ASE Project Report - GROUP 2

November 20, 2025

Contents

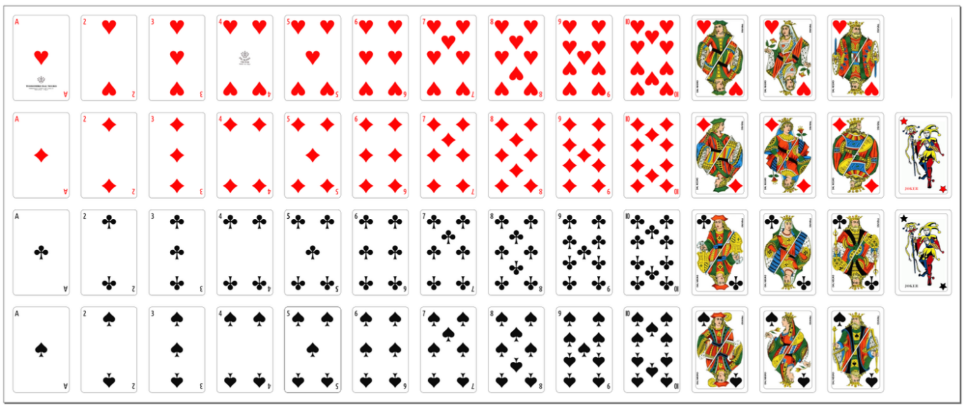
1 Group Members and Configuration

1.1 Group Members

First Name	Last Name	Student ID	E-Mail
Filippo	Morelli	608924	f.morelli38@studenti.unipi.it
Federico	Fornaciari	619643	f.fornaciari@studenti.unipi.it
Ashley	Spagnoli	655843	a.spagnoli9@studenti.unipi.it
Marco	Pernisco	683674	m.pernisco@studenti.unipi.it

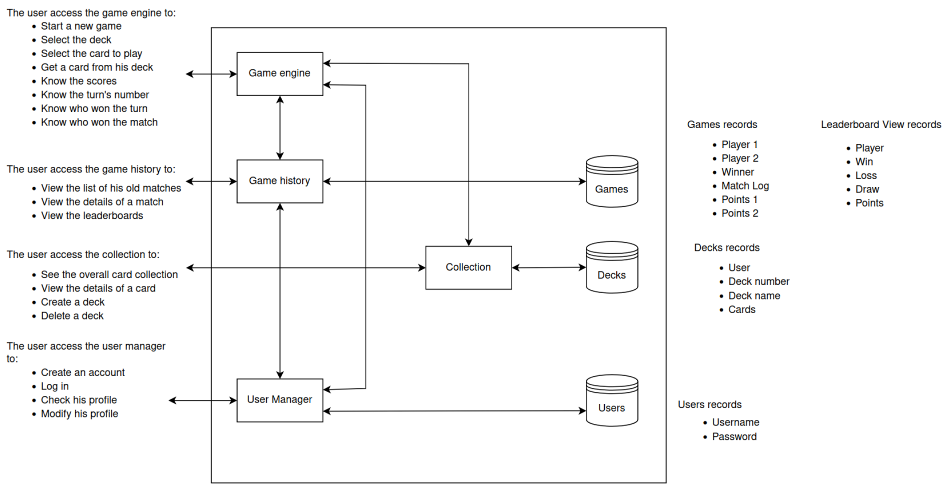
1.2 Set of Cards

The deck configuration used for this project includes 52 cards + Joker.



2 System Architecture

Below is the high-level architectural drawing of the system, illustrating the communication between microservices and the requests the users can perform.



3 Rules of the Game

3.1 Card Game Rules

3.1.1 Deck Building Constraints

Each player constructs a **personal deck** of exactly **9 cards**. The deck must adhere to the following constraints:

- **Composition:** The deck must contain exactly **1 Joker** and **8 Suited Cards**.
- **Suit Distribution:** You must include exactly **2 cards** from each suit ($\heartsuit$, $\diamondsuit$, $\clubsuit$, $\spadesuit$).
- **Cost Limit:** The combined point value of the two cards in any single suit **must not exceed 15**.

Card Point Costs
When calculating your deck limits, use the following costs (Note: Face cards have specific costs despite their combat strength):

Card Type	Ranks	Cost per Card	Example Pair Limit
Numbers	2 – 10	Face Value	$10 + 5 = 15$ ✓
Ace	A	7	$A + 8 = 15$ ✓
Jack	J	11	$J + 4 = 15$ ✓
Queen	Q	12	$Q + 3 = 15$ ✓
King	K	13	$K + 2 = 15$ ✓

3.1.2 Combat Hierarchy

When two cards are played, the winner is decided by the following hierarchy:

1. The Combat Triangle The core mechanics function like Rock-Paper-Scissors:

- **Numbers (2–10)** beat **Aces**.
- **Aces** beat **Face Cards (J, Q, K)**.
- **Face Cards (J, Q, K)** beat **Numbers**.

2. The Joker The **Joker** beats all other cards automatically. (If both players play a Joker, it is a tie).

3. Tie-Breakers If the combat rules above do not determine a clear winner (e.g., Number vs Number, or Face vs Face), compare the specific ranks:

1. **Higher Rank Wins:** (e.g., 9 beats 6, King beats Jack).
2. **Equal Rank → Suit Priority:** If ranks are identical (e.g., $\heartsuit$9 vs $\diamondsuit$9), check suits:

$$\heartsuit > \diamondsuit > \clubsuit > \spadesuit$$

3. **Mirror Match:** If players play the *exact same card* (Rank and Suit), both players win the round and gain a point.

3.1.3 Gameplay Flow

1. **Setup:** Both players shuffle their pre-built decks.
2. **Initial Draw:** Each player draws **3 cards** to form their starting hand.
3. **The Round:**
   - **Draw Phase:** At the start of every turn (including Turn 1), players draw **1 card**.
   - **Battle Phase:** Both players play one card **face down**, then reveal simultaneously.
   - **Scoring:** Determine the winner based on the Combat Hierarchy. The winner earns **1 point**.
4. **Victory:** The game ends immediately when a player reaches **5 points**.

3.1.4 Example Match Log

Alice's Deck: ♡A, ♡7, ♢A, ♢7, ♣K, ♣2, ♠K, ♠2, Joker.
Bob's Deck: ♡2, ♡3, ♢2, ♢3, ♣3, ♣4, ♠2, ♠3, Joker.

Turn	Alice	Bob	Result Reasoning	Score (A-B)
1	♡A	♣K	**Ace beats Face** (Special Rule)	1 – 0
2	♢A	♢3	**Number beats Ace** (Special Rule)	1 – 1
3	♣K	♣3	**Face beats Number** (Special Rule)	2 – 1
4	♠K	♠2	**Face beats Number** (Special Rule)	3 – 1
5	♡7	♣4	Both Numbers: $7 > 4$	3 – 2
6	♢7	♠3	Both Numbers: $7 > 3$	3 – 3
7	♣2	Joker	**Joker beats Everything**	3 – 4
8	Joker	♡2	**Joker beats Everything**	4 – 4
9	♡7	♠3	Both Numbers: $7 > 3$	**5 – 4**

4 Build and Run Instructions

This project is a multi-service card game platform designed for two players. It features authentication, deck building, a match-simulation engine, and history tracking. All services are containerized and orchestrated via Docker Compose.

4.1 Prerequisites

Before starting, ensure your environment meets the following requirements:

- **Docker & Docker Compose** (Required for orchestration).
- **Python 3.10+** (Optional, only required for local non-containerized development).

4.2 Quick Start Guide

1. **Clone the repository**

```
git clone https://github.com/ashleyspagnoli/ASE_project.git
cd ASE_project/src
```

2. **Build and launch services** Run the following command to build the images and start the containers:

   ```
   docker compose up --build
   ```

3. **Verify Service Status** Once the containers are running, the architecture exposes the following endpoints:

Service Name	Responsibility	Local URL
User Manager	Authentication & JWT	`https://localhost:5004`
Collection	Deck Management	`http://localhost:5003`
Game Engine	Core Logic & Matchmaking	`http://localhost:5001`
Game History	Match Logging	`http://localhost:5002`

4.3 Development & Testing

4.3.1 Environment Configuration

Each microservice is configured via environment variables. For specific configuration keys, refer to the `Dockerfile` and `requirements.txt` located in each service's directory.

4.3.2 Testing the Workflow

A Postman collection is provided for end-to-end testing. Import `game_workflow.postman_collection.json` into Postman to simulate a full lifecycle:

- User Registration and Login (Token generation).
- Deck creation and validation.
- Matchmaking and gameplay simulation.

4.3.3 Key API Endpoints

Authentication

- `/users/register`: user registration [POST]
- `/users/login`: user login [POST]
- `/users/validate-token`: internal JWT token validation [GET]

Deck Management

- `/collection/cards`: get the collection of cards [GET]
- `/collection/decks`: create a new deck [POST]

Game Engine

- `/game/connect`: connect to start playing the game [POST]
- `/game/matchmake`: manual request to start the match [POST] (TO REMOVE)

- `/game/play/{game_id}`: play a card [POST]
- `/game/state/{game_id}`: get the state of the game [GET]

Game History

- `/leaderboard`: get the whole leaderboard [GET]
- `/matches`: get the history of your played matches [GET]
- `/addmatch`: memorize a new match [POST]

4.4 Maintenance

To stop the application and remove containers/networks, run:

```
docker compose down
```