Advance web and mobile technologies

My Diet App

Introduction

My diet app is social network where you can save your meals for calories counting asking question and upload posts.

My diet app clients are the company that provide diet consultation that want to give there customers interface were every customer can easily manage his calorie consumption and asking the anther customer questions or talk about difficulties about the diet in purpose to make the diet process easier.

To use My diet app you need to be register user of the system and computer with internet connection.

And email address for the registration.

Product perspective

1. Client Interface:

The client application will run over browser as web application.

The browser must support protocol HTTP.

2. User Interface:

The user will use the web application in order to use all the functionalities of the system.

The web application will be written in HTML5, CSS3 and JavaScript.

3. Server Interface:

The server interface will handle all the client’s requests.

4. DB Interface:

The DB interface will handle all the queries from the server and will be

responsible for saving all the data of the system.

Product functions

* Insert new product for meal or new meal

The client will have the ability to insert new product for meal or new meal

For the calorie counting, the user will insert the product (rice ,salad etc.) the server will calculate the total calorie count for this dish, and the total for this day and mortify the user about it.

* Get the total amount of calories for specific day

The user will have the ability the check how many calories he eat in a specific day

Searching by date

* Ask a question or tell a story at the app blog

The user have the ability to post a status in the public blog and read all the comments that anther user post about it .

* Write a comment for exist post in the blog

The user have the ability to post a comment about anther user or is own status.

* Update the personal user details

The user can update details like current weight target weight total calories allowed per day

Age etc.

* Get product amount of calorie and unit measure

The user can check how many calories he have in different type of food so he can plan his meals better.

* Remove account

The user have the ability to remove himself from the service .

|  |  |
| --- | --- |
| Primary Actor | Use Cases |
| User | 1.Insert new product for meal or new meal.  2.Get the total amount of calories for specific day.  3.Ask a question or tell a story at the app blog.  4.Write a comment for exist post in the blog.  5.Update the personal user details.  6.Get product amount of calorie and unit measure.  7.Remove account. |

|  |  |
| --- | --- |
| Use case ID: | 1 |
| Use Case Name: | Insert new product for meal or new meal. |
| Actors: | User |
| Description: | Insert new product for meal |
| Preconditions: | User logged into the system |
| Normal Flow: | 1. The user logs-in to the system.  2. The server gets the request.  3. the server calculate all the received parameters  4.The server update the DB  5. The customer get success message with all the details |
| Exceptions: | 1. The Server is down.  1.1 Client will notify the user.  2. The DB is unreachable.  2.1 Client will notify the user. |
| Notes and Issues: | None. |

|  |  |
| --- | --- |
| Use case ID: | 2 |
| Use Case Name: | Get the total amount of calories for specific day. |
| Actors: | User |
| Description: | The user what to know how many calories he consumed in specific day |
| Preconditions: | User Logged into the system |
| Normal Flow: | 1. The user logs-in to the system and ask for calories count for specific day.  2. The server gets the request with date that user want to check.  3. the server send query to the DB  5. The customer get success message with all the details |
| Exceptions: | 1. The Server is down.  1.1 Client will notify the user.  2. The DB is unreachable.  2.1 Client will notify the user. |
| Notes and Issues: | If the user ask for information that he don’t have data about it(date in the future or date he wasn’t register) he will get empty response |

|  |  |
| --- | --- |
| Use case ID: | 3 |
| Use Case Name: | Ask a question or tell a story at the app blog |
| Actors: | User. |
| Description: | The user want to post a new status on the public wall |
| Preconditions: | User Logged into the system |
| Normal Flow: | 1. The User logs-in to the system and Post new Status.  2. The server gets the request with the new status.  3. the server send the post and send insert query to the DB.  4.the DB store it. |
| Exceptions: | 1. The Server is down.  1.1 Client will notify the user.  2. The DB is unreachable.  2.1 Client will notify the user. |
| Notes and Issues: | None. |

|  |  |
| --- | --- |
| Use case ID: | 4 |
| Use Case Name: | Write a comment for exist post in the blog. |
| Actors: | User |
| Description: | Write a comment for exist post in the blog. |
| Preconditions: | User Logged into the system |
| Normal Flow: | 1. The user select the post he want to post a comment and write the comment and send it 2. The server the comment and the post he want to post about 3. The server sent request for insertion to the DB 4. The DB store the comment |
| Exceptions: | 1. The Server is down.  1.1 Client will notify the user.  2. The DB is unreachable.  2.1 Client will notify the user. |
| Notes and Issues: | None. |

|  |  |
| --- | --- |
| Use case ID: | 5 |
| Use Case Name: | Update the personal user details. |
| Actors: | User |
| Description: | The user want to update his details |
| Preconditions: | User Logged into the system |
| Normal Flow: | 1. The user insert is card 2. The user choose the details he want to update and send it to the server with the new details. 3. The server get the requst and sent it to the DB 4. The DB update the new details |
| Exceptions: | 1. The Server is down.  1.1 Client will notify the user.  2. The DB is unreachable.  2.1 Client will notify the user. |
| Notes and Issues: | The user cant update the email and password the user Id |

|  |  |
| --- | --- |
| Use case ID: | 6 |
| Use Case Name: | Get product amount of calorie and unit measure. |
| Actors: | User |
| Description: | The user want to check the amount of calories for specific food |
| Preconditions: | User Logged into the system |
| Normal Flow: | 1. The user send request to the server with the food he want to check 2. The server sent the query to the DB 3. The DB return to the server the calorie amount and the unit measure . 4. The server return it to the user |
| Exceptions: | 1. The Server is down.  1.1 Client will notify the user.  2. The DB is unreachable.  2.1 Client will notify the user. |
| Notes and Issues: | All the food types exist in the DB |

|  |  |
| --- | --- |
| Use case ID: | 7 |
| Use Case Name: | Remove account. |
| Actors: | User |
| Description: | User want to remove his account form the sevice |
| Preconditions: | User Logged into the system |
| Normal Flow: | 1. The user insert to his card and press delete. 2. The server get the request. 3. The server send delete query to the DB 4. The DB remove the record 5. The server disconnect the user from the server |
| Exceptions: | 1. The Server is down.  1.1 Client will notify the user.  2. The DB is unreachable.  2.1 Client will notify the user. |
| Notes and Issues: | If the user ask to remove himself he cant restore previous data. |

DB schema

In this table we hold all the user and the personal data

Users

{

\_id // auto id gevin by mongo

user\_name // the user name

Mail // the email for registration in login

Password // encrypted password for login

Address { // the address of the user

street

city

state

country

}

diet\_start\_date // date the user register to the service

target\_calories\_for\_day // it’s the amount of calorie the user decide he want to eat

start\_weight // the weight the user was when he register to the service

target\_weight // the weight the user want to be

current\_weight // the last weight the user can edit it every time he want

Last\_wieght\_date // the date the user insert the last weight

Height // user height

Sex // user sex

Age // user age

}

// all the Products the user can choose from to insert to his meal this table save all the data all the foods // type

Products

{

\_id // id from mongo

products\_name // the product name

unit\_measure // the unit that this product measure for example gr,ml,teaspoon,tablespoon

calorie\_per\_unit\_measure // the amount of calorie per one unit measure

description // long description of the product

}

In this table we hold all the data the user insert the whole meal to just the ingredient of the meal

meal\_product

{

\_id // id from mongo

Date // the date that we need to calculate the calories for

meal\_number // user can insert how many ingredient he want tot the same meal

user\_ID // the connection to the user table

user\_name // the user name

products\_id // the connection to the products table

products\_name // the product name

calorie\_per\_Unit\_measure // the amount of calorie copy from product table

amount // the amount the user consume

total\_calories // the total calorie for this product amount \* calorie\_per\_unit\_measure

}

// this table is to hold all the topics that user post in the public blog

topics

{

\_id // id form mongo

user\_id // connecition to the user table the user that post the topic

user\_name // the user name

Title // the title of the topic

topic\_text // the topic text

date // the date of the post posted

}

// this table hold all the comment for the selected topic

comments

{

\_id // id from mongo

topic\_id // the topic that the comment refers to

user\_id // connection to the user table the user that post the comment

user\_name // the user mane

comment\_text // comment text

date // the date the topic posted

}

The top quires we expected in this app is mostly insert meals check the amount of calorie per a day

And post a topic and comments and some time update the current weight for user

In the DB schema we try to insert all the relevant data of the top querys to one table to avoid the necessaire to join two tables.