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

Github Repository Link:
https://github.com/Krisshvamsi/SOEN6441_Project

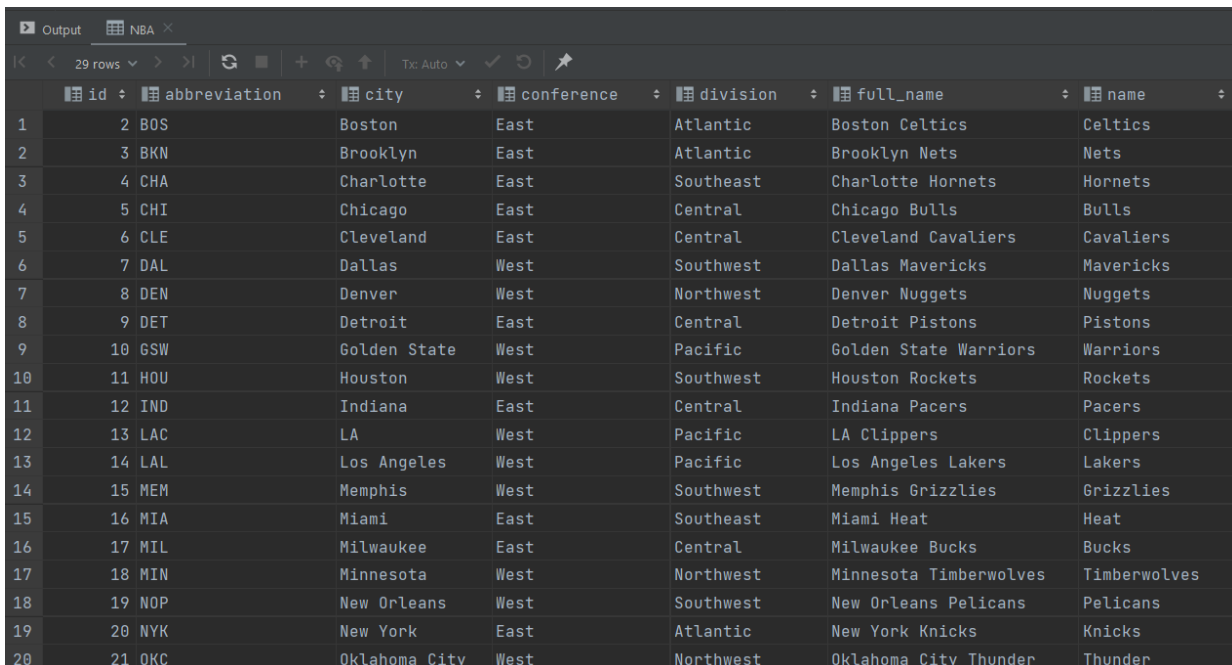# CONTENTS

1. <u>CODING STANDARDS</u>

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

2. <u>APPLICABLE PATTERNS</u>

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.

<u>Below is the screenshot of the code:</u>

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. <u>REFACTORING STRATEGIES</u>

We have used the following refactoring strategies in our code:

1. <u>Red-Green Refactoring</u>: 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. <u>Abstraction</u>: 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.
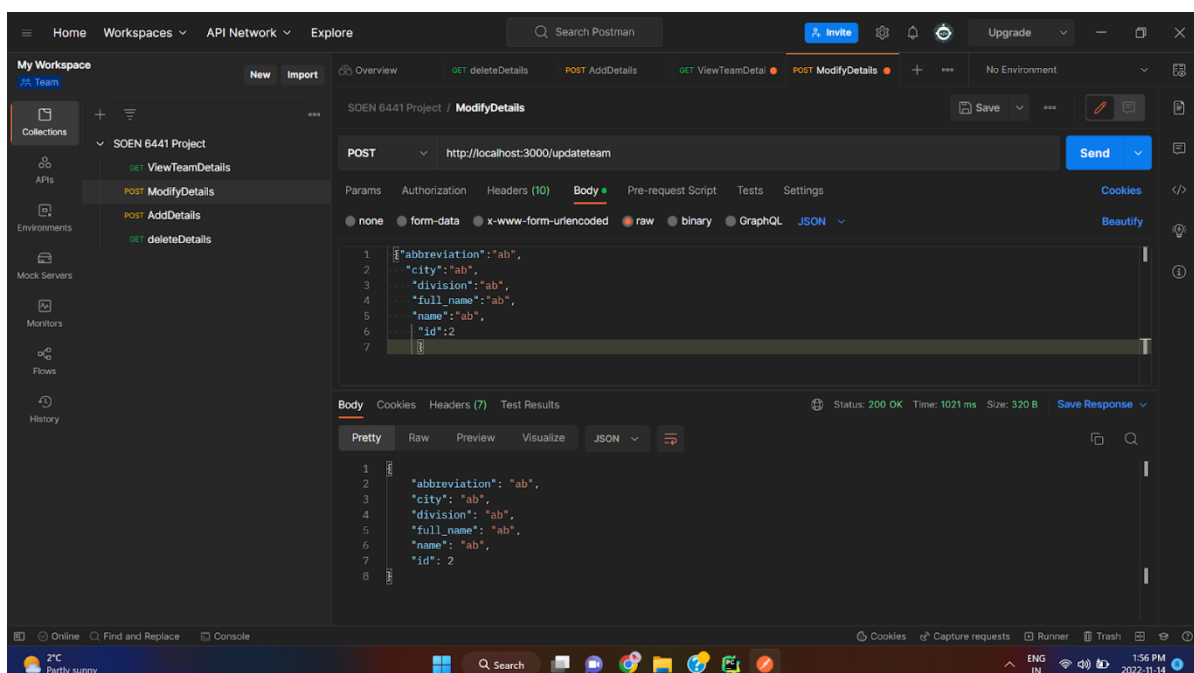
4. <u>TESTING TOOL</u>

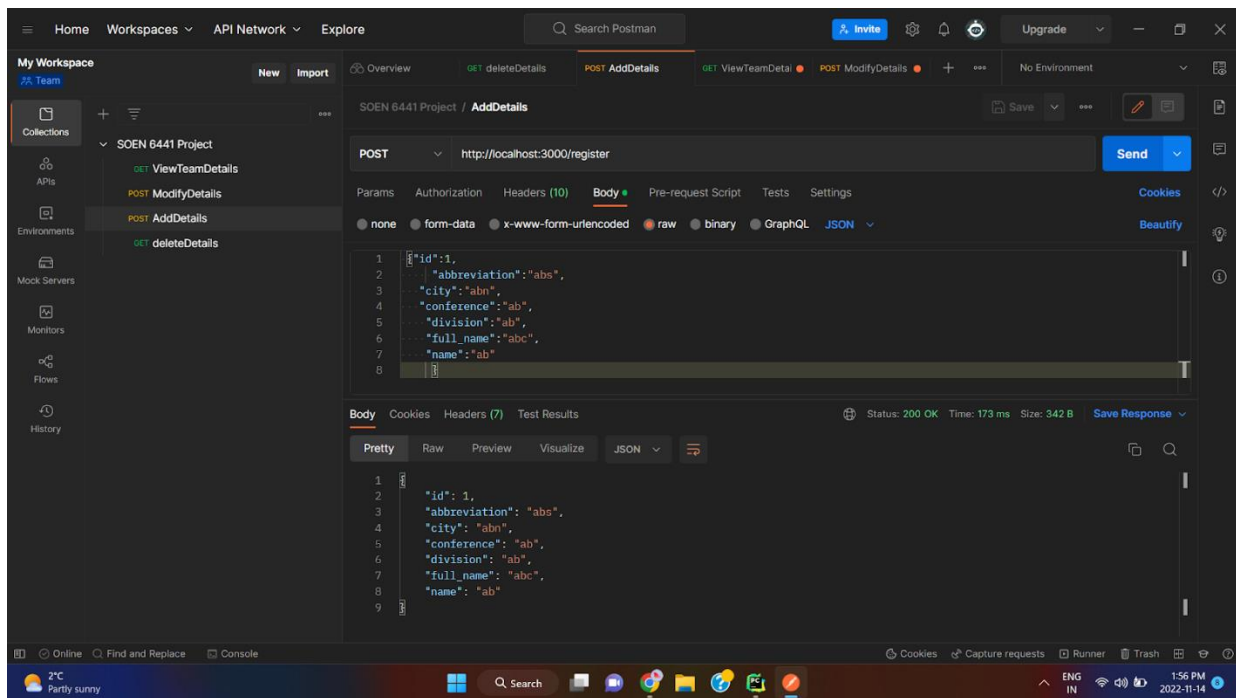We are using postman for testing. Below are the screenshots:
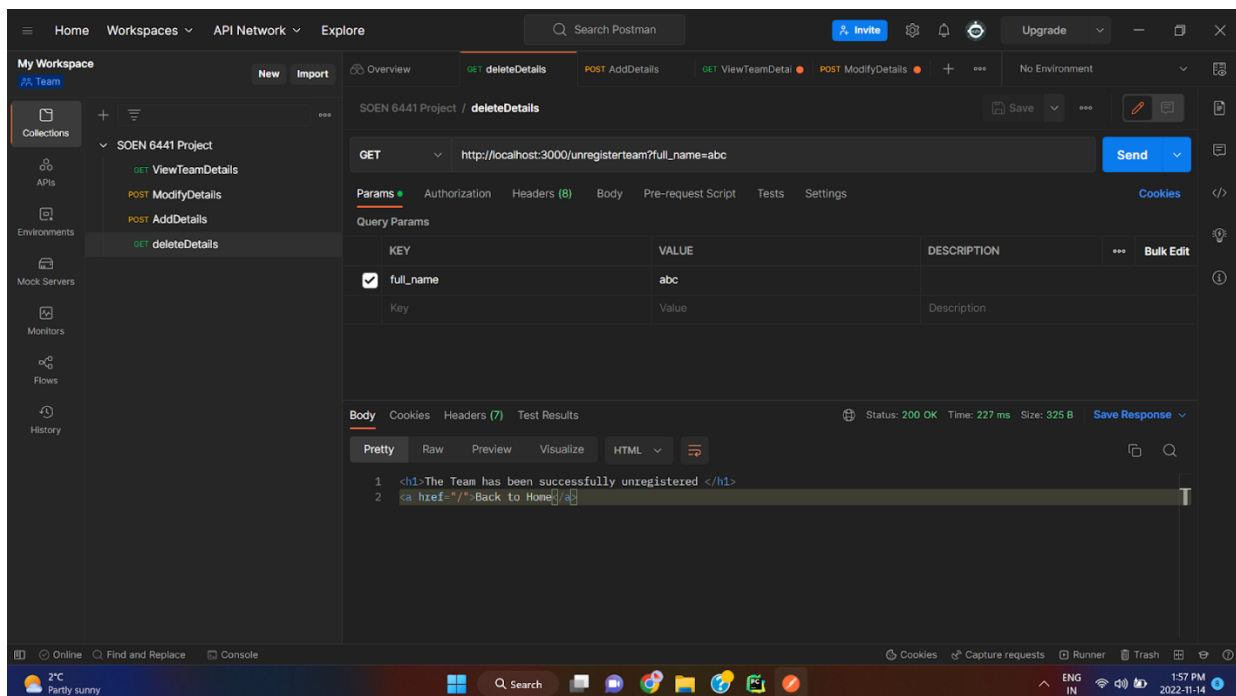
viewTeamDetails:
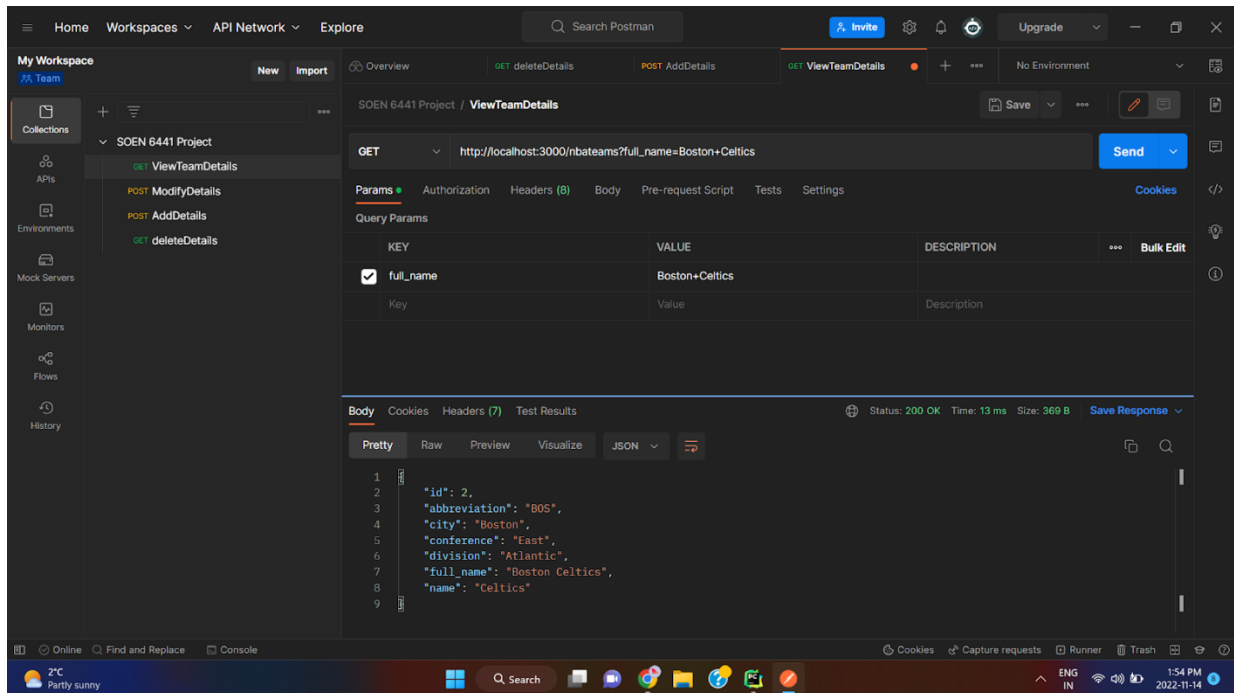
modifyDetails:
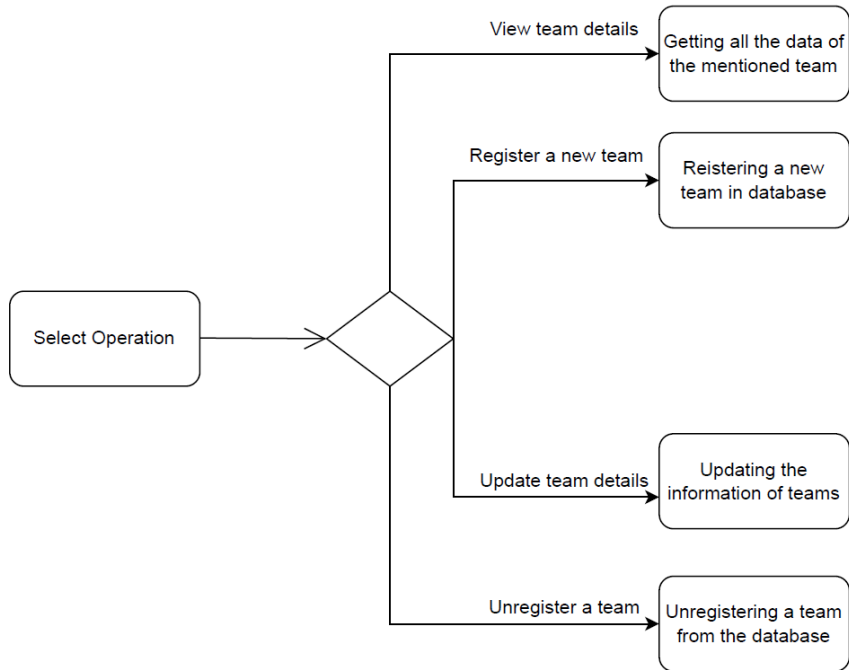
addDetails:

deleteDetails:
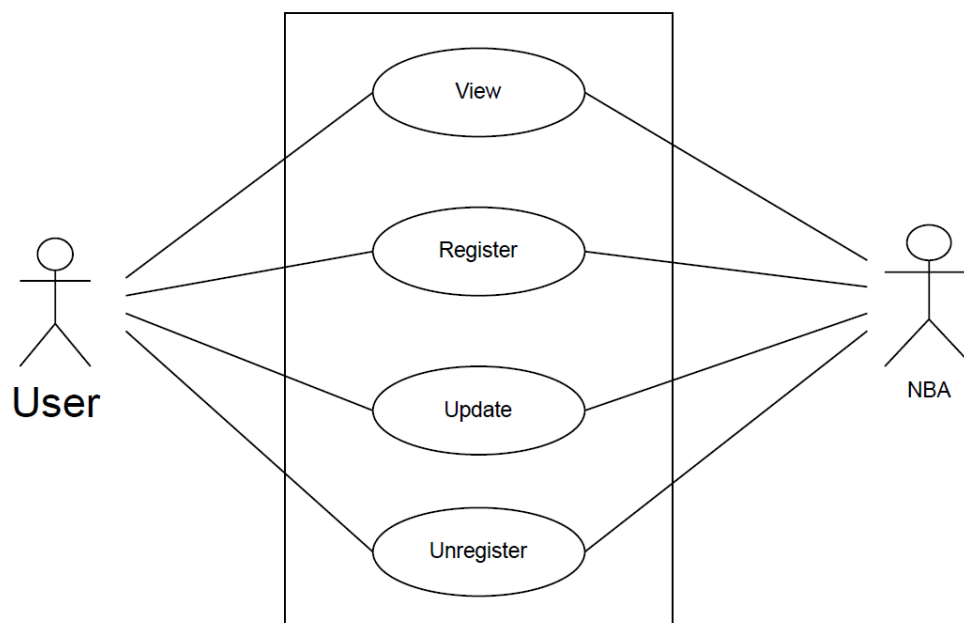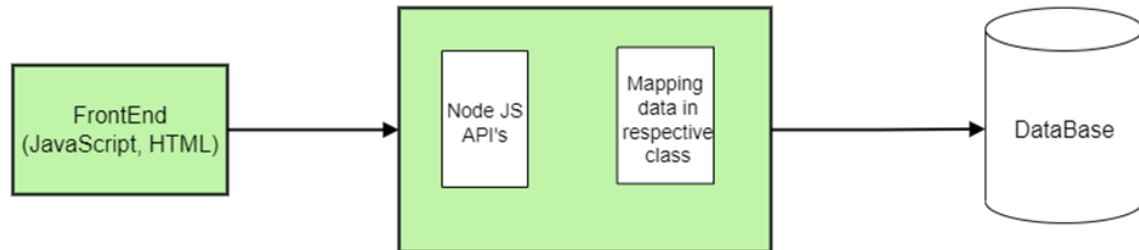
5. <u>SOFTWARE ARCHITECTURE DOCUMENT</u>

ACTIVITY DIAGRAM

USE CASE DIAGRAM

Architectural View:

The frontend Interface:

<u>REFERENCES</u>

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/