Q-1:

class Player {

constructor(name, score) {

this.name = name;

this.score = score;

}

}

const prompt = require('prompt-sync')();

class Team {

constructor(name) {

this.name = name;

this.players = [];

}

addPlayer(player) {

this.players.push(player);

}

getAverageScore() {

let totalScore = 0;

for (let player of this.players) {

totalScore += player.score;

}

return totalScore / this.players.length;

}

getMinimumScore() {

let minScore = Infinity;

for (let player of this.players) {

if (player.score < minScore) {

minScore = player.score;

}

}

return minScore;

}

getMaximumScore() {

let maxScore = -Infinity;

for (let player of this.players) {

if (player.score > maxScore) {

maxScore = player.score;

}

}

return maxScore;

}

promptForPlayers() {

let numPlayers = parseInt(prompt("Enter number of players:"));

for (let i = 0; i < numPlayers; i++) {

let playerName = prompt(`Enter player ${i + 1} name:`);

let playerScore = parseInt(prompt(`Enter player ${i + 1} score:`));

this.addPlayer(new Player(playerName, playerScore));

}

}

}

function main() {

let teamName = prompt("Enter team name:");

let team = new Team(teamName);

team.promptForPlayers();

console.log(`Average score: ${team.getAverageScore()}`);

console.log(`Minimum score: ${team.getMinimumScore()}`);

console.log(`Maximum score: ${team.getMaximumScore()}`);

}

main();

Q2:

const prompt = require('prompt-sync')();

class Player {

constructor(name, score) {

this.name = name;

this.score = score;

}

}

class Team {

constructor(name) {

this.name = name;

this.players = [];

}

promptForPlayers() {

let numPlayers = parseInt(prompt("Enter number of players:"));

for (let i = 0; i < numPlayers; i++) {

let playerName = prompt(`Enter player ${i + 1} name:`);

let playerScore = parseInt(prompt(`Enter player ${i + 1} score:`));

this.addPlayer(new Player(playerName, playerScore));

}

}

addPlayer(player) {

this.players.push(player);

}

getAverageScore() {

let totalScore = 0;

for (let player of this.players) {

totalScore += player.score;

}

return totalScore / this.players.length;

}

getMinimumScore() {

let minScore = Infinity;

for (let player of this.players) {

if (player.score < minScore) {

minScore = player.score;

}

}

return minScore;

}

getMaximumScore() {

let maxScore = -Infinity;

for (let player of this.players) {

if (player.score > maxScore) {

maxScore = player.score;

}

}

return maxScore;

}

}

const {MongoClient} = require('mongodb');

url="mongodb+srv://puneethhsbsc22:puneethhs9112004@cluster0.9zye9lm.mongodb.net/?retryWrites=true&w=majority";

const client = new MongoClient(url, { useNewUrlParser: true });

const database = client.db('Teams');

const collection = database.collection('Top');

async function createTeam(team) {

try {

await client.connect();

const db = client.db('Teams');

const collection = db.collection('Top');

let result = await collection.insertOne(team);

console.log(`Created team with id: ${result.insertedId}`);

} finally {

await client.close();

}

}

async function inserting\_data(s)

{

data=[]

for(let i=0;i<s.players.length;i++)

{

data.push({Name:s.players[i].name,Score:s.players[i].score});

}

result = await collection.insertMany(data);

await console.log("Inserted the Data");

}

async function readTeam(name) {

try {

await client.connect();

const db = client.db('Teams');

const collection = db.collection('Top');

let team = await collection.findOne({ name });

console.log(`Found team with name '${name}':`, team);

} finally {

await client.close();

}

}

async function updateTeam(name, update) {

try {

await client.connect();

const db = client.db('Teams');

const collection = db.collection('Top');

let name = prompt("Select name of the player: ");

let score = Number(prompt("Select score of the Student: "));

await collection.updateOne({Name:name},{$set:{Score:score}});

await console.log("Updated the Record")

result = await collection.find({Score:score}).toArray();

await console.log(result);

} finally {

await client.close();

}

}

async function deleteTeam(name) {

try {

await client.connect();

const db = client.db('Teams');

const collection = db.collection('Top');

let name = prompt("Select name of the player: ");

await collection.deleteOne({Name:name});

await console.log("Deleted Record")

} finally {

await client.close();

}

}

function main() {

let teamName = prompt("Enter team name:");

let team = new Team(teamName);

team.promptForPlayers();

// inserting\_data(team);

// deleteTeam();

// updateTeam();

}

main();

Q3:

const prompt = require('prompt-sync')();

class Player {

constructor(name, score) {

this.name = name;

this.score = score;

}

}

class Team {

constructor(name) {

this.name = name;

this.players = [];

}

promptForPlayers() {

let numPlayers = parseInt(prompt("Enter number of players:"));

for (let i = 0; i < numPlayers; i++) {

let playerName = prompt(`Enter player ${i + 1} name:`);

let playerScore = parseInt(prompt(`Enter player ${i + 1} score:`));

this.addPlayer(new Player(playerName, playerScore));

}

}

addPlayer(player) {

this.players.push(player);

}

getAverageScore() {

let totalScore = 0;

for (let player of this.players) {

totalScore += player.score;

}

return totalScore / this.players.length;

}

getMinimumScore() {

let minScore = Infinity;

for (let player of this.players) {

if (player.score < minScore) {

minScore = player.score;

}

}

return minScore;

}

getMaximumScore() {

let maxScore = -Infinity;

for (let player of this.players) {

if (player.score > maxScore) {

maxScore = player.score;

}

}

return maxScore;

}

}

const mongoose = require('mongoose')

database='Teams'

url=`mongodb+srv://puneethhsbsc22:puneethhs9112004@cluster0.9zye9lm.mongodb.net/${database}?retryWrites=true&w=majority`

database = mongoose.connect(url)

console.log("Connected to Database")

const coll\_schema = new mongoose.Schema(

{

Name:String,

Score:Number,

}

)

const collection = new mongoose.model("Insane",coll\_schema);

async function inserting\_data(s)

{

for(let i=0;i<s.players.length;i++)

{

let doc = new collection(

{

Name:s.players[i].name,

Score:s.players[i].score,

}

)

await doc.save();

}

await console.log("Inserted Documents");

}

async function readTeam(name) {

let team = await collection.findOne({ name });

console.log(`Found team with name '${name}':`, team);

}

async function updateTeam() {

let name = prompt("Select name of the player: ");

let score = Number(prompt("Select score of the Student: "));

await collection.updateOne({Name:name},{$set:{Score:score}});

await console.log("Updated the Record")

result = await collection.find({Score:score}).toArray();

await console.log(result);

}

async function deleteTeam() {

let name = prompt("Select name of the player: ");

await collection.deleteOne({Name:name});

await console.log("Deleted Record")

}

function main() {

let teamName = prompt("Enter team name:");

let team = new Team(teamName);

// team.promptForPlayers();

// inserting\_data(team);

readTeam();

// deleteTeam();

// updateTeam();

}

main();

Q4:

<!DOCTYPE html>

<html>

<head>

<script src="https://unpkg.com/react/umd/react.development.js"></script>

<script src="https://unpkg.com/react-dom/umd/react-dom.development.js"></script>

<script src="https://unpkg.com/babel-standalone"></script>

</head>

<body>

<div id="root"></div>

<script type="text/babel">

class Player {

constructor(name, score) {

this.name = name;

this.score = score;

}

}

function PlayerForm({ onSubmit, onUpdate, isEditMode, playerToEdit }) {

const [name, setName] = React.useState('');

const [score, setScore] = React.useState('');

React.useEffect(() => {

if (isEditMode && playerToEdit) {

setName(playerToEdit.name);

setScore(playerToEdit.score);

}

}, [isEditMode, playerToEdit]);

function handleSubmit(e) {

e.preventDefault();

if (isEditMode) {

onUpdate(playerToEdit, new Player(name, score));

} else {

onSubmit(new Player(name, score));

}

setName('');

setScore('');

}

return (

<form onSubmit={handleSubmit}>

<input

type="text"

placeholder="Player name"

value={name}

onChange={e => setName(e.target.value)}

/>

<input

type="number"

placeholder="Player score"

value={score}

onChange={e => setScore(parseInt(e.target.value))}

/>

<button type="submit">{isEditMode ? 'Update' : 'Add Player'}</button>

</form>

);

}

function PlayerList({ players, onDelete, onEdit }) {

return (

<ul>

{players.map(player => (

<li key={player.name}>

{player.name} - {player.score}

<button onClick={() => onDelete(player)}>Delete</button>

<button onClick={() => onEdit(player)}>Update</button>

</li>

))}

</ul>

);

}

function TeamComponent() {

const [players, setPlayers] = React.useState([]);

const [isEditMode, setIsEditMode] = React.useState(false);

const [playerToEdit, setPlayerToEdit] = React.useState(null);

function handleAddPlayer(player) {

setPlayers([...players, player]);

}

function handleDeletePlayer(player) {

setPlayers(players.filter(p => p !== player));

}

function handleEditPlayer(player) {

setIsEditMode(true);

setPlayerToEdit(player);

}

function handleUpdatePlayer(oldPlayer, updatedPlayer) {

setPlayers(players.map(player => {

if (player.name === oldPlayer.name) {

return updatedPlayer;

}

return player;

}));

setIsEditMode(false);

setPlayerToEdit(null);

}

function calculateAverageScore() {

if (players.length === 0) {

return 0;

}

const totalScore = players.reduce((sum, player) => sum + player.score, 0);

return totalScore / players.length;

}

function calculateMinimumScore() {

if (players.length === 0) {

return 0;

}

return Math.min(...players.map(player => player.score));

}

function calculateMaximumScore() {

if (players.length === 0) {

return 0;

}

return Math.max(...players.map(player => player.score));

}

return (

<div>

<h1>My Team</h1>

<PlayerForm

onSubmit={handleAddPlayer}

onUpdate={handleUpdatePlayer}

isEditMode={isEditMode}

playerToEdit={playerToEdit}

/>

{players.length > 0 ? (

<PlayerList

players={players}

onDelete={handleDeletePlayer}

onEdit={handleEditPlayer}

/>

) : (

<p>No players added yet.</p>

)}

<p><b>Average score: {calculateAverageScore()}</b></p>

<p><b>Minimum score: {calculateMinimumScore()}</b></p>

<p><b>Maximum score: {calculateMaximumScore()}</b></p>

</div>

);

}

ReactDOM.render(<TeamComponent />, document.getElementById('root'));

</script>

</body>

</html>

Q 5:

const express = require('express');

const { MongoClient, ObjectId } = require('mongodb');

const prompt = require('prompt-sync')();

const app = express();

const port = 3000;

const uri = 'mongodb+srv://puneethhsbsc22:puneethhs9112004@cluster0.9zye9lm.mongodb.net/?retryWrites=true&w=majority';

const client = new MongoClient(uri);

app.use(express.json());

class Player {

constructor(id, name, score) {

this.\_id = id;

this.name = name;

this.score = score;

}

}

let db;

let playersCollection;

async function connectToDatabase() {

try {

await client.connect();

db = client.db('Teamplayer');

playersCollection = db.collection('players');

console.log('Connected to the database');

} catch (error) {

console.error('Error connecting to the database:', error);

}

}

connectToDatabase();

// POST - Create a new player

app.post('/players', async (req, res) => {

const name = prompt('Enter player name: ');

const score = parseInt(prompt('Enter player score: '));

const player = new Player(null, name, score);

const result = await playersCollection.insertOne(player);

player.\_id = result.insertedId;

res.status(201).json(player);

});

// GET - Get all players

app.get('/players', async (req, res) => {

const players = await playersCollection.find().toArray();

res.json(players);

});

// GET - Get a specific player by ID

app.get('/players/gbn', async (req, res) => {

const Name = prompt('Enter the player name: ');

const player = await playersCollection.findOne({name:Name});

if (player) {

res.json(player);

} else {

res.status(404).json({ error: 'Player not found' });

}

});

// PUT - Update a player by ID

app.put('/players/ubn', async (req, res) => {

// const name = req.params.name;

const Name = prompt('Enter the player name: ');

const Score = parseInt(prompt('Enter updated player score: '));

const result = await playersCollection.updateOne(

{ name: Name },

{ $set: { score:Score } }

);

if (result.matchedCount > 0) {

const updatedPlayer = await playersCollection.findOne({name: Name});

res.json(updatedPlayer);

} else {

res.status(404).json({ error: 'Player not found' });

}

});

// DELETE - Delete a player by ID

app.delete('/players/dbn', async (req, res) => {

const Name = prompt('Enter the player name to delete: ');

const result = await playersCollection.deleteOne({name:Name});

if (result.deletedCount > 0) {

res.sendStatus(204);

} else {

res.status(404).json({ error: 'Player not found' });

}

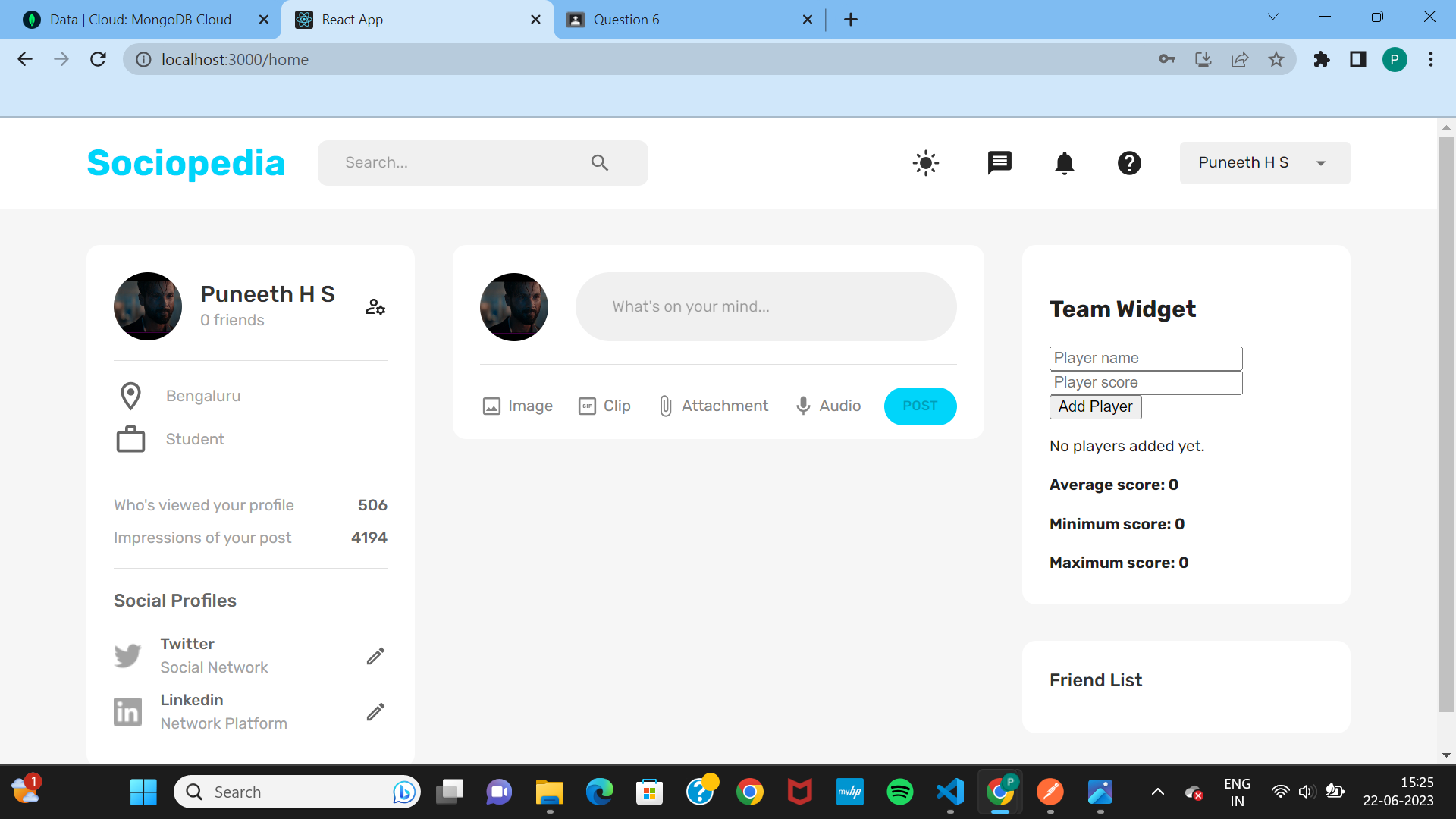
});

app.listen(port, () => {

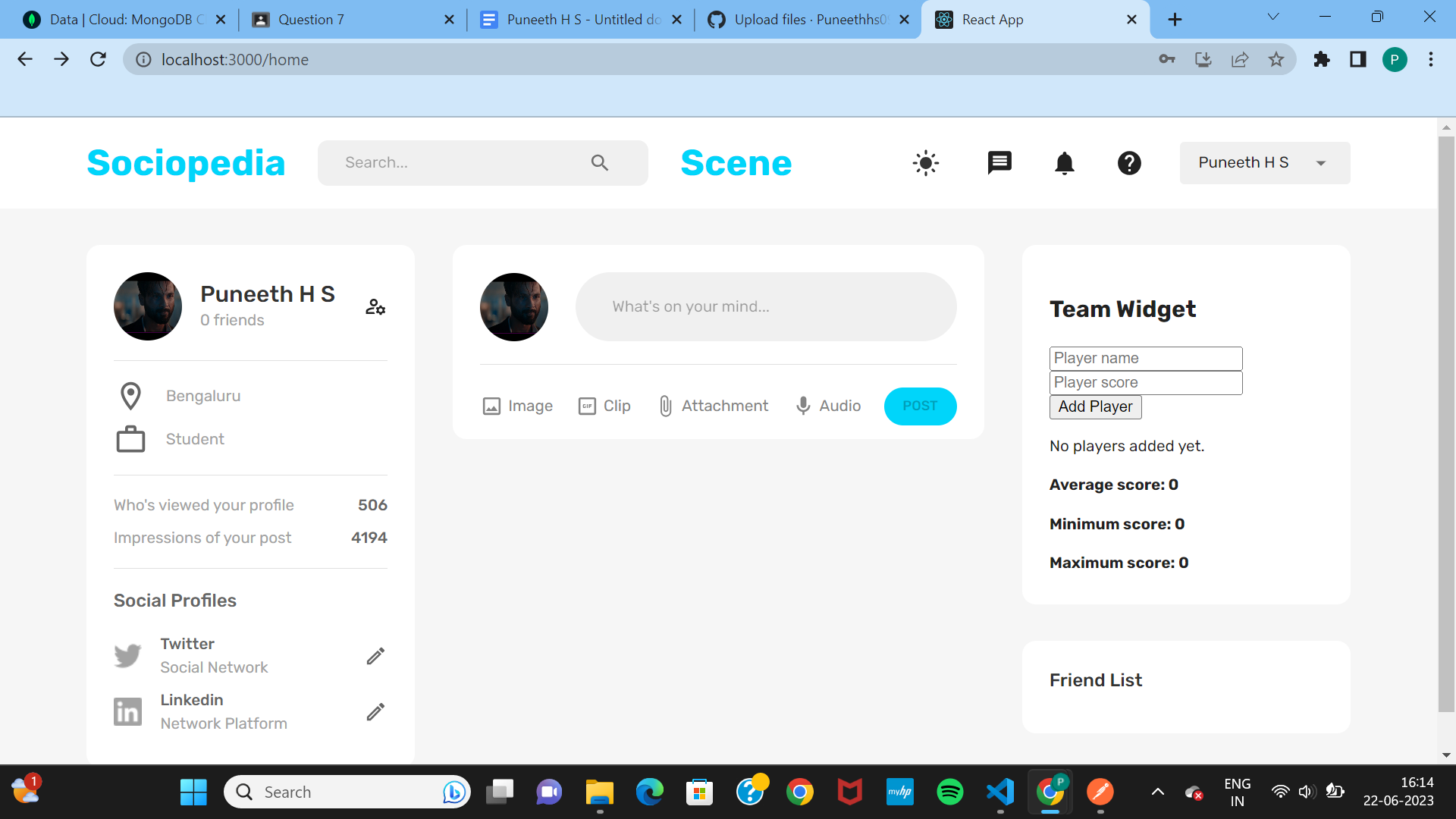
console.log(`Server is running on port ${port}`);

});

Q 6:



Q 7:



when you click scene it shows the below display,

