# Team Software Project

# COM6103

# Final Report

# Glossary of terms:

- **Client:** The System owner and he is the reference for all user needs.
- **Visitor:** a person who can browse the web application without any privileges.
- **User:** anyone who is authenticated to the system and authorized to do certain things.
- **Admin:** is a user who has the full privileges to the system.
- **Staff:** is a user who has some privileges granted by the Admin.
- **Household:** A person who is willing to donate a solar panel to a country.
- **Paypal:** is a third party payment system to be used for money transactions.
- **Donator:** can be either an Admin, Staff, or Household member and he has the privilege to make a donation process.
- **Donation:** is a process that allows the donator to make the donation in terms of solar panels.
- **Panel:** is a single solar panel unit that can be donated.
- **Basket:** list of countries aligned with number of intended donated panels.
- **Upgrading Staff Account:** the process of gaining more privileges to a staff member.
- **Carbon Intensity:** is a measure of how much CO2 emissions are produced per kilowatt hour of electricity consumed.
- **Monthly Solar Power Intensity:** is a measure of the peak power generation of each solar panel per month in a country.

# Introduction:

Solar power nowadays produces 4% of electricity in the United Kingdom, with up to 20% efficiency. Many UK homes, however, have little or no space for solar panels since they lack of roofs or gardens in which to install them. These households may aspire to become carbon neutral, but solar panels are not an option.

Another issue is that gas boilers, which are in desperate need of replacement with (possibly carbon-neutral) Hydrogen-ready boilers, are still in their infancy. Even if homeowners purchase Hydrogen-ready gas boilers, they will not be converted to Hydrogen until the entire street has been converted, as the gas supply is shared which may take at least 10 years.

The aim of this project is to build a web app that helps people to reduce the carbon benefits in the world. The system will allow users to donate solar panels to some other countries which have high carbon emissions and consider an excellent environment for installing solar panels.

This project allows the donator to compare different countries depending on their features and then choose some to donate for.

Our system provides a registration feature to allow the user to donate frequently without the need to provide his/her information each time.

## What is sunlight?

Sunlight is a donation web app that provides a platform for households in the UK wishing to reduce the world's carbon footprint might do so by funding solar power in other countries, where the benefits include the seed capital infrastructure development, localized power generation, carbon reduction, employment, etc. Users can reduce the world's carbon footprint by funding solar power panels in other countries. We offer a large number of countries as donation choices and encourage our users to see the carbon footprint savings that they reduce through donating solar panels compared to their own carbon footprint.

## Project scope and objectives

- Design and build a donation website.
- It is about donating solar panels not only money.

- Allows people to reduce the total carbon emissions even if they can't depend on the solar system in their daily life.
- Provides a secure donating method.
- Provides an easy way to donate a solar panel.
- Encourage people to donate and to help reduce carbon emissions.
- Encourage people to reduce the carbon footprint.
- Illustrate the impact of the carbon footprint on the environment in a simple way for non-specialist people.
- The targeted user is people who live in the United Kingdom, but they could be from anywhere.
- The targeted country is any country with high carbon emissions.

# Team name and list of members

## Team name: Sunlight (Team 18)

### Team members (with their own usernames on gitlab):

- **Guo Yifei***: "yguo94@sheffield.ac.uk" & "Yifei Guo"*
- **Li Zeyu:** *"zli259@sheffield.ac.uk" & "leo lee"*
- **Wang Yixiang:** *"ywang597@sheffield.ac.uk" & "HolyL"*
- **Zhou Yue:** *"yzhou209@sheffield.ac.uk" & "YanyuQiuluo"*
- **Alfitni Arwa:** *"aaalfitni1@sheffield.ac.uk"*

# Product backlog

## Household User Stories:

### Information about countries:

- As a household, I am interested to know how donations helped reduce carbon emissions.
- As a household, I want to be able to read some information about the country I want to donate to. Information like what is the current situation for providing electricity.
- As a household, I'm interested in viewing the countries with different ranking features.
- As a household, it is important to me to see how this website will help me to reduce the carbon dioxide in the world.
- As a household, I want to see some information about the impact of carbon dioxide.

### Donation process:

- As a household, I want to be able to select different countries to donate to in one payment.
- As a household, I want to be able to donate panels in an easy and secure way.
- As a household, I want to be able to compare prices and carbon emissions for different countries.
- As a household, I want to see a graph representing the history of donations andtheirs impact on the carbon footprint.
- As a household, I want to store the list of countries I chose last time but haven't donated yet so that I don't need to choose again when I donate next time.
- As a household, I want to add or delete my selections quickly and see the amount of carbon footprint and money changed by these operations.
- As a household, I want to view my carbon footprint reduction after donation.

### Log in and registration:

- As a household, I want to be able to register with the minimum number of steps.

- As a household, I want a private grant to my information, so it isn't used for commercial purposes or any other purposes.
- As a household, I want to log in to the website easily and directly.

### Client User Stories:

- As a client, my goal is to encourage people to donate panels not only money.
- As a client, I want to show the number of panels donated for each country, not the amount of donated money.
- As a client, I prefer to see how many people are donated to a particular country.
- As a client, I want the user to be able to register easily. Registration steps shouldn't be exhausting and should require the only necessary information.
- As a client, I want a "get started" button which will allow the user to donate right away.
- As a client, I want to provide some recommended countries for donation.

### Management:

- As a client, I want my staff to be able to produce reports and statistics about the donation processes.
- As a client, I want to be able to manage the provided data.
- As a client, I want to upgrade accounts to some other level to give them some privileges.

## Analysis & Design

### Programming Languages and Database:

frontend: `HTML and CSS`

backend: `Node JS`

In order for the functions of the entire project to run smoothly in the background, we used some open-source packages to ensure the implementation of some special functions.

In the paypal sandbox function, we used the "paypal-rest-sdk" package, which was developed by the PayPal team, in order to make the code jump to the paypal page smoothly.

In order to ensure that during the registration process, the confirmation code received by the user and used for verification can be sent and received through the mail system in a timely manner, we use the "nodemailer" package that comes with node.js itself to protect the background program and our team's public email account api real-time docking safely.

Database: `MySQL`

First, we decided to use MySQL because some of the team members who are familiar with the relational database structure have some experience with MySQL. But then we decided to switch to MongoDB for an easier SQL structure. After that, we decided to return to MySQL when we were trying to produce some statistical results. We could integrate Node JS with MySQL easily using **Sequelize** programming tool. Sequelize is a modern TypeScript-based Object-Relational Mapping (ORM) tool for MySQL and others. it is used to map a piece of data from a relational model into an Object-Oriented one.

### System Requirements:

- **Registration:** allows a household to register to our database. So, he/she can donate frequently without the need to enter his/her info each time. Also, it keeps a record of all donation history as well as calculating his/her carbon reduction after the donation. The user must provide a unique username and password with a valid email address.
- **Mail authentication:** The registration feature required a mail authentication step. During registration, a code will be sent to the user's email. The code must be entered into the form for confirmation.

  The authentication step was not required by the client. However, we decide to add this feature to avoid repetitive or misleading usernames.

- **Log in:** allows a registered user to log in to the system using a valid username and password.

- **Log out:** allows the user to exit from the system.
- **View countries:** a list of countries that a user can view. Each country will be presented with its name, national flag, Carbon Emissions per capita, Gross Domestic Product (GDP), price of purchasing and installing a solar panel, and Monthly Carbon Footprint savings of each solar panel.
- **Add panels to a basket:** a user may add any number of panels to the basket. He/she can select multiple countries each of which with different numbers of panels to add to one basket. All contents of the basket will be donated at once.
- **Donate:** allow users to donate panels for specific countries.
- **Payment Authentication:** as an internal step of the donation feature, payment must be authorized through Paypal.
- **Managing staff accounts:** The admin can edit, delete or add a new staff member.
- **Upgrade household account:** The admin can upgrade an exciting household account to a staff account.
- **Relegate staff account:** The admin can relegate an exciting staff account to a household account.
- **Calculate the Monthly Carbon Footprint savings of each solar panel:** Monthly Solar Power Intensity(kWh/month) * Carbon Intensity($gCO_2eq/kWh$) / 1000 = Carbon Footprint savings of each solar panel per month($kgCO_2$/month)
- **Reports:** the system shows a single report including all results that are required by the client, including each country's total amount of savings, the total quantity of solar panels, and total carbon footprint reduction.

## UML diagrams

### Use case Diagram:

Our system deals with five different types of stakeholders (Figure 1). First, a Visitor, can view the homepage and see a list of countries with their details, and navigates to the detailed page of each country. He can also register on our system as a household. Second, the Household, which represents a regular user that can log in to the system and who wants to donate to some country. Household can also identify his own current carbon footprint. Third, is the staff member who is responsible for extracting reports about the donations and managing the country list. The Staff member also could view the reports. Fourth, is the admin, which has some management tasks for the household and the staff accounts. And fifth, Paypal, is a third party that is used for payment authorization.

## Users Privileges:

### Visitor

- Can browse the Home, About, and Contact us pages.
- Can access to all countries information including panel prices and other details

### Household

- All Visitor privileges are applicable.
- Can login/logout to the system.
- Can add panel units to the basket (for each country).
- Can View the basket.
- Can donate.
- Can view the “My Account” page to browse his own details.

### Staff

- All Household privileges are applicable.
- Can view the “Report” page.

### Admin

- All Staff privileges are applicable.
- Can access the “Admin” page so that he can view all details of all users.
- Can upgrade a Household to a Staff user.

- Can relegate a Staff to Household.

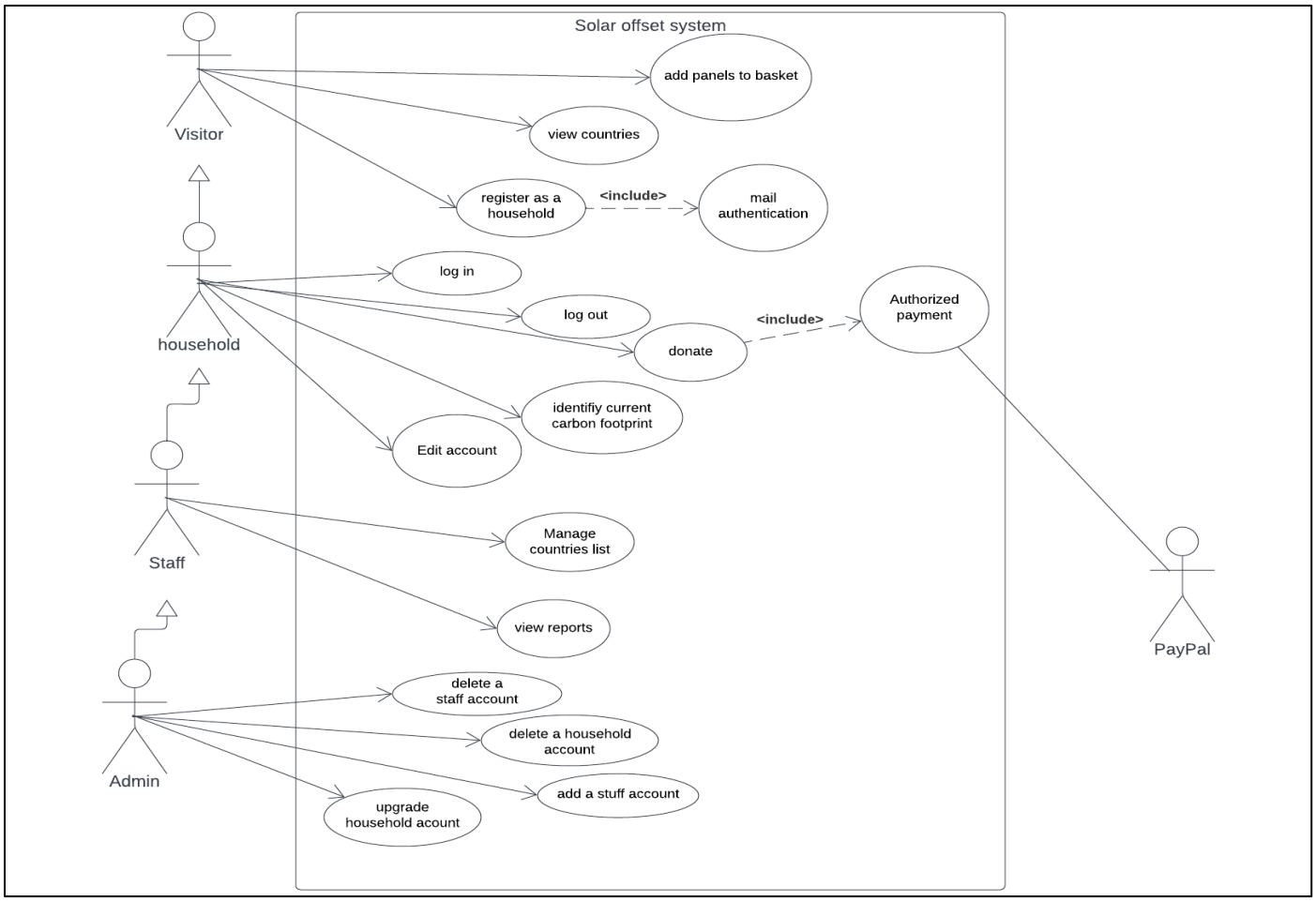



*Figure 1: Use Case diagram*

## Database design:

### Entity Relationship Diagram:

We used a simple database design. Figure 2 illustrates the design of our system. It consists of three tables: User, Transaction, and country.

The User table represents all types of users which can be an admin, a staff, or a household. The “user type” distinguishes between the three types of users. This table contains some personal information including username, password, email, and more. In addition, fuel and electricity usage per month “fuel_usage_pm” and “electricity_usage_pm” respectively which used to compute the carbon usage for each user. Each user is identified by an automatically generated unique integer number.

The Country table represents a country that a household wants to donate to. Each country has an id, name, flag picture, a description, its GDP, carbon emissions per capita, and the cost of installing a solar panel. “carbon_intensity” represents the amount of carbon dioxide produced by using electricity while “kwh/m2/mon” represents the number of kilowatt per hour produced by a square meter solar panel each month. “carbon_savinng_factor” is used to calculate the impact of using the solar panels in reducing carbon dioxide.

The Transaction table is where we keep a record of all donations. Each row represents one operation of transferring money through the sandbox. The status feature could be a success, cancel, or pending.

A donator may choose more than one country with a different number of panels for each country and then donate all those panels in one payment operation. The whole payment operation will have a unique id “uuid”. Since many donation operations can be done at one time, then we will have multiple rows with the same “uuid” and hence this id can’t be used as a primary key. An “id” is used as a primary key.

The start time and the end time are used to ensure the completion of the payment.

The “panel_amount” and “transfer_amount” represent the number of donated panels and their total price that were transferred to a specific country.

Each user can make many donations to different countries. Similarly, each country may receive many donations from either the same or different users.

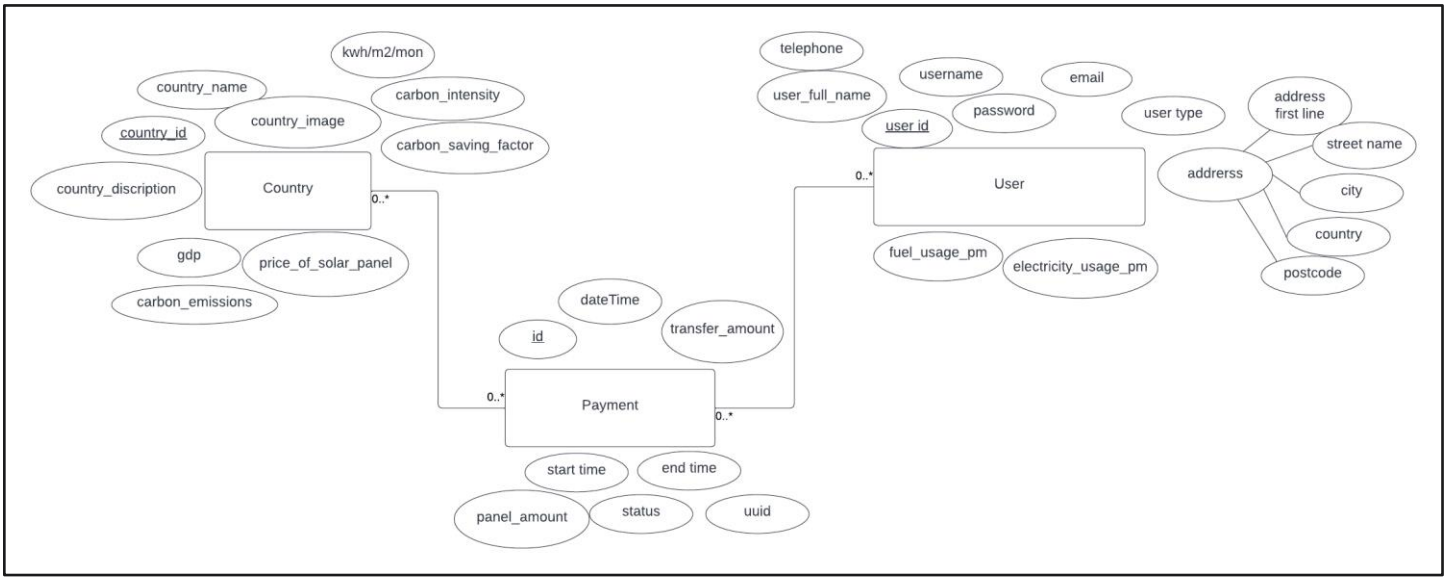



*Figure 2: ERD for the Sunlight  System*

# Evidence of Testing

## Test plan:

At first, we planned to do automated testing using **Selenium**. Selenium is a free (open-source) automated testing framework used to validate web applications across different browsers and platforms. It can use multiple programming languages like Java, C#, Python, etc to create Selenium Test Scripts. Testing done using the Selenium testing tool is usually referred to as Selenium Testing.

Selenium Software is not just a single tool but a suite of software, each piece catering to different Selenium QA testing needs of an organization.
Automated testing has several advantages such as eliminating human bugs and allowing frequent testing to be more easily. It is fast and can save time and effort.

But due to lack of time and since we have no previous experience with automated testing, we decided to do manual testing for our website. We tried to write test cases for the most important features. The test cases are generated from the requirements and the user stories. Test cases are shown in table 1.

## During development:

Postman was used for checking the connection between the frontend, backend, and the database. Postman is an API platform for building and using APIs. Postman simplifies each step of the API lifecycle and streamlines collaboration. more information about using Postman in our development process is explained in the Backend to front-end presentation of API in the Challenges section.

## Test documentation:

### Test cases:

| Test number | Test name | Actor | Pre-condition(s) | Post-condition(s) | Steps |
|---|---|---|---|---|---|
| 1 | homepage | visitor | none | Success:<br>- homepage present correctly.<br>- click the entrances on the homepage, and goto the correct pages<br>Failure:<br>- homepage present incorrectly.<br>- goto incorrect pages | - Open the website<br>- Click the navigation bars on the top and the bottom of the homepage.<br>- Click the links in the main content of the home page.<br>- Goto the correct pages |
| 2 | Log in | Household<br>Staff<br>admin | none | Success: homepage present with the username shown on the top-left corner.<br>Failure: failed message popped up | - Open the website<br>- Click on the login button<br>- Enter the username and password in the correct box<br>- Click login |
| 3 | Register | Household | none | Success: household account is created, and the home page is displayed with the username appease on the top-left corner.<br>Failed: fail message popped up. | - Open the website<br>- Click on the Register button<br>- Enter your username (email), and password in the correct box.<br>- an authentication code will be sent to the user email.<br>- Enter the code in the correct box.<br>- account verified and created. |
| 4 | browse the country list | visitor | none | Success: present the page for all countries.<br>Failed: “page not found” message popped up. | - Open the website<br>- From the top-right tabs – choose the country list |
| 5 | browse the country list | visitor | none | Success: select the sort indexes and sort order, present the correct order of country sorted by selected indexes.<br>Failed: the list of countries cannot be sorted by correct indexes | - Open the website<br>- From the top-right tabs – choose the country list<br>- Select an option of sorting index and sort order<br>- Present the correct order sorted by selection. |
| 6 | browse the country list | visitor | none | Success: click the page turning button and change into the new list of countries.<br>Failed: the list of countries does not change. | - Open the website<br>- From the top-right tabs – choose the country list<br>- Click the right page turning button, the page will show later countries.<br>- Click the left page turning button, the page will show former countries.<br>- Click the left page turning button, the page will stay if the page number is 1. |
| 7 | browse the country detail | visitor | none | Success:<br>- present the page correctly.<br>- click the - or + can minus or add the number of solar panels | - Select a country.<br>- The detail page show correctly.<br>- Click the - or + button. |

| | | | | | |
|---|---|---|---|---|---|
| | | | | Failed:<br>- present the page incorrectly.<br>- the number of solar panels does not change will click the - or + button | - The number of solar panels reduced or increased with the operation. |
| 8 | forget password | household | | Success: homepage present with the username shown on the top-left corner.<br>Failed:<br>- failed message popped up.<br>- click button dosen't work. | - open the website<br>- click on login button<br>- click forget password button<br>- enter your email<br>- a new password will be sent to your email<br>- use the new password to login |
| 9 | donate | Household<br>Staff<br>admin | Log in | Success:<br>- Success message of donation showed up.<br>- Transaction successfully finished. and stored at the transaction table.<br>- The number of donated panels is updated on the relevant country's detailed page.<br>Failed:<br>- A failed message showed up.<br>- No transaction is stored at the transaction table.<br>- The number of donated panels didn't change on the relevant country detailed page. | - Select a country.<br>- Add the number of panels you want to donate.<br>- Go to step one if you want to donate to another country.<br>- Click on the basket icon.<br>- A summary of all selected countries will be shown.<br>- Click donate.<br>- The website will navigate to paypal for authentication. |
| 10 | donate | visitor | none | Success:<br>- Jump into the login page<br>Failed:<br>- Jump into other pages | - Select a country.<br>- Add the number of panels you want to donate.<br>- Go to step one if you want to donate to another country.<br>- Click on the basket icon.<br>- Go to the login page |
| 11 | result | Household<br>Staff<br>admin | Log in | Success:<br>- Show the information of payment success and the carbon footprint reduction<br>- Click the view my carbon footprint button and jump into the account page<br>Failed:<br>- Show error message<br>- Click button and doesn't work | - Finish payment on the newly opened paypal website.<br>- Automatically close the newly opened window and click the paid successfully button on the basket page.<br>- Jump into the payment result page and show the information correctly.<br>- Click the view my carbon footprint button and jump into the account page. |
| 12 | account | Household | Log in | Success:<br>- Show the information of payment history and the total carbon footprint reduction<br>- Input the monthly electricity consumed and calculate the correct number of carbon footprint.<br>Failed:<br>- Show error message<br>- Cannot input the number or calculate the carbon footprint. | -Log in and goto the account page.<br>- Check the information of payment history and the total carbon footprint reduction.<br>- Input the number of monthly electricity consumed and calculate the correct number of carbon footprint. |

| | | | | | |
|---|---|---|---|---|---|
| 13 | upgrade household account | Admin | Log in | Success:<br>- at the admin account, the user type changed to staff.<br>- the user can log in to his account as a staff.<br>Failed:<br>- click button dosen't work.<br>- the user cannot log in to the system<br>- the user still a household. | - Log in and goto Admin page.<br>- a list of all users will showed up.<br>- click on the Upgrade button at the selected user's row. |
| 14 | relegate staff account | Admin | Log in | Success:<br>- at the admin account, the user type changed to household.<br>- the user can log in to his account as a household.<br>Failed:<br>- click button dosen't work.<br>- the user cannot log in to the system<br>- the user still a staff. | - Log in and goto Admin page.<br>- a list of all users will showed up.<br>- click on the Relegation button at the selected user's row. |

*Table 1: Test Cases for the Sunlight System*

## Test results

| Test number | Test name | Success or Failure? | Notes |
|---|---|---|---|
| 1 | Homepage | success | |
| 2 | Login | success | |
| 3 | Register | success | |
| 4 | Browse country list | success | |
| 5 | Browse country list | success | |
| 6 | Browse country list | success | |
| 7 | Browse country detail | success | |
| 8 | Forget password | failed | This feature was uncompleted |
| 9 | donate | success | |
| 10 | donate | success | |
| 11 | result | success | |
| 12 | account | success | |
| 13 | upgrade household account | success | |
| 14 | relegate staff account | sucess | |

*Table 2: Test results for the Sunlight System*

# Team management & communication

For building this project we follow the Agile methodology. The Agile methodology is a way to manage a project by breaking it up into several phases. It involves constant collaboration with stakeholders and continuous improvement at every stage. Once the work begins, teams cycle through a process of planning, executing, and evaluating. Continuous collaboration is vital, both with team members and project stakeholders. Each iteration (phase) was three weeks long.

## Workload:

### First iteration – week 3 to 5:

The first iteration was mainly about understanding the system, collecting the requirements, and finding data sources. All team members participated equally in those tasks as they required brainstorming. By the last week of the iteration, we were able to start building the system, and hence it was the time to divide the work. We agreed to manage the workload as follow:

- Frontend team: Yifei Guo and Zeyu Li
- Backend team: Yixiang Wang and Yue Zhou
- Database team: Arwa Alfitni

### Second iteration – week 6 to 8:

For the second iteration, we were working on building the system and as a team, we continued to work as we planned by the end of the first iteration.

- Frontend team: Yifei Guo and Zeyu Li
- Backend team: Yixiang Wang and Yue Zhou
- Database team: Arwa Alfitni

### Third iteration – week 9 to 11:

In the third iteration, we faced some difficulties dealing with executing queries. Yixiang Wang suggested exchanging our DBMS to MongoDB since this will facilitate dealing with inserting and retrieving data. Thus, we will deal with an object, not with rows in tables.

Two weeks later we faced another issue with our structure using MongoDB. We designed the database as a relational model but then used it as a non-relational model. This caused some difficulties when we were trying to extract some statistics like the number of people donated to a specific country or the number of donated panels.

We figured out that the best way is to use the relational model through MySQL and by using the "Sequelize" package we can deal with each row as an object.

On the other hand, it was the time to work on the final report and

- Frontend team: Yifei Guo and Zeyu Li
- Backend team: Yixiang Wang and Yue Zhou
- Database team: Arwa Alfitni, Yixiang Wang, and Yue Zhou
- Documentation:
    - Final report:
        - Introduction: Arwa
        - project scope ans objectives: Arwa
        - Glossary of terms: Arwa
        - Programming languages and Database: Arwa
        - System Requirements: Arwa
        - UML: Arwa
        - User privileges: Arwa
        - Database Design: Arwa
        - Evidence of Testing: Arwa and Yixiang
        - Team Management and Communication: Arwa
        - Planned and completed features: Arwa
        - Uncompleted Features: Yifei

        - ScreenShots of relevant pages: Yue
        - Demo Video: Yue
        - Conclusion: Yixiang, Zeyu, and Yifei
    - Setup guide: Yixiang
    - User guide:Yue
    - Meeting Minutes: Arwa
- Testing: Arwa and Yixiang

## Communication:

We decided to share our code and documents through the GitLab repository. GitLab is an open-source, single application that spans the entire software development lifecycle. Without using GitLab, our development lifecycle is likely spread across large numbers of applications. These silos take overhead to integrate, manage, configure, and maintain, slowing down our team and our deployments. Moving to a single application will speed up our workflow and help us deliver better software, faster.

As a daily communication base and for arranging meetings we used WeChat. WeChat is one of the most popular social messaging apps in the world. WeChat supports the basics like voice chatting, picture messaging, and video calling which are all we need for our communication purpose.

The team members meet twice a week. Before each meeting, we discuss what each of us will work on. Then at the meeting, we work together to solve any difficulties. Our meetings are usually 4-5 hours, each one of us works on his/her part and we share thoughts and help each other with any hardship.

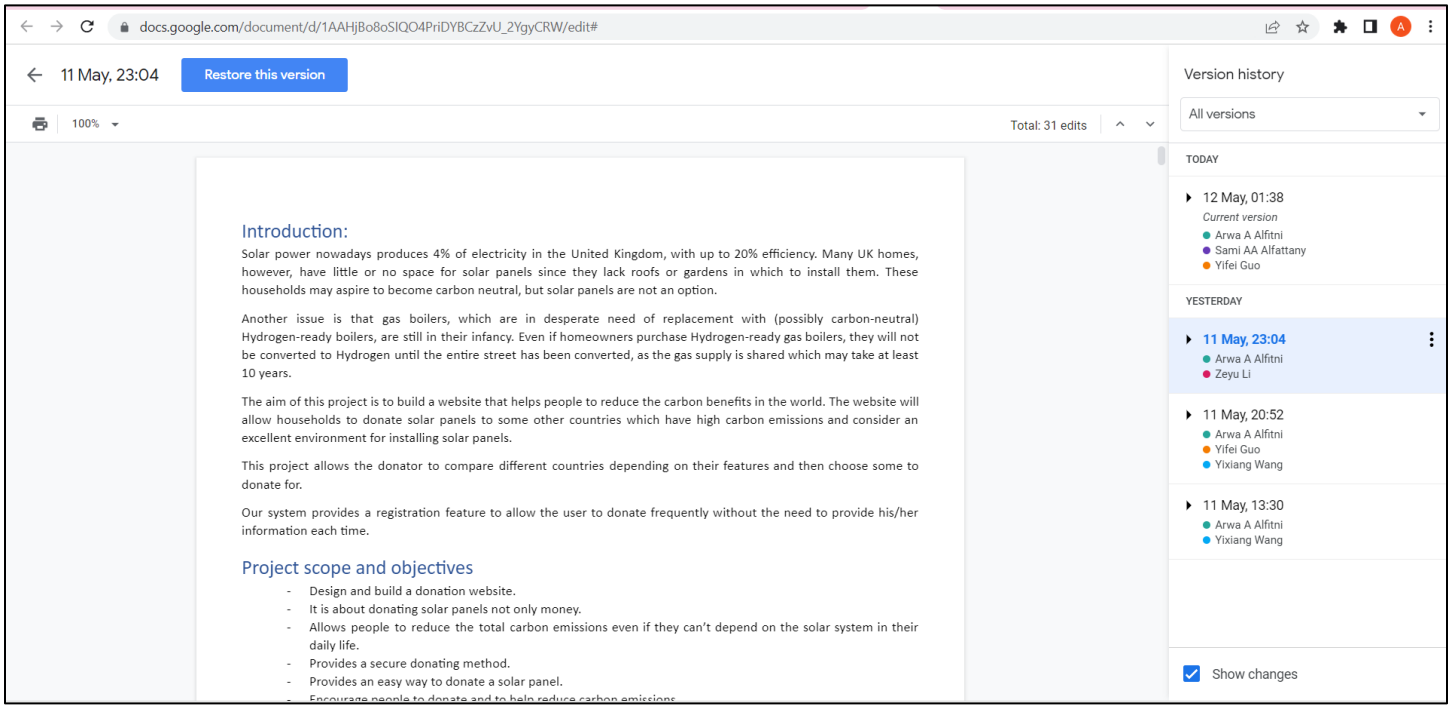



*Figure 3: Evidence of team participating in the final report*

For the final report, we used a shared file through Google Docs. Each member adds his/her part. Figure 3 is a screenshot of the version history of the final report showing the user member participating in this task.

## Development tools:

To synchronize our development progress, we also use a lot of helper software. We use github desktop to allow git commands, which is faster and less error-prone than typing git commands directly. Also, this software makes it easy for developers to view each person's commit history. And when something doesn't work, it's easy to open a new branch from history. We actually had a crash during our development process, and the history and rollback features eventually helped us to resolve it. We use navicat to manage our database. This software allows us to quickly import and export SQL files

containing the entire database, which also allows us to quickly sync our database changes to other members of the group. And postman is used to publish web API's. When we join the same team, changes to API names or parameter formats can be automatically synchronized, which also improves our development efficiency.

### Code structure:

In terms of code structure, we have developed some specifications to reduce conflicts caused by code changes. The development framework we use is express, which has a main app.js file that is usually responsible for registering web API's or linking URLs to web pages. At the start of development, we created a lot of new API's and web pages, so the app.js file was modified frequently, which caused us to encounter commit conflicts. So, we split app.js into six discrete files, one retaining the base structure and the other five being maintained by other developers (named app_developer_name). The main app.js will run the other five parts one by one at the start of the application. With this split, git commit conflicts are almost eliminated.

In addition, in order to divide each function and make it easier for members to read other people's code, we created a separate file for almost every API in the routes folder, with those ending in Page being the API for rendering web pages and the rest being the API for data exchange, making development easier and making it less difficult to find and maintain API's developed by others. This makes development easier and reduces the difficulty of finding and maintaining API's developed by others.

### Meeting minutes samples:

The team regularly meets two times a week, and most of the meetings are documented using meeting minutes templates. The templates consist of four parts. The first section is some information about the meeting time, location, and invited members. The second section is the key points discussed or the completed tasks grouped into categories. The third part is the plan for the next meeting. Then lastly is the approval part.

See Appendix C for some meeting minutes samples for each iteration.

## Planned & Completed Features

### First iteration - week 2 - 5:

| Category | Planned task | Completed? |
|---|---|---|
| Requirements | Defining the system requirements | Yes |
| | Review the requirements with the client | Yes |
| | Draw a use case diagram | Yes |
| | Determine Programming languages | Yes |
| | Determine development tools | Yes |
| | Determine Communication tools | Yes |
| Database structure | Determine the resources of the data. | Yes |
| | Designing the Database structure. | Yes |
| | Build the database. | Yes |
| | Insert some sample data. | No |
| UI design: | Determine the needed pages. | Yes |
| | Designing Log in page. | Yes |
| | Designing registration page | No |
| Backend programming: | Connect to the database | Yes |
| | Log in | Yes |
| | Setting up the GitLap repository | Yes |
| | Installing development IDEs | Yes |

*Table 3: Planned tasks for the first iteration*

### Second iteration – week 6 - 8:

| Category | Planned task | Completed? | Note |
|---|---|---|---|
| Database structure | Change to MongoDB | Yes | |
| | Finding a way to synchronize the database with the team member | No | More info in the conclusion |
| | Review the design | Yes | |
| | Insert some sample data. | yes | |
| UI design: | Registration page. | Yes | |
| | Country list page. | Yes | |
| | Ranking country list upon some features | yes | |
| Backend programming: | Mail authentication | yes | |
| | Ranking country list upon some features | yes | |

*Table 4: Planned tasks for the second iteration*

### Third iteration – week 9 - 11:

| Category | Planned task | Completed? |
|---|---|---|
| Database structure | Review the design | Yes |
| | Insert more data. | yes |
| UI design: | Building Personal information page | Yes |
| | Building Donating page | Yes |
| | Building Staff page | Yes |
| | Building Admin page | Yes |
| | Building Contact Us page | Yes |
| Backend programming: | Implementation of the Donation process | Yes |
| | Implementing reports at the Staff page | Yes |
| | Admin page | Yes |
| | Calculating the impact of the donation on reducing carbon dioxide. | Yes |
| Documentation | Writing the final report | Yes |
| | User guide | Yes |
| | Setup guide | Yes |
| Testing the system | Write test cases | Yes |
| | Executing test case | Yes |
| | Writing test result | Yes |

*Table 5: Planned tasks for the third iteration*

## Uncompleted Features

### ○ Forgot password:

The login and registration pages provide users with a function to recover their passwords. This function is easy to implement. The reason why it is not realized is that the project duration is relatively tight. This function is designed to optimize users' experience, not a core function. The priority of this function is relatively low, and we can choose to implement it in subsequent development. This function requires the front-end to add a judgment logic of recovering the password pop-up window to verify the mailbox and username entered by the user. If the verification is successful, an email to modify the password will be sent to the mailbox with the verification code. After the user enters the verification code, if the verification is successful, the user is allowed to reset the password and we will send the new password to the back-end for storage in the database.

### ○ Line graphs of carbon footprint:

On the personal account page, a line graph of the user's carbon footprint reduction through the project is shown to the user. The reason why this function is not realized is that the project duration is relatively tight. We have shown the reduction in the carbon footprint of each donation to users in the form of tables on the personal account page. In order

to improve the user experience and give users a more intuitive feeling, this function has medium priority and can be implemented in subsequent development. This function requires the front-end to draw a line chart using canvas, adding the donation data queried from the backend in the chart, and compare the current carbon footprint calculated according to the power consumption input by the user, so as to provide users with more suggestions to reduce the carbon footprint.

### ○ Recommended system on home page

There is a Recommendation part at the bottom of the home page. We originally wanted to write an algorithm to display the recommended countries based on factors such as carbon emissions, the number of donations, and the carbon footprint. However, due to time reasons, we simply show our recommended countries sorted by database data. We calculate the ratio of the monthly reduced carbon footprint of each solar panel to the price of each solar panel and get the carbon footprint that can be reduced per pound. Taking this ratio as the recommendation index, we will show the six countries with the highest ratio in the recommended section and broadcast it in rotation.

### ○ Page beautification

The personal account page, admin page and staff page just use the infrastructure and bootstrap of this project for typesetting. Since time is very limited, no further beautification is carried out. We can also provide users with the function of modifying personal information, such as modifying the binding email, adding the billing address, and displaying the monthly and annual household carbon footprint.

### ○ Delete a household account

On the management page, the admin or staff account can delete a household account if accepting a request for cancellation of an account. This function is system optimization and has little impact on customer experience. Therefore, it has low priority and can be implemented in subsequent development. We will provide searching for a specific household account using the unique userID, and the admin or staff account can view the detailed information of the household account, and the household to be canceled will also be shown on this page. If the admin or staff account chooses to delete the account, an email will be sent to the user's email address and will clear the record in the database.

### ○ Delete a staff account

On the management page, the admin account can delete a staff account. This function is system optimization and has little impact on customer experience. Therefore, it has low priority and can be implemented in subsequent development. We will provide searching for a specific staff account using the unique userID, and the admin account can view the detailed information of the staff account. If the admin account chooses to delete the account, the record in the database will also be deleted.

### ○ Manage country list page

On the management page, the admin account can add or delete information of countries in the donation list. This function is system optimization and has little impact on customer experience. Therefore, it has low priority and can be implemented in subsequent development. The information of countries in the donation list will be displayed on the management page and provide the add and delete button. When the button is clicked, the corresponding country information is added or deleted in the database.

## Screenshots of relevant pages

Screenshots of our system are provided with some explanation about each page. See Appendix A.

## Demo video:

our team prepared a demo video for the system. It covers the basic functions of the system in only 3 minutes. The video can be viewed from the following link.

https://git.shefcompsci.org.uk/com6103-2021-22/team18/project/-/tree/main/video

# Conclusion

## Challenges:

### Backend to front-end presentation of API:

Once the backend developer has developed the web API, they need to let the front-end developer know the name of the web API. At first, we engorged the front-end developer to read the beginning of the back-end code to understand what parameters the backend needed to fetch, but this was not efficient because the format of the fetching section of the back-end code was not entirely consistent due to the different development habits of different developers. These formatting inconsistencies took some time for the developer to explain.

This problem has been solved by using the team feature of Postman, where we create a postman team workspace, and when a back-end developer finishes a feature, he first tests it using postman and the saved test results can be seen by the team members in the same group. So, by using this method, the team solves the job of testing the API and presenting the API at the same time.

### Synchronizing databases:

In the early stages of development, the structure of the database is frequently changed. This makes it necessary to synchronize everyone's database structure and data. The first thing we thought of was whether we could synchronize the database via git in the same way as the code, but we soon realized that this would not work, firstly because it was difficult to change the location of the data, which was stored in a single folder. Secondly, the files are not split by table or by the library, so synchronizing all the database files is not a good option.

To solve this problem, we went on to look at several sources. We found that we could use a unified database server, but this seemed to cost some money, and setting up such a database would take some time.

In the end, we opted to solve this problem by manually exporting and importing SQL files, and we used a uniform naming convention to sort the exported database versions (date plus version). Developers can check if the latest database version is available after syncing the code repository with git, and if so, allow Navicat to import it. Now, this method still requires some extra effort and can be fixed later with some automatic detection scripts.

### The initial web architecture

The original web architecture structures: we looked at a lot of templates and common components, like the bootstrap, OwlCarousel, superfish, jquery, and so on. After a long discussion, we decided to use bootstrap to make a unified Nav bar. This is not easy, and it involves many problems with CSS. For example, admin and staff accounts have two more buttons than ordinary household accounts, which means that we must add condition judgment in JS. To adjust the correct position of buttons in the Nav bar. As for the design of the bottom bar of each web page, we did not use position:fixed to fix it, but determined whether the bottom bar needs to be fixed by judging the scrollHeight.

### Storage and use of usernames

The idea of our website is to let users register first, and then use the registered username and password to log in. After login, it will automatically jump to the home page. When jumping to the home page, the username of the login user should also be displayed next to the login icon.

At first, I planned to get the username from the database, but it was very troublesome, and there was no database at that time, so I discussed it with my partner. Finally, I decided to use the window. The sessionStorage. SetItem (" userName ") way to username in the local, at the same time, deposit and userType and userId.

After that, when we want to use username or usertype, we can use the code window.sessionStorage.getItem.

## Design of the logout button and function ：

At first, when the user logout button was clicked, the site only cleared the username in localStorage, which caused a typographical bug in the nav bar after the admin and staff users logged in. Log in as the household user and you will find that the nav bar still has "admin" and "report" buttons (which are unique to the admin user). The initial method is to write some criteria and refresh the page. Later after a discussion on the logout button function adds a window. The sessionStorage. RemoveItem (" userType "); And the window. The sessionStorage. RemoveItem (" userID "); That's the perfect solution.

## CSS problem

We had a lot of problems with CSS at the beginning of the project. For example, at the beginning of the home Page, we couldn't arrange the images and text boxes neatly and aesthetically. We usually wrote a lot of code, such as margin-left:30px; This was a bad way to move divs around. Later we used the bootstrap method, such as the "col-MD-8" grid system. This was very helpful, especially on the country list page.

## My Account page with feature create

This is a dynamic page. Before making it, we considered whether to write.js and.ejs separately and finally decided to keep it the same way as before. (page from js) the difficult point lies in the calculation of Carbon footprint savings (kg) must call two web API, respectively is URL: 'http://localhost:3000/countrylist' and URL: 'http://localhost:3000/userPage', because want to calculate the Carbon footprint savings (kg), we must get each user transaction records in the corresponding country country_id panel_amount and donations, Then use country_id to get the corresponding country's carbon_saving_factor in countrylist API , Finally, use panel_amount * carbon_saving_factor to get the Carbon footprint savings(kg) for each donation record. The sum of the Carbon footprint savings(kg) from all the donations is the sum of the individual user's Carbon footprint savings(kg). Calling both API at the same time is a bit of a problem because we have to run the function getSavingFootprint() to calculate Carbon footprint savings(kg) when the API call succeeds, but we don't know which API will succeed first, Our original solution was to add "if(result) getSavingFootprint();" to both calls. But after a group discussion, I decided that using axios was the best way to solve the problem, otherwise, when you have to call 3 or more APIs, it will not solve the problem.

There were also initial challenges in dynamically writing every user's transaction history. Due to my lack of experience, I didn't use jquery to write HTML at the beginning, which led to very redundant code, and it was very easy to make mistakes to write the table one by one. Subsequently, WE easily wrote the historical donation record information we needed by using jquery+for loop.

## For the use of the “!important;”

When the public component conflicts with the CSS code written by myself, I often use the "!important;" declaration method to make my CSS code take effect, but then I found out that this is not correct, and it took a long time. Time to find why some text or divs on the main page are not styled correctly. So I think:

a. Be sure to use the priority of style rules to solve the problem instead of !important;

b. Only use “!important;” on specific pages that need to override site-wide or external CSS

c. Never use !important in plugins

d. Never use !important in site-wide CSS code

## Ensure the status is synchronized with the UI

For non-static pages, the biggest challenge is how to dynamically obtain back-end data and user operations, and display them correctly on the page.

For getting back-end data, the front-end interacts with the backend through the API to obtain the data stored in the database. We use ajax to call the API and display the data returned by the API on the page. The first problem we encountered is that jQuery cannot directly use the for loop in HTML to show the contents that need to traverse the array. Through consulting the materials, we found that we need to use the loop in JS through the ID selector to dynamically append the HTML sentences to the HTML file. We believe that this inconvenience is due to the framework we have chosen.

Then for getting user action. jQuery is very limited in responding to user operations, mostly responding through onclick events. This response also requires frequent operation of DOM, which makes developers not only need to pay attention to business logic, but also need to focus a lot on operating DOM, which may cause rearrangement and redrawing, and will be accompanied by a huge cost of rendering.

Therefore, we can see that the above two inconvenient problems are caused by not choosing a better framework. We should choose a framework more suitable for front-end development. This not only reduces the workload but also increases the fault tolerance.

## Component-based Development

At the beginning of our development, we didn't realize the importance of component-based development. For example, most of our pages have navigation bars at the top and bottom. Instead of writing them in a public HTML file, we add the same codes to each page. This not only increases our development workload and produces many redundant codes, but also makes us need to modify these same codes in each file when making the same changes, which may lead to omissions.

Then we realized this problem in the development and wrote some common components. For example, we added the alert prompt box and toast prompt message using common components. We just need to import these public components on the page and pass in the prompt text. Additionally, we can also add some style controls for the prompt box in different scenarios, although we don’t need to implement it this time.

Similarly, some functions of calling API and methods of calculation can also be written in public JavaScript files. Just import these JS files and we can call the same methods on different pages. It can reduce code redundancy and increase the maintainability of the code.

## The storage and transfer of JSON objects

On the donation basket page, we need to record the items added or reduced by the user, so that the next time the user enters this page, he or she can view the items previously selected. In the beginning, we intended to transfer the data of the basket to the back-end through the API and store them in the database, so that users can still see the previous choices when logging on to other devices. However, we realized that users may frequently add or delete items in the basket, which makes us need to call the API for every operation of users, which has high requirements for the performance of our platform. For example, if many customers are operating the basket in the meantime, this API will be highly concurrently called. After discussing with the back-end partners, we decided to temporarily store the user information and basket data in the front-end using window.localstorage and compare it with the user's current login account. When it is the same account, the content in the basket will be displayed, so that the user will see the previous selection when logging in to the same account on the same device. We can also choose a better front-end storage method, such as indexDB.

When we transferred the data of the basket selected by the user to the backend, we also encountered another problem of passing JSON objects. The front-end takes the number of each country and solar panels selected by the user as a JSON object, and stores the multiple JSON objects in the array. When the front-end directly passes this array to the back-end,

the back end cannot correctly parse the data. Through the console, we found that the data received by the backend is actually in the format of [object, object]. Then we converted the data of each basket into JSON string format at the front-end, and then passed it to the back-end. Then we had a new problem. When there is only one piece of data in the basket, although the front-end converts it into string format and puts it into the array, after passing it to the back end, the syntax parser will directly convert the array containing only one string into string instead of array, so the back end needs to deal with this special case. Finally, we converted the whole basket array into JSON string at the front-end, and then parsed it into JSON object after receiving the passed value at the back-end, the problem was solved.

# Appendix

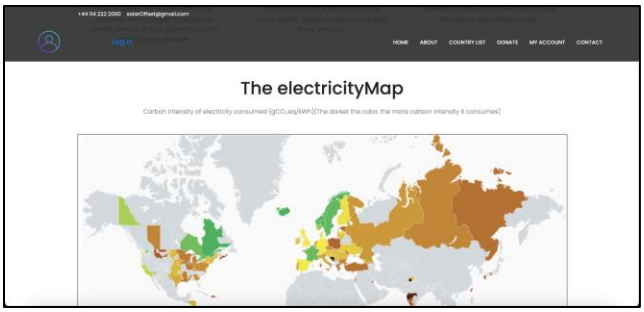
- Appendix A: User guide
- Appendix B: Setup guide
- Appendix C: Meeting Minutes Samples.

## *Appendix A: User guide*

### *Front page*

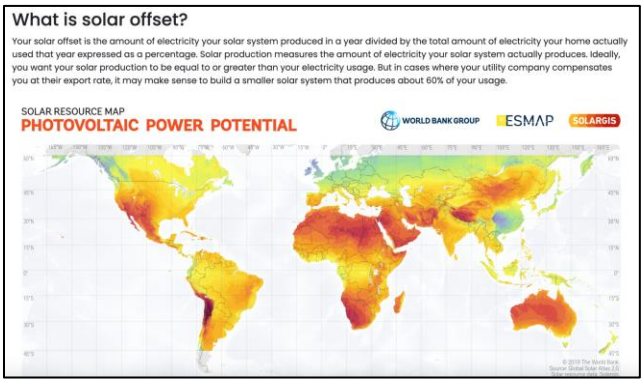
Click on the link “Get Started” and you will be taken to our Countrylist interface to donate to your preferred country.


Scroll down and you'll see the *electricityMap page*:
The electricityMap page is an external internet link that will provide you with detailed real-time carbon intensity for most European and North American countries, which represents the carbon dioxide emited per kilowatt hour of electricity produced.




### *Background Page:*

Click on “ABOUT” button, you will see thee background story, which shows you what we wanted to build this site for. Through this interface, you will deeply realize our purpose and the importance of protecting the environment of our planet.



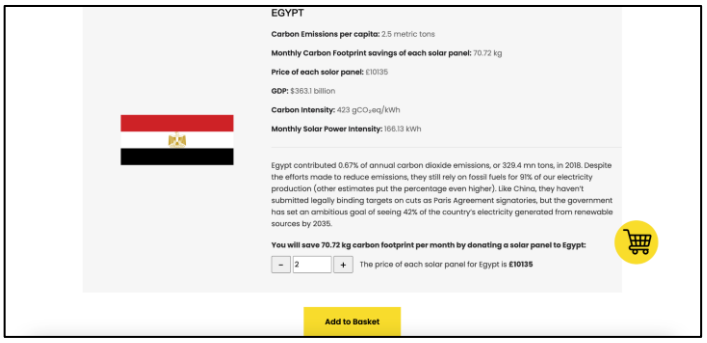

### *Country List:*

Then, in addition to the “Get Started” button mentioned earlier, you can also click on COUNTRYLIST in the upper right corner of the page to enter Country List interface. You can use it to browse the detailed information we provide on the current environmental issues facing all countries, and choose your preference countries to donate. In order to ensure that you have enough choices and ranges, we have designed a sorting function for you according to different parameters.

If you want to know in which country a single solar panel saves the most carbon emissions, you can select "sort by Carbon Footprint savings of each solar panel" and click the down arrow for descending order. As you can see in the picture, Egypt leads the pack with a figure of 70.72kg/per panel.

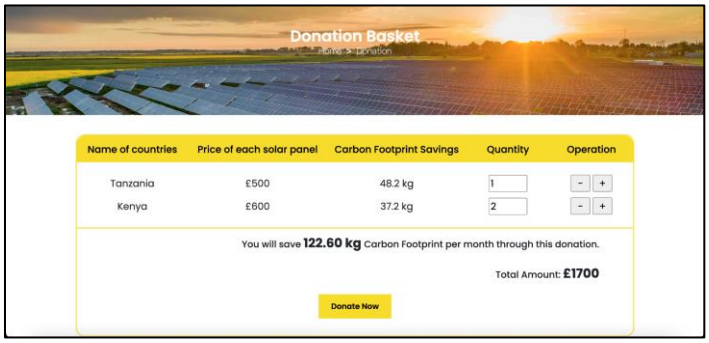
### *Country Detail:*

Clicking the country and it will take you to the page that displays more detailed information about the country. After you know the country well enough and decide to make you contribution to it, you can choose the number of solar panels and add them to your basket. (This step and subsequent functions can only be operated after logging in)




### *Basket & Pay Successfully:*

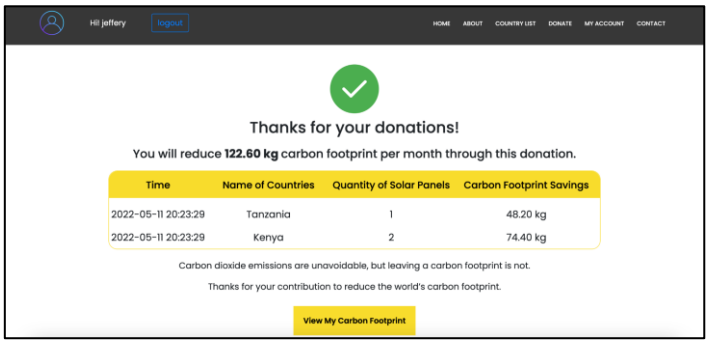
In the shopping basket page, you will pay for the amount you need to donate. You will jump to the paypal (sandbox) payment page for you. Once your payment success, we will record all the information of this donation for you. You can also access your Personal Information Center by clicking on "View my Carbon Footprint", which can also be accessed via "MY ACCOUNT".

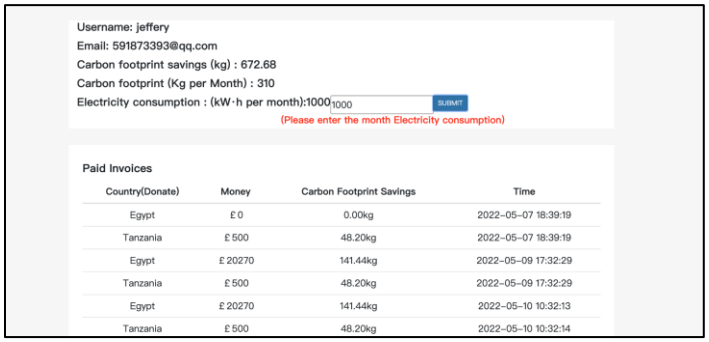



### *Personal Center:*

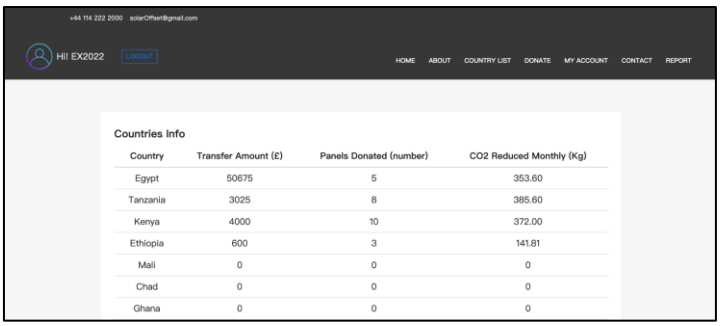
In the personal center page, you can browse all the current historical transaction records of your account. In the 'carbon footprint savings' section, we will create a summary of your current contribution to solar energy projects, calculating the total amount of carbon dioxide emissions that your donated solar panels save the world each month.

As a household, your home's carbon footprint can be calculated by multiplying your monthly electricity use by the carbon intensity of the UK area. By comparing your personal contribution to carbon emissions, you will know if you have achieved carbon balance, which is the final environment protection goal of yourself..



In our website, there are three types of users according to their authority: household (public user), staff and administer. There is an additional "REPORT" button in the staff's homepage. And on this basic, the admin's homepage has a more “ADMIN” button which is used for user management.



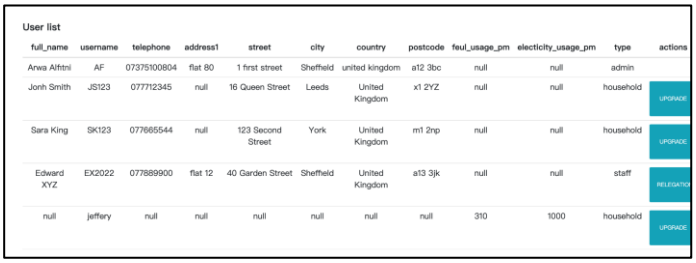

| Country | Transfer Amount (£) | Panels Donated (number) | CO2 Reduced Monthly (Kg) |
|---|---|---|---|
| Egypt | 50675 | 5 | 353.60 |
| Tanzania | 3025 | 8 | 385.60 |
| Kenya | 4000 | 10 | 372.00 |
| Ethiopia | 600 | 3 | 141.81 |
| Mali | 0 | 0 | 0 |
| Chad | 0 | 0 | 0 |
| Ghana | 0 | 0 | 0 |

### *Staff Center:*

In the staff page, we calculate for you the total number of donations currently received, the total number of panels, and the sum of the carbon footprint savings for each country.

### *Admin Center:*

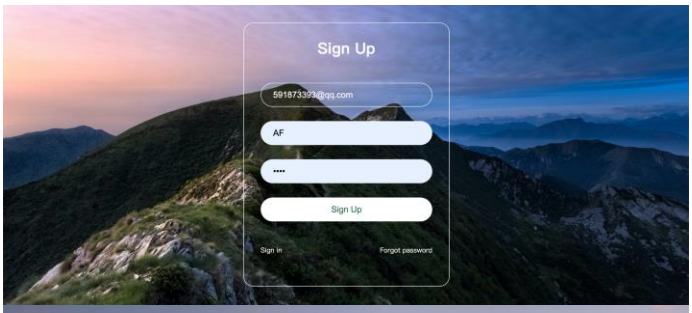
Admins have administrative privileges in addition to browsing reports of each country. The permissions of household and staff can be upgraded and relegate in the administrator page.

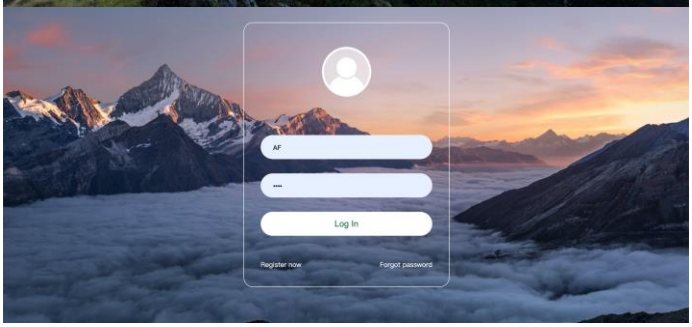



| full_name | username | telephone | address1 | street | city | country | postcode | feul_usage_pm | electicity_usage_pm | type | actions |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Arwa Alfitni | AF | 07375100804 | flat 80 | 1 first street | Sheffield | united kingdom | a12 3bc | null | null | admin | |
| Jonh Smith | JS123 | 077712345 | null | 16 Queen Street | Leeds | United Kingdom | x1 2YZ | null | null | household | UPGRADE |
| Sara King | SK123 | 077665544 | null | 123 Second Street | York | United Kingdom | m1 2np | null | null | household | UPGRADE |
| Edward XYZ | EX2022 | 077889900 | flat 12 | 40 Garden Street | Sheffield | United Kingdom | a13 3jk | null | null | staff | RELEGATIO |
| null | jeffery | null | null | null | null | null | null | 310 | 1000 | household | UPGRADE |

### *Register/Login function*

In your registration interface, you will be asked to enter your email, username and password. If the email address you entered is not registered, you will receive an email with a verification code through it. After entering the correct verification code in the verification field, you and your account will be called one of our website official users as a household.



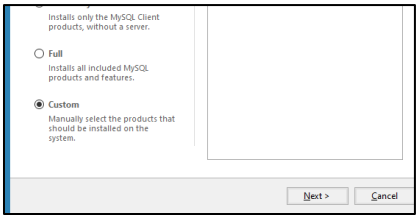
## *Appendix B: Setup guide*

This guide describes how to run this project on the windows PC.

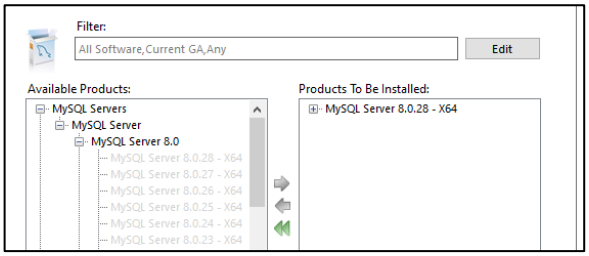
### *1.Install the MySQL database*

Download the MySQL Community version on mysql's website, get the MySQL Installer-Community.exe, and run the exe file.

Click add ... ->select custom->next

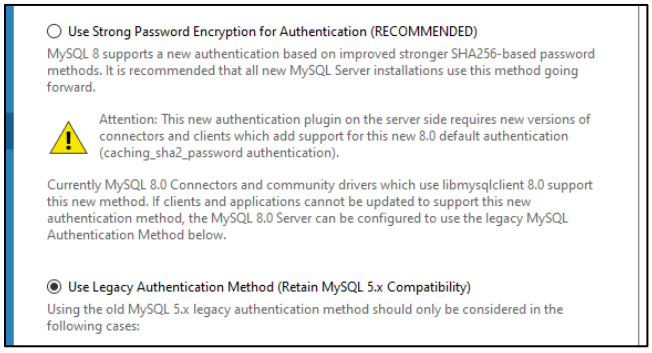




Add mySQLserver8.0 to the installation list- -> next




Continue to Click next until the page below-> select below (use Legacy version of the password)-> next->

Set the password to 234567891




Continue with Click next until the end of this installation.

### *2.Initialize the database data*

Create a database named sunlight in the database.

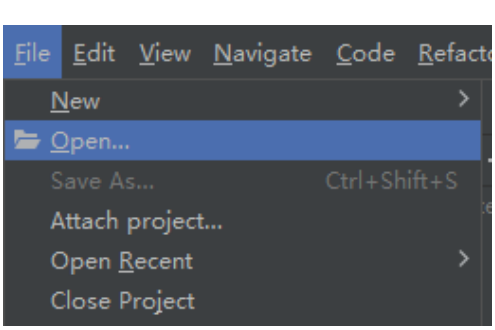
Run the SunLight.sql file in the Final Mysql DB in the command line in the sunlight database in the project.

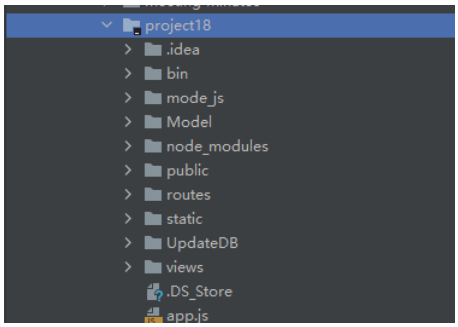
### *3.Run the project*

To simplify the set up process, here, use the ide to run the project without using the server

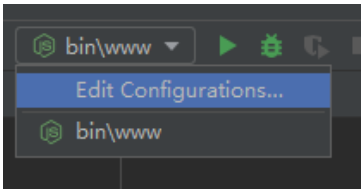
Download and install the WebStome.

Open WebStome->File-> Open opens the engineering folder.

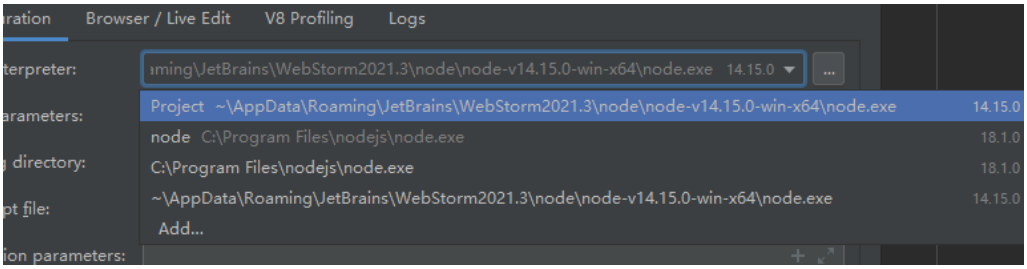


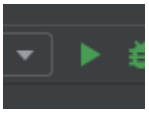

Click on the top right corner Edit Configurations.



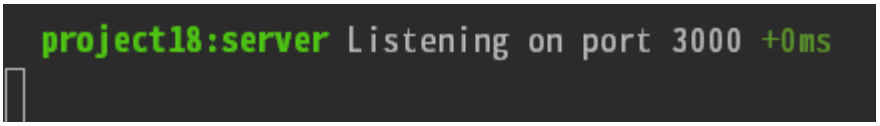

Select the node.Js for version 14.15.0. Click OK to save the configuration.



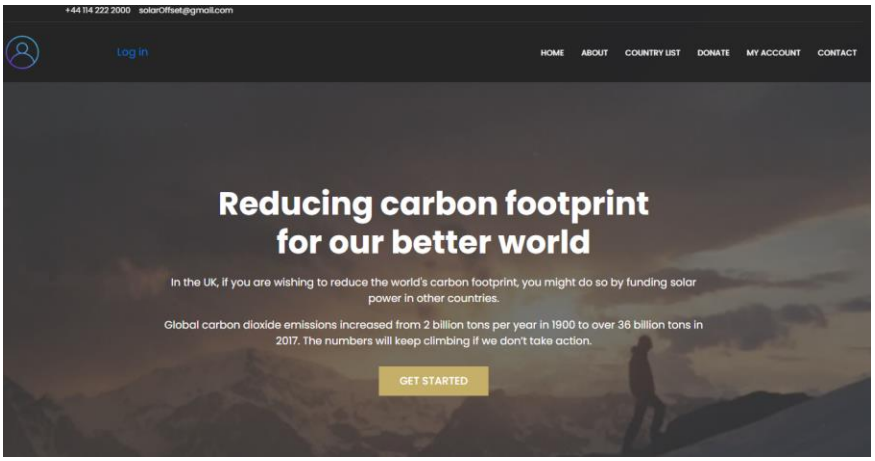

Click on the upper right Green triangle to run the project



If the output box reports no error, the project runs successfully.




Enter http: / / localhost: 3000 to open the website.



*First iteration – week 3 to 5:*

## Meeting Minutes Sample 1

| | | | |
|---|---|---|---|
| **Subject** | **Team Software Project Meeting** | **Date** | **8th February 2021** |
| **Chaired** | - | **Time** | 12:00 AM – 1:00 PM |
| **Location** | Portobello Center – room C29 | **Prepared by** | Arwa Alfitni |
| **Attendees** | Arwa Alfitni, Yifei Guo, Zeyu Li, Yixiang Wang, Yue Zhou | | |

Key Points Discussed

| No. | Topic | **Highlights** |
|---|---|---|
| 1. | Introducing members | • Each member introduced him/herself in terms of technical skills and experiences.<br>• Wanying Wang is still in China and will not join us. |
| 2. | Choosing a project | • Discus all the projects provided by the clients:<br>  ○ The project Idea<br>  ○ The required technical skills<br>• A primitive list of preferences was made.<br>• The final decision is left to the next day. Each member will read all the descriptions and vote for the preferred three projects in the WeChat group. The final list will be drawn from the voting result. |
| 3. | Contacting methods | • We will use WeChat app as a primary app to contact each other and to arrange any proceeding meeting.<br>• GitHub will be used to manage the teamwork. |

Plans for the next meeting

| No. | **Description** |
|---|---|
| 1. | Prepare questions for the client (all members) |
| 2. | Chose a name for the team (all members) |

Approvals

| | |
|---|---|
| 1. | Arwa Alfitni |
| 2. | Yifei Guo |
| 3. | Zeyu Li |
| 4. | Yixiang Wang |
| 5. | Yue Zhou |

# Meeting Minutes Sample 2

| | | | |
|---|---|---|---|
| **Subject** | **Team Software Project Meeting** | **Date** | **8th March 2021** |
| **Chaired** | Yue Zhou | **Time** | 11:00 AM – 1:00 PM |
| **Location** | Portobello Center – room C29 | **Prepared by** | Arwa Alfitni |
| **Attendees** | Arwa Alfitni, Yifei Guo, Zeyu Li, Yixiang Wang, Yue Zhou | | |

Key Points Discussed

| No. | Topic | **Highlights** |
|---|---|---|
| 1. | Client meeting | • Present the system prototype to the client.<br>• Discus any thoughts on the design.<br>• The client prefers a simple way to present all countries.<br>• There isn't any need to select two countries and compare them. |
| 2. | Database | • Discus the database design and the required data.<br>• A primitive sketch design was made. |

Plans for the next meeting

| No. | **Description** |
|---|---|
| 1. | Agreed to the final database design (all members) |
| 2. | Distribute tasks between the members (all members) |

Approvals

| | |
|---|---|
| 1. | Arwa Alfitni |
| 2. | Yifei Guo |
| 3. | Zeyu Li |
| 4. | Yixiang Wang |
| 5. | Yue Zhou |

*Second iteration – week 6 to 8:*

# Meeting Minutes Sample 3

| | | | |
|---|---|---|---|
| **Subject** | **Team Software Project Meeting** | **Date** | **14th March 2021** |
| **Chaired** | - | **Time** | 5:00 PM – 9:00 PM |
| **Location** | Information Common – room 4.23 | **Prepared by** | Arwa Alfitni |
| **Attendees** | Arwa Alfitni, Yifei Guo, Zeyu Li, Yixiang Wang, Yue Zhou | | |

Key Points Discussed

| No. | Topic | **Highlights** |
|---|---|---|
| 1. | Database<br>(Arwa) | • The database was created.<br>• Discuss with the team the sources of the data. |
| 2. | Frontend<br>(Yifei & Zeyu) | • Building Log in and Register page |
| 3. | Backend<br>(Yixiang & Yue) | • Writing code to process the data in log in and register page |

Plans for the next meeting

| No. | **Description** |
|---|---|
| 1. | Connect the database to the backend (Arwa) |
| 2. | Implementing email authentication for the Register feature. (Yue) |
| 3. | Building a country list page. (Yifei & Zeyu) |
| 4. | Ranking countries based on GDP, Carbon emission, saving, or price of solar panel installation.<br>(Arwa, Yue, Yifie, Yixiang) |

Approvals

| | |
|---|---|
| 1. | Arwa Alfitni |
| 2. | Yifei Guo |
| 3. | Zeyu Li |
| 4. | Yixiang Wang |
| 5. | Yue Zhou |

# Meeting Minutes Sample 4

| | | | |
|---|---|---|---|
| **Subject** | **Team Software Project Meeting** | **Date** | **25th March 2021** |
| **Chaired** | Yue Zhou | **Time** | 4:00 PM – 8:00 PM |
| **Location** | Information Common – room 4.23 | **Prepared by** | Arwa Alfitni |
| **Attendees** | Arwa Alfitni, Yifei Guo, Zeyu Li, Yixiang Wang, Yue Zhou | | |

Key Points Discussed

| No. | Topic | **Highlights** |
|---|---|---|
| 1. | Development tools | • Using "Postman" to check the connection between the frontend, backend, and the database. (Yixiang) |
| **2.** | Front end | • Designing and building the country list page. (Yifei and Zeyu) |
| **3.** | Backend | • Ranking the country list by (gdp, panel price, carbon emmisions) (Yue) |
| **4.** | Database<br>(Arwa) | • Add two columns to country table: country_image and country_discription<br>• Insert sample data. |

Plans for the next meeting

| | |
|---|---|
| 1. | Insert, delete, and update country data – sql statements. (Arwa) |
| 2. | The frontend team will continue their work on the country list page. (Yifei & Zeyu) |
| 3. | The backend team will continue their work on the country list page. (Yixiang & Yue) |

Approvals

| | |
|---|---|
| 1. | Arwa Alfitni |
| 2. | Yifei Guo |
| 3. | Zeyu Li |
| 4. | Yixiang Wang |
| 5. | Yue Zhou |

*Third iteration – week 6 to 8:*

| Meeting Minutes Sample 5 | | | |
|---|---|---|---|
| **Subject** | **Team Software Project Meeting** | **Date** | **3rd May 2022** |
| **Chaired** | - | **Time** | 11:00 AM – 1:00 PM |
| **Location** | Portobello center –room C29 | **Prepared by** | Arwa Alfitni |
| **Attendees** | Arwa Alfitni, Yifei Guo, Zeyu Li, Yue Zhou, Yixiang Wang | | |

| Key Points Discussed | | |
|---|---|---|
| No. | Topic | **Highlights** |
| 1. | Database structure | - Update the Transaction table to include:<br>o Transaction start time.<br>o Transaction end time.<br>o Transaction status (pending, success, failed). |
| 2. | Group discussion<br>About the report | - The test plan must be written.<br>- The setup guide will include:<br>o Any specific configuration to run the project locally on the teacher's computer.<br>o Steps to manage the website.<br>- At the demonstration for next week:<br>o Provide a demo as user story from login to the send donation successfully. |
| 3. | Project documentation | - Review the Meeting minutes.<br>- Review user stories.<br>- Strat writing the documentation. |

| Plans for the next meeting | |
|---|---|
| No. | **Description** |
| 1. | Arwa will work on the testing plan. |
| 2. | The backend team will continue working on the donation procedure. (Yifei & Zeyu) |
| 3. | The frontend team will continue working on the design of the donation page. (Yue & Yixiang) |

| Approvals | |
|---|---|
| 1. | Arwa Alfitni |
| 2. | Yifei Guo |
| 3. | Zeyu Li |
| 4. | Yue Zhou |
| 5. | Yixiang Wang |

# Meeting Minutes Sample 6

| | | | |
|---|---|---|---|
| **Subject** | **Team Software Project Meeting** | **Date** | **3rd May 2022** |
| **Chaired** | - | **Time** | 11:00 AM – 1:00 PM |
| **Location** | Portobello center –room C29 | **Prepared by** | Arwa Alfitni |
| **Attendees** | Arwa Alfitni, Yifei Guo, Zeyu Li, Yue Zhou, Yixiang Wang | | |

Key Points Discussed

| No. | Topic | **Highlights** |
|---|---|---|
| 1. | Database structure | - Update the Transaction table to include:<br>  ○ Transaction start time.<br>  ○ Transaction end time.<br>  ○ Transaction status (pending, success, failed). |
| 2. | Group discussion<br>About the report | - The test plan must be written.<br>- The setup guide will include:<br>  ○ Any specific configuration to run the project locally on the teacher's computer.<br>  ○ Steps to manage the website.<br>- At the demonstration for next week:<br>  ○ Provide a demo as user story from login to the send donation successfully. |
| 3. | Project documentation | - Review the Meeting minutes.<br>- Review user stories.<br>- Strat writing the documentation. |

Plans for the next meeting

| No. | **Description** |
|---|---|
| 1. | Arwa will work on the testing plan. |
| 2. | The backend team will continue working on the donation procedure. (Yifei & Zeyu) |
| 3. | The frontend team will continue working on the design of the donation page. (Yue & Yixiang) |

Approvals

| | |
|---|---|
| 1. | Arwa Alfitni |
| 2. | Yifei Guo |
| 3. | Zeyu Li |
| 4. | Yue Zhou |
| 5. | Yixiang Wang |