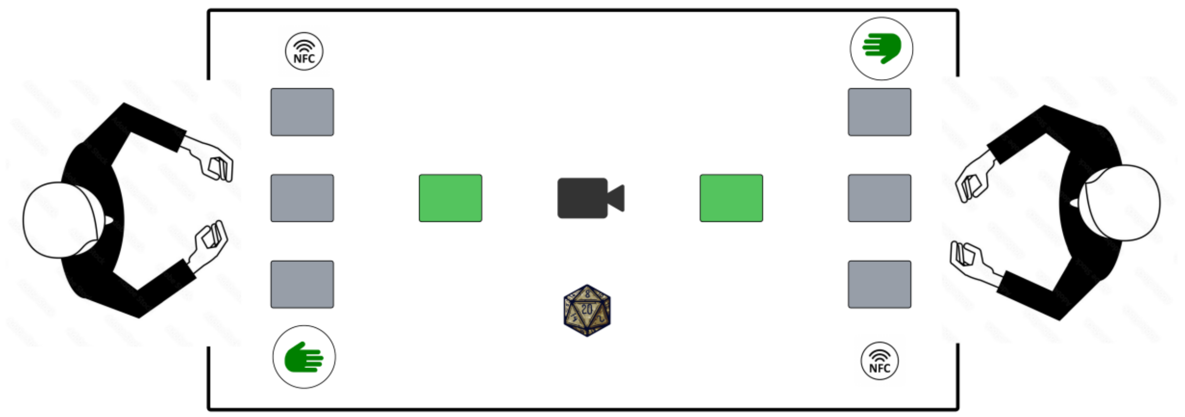
CSE396 COMPUTER ENGINEERING PROJECT

2025 SPRING – GROUP 6 - Report 2

DiceForge

“Arcane Gambit”

IBRAHIM AL SAID	200104004805
ÖZLEM KURUCA	220104004068
SÜMEYYE ZEYNEP GÜRBÜZ	200104004039
HAKAN KESER	220104004073
TARIK ÇELİK	210104004271
AHMET BAHA ÇEPNİ	210104004056
BARIŞ EREN GEZİCİ	210104004041
SEFA AKGÜN	210104004026



Modules
Backend

Computer Vision
Mobile Application
Augmented Reality

1 Unreal Engine: Game Logic and base	- sefa - hakan - baha
2 Mobile Application	- özlem - ibrahim - barış - zeynep
3 Computer Vision-	barış - Tarık - hakan
4 Backend server	hakan - tarık - ibrahim
5 AR -	Zeynep - baha - sefa - özlem

Backend Module

This module is used for managing the server-side operations of the game. It helps users to register,character creation, log in, and enter a game session. All information is stored in a MongoDB database, and the server is built using Express.js.

The main goal of this module is to create a flexible backend system. It allows users to create accounts with their username and password,create character by choosing a character, log in safely, and join a game session. When a user enters a session, the backend can save information like the user's ID and character and the time the session started.

Technologies Used:

- **Node.js** – JavaScript runtime environment
- **Express.js** – Server framework for Node.js
- **MongoDB** – For database

Main Features:

1. **Register**
   Users can create a new account by entering their username, password and choosing their character.
2. **Login**
   Users can log in using their username and password. If successful, then user can enter a game session.
3. **Enter Game Session**
   After logging in, users can join a game session with scanning NFC on game table. The backend saves information about the session, like the time and user ID.

Classes / Schemas

1. User Schema (models/User.js)

```
{
  username: String,       // Unique username
  password: String,        //password
  character: String,
}
```

- **Methods (Controller-side):**
  - registerUser(req, res)

    - loginUser(req, res)

2. Session Schema (models/Session.js)

```
{
  userId: Object ID,  // Reference to the user
  joinedAt: Date                          // Timestamp when session started
}
```

- **Methods (Controller-side):**
    - enterSession(req, res)

API Methods (Express Route Handlers)

1. POST /register – Register a new user

Method: registerUser(req, res)
Parameters:

- username: string
- password: string
  Returns: success message or error

2. POST /login – Log in an existing user

Method: loginUser(req, res)
Parameters:

- username: string
- password: string
  Returns: success message or error

3. POST /session – Enter a game session

Method: enterSession(req, res)
Headers:

- userId
  Returns: confirmation or error