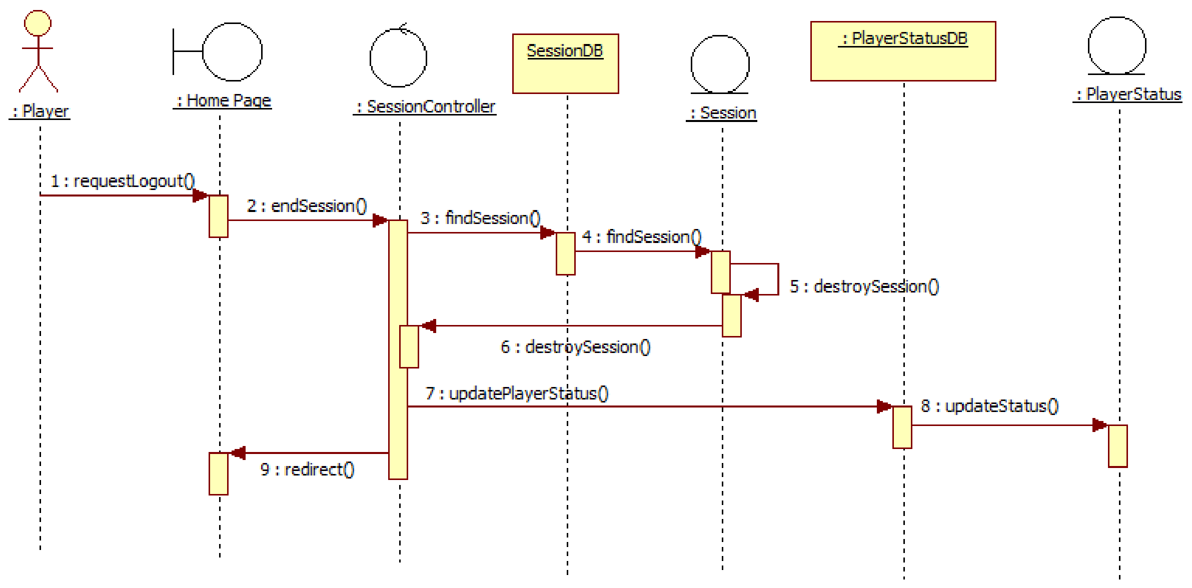


3.3 Player→Log Out

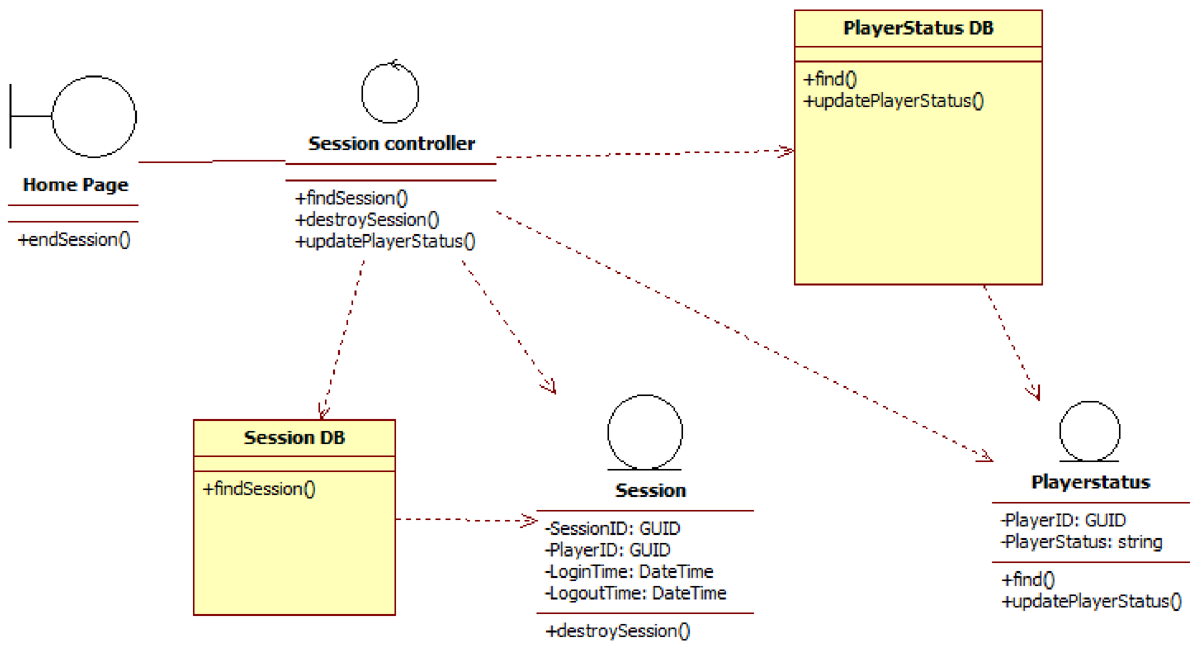
3.3.1 Description

At any juncture the player chooses to terminate their session, they could request a log out by clicking on said function on the home page. The session manager serves as a controller to identify which user session was active during the player's playing tenure, find the session and destroy it. The player status will then be set as "offline" before redirecting the player back to the home page.

3.3.2 Sequence Diagram



3.3.3 Class Diagram



3.4 Player→Manage Profile

3.4.1 Description

3.4.1.1 Manage profile (update) [Note: Player is logged in]

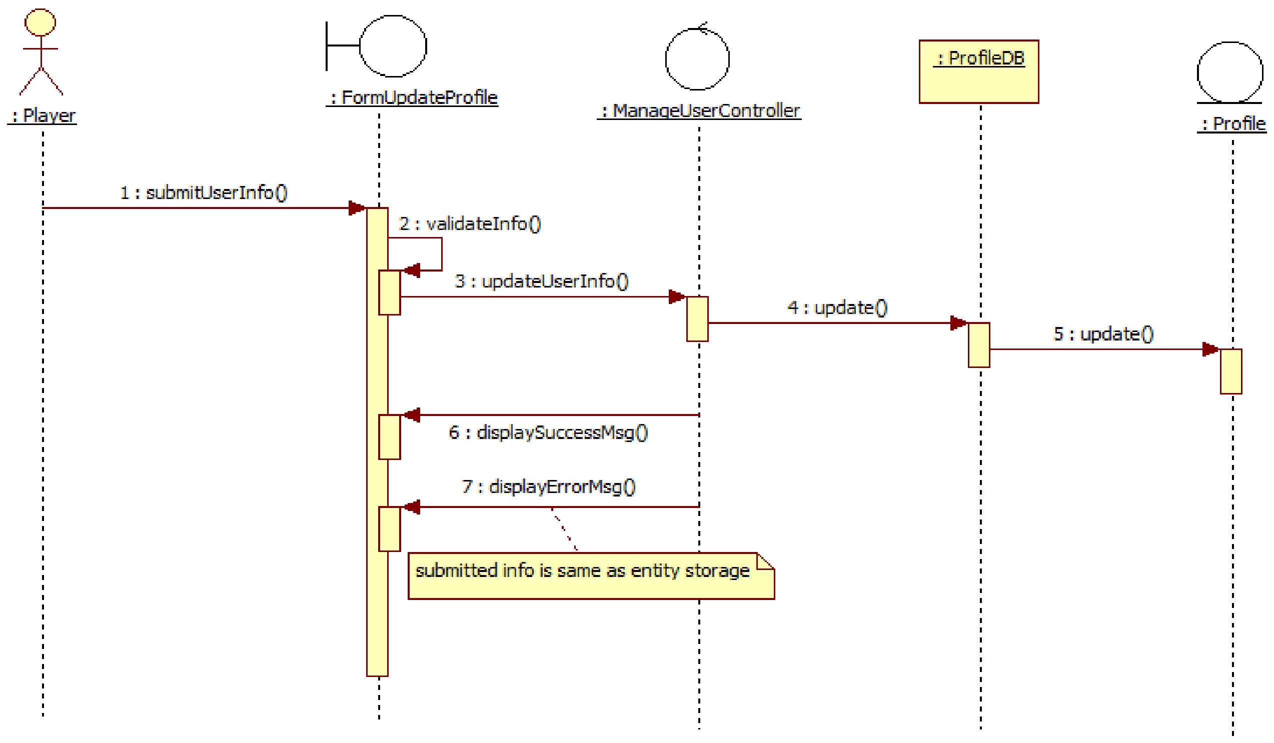
As the logged in user chooses to update his profile, he will go through a very simple step and that is to submit the info via a form. Validation will happen within the form itself to check if password/email follow a correct and acceptable format. An error could occur if the user submits the same info as stored in the Profile entity. In such an event, he will be prompted with an error message and asked to try again or abandon the operation.

3.4.1.2 Manage profile (delete) [Note: Player is logged in]

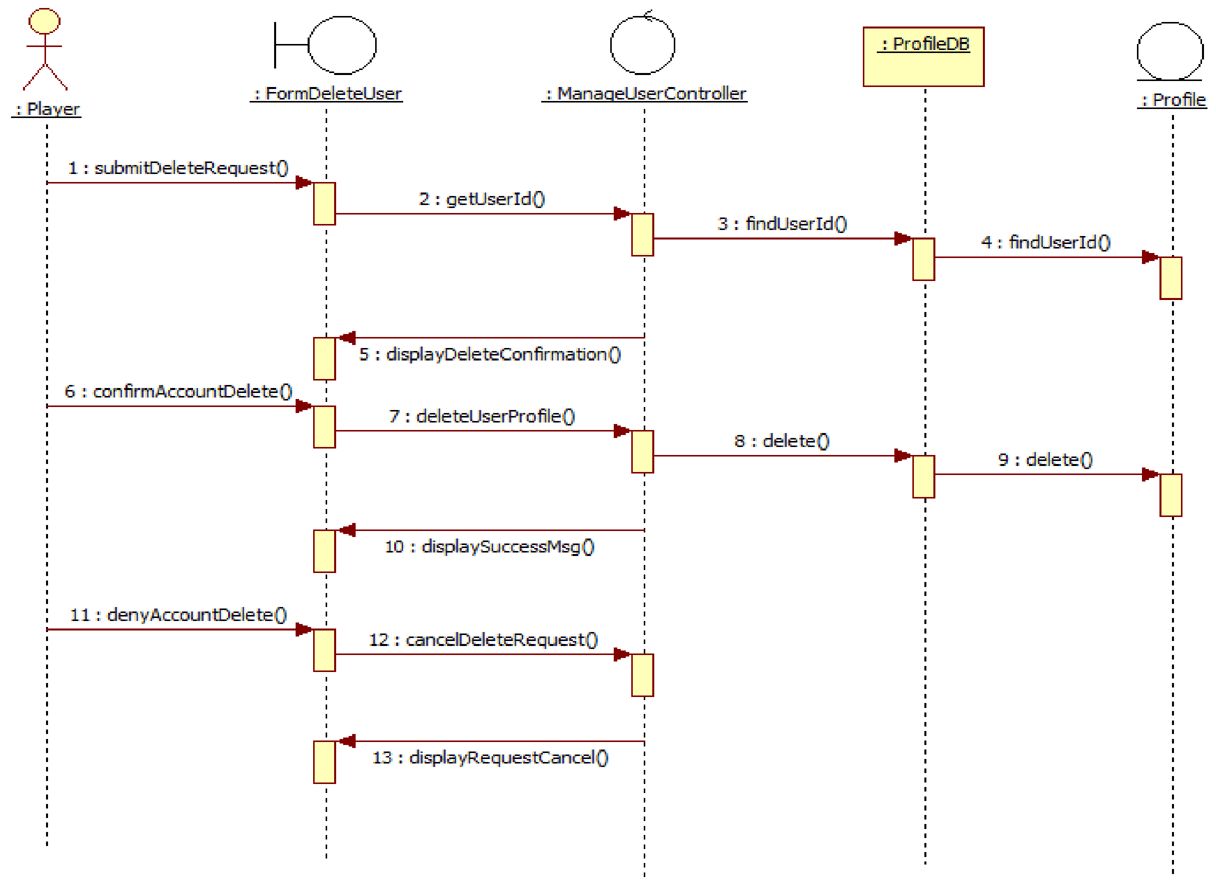
The delete profile use case will require one more validation step from the user as this is an unreversible action. Once the system confirms and retrieves all the profile information, it will display a message to the user at step 6 to give the user one more chance to back out. If the user chooses to proceed with the deletion, his records will be wiped from the Profile entity and he will be directed to the log out screen where he will become a guest user once again.

3.4.2 Sequence Diagram

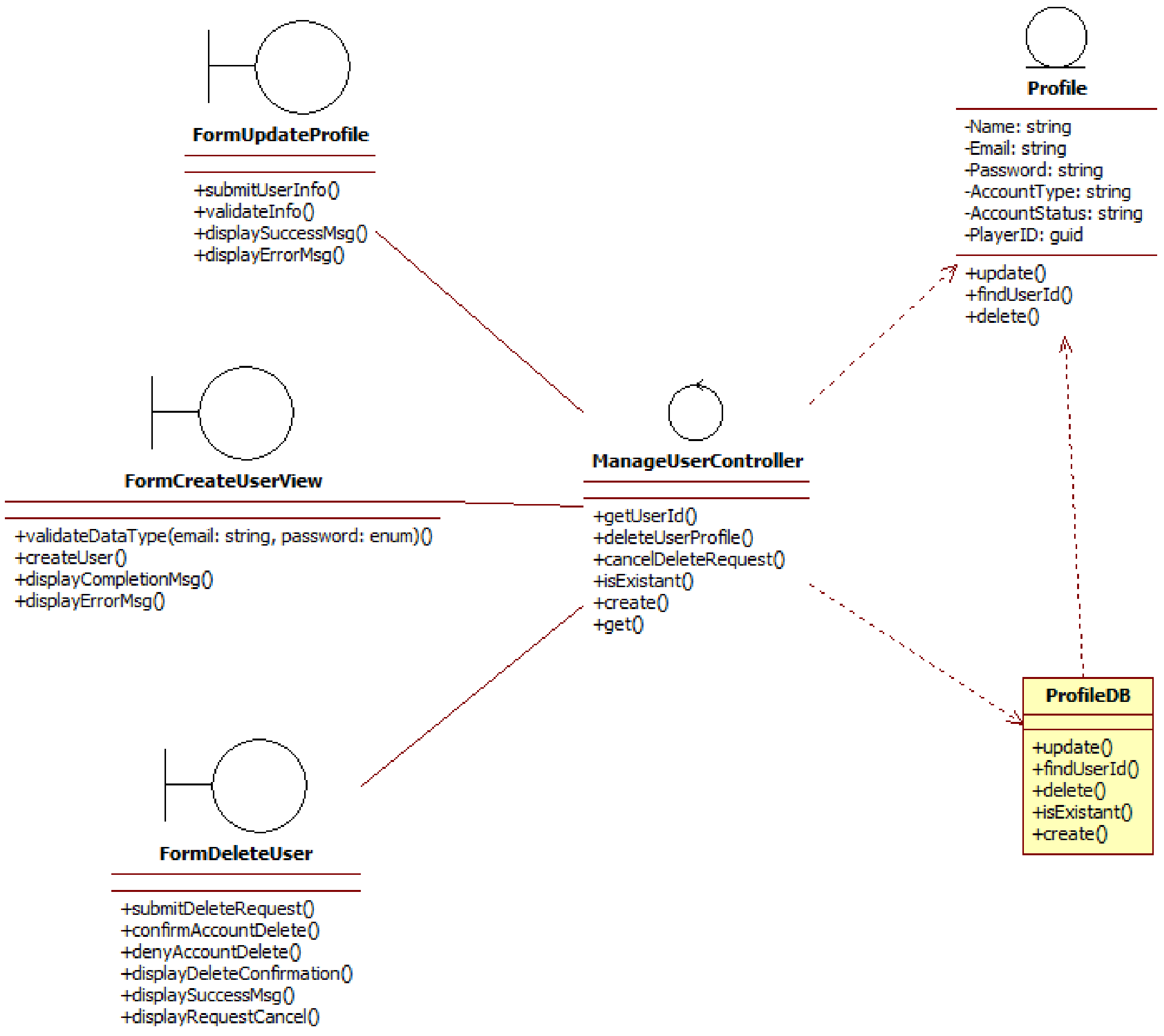
3.4.2.1 Manage profile (update)



3.4.2.2 Manage profile (delete)



3.4.3 Class Diagram



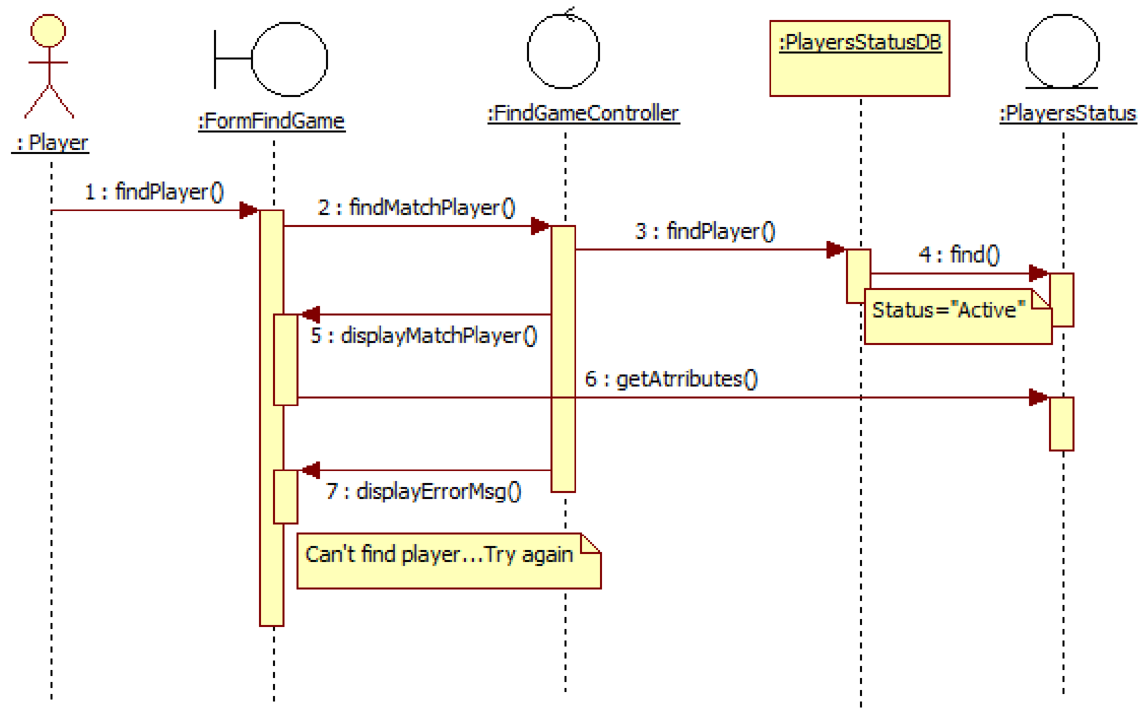
note: Manage profile class diagram is also attached to create profile

3.5 Player→Find Match

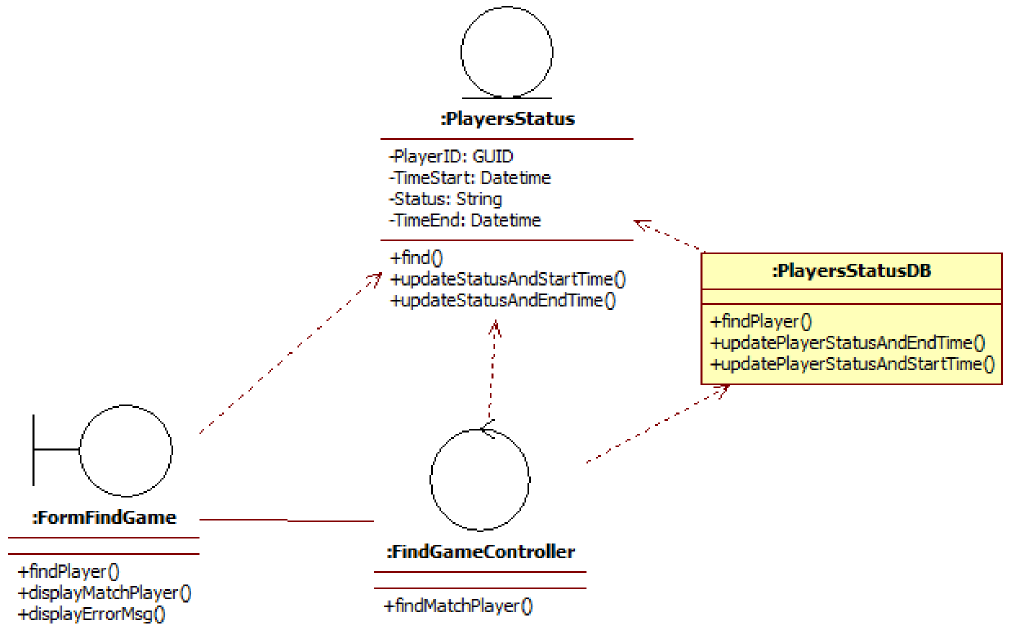
3.5.1 Description

The player finds other available player for playing game. If there is an available player, system will return the information of matched player. If can’t find available player, system will show error message “Can’t find player…..Try again”

3.5.2 Sequence Diagram



3.5.3 Class Diagram



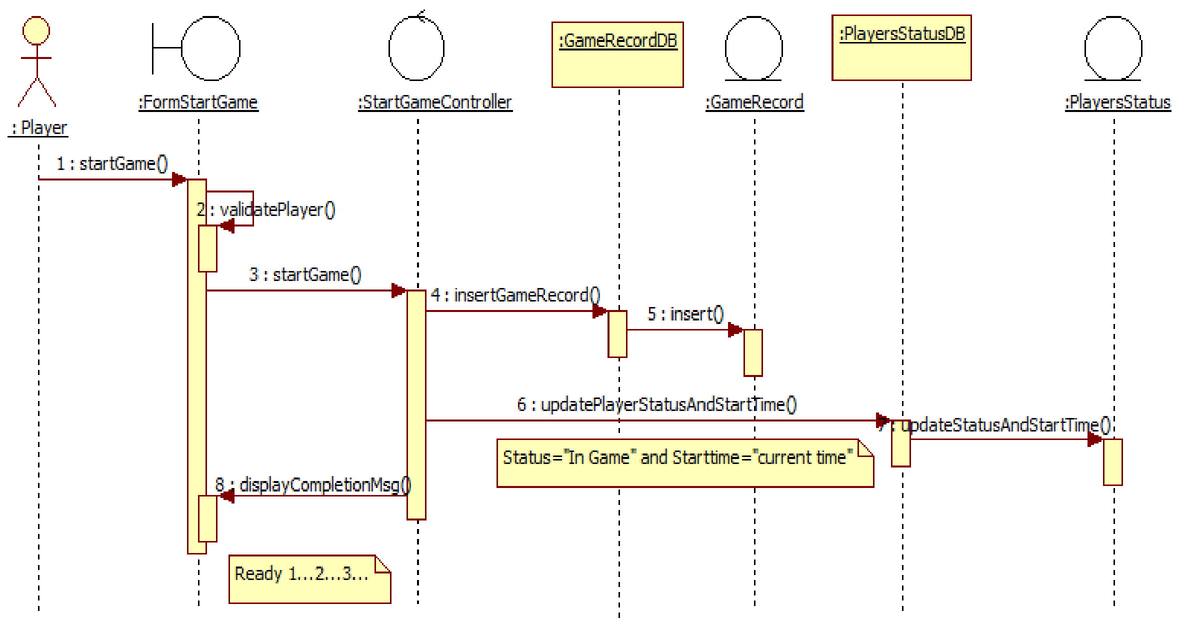
3.6 Player→Start Game

3.6.1 Description

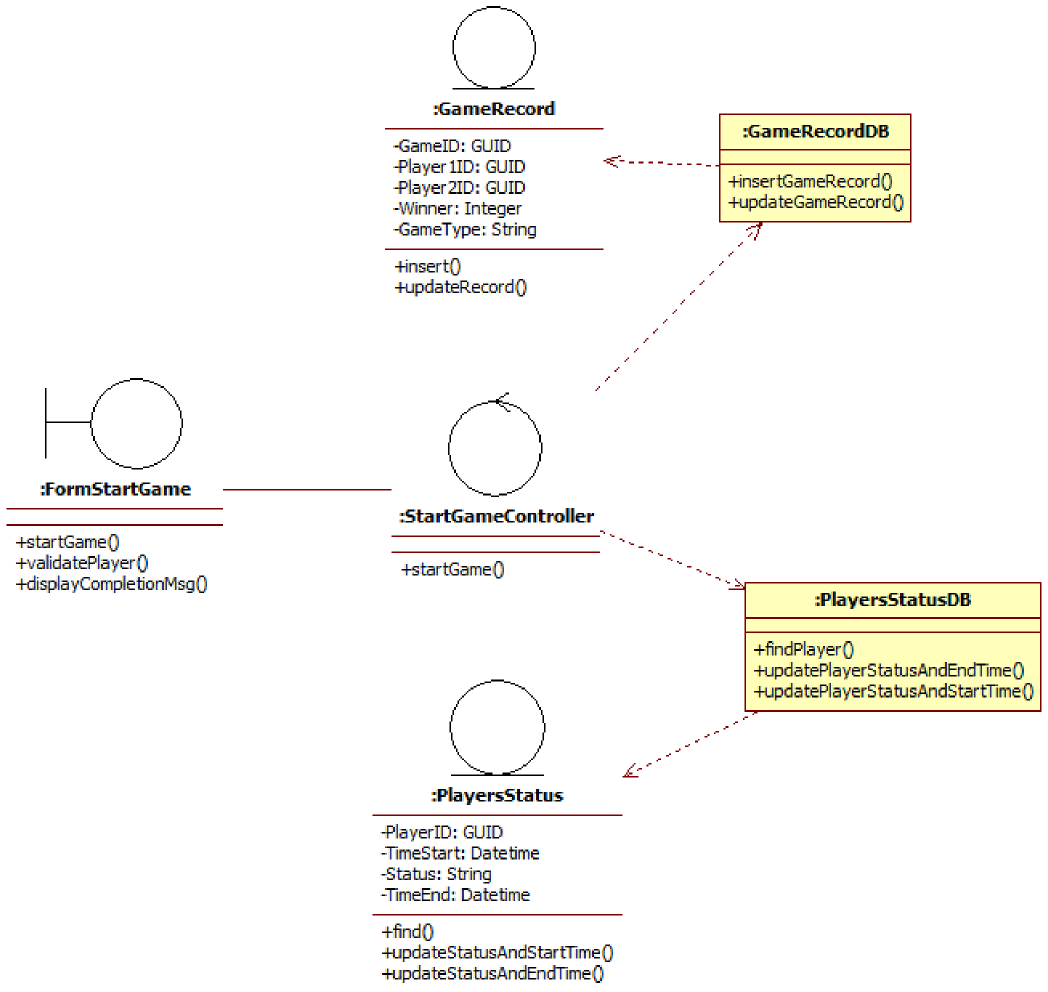
When the player click “start” button, the system will automatically check the number of players. After that system create game record with two player’s ID in “Game Record” table and also update the player’s status into “In Game” and start time into “current time”. If all are successful, the system display completion message “Ready 1…2…3….”

3.6.2 Sequence Diagram

a



3.6.3 Class Diagram

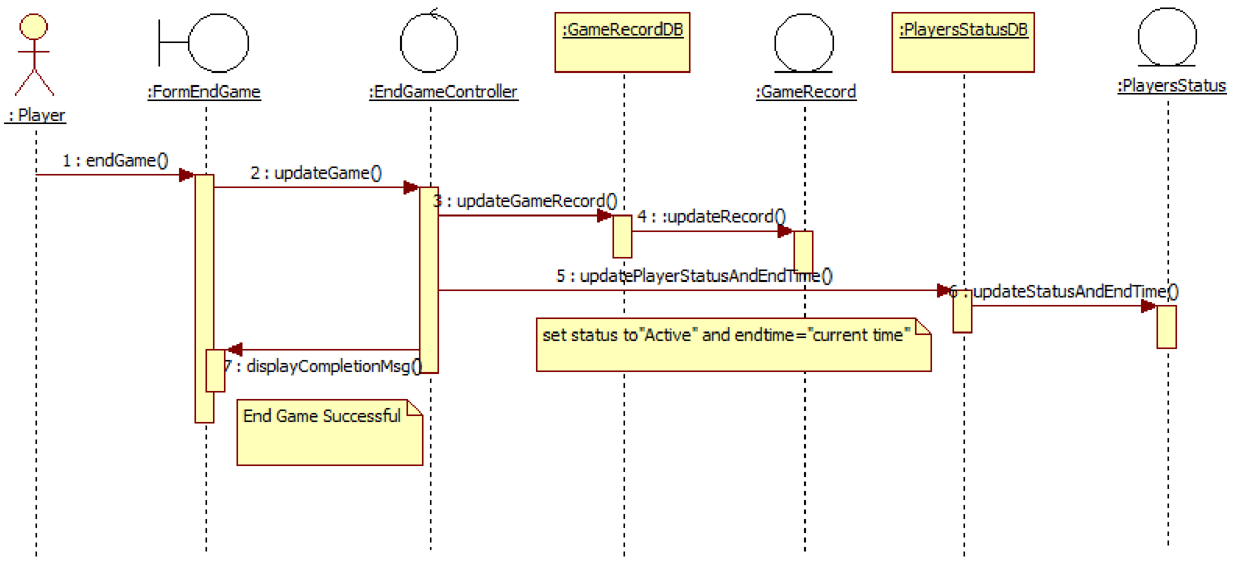


3.7 Player→Leave Game

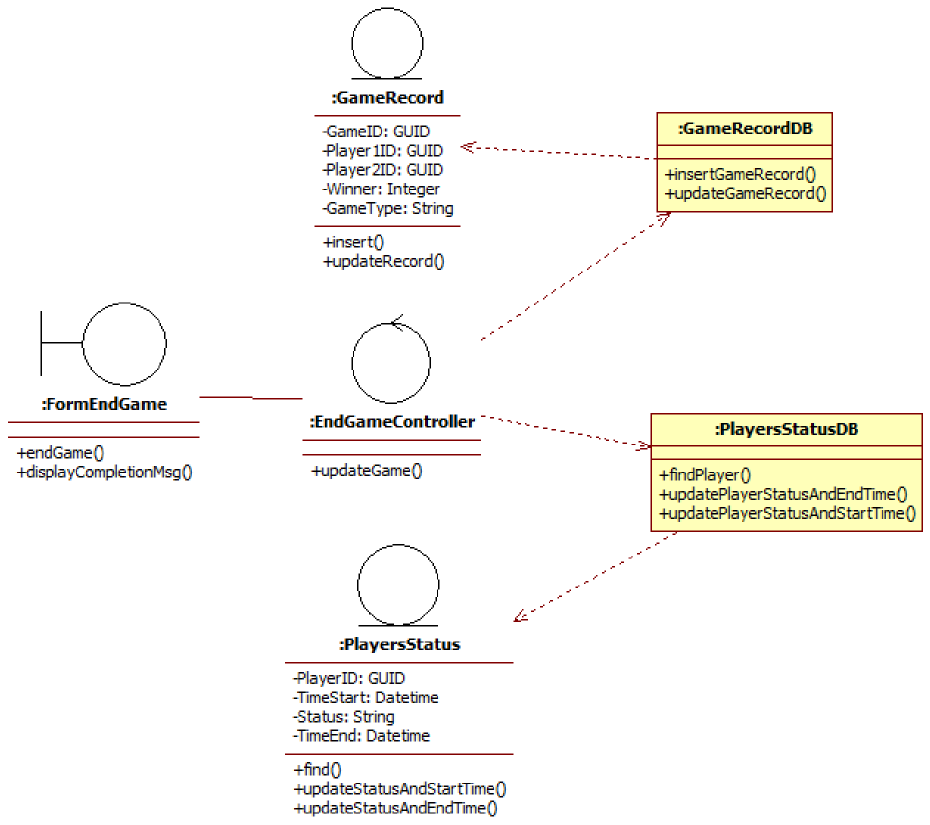
3.7.1 Description

When the player clicks “End” button during game, the system will update winner status in “Game Record” table and also update the player’s status into “Active” and end time into “current time”. If all are successful, the system display completion message “End Game Successful.”

3.7.2 Sequence Diagram



3.7.3 Class Diagram

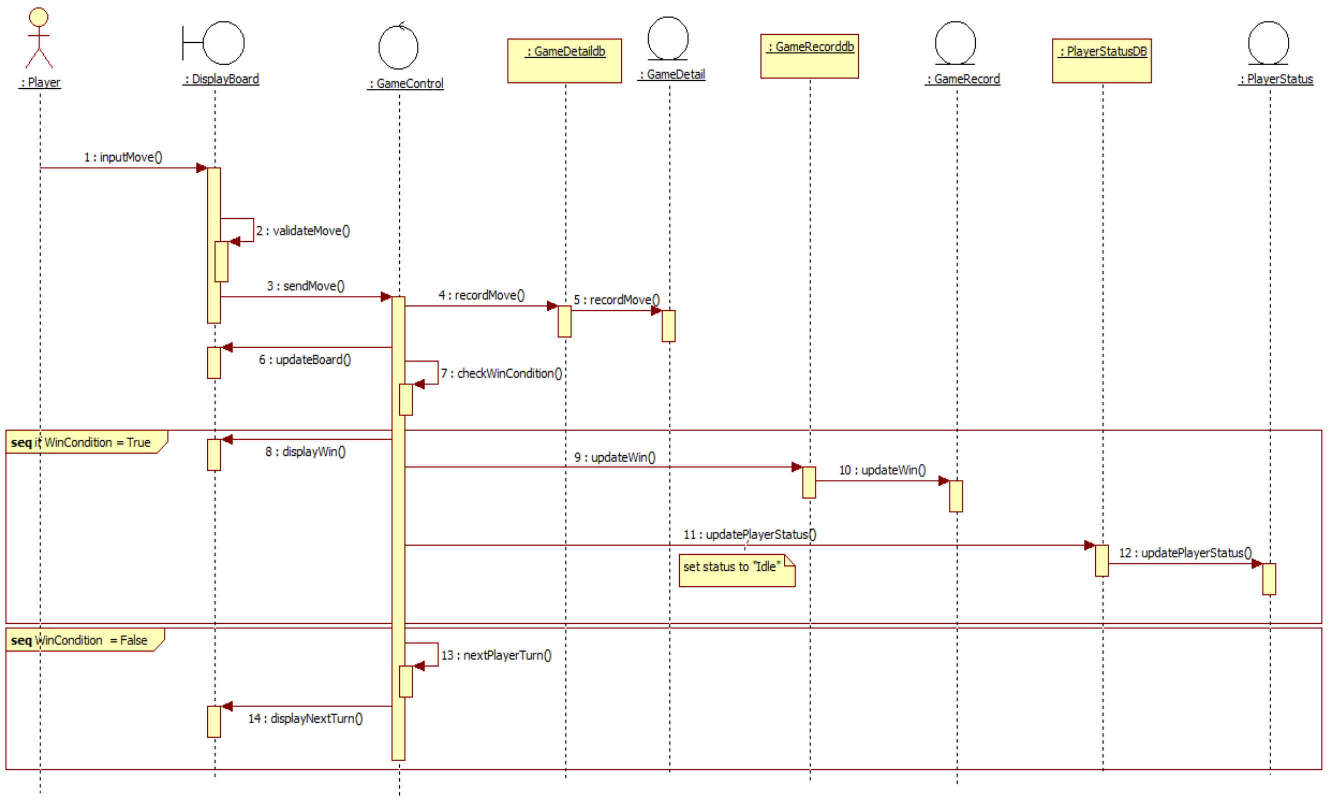


3.8 Player→Play Game

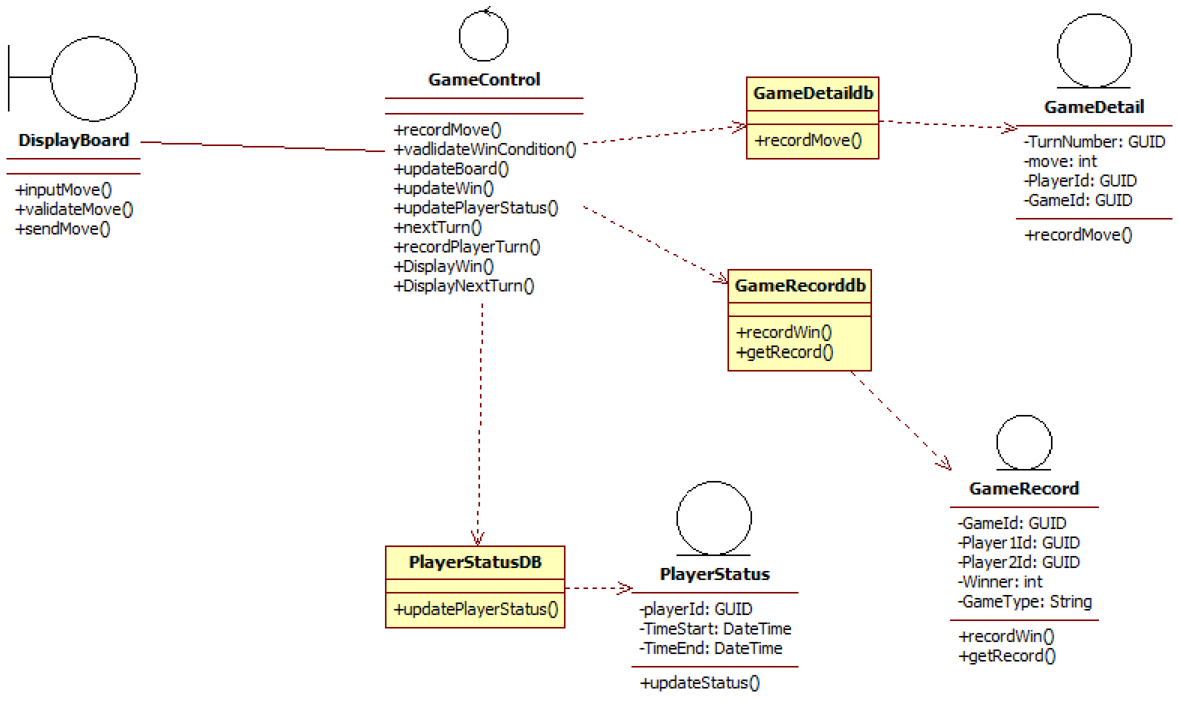
3.8.1 Description

Play Game use case will be used by players to enter the moves and for the game. The controller will also manage the game by checking the validity of the move selected and checking for the Win condition after every move to end the match and record the results of the match.

3.8.2 Sequence Diagram



3.8.3 Class Diagram

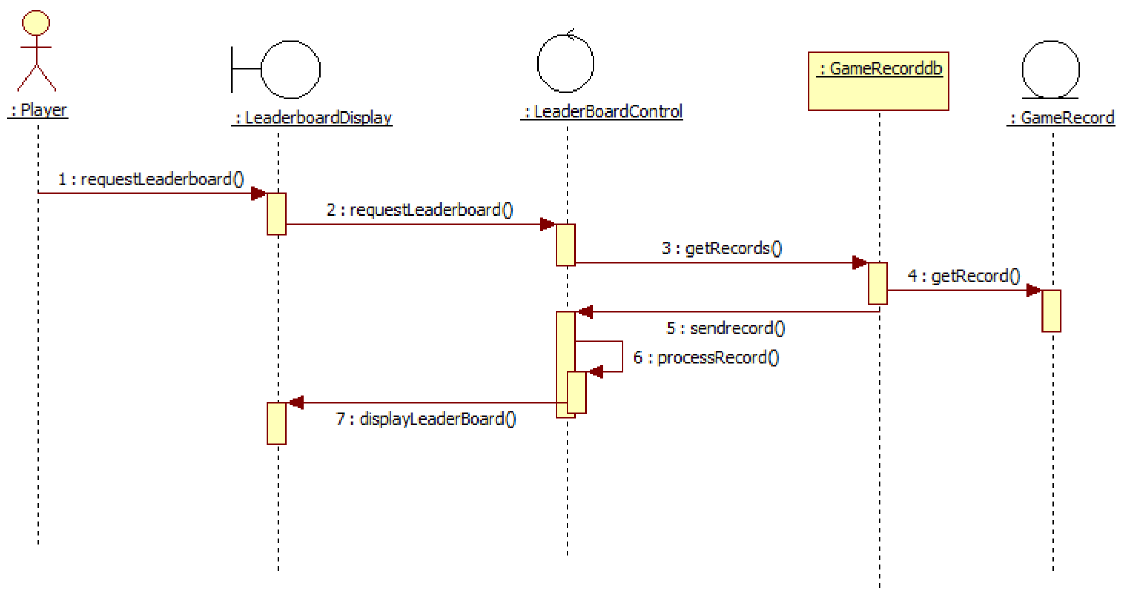


3.9 Player→View Leadership Board

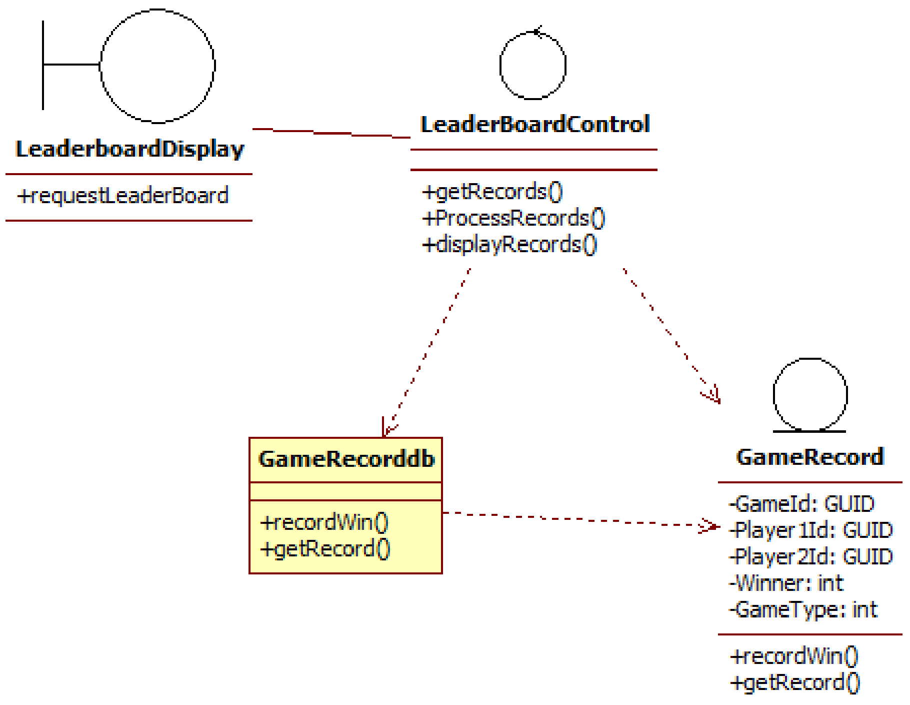
3.9.1 Description

Players will be able to view the leaderboard to see where they stand in terms of winning. Points are awarded based on the wins they have made in a “ranked” game.

3.9.2 Sequence Diagram



3.9.3 Class Diagram

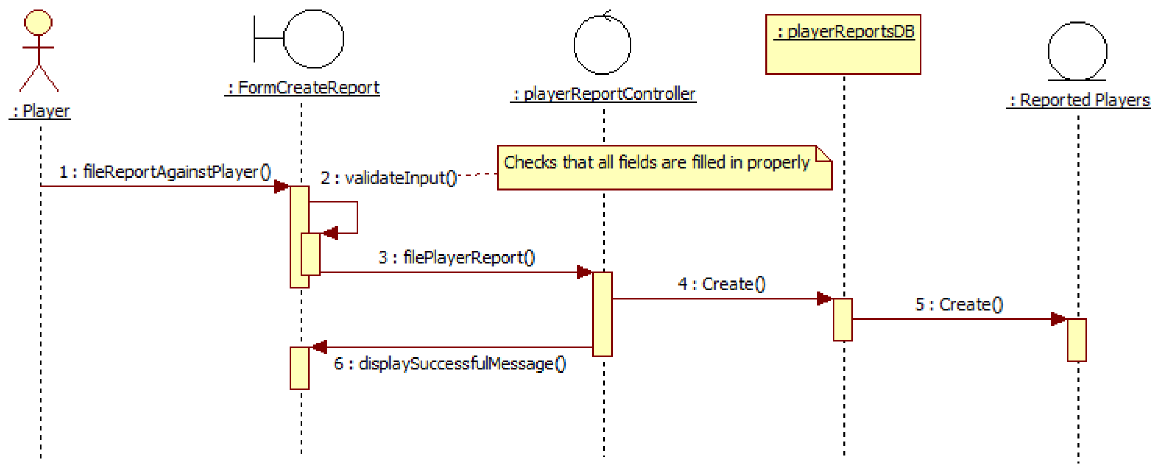


3.10 Player→Report Player

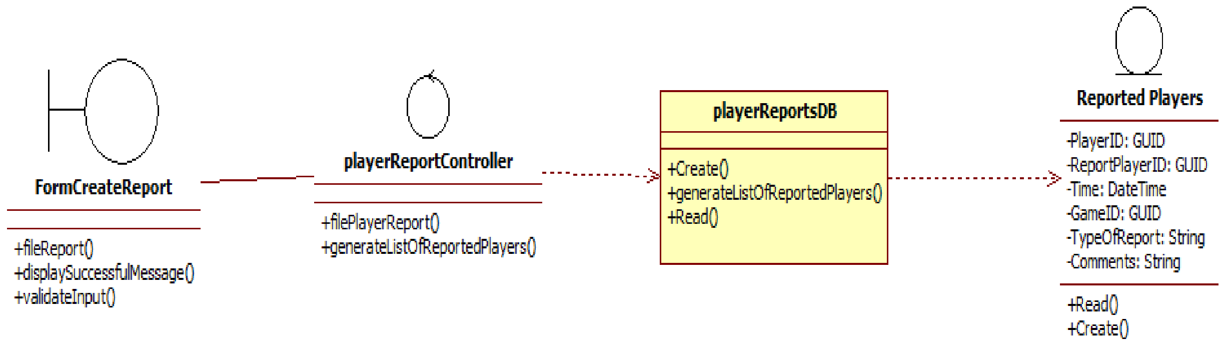
3.10.1 Description

This use case allows players to report other players for bad behavior (e.g. abusive chat) for moderation by the admin.

3.10.2 Sequence Diagram



3.10.3 Class Diagram



3.11 Player→Join Private Game

3.11.1 Description

This use case is for joining a session of private game (unranked). This use case assumes that the player has already successfully logged in.

The player requests a list of private games available for joining. If there are available games, the system will return a list of games to the player to choose from.

Player status will be updated as “Looking for Game”.

After selecting a game and joining it, player status will be updated from “Looking for Game” to “In Game”. The system displays a completion message to inform the user that they have joined the game successfully.

The system returns an error message to the player in two situations:

1. Step 8 in Sequence Diagram: When there are no games available to join. System returns an error message saying that there are no available games to join.

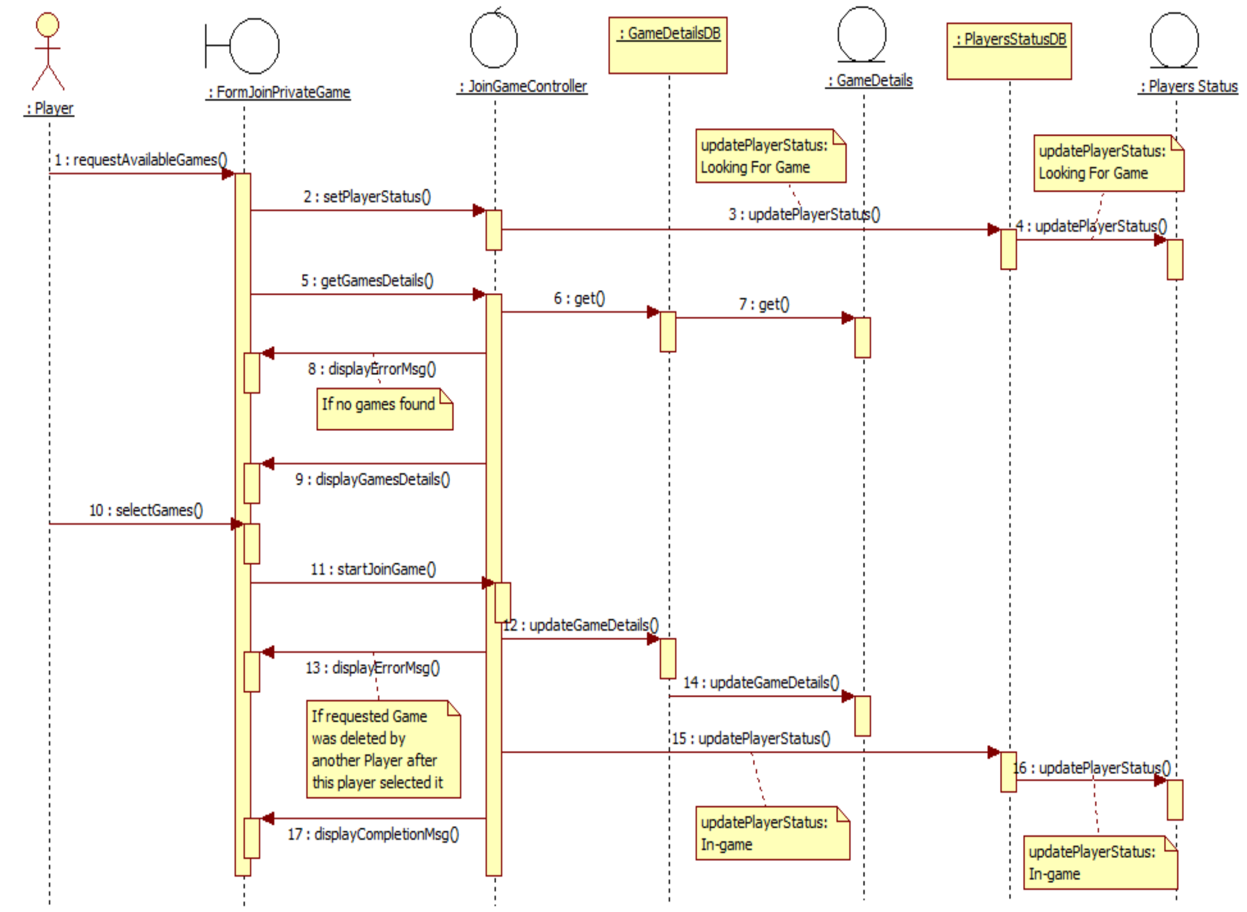
2. Step 13 in Sequence Diagram: An available game has been deleted by another player when a player selected to join it.

An example of this scenario in Step 13 is as follows:

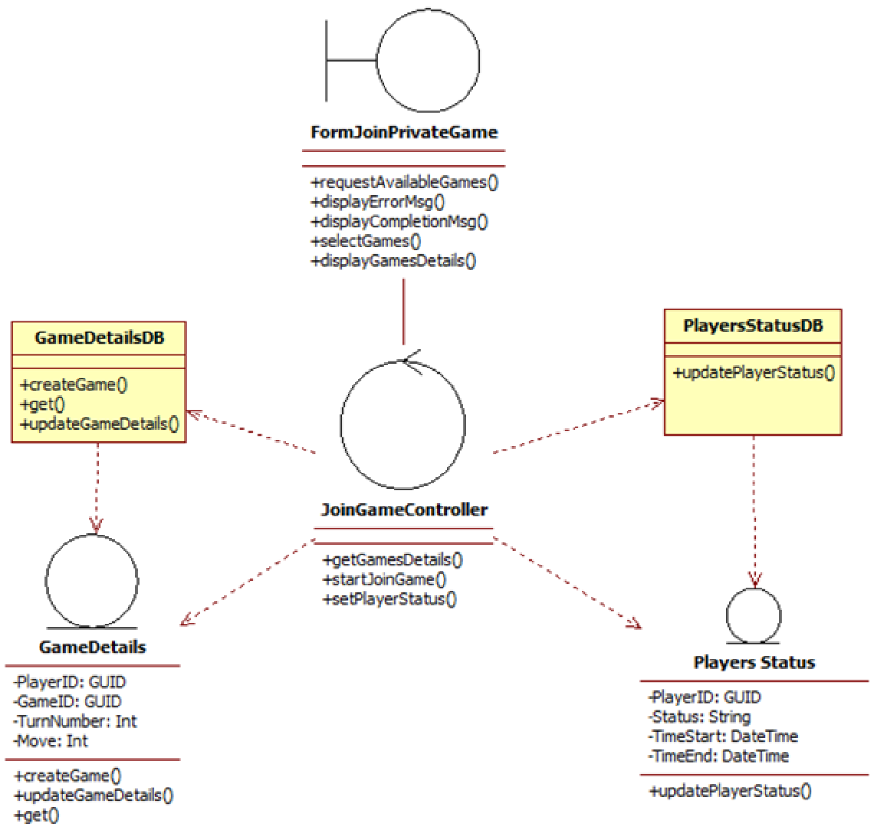
1. Player A selected to join a game which was created by Player B. However, this game was deleted by Player B after it appeared on the list of games available for joining.

2. When this happens, system returns an error message saying that the game has been deleted by another player and is not available to join.

3.11.2 Sequence Diagram



3.11.3 Class Diagram



3.12 Player→Chat With Other Player

3.12.1 Description

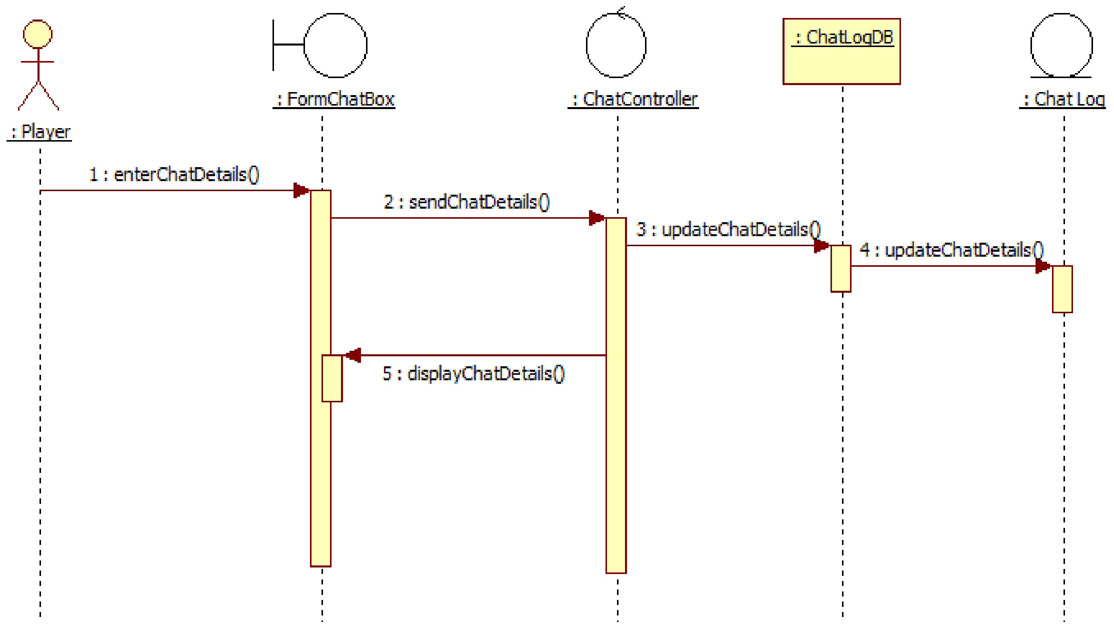
This use case assumes that the player has already successfully logged in.

The player enters chat message into FormChatBox. FormChatBox sends the chat details to the ChatController, which then sends the chat details to the ChatLog Data Broker.

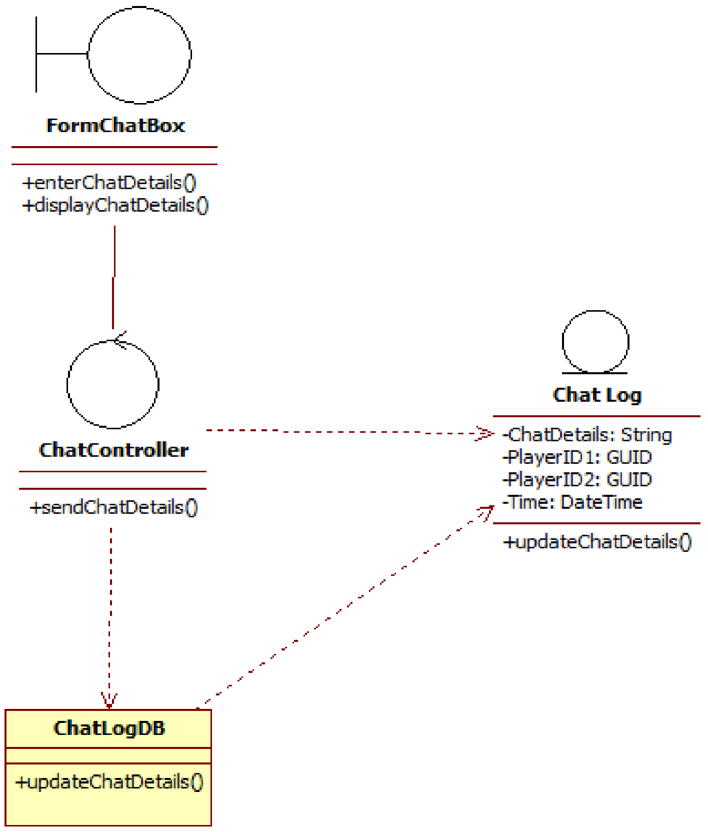
The ChatLog Data Broker updates the Chat Log entity with the new chat details.

The updated chat message is then shown on the system to the user.

3.12.2 Sequence Diagram



3.12.3 Class Diagram



3.13 Player→Create Private Game

3.13.1 Description

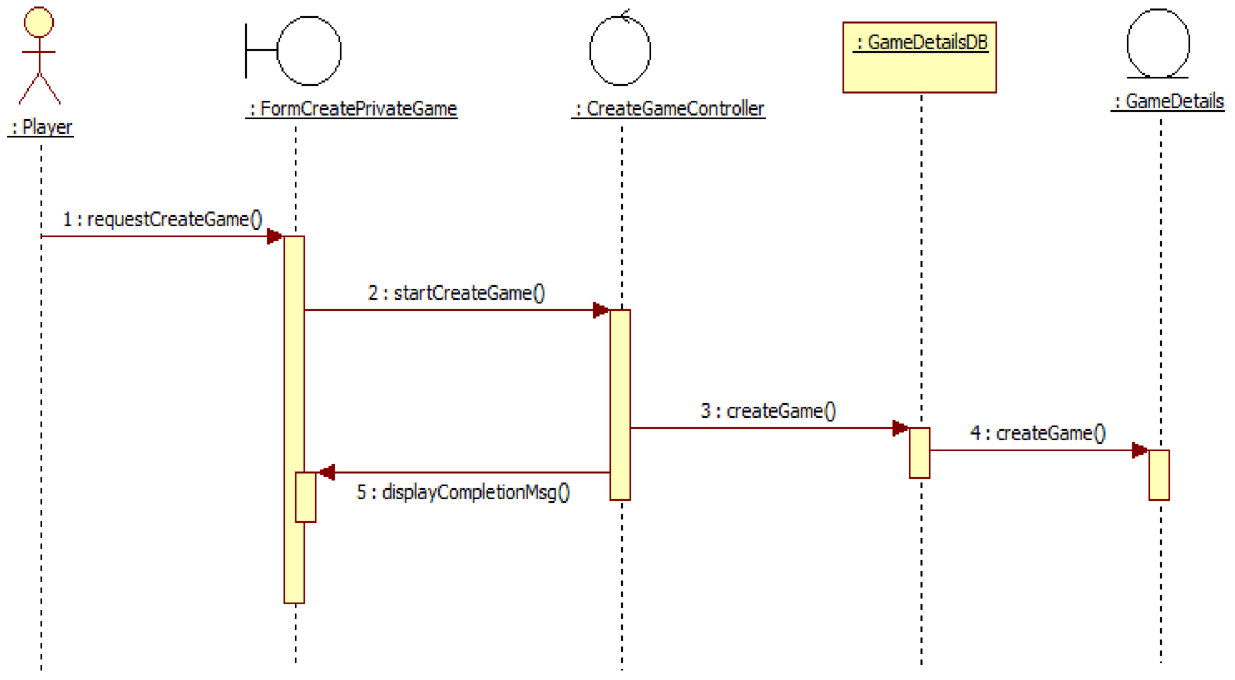
This use case is for creating a session of private game (unranked). This use case assumes that the player has already successfully logged in.

The player requests to create a private game in the FormCreatePrivateGame boundary. The boundary then sends the request to the CreateGameController, which subsequently updates the GameDetailsDB Data Broker.

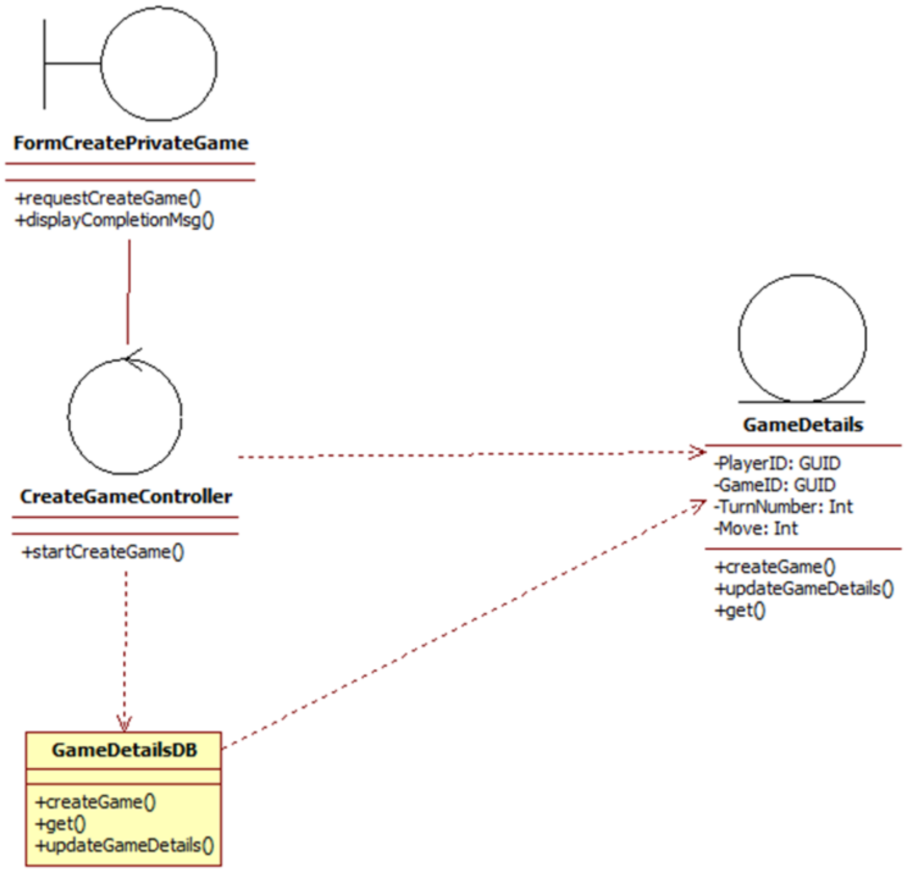
The Game DetailsDB Data Broker updates the GameDetails entity with the new game's details.

After game is successfully created, a completion message with the details of the created game is returned to the player to say that the game has been created.

3.13.2 Sequence Diagram



3.13.3 Class Diagram



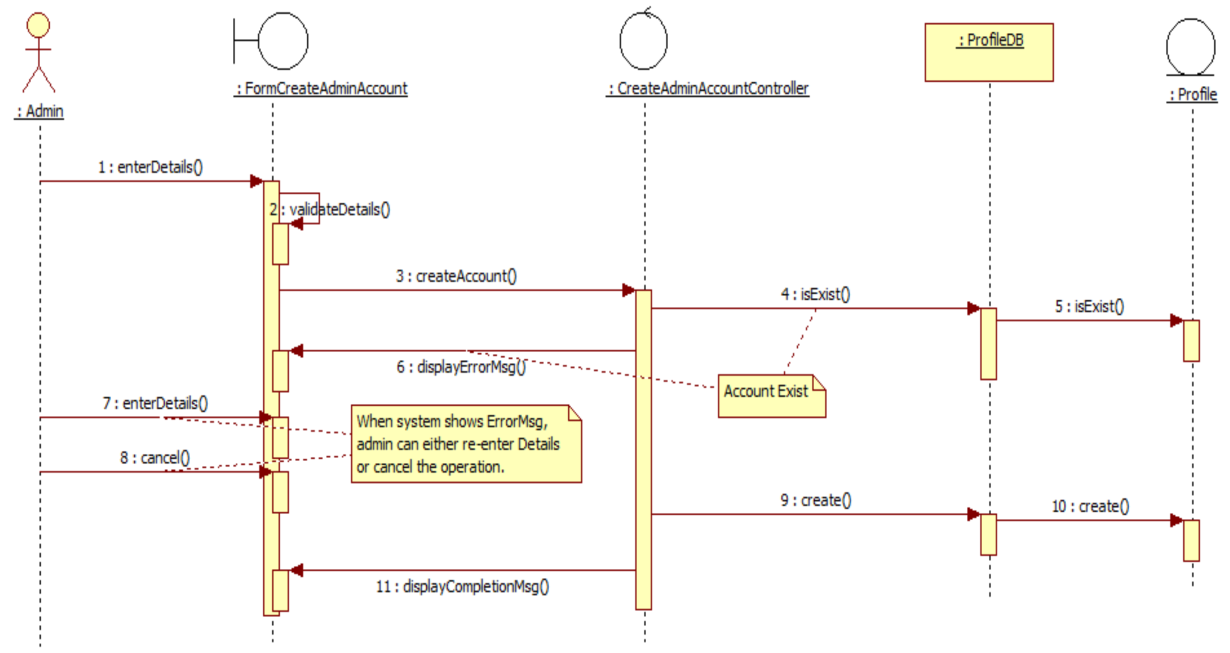
3.14 Admin→Create Admin Account

This use case is for Admin to create an Admin Account. Admin enters details for creating an account in the boundary, then system validates the data format. System checks if the account exists in current profile records, from here the flow have two options.

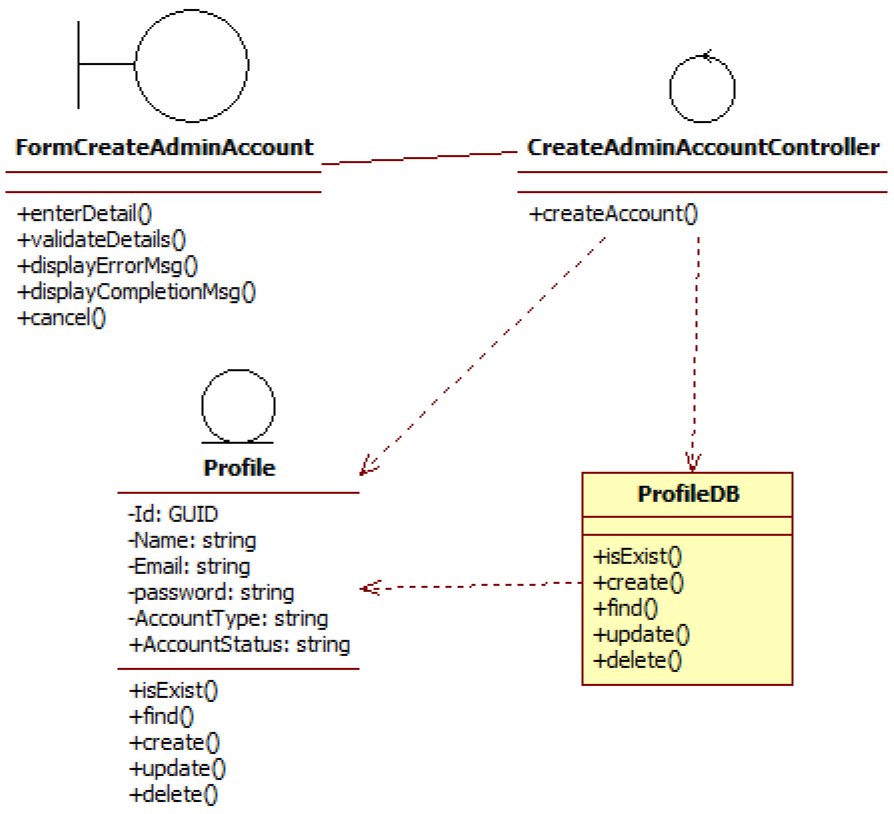
If the account does not exist, system will create a new account and finally display completion message.

If the account already exists in database, system will display error message, the admin can either re-enter details or just cancel the operation at which the use case terminates.

3.14.2 Sequence Diagram



3.14.3 Class Diagram



3.15 Admin→Manage Player Profile

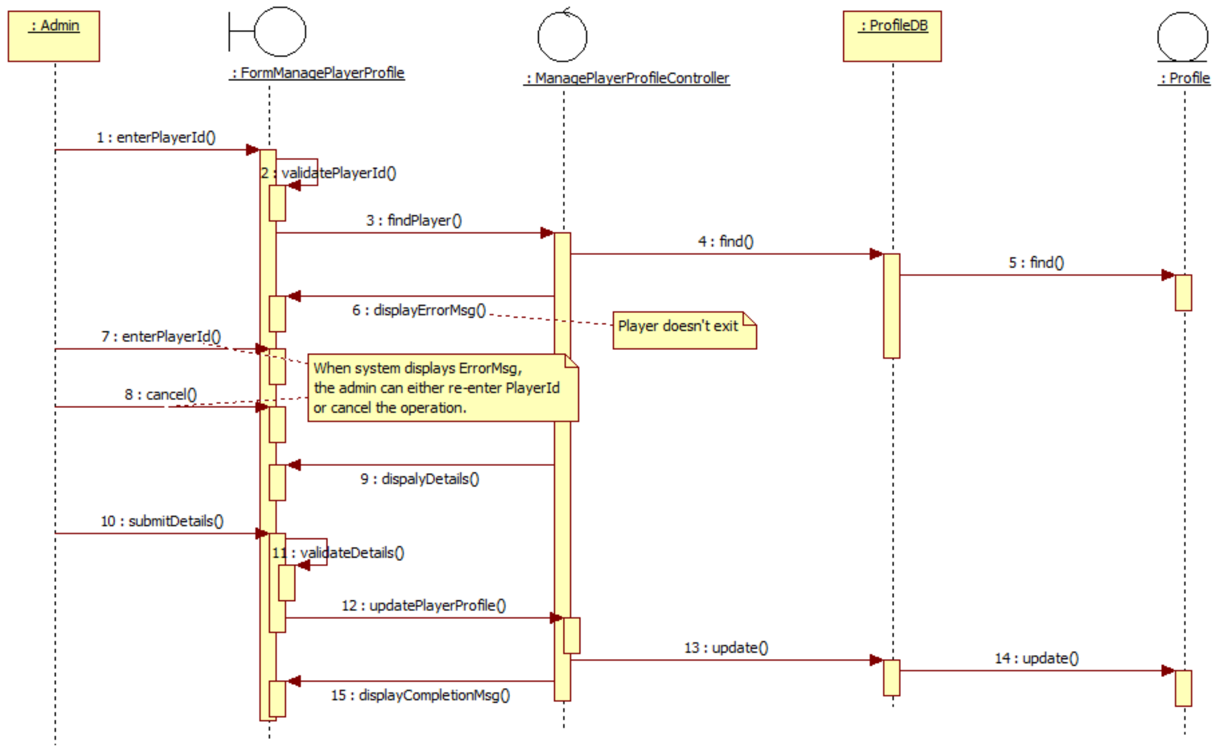
3.15.1 Description

This use case is for Admin to manage player's profile. The flow starts from Admin entering player Id, after which system will validates the data format. Then system retrieves the player profile, from here the flow has two options.

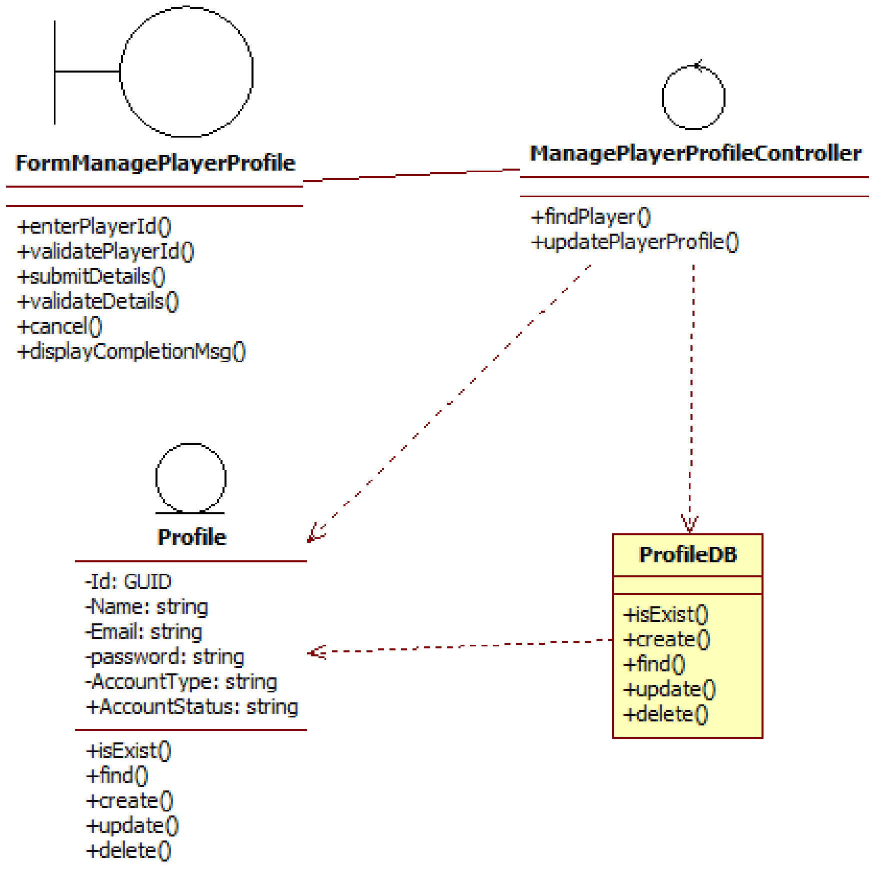
If the player can be found in database, system will display player details to the boundary. Admin modifies player profile details, then submits the modifications. System updates the player details into database.

If the player cannot be found in database, system will show error message, and the admin can either re-enter player id or just cancel the operation at which the use case terminates.

3.15.2 Sequence Diagram



3.15.3 Class Diagram



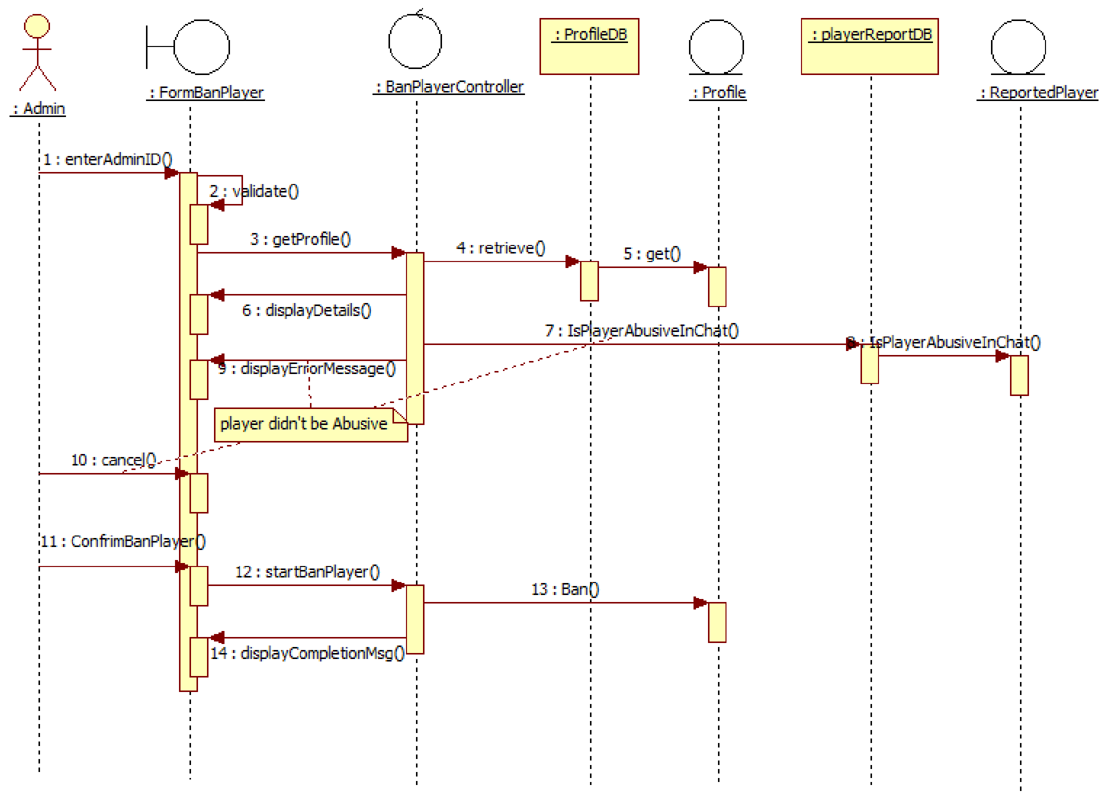
3.16 Admin→ Ban Player

3.16.1 Description

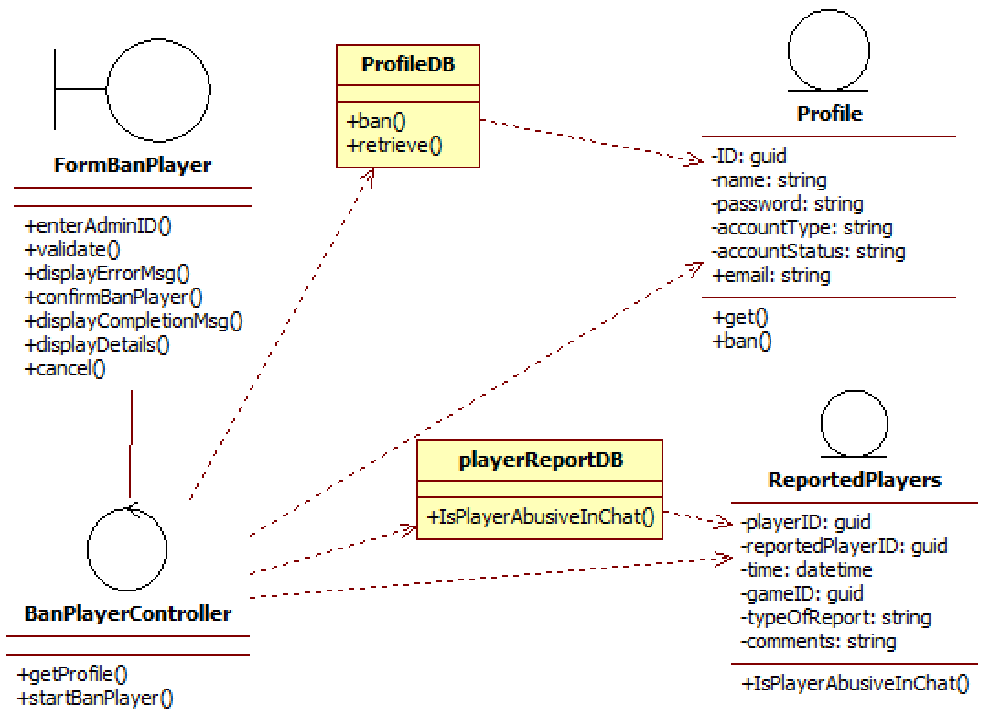
This use case allows admin to ban player account if the player is abusive in chat box. Admin enters the adminID and system validates the adminID. System retrieves player details from Reported Player entity and displays player details. Once admin confirms the ban operation, system bans the player and displays completion message.

In step 7 to 9, if player was not abusive from the chat log, the system displays an error message, “Player is not abusive”. In step 10, if player was not abusive from the chat log, admin can cancel the operation.

3.16.2 Sequence Diagram



3.16.3 Class Diagram

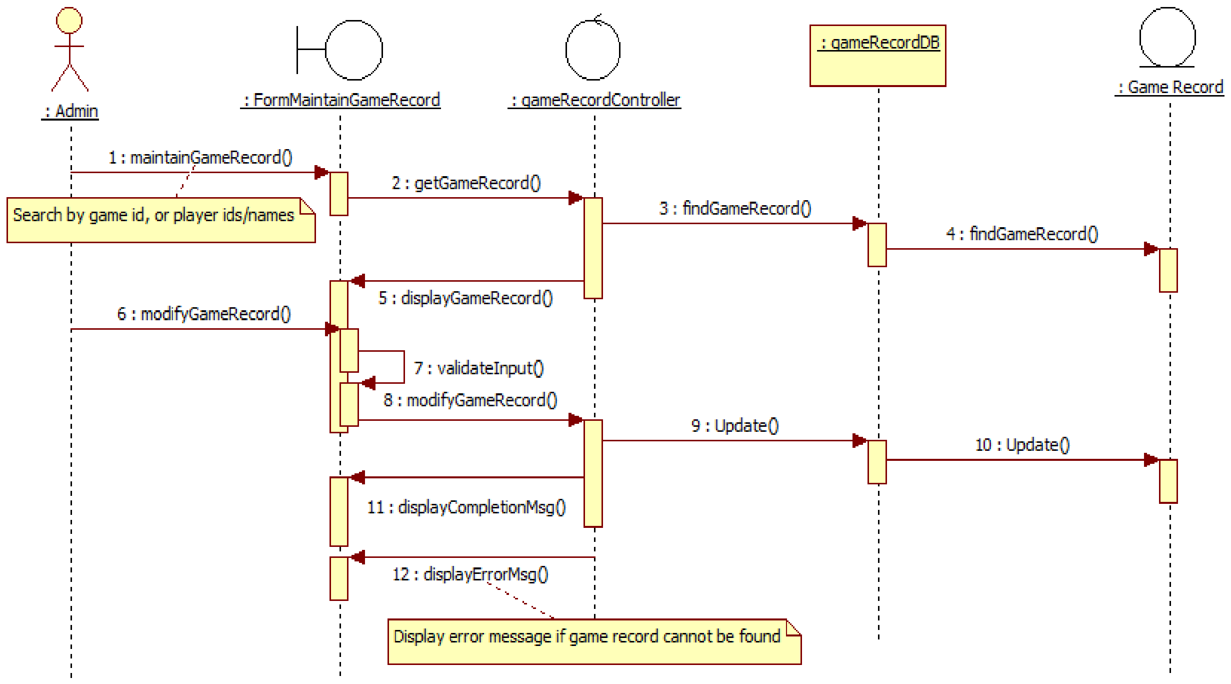


3.17 Admin→Maintain Game Record

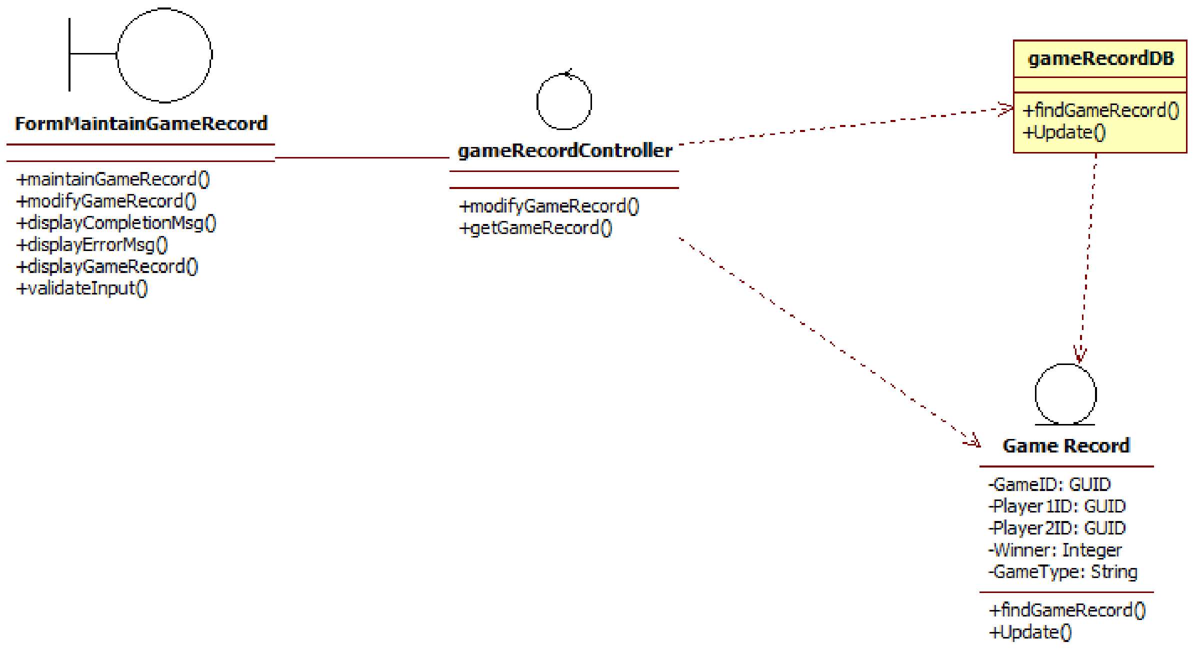
3.17.1 Description

This use case allows game admins to update game records in the system. This is used in scenarios such as when a player hacking is discovered, and those game records need to be removed from the system for overall fairness.

3.17.2 Sequence Diagram



3.17.3 Class Diagram

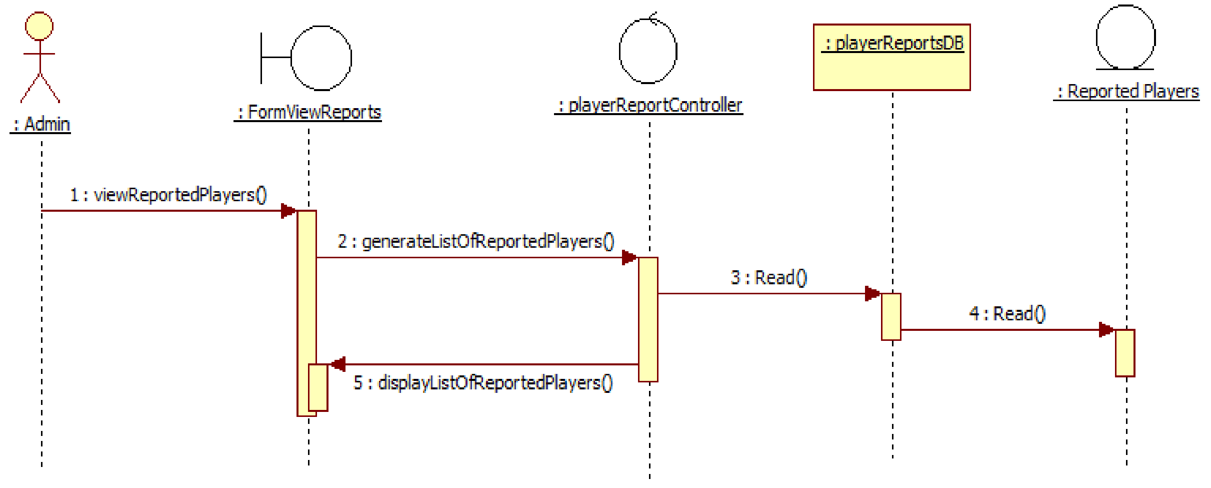


3.18 Admin→View Reported Players

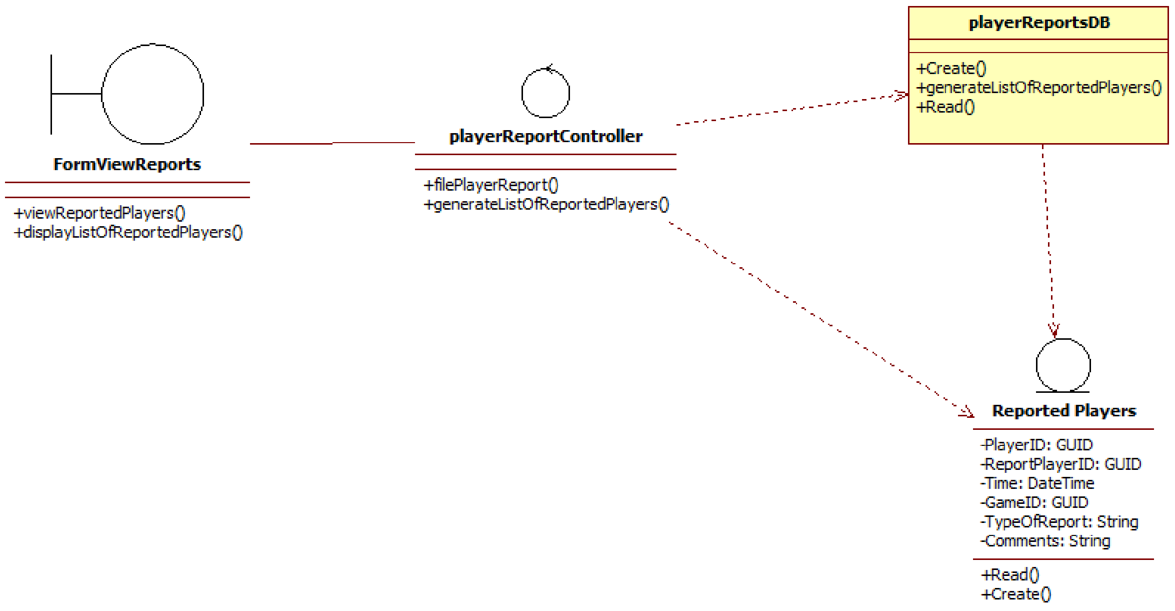
3.18.1 Description

This use case allows the game admin to view reports submitted by players.

3.18.2 Sequence Diagram



3.18.3 Class Diagram

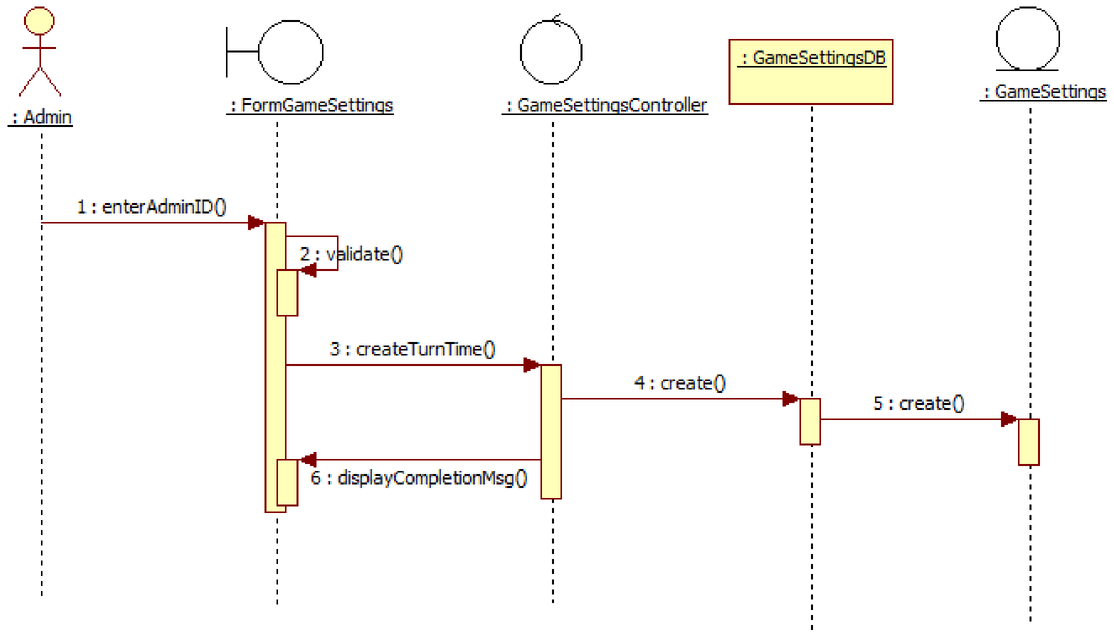


3.19 Admin→Manage Game Settings

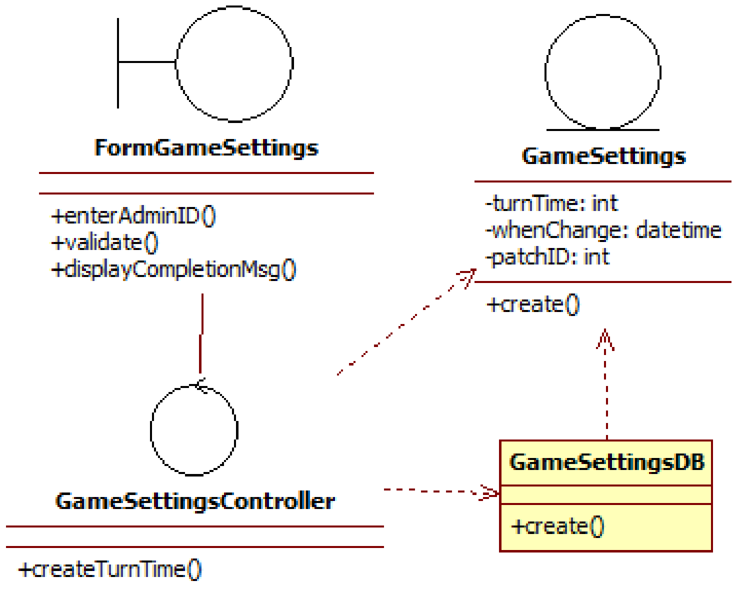
3.19.1 Description

This use case allow admin to manage game turn time in order to separate games into easy or hard mode. First, admin enters the adminID and system validates the adminID. System creates turn time of the game and displays completion message.

3.19.2 Sequence Diagram

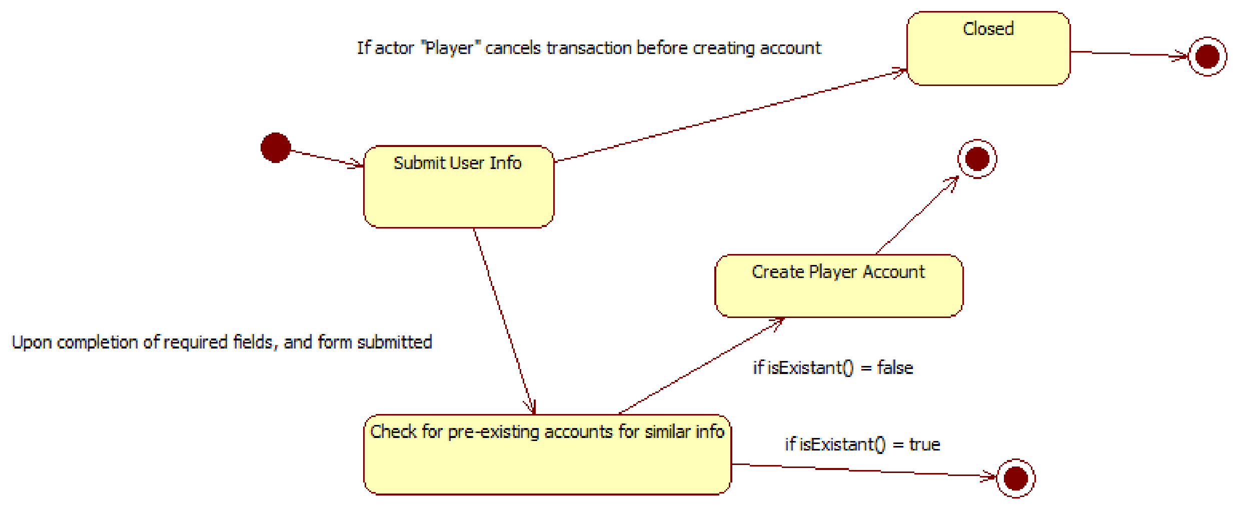


3.19.3 Class Diagram



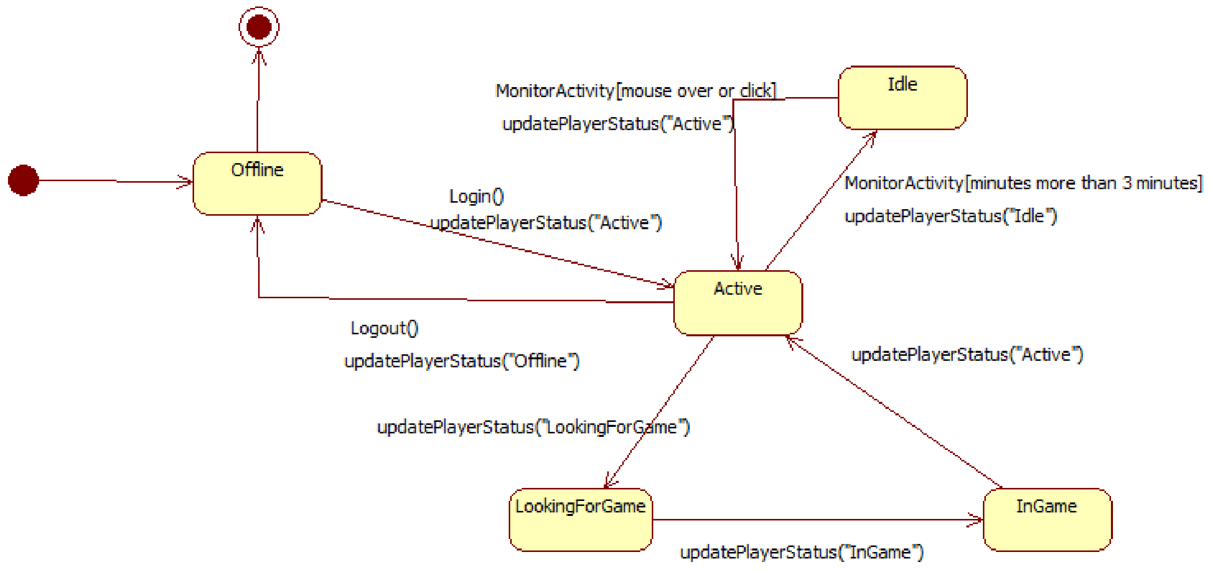
4. Design Model

4.1 Create Account Statechart Diagram



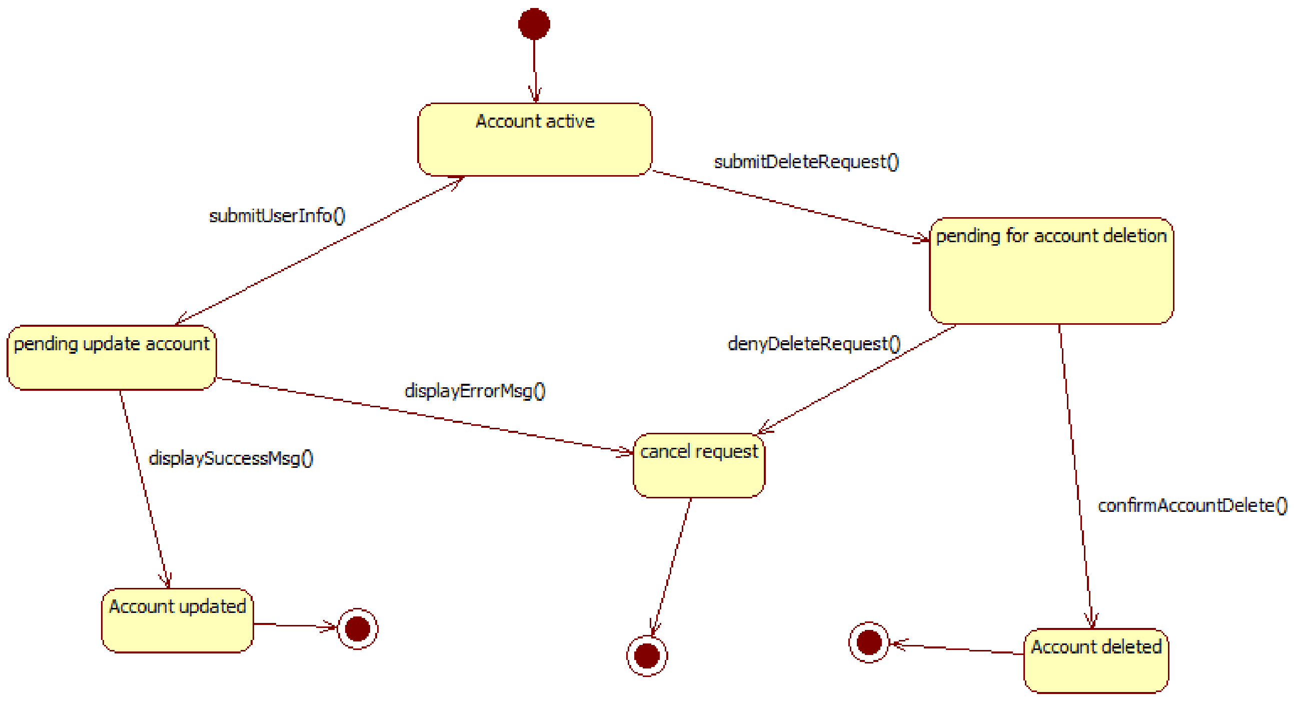
If actor "Player" cancels transaction before creating account
Closed
Submit User Info
Create Player Account
Upon completion of required fields, and form submitted
if isExistant() = false
Check for pre-existing accounts for similar info
if isExistant() = true

4.2 Player Status Statechart Diagram



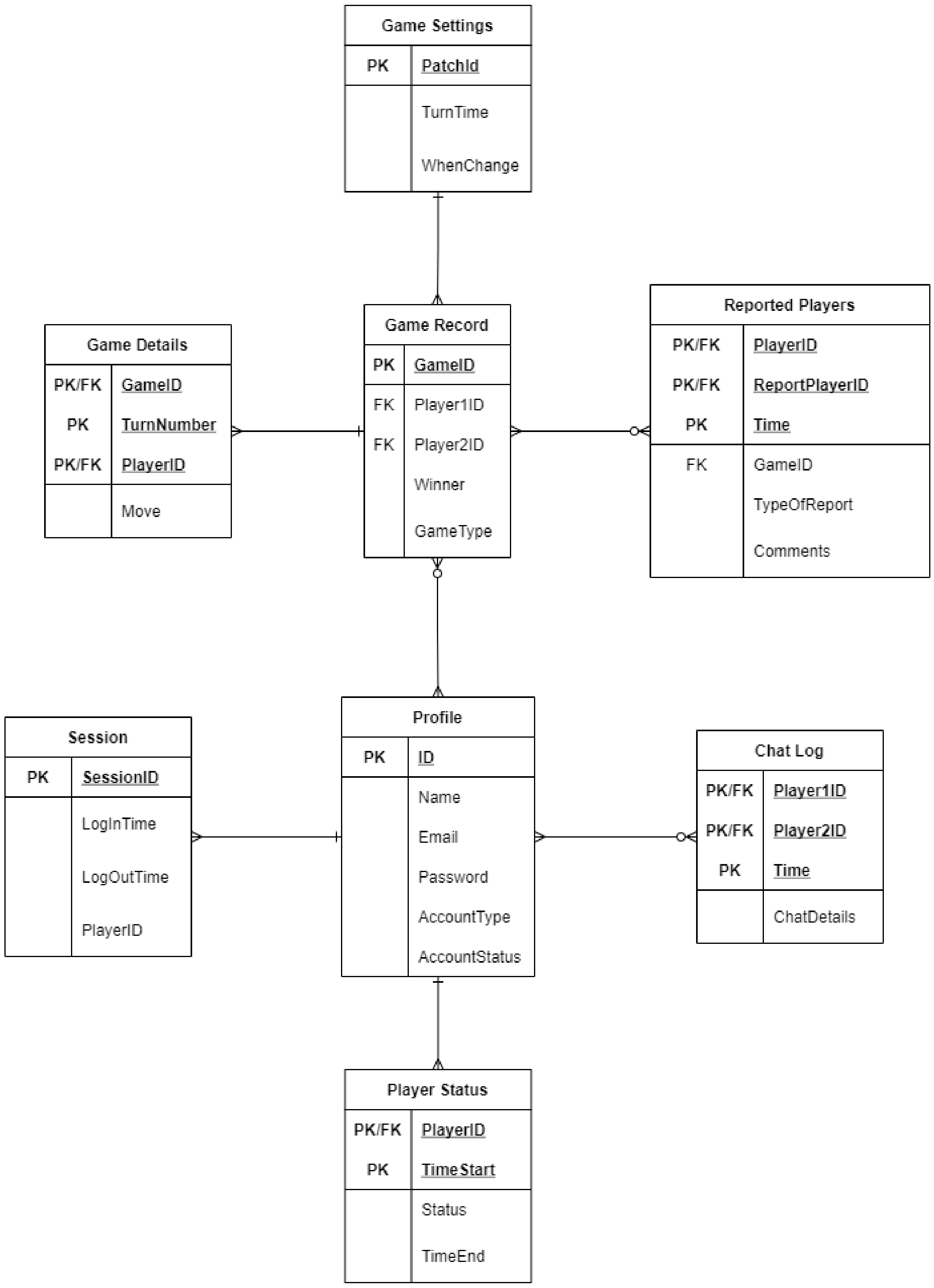
MonitorActivity[mouse over or click]
updatePlayerStatus("Active")
Idle
Offline
Login()
updatePlayerStatus("Active")
MonitorActivity[minutes more than 3 minutes]
updatePlayerStatus("Idle")
Active
Logout()
updatePlayerStatus("Offline")
updatePlayerStatus("Active")
updatePlayerStatus("LookingForGame")
LookingForGame
InGame
updatePlayerStatus("InGame")

4.3 Manage Profile Statechart Diagram



5. Data Model

5.1 ERD



5.2 Data Dictionary

Entity	Attribute Name	Type	Length	Definition
Game Record	GameID	GUID	32	Game ID, unique for each game played
	Player1ID	GUID	32	PlayerID for player going first this game
	Player2ID	GUID	32	PlayerID for player going second this game
	Winner	N	1	Player who won, 1 for player1, 2 for player2, 0 for draw
	GameType	A	16	Game Type, currently either Private or Ranked
Game Details	GameID	GUID	32	Game ID, unique for each game played
	TurnNumber	N	3	Turn number within the game
	PlayerID	GUID	32	PlayerID for player making the turn
	Move	N	1	Column number where the player put his token
Game Settings	PatchID	N	3.1	Patch ID, new for every change to the game
	TurnTime	N	4	Time per turn of Connect 4 game
	WhenChange	DT	24	Date and time when game settings were changed
Profile	ID	GUID	32	ID, unique for each person accessing the game, whether as a player or admin
	Name	A	50	Name of player
	Email	A	50	Email address of player
	Password	A	50	Hashed password of player
	AccountType	A	16	Whether account is a player or admin account
	AccountStatus	A	16	Whether account is active or banned
Reported Players	PlayerID	GUID	32	PlayerID, unique for each player
	ReportPlayerID	GUID	32	PlayerID for player who made the report

	Time	DT	24	Date and time report was made
	GameID	GUID	32	Game ID for the game when report was made
	TypeOfReport	A	20	Type of report made
	Comments	A	255	Player-added comments regarding his report
Player Status	PlayerID	GUID	32	PlayerID, unique for each player
	TimeStart	DT	24	Date and time player started the status
	Status	A	20	Player status (in game, idle, offline, looking for game, active)
	TimeEnd	DT	24	Date and time player exited the status
Chat Log	Player1ID	GUID	32	PlayerID for player sending the chat message
	Player2ID	GUID	32	PlayerID for player receiving the chat message
	Time	DT	24	Date and time chat message was sent
	ChatDetails	A	500	Chat message which was sent
Session	SessionID	GUID	32	Session ID for player
	LogInTime	DT	24	Date and time player logged in
	LogOutTime	DT	24	Date and time player logged out
	PlayerID	GUID	32	PlayerID, unique for each player

Note: DT = DateTime, A = Alphanumeric, N = Number