PROJECT REPORT

SOEN 6441- ADVANCED PROGRAMMING PRACTICES

Report

On

NBA Teams Website

Instructor:

Dr. C. Constantinides, P.Eng.

Department of Computer Science and Software Engineering
Concordia University
Montreal, Quebec
CANADA

By

Student Name	Student ID
Vanshika Singla	40201070
Krishna Vamsi Rokkam	40237902

CONTENTS

1. CODING STANDARDS

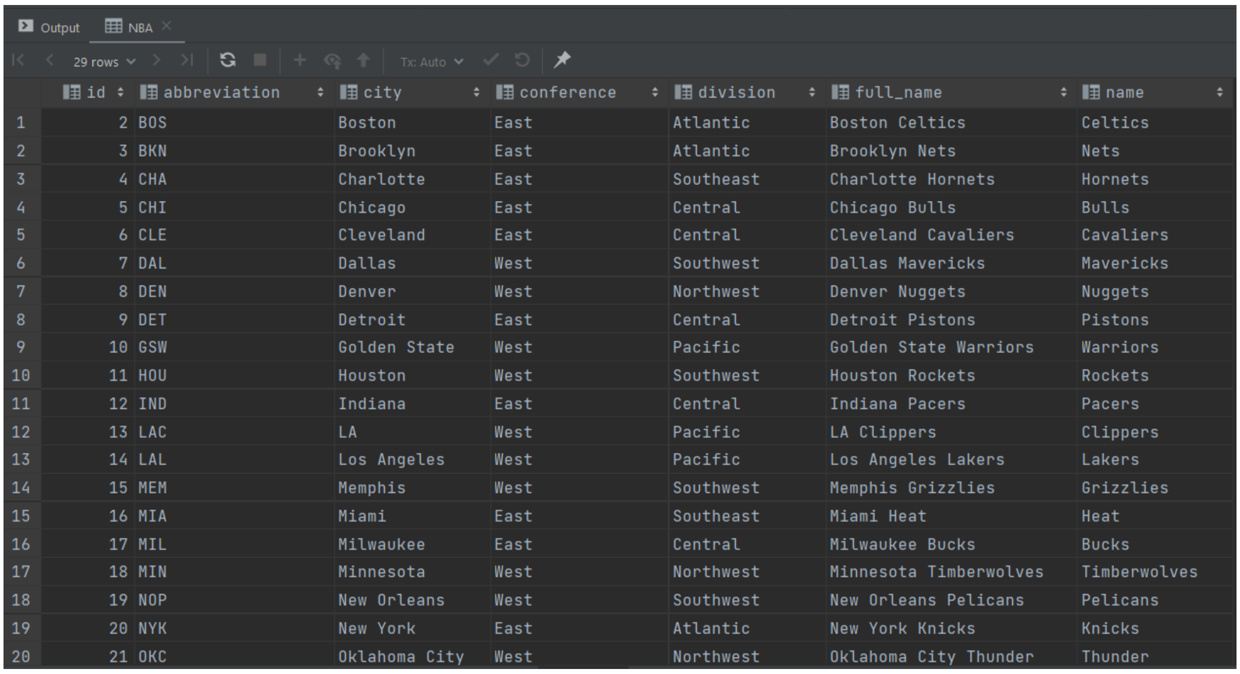
We have used JavaScript with node.js and express, only to fetch the data from the API and used python pandas library to get the JSON object from the API and to convert the data from json to an Sqlite database. The API that we have used is from https://rapidapi.com/hub

For the frontend, we have used HTML/CSS. We have made 4 buttons which perform the following operations: ViewTeam Details, RegisterTeam, Update Team Details & Unregister the team. To perform the above mentioned operations we have made a class called NBA (class file name - nba.js) which has the following methods:

modifyRow(), modifyRow1(), deleteRow(), getRow(), addRow(), addRow1()

These methods take an SQL query to return the data with respect to the operations. We created API calls in app.js that calls methods of class NBA in NBA.js and executes the SQL queries defined under those methods.

Database Screenshot:



	id	abbreviation	city	conference	division	full_name	name
1	2	BOS	Boston	East	Atlantic	Boston Celtics	Celtics
2	3	BKN	Brooklyn	East	Atlantic	Brooklyn Nets	Nets
3	4	CHA	Charlotte	East	Southeast	Charlotte Hornets	Hornets
4	5	CHI	Chicago	East	Central	Chicago Bulls	Bulls
5	6	CLE	Cleveland	East	Central	Cleveland Cavaliers	Cavaliers
6	7	DAL	Dallas	West	Southwest	Dallas Mavericks	Mavericks
7	8	DEN	Denver	West	Northwest	Denver Nuggets	Nuggets
8	9	DET	Detroit	East	Central	Detroit Pistons	Pistons
9	10	GSW	Golden State	West	Pacific	Golden State Warriors	Warriors
10	11	HOU	Houston	West	Southwest	Houston Rockets	Rockets
11	12	IND	Indiana	East	Central	Indiana Pacers	Pacers
12	13	LAC	LA	West	Pacific	LA Clippers	Clippers
13	14	LAL	Los Angeles	West	Pacific	Los Angeles Lakers	Lakers
14	15	MEM	Memphis	West	Southwest	Memphis Grizzlies	Grizzlies
15	16	MIA	Miami	East	Southeast	Miami Heat	Heat
16	17	MIL	Milwaukee	East	Central	Milwaukee Bucks	Bucks
17	18	MIN	Minnesota	West	Northwest	Minnesota Timberwolves	Timberwolves
18	19	NOP	New Orleans	West	Southwest	New Orleans Pelicans	Pelicans
19	20	NYK	New York	East	Atlantic	New York Knicks	Knicks
20	21	OKC	Oklahoma City	West	Northwest	Oklahoma City Thunder	Thunder

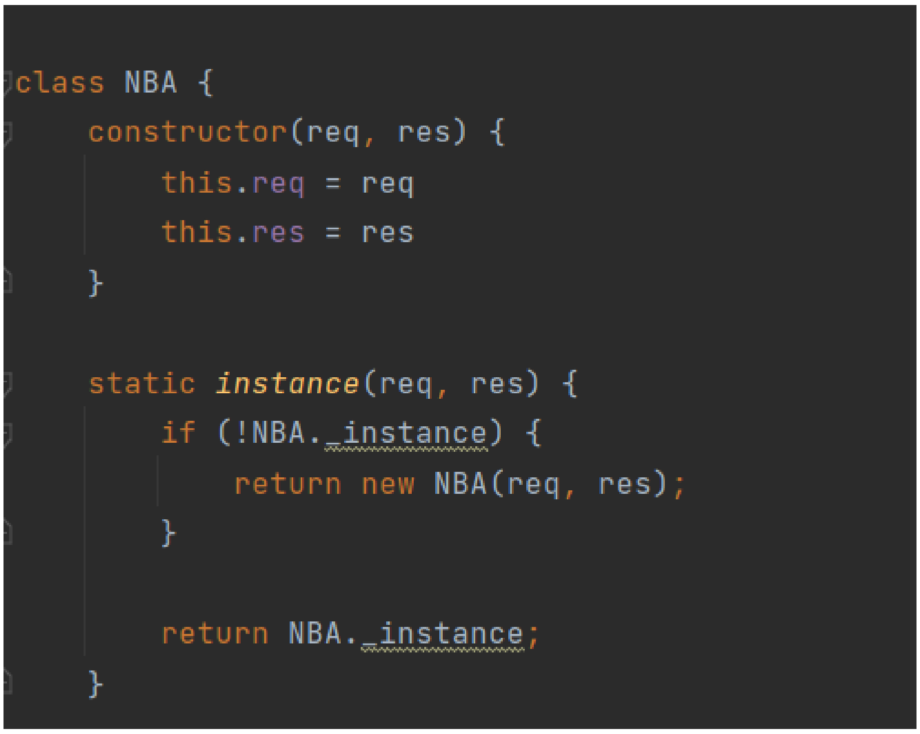
2. APPLICABLE PATTERNS

We have implemented Singleton design pattern in our system. Singleton design pattern tells us that only one instance of the class can be created which can be accessed globally. That single instance is called singleton.

It makes sure that the class acts as a single source for all the data that the users want to access.

Below is the screenshot of the code:

Class NBA which has a static method in which it check if an instance is created or not; if not created it'll create one otherwise return the previously created one.



```
class NBA {
    constructor(req, res) {
        this.req = req
        this.res = res
    }

    static instance(req, res) {
        if (!NBA._instance) {
            return new NBA(req, res);
        }

        return NBA._instance;
    }
```

3. REFACTORING STRATEGIES

We have used the following refactoring strategies in our code:

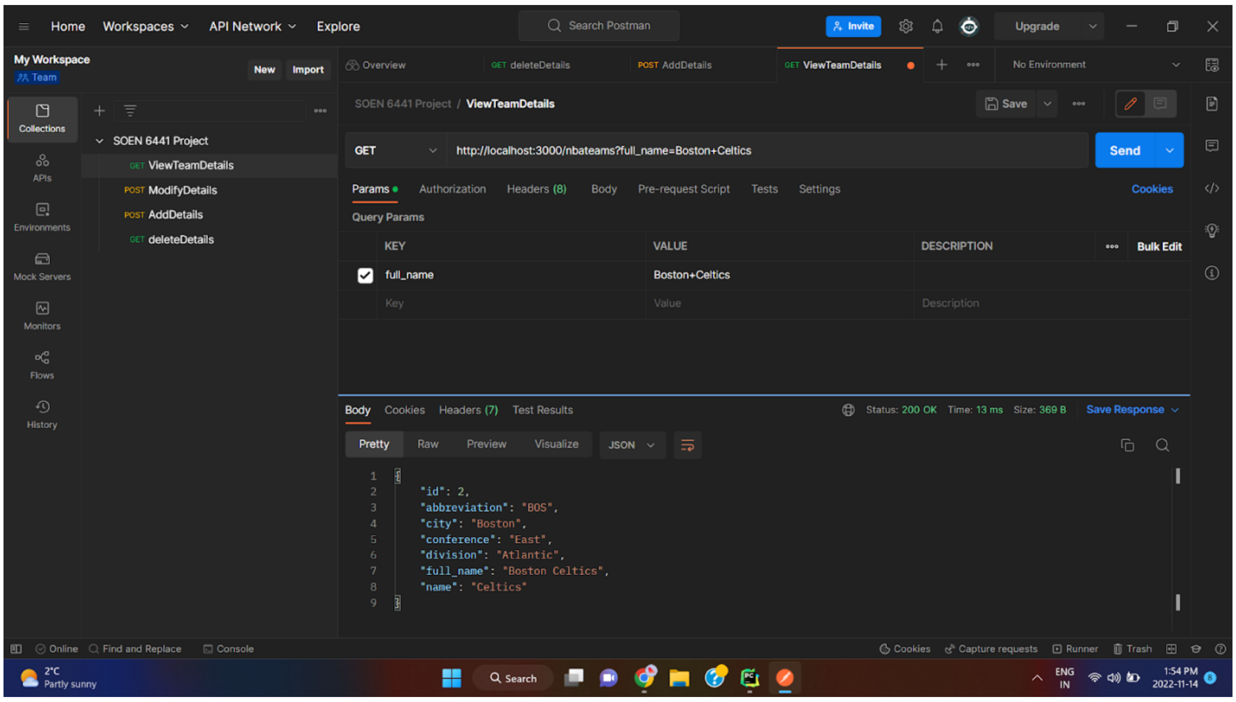
1. Red-Green Refactoring: In the Agile software development process, Red-Green code refactoring is the most well-liked and frequently employed technique. This method to design and execution, known as "test-first," lays the groundwork for all types of refactoring. For our project, we wrote the code to add the functionality (add, delete, update) first and then refactored the code according to the testing states and then simplified it by enhancing our code.

2. Abstraction: We implemented singleton design pattern so that the users can not access the database directly but rather they can only get the data through single instance of the class NBA. We simplified and extracted the code by implementing separate methods in the class and making API calls outside the class in a separate JavaScript file, thereby, providing abstraction feature to our code.

4. TESTING TOOL

We are using postman for testing. Below are the screenshots:

viewTeamDetails:



SOEN 6441 Project / ViewTeamDetails

GET http://localhost:3000/nbateams?full_name=Boston+Celtics

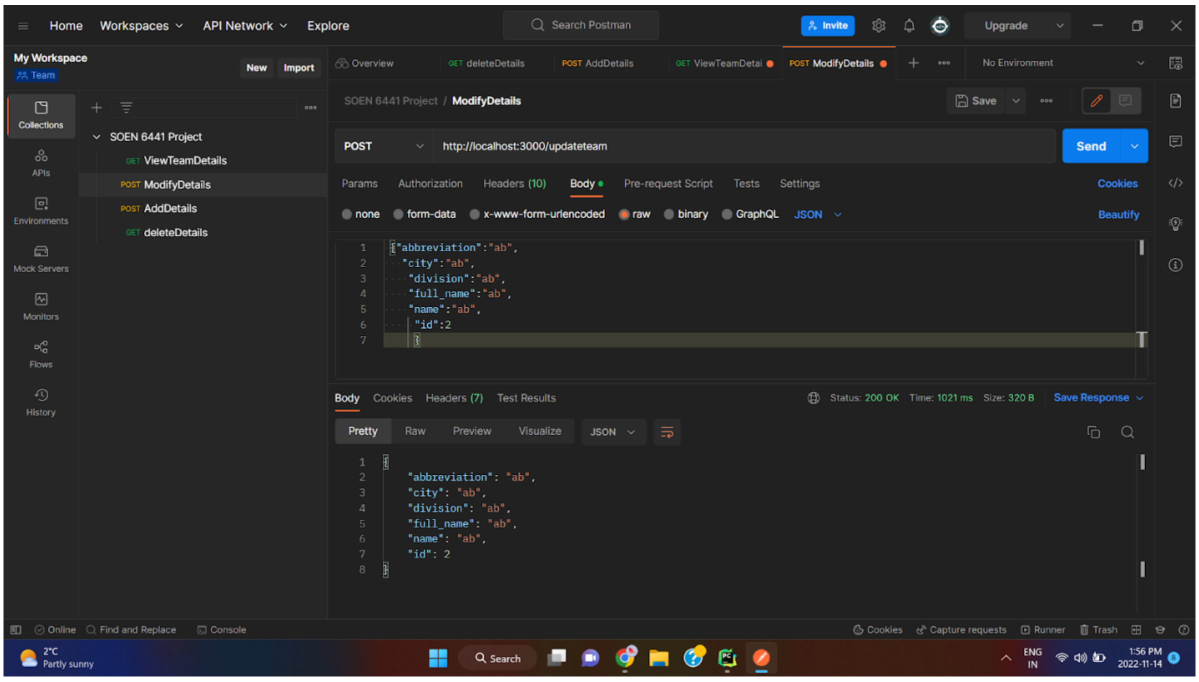
KEY	VALUE	DESCRIPTION
full_name	Boston+Celtics	

Status: 200 OK Time: 13 ms Size: 369 B

```
{
    "id": 2,
    "abbreviation": "BOS",
    "city": "Boston",
    "conference": "East",
    "division": "Atlantic",
    "full_name": "Boston Celtics",
    "name": "Celtics"
}
```

2°C Partly sunny   ENG IN   1:54 PM 2022-11-14

modifyDetails:



SOEN 6441 Project / ModifyDetails

POST http://localhost:3000/updateteam

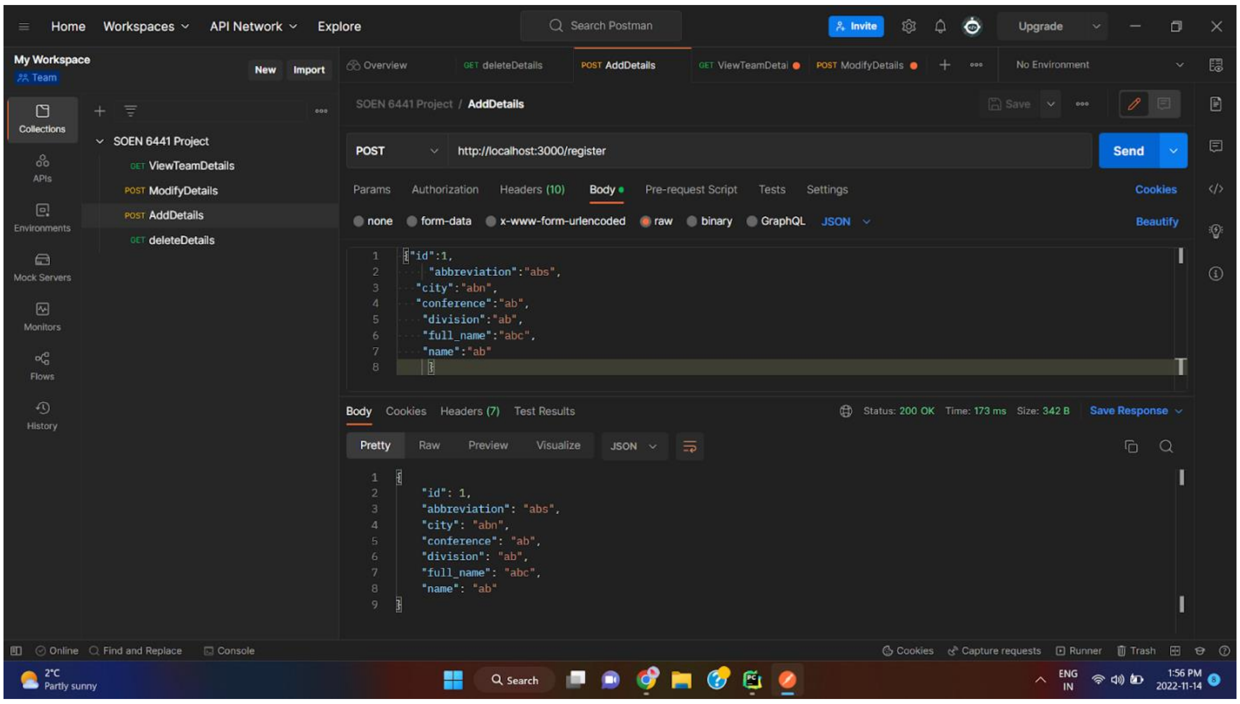
```
{"abbreviation":"ab",
  "city":"ab",
    "division":"ab",
    "full_name":"ab",
    "name":"ab",
    "id":2
}
```

Status: 200 OK Time: 1021 ms Size: 320 B

```
{
    "abbreviation": "ab",
    "city": "ab",
    "division": "ab",
    "full_name": "ab",
    "name": "ab",
    "id": 2
}
```

2°C Partly sunny   ENG IN   1:56 PM 2022-11-14

addDetails:



SOEN 6441 Project / AddDetails

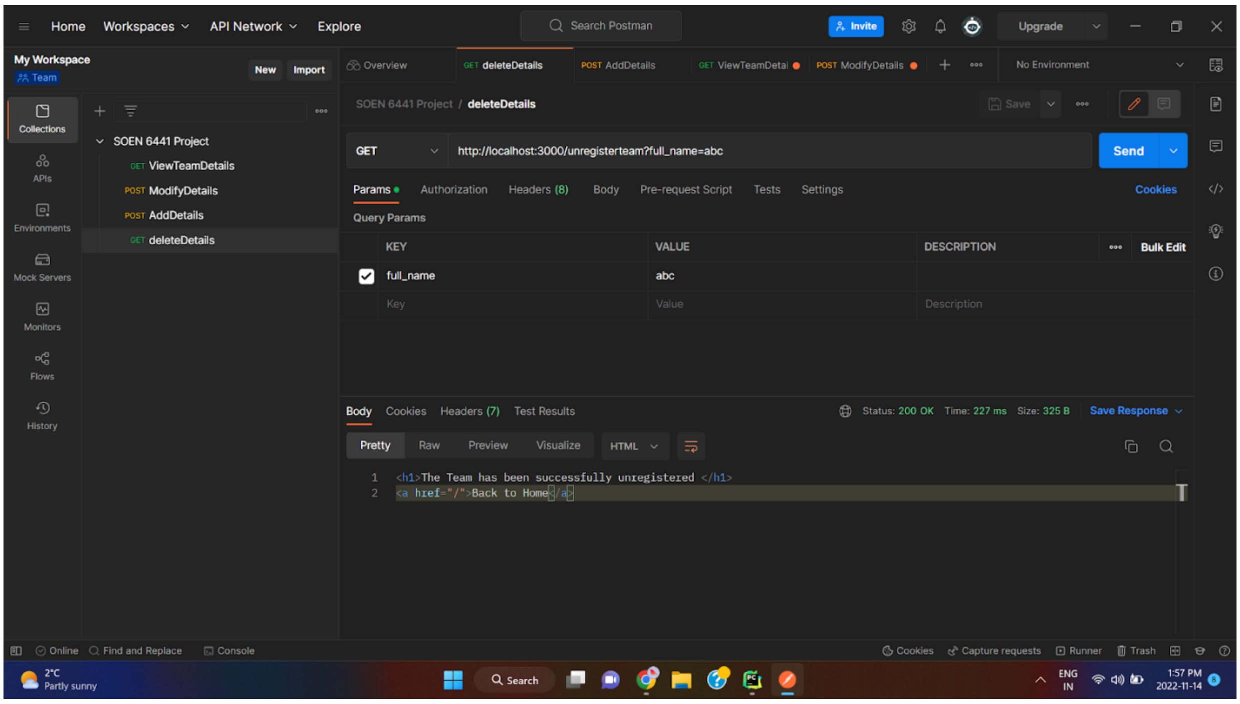
POST http://localhost:3000/register

```
{"id":1,
    "abbreviation":"abs",
  "city":"abn",
  "conference":"ab",
   "division":"ab",
   "full_name":"abc",
   "name":"ab"
  }
```

Status: 200 OK Time: 173 ms Size: 342 B

```
{
    "id": 1,
    "abbreviation": "abs",
    "city": "abn",
    "conference": "ab",
    "division": "ab",
    "full_name": "abc",
    "name": "ab"
}
```

deleteDetails:



SOEN 6441 Project / deleteDetails

GET http://localhost:3000/unregisterteam?full_name=abc

Query Params

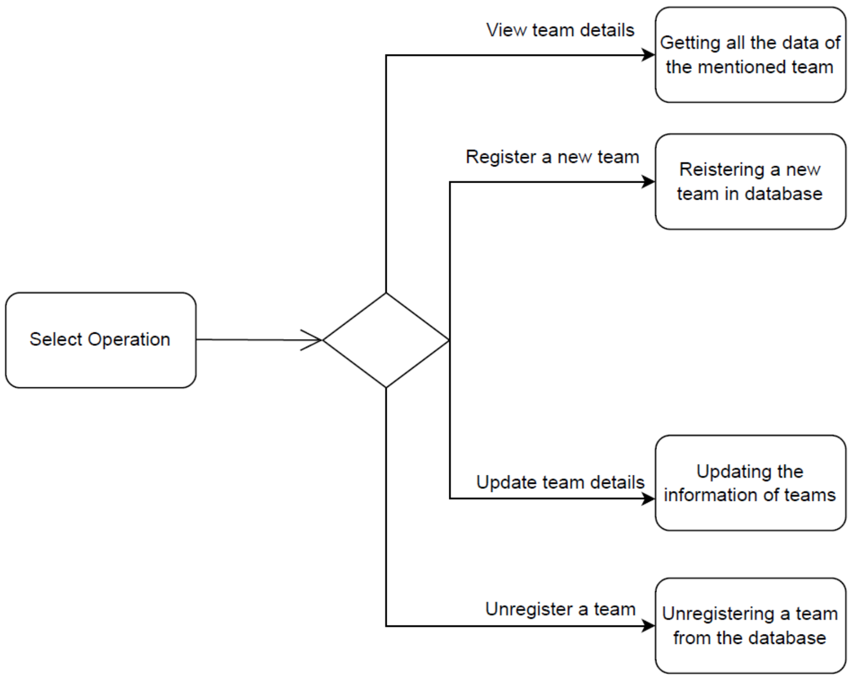
KEY	VALUE	DESCRIPTION
full_name	abc	

Status: 200 OK Time: 227 ms Size: 325 B

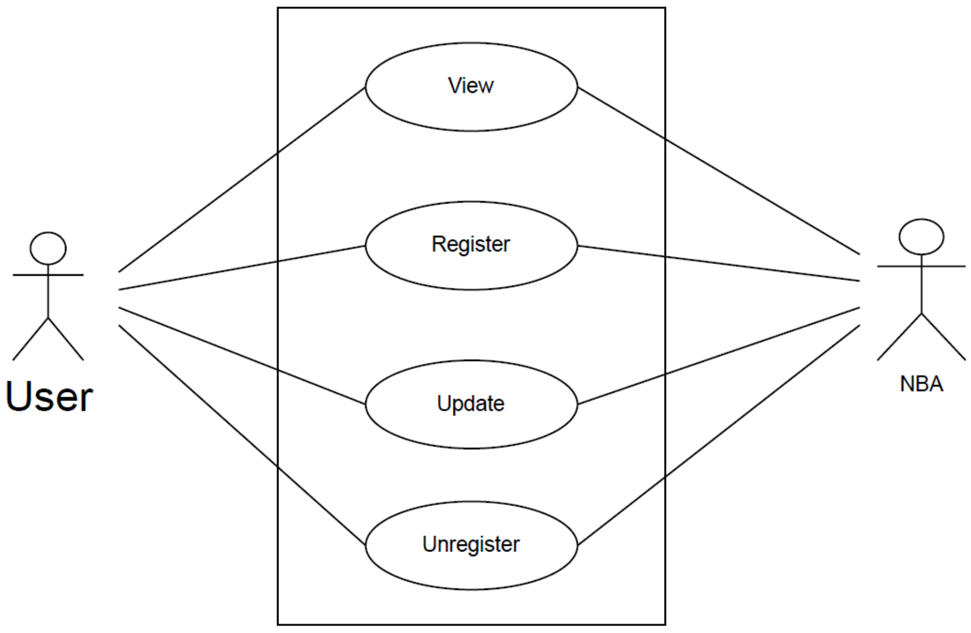
```
<h1>The Team has been successfully unregistered </h1>
<a href="/">Back to Home</a>
```

5. SOFTWARE ARCHITECTURE DOCUMENT

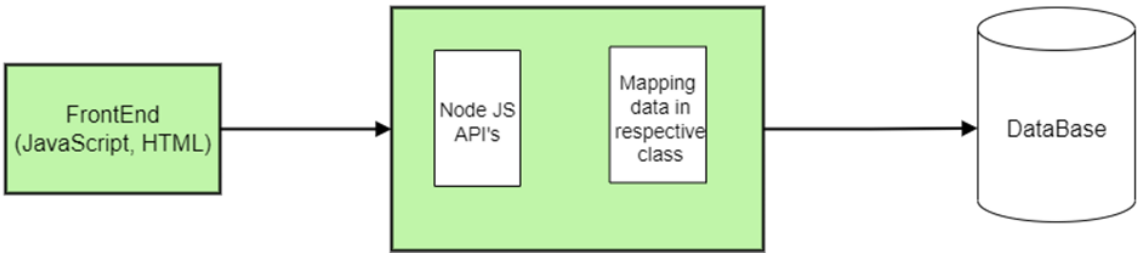
ACTIVITY DIAGRAM


View team details
Getting all the data of the mentioned team
Register a new team
Reistering a new team in database
Select Operation
Update team details
Updating the information of teams
Unregister a team
Unregistering a team from the database

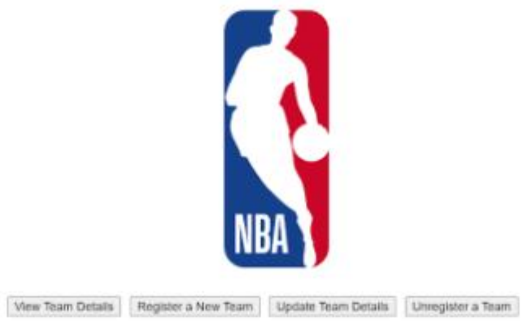
USE CASE DIAGRAM



Architectural View:



The frontend Interface:



REFERENCES

1. For API: https://rapidapi.com/hub
2. For testing : https://www.postman.com/
3. For Drawing UML Diagrams: https://app.diagrams.net/

   https://www.lucidchart.com/pages/