

18CSC303J-DATABASE MANAGEMENT SYSTEM

A PROJECT REPORT ON CALORIE-INTAKE BALANCED SYSTEM

Submitted by

JADHAV SUDHIR(RA1811003010300)

JUGAL P DUTTA(RA1811003010306)

Under the guidance of

Dr.Senthil kumar

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

of

FACULTY OF ENGINEERING AND TECHNOLOGY



SRM
INSTITUTE OF SCIENCE & TECHNOLOGY
Deemed to be University u/s 3 of UGC Act, 1956

**S.R.M. Nagar, Kattankulathur, Kancheepuram District
May 2021**

INTRODUCTION

1.1 INTRODUCTION TO CALORIE-INTAKE BALANCED SYSTEM

The population of the world is multiplying with each coming year and so are the diseases and health issues. With an increase in the population there is an increase in the need of maintaining proper balanced diet. The growing population of the world results in a lot of potential imbalanced diet. But in spite of this not more than 10% of the total world population participates in maintaining proper diet. With the growing population and the advancement in medical science the demand for keeping a check over calories has also increased. Due to the lack of knowledge on how much calorie-intake a person consumes per day, most of the patients in need of treatment do not get treatment on time and hence lose their lives. There is a dire need of synchronization between the daily calorie intake and the comparison with the required calorie intake prescribed by scientists. Improper communication and synchronization between the daily calorie intake and prescribed calorie intake leads to loss of the several lives. These problems can be dealt with by automating the existing manual calorie-intake management system. A high-end, efficient, highly available and scalable system has to be developed to bridge the gap between the calorie intake and the balanced diet and to reduce the required treatment for obesity and other heart risks.

- The project calorie-intake management system is known to be a pilot project that is designed for the every people to gather knowledge on how to maintain daily calorie intake from various sources and share it to the obesity or high calorie people who have high requirements for it.
- The software is designed to handle the daily transactions of the calorie intake and search the details when required.
- It also helps to register the details of person who come for check, daily calorie collection details as well as difference in calorie issued reports.
- The software application is designed in such a manner that it can suit the needs of all the people who wanted to keep a proper balanced diet requirements in the course of the future.

1.2 NEED FOR CALORIE INTAKE BALANCED SYSTEM

The management-balanced is ad-hoc with no semblance of organization or standard operating procedures. Donors cannot access your daily day-night calorie other than the system where you can keep your daily track. In the present system all the calorie intake records are attached to backend and there is no stand-alone record. Some hospital has its own systems and limitations. Because of the low number of proper calorie checking systems and more number of people with obesity, the required for proper balanced diet are avoided, resulting in thousands of loss of lives every day due to obesity which leads to various health risks. So to remove the priority continuing since long time on less awareness among people about proper balanced diet and calorie intake the system for calorie intake balanced system is designed.

1.3 AIM OF THIS PROJECT

The basic building aim is to keep check of every citizen calories and a proper balanced diet service from entire day food intake. Calorie Intake Balanced System (CIBS) is a Web-based application that is designed to store, process, retrieve and analyze information concerned with the administrative and inventory management within the citizen. This project aims at maintaining all the information pertaining to calorie checker, different preferred calorie intake for male and female available help them manage in a better way.

Also, project aim is to provide transparency in this field, make the process of checking calorie intake from a required calorie as mentioned and make the system of Calorie Intake management effective.

SPECIFICATIONS

2.1 HARDWARE REQUIREMENTS

- ☐ PROCESSOR : Intel Pentium or Higher Version
- ☐ RAM : Minimum 1GB
- ☐ HARD DISK : 60GB and above
- ☐

2.2 SOFTWARE REQUIREMENTS

- ☒ SOFTWARE : Xampp ,VS Code
- ☒ DATABASE : Xampp-MySQLdb-0.2.0
- ☐ SUPPORTED BROWSERS : Google Chrome / Mozilla Firefox / Internet Explorer
- ☐ EDITOR : Atom / Visual Studio Code
- ☒ FRAMEWORK : Node J.S
- ☒ OPERATING SYSTEM : Windows , or MACos, or Linux (32/64 bit)
- ☐ LANGUAGES USED :Mysql,Javascript,HTML/CSS,Bootstrap

2.3 FUNCTIONAL REQUIREMENTS

The Functional Requirements Specification documents the Operations and activities that a system must be able to perform. Functional Requirements include:

- Manage the information of customers.
- Manage the information of daily login and registration.
- Descriptions of operations performed by each system.
- Descriptions of work-flows performed by the system.
- To maintain storage area and data storage.
- Who can enter the data into the system.

The Functional Requirements Specifications is designed to be read by a general audience. Readers should understand the system, but no particular technical knowledge should be required to understand the document.

These are the functional requirements specification documents for the project analysis. A software requirement specification helps to attenuate the time and energy needed by the developers to attain their desired goals and additionally minimizes the value of development.

Following Factors are used to measure software development quality:

Each attribute may be accustomed measure of the product performance. These attributes may be used for Quality assurance similarly as quality control. Quality assurance activities are directed towards prevention of introduction of defects and internal control activities are aimed toward detecting defects in products and services.

1. Reliability

Measure if product is reliable enough to sustain in any condition. Give systematically correct results. Product dependability is measured in terms of operation of project underneath different operating atmosphere and different conditions.

2. Maintainability

Different versions of the product ought to be easy to maintain. For development it ought to be easy to feature code to existing system, ought to be easy to upgrade for brand new options and new technologies from time to time. Maintenance ought to be value effective and simple. System be easy to take care of and correcting defects or making a change within the software system.

3. Usability

This can be measured in terms of ease of use. Application should be user friendly. Easy to use for input preparation, operation and also for interpreting of output.

4. Portability

This can be measured in terms of Costing issues related to porting, Technical issues related to porting, Behavioural issues related to porting.

SYSTEM DESIGN

3.1 INTRODUCTION

System design is the first design stage for devising the basic approach to solving the problem. During system design, developers decide the overall structures and styles. The system architecture determines the organization of the system into subsystems. In addition, the architecture provides the context for the detailed decisions that are made in later stages. During design, developers make decisions about how the problem will be solved, first at the high level and then with more detail.

3.1.1 ARCHITECTURAL DIAGRAM

Any software should have a design structure of its functionality i.e. the architecture which defines about its inside view, likewise there is a database architecture in DBMS. The interaction of the database in DBMS with the system and the languages used in the database architecture is as shown in the below diagram and At the end of this article, you will be given a free pdf copy of Database Architecture in DBMS. The database architecture has three levels and they are as follows:

1. External level
2. Conceptual level
3. Internal level

3.1.2 DATA FLOW DIAGRAM

A Data Flow Diagram (DFD) is a graphical representation of the “flow” of data through an information system. Data Flow modules are used to show how data flows through a sequence of processing steps. The data is transformed at each step before moving on to the next stage. These processing steps or transformations are program functions when Data Flow Diagrams are used to document a software design.

Data flow modules are an intuitive way of showing how data is processed by a system. At the analysis level, they should be used to model the way in which data is processed in the existing system. The notation used in these modules represents functional processing, data stores and data movements between functions.

With a data flow diagram, users are able to visualize how the system will operate, what the system will accomplish and how the system will be implemented. Old system data flow diagrams can be drawn up and compared with the new system data flow.

These are several common modelling rules to be followed while creating DFD's are as follows:

- All processes must have at least one data flow in and one data flow out.
- All processes should modify the incoming data, producing a new form of outgoing data.
- Each data store must be involved with at least one data flow.
- Each external entity must be involved with at least one data flow.
- A data flow must be attached to at least one process.

3.1.3 USE CASE DIAGRAMS

The use case module is a catalogue of system functionality described using UML use cases. Each use case represents a single, repeatable interaction that a user or actor experiences when using the system. A use case typically includes one or more "scenarios" which describes the interactions that go on between the actor and the system, and documents the results and exceptions that occur from the user's perspective. Use case may include other use cases as part of a larger pattern of interaction and may also be extended by other use cases to handle exceptional conditions.

A use case is a coherent piece of functionality that a system can provide by interacting with actors. For example, a customer actor can buy a beverage from a vending machine. The customer inserts money into the machine, makes a selection, and ultimately receives a beverage. Similarly, a repair technician can perform scheduled maintenance on a vending machine.

IMPLEMENTATION

4.1 STAGE 1: INSTALLATION

Steps to install XAMPP on Windows

- During the installation process, select the required components like MySQL, FileZilla ftp server, PHP, phpMyAdmin or leave the default options and click the **Next** button.
- Uncheck the **Learn more about bitnami** option and click **Next** button.
- Choose the root directory path to set up the *htdocs* folder for our applications. For example '*C:\xampp*'.
- Click the **Allow access** button to allow the XAMPP modules from the Windows firewall.
- After the installation process, click the **Finish** button of the XAMPP Setup wizard.
- Now the XAMPP icon is clearly visible on the right side of start menu. Show or Hide can be set by using the control panel by clicking on the icon.
- To start Apache and MySQL, just click on the **Start** button on the control panel.
- **Note:** Suppose Apache is not starting, it means some other service is running at port **80**. In this case, stop the other service temporarily and restart it.
- **Making server request:** Open your web browser and check whether the XAMPP service has properly installed or not. Type in the URL: **http://localhost**. If you are able to see the default page for XAMPP, you have successfully installed your XAMPP Server.
- **To Check if PHP is Working:** All the website related files are organized in a folder called *htdocs* and then run *index.php* file by using **http://localhost/index.php** or **http://localhost**.
- **Note:** For every new website or application, its always better to create a different folder inside *htdocs*, to keep it organized and avoid confusion.
- For example, if we create a folder *geeksforgeeks* and then create a file named as '*helloWorld.php*'. All the contents related to it are put inside the folder '*geeksforgeeks*'. So the root 'URL' of the website will be '**http://localhost/geeksforgeeks/**'. So any home page is accessed by typing the root URL in the browser. To see the output, just type '**http://localhost/geeksforgeeks/helloWorld.php**'.
- Generally web servers look for index file (landing page) namely **index.html** or **index.php** in the root of the website folder. Go to */xampp/htdocs/* folder and create a file with **.php** extension (*test.php*) and type or copy the below code and save it.
- Now open your browser and go to "**http://localhost/test.php**" if you see the page same as
- below then PHP has successfully installed

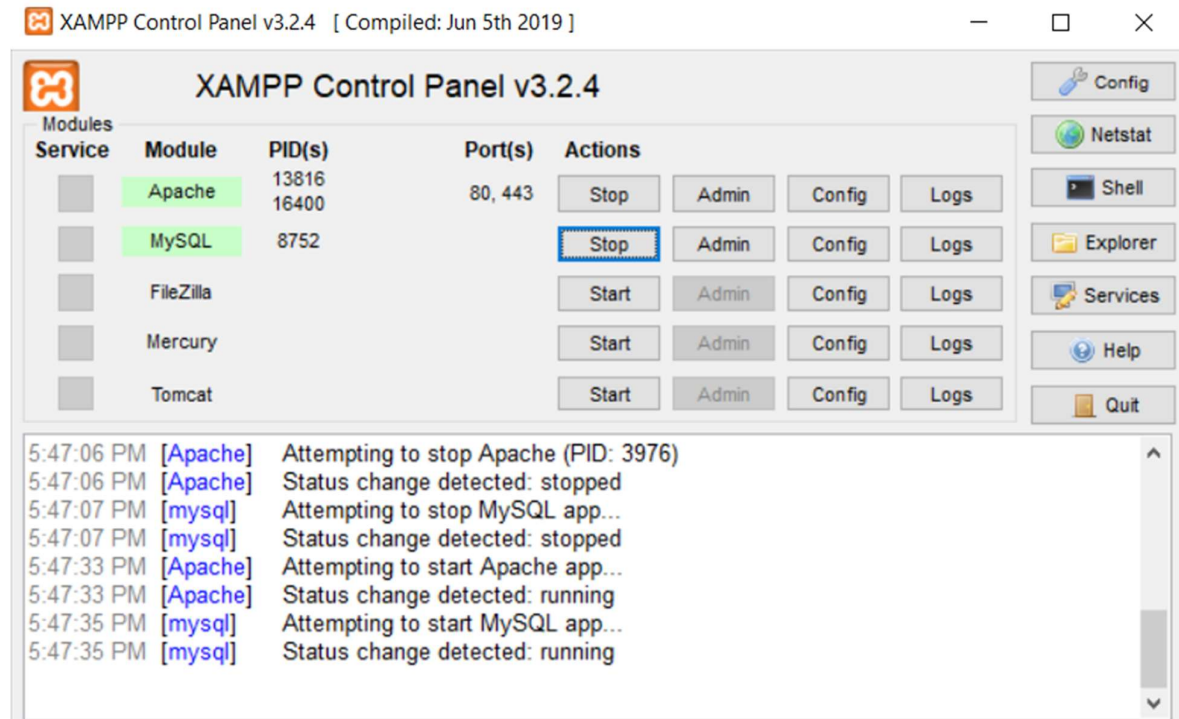
Install MySQL Server

Install the MySQL server by using the Ubuntu package manager.

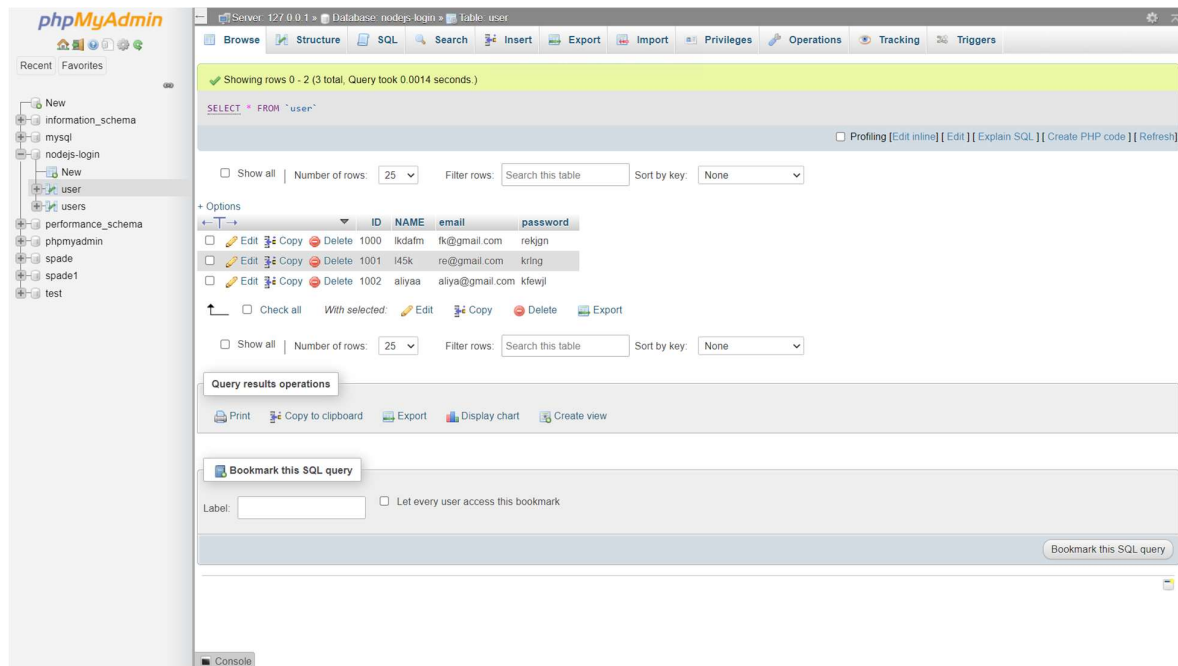
```
$ sudo apt-get install mysql-server
```

Setting up the database

- Open the xampp control panel and start the apache and MySQL module



- When you navigate to the <http://localhost/phpmyadmin/> site you will see database manipulation GUI



Here you can handle database operations manually like filtering the results, applying triggers, etc.

- When you create the new database using the left navigation panel finally, you can handle all the operations from the code. We can link database with our code using the following code

```
const db=mysql.createConnection({
  host:'localhost',
  user:'root',
  password:'',
  database:'nodejs-login'
});
```

The database name is 'nodejs-login' and user and password and credentials to be entered by the user.

4.2 STAGE 2: AUTHORIZATION

CODE:

1. Database interaction for register page

```
exports.register=(req,res)=>{
  const {name,email,password} = req.body;
  // console.log();
  db.query('select email from user where email=?',[email],async(error,result)=>{
    if(error)
      console.log(error);
    else if(result.length > 0){
      return res.render('register',{
        message:'user has already been registered.'
      });
    }
    else if(name!='' && email!='' && password!=''){
      db.query('insert into user set ?',{name:name,email:email,password:password}),(error,result)=>{
        if(error)
          console.log(error);
        else
          res.send(`Dear ${name} , you have been succesfully registered`);
      });
    }
  })
  else{
    return res.render('register',{
      message:'please enter valid information.'
    });
  }
};
```

2. Database interaction for login page

```
exports.login=(req,res)=>{
  const {email,password}=req.body;
  // console.log(req.body);
  db.query('select email from user where email=?',[email],async(error,result)=>{
    // console.log(email);
    if(result.length==0){
      res.render('login',{
        message:'please register'
      });
      console.log("lerjnjrn");
    }
    else{
      db.query('select name,password from user where email=?',[email] ,(err,re)=>{
        if(re[0].password!=password){
          res.render('login',{
            message:'please enter correct password'
          });
        }
        else{
          res.send('successfully logged in...');
        }
      });
    }
  });
};
```

3. Connecting to database

```
const db=mysql.createConnection({
  host:'localhost',
  user:'root',
  password:'',
  database:'nodejs-login'
});
```

4. Registration form html code

```
<section class='container card border-dark h-100'>
  <div class='card-header'>Register yourself..</div>
  <form action="auth/register" method="POST" class='text-dark'>
    <div class="mb-3">
      <label for="name" class="form-label">Name</label>
      <input type="name" class="form-control" id="name" name="name">
    </div>
    <div class="mb-3">
      <label for="email" class="form-label">Email address</label>
      <input type="email" class="form-control" id="email" name="email">
      <div id="emailHelp" class="form-text">We'll never share your email with anyone else.</div>
    </div>
    <div class="mb-3">
      <label for="password" class="form-label">Password</label>
      <input type="password" class="form-control" id="password" name="password">
    </div>
    <button type="submit" class="btn btn-primary">Submit</button>
  </form>
  {{#if message}}
  <h4 class="alert alert-danger mt-4">{{message}}</h4>
  {{/if}}
</section>
```

5. login page html code

```
<section class='container card border-dark h-100'>
  <div class='card-header'>Log in if you are already registered</div>
  <form action="auth/login" method="POST" class='card-body text-dark'>
    <div class="mb-3">
      <label for="exampleInputEmail1" class="form-label">Email address</label>
      <input type="email" class="form-control" id="exampleInputEmail1" name="email">
      <div id="emailHelp" class="form-text">We'll never share your email with anyone else.</div>
    </div>
    <div class="mb-3">
      <label for="exampleInputPassword1" class="form-label">Password</label>
      <input type="password" class="form-control" id="exampleInputPassword1" name="password">
    </div>
    <button type="submit" class="btn btn-primary">Submit</button>
  </form>
  {{#if message}}
  <h4 class="alert alert-danger mt-4">{{message}}</h4>
  {{/if}}
</section>
```

6. routing between the pages

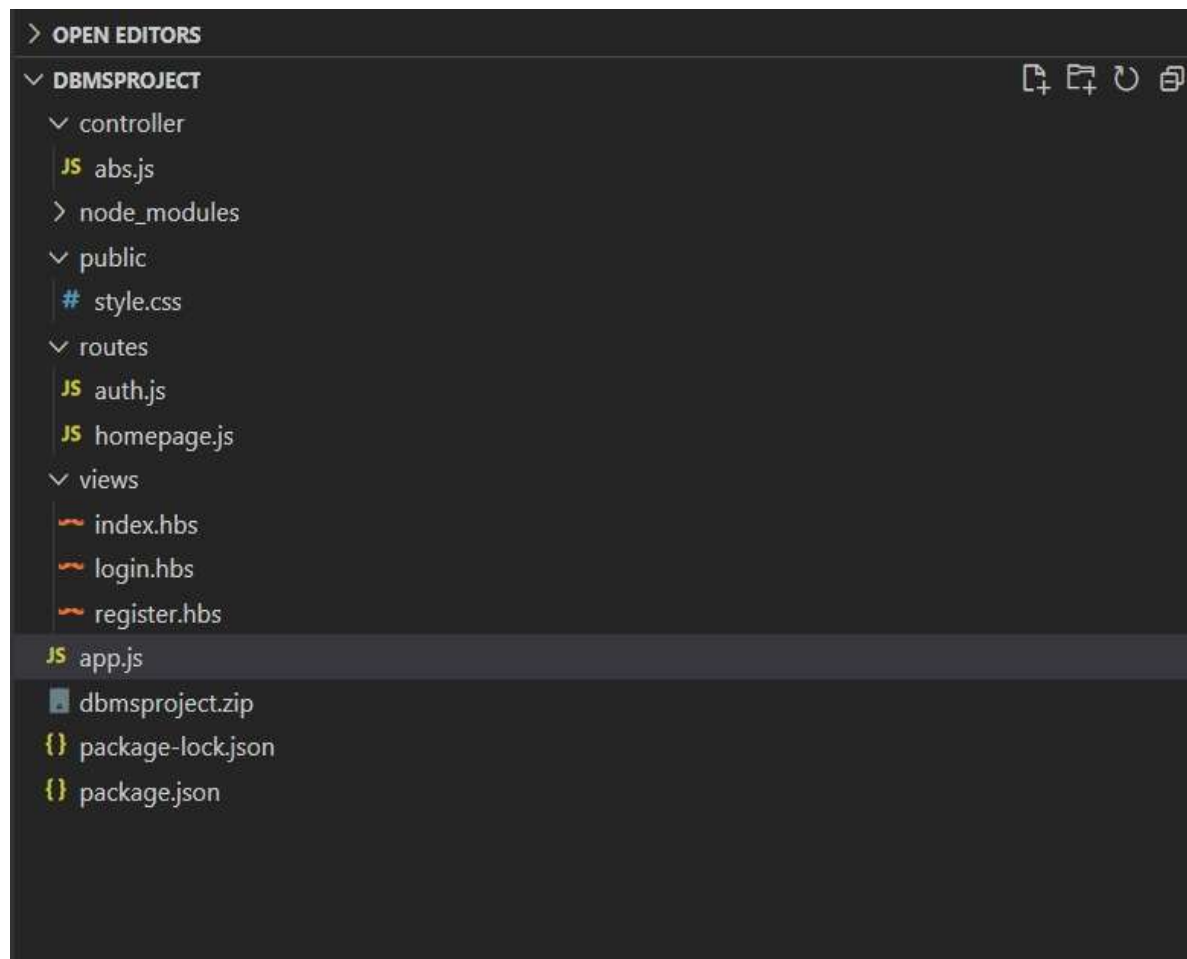
```
// const { Router } = require('express');
const express=require('express');

const router=express.Router();

router.get("/",(req,res)=>{
  res.render('index');
});
router.get("/login",(req,res)=>{
  res.render('login');
});
router.get("/register",(req,res)=>{
  res.render('register');
});

module.exports=router;
```

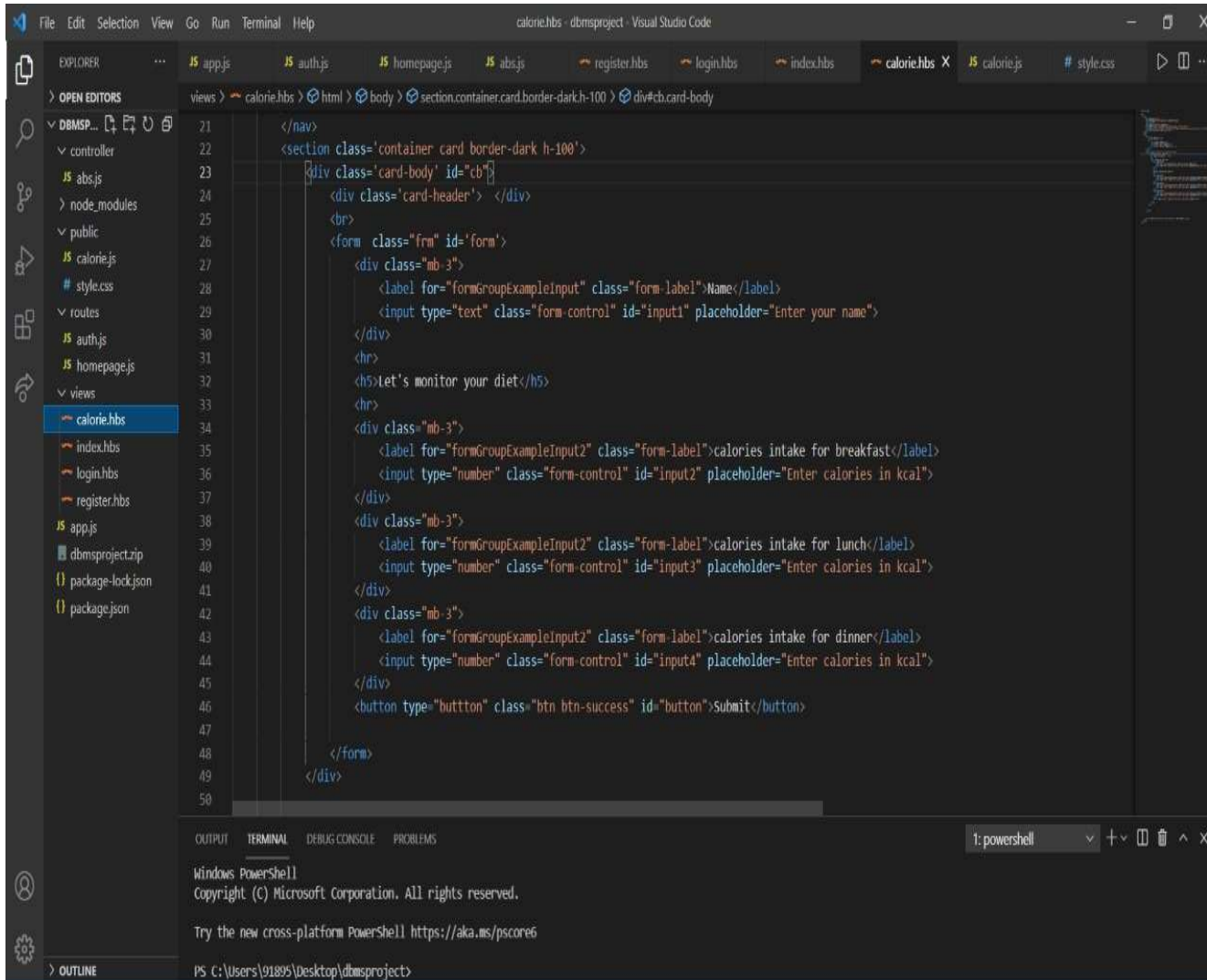
7. Bunch files involved in implementing the code



4.3 STAGE 3: CALORIE-CHECK

CODE:

Javascript for calorie intake



The screenshot shows the Visual Studio Code editor with the file 'calorie.hbs' open. The file contains HTML code for a calorie intake form. The form has three sections for breakfast, lunch, and dinner, each with a label, a text input field, and a submit button. The terminal window at the bottom shows the Windows PowerShell prompt.

```
21 </nav>
22 <section class='container card border-dark h-100'>
23   <div class='card-body' id='cb'>
24     <div class='card-header'> </div>
25     <br>
26     <form class='frm' id='form'>
27       <div class='mb-3'>
28         <label for='formGroupExampleInput' class='form-label'>Name</label>
29         <input type='text' class='form-control' id='input1' placeholder='Enter your name'>
30       </div>
31       <hr>
32       <h5>Let's monitor your diet</h5>
33       <hr>
34       <div class='mb-3'>
35         <label for='formGroupExampleInput2' class='form-label'>calories intake for breakfast</label>
36         <input type='number' class='form-control' id='input2' placeholder='Enter calories in kcal'>
37       </div>
38       <div class='mb-3'>
39         <label for='formGroupExampleInput2' class='form-label'>calories intake for lunch</label>
40         <input type='number' class='form-control' id='input3' placeholder='Enter calories in kcal'>
41       </div>
42       <div class='mb-3'>
43         <label for='formGroupExampleInput2' class='form-label'>calories intake for dinner</label>
44         <input type='number' class='form-control' id='input4' placeholder='Enter calories in kcal'>
45       </div>
46       <button type='button' class='btn btn-success' id='button'>Submit</button>
47     </form>
48   </div>
49 </section>
50 </div>
```

OUTPUT TERMINAL DEBUG CONSOLE PROBLEMS

1: powershell

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell <https://aka.ms/pscore6>

PS C:\Users\91895\Desktop\dbmsproject>


```
File Edit Selection View Go Run Terminal Help
calorie.js - dbmsproject - Visual Studio Code

EXPLORER
> OPEN EDITORS
  DBMSP...
  > controller
    JS abs.js
  > node_modules
  > public
    JS calorie.js
    # style.css
  > routes
    JS auth.js
    JS homepage.js
  > views
    calorie.hbs
    index.hbs
    login.hbs
    register.hbs
  JS app.js
  dbmsproject.zip
  package-lock.json
  package.json

public > JS calorie.js > runevent2
24     name1=e.target.value;
25     // console.log(name1);
26   }
27
28   function runevent2(e){
29     e.preventDefault();
30     // console.log("ok");
31     console.log(e.target.value);
32     total=total+parseInt(e.target.value);
33   }
34
35   function runevent3(e){
36     e.preventDefault();
37     // console.log("lol");
38     const card = document.getElementById('cb');
39     document.getElementById('form').style.display='none';
40
41     var n=document.createElement('div');
42     var act=2200-total;
43     var res=`${name1}, your calorie intake is ${total}. Average Indian has an intake of 2200 kcal,you should ramp up your diet with ${act} kcal. Sta
44     var nt=document.createTextNode(res);
45     // n.classList.add('container','card', 'border-dark', 'h-100');
46     n.className='mb-3';
47     n.classList.add('h3');
48     n.appendChild(nt);
49     card.insertBefore(n,form);
50     console.log(name1);
51     console.log(total);
52   }

OUTPUT TERMINAL DEBUG CONSOLE PROBLEMS
1: powershell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/powershell

PS C:\Users\91895\Desktop> dbmsproject>
```

```
File Edit Selection View Go Run Terminal Help
calorie.js - dbmsproject - Visual Studio Code

EXPLORER
> OPEN EDITORS
  DBMSP...
  > controller
    JS abs.js
  > node_modules
  > public
    JS calorie.js
    # style.css
  > routes
    JS auth.js
    JS homepage.js
  > views
    calorie.hbs
    index.hbs
    login.hbs
    register.hbs
  JS app.js
  dbmsproject.zip
  package-lock.json
  package.json

public > JS calorie.js > runevent2
1   const Name=document.getElementById('input1');
2   Name.addEventListener('keyup',runevent);
3
4
5   const input2=document.getElementById('input2');
6   const input3=document.getElementById('input3');
7   const input4=document.getElementById('input4');
8
9   // input1.addEventListener('keyup',runevent);
10  input2.addEventListener('mouseleave',runevent2);
11  input3.addEventListener('mouseleave',runevent2);
12  input4.addEventListener('mouseleave',runevent2);
13  // input3.addEventListener('keyup',runevent);
14
15  const submit=document.getElementById('button');
16  submit.addEventListener('click',runevent3);
17
18
19
20  let total = 0;
21  let name1;
22  function runevent(e){
23    e.preventDefault();
24    name1=e.target.value;
25    // console.log(name1);
26  }
27
28  function runevent2(e){
29    e.preventDefault();
30    // console.log("ok");
31  }
32
33  function runevent3(e){
34    e.preventDefault();
35    // console.log("lol");
36    const card = document.getElementById('cb');
37    document.getElementById('form').style.display='none';
38
39    var n=document.createElement('div');
40    var act=2200-total;
41    var res=`${name1}, your calorie intake is ${total}. Average Indian has an intake of 2200 kcal,you should ramp up your diet with ${act} kcal. Sta
42    var nt=document.createTextNode(res);
43    // n.classList.add('container','card', 'border-dark', 'h-100');
44    n.className='mb-3';
45    n.classList.add('h3');
46    n.appendChild(nt);
47    card.insertBefore(n,form);
48    console.log(name1);
49    console.log(total);
50  }

OUTPUT TERMINAL DEBUG CONSOLE PROBLEMS
1: powershell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/powershell

PS C:\Users\91895\Desktop> dbmsproject>
```


4.4 STAGE 4: VIEW DATA

✓ Showing rows 0 - 2 (3 total, Query took 0.0013 seconds.)

```
SELECT * FROM `user`
```

| ID | NAME | email | password |
|------|--------|-----------------|----------|
| 1000 | lkdafm | fk@gmail.com | rekjgn |
| 1001 | l45k | re@gmail.com | krlng |
| 1002 | aliyaa | aliya@gmail.com | kfewjl |

[Back](#)[Print](#)

| # | Name | Type | Collation | Attributes | Null | Default | Comments | Extra |
|---|----------|--------------|--------------------|------------|------|---------|----------|----------------|
| 1 | ID | int(11) | | | No | None | | AUTO_INCREMENT |
| 2 | NAME | varchar(50) | utf8mb4_general_ci | | No | None | | |
| 3 | email | varchar(50) | utf8mb4_general_ci | | No | None | | |
| 4 | password | varchar(150) | utf8mb4_general_ci | | No | None | | |

Indexes

| Keyname | Type | Unique | Packed | Column | Cardinality | Collation | Null | Comment |
|---------|-------|--------|--------|--------|-------------|-----------|------|---------|
| PRIMARY | BTREE | Yes | No | ID | 2 | A | No | |


Partitions

⚠ No partitioning defined!

Information

| | | | |
|-------------|----------|----------------|--------------------------|
| Data | 16.0 KiB | Format | dynamic |
| Index | 0 B | Collation | utf8mb4_general_ci |
| Overhead | 0 B | Next autoindex | 1,003 |
| Effective | 16.0 KiB | Creation | May 01, 2021 at 06:48 PM |
| Total | 16.0 KiB | Last update | May 28, 2021 at 11:45 AM |
| Space usage | | Last check | May 28, 2021 at 11:45 AM |
| | | Row statistics | |

[Back](#) [Print](#)

| # | Name | Type | Collation | Attributes | Null | Default | Comments | Extra |
|---|--|--------------|--------------------|------------|------|---------|----------|----------------|
| 1 | ID  | int(11) | | | No | None | | AUTO_INCREMENT |
| 2 | NAME | varchar(50) | utf8mb4_general_ci | | No | None | | |
| 3 | email | varchar(50) | utf8mb4_general_ci | | No | None | | |
| 4 | password | varchar(150) | utf8mb4_general_ci | | No | None | | |

Indexes

| Keyname | Type | Unique | Packed | Column | Cardinality | Collation | Null | Comment |
|---------|-------|--------|--------|--------|-------------|-----------|------|---------|
| PRIMARY | BTREE | Yes | No | ID | 2 | A | No | |

Information

| | | |
|-----------|------|-----|
| Data | 16.0 | KiB |
| Index | 0 | B |
| Overhead | 0 | B |
| Effective | 16.0 | KiB |
| Total | 16.0 | KiB |

Space usage

| | |
|----------------|--------------------------|
| Format | dynamic |
| Collation | utf8mb4_general_ci |
| Next autoindex | 1,003 |
| Creation | May 01, 2021 at 06:48 PM |
| Last update | May 28, 2021 at 11:49 AM |
| Last check | May 28, 2021 at 11:49 AM |

Row statistics

RESULTS AND SNAPSHOTS

Register yourself..

Name

Email address

We'll never share your email with anyone else.

Password

Submit

Log in if you are already registered

Email address

We'll never share your email with anyone else.

Password

Submit

OUTPRODUCT:

SAMPLE:1

Name

Riya

Let's monitor your diet

calories intake for breakfast

190

calories intake for lunch

230

calories intake for dinner

190

Submit

Riya, your calorie intake is 610. Average Indian has an intake of 2200 kcal, you should ramp up your diet with 1590 kcal. Stay healthy 😊

SAMPLE:2

Name

Aliya

Let's monitor your diet

calories intake for breakfast

450

calories intake for lunch

300

calories intake for dinner

100

Submit

Aliya, your calorie intake is 850. Average Indian has an intake of 2200 kcal, you should ramp up your diet with 1350 kcal. Stay healthy 😊

CONCLUSION

Technology is introducing new innovations day by day, thus reducing the time required to do things. The proposed system can be used to reduce the time required to deliver required blood to the needy in cases of emergency. The web application provides a way of communication synchronization between the daily calorie intake and the comparison with the required calorie intake prescribed by scientists. The database is a vital aspect of the system. The database then gives an comparable outcome from the synchronization and suggest to be health cautious if required.

BIBLIOGRAPHY

BOOKS

- [1] Python: For Beginners A Crash Course Guide To Learn Python in 1 Week by Timothy C. Needham
- [2] MySQL: The Complete Reference by Vikram Vaswani
- [3] Flask Framework Cookbook by Shalabh Aggarwal
- [4] Essentials of Blood Banking by Mehdi S.R.

LINKS

- [1] <https://docs.python.org/3/tutorial/>
- [2] <http://www.mysqltutorial.org/>
- [3] <https://www.javatpoint.com/mysql-tutorial>
- [4] <https://www.coursera.org/learn/python-databases>
- [5] [h https://www.w3schools.com/nodejs/](https://www.w3schools.com/nodejs/)
- [6] <https://www.google.com/>

