Candidate name: ABILASH D

Register number: 20BDS0408

Statement for Candidates:

Below is the problem statement for your project.

1. Your task is to create an API that supports the requirements mentioned in Problem Statement

2. You have 3 hours to develop the API.

3. You can use Google to find solutions, however, copying an existing project is not allowed.

4. At the end of the 3 hours, you will be given get 15 minutes to make your submission on Google Forms & to push your project into a

GitHub repository

5. Submission form: API Round Submissions 2023

6. Our team will evaluate your API submission and the result will be published at 1 PM on the same day.

7. For any other queries feel free to join this meet link : Meet

Problem Statement

Hey there, Mr. X. You have been appointed to design a platform like Cricbuzz, wherein guest users can come on the platform and browse

across multiple matches and can see either of them in detail.

There is a Role Based Access provision and 2 types of users would exist :

1. Admin - can perform all operations like adding matches, players in the teams, updating stats and scores, etc.

2. Guest - can only view matches and their details.

Tech Stack:

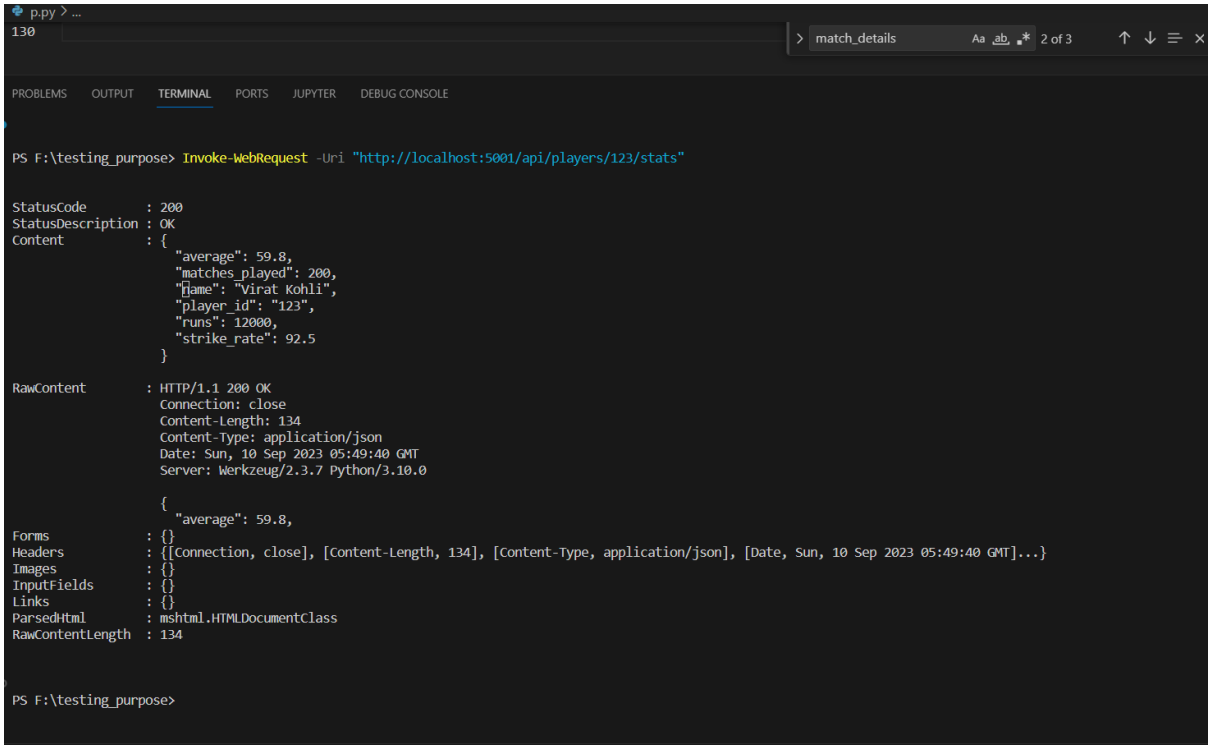
1. Any web server of your choice (Python Flask / Django, NodeJS Express / Koa, Java, etc)

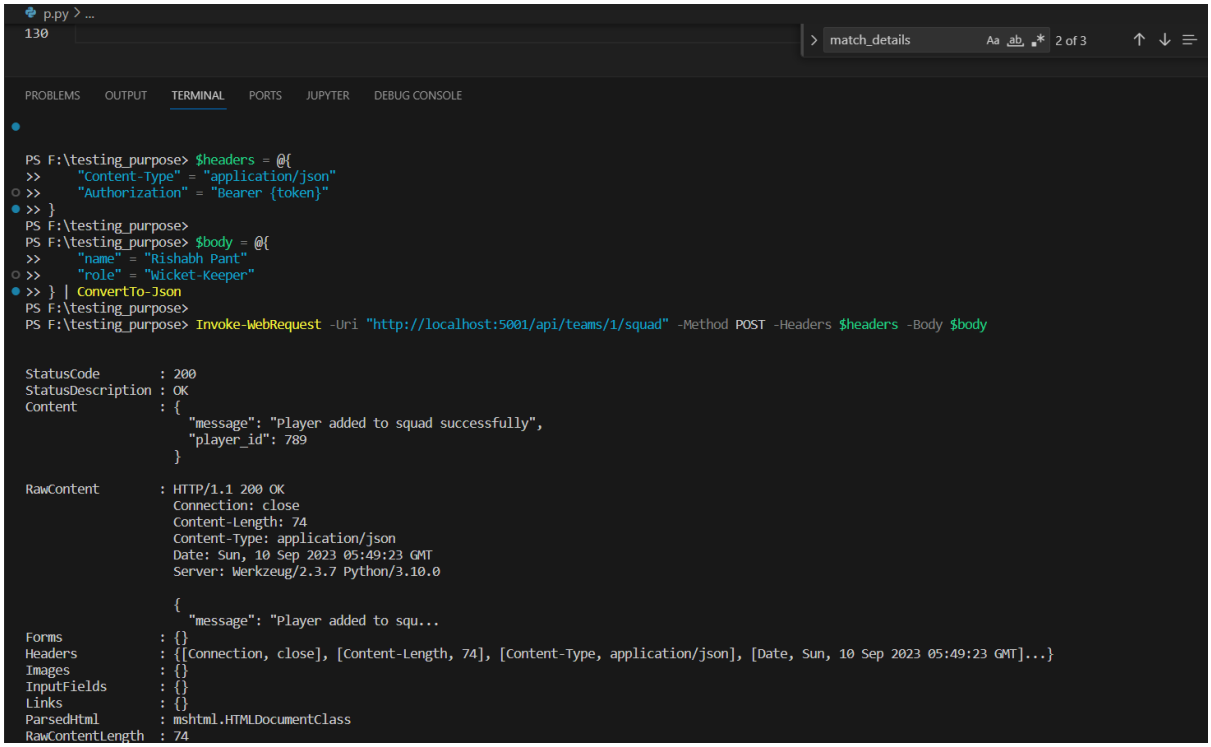
2. Database: MySQL/PostgreSQL (Compulsory)

Framwork used: flask, apache mysql (xamp)

Screenshots:

Testing using curl :


p.py > ...
130
match_details
2 of 3
PROBLEMS
OUTPUT
TERMINAL
PORTS
JUPYTER
DEBUG CONSOLE
PS F:\testing_purpose> Invoke-WebRequest -Uri "http://localhost:5001/api/players/123/stats"
StatusCode : 200
StatusDescription : OK
Content : {
"average": 59.8,
"matches_played": 200,
"name": "Virat Kohli",
"player_id": "123",
"runs": 12000,
"strike_rate": 92.5
}
RawContent : HTTP/1.1 200 OK
Connection: close
Content-Length: 134
Content-Type: application/json
Date: Sun, 10 Sep 2023 05:49:40 GMT
Server: Werkzeug/2.3.7 Python/3.10.0
{
"average": 59.8,
Forms : {}
Headers : {[Connection, close], [Content-Length, 134], [Content-Type, application/json], [Date, Sun, 10 Sep 2023 05:49:40 GMT]...}
Images : {}
InputFields : {}
Links : {}
ParsedHtml : mshtml.HTMLDocumentClass
RawContentLength : 134
PS F:\testing_purpose>


p.py > ...
130
match_details
2 of 3
PROBLEMS
OUTPUT
TERMINAL
PORTS
JUPYTER
DEBUG CONSOLE
PS F:\testing_purpose> $headers = @{
>> "Content-Type" = "application/json"
>> "Authorization" = "Bearer {token}"
>> }
PS F:\testing_purpose>
PS F:\testing_purpose> $body = @{
>> "name" = "Rishabh Pant"
>> "role" = "Wicket-Keeper"
>> } | ConvertTo-Json
PS F:\testing_purpose>
PS F:\testing_purpose> Invoke-WebRequest -Uri "http://localhost:5001/api/teams/1/squad" -Method POST -Headers $headers -Body $body
StatusCode : 200
StatusDescription : OK
Content : {
"message": "Player added to squad successfully",
"player_id": 789
}
RawContent : HTTP/1.1 200 OK
Connection: close
Content-Length: 74
Content-Type: application/json
Date: Sun, 10 Sep 2023 05:49:23 GMT
Server: Werkzeug/2.3.7 Python/3.10.0
{
"message": "Player added to squ...
Forms : {}
Headers : {[Connection, close], [Content-Length, 74], [Content-Type, application/json], [Date, Sun, 10 Sep 2023 05:49:23 GMT]...}
Images : {}
InputFields : {}
Links : {}
ParsedHtml : mshtml.HTMLDocumentClass
RawContentLength : 74

```
PROBLEMS  OUTPUT  TERMINAL  PORTS  JUPYTER  DEBUG CONSOLE

PS F:\testing_purpose> Invoke-WebRequest -Uri "http://localhost:5001/api/matches"
>>


StatusCode        : 200
StatusDescription : OK
Content           : {
                      "matches": [
                        [
                          "",
                          "India",
                          "Australia",
                          "Wed, 12 Jul 2023 00:00:00 GMT",
                          "Sydney Cricket Ground"
                        ]
                      ]
                    }

RawContent        : HTTP/1.1 200 OK
                    Connection: close
                    Content-Length: 148
                    Content-Type: application/json
                    Date: Sun, 10 Sep 2023 05:49:10 GMT
                    Server: Werkzeug/2.3.7 Python/3.10.0

                    {
                      "matches": [
                        [
                          "",
                    ...
Forms             : {}
Headers           : {[Connection, close], [Content-Length, 148], [Content-Type, application/json], [Date, Sun, 10 Sep 2023 05:49:10 GMT]...}
Images            : {}
InputFields       : {}
Links             : {}
ParsedHtml        : mshtml.HTMLDocumentClass
RawContentLength  : 148
```

```
PROBLEMS  OUTPUT  TERMINAL  PORTS  JUPYTER  DEBUG CONSOLE

>>     "Content-Type" = "application/json"
>>     "Authorization" = "Bearer {token}"
>> }
PS F:\testing_purpose>
PS F:\testing_purpose> $body = @{
>>     "team_1" = "India"
>>     "team_2" = "Australia"
>>     "date" = "2023-07-12"
>>     "venue" = "Sydney Cricket Ground"
>> } | ConvertTo-Json
PS F:\testing_purpose>
PS F:\testing_purpose> Invoke-WebRequest -Uri "http://localhost:5001/api/matches" -Method POST -Headers $headers -Body $body


StatusCode        : 200
StatusDescription : OK
Content           : {
                      "match_id": 0,
                      "message": "Match created successfully"
                    }

RawContent        : HTTP/1.1 200 OK
                    Connection: close
                    Content-Length: 63
                    Content-Type: application/json
                    Date: Sun, 10 Sep 2023 05:48:56 GMT
                    Server: Werkzeug/2.3.7 Python/3.10.0

                    {
                      "match_id": 0,
                      "message": "Ma...
Forms             : {}
Headers           : {[Connection, close], [Content-Length, 63], [Content-Type, application/json], [Date, Sun, 10 Sep 2023 05:48:56 GMT]...}
Images            : {}
InputFields       : {}
Links             : {}
ParsedHtml        : mshtml.HTMLDocumentClass
RawContentLength  : 63
```

```
PS F:\testing_purpose> $headers = @{
>>     "Content-Type" = "application/json"
>> }
PS F:\testing_purpose>
PS F:\testing_purpose> $body = @{
>>     "username" = "admin_user"
>>     "password" = "admin_password"
>> } | ConvertTo-Json
PS F:\testing_purpose>
PS F:\testing_purpose> Invoke-WebRequest -Uri "http://localhost:5001/api/admin/login" -Method POST -Headers $headers -Body $body


StatusCode        : 200
StatusDescription : OK
Content           : {
                      "access_token": "dummy_token",
                      "status": "Login successful",
                      "status_code": 200,
                      "user_id": ""
                    }

RawContent        : HTTP/1.1 200 OK
                    Connection: close
                    Content-Length: 107
                    Content-Type: application/json
                    Date: Sun, 10 Sep 2023 05:48:41 GMT
                    Server: Werkzeug/2.3.7 Python/3.10.0

                    {
                      "access_token": "dummy_token",...
Forms             : {}
Headers           : {[Connection, close], [Content-Length, 107], [Content-Type, application/json], [Date, Sun, 10 Sep 2023 05:48:41 GMT]...}
Images            : {}
InputFields       : {}
Links             : {}
ParsedHtml        : mshtml.HTMLDocumentClass
RawContentLength  : 107
```

```
PS F:\testing_purpose> $headers = @{
>>     "Content-Type" = "application/json"
>> }
PS F:\testing_purpose>
PS F:\testing_purpose> $body = @{
>>     "username" = "admin_user"
>>     "password" = "admin_password"
>>     "email" = "admin@example.com"
>> } | ConvertTo-Json
PS F:\testing_purpose>
PS F:\testing_purpose> Invoke-WebRequest -Uri "http://localhost:5001/api/admin/signup" -Method POST -Headers $headers -Body $body


StatusCode        : 200
StatusDescription : OK
Content           : {
                      "status": "Admin Account successfully created",
                      "status_code": 200,
                      "user_id": 0
                    }

RawContent        : HTTP/1.1 200 OK
                    Connection: close
                    Content-Length: 91
                    Content-Type: application/json
                    Date: Sun, 10 Sep 2023 05:48:26 GMT
                    Server: Werkzeug/2.3.7 Python/3.10.0

                    {
                      "status": "Admin Account succes...
Forms             : {}
Headers           : {[Connection, close], [Content-Length, 91], [Content-Type, application/json], [Date, Sun, 10 Sep 2023 05:48:26 GMT]...}
Images            : {}
InputFields       : {}
Links             : {}
ParsedHtml        : mshtml.HTMLDocumentClass
RawContentLength  : 91
```

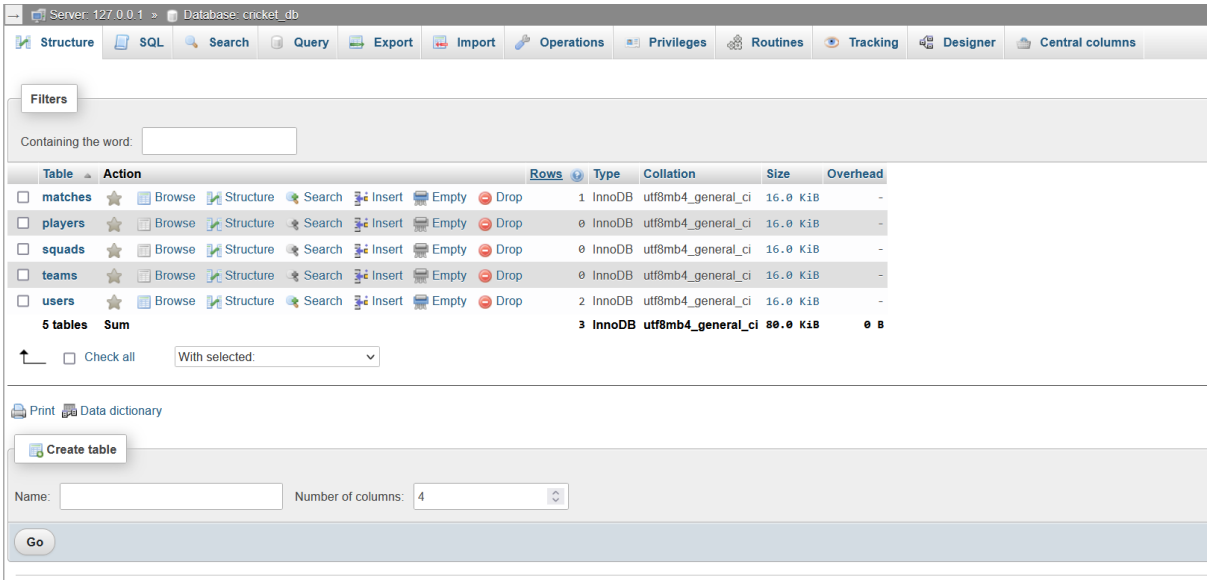
```
PS F:\testing_purpose> curl http://127.0.0.1:5001/api/matches/0


StatusCode        : 200
StatusDescription : OK
Content           : {
                      "message": "Match not found",
                      "status_code": 404
                    }

RawContent        : HTTP/1.1 200 OK
                    Connection: close
                    Content-Length: 57
                    Content-Type: application/json
                    Date: Sun, 10 Sep 2023 05:42:18 GMT
                    Server: Werkzeug/2.3.7 Python/3.10.0

                    {
                      "message": "Match not found",
                    ...
Forms             : {}
Headers           : {[Connection, close], [Content-Length, 57], [Content-Type, application/json], [Date, Sun, 10 Sep 2023 05:42:18 GMT]...}
Images            : {}
InputFields       : {}
Links             : {}
ParsedHtml        : mshtml.HTMLDocumentClass
RawContentLength  : 57
```

Data base:



Code:

```python
from flask import Flask, request, jsonify
from flaskext.mysql import MySQL

app = Flask(__name__)

# configure
app.config['MYSQL_HOST'] = 'localhost'
app.config['MYSQL_USER'] = 'root'
app.config['MYSQL_PASSWORD'] = 'abilash'
app.config['MYSQL_DATABASE_DB'] = 'cricket_db'

# intialize
```

```
mysql = MySQL(app)

# Function to get a cursor
def get_cursor():
    return mysql.get_db().cursor()

# Function to commit changes
def commit_changes():
    mysql.get_db().commit()

# Endpoint for registering an admin user
@app.route('/',methods = ['GET'])
def greet():
    return "hi"

@app.route('/api/admin/signup', methods=['POST'])
def register_admin():
    data = request.json
    username = data['username']
    password = data['password']
    email = data['email']
    cur = get_cursor()
    cur.execute("INSERT INTO users (username, password, email, role) VALUES
(%s, %s, %s, %s)", (username, password, email, 'admin'))
    commit_changes()
    cur.close()
    return jsonify({"status": "Admin Account successfully created",
"status_code": 200, "user_id": cur.lastrowid})

# Endpoint for user login
@app.route('/api/admin/login', methods=['POST'])
def login_user():
    data = request.json
    username = data['username']
    password = data['password']
    cur = get_cursor()
    cur.execute("SELECT * FROM users WHERE username = %s AND password = %s",
(username, password))
    user = cur.fetchone()
    if user:
        return jsonify({"status": "Login successful", "status_code": 200,
"user_id": user[0], "access_token": "dummy_token"})
    return jsonify({"status": "Incorrect username/password provided. Please
retry", "status_code": 401})

# Endpoint for creating a match
@app.route('/api/matches', methods=['POST'])
def create_match():
```

```
    data = request.json
    team_1 = data['team_1']
    team_2 = data['team_2']
    date = data['date']
    venue = data['venue']
    cur = get_cursor()
    cur.execute("INSERT INTO matches (team_1, team_2, date, venue) VALUES (%s,
%s, %s, %s)", (team_1, team_2, date, venue))
    commit_changes()
    cur.close()
    return jsonify({"message": "Match created successfully", "match_id":
cur.lastrowid})

# Endpoint for getting match schedules
@app.route('/api/matches', methods=['GET'])
def get_match_schedules():
    cur = get_cursor()
    cur.execute("SELECT * FROM matches")
    matches = cur.fetchall()
    return jsonify({"matches": matches})

# Endpoint for getting match details by match ID
@app.route('/api/matches/<match_id>', methods=['GET'])
def get_match_details(match_id):
    cur = get_cursor()
    cur.execute("SELECT * FROM matches WHERE match_id = %s", (match_id,))
    match = cur.fetchone()
    if match:

        squads = {
            "team_1": [
                {"player_id": "123", "name": "Virat Kohli"},
                {"player_id": "456", "name": "Jasprit Bumrah"}
            ],
            "team_2": [
                {"player_id": "789", "name": "Kane Williamson"},
                {"player_id": "1011", "name": "Trent Boult"}
            ]
        }
        match_details = {
            "match_id": match[0],
            "team_1": match[1],
            "team_2": match[2],
            "date": match[3].strftime('%Y-%m-%d'),
            "venue": match[4],
            "status": "upcoming",
            "squads": squads
        }
```

```
        return jsonify(match_details)
    return jsonify({"message": "Match not found", "status_code": 404})

# endpoint for adding a player to a team's squad
@app.route('/api/teams/<team_id>/squad', methods=['POST'])
def add_player_to_squad(team_id):
    data = request.json
    name = data['name']
    role = data['role']
    player_id = 789
    return jsonify({"message": "Player added to squad successfully",
"player_id": player_id})

# Endpoint for getting player statistics
@app.route('/api/players/<player_id>/stats', methods=['GET'])
def get_player_stats(player_id):
    # Retrieve player statistics from the database based on player_id
    player_stats = {
        "player_id": player_id,
        "name": "Virat Kohli",
        "matches_played": 200,
        "runs": 12000,
        "average": 59.8,
        "strike_rate": 92.5
    }
    return jsonify(player_stats)

if __name__ == '__main__':
    app.run(debug=True,port=5001)
```