

# CSC667-867 Spring 2019

## I. Title Page

**Team Number:** 08

**List of Members:** Jianfei Zhao

Alan Ng

Philip Yu

Hang Li

Eric Chen

**Milestone:** 3

**Due Date:** 04/05/2019

**Link:** <https://github.com/csc667-02-sp19/csc667-sp19-Team08> (master branch of repo)

## II. List of Operations for Each Entity

### i). User

- **CREATE** operation:

The creation process of each user is known as registration. Once created, the user (player) will never be deleted. Thus, there will not be DELETE operation for **User** entity. This operation requires username and password as parameters, where the newly created user will be assigned with a unique id (primary key).

- **READ** operation:

The retrieval process of each user is known as login. To play uno with others, the user has to either log in his/her account or register as a new user.

- **UPDATE** operation:

After a uno game has ended, the statistics (WIN/DRAW/LOSE) for each participating user should be updated. A unique user id is needed to execute this UPDATE operation, which can be retrieved from the game status.

### ii). Game

- **CREATE** operation:

Any registered user may create a new game room, and the user who creates this room is the host. The creation process will assign the game room with a unique id (primary key). When a room has more than 2 players, the host may choose to either launch the game or wait for more users.

- **READ** operation:

When needed, the game should be restored to where it left off. This READ operation will retrieve the saved game status, by using the given game id as the parameter.

- **UPDATE** operation:

When a participating player loses connection or leaves before the game ends, the game status should be saved. All the players as well as remaining cards have to be updated, where all the information will be encoded as json string.

- **DELETE** operation:

After a player wins the current game, the host may choose to terminate this game room. Once deleted, the game room will be permanently removed from the database and never be retrieved. A unique game id should be passed as the parameter to delete a uno game.

### iii). Card

- **CREATE** operation:

This one-time creation will be done during the implementation of this internet application. Once created, the cards will never be changed anymore, since the uno cards

are predefined. In other words, there will not be UPDATE or DELETE operation for **Card** entity.

- **READ** operation:

Each card will be assigned with a unique id (primary key). This unique id will be passed as the parameter to retrieve card information, including number (0~9: normal cards; >9: functional cards) and color (0: functional cards; 1~4: blue, red, green, yellow).

#### iv). **Record**

- **CREATE** operation:

When a player has no remaining cards in his/her hand, the uno game is considered to be over. At this time, the game results for all participating players should be saved to the **Record** entity through this CREATE operation. Once created, the records will never be changed anymore, since the game has ended. In other words, there will not be UPDATE or DELETE operation for **Record** entity.

- **READ** operation:

In user dashboard, each user can retrieve his/her playing records through this READ operation. A user id will be passed as the parameter, where all the records related to this player will be retrieved as a result set.

### III. List of Route Paths

Some general rules about route paths are listed as follows.

- The “**dir**” ahead of each API refers to the root directory of our team project on server.
- Each URL is completely in lower-case format, without any white spaces.
- The name of entity (e.g., user, game, card, etc.) comes right after the root directory, notifying the database model that will be used in this route path.
- The name of operation (e.g., create, read, update, delete) comes right after the entity, which specifies what type of operation will be done.
- The required parameters (e.g., primary id key) comes right after the operation.

Below are some basic route paths. More APIs might be added later according to the implementation requirements.

- **dir/user/create:** create a user (registration)

Parameter: {

```
        "email": email address,
        "username": username,
        "password": password
    }
    Return: {
        "code": success or not,
        "user_id": primary key of user's account,
        "username": username
    }
```

- **dir/user/read:** read the info of a user (login)

```
Parameter: {
    "username": username,
    "password": password
}
Return: {
    "user_id": primary key of user's account,
    "username": username,
    "win": number of WINS,
    "draw": number of DRAWS,
    "lose": number of LOSES
}
```

- **dir/user/update:** update the info of a user

```
Parameter: {
    "user_id": primary key,
    "win": number of WINS,
    "draw": number of DRAWS,
    "lose": number of LOSES
}
Return: {
    "code": success or not
}
```

- **dir/game/create:** create a new game

```
Parameter: {
    "host_id": user id of the host player,
    "status": game status,
    "unplayed": unplayed cards in pile, encoded as json string,
```

```
        "remaining": remaining cards for each user, encoded as json string
    }
    Return: {
        "code": success or not
    }
```

- **dir/game/read:** read the info of a game

```
Parameter: {
    "game_id": primary key
}
Return: {
    "host_id": user id of the host of this room,
    "status": game status,
    "unplayed": unplayed cards in pile, encoded as json string,
    "remaining": remaining cards for each user, encoded as json string
}
```

- **dir/game/update:** update the info of a game

```
Parameter: {
    "game_id": primary key,
    "status": game status,
    "unplayed": unplayed cards in pile, encoded as json string,
    "remaining": remaining cards for each user, encoded as json string
}
Return: {
    "code": success or not
}
```

- **dir/game/delete:** delete a game

```
Parameter: {
    "game_id": primary key
}
Return: {
    "code": success or not
}
```

- **dir/card/create:** create a new card

```
Parameter: {
    "color": 1~4 for normal colors (blue, red, green, yellow); 0 for functional cards,
```

“number”: 0~9 for normal cards; >9 for functional cards

}

Return: none (one-time creation process)

- **dir/card/read:** read the info of a card

Parameter: {

“card\_id”: primary key

}

Return: {

“color”: 1~4 for normal colors (blue, red, green, yellow); 0 for functional cards,

“number”: 0~9 for normal cards; >9 for functional cards

}

- **dir/record/create:** create a new record

Parameter: {

“game\_id”: game id of the current game,

“user\_id”: user id of the participating player,

“result”: 0 for draw; 1 for win; -1 for lose

}

Return: {

“code”: success or not

}

- **dir/record/read:** read the info of a record

Parameter: {

“record\_id”: primary key,

}

Return: {

“game\_id”: game id of the current game,

“user\_id”: user id of the participating player,

“result”: 0 for draw; 1 for win; -1 for lose

}

## IV. Basic Structure of HTML

Below are the front-end web pages for our team project. The interaction with back-end will be implemented later.

### i). Lobby Page (Home Page)

UNO

Option 1Option 2Option 3

Game ID	Host User	Game Status	Game Time
112	Timmy	Pending	03/22/2019
113	Zachary	In-Game	03/23/2019
114	Lilian	In-Game	03/23/2019
115	Jason	In-Game	03/24/2019
116	William	Pending	03/24/2019

Welcome to Socket.IO Chat –

jianfei

 hi, this is jianfei

mike

 hello, i'm mike

there are 3 participants

christian

 nice to meet you.

your last message: 2019-04-05 19:29:00

Type here...

## ii). Game Page (Game Room)

UNO			Option 1	Option 2	Option 3
ID	Player	Remaining Cards	Current Uno Card Will be Here		
18	Mark	7			
207	Jacob	4			
56	Larry	3			
132	Tobias	6			

there's 1 participant  
Jacob joined  
there are 2 participants

**Jacob** my turn  
**Tobias** 6 cards left ... bad luck :-(  
your last message: 2019-04-05 19:25:24

i'm typing now ...|

## iii). Login Page

UNO		Option 1	Option 2	Option 3
<h1>Login</h1>				
Username				
<input type="text" value="Enter Username"/>				
Password				
<input type="text" value="Enter Password"/>				
<input type="checkbox"/> I'm not a robot				
 reCAPTCHA <a href="#">Privacy</a> - <a href="#">Terms</a>				
<input type="button" value="Log in"/>				



## iv). Registration Page

UNO

Option 1Option 2Option 3

# Registration

Required fields are marked with \*

Email

Enter Email

Username\*

Enter Username


Password\*

Enter Password

Confirm Password\*

Confirm Placeholder

☐ I'm not a robot

  
reCAPTCHA  
[Privacy](#) - [Terms](#)

☐ Agree to [Terms of Services\\*](#)

Register


Already have an account? [Log in.](#)

## v). Profile Page (User Dashboard)

UNO

Option 1Option 2Option 3

Profile Details



catman123  
Rank #123

catman123 Statistics

Won: 162

Lost: 123

Draw: 5

[View more details...](#)

Miscellaneous

## vi). Record Page (Playing Results)

UNO

Option 1Option 2Option 3

Game Records

Game ID	Host User	Game Result	Game Time
112	Timmy	Win	03/22/2019
113	Zachary	Draw	03/23/2019
114	Lilian	Win	03/23/2019
115	Jason	Lose	03/24/2019
116	William	Win	03/24/2019
117	Hannah	Win	03/24/2019
118	Paul	Lose	03/25/2019
119	Tobias	Draw	03/25/2019

Miscellaneous