# 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 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 website that helps people to reduce the carbon benefits in the world. The website will allow households 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 for donating frequently without the need to provide his/her information each time.

# 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.
* 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 member:

* Arwa Alfitni
* Yifei Guo
* Zeyu Li
* Yixiang Wang
* Yue Zhou (leader)

Product backlog

## Household User Stories:

### Information about countries:

* As a household, I prefer to see how many people are donated to a particular country.
* As a household, I prefer to be able to see how many panels are donated to a particular country.
* As a household, I am interested in 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.

### 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.

### Log in and registration:

* As a household, I want to be able to register with the minimum number of steps and required information.
* As a household, I want a privacy grantee 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 not the amount of donated money.
* 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 suggested 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 the staff accounts to some other level to give them some privileges.

Analysis & Design

System architecture:

### Programming Languages and Database:

frontend: HTML and CSS

backend: Node JS

Database: MySQL

First, we decide to use MySQL because "Arwa" who oversaw the database structure has some experience with MySQL. But, then we decide to switch to MongoDB for more easily SQL structure. After that, we decide to return to MySQL when we were trying to produce some statistics from the data.

## UML diagrams

### Use case Diagram:

Our system deals with five different types of stakeholders. Firstly, a Visitor, which is any person who visits our website. This actor can view the home page and see the country list, navigates to the detailed page of each country, and can add any number of panels to the basket. he can also log in if he has a valid username and password or can register to our website as a household. Secondly, the admin, which has some management tasks for the household and the staff accounts. Thirdly, the staff who is responsible for extracting reports about the donations and managing the country list. Then the Household, which represents a regular user who wants to donate to some country. Paypal is a third party that is used for payment authorization.

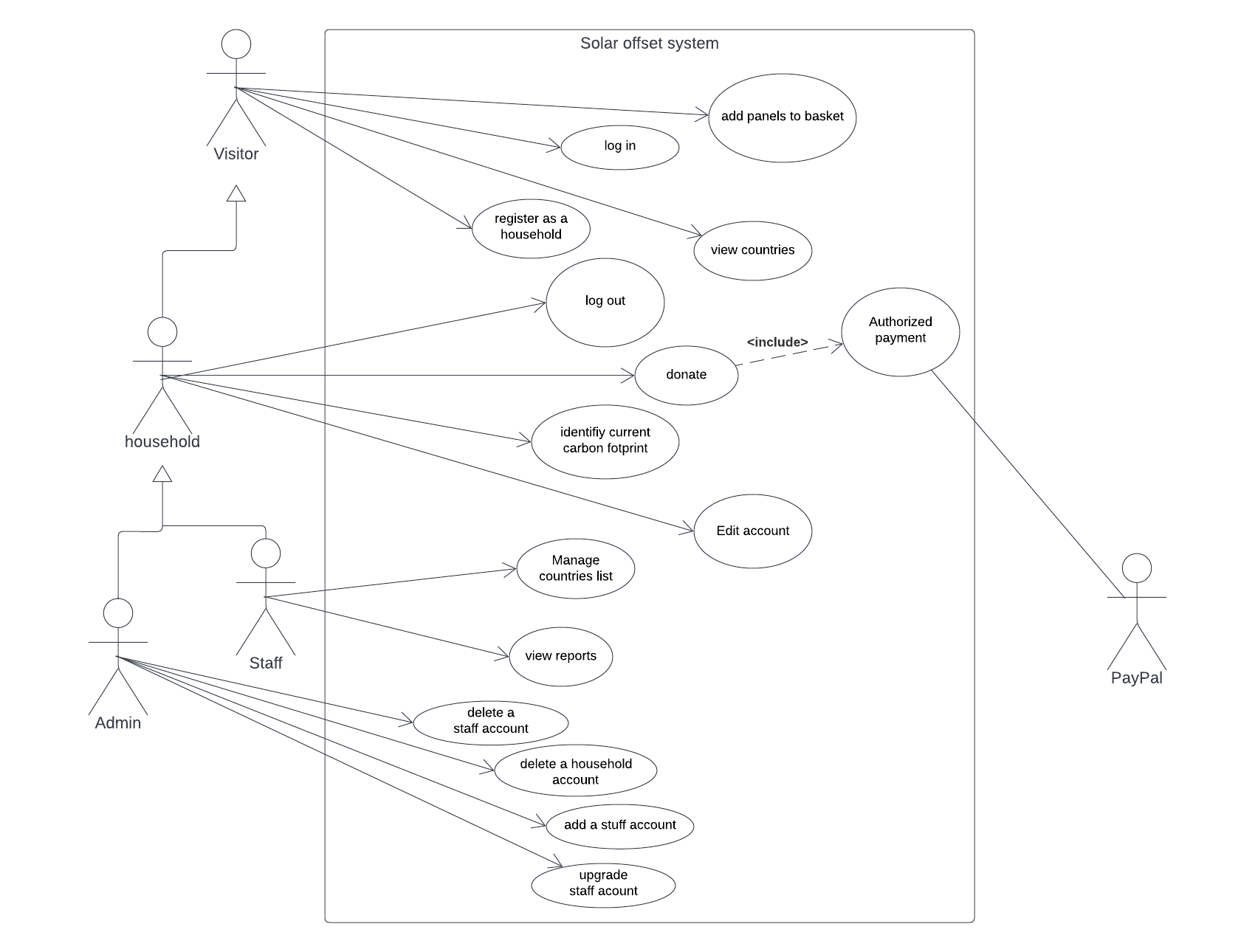


Figure 1: use case diagram

## Algorithm:

## Database design:

### Entity Relationship Diagram:

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

The User table represents all types of users which cloud be an admin, a staff, or a household. The “user type” feature distinguishes between the three types. This table contains dome personal information whereas fuel and electricity usage per month “fuel\_usage\_pm” and “electricity\_usage\_pm” respectively are used to compute the carbon usage for each person.

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 capital, and the cost of installing a solar panel. “carbon\_intensity” represents the amount of produced carbon dioxide by using electricity while “kwh/m2/mon” represents the number of kilowatt per hour produced by a squared meter solar panel each month. “carbon\_savinng\_factor” is used to calculate the impact of using the solar panels in reducing the 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 and a different number of panels for each country and then donate for 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.

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 price that 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.

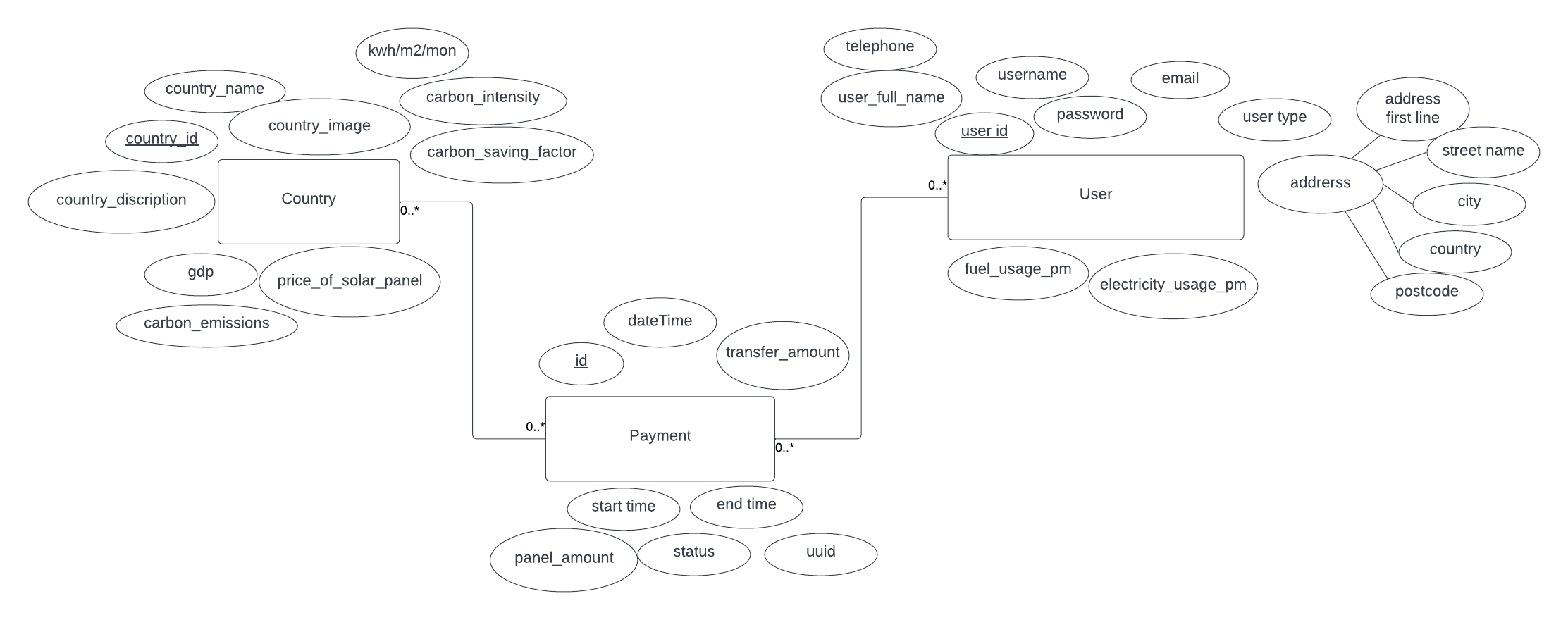


Figure 2: ERD

Evidence of Testing

## Test plan:

We did manual testing for our website. The test cases are generated from the requirements and the user stories.

### During development:

Postman was used for checking the connection between the frontend, backend, and the database.

## Test documentation:

### Test cases:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test number | Test name | Actor | Pre-condition(s) | Post-condition(s) | Steps |
| 1 | Log in | Household  Staff  admin | none | Success: homepage present with the username shown on the top-left corner.  Failure: failed message popped up | - Open the website  - Click on the login button  - Enter the username and password in the correct box  - Click login |
| 2 | Register | Household | none | Success: household account is created, and the home page is displayed with the username appease on the top-left corner.  Failed: fail message popped up. | - Open the website  - Click on the Register button  - Enter your username (email), and password in the correct box.  - an authentication code will be sent to the user email.  - Enter the code in the correct box.  - account verified and created. |
| 3 | browse the country list | visitor | none | success: present the page for all countries.  failed: “page not found” message popped up. | - Open the website  - From the top-right tabs – choose the country list |
| 4 | forget password | household |  | success: homepage present with the username shown on the top-left corner.  failed: failed message popped up. | - open the website  - click on login button  - click forget password button  - enter your email  - a new password will be sent to your email  - use the new password to login |
| 5 | donate | Household  Staff  admin | Log in | success:  Success message of donation popped up.  Transaction successfully finished. and stored at the transaction table.  The number of donated panels is updated on the relevant country detailed page.  failed:  a failed message popped up.  no transaction is stored at the transaction table.  the number of donated panels didn't change on the relevant country detailed page. |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

## Test results

|  |  |  |  |
| --- | --- | --- | --- |
| Test number | Test name | Success or Failure? | Notes |
| 1 | Login |  |  |
| 2 | Register |  |  |
| 3 | Browse country list |  |  |
| 4 | Forget password |  |  |
| 5 | donate |  |  |
|  |  |  |  |

# Team management & communication

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 participate 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.

* 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 quires. Yixiang Wang suggested exchanging our DBMS to MongoDB since this will facilitate dealing with inserting and retrieving data. 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 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: Arwa Alfitni
    - UML: Arwa Alfitni
    - Meeting Minutes: Arwa
    - Conclusion: Yixiang and Zeyu Li and Yifei Guo
  + Setup guide: Yixiang and Yue
  + User guide:
* Testing:
  + Test cases: Arwa

Communication:

We decided to share our code and documents through the GitLab repository. As a daily communication base and for arranging meetings we used WeChat. 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 hardness.

Meeting minutes samples:

Planned & Completed Features

## First iteration - week 2 - 5:

|  |  |  |
| --- | --- | --- |
| Category | Planed 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 |

## Second iteration – week 6 - 8:

|  |  |  |  |
| --- | --- | --- | --- |
| Category | Planed 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 |  |
|  |  |  |

## Third iteration – week 9 - 11:

|  |  |  |
| --- | --- | --- |
| Category | Planed task | Completed? |
| Database structure |  | Yes |
|  | 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 |
|  |  | Yes |
|  | Implementing reports at the Staff page | Yes |
|  | Admin page |  |
|  | Calculating the impact of the donation on reducing carbon dioxide. | Yes |
| Documentation | Writing the final report | Yes |
|  | User guide |  |
|  | Setup guide |  |
| Testing the system | Write test cases | Yes |
|  | Executing test case |  |
|  | Writing test result |  |

Uncompleted Features

○ What features were not completed and why?

Screenshots of relevant pages

Conclusion

## Challenges:

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

Once the backend developer has developed the interface, they need to let the front-end developer know the name of the interface. 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 interface 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 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.

Appendix

○ User guide (maximum 2 pages)

○ Setup guide (maximum 2 pages)

○ Other documents you think are relevant can be added here