CFG: Nanodegree Software Specialisation (Summer 2021): Group Project Report

Walk2Zero

CLIMATE CONSCIOUS STEPS

Team Members

Lakshika Juneja

Rajwinder Bhatoe

Robyn Seymour-Jones

Shuyan Liu

Sravya Betina

Contents

1. Introduction

This report documents the group project undertaken as one of the requirements for the Code First Girls Nanodegree (Software Specialisation).

1.1 Aim of the application

To reduce the greenhouse gas (GHG) emissions resulting from personal travel over land (i.e. not freight transport and not travel by air or sea).

1.2 Objectives

To achieve our aim, we will create an app that:

1. allows an individual to calculate the GHG emissions of each journey they take and keeps track of their total GHG emissions resulting from personal travel.
2. incentivises users to choose to travel by a transport method that emits a smaller volume of GHGs by:
    a. calculating and displaying the total GHG emissions for a planned journey according to the different transport vehicles a user has available to them (e.g. walking, cycling, bus, taxi, car).
    b. allowing the user to select the mode of transport they wish to take for their journey and calculating the difference in emissions between the biggest emitter and the method they decide to take.
    c. keeping track of a user's journey history and letting the user view the total GHG emissions they have offset by opting to travel via a mode of transport that emits a smaller volume of GHGs.

1.3 Report Outline

1. Introduction – Introduces the project and outlines the aims and objects of the application we are creating.
2. Background – A brief overview of the climate crisis, how transport emissions contribute to this and how our application can help.
3. Specifications and Design – Details the application's requirements, considerations, architecture and program flow.
4. Implementation and Execution – Explains how we have collaborated as a team and the steps we have taken to produce the application.
5. Testing and Evaluation – Details how we have ensured the application is of the highest possible quality.
6. Conclusion – Summarises the project and our experiences.

2. Background

1.1 Scientific Basis

The Climate Crisis

Climate scientists have warned that human activity is changing the Earth's climate in ways that are unprecedented with some of the effects now irreversible. Temperatures are expected to climb by more than 1.5 degrees Celsius beyond pre-industrial levels over the next two decades, exceeding the 2015 Paris climate agreement's goal and causing widespread devastation and extreme weather.

With recent flooding catastrophes that have unfolded in China and Germany and flash floods that have occurred in parts of London, it can be seen that the water cycle is becoming more intense as a result of climate change resulting in heavy rainfall. The Intergovernmental Panel on Climate Change (IPCC) has taken the recent climate change disasters into consideration and has declared this a code red for humanity and calls for drastic carbon cuts to pursue efforts to keep the global temperatures under 1.5C.

The Role of Greenhouse Gases

According to the newly published IPCC report, human-induced greenhouse gas emissions have caused global warming at a rate not witnessed in at least 2,000 years.

Worldwide, net emissions of greenhouse gases from human activities increased by 43 percent from 1990 to 2015.”According to UK national statistics , only green houses gases accounted for 97% of emissions” (***final-uk-greenhouse-gas-emissions-national-statistics-1990-to-2019)***[1] showing their significant contributions to climate crisis owing to temperature rise.

The Contribution of Land Transportation to Greenhouse Gas Concentration

In countries like the UK and the US, the transport sector is now responsible for emitting more greenhouse gases than any other mode of carbon emissions, including electricity production and agriculture. “For example, in the UK, transport emitted 122 $MtCO_2e$ in 2019, the largest amount of any industry and accounting for 27% of all UK greenhouse gas emissions that year” ***(Department for Transport, 2021)*** [2]. Globally, transport accounts for around a quarter of CO_2 emissions and there are concerns that this is hindering progress towards the Paris Climate Agreement, which saw countries set an ambitious goal to limit the global temperature rise this century to "preferably to 1.5 degrees Celsius, compared to pre-industrial levels" (***UNFCCC, 2021***). [3]

1.2 How our app will help

In this world of devastating consequences fuelled as byproducts of climate change caused by rising temperatures, our project idea helps every individual to act as responsible citizens of the earth by contributing a small step towards a bigger change.

Our application aims to guide users, reduce their daily carbon footprint by proposing more sustainable alternative methods of transportation. The service will display emissions associated with each mode of transport and allow the user to make a wise decision on the intended mode of travel. One bright feature of our application is, displaying offset of their carbon footprint, on the basis of time, priority and accessibility. The main idea behind building this application is, proposing the best user and environment friendly mode by presenting the impact of their journey in numbers (in values of carbon offset) is expected to inhibit more people into making a better transport decision. Allowing users to track their total carbon offset on every journey can help them to keep track of the difference they are making.

Finally, the emission difference which is being shown as the number of trees saved brings a clear insight of individual contribution to climatic change.

3. Specifications and Design

3.1 Requirements

When deciding on the requirements for the app, we decided to split them into a list of must have features and nice to have features. We realised that our project idea was ambitious and wanted to make sure that we were able to produce a minimum viable product within the time frame allowed but still convey our full idea of what the ultimate version of the app would look like.

Must have features

- Allows the user to register to the service.
- Allows the user to store their available modes of transport to their account. For example, they might have access to a car, motorbike, bicycle etc.
- Stores the carbon equivalent GHG emissions per km of the different transport vehicles that a user can register. Uses average figures, e.g. CO2e/km for an average sized petrol car.
- Identifies user location through user input.
- Identifies final destination through user input.
- Returns the distance of travel in km from an existing map API such as Google Maps.
- Proposes all modes of viable transport based on the distance, the user's mobility (e.g. age, physical abilities) and their saved modes of transport. For example, if the journey is <5km they could walk, if >5km they could take public transit or drive.
- Calculates emissions based on their saved modes of transport and potential distance travelled.

- Returns a carbon offset figure based on the user's chosen mode of transport vs their worst (in terms of GHG emissions) saved mode of transport. For example, a user has saved walking, cycling, bus and diesel car as their available modes of transport and this information is stored to their account in the DB. The app calculates the emissions for the worst mode of transport (in this case diesel car) and then returns the difference in carbon equivalent emitted between this mode of transport and the one they select to use.

Nice to have

- Allows the user to register their specific vehicle (e.g. make, model, age, MPG) and calculates more precise GHG emissions for that vehicle rather than using the average figures.
- Allows the user to enter the number of other passengers who are also making the journey to calculate more precise emissions for themselves, e.g. they are sharing a taxi so their emissions will only be their portion of the journey's emissions.
- Allows the user to set their own maximum distances for each transport method rather than using the built-in ones. E.g. they might only ever want it to return walking for journeys less than 2km.
- Allows the user to set their preferred distance unit – km or miles.
- Identifies user location through GPS rather than user input.
- A front end with a fancy GUI built from front end technologies (e.g. HTML, CSS, Flask, Django).
- Provides other incentives to walk, e.g. calculating calories burned. Could also be things like receiving a discount with a partner business but would need to look into the GHG emissions of that too – you wouldn't want to incentivise someone with a free coffee if the production of the coffee emitted more GHGs than they had saved!
- Allows the user to create and participate in challenges with friends (like Fitbit challenges) to see who can offset the most carbon equivalent.
- Congratulates the user for the amount of carbon they have offset by comparing their offset to things they can relate to more easily. An example could be the equivalent number of trees they would have had to plant, e.g. "your total carbon offset has amounted to 20 trees being planted, good job!")
- It could provide directions. The app could ultimately be an eco maps app that the user would use instead of something like Google Maps or Waze.

However, worth mentioning that despite time constraints, we did manage to successfully implement a feature that translates the amount of carbon offset to the amount of trees.

3.2 Considerations

- The app needs to take into account the user's physical ability. They might not be capable of walking or cycling.

- The app needs to be a mobile app as someone might be out and about when they want to calculate a journey. We are not implementing this in our project but are aware that this is what it would need to be.
- The app needs to be responsive and compatible to most devices to ensure the best possible visual and practical results for the user. To reiterate, we are not implementing this as of the moment but this is how it ideally needs to be.

3.3 Architecture

The app has a MySQL database, backend logic with Python and a front end implemented [in the command line/with Flask/fill in what we end up doing]. It uses our own API for communication between the client-side app, the server-side logic and the database. It fetches data from an external API – Google Maps. Figure 1 illustrates how these components work together.

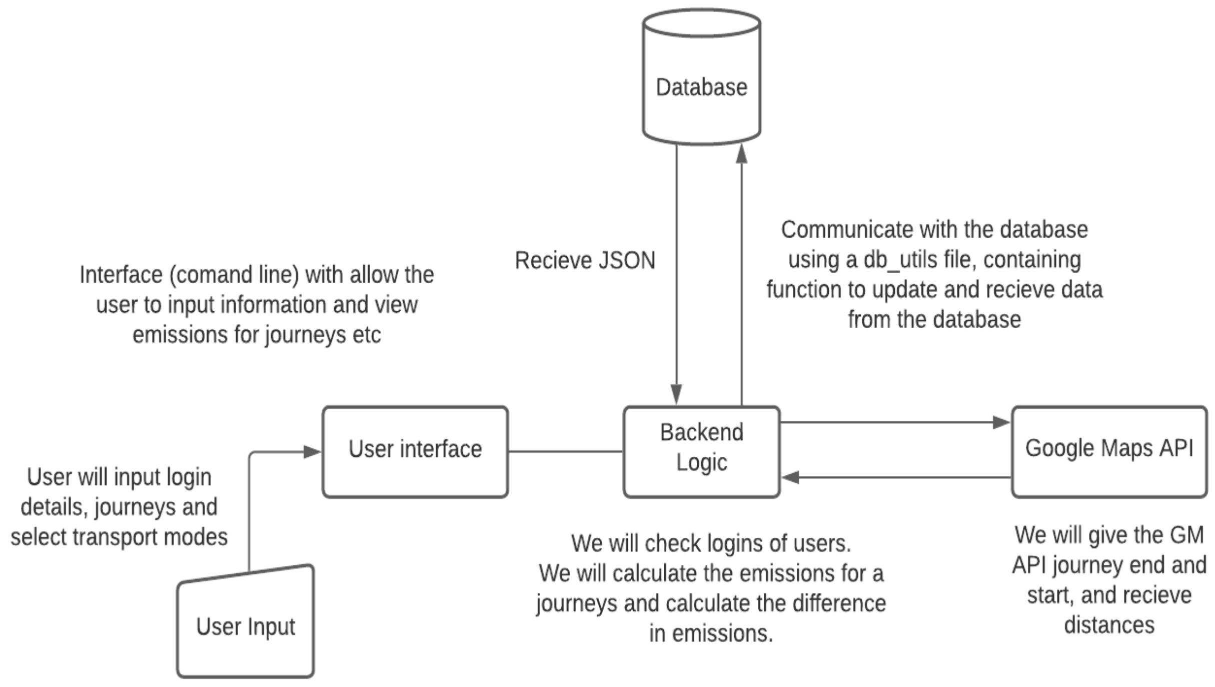

Database
Recieve JSON
Communicate with the database using a db_utils file, containing function to update and recieve data from the database
Interface (comand line) with allow the user to input information and view emissions for journeys etc
User interface
Backend Logic
Google Maps API
User will input login details, journeys and select transport modes
User Input
We will check logins of users. We will calculate the emissions for a journeys and calculate the difference in emissions.
We will give the GM API journey end and start, and recieve distances

Figure 1 – Diagram of the system architecture illustrating the main components of the Walk2Zero application and how they interact.

3.4 Program Flow

The flowchart in Figure 2 illustrates the program flow.

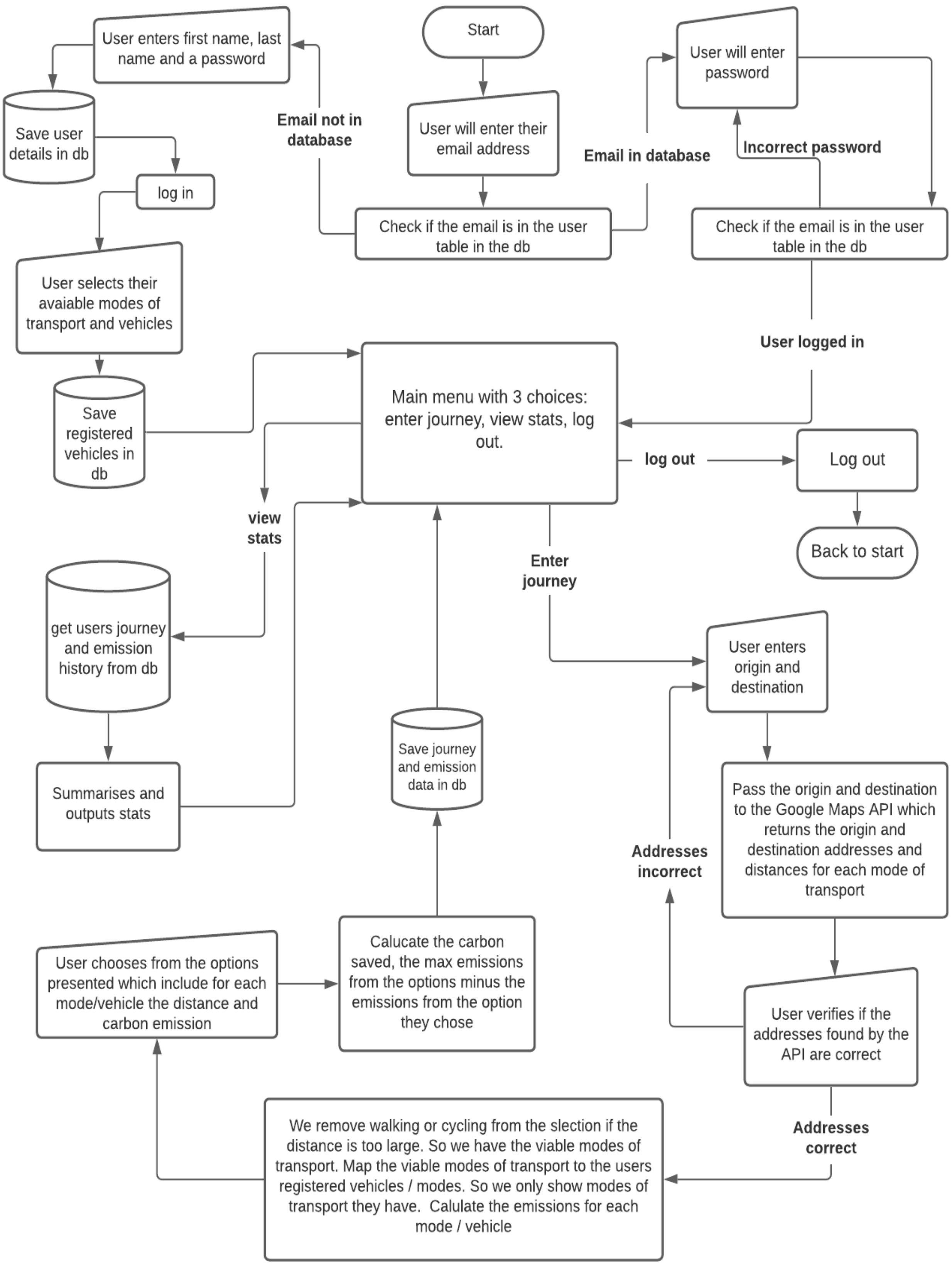

Start
User will enter their email address
Check if the email is in the user table in the db
Email not in database
User enters first name, last name and a password
Save user details in db
log in
User selects their avaiable modes of transport and vehicles
Save registered vehicles in db
Email in database
User will enter password
Incorrect password
Check if the email is in the user table in the db
User logged in
Main menu with 3 choices: enter journey, view stats, log out.
log out
Log out
Back to start
view stats
get users journey and emission history from db
Summarises and outputs stats
Enter journey
User enters origin and destination
Pass the origin and destination to the Google Maps API which returns the origin and destination addresses and distances for each mode of transport
User verifies if the addresses found by the API are correct
Addresses incorrect
Addresses correct
We remove walking or cycling from the slection if the distance is too large. So we have the viable modes of transport. Map the viable modes of transport to the users registered vehicles / modes. So we only show modes of transport they have. Calulate the emissions for each mode / vehicle
User chooses from the options presented which include for each mode/vehicle the distance and carbon emission
Calucate the carbon saved, the max emissions from the options minus the emissions from the option they chose
Save journey and emission data in db

Figure 2 – Diagram illustrating the program flow.

4. Implementation and Execution

4.1 Development Approach

Methodology

The development methodology adopted was Agile and we decided to undergo two sprints. Based on this, first intelligence and requirements were gathered. Depending on requirements, a list of tasks were broken down and were drafted on github kanban board, product backlog was designed and tasks were prioritized. Based on priority rank, tasks were selected and assigned to individuals as an issue.

Tasks were selected, coded and progress was updated and finally finished tasks were moved to review with a pull request to remaining team members. Based on reviews, changes are made and only after acceptance by the whole team, pull request was merged and issue was closed. Meanwhile, each finished function was tested using unittest and reviewed based on errors. Finished tasks were integrated one by one and integration testing was performed and were reviewed continuously for better performance of application.

We have finished all this functionality of the application in one sprint and we hope to add more colors in the second sprint in future.

Team Member Roles

We didn’t assign specific roles to any of the team members as we all wanted to gain experience of all aspects of app creation. As a team we discussed each task in our backlog and team members volunteered for a task they felt they were suited to or had time to work on.

4.2 Tools and Libraries

Management Tools

In order to manage our project we created a GitHub organisation which held our project repository. This allowed us to utilize the features included in GitHub to effectively manage our project. Within our repository we used the Kanban feature to create tickets for the features which we wanted to add into our project, and used new branches to work on these features.

Emissions Database

Emission values for the modes of transport in our database are sourced from gov.uk website from GHG conversion data published by the BEISS(The Department for Business, Energy and Industrial Strategy)[4].

The data for each vehicle includes a direct and indirect emissions factor. The direct emissions are those that result from burning fuel during the journey (i.e. tailpipe emissions) while indirect emissions are those that result from the production and transport of fuel prior to its consumption and electricity. This seems to be the official way businesses in the UK calculate their GHG emissions.

Emissions data for private vehicles and taxis are for a single person in the vehicle. Data for shared vehicles are for a single person based on average occupation, i.e. per person per km for public transport.

To translate carbon saved to trees planted

Considering carbon emissions, the amount of CO2(which contributes nearly 85% of greenhouse gases) was calculated and this value was depicted in terms of ‘trees saved’ by considering the amount of CO2 consumed by a typical tree in its lifetime.

1gram of carbon produces 3.67 grams of co2

1 tree consumes 21,772.67 grams of co2 in its lifetime

(calculated from molecular weights of components)

Libraries

Pandas

To create a mapping between the modes of transport which the google map API returns and the different types of vehicles which are saved in our database, we needed to use pandas dataframes which made this mapping process easier and more streamline.

NumPy

In order to manipulate data in the pandas dataframe NumPy had to be used.

Requests

As we were using a google maps api we had to use the requests library to be able to fetch the information we needed and receive it as a JSON which we could easily manipulate.

Mysql.connector

In order to connect to our mySQL database we had to use the mysql.connector library. This meant that we could read and write data in the database easily

Datetime

To save the date and time that a journey was entered by a user, we use the datetime library to get the current time.

Pyfiglet & Colorama

We used these to change the colour of the command line text and add other aesthetic elements.

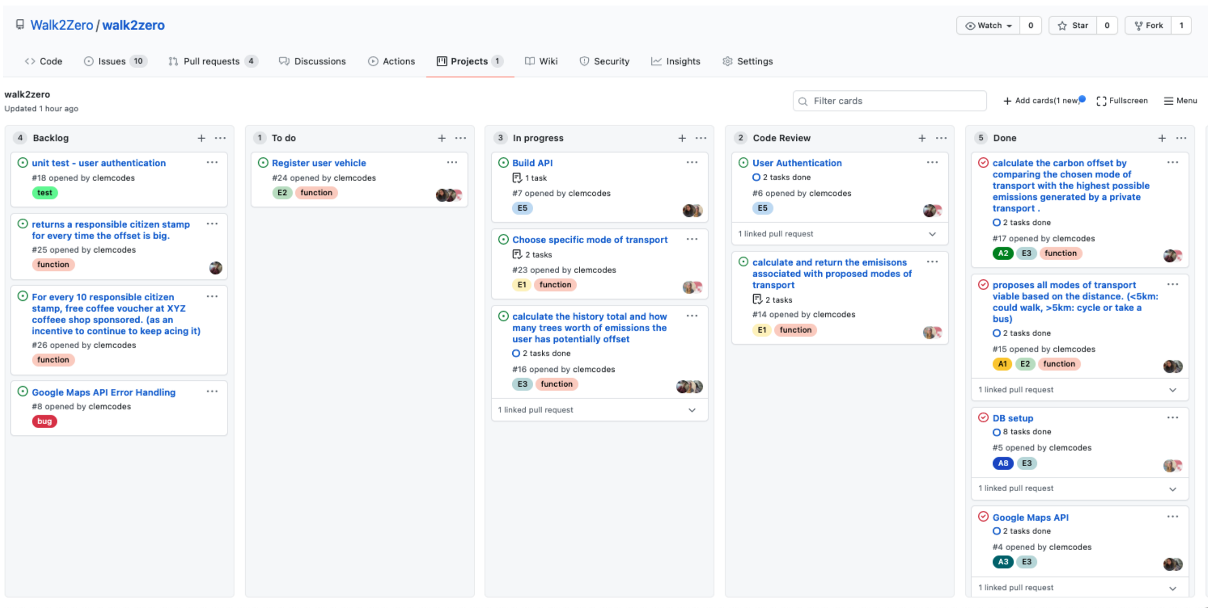
4.3 Implementation Process

Our project took the Agile Development approach where we initially divided the tasks to be completed within 2 sprints spanning 4 weeks.

Our first sprint planning meeting was kickstarted by listing out all the features that our MVP should have, populating the backlog on the Kanban Board(as can be seen from Figure 3) and delegating tasks within ourselves to finish at the end of sprint 1. Each feature on the backlog was converted to an issue and assigned to 2 people to mitigate any bottlenecks in progress and try out pair programming.

Further, when assigning tickets we estimated the time each ticket might take and after finishing each ticket, we observed the time actually taken aiding us to manage our time more effectively. We would then have frequent stand ups to update on completed tasks, assign more tickets and see what issues were holding us back. We also kept in contact on slack so we could communicate easily and often.

Our kanban board as mentioned in our previous section, allowed us to manage our tasks and keep track of what was in progress, needed reviewing,completed or blocked. When each ticket (equating to a branch) was complete we would create a pull request where the members of the team who did not work on the code were added as reviewers, and a pull request wasn't merged until all of the reviewers had approved. This means that bugs or issues were fixed before merging into main and everyone had an understanding of the code.



walk2zero
Updated 1 hour ago

4 Backlog

- unit test - user authentication
  #18 opened by clemcodes
  test
- returns a responsible citizen stamp for every time the offset is big.
  #25 opened by clemcodes
  function
- For every 10 responsible citizen stamp, free coffee voucher at XYZ coffeee shop sponsored. (as an incentive to continue to keep acing it)
  #26 opened by clemcodes
  function
- Google Maps API Error Handling
  #8 opened by clemcodes
  bug

1 To do

- Register user vehicle
  #24 opened by clemcodes
  E2 function

3 In progress

- Build API
  1 task
  #7 opened by clemcodes
  E5
- Choose specific mode of transport
  2 tasks
  #23 opened by clemcodes
  E1 function
- calculate the history total and how many trees worth of emissions the user has potentially offset
  2 tasks done
  #16 opened by clemcodes
  E3 function
  1 linked pull request

2 Code Review

- User Authentication
  2 tasks done
  #6 opened by clemcodes
  E5
  1 linked pull request
- calculate and return the emisisons associated with proposed modes of transport
  2 tasks
  #14 opened by clemcodes
  E1 function

5 Done

- calculate the carbon offset by comparing the chosen mode of transport with the highest possible emissions generated by a private transport .
  2 tasks done
  #17 opened by clemcodes
  A2 E3 function
- proposes all modes of transport viable based on the distance. (<5km: could walk, >5km: cycle or take a bus)
  2 tasks done
  #15 opened by clemcodes
  A1 E2 function
  1 linked pull request
- DB setup
  8 tasks done
  #5 opened by clemcodes
  A8 E3
  1 linked pull request
- Google Maps API
  2 tasks done
  #4 opened by clemcodes
  A3 E3
  1 linked pull request

Figure 3 - Kanban Board

Challenges

In the context of our collaborative workflow, we were initially not completely confident with git. We faced some challenges with branching for example branching off another branch rather than main and avoiding conflicting changes. However, we were able to figure this out quite quickly and the process went smoothly from there on.

Our 2 sprints also merged into 1 prolonged sprint run due to team members' work commitments and other submissions. However, everyone was very understanding, accommodative and proactively opted to own and re-delegate tasks to keep the progress going.

In the context of coding, we did face a lot of challenges in consolidating the code. One of our main challenges was structuring our code into OOP and more specifically trying to implement classes in the most effective way. Our initial buildup of the code was very functional in technique . In the process of converting that into OOP, the first thing we realised was that for each main function, we would need a corresponding db.utils function to write the data in the SQL DB.

Secondly, during our first system assembly, we grouped all our functions in 2 utils and 1 db.utils file which we soon realised was bad coding practice and went against the SOLID principles even though the program was fully functional. We needed to find a good balance between the factors reusability, locality, and ease of understanding and therefore took the step to refactor the functions by class into delegated files. This process was very cumbersome and stressful as it broke the code, introduced more bugs and some data discrepancies. This was a good learning experience though as with communication, code reviews and a lot of motivation, we did get it to work.

Additionally, making sure functions communicated with each other in the right way i.e returned and passed arguments in the right data structure format, proved to be a bit challenging. To facilitate easy passing of arguments, we either ended up creating helper functions for example: converting list of tuples to dictionary,converting dataframe to dictionary etc and in some cases decided to revise the functions as per the requirement of the consequent functions.

Unit testing for functions in classes took a while to figure out as well. Since some of our functions made use of dataframes, we were unable to look into mocking dataframes and testing more so because of time constraints.

Changes

We made small changes to work better, such as more frequent meetings and mapping out how our function will behave beforehand. One of the main changes we made was the amount of features we wanted to include. We realised that our initial backlog was too ambitious so we prioritised the essentials we needed to make the app work.

Achievements

We have managed to make a working app which has the functionality we aimed to have. Through the process we have understood how to use git to manage projects, keep track of our tasks, review codes and debug exhaustively.

We've also learned how to work in a team of developers, delegate tasks and complement each other's skillset to produce a MVP of a service in minimum possible time.

5. Testing and Evaluation

5.1 Testing Strategy

We implemented three strategies for testing our code:

Functional Testing

We tested our code with simple print statements before (almost) each return statement and as we were writing it. We ensured two developers were assigned to each task so that we could pick up on things that the other might have missed. Some of this was done by pair programming where one person shared their screen and both developers contributed to the code.

The final assembly of our code was also done in a pair programming technique where we exchanged ideas on how to best structure and consolidate it together.

We also made use of Spyder IDE to visualise our dataframe manipulations at each stage and to make sure it was returning the desired output in the right data structure format. It helped a lot in seeing what data frames stored after each reshape minimising our use of print statements.

Unit testing

Each function after review, was transferred to the testing phase. Functionality was tested with various sample cases and one in each type was presented. Based on function, end cases were written and tested while for a couple of functions, test cases were listed along with functionality requirements and were followed during the coding phase. Every function was finally tested for any undefined cases before finalising.

Blackbox testing

Once we had a working prototype we asked someone not involved in the development of the app to use it. We took feedback from them but as we were pushed for time, we were not able to implement any changes. These would be implemented in a subsequent sprint if we had more time on the project.

5.2 Code Evaluation/Review

As mentioned in section 4.2, we made extensive use of GitHub throughout this project. When a feature was complete, a pull request was submitted with every other team member assigned as a reviewer. A branch was only merged once every team member had had the opportunity to try out the code on their own computer and to leave any comments or suggestions.

We also made good use of the call feature in Slack and walked each other through our codes. This was very efficient when fixing bugs and implementing immediate feedback.

5.3 System Limitations

As outlined in section 3.1, we were aware that our idea was ambitious for the given time frame and have implemented a minimum viable app that works but does not do all of the things that a final completed version would. Some of the limitations of the app we have created are:

- The system does not allow a ‘registered’ user to amend their vehicle registration i.e. they cannot add or delete a vehicle after they’ve undergone their initial registration process.
- The modes of transport that the API returns groups all public transport into the ‘transit’ category that inhibits us from mapping specific options of bus, metro,tube etc to it. Our emission value hence was an average value sourced from gov.uk published by BEISS .
- Since the emissions data for private vehicles and taxis are for a single person in the vehicle, the carbon emitted during each journey for one user. Sharing or pool journeys were not considered at this moment as this requires the user to input the number of passengers each time when planning out the journey.
  We decided not to implement this feature given the constraint on time however this does limit the system from giving accurate data by over accounting the emissions for a single passenger.

6. Conclusion

‘Climate Crisis’ is a race of devastating consequences which can be won with some sort of individual responsibility. Every individual contributing to this crisis is accountable for its retrieval. As research clearly publishes the role of combustion of fossils for this crisis, our idea to reduce ignition levels is expected to bring a little sense of responsibility among individuals for this global issue.

Our motto to suggest a user and environment friendly mode of transport was clearly displayed by calculating individual contributions to climate in pictures of trees saved, bringing a little motivation and individual liability for future life on earth.

References

1. Waite, C .(2019), *2019 UK Greenhouse Gas Emissions.* Available at: https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/957687/2019_Final_emissions_statistics_one_page_summary.pdf
2. UNFCCC. (2021). *The Paris Agreement.* Available online: https://unfccc.int/process-and-meetings/the-paris agreement/the-paris-agreement [Accessed: 2021-08-01]
3. Department for Transport .(2021). *Transport and Environment Statistics 2021 Annual report.* Available online: https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/984685/ transport-and-environment-statistics-2021.pdf [Accessed: 2021-08-01]
4. https://www.gov.uk/government/publications/greenhouse-gas-reporting-conversion-factors-2021