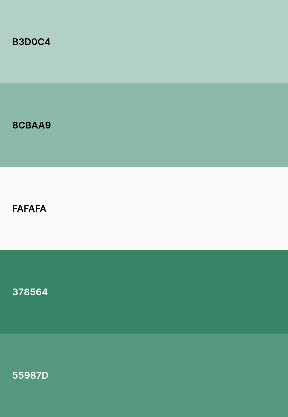
**Task 1 Proposal**

The proposal for a digital solution that will provide customers with relevant information that is up to date. This information will include green energy products that are currently on the market and information on how to reduce the carbon footprint. It will also allow for customers to schedule installations and consultation right from the application. Another feature it will also allow is for customers to calculate their carbon footprint.

I have created a logo for the Rolsa Technologies which I did using Adobe Express Logo Maker:  
  


The logo has a tree to signify their green ambition and efforts through the logo.



The following is the colour pallet for the designs. I have chosen these colours to match the logo design, and it will make a great user experience because the company is about green energy, so we need to make sure that the colours reflect that.

The application will feature the following features:

|  |  |
| --- | --- |
| **Requirement** | **Description** |
| Account registration | This allows customers to register and account and manage their data such as consultations and installation. |
| Accessibility features | These features will be added to help make it accessible to a wider range of users making it user friendly for all. |
| Energy Tracker | This feature will track the energy use of home and give suggestions and tips on how to improve the usage and give valuable insight into there usage. |
| Account Sign In | This is a feature that will let existing users of the system authenticate. |

The application will feature the basic features that an application that offers these services normally have as a standard. These features will attract potential users as people would like to view their usage to keep the costs down and if it’s made easy to view, they will attract more customers.

In term of the business, it will help speed up operations as they won’t need to go and read the data to get a usage reading it will all be digital so they can free up the staff to do some more complicated tasks.

In terms of data handling, the application will use a cloud database which is managed by a third party who are dedicated into keeping the database data safe which will be cheaper and more reliable than a local database. The platform will also use an AI personalisation feature that will identify trends in the energy usage and give you dedicated feedback.

We also understand the importance of a secure platform which is why we are taking these measures to make sure it stays this way.

|  |  |
| --- | --- |
| **Measure** | **Description** |
| Password Policy | Make sure that passwords are at the correct standard of at least 8 characters and one special character. |
| Password Hashing | Make sure that the password is hashed so that people can not understand it if they get access into the database. |
| Password Timeouts | If the user enters the password incorrectly a number of times the account will either be locked until more authentications can be provided or a selected amount of time for example 12 hours. |
| Access Control | Make sure only whitelisted ips have access to the database to prevent anyone from connecting to it. |

To comply with GDPR laws, we will offer a delete information button which will delete the user’s information within 30 days of them requesting.

**Tech Stack**

|  |  |
| --- | --- |
| **Component** | **Technology** |
| Front End | EJS (embedded JavaScript) |
| Backend | Node.js with Express.js |
| Database | Local MySQL database |
| Authentication | JSON Web Token |

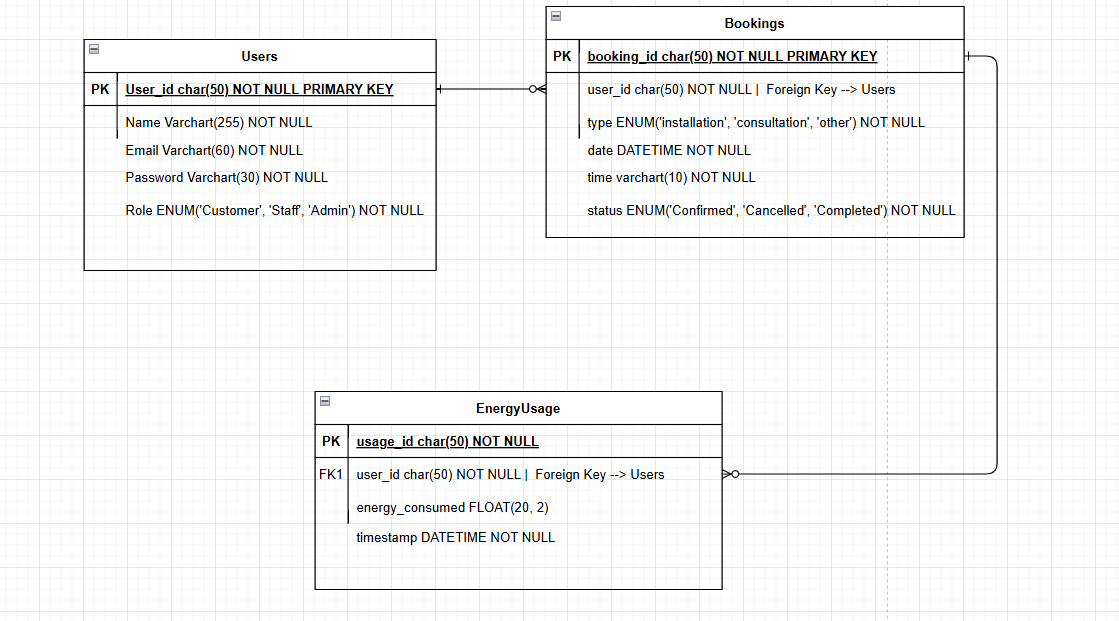
The reason for using EJS as a front end is that we can easily render JavaScript into HTML with this language and allows us to use EJS tags to embed variables onto a html page so we can make a login page easily. It’s also good for conditions so for example we need to only display a component for only admins we can do so easily with EJS.

The reason for using Node.js with Express.js for the backend is that they are good tools for using EJS with as they allow us to pass through variables to the front end to render them and they allow us to render the pages so we can do login and dashboard easily.

The reason for using a local MySQL database is that the stored storage is local on the server which allows only us to control the data which is better because then no third party has access to the data which is more secure.

I have decided for authentication to use JSON web token which is a standard when choosing this type of tech stack as it offers great security and is used by businesses all over the world.

**Database Structure**



The database is structured in a way that it doesn’t have many tables, but we can have the information we need in there. Users table is the table that will store all the customers login information. We will use bcrypt for password hashing to make sure it’s secure. There is a role value that will determine what the user is and what permissions they should get.

The booking table will help the staff and admins know what booking they have at what type and the status of the booking to see if it has been complete by the team.

Energy usage table is the final table of the database and tells the customer their energy that they are using and it is linked to their user\_id.

**API Structure**

|  |  |  |
| --- | --- | --- |
| Endpoint | Method | Description |
| /api/auth/register | POST | This will send a post request that will add the user to the database and then they will be registered. |
| /api/auth/login | POST | This post request will send the request to the database and make sure that the username and password are correct then it will authenticate the user. |
| /api/bookings | GET | This will retrieve the authenticated users bookings. |
| /api/bookings/:id | DELETE | Cancel a booking |
| /api/energy-usage | GET | Get the authenticated users energy usage. |
| /api/admin/report | GET | This will get the data for a admin report. |
| /api/book | POST | This will send the booking to the database and create a booking. |
| /api/bookings/:id | PUT | Update a booking |

This API structure will give the front end everything it needs to access the data to the backend. It has authenticated routes so they are protected so only the authenticated user can access them which is good API security. Admins will be able to delete, add and edit new bookings and the customer will be able to edit their own booking as well.