FAKULTI TEKNOLOGI DAN KEJURUTERAAN ELEKTRONIK DAN KOMPUTER
SESSION 2023/2024 SEMESTER 2
BENR2423 DATABASE AND CLOUD SYSTEM

ASSIGNMENT PROJECT EVALUATION FORM **Group Members:**

Student Name	Student Number	Signature
NUR ANIS IDYANA BINTI ABDUL RASHID (developer)	B022210129	anis
WAFA BINTI EZIZAM (developer)	B022210161	wafa
NURATIQAH BINTI MUHAMMAD MUSLIM (developer)	B022210149	tiqah
NUR HASNI BATRIESYIA BINTI ZULKIFLI (developer)	B022210167	hasni
SITI QYRUNNISAE BINTI ABD RASHID (player)	B022210158	nisae
NUR AIN FAQIHAH BINTI MASHLINO (player)	B022210181	qihah
NUR FAIQAH BINTI JAFLIN (hacker)	B022210118	faiqah
NOR AINI BINTI SYARIF (hacker)	B022110155	aini

	Rubrics					CLO/PO	Marks
Cloud and database development skillset	A complete application with clear and precise comment, main program functions. Detail development progress, track records and team works can be seen on Github.	A good application developed with most of the main program functions has been presented. Important development details can be seen on Github, but lacking team working records.	An average application with lacking few important API. Partial implementation of application security. Not much development progress can be seen on Github.	A basic application with only few APIs. No consideration on the application security. No development progress can be seen on Github.	No submission	CLO3/PO5	/10
	4	3	2	1	0		
Project Functionality	The system is hosted on the cloud and working properly. All test procedures are provided in Postman.	The design of system is carried out, but partially working. Most test procedures are provided in Postman.	The design of system is carried out, but partially working. No procedures are provided.	The design of system is carried out, but not working accordingly. No procedures are provided.	No submission	CLO3/PO5	/10
	4	3	2	1	0		
Total (x/20)							

TITLE: BATTLESHIP GAME SYSTEM

OBJECTIVE

1. To create an interactive Battleship game system that supports multiplayer gameplay
2. To create a backend system using RESTful principles to handle game-related interactions.
3. To implement authentication and authorization ensuring that only authenticated and authorized players can access and interact with the game.
4. To host the working version of the RESTful API on the Microsoft Azure Cloud platform.

INTRODUCTION

Battleship is a game of strategic guessing played on a grid where two players position their ships and then take turns attempting to identify the whereabouts of their opponent's ships. Every player possesses their individual grid, which is segmented into rows and columns, along with a fleet of ships in different sizes. At the beginning of the game, players position their ships on their grid privately, keeping their placements hidden from the opponent. Throughout the game, participants alternate between calling out specific grid coordinates to aim at and attack their opponent's ships. The rival replies with "hit" if a ship is present at the selected coordinate or "miss" if it is not. If a ship is in the designated area and every part of it is hit, it is classified as sunk. Players record their own guesses and the opponent's responses on a different grid to plan.

The aim is to make all the rival's ships sink before they sink yours. The game goes on until one player manages to sink all the other player's ships and win. The combination of strategy, prediction, and luck creates an engaging and enduring game of naval battle in Battleship.

Our platform uses an advanced database system to create a smooth and fun environment for players around the globe, combining traditional gameplay with contemporary technology. The core of our platform is a robust, scalable database that effectively oversees all game elements, including player profiles, game states, leaderboards, and matchmaking. This strong database guarantees that all actions, shots, and wins are logged immediately, enabling seamless and continuous gaming. Thanks to our advanced data management systems, players

can compete against opponents worldwide, participate in strategic battles, and easily monitor their progress.

The game provides various modes such as ranked matches, casual play, and thrilling tournaments, appealing to competitive players and casual fans alike. Our advanced matchmaking system utilizes player statistics from the database to match opponents with comparable skill levels, guaranteeing exciting and fair matches. Player performance is carefully monitored with in-depth statistics on wins, losses, accuracy, and duration, enabling players to assess their skills and enhance their tactics. Ensuring security is our platform's focus. We use state-of-the-art encryption and authentication protocols to protect player data and ensure the security of all transactions. Our database structure is created to grow as needed, accommodating numerous players simultaneously while maintaining high performance and reliability.

ENTITY DIAGRAM

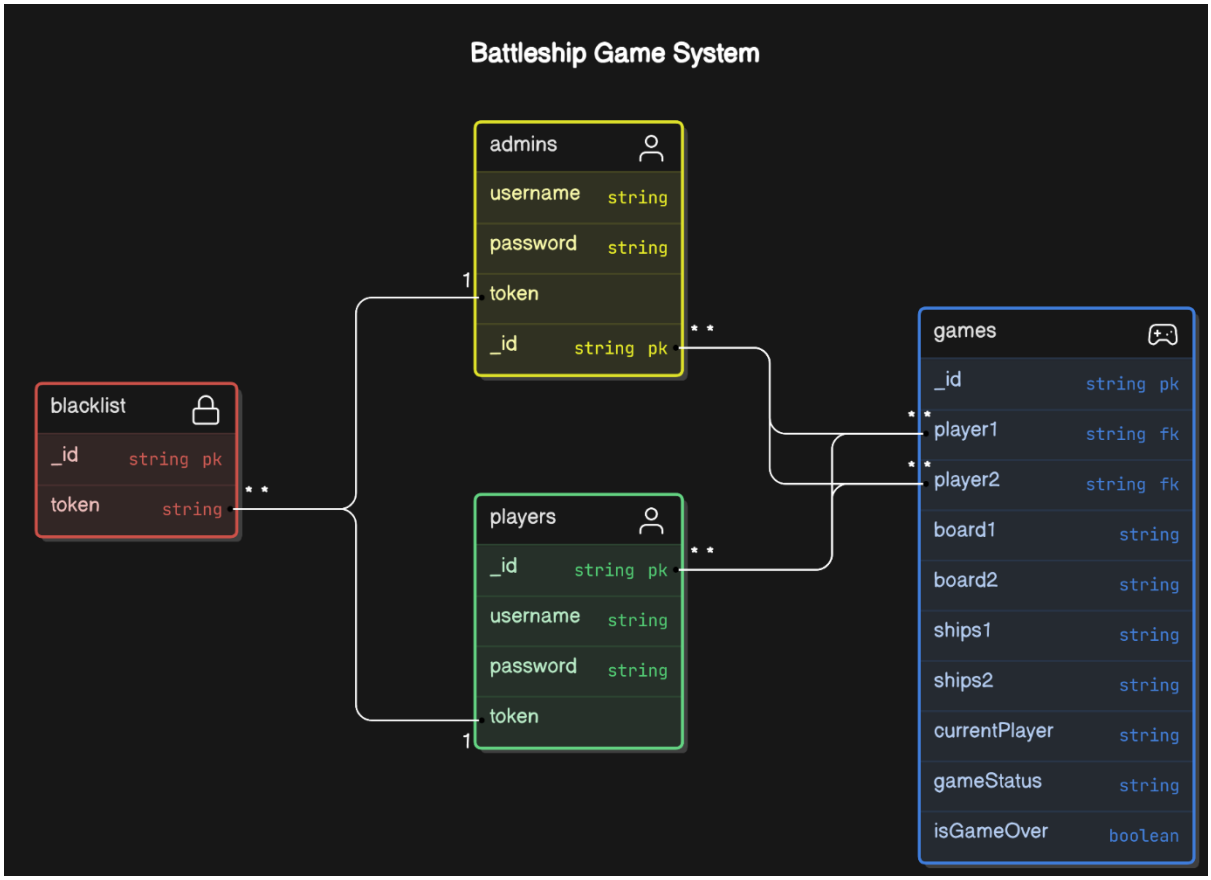

Figure 3

The database structure for a Battleship game system appears in the diagram. The collections are called "admins," "players," "games," and "blacklist." User information such as password, username, and token are stored in the "admins" and "players" collections using the primary key "_id." The "blacklist" collection, which has tokens for managing blocked users, is connected to both collections. If the token is valid and not blacklisted, it allows the request to proceed to the next middleware or route handler. If the token is missing, invalid, or blacklisted, it sends an appropriate HTTP response with an error status and message. With player references connected as foreign keys, the "games" collection maintains game-specific data such as player references (player1, player2), boards (board1, board2), ships (ships1, ships2), currentPlayer, gameStatus, and isGameOver. This schema effectively arranges users, game states, and blocked users.

SEQUENCE DIAGRAM

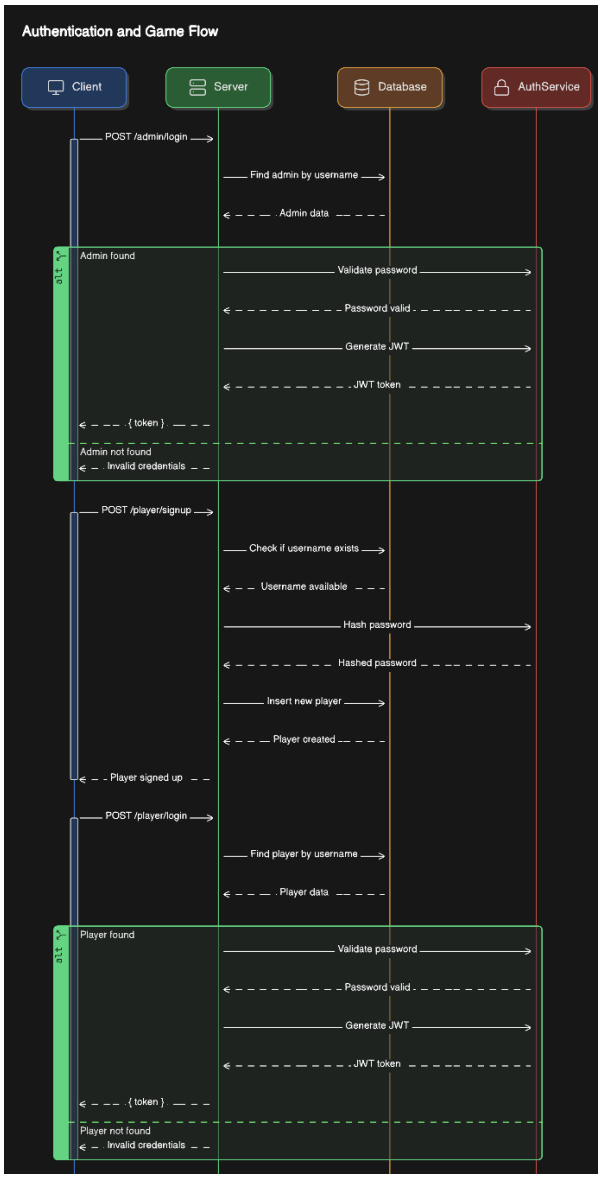

Figure 1

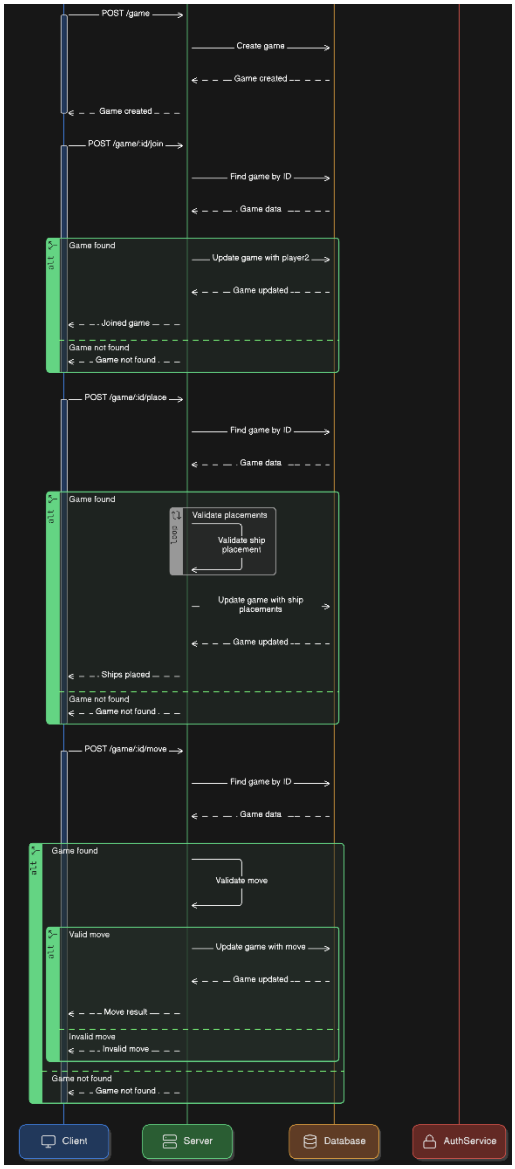

Figure 2

The sequence of events involving the client, server, database, and authentication service for both game flow and authentication in a Battleship gaming system is depicted in the diagram. The process starts with an admin login (POST /admin/login), during which the server retrieves the admin information, verifies the password, and, if successful in authentication, generates a JWT token. Comparably, establishing a new player, hashing the password, and determining whether the username already exists are all part of the player signup process (POST /player/signup). A JWT token is issued by the server upon successful player login (POST /player/login), after validating the credentials. Game data must be stored in the database as part of the game creation (POST /game) process. Gamers join games (POST /game/{id}/join) by adding the second player's details to the game record. Moving validation (POST /game/{id}/move) and ship placement (POST /game/{id}/place) require retrieving

game data, validating actions, and updating the game state accordingly.

TESTING ENDPOINT USING POSTMAN

1. Admin Login:

- Method: POST
- URL: https://battleship-game.azurewebsites.net/admin/login
- Body:

```
json
Copy code
{
  "username": "adminUsername",
  "password": "adminPassword"
}
```

- Description: Endpoint to authenticate an admin user. Only admins can login; admin roles are manually inserted into MongoDB.

- Admin Logout:

  - Method: POST
  - URL: https://battleship-game.azurewebsites.net/admin/logout
  - Description: Endpoint for logging out an administrator. It ensures that once logged out, the admin's token is immediately invalidated and blacklisted. If there are more than one admin, one admin will not be able to log out another admin.

- Player Sign Up:

  - Method: POST
  - URL: https://battleship-game.azurewebsites.net/player/signup
  - Body:

```
json
Copy code
{
  "username": "playerUsername",
  "password": "playerPassword"
}
```

  - Description: Endpoint for player registration. Each player receives a unique ID for future reference; usernames must be unique.

- Player Login:
  - Method: POST
  - URL: https://battleship-game.azurewebsites.net/player/login
  - Body:

```
json
{
  "username": "playerUsername",
  "password": "playerPassword"
}
```

  - Description: Endpoint to authenticate a player and retrieve a unique token for each login session.

- Create a Player (Admin only):
  - Method: POST
  - URL: https://battleship-game.azurewebsites.net/player
  - Body:

```
json
{
  "username": "newPlayerUsername",
  "password": "newPlayerPassword"
}
```

  - Description: Endpoint for admin to create a new player. Only accessible to admin users.

- Get All Players (Admin only):
  - Method: GET
  - URL: https://battleship-game.azurewebsites.net/players
  - Description: Endpoint to retrieve a list of all players. Accessible only to admin users.

- Update Player by ID:
  - Method: PATCH
  - URL: https://battleship-game.azurewebsites.net/player/{playerId}
  - Body:

```
Copy code
{
  "username": "newUsername",
  "password": "newPassword"
```

```
}
```

- Description: Endpoint to update player information by player ID. Requires admin or player authentication. It allows players to update their own data while admins to update any player's data

- Delete Player by ID (Admin only):
  - Method: DELETE
  - URL: https://battleship-game.azurewebsites.net/player/{playerId}
  - Description: Endpoint for admin to delete a player by player ID. Only admins are authorized to perform this action.

- Create Game:
  - Method: POST
  - URL: https://battleship-game.azurewebsites.net/game
  - Description: Endpoint for creating a game from one of the players. The player who creates the game becomes its owner.

- Get Game by ID:
  - Method: GET
  - URL: https://battleship-game.azurewebsites.net/game/{gameId}
  - Description: Endpoint to retrieve game information by game ID. Shows game status and other details. It is accessible only to admins and players.

- Join Game:
  - Method: POST
  - URL: https://battleship-game.azurewebsites.net/game/{gameId}/join
  - Description: Endpoint for a player to join an existing game. Prevents the game owner from joining their own game and restricts access if the game is already full.

- Place Ships in Game:
  - Method: POST
  - URL: https://battleship-game.azurewebsites.net/game/{gameId}/place

- Body:

  { "placements": [{ "x": 0, "y": 0, "direction": "horizontal", "shipName": "Aircraft Carrier", "size": 5 }, { "x": 1, "y": 0, "direction": "horizontal", "shipName": "Battleship", "size": 4 }, { "x": 2, "y": 0, "direction": "horizontal", "shipName": "Cruiser", "size": 3 }, { "x": 3, "y": 0, "direction": "horizontal", "shipName": "Submarine", "size": 3 }, { "x": 4, "y": 0, "direction": "horizontal", "shipName": "Destroyer", "size": 3 }] }

- Description: This endpoint allows each player in a specific game (:id) to place their ships simultaneously during the setup phase. The endpoint enforces access control and provides feedback based on the player's status and actions within the game. When the first player successfully places their ships, the endpoint responds with "Ships placed successfully". This player then waits for the opponent to place their ships. To check if the opponent has placed their ships, the player needs to send another request to the server. If the opponent has placed their ships, the server responds with "The game has started". At this point, the player can proceed to other game-related endpoints, such as /move.

- Make Move in Game:
  - Method: POST
  - URL: https://battleship-game.azurewebsites.net/game/{gameId}/move
  - Body

```
{
  "x": 0,
  "y": 0,
}
```

  - Description: This endpoint allows a player in the game (:id) to make a move by targeting a specific grid coordinate on their opponent's board. The endpoint enforces turn-based gameplay, validates coordinates, and provides feedback on the outcome of the move. The player who successfully placed their ships first gets the privilege to make the first move in the game. If the player targets a grid coordinate that has already been

targeted (hit or miss), the endpoint returns "Already targeted the grid". If the player inputs coordinates that are out of the grid boundaries (0 to 9), the endpoint returns "Invalid coordinate, must be 0 to 9". The current player can view their own board, which displays the placements of their own ships. The player can view the opponent's board, which only shows the hit or miss grid that has been targeted. If a player successfully targets all the ships of their opponent (after hitting the last ship target), they win the game. The endpoint then returns a congratulatory message displaying the username of the winning player. Upon game completion, the winning player receives immediate feedback. The other player is notified once they send a subsequent request, confirming the game's end.

GITHUB LINK

https://github.com/vampskitty/Battleship-Game

AZURE URL

https://battleship-game.azurewebsites.net/

PLAYER'S REPORT

In this assignment, 2 players need to find a few mistakes or bug from the game or the coding that has been made by developer from their group. So, from the game and coding itself, we detected a few mistakes

1. The grid for this battleship game is difficult to understand. From our experience playing the game, we discovered that the 10 by 10 grids' rows positioned side by side rather than below the others. It is difficult for players to identify the coordinate of the mark on the grid.

2. The game state does not update correctly after a move, such as not recording hits or misses accurately. From our experience, we as a player can only know whether we hit or misses the opponent ships, but the game does not state if the opponent hit or miss our ship. We can just see that from the grid of battleship.

3. As a player, we might get logged out but still be able to perform actions that should be restricted if the token blacklisting logic is not correctly implemented or checked in protected routes. This can occur when the system fails to properly add the token to a blacklist upon logout or does not consistently check the blacklist in all protected routes, allowing the player to continue using an invalid token.

The coding part:

```
const token = req.header('Authorization').replace ('Bearer ', '');
if (blacklist.includes(token)) {
        return res.status(401).send ('Token is blacklisted');
}
```

4. As a player, we cannot know whose turn should go first. Based on what we experience as a player, it might be difficult for both players if we are playing the game at different places. We can only take the first turn if we are the one whose make the game then the opponent can join the game and automatically take the second turn.
5. We noticed that when one player finishes the game or wins first, the opponent remains unaware until they take their turn and receive a message indicating that their opponent has completed the game first.

CONCLUSION

In the classic strategy game Battleship, two players alternately hide their ships on a grid and blast out coordinates to target and destroy the other player's fleet. The system effectively monitors everything from player profiles and game states to leaderboards and matchmaking, thanks to unique IDs and secure passwords for administrators and players. Every game offers a variety of modes, such as ranked matches, casual play, and tournaments, to accommodate all skill levels and captures player actions, shots, and victories in real time. Our smart matching system matches players with comparable skill levels to provide equitable and engaging matches. We monitor detailed information on victories, losses, accuracy, and length of game to assist participants in improving their tactics. We prioritize security and safeguard data and transactions with the best encryption and authentication available. Numerous players can be accommodated at once by our scalable database, which maintains excellent performance and dependability for a seamless gaming experience.