DCUMunch

"More than just food."

CA326 Technical Specification:

Genesis Uwumangbe -20459666

Joseph Adedayo - 20303853

Table of contents page

1. Introduction

1.1 Overview

DCU much is a food web application that focuses on DCU’s catering team (Nubar, Cafeteria, and the Business Cafe). This application was developed to make it easier for students to pick between meals and quickly know the meals that will match their dietary requirements.

Why? Both of us realised that the catering team in DCU don't show the calories per meal and all the ingredients in each meal which isn't ideal for those who like to watch their diet. We also created a survey for DCU students to see if the idea of the web application will suit them. The majority agreed that it'll be a good idea.

Our goal is to provide students with a fun and enjoyable method of finding **affordable** and **healthy** food based on their dietary requirements and improving a students health and well-being.

1.2 Glossary

Database - A structured set of data held in a computer

Edamam -A food API that provides the calories for meals

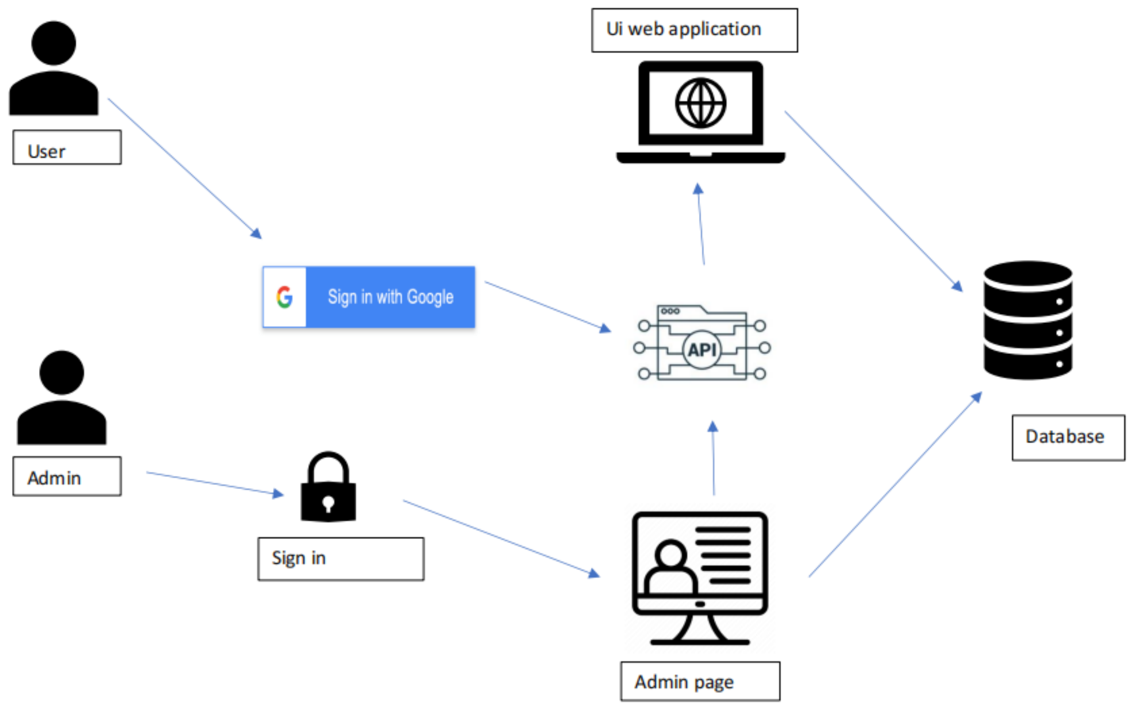
Integration testing - testing if individual software modules work together as a group

Git CI - Git continuous integration. It builds and tests software.

2.0 System Architecture

This section describes the high-level overview of the system architecture showing the distribution functions across (potential) system modules. Architectural components that are reused or 3rd party should be highlighted.

Operational Overview



2.1 User

In this diagram we illustrate how the user and admin interact with our system. Firstly the users (Dcu students) sign in with google. They will then be greeted with a page asking for their dietary requirements and fitness level. Our app will then recommend a calorie estimate for the user for the day. This data is collected and fed to our database. Users will be recommended different meals that will relate to what they entered in for their dietary requirements this is done by our feed algorithm called the munch algorithm.

2.2 Admin

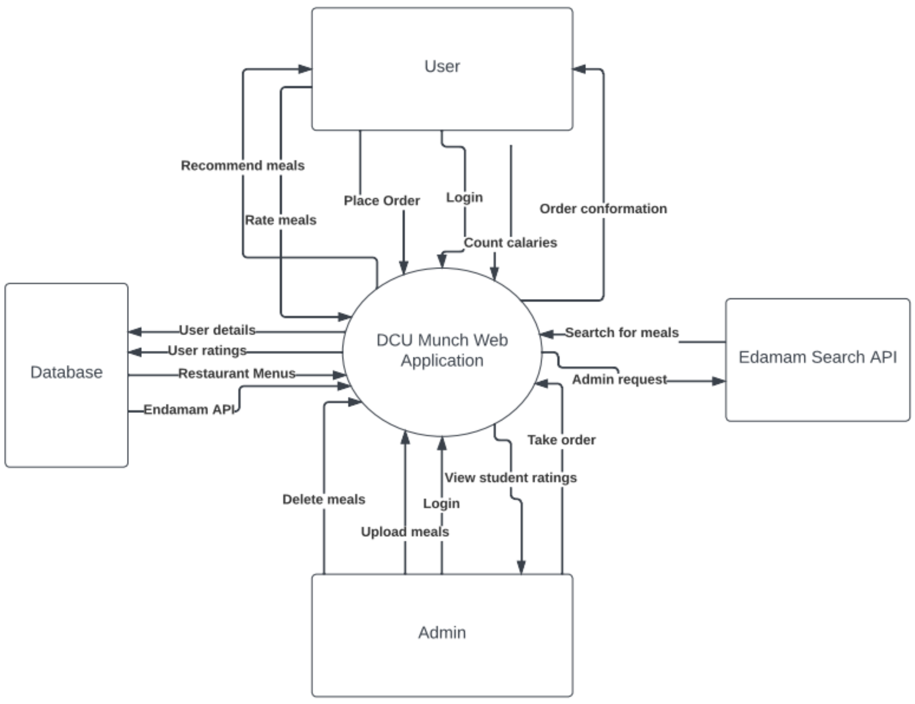
When the admin signs into their section they are greeted with information about the meals the have added to the database. The current system architecture has stayed in line with the proposed system architecture written in our functional specification. We included the Edamam API, this API is used to fetch meals from the internet that the catering team might not have time to manually add in themselves it collects the calories and all the dietary requirements. This API will then display the content on the Ui web application Admins can add their meals to the database which will be then shown to the users. They can also view meals that students like the most/ take orders from the web app.

3. High-Level Design

In this section, we will discuss the high-level design of the system. It will include system models showing the relationship between system components and the systems and its environment.

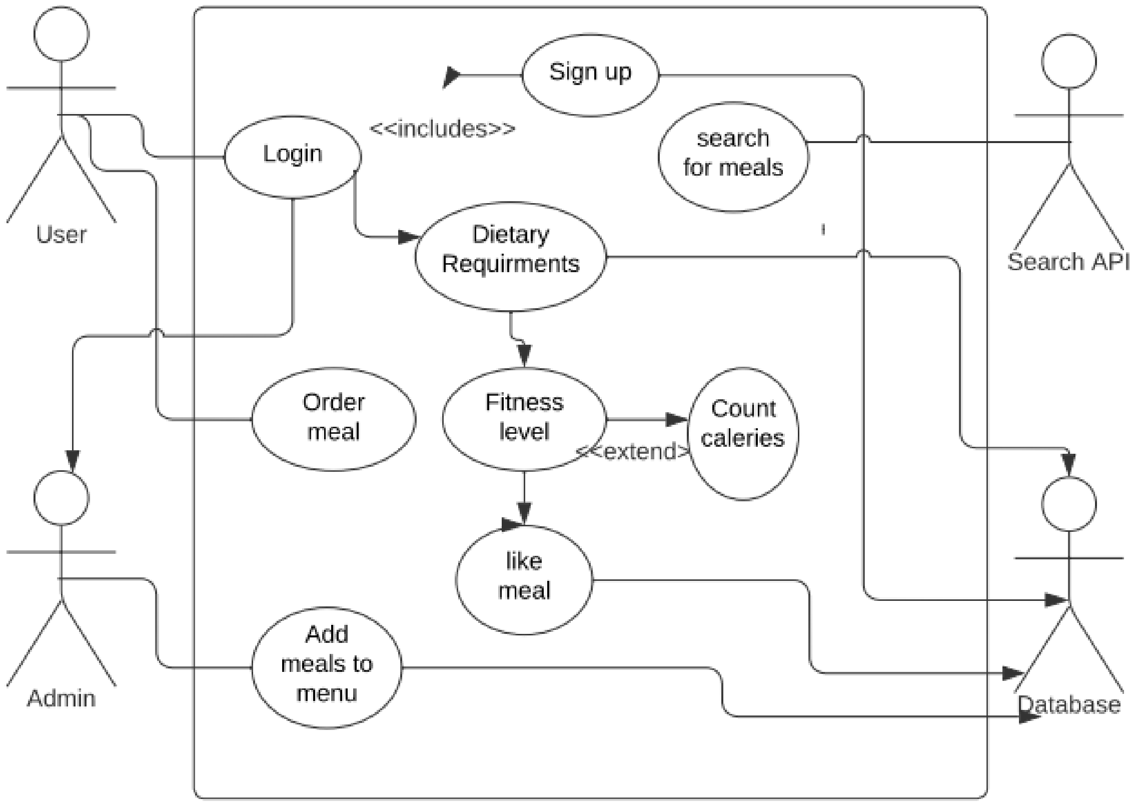
3.1 Context Diagram

The diagram below outlines how the external entities interact with an internal software system



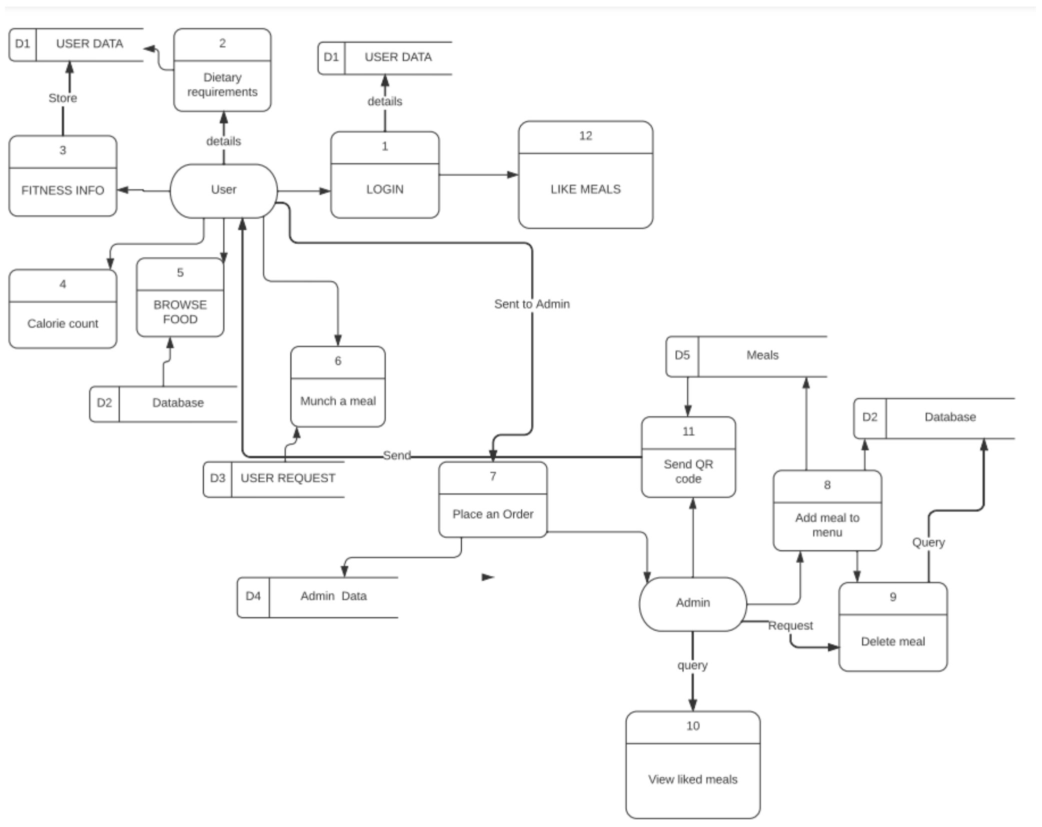
3.2 Use case diagram

Below is our use case diagram, it shows the high level and scope of our system. The diagram identifies the relationship between the users and the system.



3.3 Data flow Diagram

This diagram maps out the flow of information for our system. It showcases it from the User to the Admin.



4. Problems and Resolution

This section should include a description of any major problems encountered during the design and implementation of the system and the actions that were taken to resolve them.

4.1 Paying through the website

Problem

Our initial plan was for once the users have made their order they can pay with a card through the website and receive a QR code to scan at the food outlets to pick up the food when it is ready. However, coming close to our deadline we were looking to implement this feature and saw that this world needed some verification and our website would need to be on a server in order for it to work.

Solution

To resolve this issue we decided that we would scrap the idea of card payments as it would take a lot of time to complete. It is also something that we will look at in the future. Alternatively, users can place an order and receive a QR code that they will then scan at the restaurant and pay there.

4.2 Basket branch had issues with APIUser

Problem

When working on adding an item to the shopping basket we came across a long error which was saying that the API user was invalid. This was due to us having different apps such as the orders app and dcumunch app.

Solution

To resolve this issue we deleted the 'basket' branch that had the error code in it because we could not locate the error in our code. We created a new branch and place all our code into the same app. We also changed the API user as we already had it defined.

4.3 Admin adding meals to the menu manually

Problem

After speaking to our supervisor we realised that it would be very time-consuming for the admin to enter every meal into our database. Things could change with certain meals or just having to add in each ingredient, and calorie intake for the meal would take time.

Solution

We implemented a search API into the admin section. This search API is called the Edamam API. It allows the admin to search up any meal online and add it to their menu. It also comes with a calorie count, protein, fats etc intake. We believe that this feature will make life easier for the admins to upload their meals.

4.4 Deploying on the School webserver

Problem

We had the idea to deploy our app on the school web server. But we couldn't go ahead with this because we didn't start our project in a virtual environment.

Solution

Our project will be on local host for now and we have hopes of running it on a server in the future.

5. Installation Guide

Our software is compatible with most modern operating systems, e.g Linux, windows Mac OS. The software can be managed on both powerful and less powerful machines.
In our code we use unix command line examples. You can find similar code for windows. We would recommend that you run the following commands on an IDE such as VS code.

5.1 Install and run from Gitlab

Clone our project from our gitlab repository: `git clone` `https://gitlab.computing.dcu.ie/adedayj2/2023-ca326-jadedayo-dcumunch.git`

5.2 Change repository

You will need to change directory into desktops client directory : `cd 2023-ca326-jadedayo-dcumunch\code\backend`

5.3 Pip installations

Once you have completed this there will be a few installations that will need to be done in order for our web application to run smoothly on your end.

For all the code bellow run `pip install` before it
E.g `pip install asgiref==3.6.0`
(Helps python libraries to sync with each other)

`charset-normalizer==2.1.1`
(Helps you read text from an unknown charset encoding)

`cryptography==39.0.0`
(provides cryptographic recipe)

`defusedxml==0.7.1`
(XLM bomb protection for python stdlib modules)

`Django==4.1.4`
(Fixes several bugs)

`django-allauth==0.52.0`
(Integrated set of Django applications which address verification)

`django-cors-headers==3.11.0`
(handling the server headers required for Cross-Origin Resource Sharing)

`django-filter==21.1`
(allow users to filter querysets dynamically.)

`django-multiselectfield==0.1.12`
(A new model field and form field)

`djangorestframework==3.13.1`
(Web APIs for django)

`djangorestframework-simplejwt==5.0.0`
(A JSON Web Token authentication plugin for Django REST Framework)

`importlib-metadata==4.11.1`
(Read metadata from python packages)

`Markdown==3.3.6`
(python version of markdown)

`oauthlib==3.2.2`
(implementation of the OAuth request-signing logic)

`Pillow==9.4.0`
(Python Imaging Library by Fredrik Lundh and Contributors)

`py-edamam==0.2`
(food search algorithm)

`pycparser==2.21`
(A C parser in Python)

`PyJWT==2.3.0`
(Json web token implementation for python)

`pypng==0.20220715.0`
(python library for saving and loading PNG images)

`python3-openid==3.2.0`
(Open ID support for modern servers)

`pytz==2021.3`
(World time zone definitions)

`qrcode==7.4.2`
(QR code image generator)

`requests==2.28.1`
(python http for humans)

`requests-oauthlib==1.3.1`
(OAuthlib authentication support for requests)

`sqlparse==0.4.2`
(A non validating SQL parser)

`typing_extensions==4.5.0`
(Backported and Experimental Type Hints for Python)

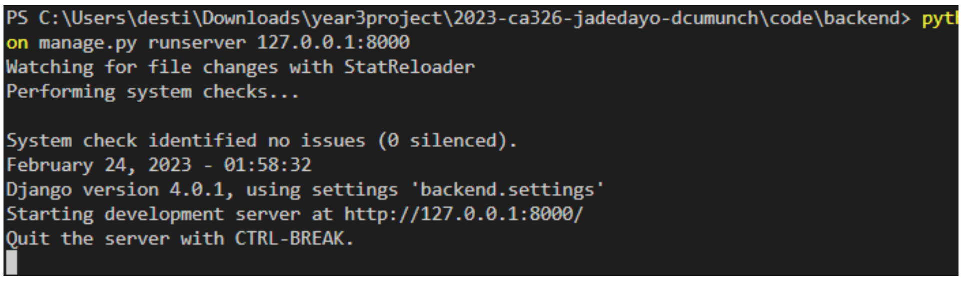
`urllib3==1.26.12`
(HTTP library with thread-safe connection pooling)

`wget==3.2`
(python download utility)

`zipp==3.7.0`
(Backport of pathlib-compatible object wrapper for zip files)

5.4 Run server

To run our server enter the following code into your terminal:
`python manage.py runserver 127.0.0.1:8000`



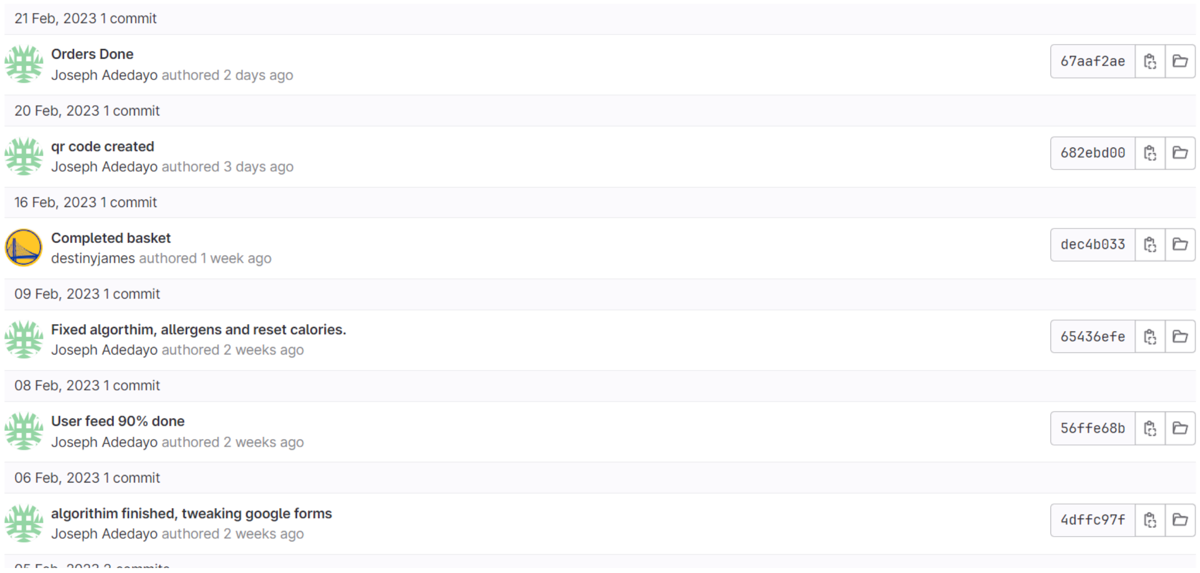
```
PS C:\Users\desti\Downloads\year3project\2023-ca326-jadedayo-dcumunch\code\backend> pyth
on manage.py runserver 127.0.0.1:8000
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
February 24, 2023 - 01:58:32
Django version 4.0.1, using settings 'backend.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

Click on the server link to load up our application

6. Testing

6.1 Git + CI/CD

We wanted to make sure the quality of our code was to high standards so we made use of the GitLab Continuous Integration runner which builds and tests our code upon each commit. In order to prevent errors when merging we only merge when the CI does not fail.
Below is an example of us committing our code and pushing it for each of us to see the changes and test it on our own devices.

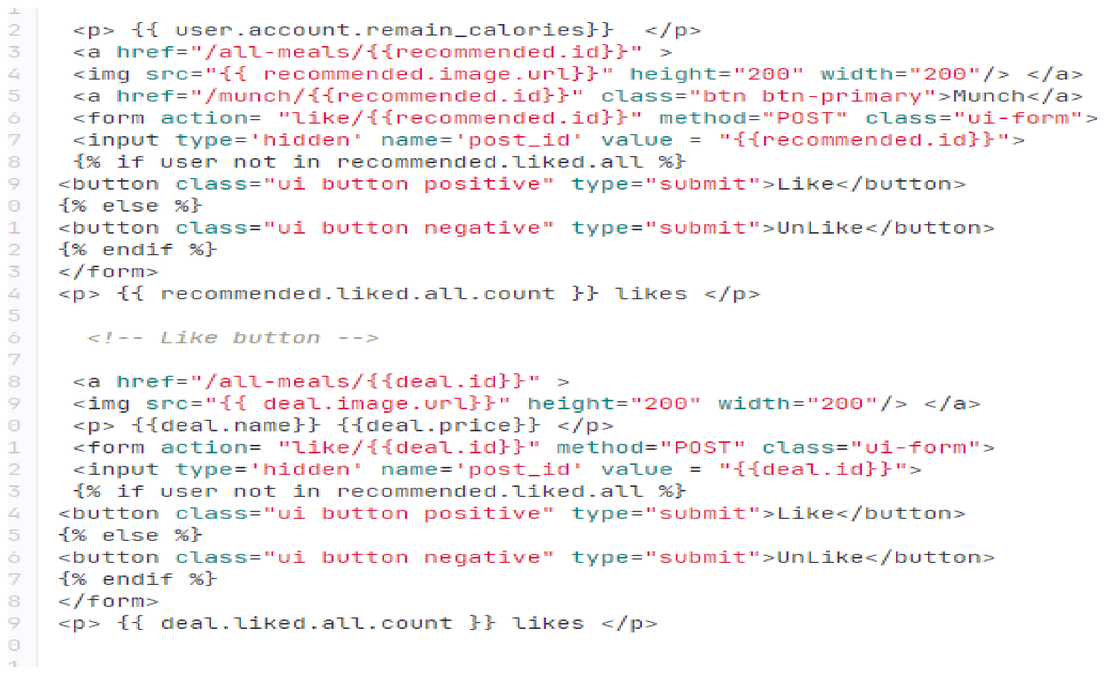


```
21 Feb, 2023 1 commit
Orders Done
Joseph Adedayo authored 2 days ago                                67aaf2ae
20 Feb, 2023 1 commit
qr code created
Joseph Adedayo authored 3 days ago                                682ebd00
16 Feb, 2023 1 commit
Completed basket
destinyjames authored 1 week ago                                  dec4b033
09 Feb, 2023 1 commit
Fixed algorthim, allergens and reset calories.
Joseph Adedayo authored 2 weeks ago                               65436efe
08 Feb, 2023 1 commit
User feed 90% done
Joseph Adedayo authored 2 weeks ago                               56ffe68b
06 Feb, 2023 1 commit
algorithim finished, tweaking google forms
Joseph Adedayo authored 2 weeks ago                               4dffc97f
```

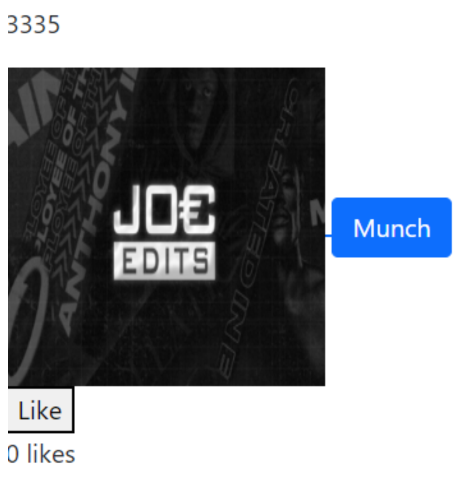
6.2 Unit Testing

Unit testing is when segments of code are written to test other small pieces of code.

We made a small test to see how the munch button and like button would look like on our webpage before adding any css



```
 <p> {{ user.account.remain_calories}}  </p>
 <a href="/all-meals/{{recommended.id}}" >
 <img src="{{ recommended.image.url}}" height="200" width="200"/> </a>
 <a href="/munch/{{recommended.id}}" class="btn btn-primary">Munch</a>
 <form action= "like/{{recommended.id}}" method="POST" class="ui-form">
 <input type='hidden' name='post_id' value = "{{recommended.id}}">
 {% if user not in recommended.liked.all %}
<button class="ui button positive" type="submit">Like</button>
{% else %}
<button class="ui button negative" type="submit">UnLike</button>
{% endif %}
</form>
<p> {{ recommended.liked.all.count }} likes </p>

  <!-- Like button -->

 <a href="/all-meals/{{deal.id}}" >
 <img src="{{ deal.image.url}}" height="200" width="200"/> </a>
 <p> {{deal.name}} {{deal.price}} </p>
 <form action= "like/{{deal.id}}" method="POST" class="ui-form">
 <input type='hidden' name='post_id' value = "{{deal.id}}">
 {% if user not in recommended.liked.all %}
<button class="ui button positive" type="submit">Like</button>
{% else %}
<button class="ui button negative" type="submit">UnLike</button>
{% endif %}
</form>
<p> {{ deal.liked.all.count }} likes </p>
```



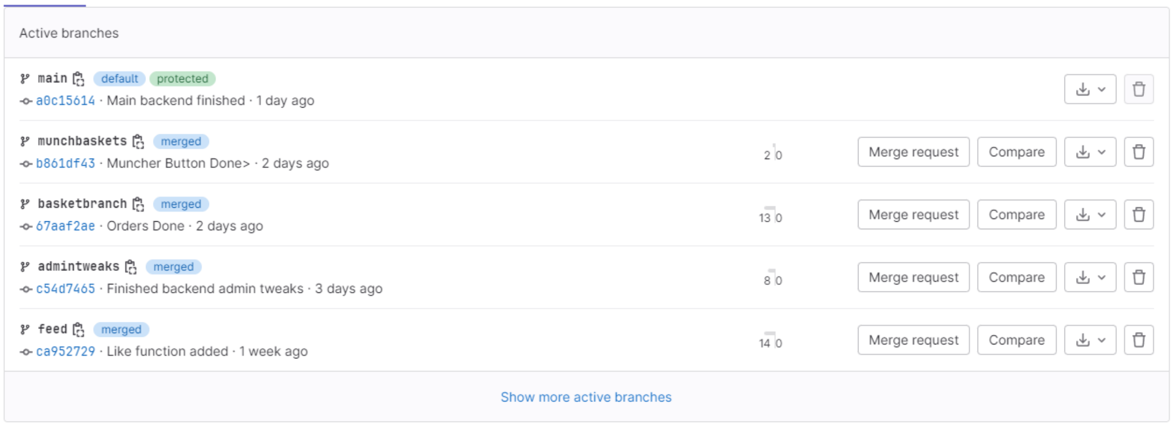
3335

0 likes

As you can see in the image above it displays our like button and munch button with no css

6.3 Integration testing

Our web application contains a lot of code and different sections. In order to prevent a lot of errors when pushing and merging code we created a branch for different sections in our web app for example, we had a feed branch and a basket branch.



Active branches

main default protected
a0c15614 · Main backend finished · 1 day ago

munchbaskets merged
b861df43 · Muncher Button Done> · 2 days ago — 2 0

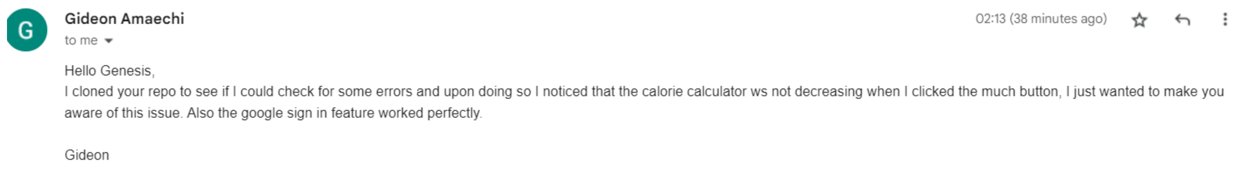
basketbranch merged
67aaf2ae · Orders Done · 2 days ago — 13 0

admintweaks merged
c54d7465 · Finished backend admin tweaks · 3 days ago — 8 0

feed merged
ca952729 · Like function added · 1 week ago — 14 0

Once we have done Git CI testing and are finished with each branch we would then merge it to the main branch where finished code will be. We then test the whole system together in the main branch to see if it runs correctly.

6.4 User Testing

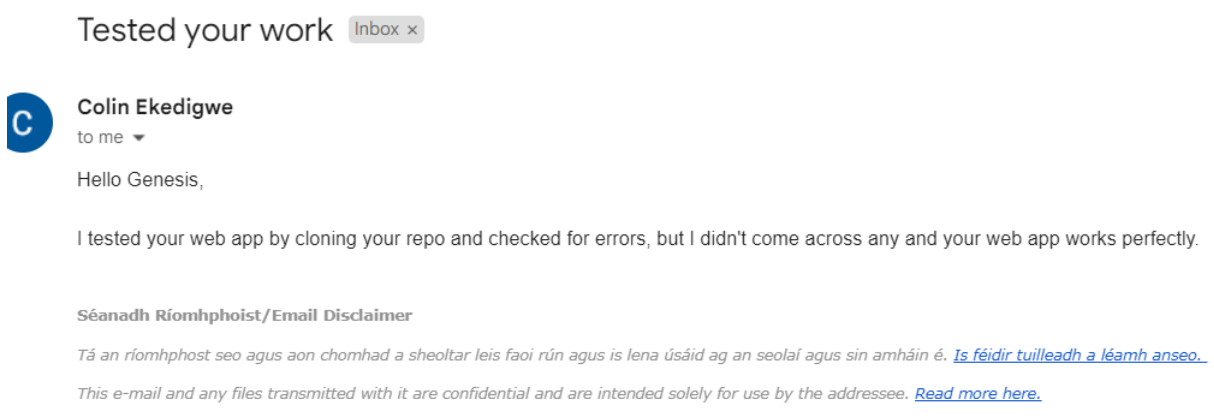


Gideon Amaechi
to me
02:13 (38 minutes ago)

Hello Genesis,
I cloned your repo to see if I could check for some errors and upon doing so I noticed that the calorie calculator ws not decreasing when I clicked the much button, I just wanted to make you aware of this issue. Also the google sign in feature worked perfectly.

Gideon

In order to perform user testing which is getting real users to perform some tasks on an application, we contacted a few of our friends to clone the repo and check if they bump into any errors which we might not have seen ourselves. Gideon above noticed that the calorie calculator wasn't decreasing when he clicked on much. This error was due to us not pushing the final code for the calorie counter. After we pushed it we told another friend to try out our application.


Tested your work Inbox ×

Colin Ekedigwe
to me

Hello Genesis,

I tested your web app by cloning your repo and checked for errors, but I didn't come across any and your web app works perfectly.

Séanadh Ríomhphoist/Email Disclaimer

Tá an ríomhphost seo agus aon chomhad a sheoltar leis faoi rún agus is lena úsáid ag an seolaí agus sin amháin é. Is féidir tuilleadh a léamh anseo.

This e-mail and any files transmitted with it are confidential and are intended solely for use by the addressee. Read more here.

We have also performed verbal user testing. Where the user looked at the webpage on my laptop and viewed our applications colour scheme functionality and layout.