Software Requirements Specification

Zombie Apocalypse

Chad Nelson
Cole Anagnost
Tasewell Fox
Tim Flanigan

TABLE OF CONTENTS

REVISION HISTORY

Version	Date	Author	Change
0.1	9/19/2009	Group 10	Initial Document
0.2	9/20/2009	Tasewell Fox	Added Cases

1. INTRODUCTION

1.1 PURPOSE

The purpose of this Software Requirements Specification document is to describe the architecture and design of the Zombie Apocalypse game. The intended audience includes the project's developers and the Professor, TAs, and students of the ComS 309 class for Fall 2009.

1.2 SCOPE

The scope of this document is the low-level architecture and intended design of the Zombie Apocalypse game. This document should give a detailed outline of the goals and expected functionality of the Zombie Apocalypse game.

The scope of this initial document does not include technical specifications intended for development.

1.3 DEFINITIONS, ACRONYMS, ABBREVIATIONS

Term	Description
Alpha	Transparency
AS3	(ActionScript 3)The programming language used with Adobe Flash
Chat Room	A collection of client connections to the server that are grouped together for the purpose of sharing text messages; the lobby and all game rooms are chat rooms
Client	(Actor) An instance of the game run in a web browser; there are many clients
DisplayObject	An interface that any visible item implements
Display Tree	A tree containing a root node and it's children; any visible object must be attached to the display tree before it is visible on the stage
Game Room	A chat room where all players have the intent of playing a multiplayer game together; limited number of players; there can be any number of game rooms
Lobby	The default chat room where players are allowed to create/join game rooms; unlimited number of players
MovieClip	An AS3 object that implements the DisplayObject interface; contains a number of frames that are animated
PC	(Player Character) The player's sprite
Player	(Actor) A person that plays the game by running an instance of the client
Server	(Actor) A program that manages connections and stores information. Manages room creation and multiplayer communication, and stores data about player accounts, high scores, and game status.
Sprite	An AS3 object that implements the DisplayObject interface; contains a single static image
Stage	A static AS3 object that represents the view of all visible objects

*.SWC	(Shockwave Flash Component extension) Contains code and graphics that can be instantiated by another Flash program
*.SWF	(Shockwave Flash extension) A Flash movie file; Contains compiled byte code and graphics that will be interpreted by a Flash Player and displayed on a webpage

1.4 DESIGN GOALS

1. Usability - The game must provide an intuitive player interface and control scheme, in addition to help or tips for an inexperienced player. The game should also remain responsive during intensive processing for many onscreen opponents. This design goal is met by implementing a widely used control scheme (WASD keyboard controls for player movement with QER for other actions, mouse for aiming and firing) with all necessary information shown in a GUI overlay.
2. Multiplayer - The game must allow multiple players to play on the same map in real time. This design goal will be met using a client-server architecture, where central servers store the player's data and relay information between client machines.

1.5 REFERENCES

[None]

1.6 OVERVIEW

[Omit]

2. OVERALL DESCRIPTION

Zombie Apocalypse is a browser-based, multiplayer, shooter game. It has many single player levels, and gives the players the ability to shoot zombies cooperatively with multiplayer. This feature includes being able to chat with other players. Players earn cash by shooting zombies, and can purchase weapons and other items with the money.

2.1 PRODUCT PERSPECTIVE

Zombie are a fairly common subject in mainstream video game culture. Zombie Apocalypse will differentiate itself by offering flash-based multi-player and persistent stat tracking. This will allow casual gamers and hardcore gamers alike to quickly pick-up our game and play it.

2.1.1 Concept of Operations

Zombie Apocalypse is developed in two different environments. The client is web based, written in ActionScript 3, and displayed in the browser using a Flash Player. The server is written in Python.

When a player loads an instance of the client in his web browser, it automatically creates a socket connection to the server. After logging in, the client fetches data about the player from the server and then displays the main menu interface. The user can choose either single player of multiplayer. In single player mode, the player begins a level and shoot zombies until the level is complete (or he dies). After a level is beaten, statistics are transferred to the server for storage. In multiplayer mode, the player is able to take advantage of the socket connection by chatting with other players connected to the server, and using to server the enter the same level with multiple players at the same time.

2.2.2 Major User Interfaces

Please see ***Appendix A*** for screenflow and interface mock-ups.

2.1.2.1 Example Screenshots and Descriptions

Please see ***Appendix A-2*** for screen shots and descriptions.

2.1.3 Hardware Interfaces

The player will interface with the client using the keyboard and mouse. Specifically:

Keyboard Controls:

- W - up
- A - left
- S - down
- D - right
- R - reload
- E - switch weapon
- Q - use powerup (if applicable)

- H - display help menu
- P - pause (single player only)

Mouse Controls:
- Click - Shoot

2.1.4 Software Interfaces

[Omit]

2.1.5 Communication Interfaces

[Omit]

2.1.6 Memory Constraints

[Omit]

2.1.7 Operations

[Omit]

2.1.8 Site Adaptation Requirements

[Omit]

2.2 PRODUCT FUNCTIONS

[TODO] A pithy introduction.

2.2.1.a Use Case 1 (Text)

Name: The player wants to log on to the game.
Author: Chad Nelson
Description: The player interacts with the client to provide credentials, the client then contacts the server to validate these credentials.
Actors: Player, Client, Server
Primary Flow:
Happy Path
1. The player enters his playername into the client
2. The player enters his password into the client
3. The player indicates he wants to log on with these credentials
4. The client confirms the credentials with the server; they are valid
5. The client downloads player data from the server and starts the game

Alternative Flows:
Invalid playername
1. The player enters an incorrect playername into the client
2. The player enters his password into the client
3. The player indicates he wants to log on with these credentials
4. The client confirms the credentials with the server; they are invalid
5. The client displays an error message and the player is allowed to try again

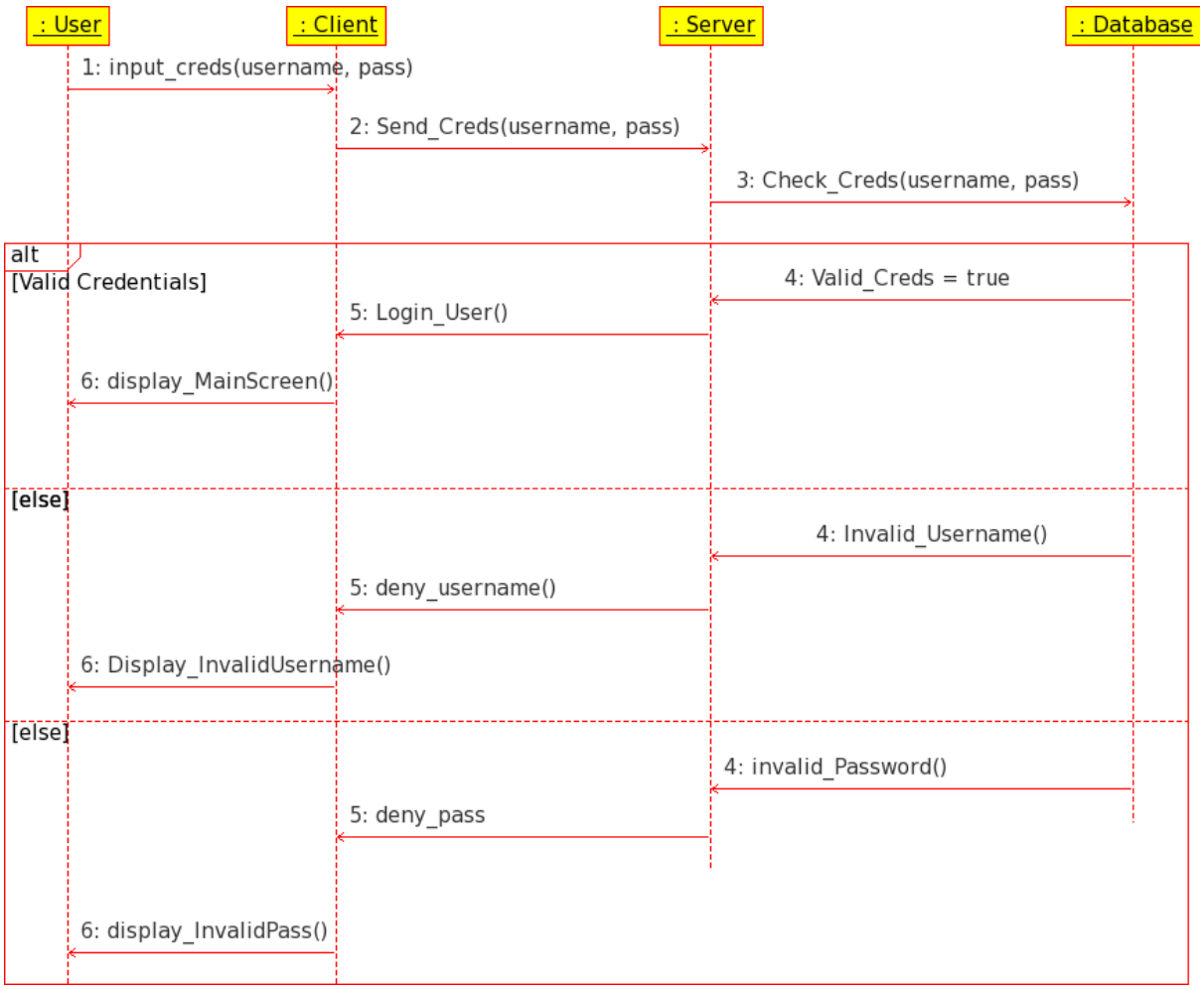
Invalid Password
1. The player enters his playername into the client
2. The player enters an incorrect password into the client

3. The player indicates he wants to log on with these credentials
4. The client confirms the credentials with the server; they are invalid
5. The client displays an error message and the player is allowed to try again

Preconditions: A player has instantiated an instance of the client.
Postconditions: The player is now logged in, and is presented with the main menu.

2.2.1.b Use Case 1 (Diagram)

UML Sequence Diagram for Use Case 2.2.1



2.2.2.a Use Case 2 (Text)

Name: The player wants to play the game with a friend.
Author: Chad Nelson
Description: The player wants to initialize a multiplayer game with another player or friend. The goal of this use case is to allow the player to have a straight forward process for communication and game creation.
Actors: Players, Client, Server
Primary Flow:
Happy Path:

1. The player selects "Multi-player Game" from the main menu.
2. The player is then taken to the game lobby and chat system.
3. The game lobby and chat system presents the player with a list of available games and other players currently logged into the chat system.
4. The player finds the person he/she wishes to play with and initiates communication using the chat system.
5. The player creates a multiplayer game.
6. The other players join the game

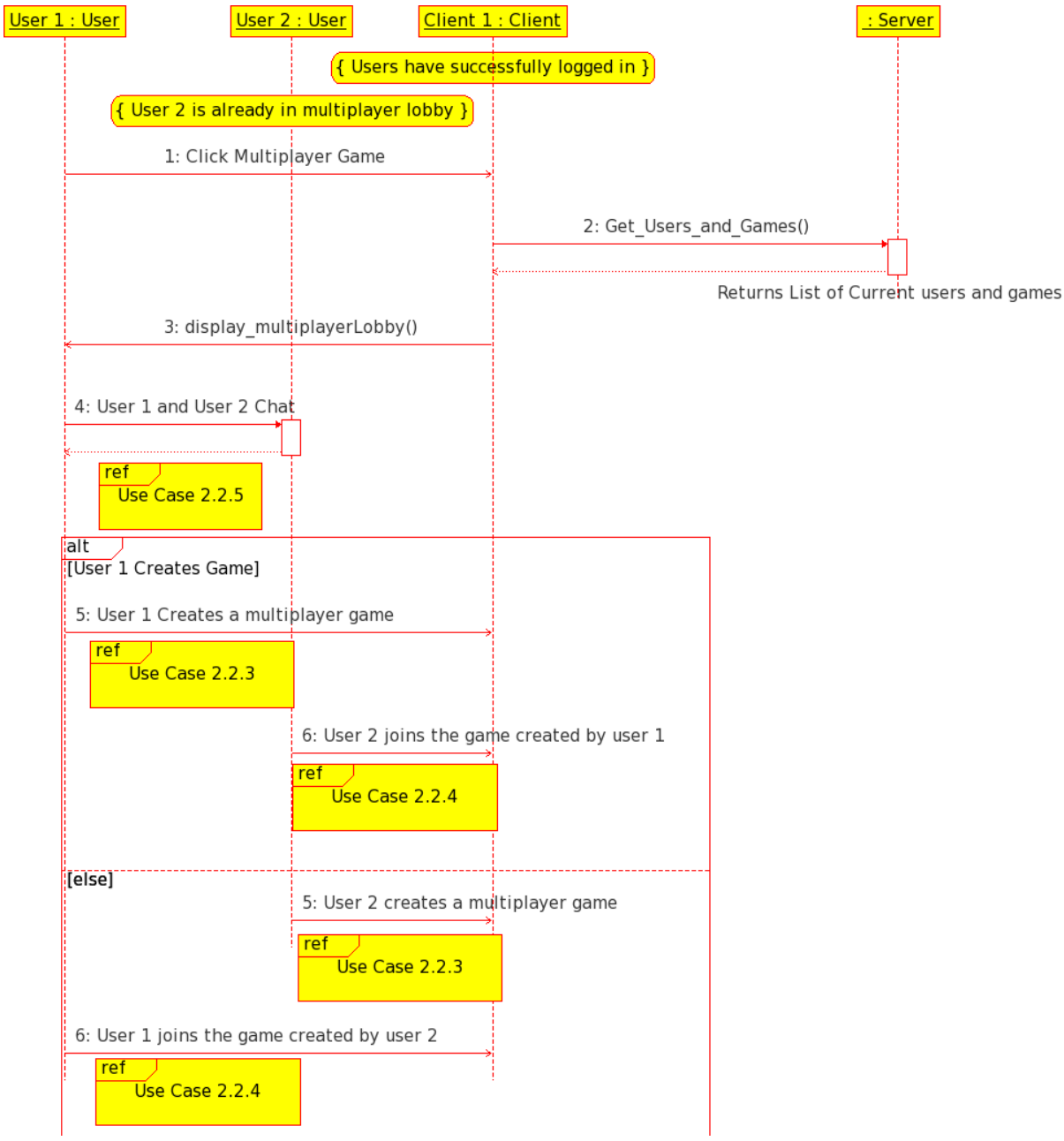
Alternative Flows:
Join Game:

1. Steps 1-4 from *Happy Path*
2. The player joins another players multiplayer game.

Preconditions: The player has successfully logged into the game system
Postconditions: The player has communicated to another player and is now in a game.

2.2.2.b Use Case 2 (Diagram)

UML Sequence Diagram for Use Case 2.2.2



2.2.3.a Use Case 3 (Text)

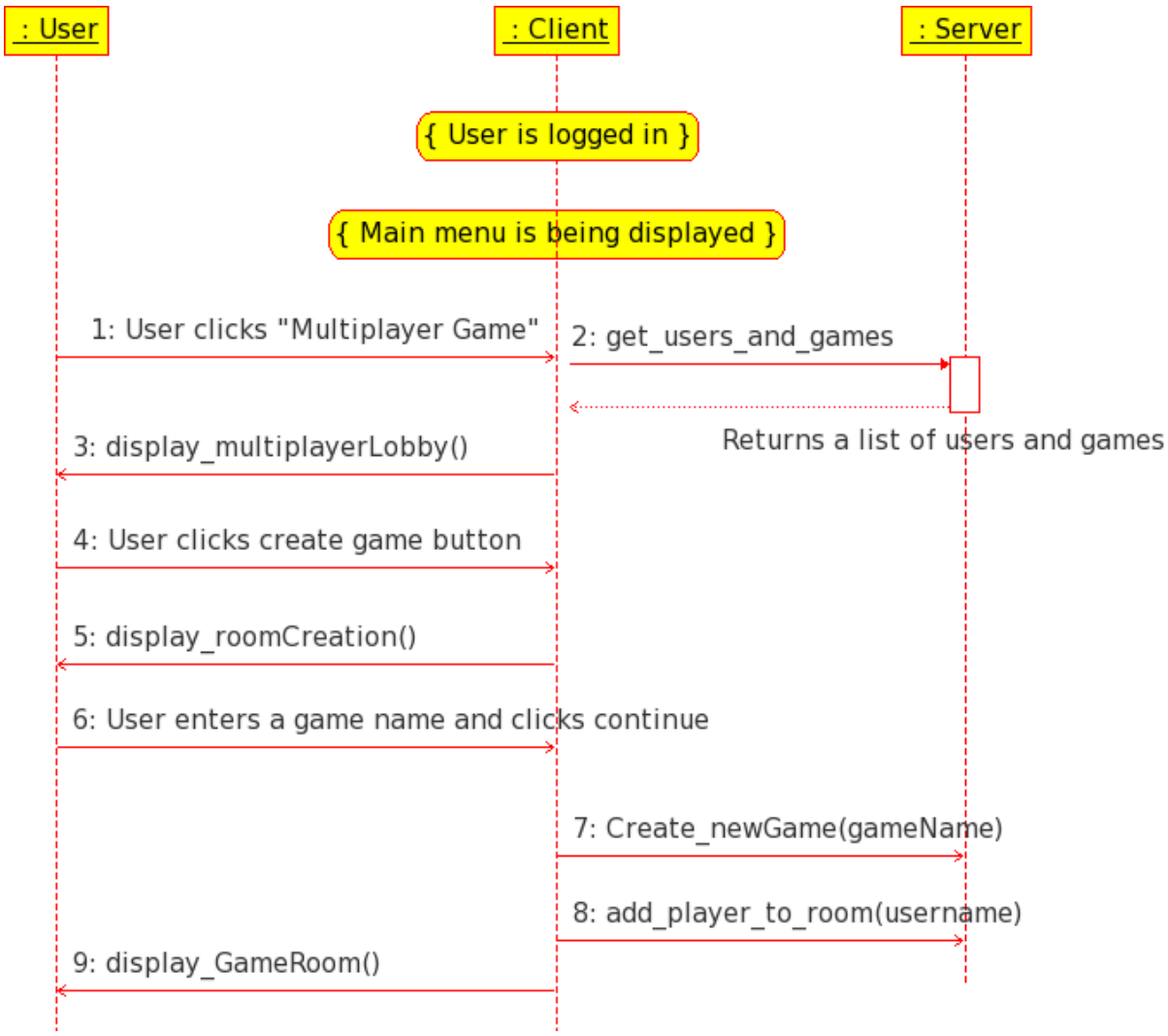
Name: The player wants to create a multiplayer game.
Author: Tim Flanigan
Description: The player interacts with the client to start a game, and the client communicates with the server to display the game in other clients.
Actors: Player, Client, Server
Primary Flow:
Happy Path:

1. The player selects "Multi-player Game" from the main menu
2. The player selects "Create Game" from the multi-player lobby menu
3. The player enters a name for the game room
4. The client contacts the server and adds the new game room to the list of open rooms
5. The client takes the player to the game room as the host, where they can change the game settings

Alternative Flows:
Preconditions: The player has logged into the client. Main menu is currently being displayed.
Postconditions: The player is in the game room as the host, waiting for other players to join.

2.2.3.b Use Case 3 (Diagram)

UML Sequence Diagram for Use Case 2.2.3



2.2.4.a Use Case 4 (Text)

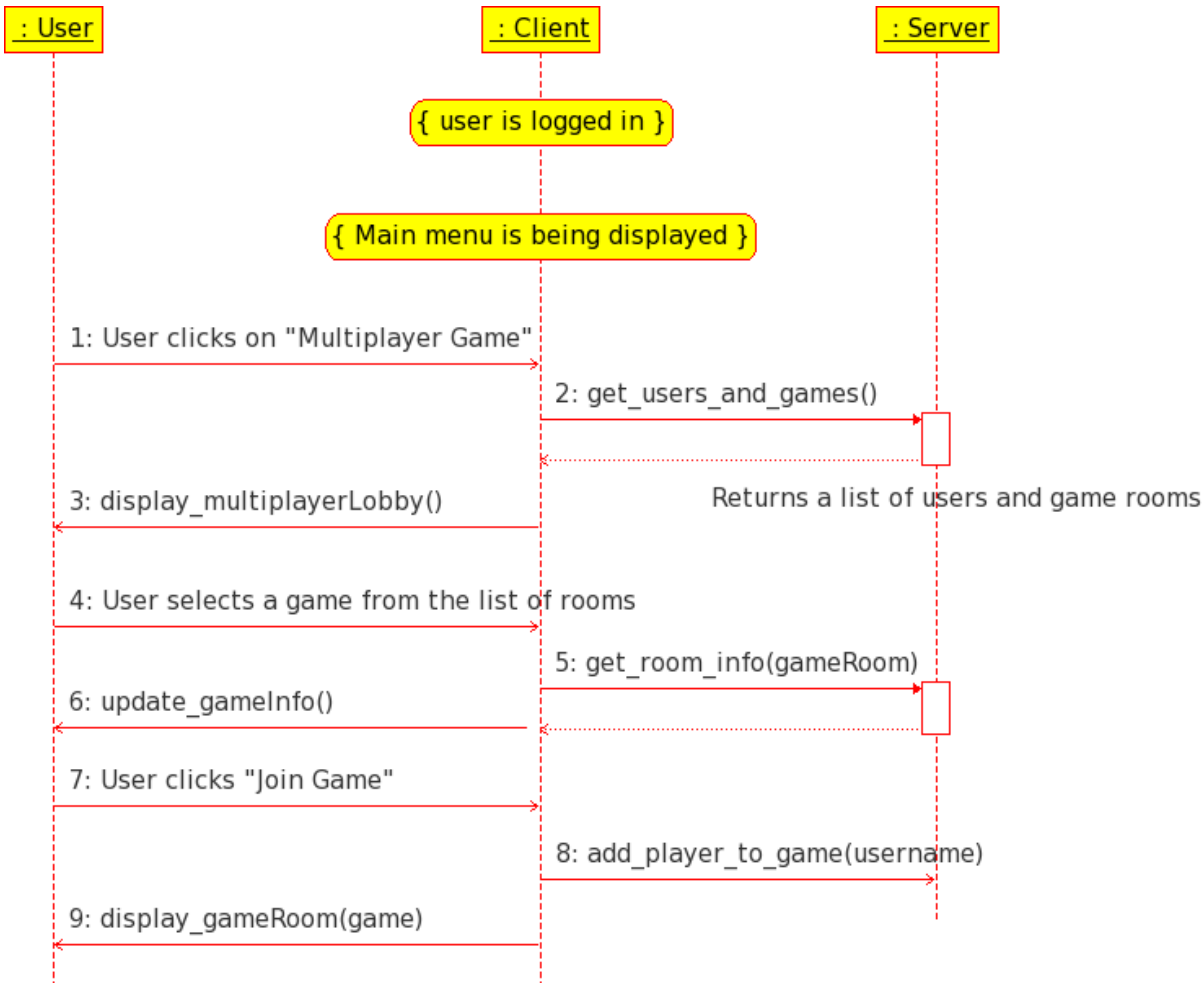
Name: The player wants to join a multiplayer game.
Author: Tim Flanigan
Description: The player interacts with the client to select a game room to join, and the client retrieves game information from the server.
Actors: Player, Client, Server
Primary Flow:
Happy Path

1. The player selects "Multi-player Game" from the main menu
2. The client gets a list of currently open game rooms from the server and displays them to the player
3. The player selects a game from the list
4. The client gets the game information from the server and takes the player to the game room

Alternative Flows:
Preconditions: The player has logged into the client. Main menu is being displayed.
Postconditions: The player is in the game room.

2.2.4.b Use Case 4 (Diagram)

UML Sequence Diagram for Use Case 2.2.4



2.2.5.a Use Case 5 (Text)

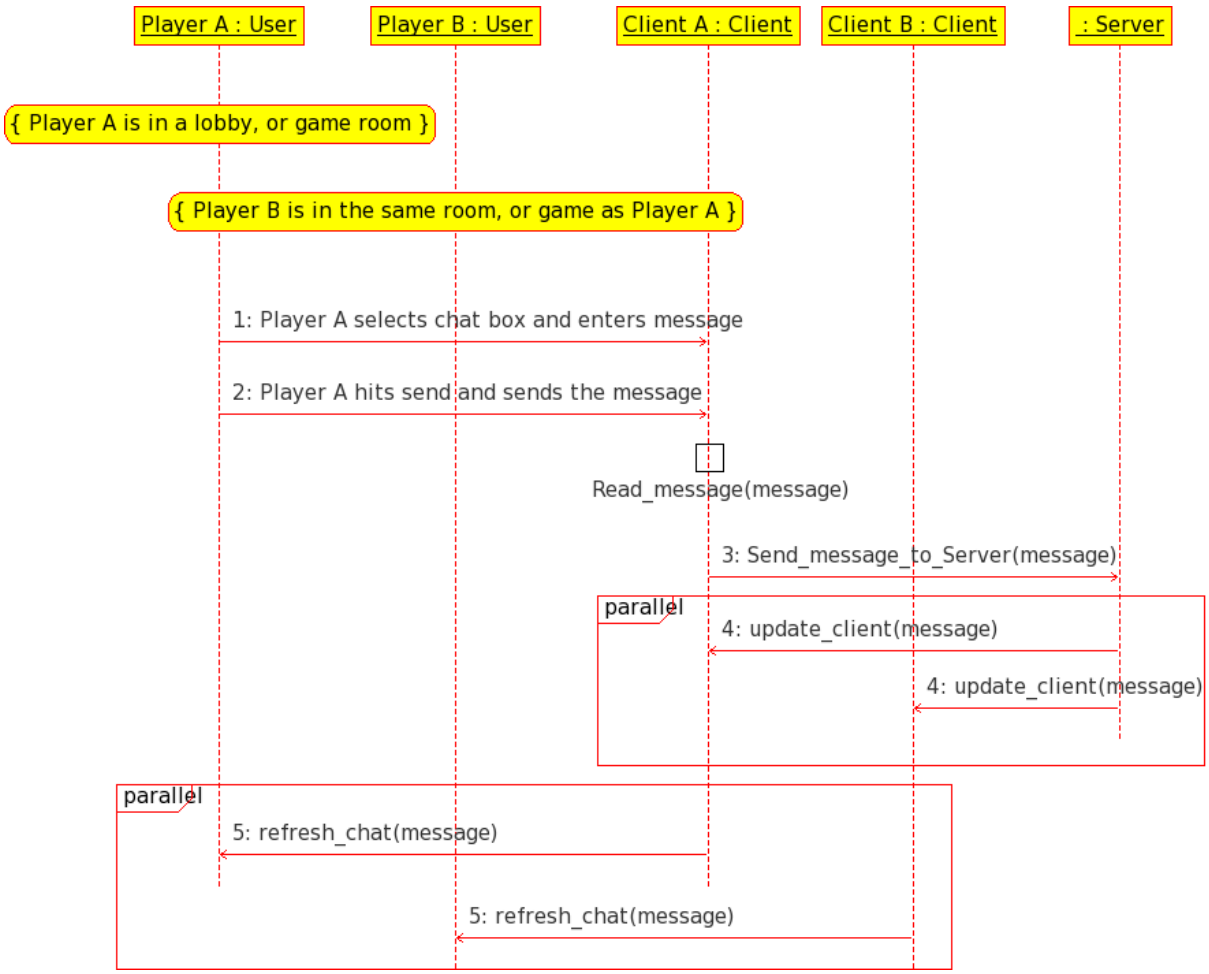
Name: The player wants to chat with other players.
Author: Tasewell Fox
Description: The player wants to communicate with other players while in the lobby, a game room, or while playing a multiplayer game.
Actors: PlayerA, PlayerB, Client, Server
Primary Flow:
Happy Path

1. PlayerA selects the chat box on the screen of the client
2. PlayerA types a message
3. PlayerA indicates he wants to submit his message
4. The client reads the message and sends it to the server
5. The server forwards the message to the other clients in PlayerA's chat room

Alternative Flows:
Preconditions: PlayerA is already in either the lobby, a game room, or a multiplayer game. PlayerB is in the same chat area (lobby or game room) as PlayerA.
Postconditions: PlayerB receives PlayerA's chat and it is displayed on his screen.

2.2.5.b Use Case 5 (Diagram)

UML Sequence Diagram for Use Case 2.2.5



2.2.6.a Use Case 6 (Text)

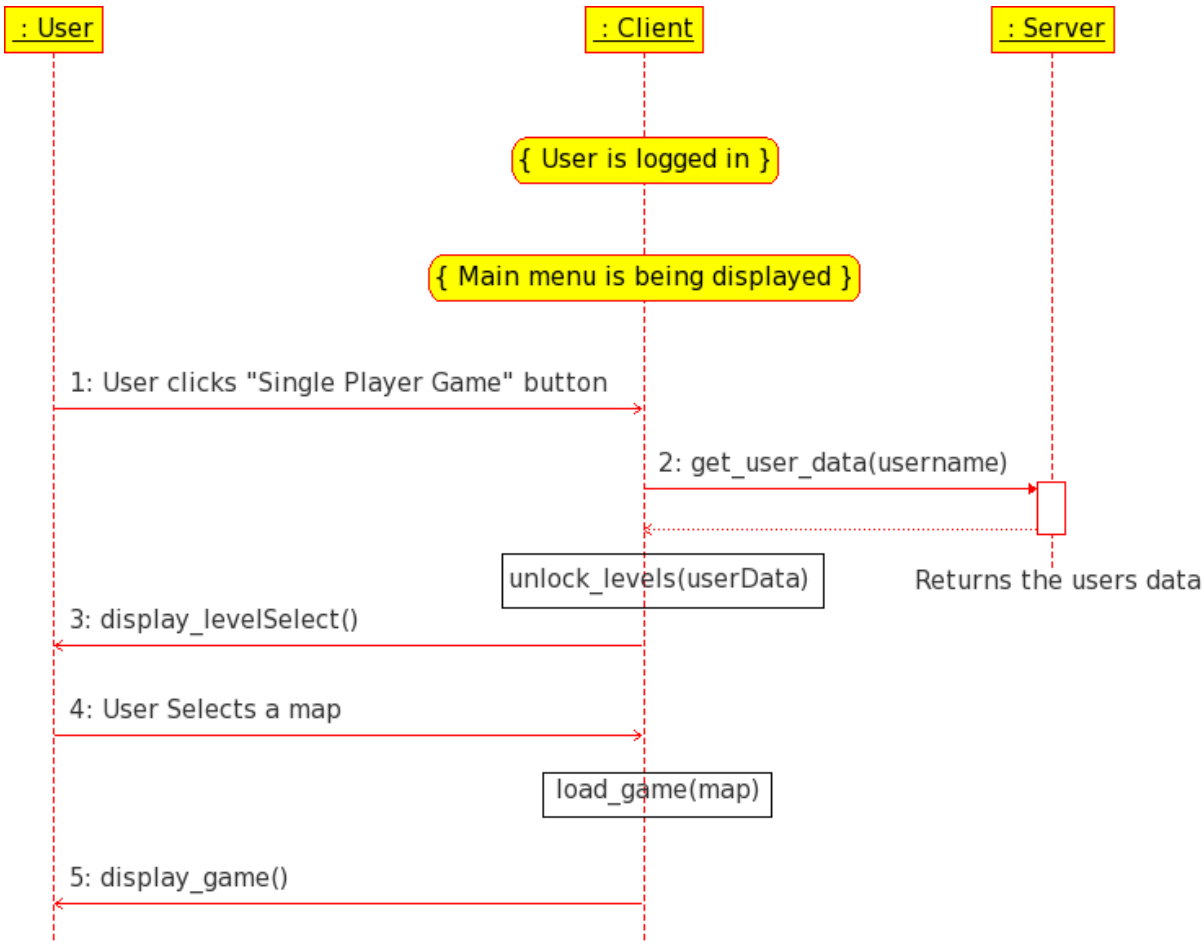
Name: The player wants to play the game alone.
Author: Tasewell Fox
Description: The players interacts with the client to start a single player game..
Actors: Player, Client, Server
Primary Flow:
Happy Path

1. The player selects "Single Player Game" from the main menu
2. The client checks the player's data, and presents the player with the maps he can play
3. The player selects a map
4. The client loads the map data and starts the game

Alternative Flows:
Preconditions: The player has logged in to the client and his player data has been downloaded.
Postconditions: The player is playing the game.

2.2.6.b Use Case 6 (Diagram)

UML Sequence Diagram for Use Case 2.2.6



2.2.7.a Use Case 7 (Text)

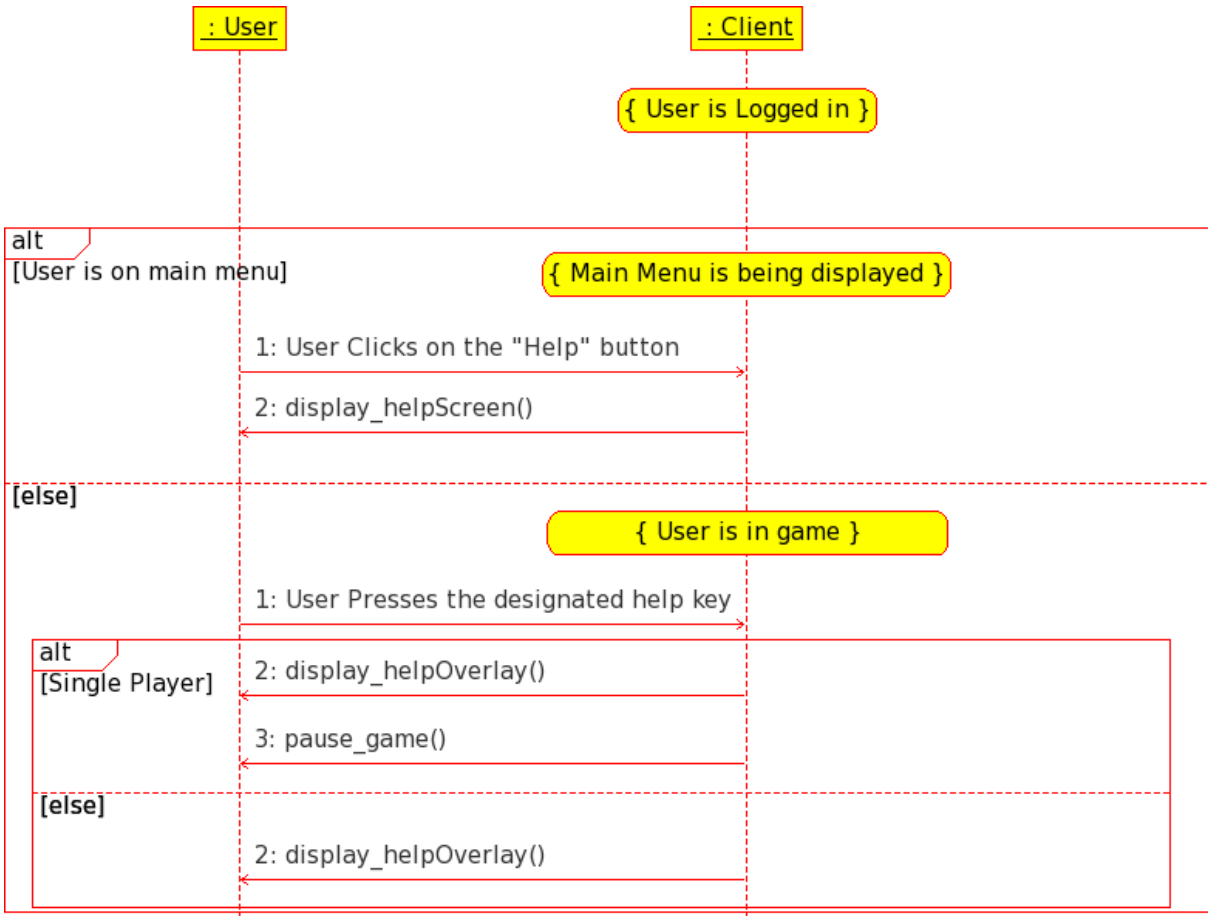
Name: The player wants to know how to play the game.
Author: Cole Anagnost
Description: The player interacts with the client to show the help screen.
Actors: Player, Client
Primary Flow:
Happy Path
1. The player selects "Help" from the main menu
2. The client displays the help screen

Alternative Flows:
During gameplay
1. The player presses a key ('H' for example)
2. The client displays the help screen (and pauses gameplay if single player)

Preconditions: The player has logged in to the client.
Postconditions: The player is at the help screen.

2.2.7.b Use Case 7 (Diagram)

UML Sequence Diagram for Use Case 2.2.7



2.2.8.a Use Case 8 (Text)

Name: The player wants to know statistics about other players / themselves.
Author: Cole Anagnost
Description: The player wants to know about their current statistics or the statistics of other players. The goal of this is to allow the player to track their stats over time to see improvement in their scores, and vie for the opportunity to reach a high score list. It also allows them to gauge the performance of other players to help with matchmaking decisions.
Actors: Player, Client, Server, Database
Primary Flow:
Happy Path:

1. The player clicks on the "High Scores" button on the main menu
2. Server sends a request to the database for the current list of high scores
3. The server displays the list of high scores to the client
4. The player is presented with a window containing the high scores and their current high scores.
5. The player can input another players name into the "Search player" box and click "Search"
6. The server sends a request to the database for that players high score
7. The player can then view that players score

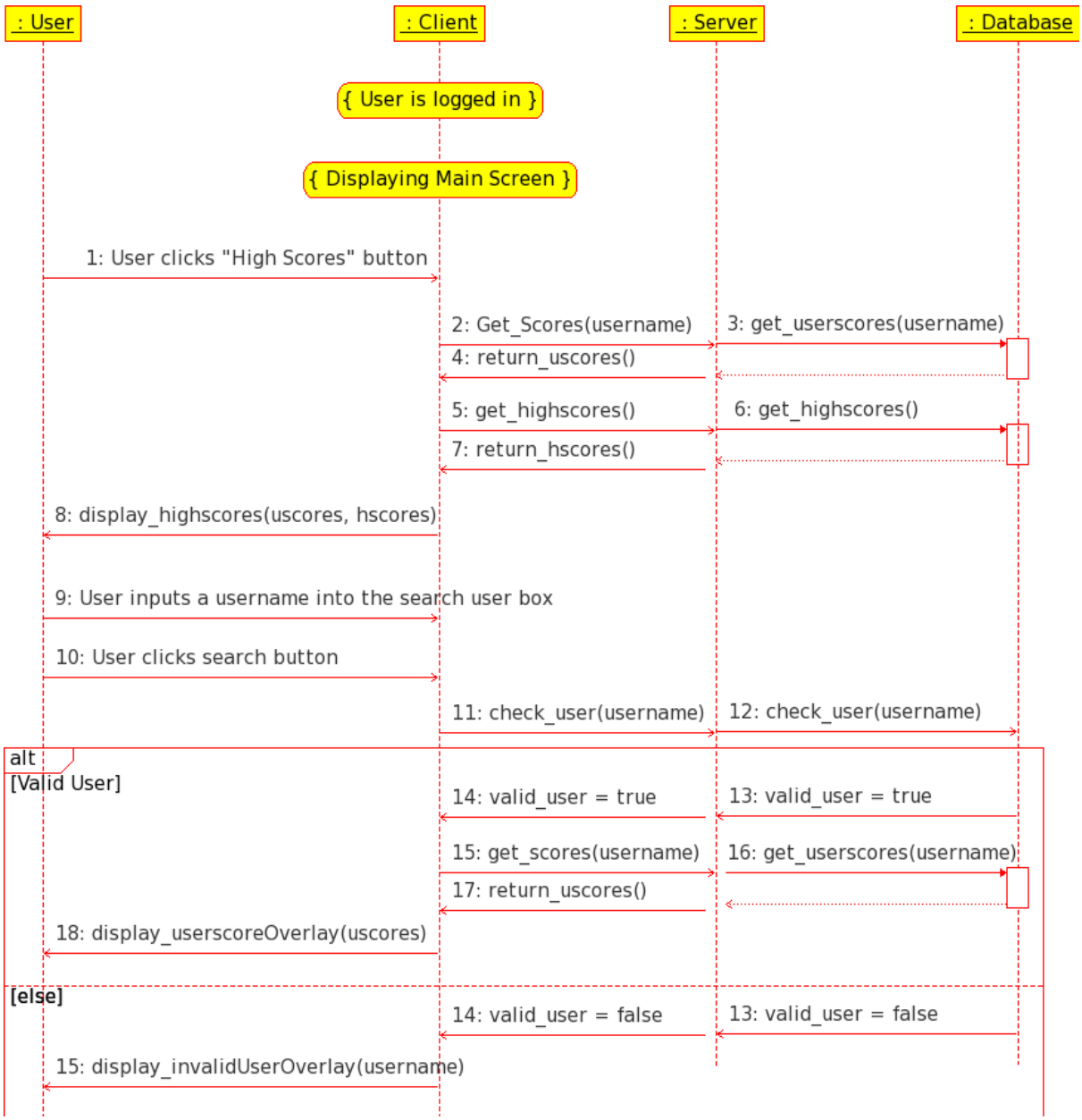
Alternative Flows:
Invalid player:

1. If the player enters an invalid name at step 5 of the *Happy Path* then the player will be presented with an error message stating that the player is invalid.

Preconditions: The player is logged onto the game and on the main screen
Postconditions: The player is presented with a list of high scores

2.2.8.b Use Case 8 (Diagram)

UML Sequence Diagram for Use Case 2.2.8



2.3 USER CHARACTERISTICS

Our typical user will be a casual gamer who wants to play a game that is easy to pick-up play for a short time and then put down. There will also be the more hardcore users who will want to play for several rounds and keep track of their statistics and the statistics of their friends.

2.4 CONSTRAINTS

// all conditions that may limit design options (INCLUDE NON FUNCTION CONSTRAINTS)

2.5 ASSUMPTIONS AND DEPENDENCIES

// hardware and software assumptions and dependencies

3. SPECIFIC REQUIREMENTS

APPENDIX A

A-1 SCREEN FLOW DIAGRAM

A-2 SCREENSHOT MOCKUPS