**Hong Kong Institute of Vocational Education (Chai Wan)**

**HD in Game Software Development (IT114107)**

**ITP4708 Game Servers Design and Implementation**

**EA Project**

****

**Game Title:**

**星辰記憶 • Starry Memories**

**Student Name: Yung Kwan Wai**

**Student Number: 230033193**

**Class: IT114107 - 2A**

Contents

[1. Overview 3](#_Toc185284261)

[*Game Introduction* 3](#_Toc185284262)

[*Technical Architecture* 3](#_Toc185284263)

[*Game Background Story* 3](#_Toc185284264)

[2. Game Play 3](#_Toc185284265)

[*Basic Rules* 3](#_Toc185284266)

[*Victory Conditions* 3](#_Toc185284267)

[3. Communication between clients and server 3](#_Toc185284268)

[*Communication Flow* 3](#_Toc185284269)

[4. Installation and Setup 5](#_Toc185284270)

[*Environmental Requirements* 5](#_Toc185284271)

[*Installation Step* 5](#_Toc185284272)

[*Open Game Step* 6](#_Toc185284273)

[5. Usage Guide 6](#_Toc185284274)

[*Game Operations* 6](#_Toc185284275)

[*Game Interface* 6](#_Toc185284276)

[6. External Libraries and Frameworks 6](#_Toc185284277)

[*Node.js* 6](#_Toc185284278)

[*Express* 7](#_Toc185284279)

[*Mongoose* 7](#_Toc185284280)

[7. Known Issues and Limitations 7](#_Toc185284281)

[*Duplicate names* 7](#_Toc185284282)

[*Player Capacity* 7](#_Toc185284283)

[8. Testing Plan 8](#_Toc185284284)

[9. GitHub Repository 8](#_Toc185284285)

[10. Demo Video 8](#_Toc185284286)

# Overview

## *Game Introduction*

"Starry Memories(星辰記憶)" is a tarot-based turn-based multiplayer memory battle game for two players online. The game combines memory and strategy, by matching the same tarot card scores, the player with the highest score will win.

## *Technical Architecture*

* The front-end is developed using JavaScript, HTML, and CSS.
* The backend is implemented by Node.js and WebSocket, which supports real-time client-server communication.

## *Game Background Story*

Among the vast stars, each tarot card contains a mysterious memory. By turning over the cards, can you recall their corresponding positions and solve the mystery of memory before your opponent becomes a guardian of the stars?

# Game Play

## *Basic Rules*

* At the beginning of the game, the system will randomly generate 10 tarot cards arranged in a 5x2 grid.
* Players take turns turning over two tarot cards:
  + If the two tarot cards are the same, the player gets 1 point and continues to flip.
  + If the two tarot cards are different, the cards will be flipped back to the back, and it will be another player’s turn.
* The game ends when all matched tarot cards are turned over.
* The player with the highest score wins.

## *Victory Conditions*

* At the end of the game, the system will automatically calculate the player’s total score, with the highest scorer winning.
* If the scores are the same, there is a tie.

# Communication between clients and server

## *Communication Flow*

* Clients use the link to connect the server

一張含有 文字, 螢幕擷取畫面, 設備, 設計 的圖片

自動產生的描述

* Set Player Name:

Players send their names to the server when they join the game.

一張含有 文字, 螢幕擷取畫面 的圖片

自動產生的描述

* Game Start:

When two players join the game, the server initializes the game and start the game.

一張含有 文字, 螢幕擷取畫面, 設計 的圖片

自動產生的描述

* Chat Message:

When a player sends a message, it is transmitted to the server, which then relays the message back to both players and saves it in MongoDB's message database.

* Player 1 sends the message

一張含有 文字, 軟體, 多媒體軟體, 螢幕擷取畫面 的圖片

自動產生的描述

* Player 2 sends the message

一張含有 文字, 軟體, 多媒體軟體, 螢幕擷取畫面 的圖片

自動產生的描述

* Update Game State:

When the player flips the card, the action is sent to the server. The server checks the card match, updates the game status, and calculates the score. It generates an update message including the card status, player score, and next player ID, and broadcasts it to both players. After the client receives the message, it updates the interface to display the latest score and card status.

* When player 1 matches the card successfully, then player 1 can match the card again. Player 1's score will add one score, and player 2's card status will show which card is matched.

一張含有 螢幕擷取畫面, 文字, 設計 的圖片

自動產生的描述

* When player 1 matches the card failed, then the round will give player 2.

一張含有 螢幕擷取畫面, 文字, 設計 的圖片

自動產生的描述

* When player 2 matches the card successfully, then player 2 can match the card again. Player 2's score will add one score, and player 1's card status will show which card is matched.

一張含有 文字, 螢幕擷取畫面, 設計 的圖片

自動產生的描述

* Game Over:

Once all matches are found, the server sends the final scores and declares the winner to both players.

# Installation and Setup

## *Environmental Requirements*

* Node.js: Requires at least v18.20.4.
* Browser: Modern browsers that support WebSocket (e.g. Chrome, Firefox).

## *Installation Step*

* Install dependencies:
* npm init -y
* npm install express
* npm install mongoose
* npm install react-dom
* npm install react-scripts
* npm install WebSocket
* text {"build": "react-scripts build"} in scripts with package.json
* text {“server": "node server/server.js"} in scripts with package.json

## *Open Game Step*

* Build The Game:
* npm run build
* Start The Server:
* npm run server
* Run The Client:
* <http://localhost:7101/>

# Usage Guide

## *Game Operations*

* Logging into the Game:
  + Use the chat room, enter the name and click the "Send" button in the chat box.
* Game Process:
  + Click on the card face to flip and attempt to match tarot cards.
  + If a match is successful, the score will update automatically.
  + If the match fails, the cards will flip back, and it will be the opponent's turn.
* Viewing Chat Messages:
  + Enter messages in the chat box and click "Send" to communicate with the opponent.

## *Game Interface*

* Card Area:
  + Displays the layout of the tarot cards.
* Info Area:
  + Show the player’s name, the total round, the real-time scores of both players and the round are who.
* Chat Area:
  + For communication with the opponent.

# External Libraries and Frameworks

## *Node.js*

* Description: Node.js is an open-source JavaScript execution environment that enables developers to run JavaScript on the server side. It makes the development of real-time applications such as online games and chat applications easy.
* Purpose: Used to create servers, handle WebSocket connections and database operations.

## *Express*

* Description: Express is a fast, flexible Node.js web application framework that provides a set of powerful yet simple features for building web applications.
* Purpose: Handle HTTP requests, serve static files, and set up a WebSocket server.

## *Mongoose*

* Description: Mongoose is an object modelling tool for MongoDB that provides an intuitive API for Node.js to operate the database.
* Purpose: Used to define data models (such as messages and game records) and perform database operations.
* WebSocket
* Description: WebSocket is a communication protocol that provides a full-duplex communication channel for use in real-time applications.
* Purpose: Allows real-time data transmission between the server and the client, supporting real-time interaction in the game.
* React
* Description: React is a JavaScript library for building user interfaces, focusing on building reusable UI components.
* Purpose: Used to build the front-end interface of the game, including cards, chat rooms and game information display.
* CSS
* Description: Cascading Style Sheets (CSS) is a style sheet language used to describe the display style of HTML documents.
* Purpose: Used to design the appearance and layout of the game to make the interface more attractive and easier to use.

# Known Issues and Limitations

"Starry Memories" game server, while operational and providing an engaging gaming experience, has certain limitations and areas that could benefit from further refinement:

## *Duplicate names*

The current version of the game allows players to have duplicate names, which can lead to player confusion. In the future, the checking of entered names will be improved to prevent this from happening.

## *Player Capacity*

At present, the game is designed to accommodate only two players simultaneously. This limitation could be addressed by redesigning the game to support more players, enhancing the multiplayer experience.

# Testing Plan

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case ID** | **Feature to Test** | **Test Steps** | **Expected Result** | **Tested Result** |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

# GitHub Repository

<https://github.com/SnowYung/ITP4708_Game-Server.git>

# Demo Video

[..\VideoDemo\Game Server Video.mkv](../VideoDemo/Game%20Server%20Video.mkv)