COSC 499

Capstone Software Engineering Project

2022-2023 Winter Term 2

Final Report

Multiple Non-Profits Group B

Created by:

Abdulaziz Almutlaq - 79960175
Maysey Lu - 19226646
Jordan Onwuvuche - 61007530
Harman Sahota - 28337426

Table of Contents

Overview

FoodSaviour is an online web application that is designed to help users record, calculate, and visualize the amount of food waste they have diverted to other sources besides the landfill. The motivation for this software development was to make it easier for organizations to process their large amounts of data. This application can also be used to connect with other organizations that are part of the FoodSaviour network to further reduce food waste.

Target User Groups

Non-Profit Organizations

The primary users of this software and the motivator for this project was non-profit organizations as many of these organizations do not have enough resources and/or man-power to complete some of their necessary tasks.

Example: A non-profit organization that focuses on finding ways to reduce food waste needs to apply for government grants to fund their projects. The organization can use this software to help generate visuals to support their inventory data that show that they have made a positive impact/progress on reducing food waste. These visuals can then be used as supporting documentation in their grant application.

Small Businesses

Much like non-profit organizations, small (and local) businesses may not have the resources to access other third-party softwares that complete the same tasks as ours. Or, they may not need as advanced software that has many different features, but want something simple and easy to use.

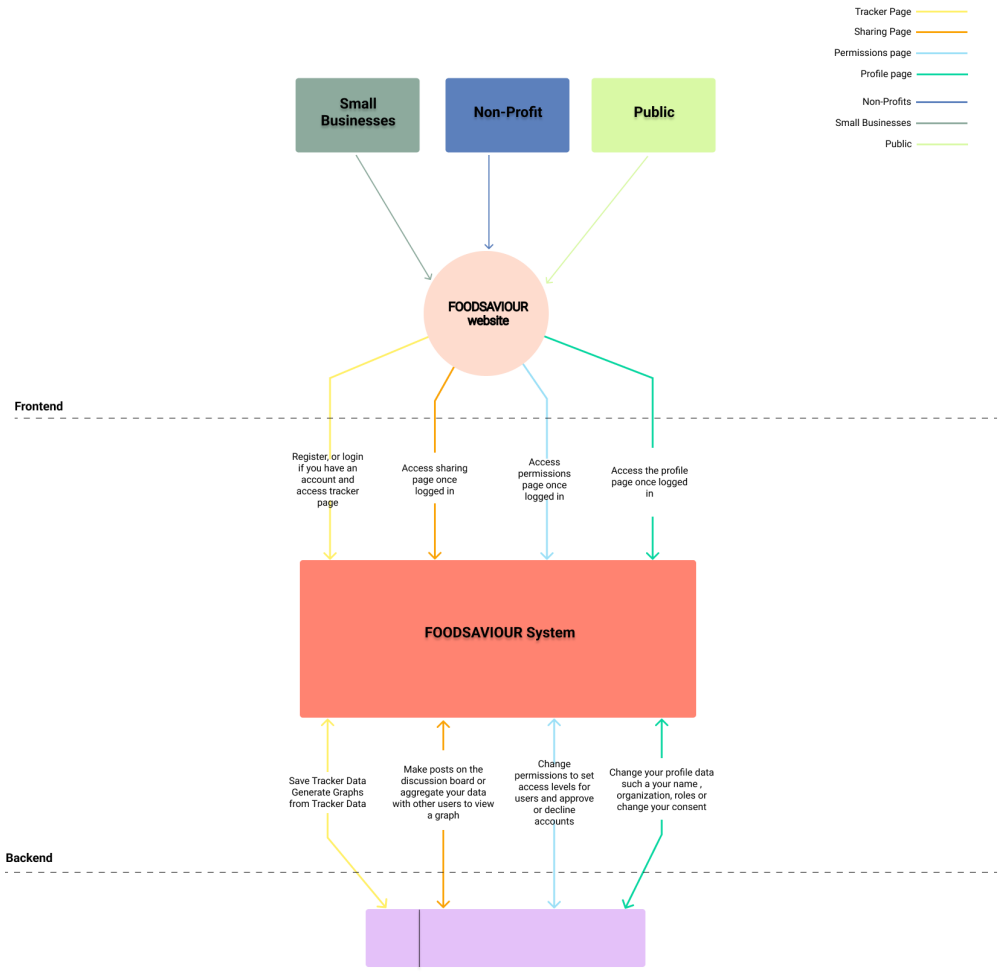
Example: A small, local farm wants to reduce the amount of crop that goes to waste at the end of a season. Due to inflation, or other external factors, the farmers were not able to sell as much of their crop as they did the previous years, leaving them with an abundance that would go bad if not dealt with soon. The farmer can use the software to connect with the FoodSaviour community to find alternative ways for their excess crop to be used so that it is not wasted.

General Public

Because any user on the internet can access this website, and there are no crucial requirements that limit a user from creating a FoodSaviour account, this software can also be used by the general public.

Example: Low-income families may want to utilize the website's public sharing page. Due to inflation, which has caused restaurants and grocery stores to increase their prices, it has become increasingly more difficult to purchase food to feed enough people and remain within a reasonable budget. Thus, families can use the public sharing page to see if there are any food items that they may receive from the FoodSaviour community for free or at a discounted price.

Data Flow Diagrams (DFD's)



DFD LEVEL 0

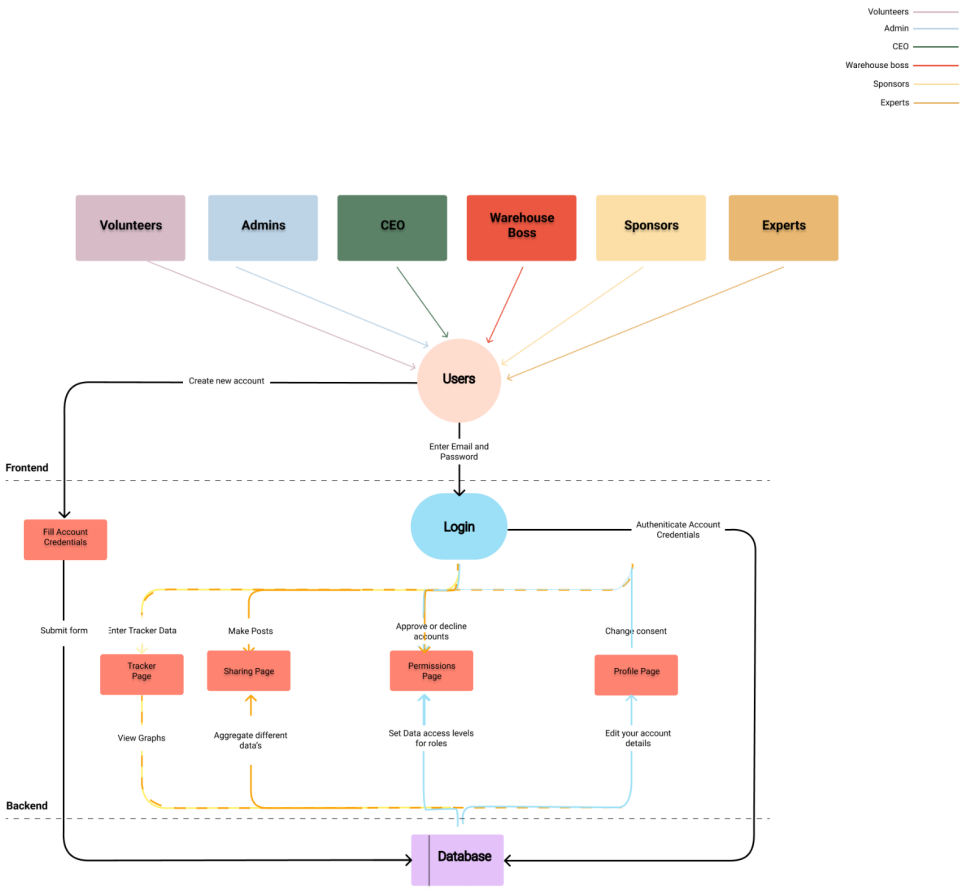
Users of Foodsaviour will interact with the Foodsaviour system to perform various actions depending on the type of user they are, such as small businesses, non-profits, or volunteers.

All users follow a general streamlined process. They start by creating an account on the website, and upon completion, the first in-system page they gain access to is the tracker page. On this page, users can enter and save data to the database, which is then used to automatically generate graphs on the same page.

The next page users have access to is the sharing page. On this page, users can view and create posts on the discussion board. Additionally, they can aggregate their own data or their organization's data with another user or organization, and graphs will be displayed as a result of this aggregation.

Users can also access the permissions page, which gives them admin controls such as the ability to change permissions for different roles within the organization to provide appropriate data access levels. They also have control over who can access the sharing page and who can read/write data in the tracker page. Another functionality of this page is that users can approve or decline access requests from specific users who are requesting access to the organization's data.

Lastly, users gain access to a profile page that displays all essential information about their account, such as their first name, last name, email, organization, roles, and consent. This page also allows users to edit most of this information except for their email. If a user disagrees with the consent, their account will be deleted immediately.



DFD LEVEL 1

There are six types of users, each representing a different role in the organization that the website is intended for: Volunteers, Admins, CEOs, Warehouse Bosses, Sponsors, and Experts.

The login process involves the user entering their email and password. This data is then queried in the database where it is authenticated to check if the user exists in the database. If the account exists and the correct email-password combination was entered, the user can log into the website. However, if the user entered wrong information or the account does not exist an error is displayed to the user.

If the user does not already have an account, they can proceed to the registration page and provide their credentials, select their desired roles, and agree to the consent to create an account.

After the user has logged in successfully or created an account they are able to access four main pages of the system - Tracker page, Sharing page, Permissions page and Profile page. Each of these pages have important features of the system.

On the tracker page, you can fill out a form, which lets you enter details such as the total quantity of a product, its category (e.g., Fresh Produce, Meat, Bread, Canned Food, Dairy, or Reclaimed), and the quantities diverted to different sources such as Clients, Partner Network, Animal Feed, Compost. The system automatically calculates the amount that went to landfill by subtracting the total quantity diverted to other sources from the total quantity of the product entered.

After you submit the form, the system automatically converts these values to percentages and stores them in the database. These percentage values are then used to generate two pie charts on the page, which provide a high-level analysis of the data. The first pie chart plots the amount diverted to each source, while the second one plots the total amount in each different category.

On the sharing page, users can view and create posts on the discussion board. Additionally, they can aggregate their own data or their organization's data with another user or organization, and graphs will be displayed as a result of this aggregation.

Users can also access the permissions page, which gives them admin controls such as the ability to change permissions for different roles within the organization to provide appropriate data access levels. They also have control over who can access the sharing page and who can read/write data in the tracker page. Another functionality of this page is that users can approve or decline access requests from specific users who are requesting access to the organization's data.

Lastly, users gain access to a profile page that displays all essential information about their account, such as their first name, last name, email, organization, roles, and consent. This page also allows users to edit most of this information except for their email. If a user disagrees with the consent, their account will be deleted immediately.

All these pages communicate with the database to perform varying operations.

Functional Requirements

Out-of-System Requirements

These are the requirements associated with the pages that do not require the user to be logged in.

Page	Requirement	Additional
Home	- Large icons used to make navigation around pages easy as many of the users may be older and/or may not be familiar with using technology - Navigation bar to navigate to login,	

	register, and sharing page	

Register	- Registration form must require the user to provide the following information and have it saved in the database:
	  - Full Name
	  - Email
	  - Password
	  - Organization Name
	  - Organization Role
	  - Consent	

Login	- Allow the user to enter email and password to login - Notify the user if the wrong login credentials were entered	
Public Sharing	- Allow public (non-logged in users) to view the public sharing page - Posts should only include the ones that organizations have allowed to be public and posts created by public users - Non-logged in users can create posts on the sharing page if they provide an email address for contact	
Reset Password (Optional feature)		- Allow the user to enter their email to reset their password if they forgot - An email is sent to the entered email with a link to allow them to reset their password
FAQ (Optional feature)		- Users can view step-by-step instructions on how features work on each page

In-System Requirements

These are the requirements associated with the pages that users can access once they've logged into the website.

Page	Requirements	Additional

Tracker	- The user must be able to enter data information:
	  - Category
	  - Description*
	  - Total amount
	  - Amount diverted to each source
	- Generate and view a graph based on the data entered
		In additional to the requirements, we have implemented:
		- Inputted data is automatically calculated into percentages
		- View, edit, and delete data in the database
		- Export the data as a CSV file

Sharing	- View aggregated data graphs - Allow users to create a post - Posts should display contact information of the poster - Differentiate between sharing and receiving posts - When creating a post, allow poster to decide if it can be viewed by the public	In addition to the requirements, we have implemented: - Asynchronous search - Filter by email or product - Sort by time (exclusive)
Permissions	- Set permissions for each organization role to determine their level of access - Approve/disapprove pending accounts	In addition to the requirements, we have implemented: - Asynchronous update of pending accounts - Change accounts from approved to disapproved and vice versa
Profile (Optional feature)	- View account credentials	In addition to the requirements, we have implemented: - Update name, password, and roles - Update consent. If the user removes consent, the account and associated data will be deleted.
404 Page (Optional feature)		- As a security measure, a 404 error page is displayed when an unauthorized user attempts to access an in-system page without

		loggin in.

Technical Specifications

Frontend

- Built using React.js (version 18.2.0) as the frontend framework.
- Uses HTML, CSS, and JavaScript to create the user interface as part of react default JSX.
- Uses asynchronous updates to prevent refreshing after every task to show updates, this is done via react packages like '*axios' (version 1.3.2))* which is also used to communicate with the backend, other packages which help are the inbuilt *'useEffect, useState'* packages in react.
- Uses packages like Bootstrap (version 5.2.3), Font Awesome (version 0.2.0) to provide aesthetically pleasing design.
- Uses d3.js (version 7.8.2) to handle the charting of data.
- Figma for designing.

We decided to use HTML and CSS as they were the building blocks of any website. Additionally, we used Bootstrap with CSS to ensure that our website was responsive enough to run on any device, regardless of the screen width. JavaScript was used to validate form inputs, preventing data attacks such as SQL injections.

We chose Django over other frameworks like FastAPI and Flask due to its ease of coding, updated documentation, and compatibility with HTML. The client believed that Django had all the functionality required for the website she wanted us to build, while other frameworks may have been lacking in some areas. It also provided a wealth of built-in libraries and APIs that assisted us in our project. We selected d3.js over other frameworks like chart.js because our client and we agreed that it offered more chart plotting options.

We used Figma for designing as it allowed us to work collaboratively on the designs. Moreover, we created design prototypes in Figma, which was beneficial when showing these designs to the client since they could interact with the prototype rather than merely imagining how navigating through the pages would work.

Backend

- Built using Django Rest Framework(version 3.14.0) as the backend framework (Django version 4.1.3).
- Uses MySQL(version 8.0.30) as the database management system.
- Implements a RESTful API to provide communication between the frontend and

backend.
- Uses Django's ORM to manage the database and facilitate database migrations
- Uses Django's inbuilt token generating system to generate tokens which provide more security .
- Uses Django's default 'make_password' to hash the password before storing it in the database.
- Uses Django's default 'send_mail' to send password reset emails.
- Use Django package django-cors-headers (version 3.13.0) to avoid errors between the frontend and backend communication due to cors headers

We decided to use a MySQL database as instructed by our client. Additionally, we chose to use the rest framework integrated into Django to add an extra layer of security to the database connection via the front-end. Furthermore, we utilized many built-in frameworks inside Django, such as the database migrator, token system, make_password, and send_email to perform various tasks. We chose these tools as it was most convenient to use the built-in technology within our already selected backend stack. Lastly, we used django built in package django-cors-headers to provide smooth, error free communication between the frontend and the backend.

Miscellaneous

Once completed, it should be uploaded and run on a Linux virtual machine where it can be hosted to a server and accessed via URL.

Software Implementation

Testing

Github Actions

Continuous Integration (CI) was used to test the code on every push, using a YAML workflow created by us. This workflow tested the code for errors and ensured proper formatting. By using CI, we were able to catch issues early in the development cycle, ensuring a stable and functional codebase for our application.

Manual Testing

Due to the time constraint, our team decided to focus more on implementing as many of the features as possible rather than researching, learning, and understanding how to use a software testing tool and writing the tests. Thus, we focussed more on using manual testing every time something new was committed to the code.

Manual testing involved us creating use-case scenarios to recreate the potential issues a user may run into while using the website. The following criterias were tested and passed:

Page	**Passing Test Criteria**
All pages	- Ensure all navigation links are up-to-date and not broken. - Each page displays the expected output (no blank pages).
Registration	- Ensure an account is not created if the email entered is already associated with a database. If it exists, notify the user. - Ensure the user cannot proceed to the next form section until all the required fields of the current section are fulfilled - Ensure any fields that require a specific format (i.e. email or password) meet the specified criteria. If it doesn't, notify the user which fields need to be fixed. - Ensure a qualifying account is successfully created and saved in the database.
Login	- Ensure the email and password combination entered match the credentials stored in the database. If not, notify the user.
Reset Password	- Ensure a password reset link is sent to the email address only if the email entered exists in the database. - Ensure the link expires after 5 minutes so that security isn't compromised.
Public Sharing	- Same requirements as 'Sharing' (see below) - Ensure the user must provide an email address as contact information in order to create a post.
Tracker	- Ensure the user-entered data is not saved into the database if not all required fields are filled in. - Ensure the auto-calculated percentages are correct. - Ensure that the data is saved in the database correctly when the form is submitted. - Ensure all the data entered is displayed in the user's data table. - Ensure the data entered is reflected accurately in the graphs. - Ensure the exported CSV file is not empty.
Sharing	- Ensure that all posts are displayed correctly. - Ensure the search functionality searches for the correct posts with the provided keyword. - Ensure all the filters work. - Ensure a post is not created if not all the required fields are filled in. - Ensure posts with 'public' enabled display in the public sharing

	page. - Ensure the poster's contact information and the post created data is displayed in the post. - Ensure the aggregated data graph is displayed correctly and accurately.
Profile	- Ensure the correct account information is displayed back to the user. - Ensure the user is able to edit their profile. - Ensure changes made by the user (update name, password, roles) are reflected in the database. - Ensure the user's account is removed from the database if the user changes their consent to unconsented.
Permissions	- Ensure changes to the data access permissions are updated in the database and applied. - Ensure that new accounts are marked as 'pending' and are displayed in the accounts table. - Ensure changing an account's status to 'approved' or 'declined' is reflected in the database and displayed in the respective table.
Other	- All APIs are tested extensively to ensure they do their assigned tasks. - Tokens were tested by setting various expiration times and seeing if a token was expired when the expiration time was reached.

Missing Requirements

Incomplete

- The final program was not uploaded to the virtual machine.
- The website is not responsive to different screen sizes. Currently, it only works well on large screen devices like laptops and desktops, but not mobile devices.
- When the registered user changes their consent to unconsented, their account is deleted, but not the data (tracker data and sharing posts) that is associated with the account.

Partially Complete

- Token based authentication system is not the most secure as the token is stored in the local storage.

- The aggregated data on the sharing page can either aggregate the data of the logged-in user and a user that they select, or the aggregated data can be of the entire organization. But the logged-in user cannot select more than one user to aggregate their data with.

Known Bugs

There is an issue where on the profile page, the roles list should have boxes checked on the roles that apply to the logged-in user and the consent radio buttons should have the consented button selected. However, these checked/selected fields only display on certain browsers on certain machines. For example, this feature works on a Windows laptop with a Chrome browser, but not with a Macbook. On a Macbook, it works well with a Firefox browser.

There is another issue where on the registration page, if the user fills in the registration form, but declines consent, the system still creates an account. This should not happen as successfully creating an account enters their account data into the database and saves all subsequently entered data as well. Since they are required to agree to the consent in order to create an account, this is a bug.

Areas of Improvement

Due to the limited amount of time to implement all the required features, there are refinements that need to be made to further improve the user interface and experience on the website.

Page	**Refinements**
Tracker	- Ensure the data entered does not lead to negative data values. If a negative data value exists, notify the user to fix it and prevent the data from being saved until it is fixed. - Display an error message if not all required fields are entered. - Improve the display of the graphs as the graph labels can overlap when data values are small.
Sharing	- There are display issues when no posts exist in the database. - Allow the poster to edit or delete their posts.
Profile	- Implement a feature to download their data from the profile page rather than having to navigate back to the tracker page before confirming their change in consent.
Permissions	- Display back to the user the currently saved permissions. - Simplify the page by perhaps separating the data access permissions and account approval features into two pages, or implement page tabs so that the number of pages stays the

	same and doesn't require page loading/reloading.
Register	- Create the 'More Info' page for when the registering user is still unsure of whether they want to consent to data sharing yet. This was an option requested by the client, but no further information was received. Currently, it is an empty link. - Improve formatting of the 'Confirm Details' page
FAQ	- The FAQ page needs to be updated to include instructions on how to use all, or the most important and/or complex, features on the website. Currently, it only includes instructions on how to create and log into an account and how to use the tracker page.

Software Instruction Manual

This user manual provides detailed information about our website, including installation instructions, customizing email settings, missing and partially implemented features, and known bugs. Additionally, links to our GitHub repository and promo video are provided.

1. Installation

1.1 Cloning the GitHub Repository

To clone the GitHub repository, open your terminal or command prompt and run the following command:

```
git clone https://github.com/COSC499-CAPSTONE-GROUP/Multiple-days-Non-Profits-and-Health-Wellness-Group-B.git
```

1.2 Installing Dependencies

Install React and Node dependencies for this project by using the npm install command.

Install the backend dependencies using pip by running the following commands:

```
pip install djangorestframework-jwt
```

```
pip install django-cors-headers
```

1.3 Database Setup

Open the foodsaviour.sql file in your MySQL Workbench.

Execute the SQL code to create the database and admin user with the provided password in settings.py file.

2. Customizing Email Settings

To customize the email used for sending password reset emails, modify the settings in the settings.py file. You will need to generate an app password for the new Gmail account you want to use.

3. Missing and Partially Implemented Features

3.1 Missing Requirements

The final program was not uploaded to the virtual machine.

The website is not responsive to different screen sizes. It only works well on large screen devices like laptops and desktops but not mobile devices.

When a registered user changes their consent to unconsented, their account is deleted, but not the data (tracker data and sharing posts) associated with the account.

3.2 Partially Complete

The token-based authentication system is not the most secure, as the token is stored in local storage.

The aggregated data on the sharing page can either aggregate the data of the logged-in user and a user they select or the entire organization's aggregated data. However, the logged-in user cannot select more than one user to aggregate their data with.

4. Known Bugs

There is an issue where, on the profile page, the roles list should have boxes checked on the roles that apply to the logged-in user, and the consent radio buttons should have the consented button selected. However, these checked/selected fields only display on certain browsers on certain machines.

There is another issue where, on the registration page, if the user fills in the registration form but declines consent, the system still creates an account. This should not happen, as successfully creating an account enters their account data into the database and saves all subsequently entered data as well. Since they are required to agree to the consent to create an account, this is a bug.

5. Remaining Features

Besides the known bugs in the system that need to be fixed, these remaining features need to be implemented to ensure the website runs well at a minimum.

Issue	**Description**
Clarify Permissions	There needs to be more clarification on how the permissions page is supposed to work as it was unclear whether all users

	can access permissions, or certain roles, or if only the administrator has access and can set these permissions. This is something that must be clearly defined before the software is deployed as this is a serious security issue.
Delete Data Upon Account Deletion	When a user chooses to withdraw their consent, their account details are deleted from the database. However, the associated data is not. This must be implemented as keeping the data saved in the database would be a breach of the user's rights since they are withdrawing their consent to share their data.
Negative Values	On the tracker page, we must ensure the data entered does not lead to negative data values. If a negative data value exists, notify the user to fix it and prevent the data from being saved until it is fixed.
Error Messages	Notify the user if an error has occurred (data was not saved, missing required input fields, etc.) and how to fix it. While some of the features have this error handling, it was unfortunately not applied to all features (i.e. tracker form).

6. Additional Resources

6.1 GitHub Repository

Our project's GitHub repository can be accessed using the following link: https://github.com/COSC499-CAPSTONE-GROUP/Multiple-days-Non-Profits-and-Health-Wellness-Group-B.git

This repository is private so it can only be accessed by the client or the TA.

6.2 Promo Video

To watch our software's promo video, click on the following link: https://www.youtube.com/watch?v=A7fbcjE7JUs