Use Cases

Use Case id:	EU-001
Use Case Name:	Registers to System
Overview:	Allows a user to register for a game system
Primary Actors:	New User [initiator][Primary]
Secondary Actors:	None
Properties:	**Performance**: Registration must happen immediately and allow user to start a new game right away **Security:** Email address entered by user and user-set password must both be secure and inaccessible to the public **Other:**
Preconditions:	User must already have an email address that has not yet been used. User must have a nickname that has not yet been used.
Main Flow:	A user will enter their email address, a nickname, and a password and click a button to sign up for an account. If the email and nickname are not already in use, the account will be created.
Postconditions:	A user will have an account and will be able to play games.
Alternative Flows:	If a user's selected email or nickname have been previously used, the user will be notified and asked to use a different email or nickname depending on if one or both have been previously taken. The account will not be created.

Use Case id:	EU-002
Use Case Name:	Creates a game
Overview:	A user creates a new game and becomes a player of the game.
Primary Actors:	The user creating a new game.
Secondary Actors:	None.
Properties:	Performance:

	Security: Other:
Preconditions:	User must already have an email address that has not yet been used. User must have a nickname that has not yet been used.
Main Flow:	A user initiates the “create game” function. The game is created and the user that initiated the create game is added as a player of this game.
Postconditions:	The game is created with the user that created the game as a player.
Alternative Flows:	None.

Use Case id:	EU-003
Use Case Name:	Handles Invitations
Overview:	A registered user can invite another registered user (or set of users) to join a created game or set of created games.
Primary Actors:	User [Receiver] [Primary][Registered]
Secondary Actors:	User [Initiator][Secondary][Registered]
Properties:	Performance: The receiver should get an invitation. And once accepted the initiator should get a notification. And both players should appear in the game. Security: Only the Initiator can invite other users to the instance of the game they created. Only 2 players can enter the game instance and play at the same time.
Preconditions:	-Both users have to be registered. -The Initiator can invite multiple users at the same time. -A game instance should be created for each invitation that is sent.
Main Flow:	The secondary player sends a invite and the primary player accepts it.
Postconditions:	Notification is sent and once accepted the primary player is apart of the game instance that the secondary player created.
Alternative Flows:	The player denies the request or simply does not respond. The game instance is deleted or pending.

Use Case id:	EU-004
Use Case Name:	Registered User Accepts or Rejects Invitation
Overview:	A registered user can accept or reject an invitation to join a game. If the user accepts the invitation, she becomes a player of the game
Primary Actors:	User who receives [Primary]
Secondary Actors:	User who invites [Initiator]
Properties:	**Performance:** User is invited without large delay. **Security:** Only the intended user is invited to the game. Other users do not see the invitation. **Other:**
Preconditions:	Both users are registered with the system
Main Flow:	User accepts the invitation to join game. The system associates that user with the specific game. The user is now playing the game from which the invitation specified.
Postconditions:	User is placed onto game with initiating user
Alternative Flows:	User rejects the game invitation. The system does not associate user with the game the invitation specified.

Use Case id:	EU-005
Use Case Name:	Joins different game
Overview:	A user can be a part of multiple games simultaneously without the games affecting each other
Primary Actors:	Registered user [Initiator][Primary]
Secondary Actors:	
Properties:	Performance: No impact

	Security: Only the primary actor and opponents can view the games in which the primary actor is playing Other:
Preconditions:	User must be registered
Main Flow:	A user has the ability to play multiple games at one time.
Postconditions:	User plays multiple games without issue
Alternative Flows:	A user does not join multiple games

Use Case id:	EU-006
Use Case Name:	Accesses Game
Overview:	A registered user only has access to the games they are a player of.
Primary Actors:	A registered user [Initiator][Primary]
Secondary Actors:	None.
Properties:	Performance: No other game systems should appear in the registered user's account that they have not created themselves or have entered due to invitation. Security: Cannot access or change data from other games that they are not a part of. Other: Players can look at other game data via player history.
Preconditions:	User must already have an email address that has not yet been used. User must have a nickname that has not yet been used. User must be registered with a proper account.
Main Flow:	The user can access a game by creating one and being the admin for that game by hosting it.
Postconditions:	The player can only see what games they have been apart via the main flow or the alternative flow. All games will be displayed under the user's account whether it is in process or finished.
Alternative Flows:	A user can also join a game via invitation as well from another user.

Use Case id:	EU-007
Use Case Name:	Quits
Overview:	A player can quit a game at any time
Primary Actors:	Registered user [Initiator][Primary]
Secondary Actors:	
Properties:	**Performance:** No impact **Security:** Only the user can quit games he/she is a part of. Other users should not be able to quit a game for another user **Other:**
Preconditions:	User must be a part of one or more games
Main Flow:	User quits a game
Postconditions:	User is no longer a play on the quit game. The other player is given a win and the initiator is given a loss
Alternative Flows:	User chooses not to quit game and remains a player

Use Case id:	EU-008
Use Case Name:	Unregisters
Overview:	A registered user can unregister from the system.
Primary Actors:	Registered user [Initiator] [Primary]
Secondary Actors:	
Properties:	Performance: No impact Security: Only the primary actor can unregister him/herself from the system Other:
Preconditions:	User must be registered

Main Flow:	User unregisters
Postconditions:	The user no longer has an account. The email address and nickname are now free to use for registration
Alternative Flows:	User does not unregister and the account remains active

Use Case id:	EU-009
Use Case Name:	Records History
Overview:	The system must record a history of games played by a user. The record must include opponent, start date and time, end date and time, and result of game.
Primary Actors:	Registered user [Initiator] [Primary]
Secondary Actors:	
Properties:	Performance: No impact Security: Only registered users can view the history of other players Other:
Preconditions:	User is registered
Main Flow:	Each time a game is played, the system records the opponent's name, start time and date, end time and date, and the result of the game.
Postconditions:	All registered users can view all other users' game histories with the information above included
Alternative Flows:	User has not played any games and therefore has a blank history

Use Case id:	EU-010
Use Case Name:	Views History
Overview:	A registered user has a profile which consists of his/her nickname and history of played games. User profiles are only visible to other registered users.

Primary Actors:	Registered user [Initiator] [Primary]
Secondary Actors:	Friend of the Registered User
Properties:	Performance: A registered user that is not in a game making a move should be able to look at the account of another registered player. Once a game instance is finished both players accounts must be updated accordingly. Security: There is no security. Any player can look at any other player in the game system. Other: Need a look-up system that allows players to search other players via their nickname.
Preconditions:	- Both players must be registered in the game system. - A player should have at least played and finished one game in the past to have a history. - That game data is stored under the player's account.
Main Flow:	Player uses the look-up dictionary to look up a player by their nickname, selects their account and all of the player's history including (wins and loses, scores, rank, etc)
Postconditions:	The player sees another player's information and can be directed to look up another account or can exit the display to start a game instance.
Alternative Flows:	If the player searched on does not have history yet and hasn't played on game yet a default profile must be set.

Use Case id:	EU-011
Use Case Name:	Minimum Number of Players Have Joined Game
Overview:	A game cannot start until the minimum number of players required for the game have joined.
Primary Actors:	User [Joining] [Joined][Registered]
Secondary Actors:	
Properties:	**Performance:** The game eventually closes if the minimum numbers of players is not met so resources are not wasted.

	Security: Only registered users can join the game. **Other:**
Preconditions:	There are enough users online that are free who can join a game while meeting the minimum player requirements.
Main Flow:	The game starts when the minimum number of player have joined the game.
Postconditions:	The game has started.
Alternative Flows:	The minimum number of players has not been met , the game cannot start.

Use Case id:	EU-012
Use Case Name:	Game Exclusivity
Overview:	Once a game starts, a new players cannot join.
Primary Actors:	2 players inside the game. [Registered][Joined]
Secondary Actors:	1..* players outside the game instance. [Registered][Unjoined]
Properties:	Performance: If the game has exactly two players that can play at any giving time. Security: Once the game has started and an invitation has be accepted both players inside the initiated the game cannot send out more invitation or create games until they save and exit the game. Only one invitation can be sent per game instance so there should be no crossover. Other: Message must be sent to the outside player that the invitation is no longer valid and the game has been started.
Preconditions:	- Game instance created. - Multiple invitations sent and accepted at different times. - Invitations accepted at different times.
Main Flow:	A invitation is sent to a player who accepts and gets to enter the game. Optional -(Delete all invitations sent by that player or have multiple game instance with that opponent)

Postconditions:	The game is in a secure state in which only the two players are engaged in.
Alternative Flows:	If a player leaves an unfinished game the game is over and no other player can join to finish it.

Use Case id:	EU-013
Use Case Name:	Initiates
Overview:	The system must determine which player starts the game.
Primary Actors:	Game [Registered]
Secondary Actors:	None
Properties:	**Performance:** The player that creates the game will be set to admin privileges for that game and the system will distinguish them as the host. **Security:** Other players that participate in this game cannot change the permissions of the game or identify as the host. The person who is the creator of the game instance will forever be the leader of that particular game even after it's finished. **Other:** None
Preconditions:	-User must be registered -The user must create a game instance through their account. -The player must not be inside another game to initiate another game.
Main Flow:	Once the game is created the game system must recognize the user and promote them as the leader or host of the game.
Postconditions:	-The user can't start the game instance until another player joins the game but should have the ability to invite others since they are the leader of that game.
Alternative Flows:	Since this process will be automatic once the game instance is produce there should not be any alternative flows.

Use Case id:	EU-014

Use Case Name:	Takes Turn
Overview:	The system determines whose turn it is.
Primary Actors:	2x User [Registered][Joined]
Secondary Actors:	None
Properties:	**Performance:**Turns should alternate until the game is finished. Each Player is only allowed one move per turn. Offense and Defense should be decided at random. The player will be notified which position they play. Offense always goes first. **Security:** Players should be able to see each others move once the turn is switched. A turn can not be overturned or redone after its been committed.
Preconditions:	-Two registered players are engaged in a game instance, so a invitation has been accepted.
Main Flow:	A notification will appear once one of the registered player has accepted the invitation that it's the given player's turn to play. The notification can be set on a timer to disappear or stay until the turn is over.
Postconditions:	This notification will rotate from player to player until the win conditions of the game are met.
Alternative Flows:	If a player quits the turn workflow will cease directly following and the player still active in the game will be notified of the absent and win conditions will be set.

Use Case id:	EU-015
Use Case Name:	Playing in active games
Overview:	A player can only make moves in their active games
Primary Actors:	A player
Secondary Actors:	
Properties:	**Performance:** None. **Security:** Players can not make moves in games they are not apart of.

	Other: None.
Preconditions:	Active games are only impacted by the two players in it.
Main Flow:	A player makes a move in a game they are in.
Postconditions:	The games a player moves in is only a part of their game list.
Alternative Flows:	

Use Case id:	EU-016
Use Case Name:	Moves by the correct player.
Overview:	A player can only make moves if it is their turn to play.
Primary Actors:	A player
Secondary Actors:	
Properties:	**Performance:** Enforcing the game rules to ensure the game flow. **Security:** Players not making moves out of turn to keep the integrity of the game. **Other:**
Preconditions:	A game exists between two players. Only one player has the ability to make a move.
Main Flow:	The player whose turn it is can move. They make a move. Now the other player can move.
Postconditions:	The move is made by the player who has the turn, now the turn switches and the other player can only make a move.
Alternative Flows:	

Use Case id:	EU-017
Use Case Name:	Makes move

Overview:	User
Primary Actors:	A player making a move.
Secondary Actors:	
Properties:	**Performance:** **Security:** **Other:**
Preconditions:	A player has a game and it is his turn.
Main Flow:	A player tries to make a move. The only moves he can make are game-allowed. The player makes an allowed move.
Postconditions:	The move is made and it adheres to the game rules.
Alternative Flows:	

Use Case id:	EU-018
Use Case Name:	Saves game
Overview:	The system saves the state of active games.
Primary Actors:	User [Registered][Joined]
Secondary Actors:	None
Properties:	**Performance:** If a game instance is in progress, a game can be saved by either player. **Security:** A player who's not on their turn can not save and quit the game while another player is making their move. Must have versioning control within the game system. **Other:** Have a save and quit option (optional), a save game also can update the player history and show the status of the game to other players.
Preconditions:	-A game instance with registered users must be in process. -The player that is trying to save the game must be on their turn before the commit to making a move.

Main Flow:	If preconditions are met, the player can save the game which means a notification would be triggered and the opposite player is notified and confirmed that the game is save.
Postconditions:	Once the game is saved the game state is stored under both accounts so if either player quits with the game they can pick up where they left off.
Alternative Flows:	If the game is saved but the player does not want to finish it and deletes the game instance under their account.

Use Case id:	EU-019
Use Case Name:	Ends Game
Overview:	The system must determine when a game is over. The system must also determine who is the winner and the loser of each game, when there is a tie or when there is a draw according to the game rules.
Primary Actors:	User [Registered][Joined]
Secondary Actors:	None
Properties:	**Performance:** Conditions for the game to finished will be based on the Hnefatafl rule set. Once those conditions are met player history will be updated automatically and the game instance will be deleted and no longer saved. **Security:** Players should not be able to change the results of the game once the winner/loser has been set.
Preconditions:	- All of Hnefatafl win conditions are met.
Main Flow:	Once the preconditions are met, the both players will be sent a notification telling the the game is over, it has been deleted, and the conclusion/score of the game. After the players tell the system to continue, both players a ejected from the game instance.
Postconditions:	- The game instance must be deleted. - Player history must be updated.
Alternative Flows:	No alternative flows, the game is either in progress or is finished once the win conditions are meet.

Use Case id:	EU-020
Use Case Name:	AI gameplay
Overview:	A registered user can start a new game against a bot player. The bot player is an AI who decides the best move to play
Primary Actors:	User[Registered][Joined]
Secondary Actors:	AI Bot[Joined]
Properties:	**Performance:** The AI algorithm should decide moves quickly and make correct decisions. **Security:** **Other:**
Preconditions:	The User is registered
Main Flow:	Registered user gets online. Users choses to play against AI bot. A game is created with a user who joins and an AI bot.
Postconditions:	The game is started with the minimum number of players in the game.
Alternative Flows:	The user does not choose to play a game vs the AI bot

Use Case id:	EU-021
Use Case Name:	Game Tournament
Overview:	A registered user can start a tournament. Registered users can join the tournaments without invitation. A tournament starts with eight games between unique players. The hierarchy of matches is randomly determined by the system before the first round of games starts. The winner of the tournament receives a gold badge. The second place receives a silver badge. The badge becomes part of their user profile. The system must provide a public rank of players according to their badges.
Primary Actors:	User[Registered][Uninvited]
Secondary Actors:	

Properties:	**Performance:** **Security:** **Other:**
Preconditions:	Users are registered before joining tournament. Minimum number of players have joined the game. Hierarchy of matches is randomly determined before first round of the game starts.
Main Flow:	Registered users join the game. The systems appoints matches randomly for each users in the game. The users play against one another until and first place and second place winner has been found. The first place winner is awarded a gold badge that is public on their user profile. The second place winner is awarded a silver badge that is public on their user profile. The user profile then displays the updated public rank based on the badges they have won via tournaments.
Postconditions:	The first and second place winners have new badges on their public profile. The public profile has an updated public rank.
Alternative Flows:	The user does not take first or second place.