Software Requirement Specification

1. Introduction

1.1 Purpose

This software requirement specification document is describing the functional and nonfunctional requirements for the 1.0 release of the software Simple Social Card Collection Battling Game. The project member uses this document to review the correctness of software implementation. Under normal circumstances, everything specified here has the highest priority for implementation.

1.2 Project Features

The Simple Social Card Collection Battling Game allows player to build a deck of his/her own, and battling with other players and make friends with them.

2. Overall Description

2.1 Product Perspective

The Simple Social Card Collection Battling Game is bringing up a new game for social and time killing. Figure 1 shows the structure of the software in 1.0 release.

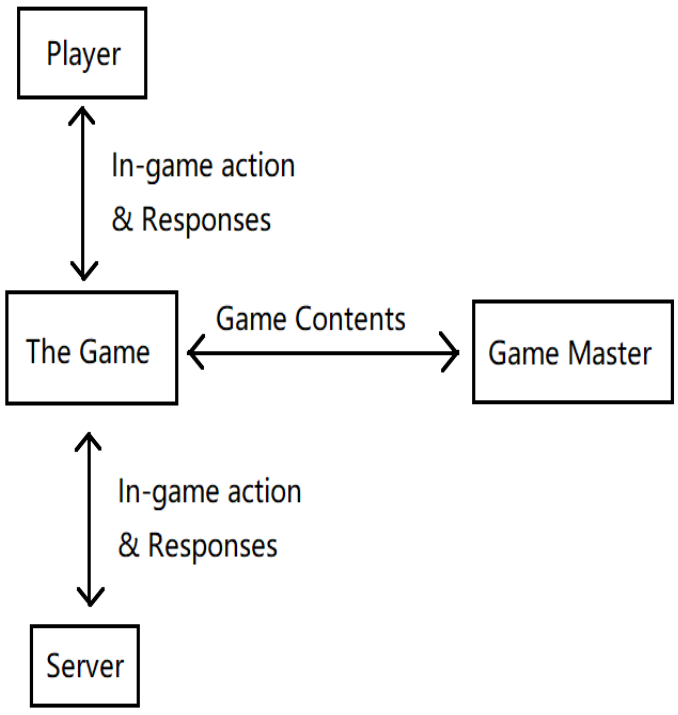

Figure 1.

2.2 User classes and interfaces

Players: Players are the people who will play this game.

2.3 Operating environment

Capable of running in desktop operating systems with Java runtime environments, under default memory setting.

2.4 Design and Implementation Constraints

All code should be written in Java. Tests should use JUnit functional tests.

2.5 User Documentation

A help document explaining basic operations in the game and gaming rules will be provided.

2.6 Assumptions and Dependencies

Assumption: Server should be up 24/7 unless is undergoing maintenance.
Dependencies: Server program is uploaded to eecslinab1 engineering department lab server under the account of qxm28.

3. System Features

3.1 Login

3.1.1 Description and Priority

A player who provided correct login information should be redirected to the mainwindow. Priority = medium.

3.1.2 Stimulus / Response Sequences

Stimulus: Player inputs username and password, and clicks “login”.
Response: System sends the username and password to the server. If server can log the player in, system directs player to mainwindow. Otherwise, clears the fields and ask the user to try again.

3.1.3 Functional Requirements

Player.Login: System opens up a window and corresponding fields for player to input username and password in.
Player.Login.Send: System uploads the login information to server.
Player.Login.Parse: System process the login result sent back from the server. Redirects player to mainwindow if passed, or clears the field and ask to try again if not.

3.2 Mainwindow

3.2.1 Description and Priority

The mainwindow for accessing all aspects of the game software. Priority = medium.

3.2.2 Stimulus / Response Sequences

Stimulus: Player clicks quickmatch
Response: System redirects player into a queue of matchmaking.
Stimulus: Player clicks to view his friends
Response: System displays his friend list.
Stimulus: Player clicks to edit his deck
Response: System redirects player to deck-editing.
Stimulus: Player close the program
Response: System aborts all chats and matches and close the program.

3.2.3 Functional Requirements

State0:
Player.Mainwindow: System opens up a window with several buttons.
Player.Mainwindow.Quickmatch: Calls matchmaking. Enter state1.
Player.Mainwindow.ViewFriends: Retrieve the newest friend list from server. Enter state2.
Player.Mainwindow.DeckEditing: Calls deck editing. Enter state3.
Player.Mainwindow.Quit: Shutdown all chat windows, matches. Close the program.

State1:
Player.Mainwindow.Quickmatch: Load panel elements. If a match is found, enter state0.
Player.Mainwindow.QuickMatch.Cancel: Enter state0

State2:
Player.Mainwindow.ViewFriends: Load panel elements according to friends, and display all battle requests as notifications.
Player.Mainwindow.ViewFriends.Select: Selects a friend and change the background color for better visualization.
Player.Mainwindow.ViewFriends.StartChat: Start a chat with the selected friend.
Player.Mainwindow.ViewFriends.RequestBattle: Request a battle with the selected friend.
Player.Login.Send: System uploads the login information to server.
Player.Mainwindow.ViewFriends.AcceptBattle: Sends confirmation to the server, rejects all pending battle requests, and calls Match.
Player.Mainwindow.ViewFriends.RejectBattle: Rejects the battle.
Player.Mainwindow.ViewFriends.Back: Enter state0.

State3:
Player.Mainwindow.DeckEditing: Load panel elements.

Player.Mainwindow.DeckEditing: Enter state0.

3.3 Matchmaking

3.3.1 Description and Priority

Player can enter a quickmatch with other players who has also requested a quickmatch. Priority = low.

3.3.2 Stimulus / Response Sequences

Stimulus: A match is found, both players accept
Response: System calls Match, and notifies mainwindow a match is found.
Stimulus: A match is found, the player rejects
Response: System redirects player back to mainwindow.
Stimulus: A match is found, opponent rejects
Response: System keeps matchmaking.
Stimulus: No match is found, player cancels matchmaking
Response: System redirects player back to mainwindow.

3.3.3 Functional Requirements

Player.Matchmaking: Periodically sending request to server.
Player.Matchmaking.Accept: Send acceptance to server.
Player.Matchmaking.Accept.OpponentAccepted: Match starts after both players accepted.
Player.Matchmaking.Accept.OpponentReject: Matchmaking will continue.
Player.Matchmaking.Reject: Sends rejection to server, stop matchmaking.
Player.Matchmaking.Cancel: Stops matchmaking.

3.4 Gameplay

3.4.1 Description and Priority

Player who accepted a match enters actual gameplay. Priority = high.

3.4.2 Stimulus / Response Sequences

Stimulus: Player selects a card in his hand for summoning.
Response: System asks for a location on board to summon to.
Stimulus: Player selects an empty space on board for the selected card summon.
Response: System summons the card to board and removes the card from hand. If the SP check passed
Stimulus: Player selects a card on board and activates its skill
Response: System read in the card, and if the skill that card possesses requires an additional target, system will ask the user to select a corresponding location (an enemy, an ally, an empty space etc.). Otherwise, the skill will activate immediately.
Stimulus: Player selects a corresponding location for skill activation.

Response: System process the skill activation.
Stimulus: Player cancels summoning or activating skills that requires additional target besides the card itself.
Response: System aborts summon and skill activation.
Stimulus: Player surrenders.
Response: System quits, and notifies the other player that this player surrendered.
Stimulus: Player ends his turn.
Response: System process the players after-turn checks, and determine if the victory condition is achieved. If victory, match ends and the other player is notified. If not victory, process opponent's pre-turn checks and notify opponent that it is his turn.
Stimulus: Player input anything except surrender while not in his turn.
Response: System rejects them all.

3.4.3 Functional Requirements

State0: Normal state in turn.
Player.Match.Summon: Remembers the card in hand, and enters state1.
Player.Match.Skill: Check if skill requires target -> b1. If b1, remembers card, and enter state2. Else, process skill, and enter state0.
Player.Match.EndTurn: calls after-turn check, victory check, (opponent's pre-turn check) consequently. Enter state -1.
Player.Match.Surrender: closes the match.

State1: Next input selects an ally empty space for summoning
Player.Match.Summon: Summons the remembered card to field, and enter state0.
Player.Match.Cancel: Forget the card, and enter state0.

State2: Next input selects a location on board for skill activation
Player.Match.Skill: If skill requires enemy, enter state3. If skill requires ally, enter state4. If skill summons a monster, enter state1.

State3: Next input selects an enemy
Player.Match.Skill: Process skill. Enter state0.
Player.Match.Cancel: Forget the card, and enter state0.

State4: Next input selects an ally
Player.Match.Skill: Process skill. Enter state0.
Player.Match.Cancel: Forget the card, and enter state0.

State -1: Waiting for opponent to finish his turn
Player.Match.ProcessActions: System retrieve actions from the server, and reflect them. If action is EndTurn, enter state0.
Player.Match.Surrender: closes the match.

General

Player.Match.SendAction: System sends players action to the server for his opponent to retrieve.
Player.Match.RetrieveAction: System retrieves opponents actions and reflect them.
Player.Match.PreTurnCheck: Inputs a player, and process all cards' pre-turn effects on this player's field according to their orders being summoned.
Player.Match.AftTurnCheck: Inputs a player, and process all cards' after-turn effects on this player's field according to their orders being summoned.
Player.Match.VictoryCheck: Inputs a player, if the player's VP exceeds or equals to 100, declare victory and close match.
Player.Match.Close: close match.

3.5 Deck Editing

3.5.1 Description and Priority

Player can build a deck of his own and bringing it into matches. Priority = high.

3.5.2 Stimulus / Response Sequences

Stimulus: Player adds a card to the deck.
Response: System adds the card if the card number does not exceeds limit.
Stimulus: Player removes a card from the deck.
Response: System removes the card if there is such card in the deck.
Stimulus: Player leaves deck editing.
Response: System saves the deck for further matches, and notifies player when the deck is not complete.

3.5.3 Functional Requirements

Player.Deckbuilding.Add: Adds the card to the deck. System displays updated results.
Player.Deckbuilding.Remove: Removes the card from the deck. System displays updated results.
Player.Deckbuilding.Saved: The original deck
Player.Deckbuilding.Abort: Update the deck using Player.Deckbuilding.Saved.
Player.Deckbuilding.Save: Update the Player.Deckbuilding.Saved using the current deck.

4. External Interface Requirements

4.1 User Interfaces

The game client.

4.2 Hardware Interfaces

Standard keyboard and mouse.

4.3 Software Interfaces

SI 1: Game Server
SI 1.1: Server should send client information about the latest changes and news made to the game.
SI 1.2: Server should update maintenance schedule on a regular basis.

4.4 Communication Interfaces

Internet connectivity.

5. Other non-functional Requirements

5.1 Performance Requirements

PE-1: The server program should be efficient enough to handle at least 10 matches at the same time.

5.2 Safety Requirements

None.

5.3 Security Requirements

None.

5.4 Software Quality Requirements

Neat and nice-looking.