BMS COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

NOSQL AAT REPORT
on

ANALYSIS ON BEVERAGES

Submitted in partial fulfillment for the award of degree of

Bachelor of Engineering
in
Computer Science and Engineering

Submitted by:
Abhijnya K G – 1BM18CS002
Akanksha Laddha – 1BM18CS007
Ankitha – 1BM18CS016
Shikha N – 1BM18CS149

Under the Guidance of
Dr. Pallavi GB
Assistant Professor, BMSCE

Department of Computer Science and Engineering
BMS College of Engineering
Bull Temple Road, Basavanagudi, Bangalore 560 019

2021-2022

BMS COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the NoSQL mini Project titled "**ANALYSIS ON BEVERAGES**" has been carried out by **Akanksha Laddha(1BM18CS007), Abhijnya.K.G(1BM18CS002),Shikha.N(1BM18CS149),Ankitha(1BM18CS01)** during the academic year 2020-2021.

Dr Pallavi G B
Assistant Professor,
Department of Computer Science and Engineering
BMS College of Engineering, Bangalore

BMS COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DECLARATION

We,**Abhijnya.K.G(1BM18CS002),Shikha.N(1BM18CS149),Ankitha(1BM18CS01),Akanksha Laddha(1BM18CS007)** students of 7th Semester, B.E, Department of Computer Science and Engineering, BMS College of Engineering, Bangalore, hereby declare that, this assignment work entitled " ANALYSIS ON BEVERAGES" has been carried out by us under the guidance of Dr Pallavi G B Assistant Professor, Department of CSE, B.M.S College of Engineering, Bangalore during the academic semester Aug 2020- Jan 2021. We also declare that to the best of our knowledge and belief, the assignment reported here is not from part of any other report by any other students.

Signature of the Candidates

Abhijnya K G – 1BM18CS002
Akanksha Laddha – 1BM18CS007
Ankitha – 1BM18CS016
Shikha N – 1BM18CS149

1. INTRODUCTION

The Objective of the project is to build a full-stack web application that is a basic Beverage Management System using MongoDB, React.js and Flask with following features:

- Add Drink
- List Drinks
- Update Drink
- Delete Drink
- View Drink
- Search Drinks
- Analysis on Drinks

The dataset, **Caffeine Content of Drinks** used is taken from Kaggle which contains the list of drinks that generally contains caffeine and can be used to find caffeine amounts and calories of most drink brands and types.

The Dataset has the following attributes:

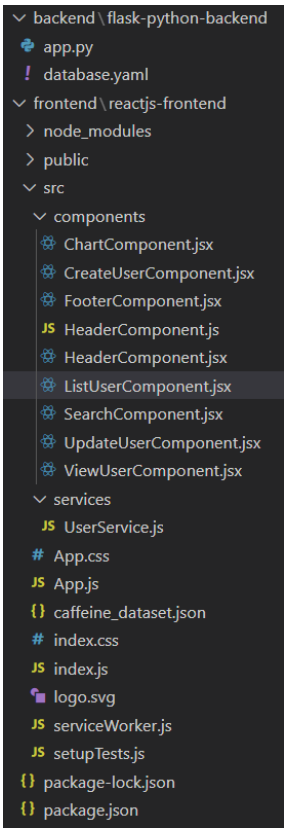
- drink: Drink's name
- Volume (ml): Volume quantity
- Calories: Calorie quantity
- Caffeine (mg): Caffeine quantity
- type: Drink's type

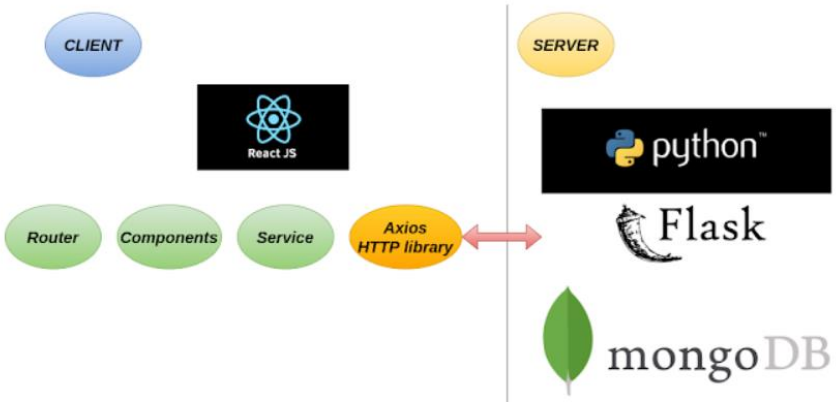
2.DESIGN MODULES

The Project is divided into five parts:

1. Rest APIs Development using Flask
2. Importing the Kaggle dataset into MongoDB
3. Connecting to MongoDB using PyMongo to perform CRUD Operations
4. UI development using React.js
5. Designing an analytical query and visualization

The Application has the following architecture and code structure:





3.CODE

app.py

```
app = Flask(__name__)
config = yaml.safe_load(open('database.yaml'))
client = MongoClient(config['uri'])
db = client['crud_db']
CORS(app)
```

```python
@app.route('/drinks', methods=['POST', 'GET'])
def data():

    # POST a data to database
    if request.method == 'POST':
        _json = request.json
        _drink_name = _json['drinkName']
        _volume = _json['volume']
        _calories = _json['calories']
        _caffeine = _json['caffeine']
        _type_of_drink = _json['type']
        # db.users.insert_one({
        db['drinks'].insert_one({
            'Drink Name': _drink_name,
            'Volume (ml)': _volume,
            'Calories':_calories,
            'Caffeine (mg)': _caffeine,
            'Type of Drink': _type_of_drink
        })
        return jsonify({
            'status': 'Data is posted to MongoDB!',
            'Drink Name': _drink_name,
            'Volume (ml)': _volume,
            'Calories':_calories,
            'Caffeine (mg)': _caffeine,
            'Type of Drink': _type_of_drink
        })
```

```python
# GET all data from database
if request.method == 'GET':
    allData = db['drinks'].find()
    dataJson = []
    for data in allData:
        id = data['_id']
        _drink_name = data['Drink Name']
        _volume = data['Volume (ml)']
        _calories = data['Calories']
        _caffeine = data['Caffeine (mg)']
        _type_of_drink = data['Type of Drink']
        dataDict = {
            'id': str(id),
            'Drink Name': _drink_name,
            'Volume (ml)': _volume,
            'Calories':_calories,
            'Caffeine (mg)': _caffeine,
            'Type of Drink': _type_of_drink
        }
        dataJson.append(dataDict)
    print(dataJson)
    return jsonify(dataJson)
```

```python
@app.route('/view-analytics/<string:chart>', methods=['GET'])
def graph(chart):
    if request.method == 'GET':
        types= ["Coffee", "Tea", "Energy Drinks", "Energy Shots", "Water", "Soft Drinks"]
        # To calculate average calorie content in each type of drink
        dataDict = {}
        if str(chart) == "Bar":
            for type in types:
                data = db['drinks'].find({'Type of Drink': type}, {'_id' : 0, 'Calories':1})
                length = len(list(data.clone()))
                sum = 0.0
                for val in data:
                    sum = sum + float(val['Calories'])
                avg = sum/length
                dataDict[type] = avg
            print(dataDict)
            return jsonify(dataDict)
        # To find max caffeine content in each type
        else:
            for type in types:
                data = db['drinks'].find({'Type of Drink': type}).sort('Caffeine (mg)', pymongo.DESCENDING).limit(1)
                for val in data:
                    caffeine = val['Caffeine (mg)']
                    dataDict[type] = caffeine
            print(dataDict)
            return jsonify(dataDict)
```

```python
@app.route('/drinks/<string:id>', methods=['GET', 'DELETE', 'PUT'])
def onedata(id):

    # GET a specific data by id
    if request.method == 'GET':
        data = db['drinks'].find_one({'_id': ObjectId(id)})
        id = data['_id']
        _drink_name = data['Drink Name']
        _volume = data['Volume (ml)']
        _calories = data['Calories']
        _caffeine = data['Caffeine (mg)']
        _type_of_drink = data['Type of Drink']
        dataDict = {
            'id': str(id),
            'Drink Name': _drink_name,
            'Volume (ml)': _volume,
            'Calories':_calories,
            'Caffeine (mg)': _caffeine,
            'Type of Drink': _type_of_drink
        }
        print(dataDict)
        return jsonify(dataDict)

    # DELETE a data
    if request.method == 'DELETE':
        db['drinks'].delete_many({'_id': ObjectId(id)})
        print('\n # Deletion successful # \n')
        return jsonify({'status': 'Data id: ' + id + ' is deleted!'})
```

```python
# UPDATE a data by id
if request.method == 'PUT':
    _json = request.json
    _drink_name = _json['drinkName']
    _volume = _json['volume']
    _calories = _json['calories']
    _caffeine = _json['caffeine']
    _type_of_drink = _json['type']

    db['drinks'].update_one(
        {'_id': ObjectId(id)},
        {
            "$set": {
                'Drink Name': _drink_name,
                'Volume (ml)': _volume,
                'Calories':_calories,
                'Caffeine (mg)': _caffeine,
                'Type of Drink': _type_of_drink
            }
        }
    )

    print('\n # Update successful # \n')
    return jsonify({'status': 'Data id: ' + id + ' is updated!'})
```

app.js

```
function App() {
  return (

    <div>
        <Router>
            <HeaderComponent />
              <div className="container">
                  <Switch>
                      <Route path = "/" exact component = {ListComponent}></Route>
                      <Route path = "/drinks" component = {ListComponent}></Route>
                      <Route path = "/add-drink/:id" component = {CreateComponent}></Route>
                      <Route path = "/view-drink/:id" component = {ViewComponent}></Route>
                      <Route path = "/view-analytics" component = {ChartComponent}></Route>
                      {/* <Route path = "/search/:name" component = {searchComponent}></Route> */}
                      {/* <Route path = "/update-drink/:id" component = {UpdateComponent}></Route> */}
                  </Switch>
              </div>
            <FooterComponent />
        </Router>
    </div>
```

DrinkService.

```
const USER_API_BASE_URL = "http://127.0.0.1:5000/drinks";
const GRAPH_API_BASE_URL = "http://127.0.0.1:5000/view-analytics";


class UserService {

    getUsers(){
        return axios.get(USER_API_BASE_URL);
    }

    createUser(user){
        return axios.post(USER_API_BASE_URL, user);
    }

    getUserById(userId){
        return axios.get(USER_API_BASE_URL + '/' + userId);
    }

    updateUser(user, userId){
        return axios.put(USER_API_BASE_URL + '/' + userId, user);
    }

    deleteUser(userId){
        return axios.delete(USER_API_BASE_URL + '/' + userId);
    }

    graphOne(type){
        return axios.get(GRAPH_API_BASE_URL + '/' + type);
    }
```

js

[database.yaml]

```
uri: 'mongodb://127.0.0.1:27017/crud_db'
```

[chartcomponent.js]

```
<Bar
 data={barData}
    options={{
        plugins:{
            title:{
                display:true,
                text:'Average Calorie Content in drinks',
                fontSize:20
            },
                legend:{
                display:true,
                position:'top'
            }
        }
    }}
/>
```

```
const labels1 = Object.keys( this.state.chartData)
const values1 = Object.values(this.state.chartData)
const barData = {
    labels: labels1,
    datasets: [
        {
            label: 'Avg Calorie Content',
            backgroundColor: ['rgba(75,192,192,1)'],
            borderColor: ['rgba(0,0,0,1)'],
            borderWidth: 2,
            data: values1
        }
    ]
}
```

```
const labels2 = Object.keys(this.state.pieData)
const values2 = Object.values(this.state.pieData)
const pieChart = {
    labels : labels2,
    datasets: [
        {
            label: 'Caffeine Content',
            backgroundColor: [
                '#B21F00',
                '#C9DE00',
                '#2FDE00',
                '#00A6B4',
                '#6800B4',
                '#FFC0CB'
              ],
              hoverBackgroundColor: [
              '#501800',
              '#4B5000',
              '#175000',
              '#003350',
              '#35014F',
              '#742a40'
              ],
              data : values2
        }
    ]
}
```

Listcomponent

```
class ListUserComponent extends Component {
    constructor(props) {
        super(props)

        this.state = {
                drinks: []
        }
        this.addUser = this.addUser.bind(this);
        this.editUser = this.editUser.bind(this);
        this.deleteUser = this.deleteUser.bind(this);
    }

    deleteUser(id){
        UserService.deleteUser(id).then( () => {
            this.setState({drinks: this.state.drinks.filter(drink => drink.id !== id)});
        });
    }
    viewUser(id){
        console.log(id);
        this.props.history.push(`/view-drink/${id}`);

    }
    editUser(id){
        this.props.history.push(`/add-drink/${id}`);
    }
```

```
componentDidMount(){
    UserService.getUsers().then((res) => {
        this.setState({ drinks: res.data});
    });
}

addUser(){
    this.props.history.push('/add-drink/_add');
}

addGraph(){
    this.props.history.push(`/view-analytics`);
}

render() {

    const filterDrinks = (drinksList, query) => {
        return drinksList.filter((drink) => {
            const drinkName = drink['Drink Name'];
            return drinkName.includes(query);
        });
    };

    const { search } = window.location;
const query = new URLSearchParams(search).get('s');
const filteredDrinks = filterDrinks(drinksList,query);
```

```
<ul>
    {filteredDrinks.map((drink) => (
        <li key={drink['id']}>
            <div className = "card col-md-6 offset-md-3">
        <br></br><h3 className = "text-center"> View Details</h3>
        <div className = "card-body">
        <div className="row">
            <label> Drink Name:
            {drink['Drink Name']}
            </label>
        </div>

        <div className="row">
            <label> Volume (ml):
            {drink['Volume (ml)']}
            </label>
        </div>

        <div className="row">
            <label> Calories:
            {drink['Calories']}
            </label>
        </div>

        <div className="row">
            <label>Caffeine (mg):
            {drink['Caffeine (mg)']}
            </label>
        </div>
```

```
{
    this.state.drinks.map(
        drink =>
        <tr key = {drink['_id']}>
        <td> {drink['Drink Name']} </td>
       <td> {drink['Volume (ml)']} </td>
       <td> {drink['Calories']} </td>
       <td> {drink['Caffeine (mg)']} </td>
       <td> {drink['Type of Drink']} </td>
       <td>
                <button onClick={ () => this.editUser(drink['id'])} className="btn btn-info">Update </button>
                <button style={{marginLeft: "10px"}} onClick={ () => this.deleteUser(drink['id'])}
                className="btn btn-danger">Delete </button>
                <button style={{marginLeft: "10px"}} onClick={ () => this.viewUser(drink['id'])}
                className="btn btn-info">View </button>
            </td>
        </tr>
    )
}
```

createComponent

```
class CreateDrinkComponent extends Component {
    constructor(props) {
        super(props)

        this.state = {
            id: this.props.match.params.id,
            drinkName: '',
            volume: '',
            calories: '',
            caffeine: '',
            type: ''
        }
        this.changeDrinkNameHandler = this.changeDrinkNameHandler.bind(this);
        this.changeVolumeHandler = this.changeVolumeHandler.bind(this);
        this.changeCaloriesHandler = this.changeCaloriesHandler.bind(this);
        this.changeCaffeineHandler = this.changeCaffeineHandler.bind(this);
        this.changeTypeHandler = this.changeTypeHandler.bind(this);
        this.saveOrUpdateUser = this.saveOrUpdateUser.bind(this);
    }
```

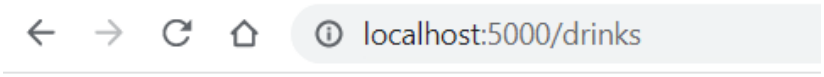
```
componentDidMount(){

    if(this.state.id === '_add'){
        return
    }else{
        UserService.getUserById(this.state.id).then( (res) =>{
            let drink = res.data;
            this.setState({drinkName: drink['Drink Name'],
            volume: drink['Volume (ml)'],
            calories: drink['Calories'],
            caffeine: drink['Caffeine (mg)'],
            type: drink['Type of Drink']
            });
        });
    }
}
```

```
saveOrUpdateUser = (e) => {
    e.preventDefault();
    let user = {drinkName: this.state.drinkName,
                volume: this.state.volume,
                calories: this.state.calories,
                caffeine: this.state.caffeine,
                type: this.state.type};
    console.log('user => ' + JSON.stringify(user));

    if(this.state.id === '_add'){
        UserService.createUser(user).then(res =>{
            this.props.history.push('/drinks');
        });
    }else{
        UserService.updateUser(user, this.state.id).then( res => {
            this.props.history.push('/drinks');
        });
    }
}

changeDrinkNameHandler = (event) => {
    this.setState({ drinkName: event.target.value });
}
changeVolumeHandler = (event) => {
    this.setState({ volume: event.target.value });
}
changeCaloriesHandler= (event) => {
    this.setState({ calories: event.target.value });
}
```

caffeine_dataset.json


```
{
  "Drink Name": "Nespresso Coffee Capsules",
  "Volume (ml)": 39.924225,
  "Calories": 0,
  "Caffeine (mg)": 60,
  "Type of Drink": "Coffee"
},
{
  "Drink Name": "Stok Coffee Shots",
  "Volume (ml)": 12.716605,
  "Calories": 10,
  "Caffeine (mg)": 40,
  "Type of Drink": "Coffee"
},
{
  "Drink Name": "Starbucks Bottled Iced Coffee",
  "Volume (ml)": 1419.528,
  "Calories": 240,
  "Caffeine (mg)": 640,
  "Type of Drink": "Coffee"
},
{
  "Drink Name": "Baskin Robbins Cappuccino Blast",
  "Volume (ml)": 709.764,
  "Calories": 470,
  "Caffeine (mg)": 234,
  "Type of Drink": "Coffee"
},
```

MongoDB server - View collection




```
[
  {
    "Caffeine (mg)": 277,
    "Calories": 0,
    "Drink Name": "Costa Coffee",
    "Type of Drink": "Coffee",
    "Volume (ml)": 256.99371499999995,
    "id": "61be487d76880d92e8208760"
  },
  {
    "Caffeine (mg)": 145,
    "Calories": 0,
    "Drink Name": "Coffee Friend Brewed Coffee",
    "Type of Drink": "Coffee",
    "Volume (ml)": 250.19181000000003,
    "id": "61be487d76880d92e8208761"
  },
  {
    "Caffeine (mg)": 100,
    "Calories": 150,
    "Drink Name": "Hell Energy Coffee",
    "Type of Drink": "Coffee",
    "Volume (ml)": 250.19181000000003,
    "id": "61be487d76880d92e8208762"
  },
  {
    "Caffeine (mg)": 430,
    "Calories": 0,
    "Drink Name": "Killer Coffee (AU)",
    "Type of Drink": "Coffee",
    "Volume (ml)": 250.19181000000003,
    "id": "61be487d76880d92e8208763"
  },
  {
    "Caffeine (mg)": 66,
    "Calories": 0,
    "Drink Name": "Nescafe Gold",
    "Type of Drink": "Coffee",
    "Volume (ml)": 250.19181000000003,
    "id": "61be487d76880d92e8208764"
  },
  {
    "Caffeine (mg)": 160,
    "Calories": 170,
    "Drink Name": "Espresso Monster",
    "Type of Drink": "Coffee",
    "Volume (ml)": 248.41740000000001,
    "id": "61be487d76880d92e8208765"
  }
```

updateDrinkcomponent.jsx

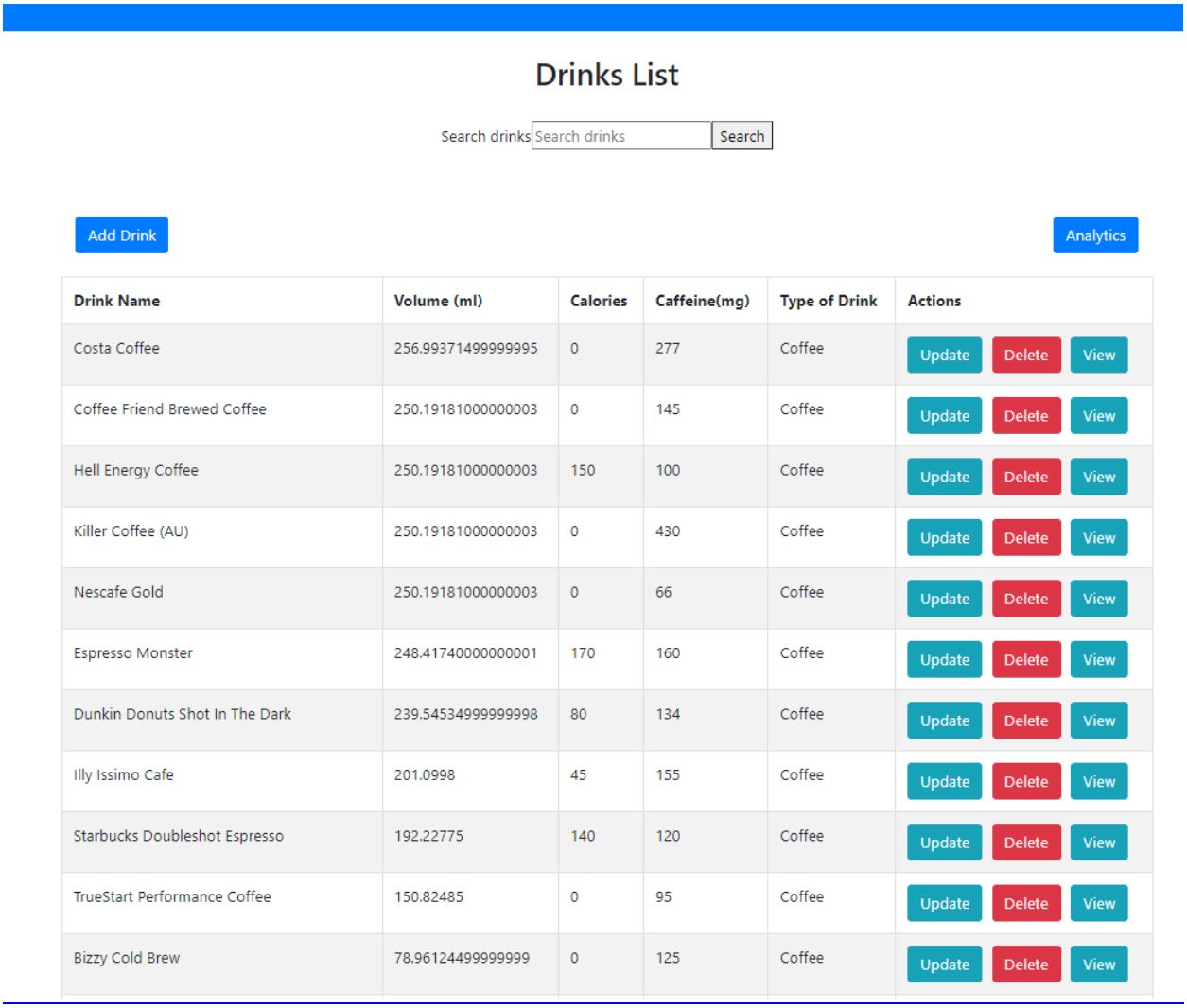
```
updateDrink = (e) => {
    e.preventDefault();
    let user = {drinkName: this.state.drinkName,
                volume: this.state.volume,
                calories: this.state.calories,
                caffeine: this.state.caffeine,
                type: this.state.type};
    console.log('user => ' + JSON.stringify(user));
    console.log('id => ' + JSON.stringify(this.state.id));
    DrinkService.update(user, this.state.id).then( res => {
        this.props.history.push('/drinks');
    });
}
```

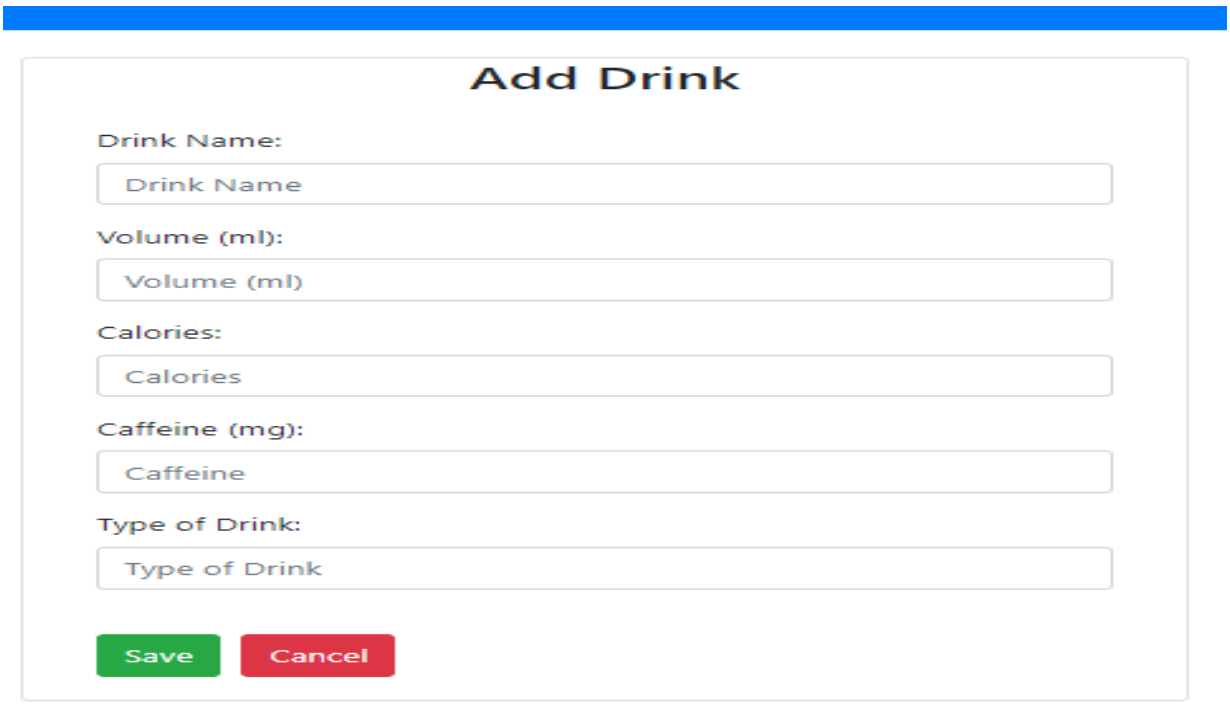
```
render() {
    return (
        <div>
            <br></br>
                <div className = "container">
                    <div className = "row">
                        <div className = "card col-md-6 offset-md-3 offset-md-3">
                            <h3 className="text-center">Update Drink</h3>
                            <div className = "card-body">
                                <form>
                                <div className="form-group">
                                <label> Drink Name : </label>
<input placeholder="Drink Name" name="drinkName" className="form-control"
  value={this.state.drinkName} onChange={this.changeDrinkNameHandler} />
                                </div>

                                <div className="form-group">
                                    <label> Volume (ml): </label>
<input placeholder="Volume (ml)" name="volume" className="form-control"
  value={this.state.volume} onChange={this.changeVolumeHandler} />
                                </div>

                                <div className="form-group">
                                    <label> Calories: </label>
<input placeholder="Calories" name="calories" className="form-control"
  value={this.state.calories} onChange={this.changeCaloriesHandler} />
                                </div>
```

viewDrinkcomponent.jsx

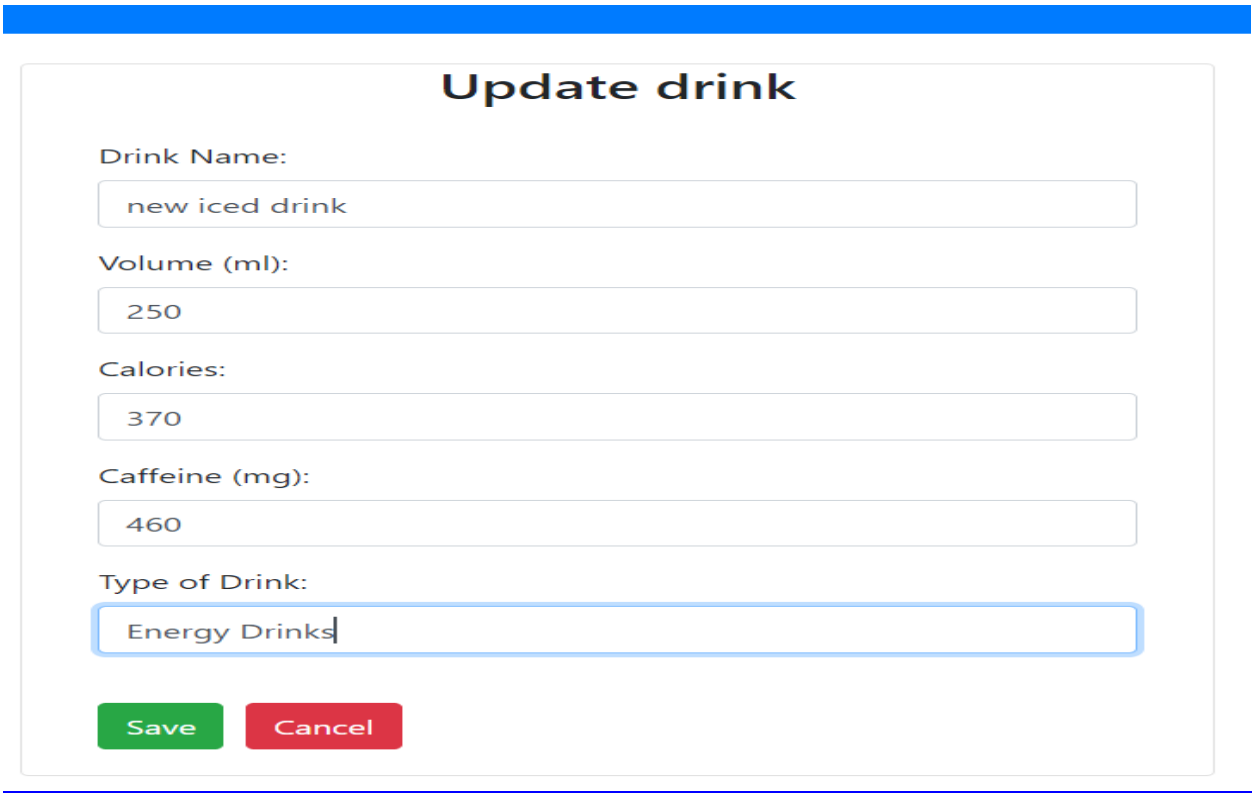
```
        this.state = {
            id: this.props.match.params.id,
            drink: {}
        }
    }

    componentDidMount(){
        UserService.getUserById(this.state.id).then( res => {
            this.setState({drink: res.data});
        })
    }

    render() {
        return (
            <div>
                <br></br>
                <div className = "card col-md-6 offset-md-3">
                    <h3 className = "text-center"> View Details</h3>
                    <div className = "card-body">
                    <div className="row">
                        <label> Drink Name:
                        {this.state.drink['Drink Name']}
                        </label>
                    </div>

                    <div className="row">
                        <label> Volume (ml):
                        {this.state.drink['Volume (ml)']}
                        </label>
                    </div>
```

User Interface:





new iced drink	250	370	460	Coffee	Update Delete View



new iced drink	250	370	460	Energy Drinks	Update Delete View

Yerbae Sparkling Water	354.882	0	100	Water	Update Delete View
Tonic Water	295.735	110	0	Water	Update Delete View
MiO Energy Water Enhancer	236.588	0	60	Water	Update Delete View

Yerbae Sparkling Water	354.882	0	100	Water	Update Delete View
MiO Energy Water Enhancer	236.588	0	60	Water	Update Delete View

View Details

Drink Name:Bubly Bounce Sparkling Water

Volume (ml):354.882

Calories:0

Caffeine (mg):35

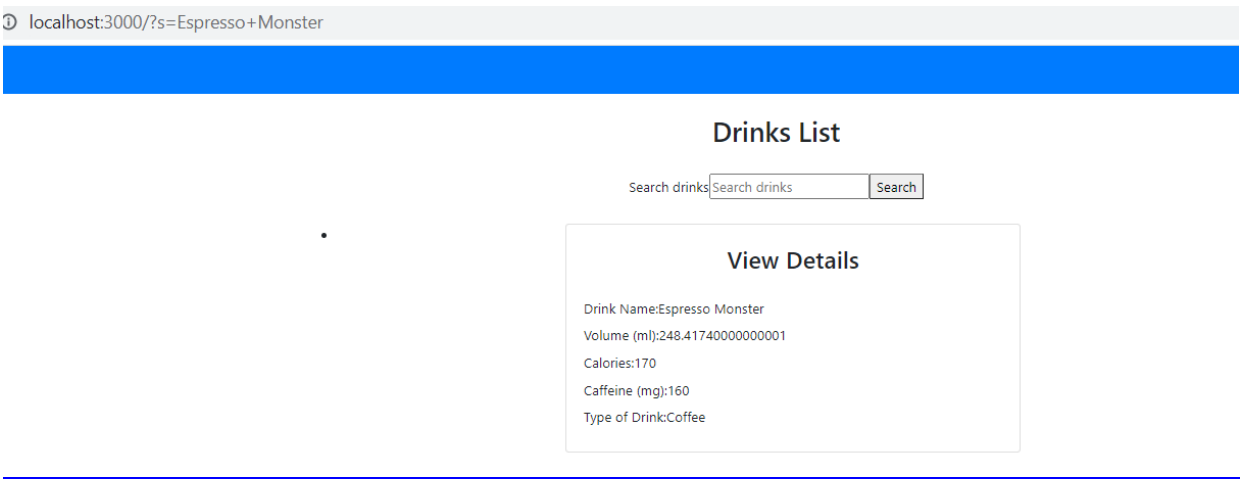
Type of Drink:Water

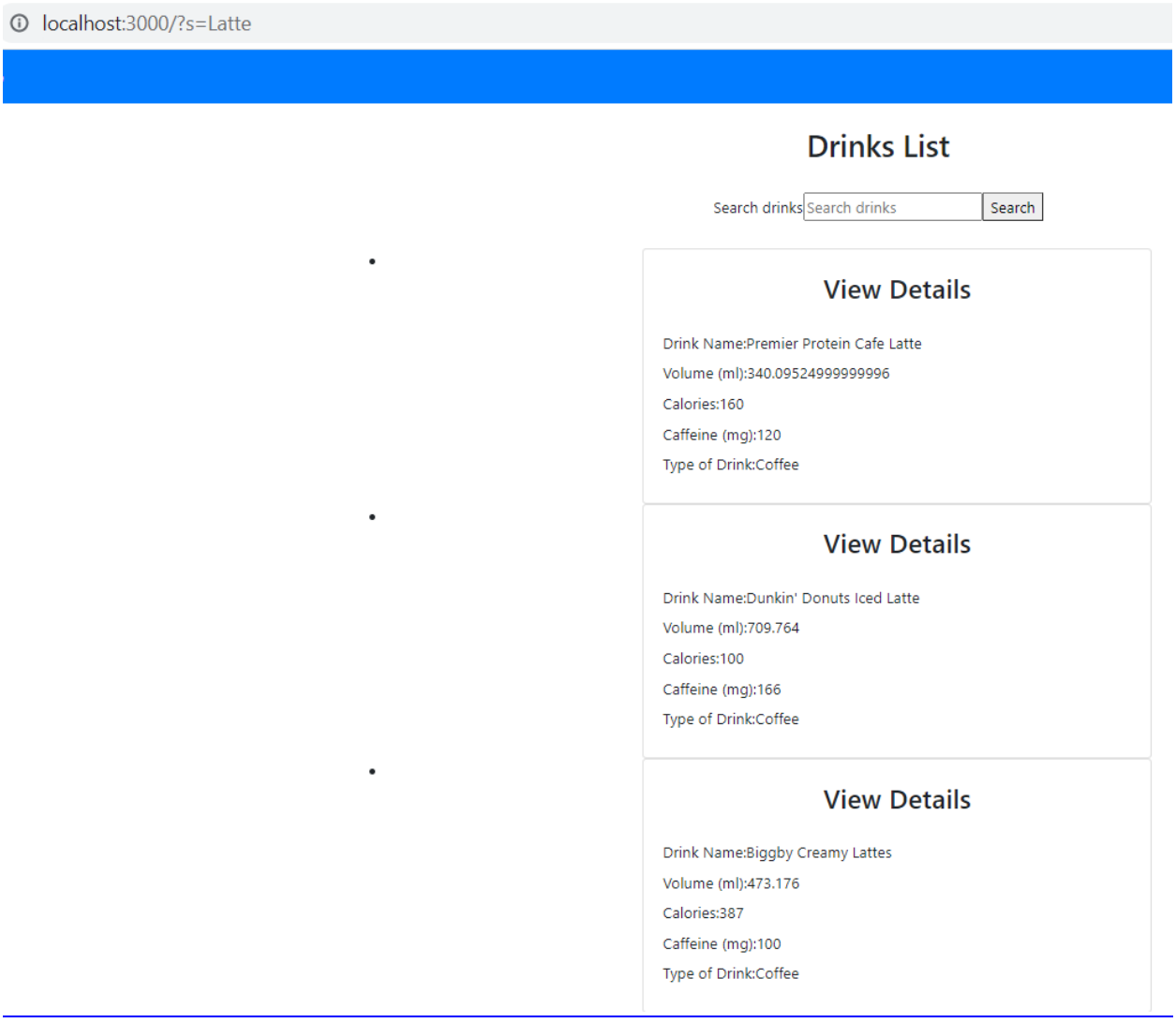
Drinks List

Search drinks Espresso Monster Search

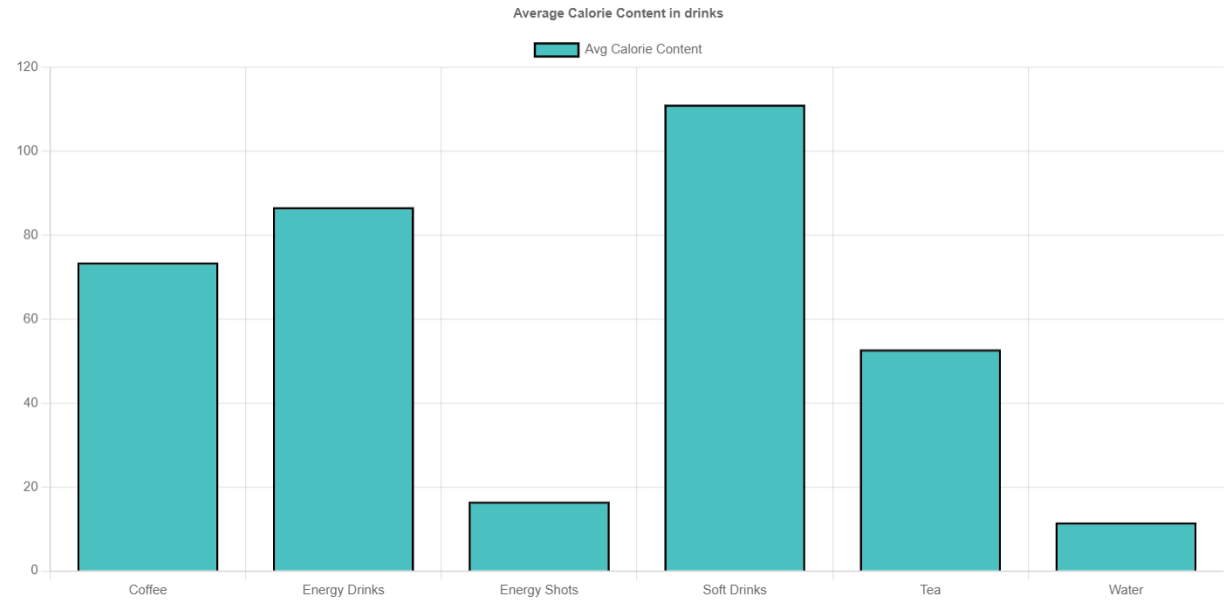
Add Drink Analytics

Drink Name	Volume (ml)	Calories	Caffeine(mg)	Type of Drink	Actions
Costa Coffee	256.99371499999995	0	277	Coffee	Update Delete View
Coffee Friend Brewed Coffee	250.19181000000003	0	145	Coffee	Update Delete View
Hell Energy Coffee	250.19181000000003	150	100	Coffee	Update Delete View
Killer Coffee (AU)	250.19181000000003	0	430	Coffee	Update Delete View
Nescafe Gold	250.19181000000003	0	66	Coffee	Update Delete View
Espresso Monster	248.41740000000001	170	160	Coffee	Update Delete View

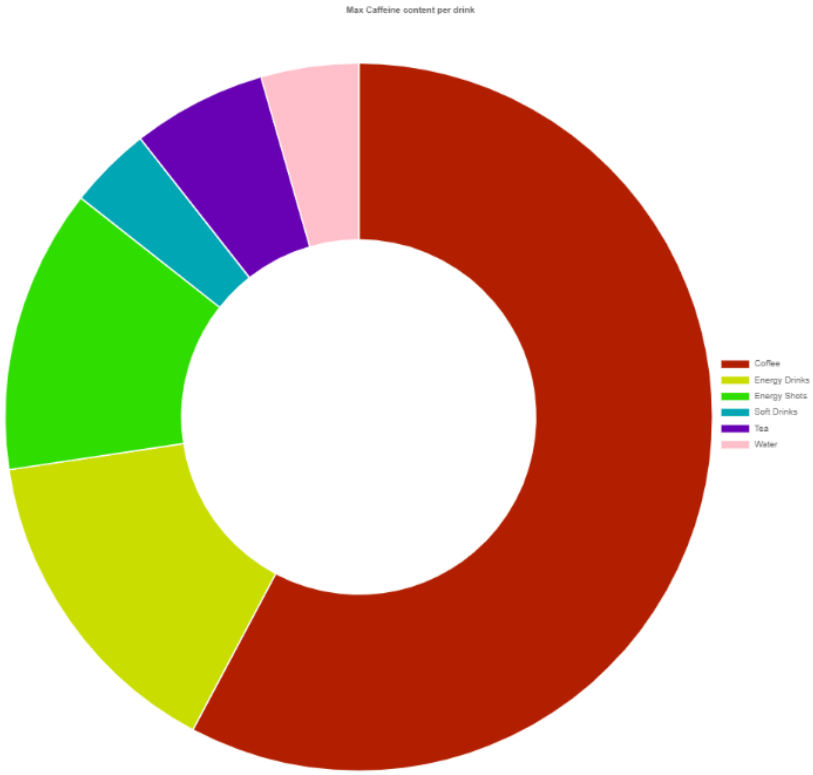

localhost:3000/?s=Espresso+Monster
Drinks List
Search drinks
Search drinks
Search
View Details
Drink Name:Espresso Monster
Volume (ml):248.41740000000001
Calories:170
Caffeine (mg):160
Type of Drink:Coffee


localhost:3000/?s=Latte
Drinks List
Search drinks
Search drinks
Search
View Details
Drink Name:Premier Protein Cafe Latte
Volume (ml):340.09524999999996
Calories:160
Caffeine (mg):120
Type of Drink:Coffee
View Details
Drink Name:Dunkin' Donuts Iced Latte
Volume (ml):709.764
Calories:100
Caffeine (mg):166
Type of Drink:Coffee
View Details
Drink Name:Biggby Creamy Lattes
Volume (ml):473.176
Calories:387
Caffeine (mg):100
Type of Drink:Coffee

Results/Graphs:



The above bar graph shows that Soft drinks has the highest calorie content.Followed by energy drinks and coffee.



The above pie chart shows that coffee has the highest caffeine content,followed by energy drinks and shots. Water has the Least caffeine content.

So from combining the above analysis it can be said that high consumption of coffee and energy drinks especially can have serious effects on a person's health.Whereas consuming Tea and water regularly might have less impact compared to all other types of drinks.

4. NEW LEARNINGS FROM THE PROGRAMMING ASSIGNMENT

- Had solid understanding of the basics of how MongoDB stores data.
- Connecting to MongoDB using PyMongo.
- Learned how to run queries against a MongoDB instance in order to store, manipulate, and retrieve data on it.
- How to filter for data efficiently.
- How to use React, JavaScript and native components and understood how to navigate in React Native apps.
- Build backend applications with Flask and Python and rest APIs development using Flask.
- Designing an analytical query and visualization using graphs.

5. FUTURE ENHANCEMENTS

We are able to search drinks based on the name of the drink in our case so in the future we can update the code to search based on type of drink, Calorie quantity and Caffeine quantity and also enable dynamic updates.

Since it is running locally now, we can also deploy the application of cloud so that many users can use this application for drawing insights and keeping their knowledge updated regarding drinks and there will be easy load balancing. We can also integrate machine learning models to perform classification, etc.