भारतीय सूचना प्रौद्योगिकी संस्थान गुवाहाटी
INDIAN INSTITUTE OF INFORMATION TECHNOLOGY GUWAHATI

GridWars Gaming System

A Turn-Based Strategy Game

Project Report

Submitted By:

Abhinav Pangaria (Roll No: 2201005)

Ashish Ranjan Kumar (Roll No: 2201039)

Ashutosh Kumar (Roll No: 2201040)

Divyanshu Vyas (Roll No: 2201068)

Submitted To:

Prof. Shubha Brata Nath

Department of Computer Science and Engineering

Academic Year 2025-26

Assignment Questions

==

Business Logic Layer (BLL) Analysis for GridWars

A ”business logic layer” (BLL) is a part of a software application architecture where the core business rules and logic are implemented, acting as a mediator between the presentation layer (user interface) and the data access layer (database), ensuring that data manipulation and processing adheres to the specific business requirements and guidelines of the applications.

Q1. Core Functional Modules and Their Interaction with the Presentation Layer

— Core Functional Modules:

- **User Management Module:**
  - Handles user registration, log-in, and profile updates

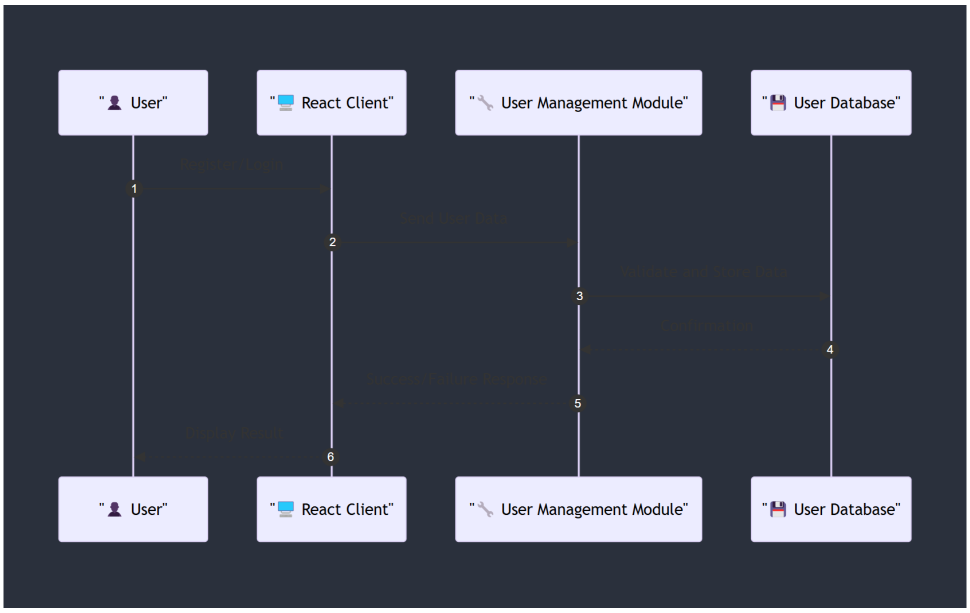

Figure 1: User Management Module Interaction

- **Game Management Module:**
  - Manages game sessions, state transitions, and move validation

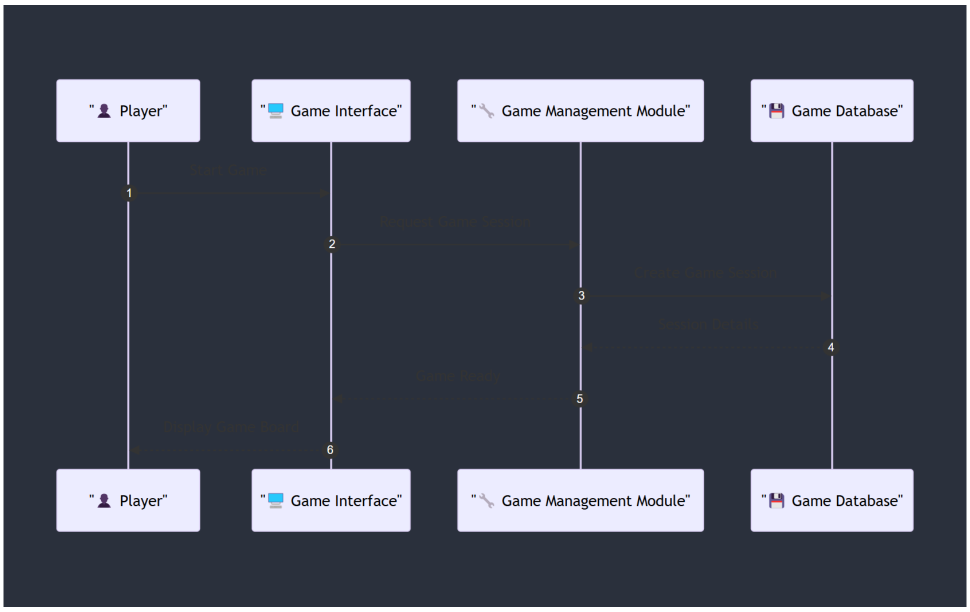

Figure 2: Game Management Module Interaction

- **Chat Management Module:**
  - Facilitates real-time messaging and chat history

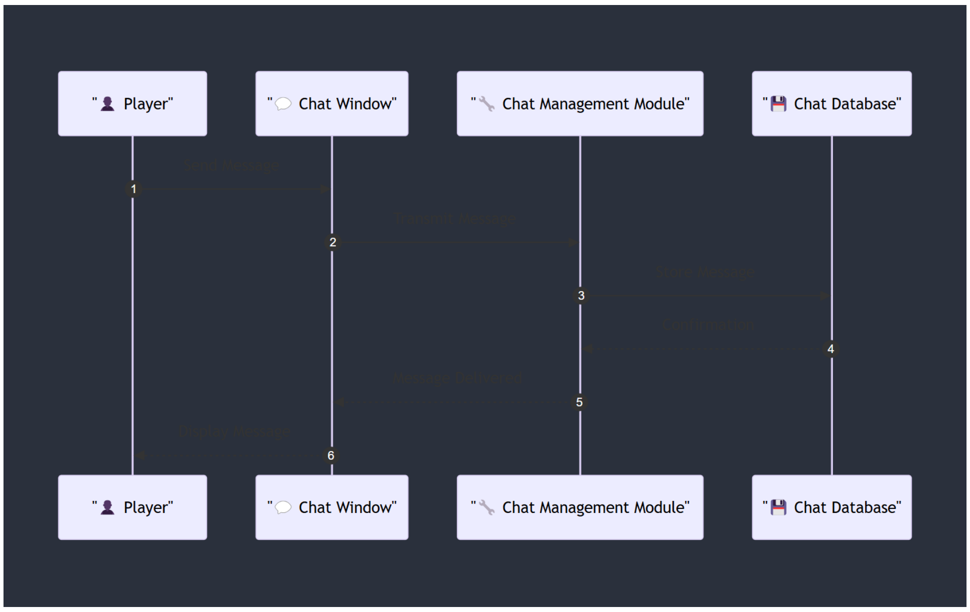

Figure 3: Chat Management Module Interaction

- **Leaderboard Management Module:**
  - Updates player rankings and statistics

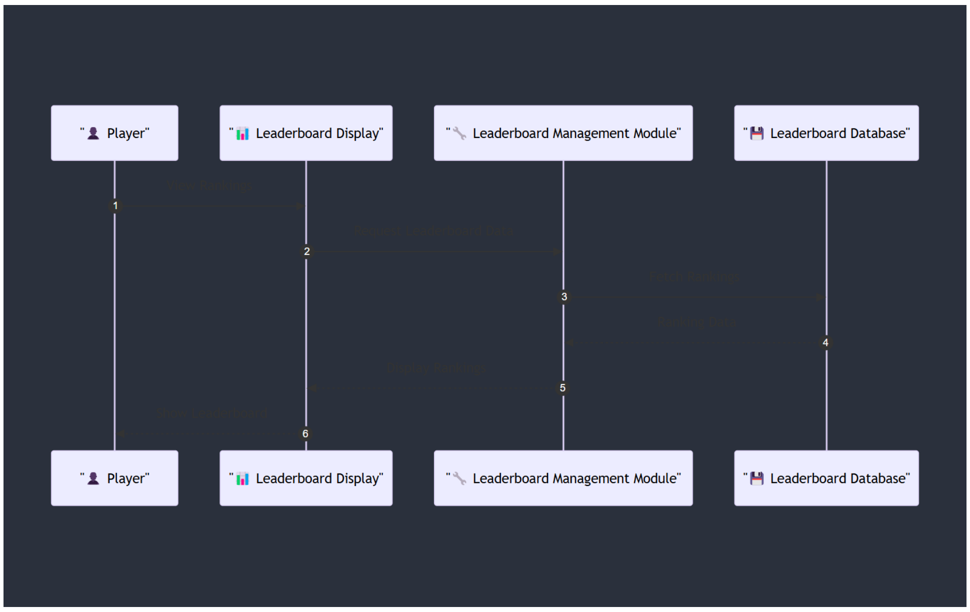

Figure 4: Leaderboard Management Module Interaction

— Interaction with Presentation Layer:

- **User Interface Components:**
  - React components interact with BLL via API calls
  - WebSocket connections for real-time updates
- **Example Code:**

Listing 1: User Management Module

```
class UserManagement {
    async registerUser(userData) {
        // Validate user data
        if (!this.validateUserData(userData)) {
            throw new Error('Invalid user data');
        }
        // Interact with data layer
        const user = await UserModel.create(userData);
        return user;
    }

    validateUserData(data) {
        // Basic validation logic
        return data.email && data.password && data.username;
    }
}
```

Listing 2: Game Management Module

```
class GameManagement {
    async createGameSession(players) {
        // Validate players
        if (!this.validatePlayers(players)) {
            throw new Error('Invalid players');
        }
        // Create game session
        const session = await GameSessionModel.create({ players });
        return session;
    }

    validatePlayers(players) {
        // Ensure valid player list
        return Array.isArray(players) && players.length > 0;
    }
}
```

Q2. Business Rules, Validation Logic, and Data Transformation

A) Business Rules Implementation:

- **User Access Control:**
  - Users must be authenticated to access game sessions
  - Admins have additional privileges for user management
- **Game Rules:**
  - Moves must be valid according to game rules
  - Game state transitions are controlled by the BLL
- **Leaderboard Updates:**
  - Rankings are updated based on game outcomes
  - ELO rating system is used for ranking calculations

B) Validation Logic:

- **Data Validation:**
  - User input is validated for format and completeness

    - Game moves are validated for legality

- **Example Code:**

Listing 3: Validation Logic Example

```
class ValidationService {
    validateEmail(email) {
        const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
        return emailRegex.test(email);
    }

    validateMove(move) {
        // Check if move is within game rules
        return move && move.isValid;
    }
}
```

C) Data Transformation:

- **Data Formatting:**
    - Data from the database is transformed into UI-friendly formats
    - JSON responses are structured for easy consumption by the frontend
- **Example Code:**

Listing 4: Data Transformation Example

```
class DataTransformer {
    transformUserData(user) {
        return {
            id: user._id,
            username: user.username,
            email: user.email,
            elo: user.elo
        };
    }

    transformGameData(game) {
        return {
            id: game._id,
            players: game.players,
            state: game.state
        };
    }
}
```