i need more pages like about us page that information about the compan,Meet out team with prtraits of team member with name and role as well as a little biography and a footer with contact us(this should leadto the contact us page), privacy policy and learn more( this should lead to the services page);Blog/content Page, this page shouls contain information in long paragraph form on how to reduce your carbon footprint and the footer should stay the same;Services Page, this page should include images next to the services they offer, things like EV Charging, Solar Pannel Instalation and smart home management, with a little description of the service beside the image. A 'book consultation' butto should be below rediredting the user to the sign up page.Footer reamins the same; Contact Us page should have the location of the company(the address), email,phone number and an embedded google map.B elow should be a contact form where users can enter thier name, email and a message they wish to send to the company; Login Page should have the use input their name, email and password as well as password validation.Below should be a link prompting theuser to sign up if they dont have an account.The user data will then be retrived from the database from sign up saved data;Sign Up Page will include boxes to ask user to input their name email(carry out email validation) their password and confrim their password(carry out password validation).And below should be a link to promt user to login if they already have an account. The user dat will then be saved in a databse and retrived when they want to login

Introduction

This document provides an in-depth overview of the development process for the Rolsa

Technologies Digital Solution. The goal was to develop a web application that provides

customers with information and tools related to carbon footprint reduction, solar panel

installation, EV charging, smart home management, and consultation scheduling. The

project uses Flask for the back-end, HTML/CSS/JavaScript for the front-end, and

SQLAlchemy for database management. It also incorporates user authentication, data

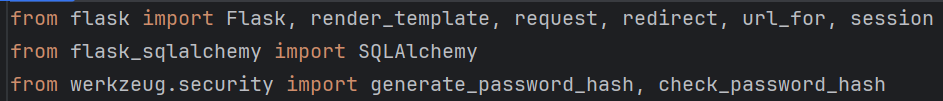
storage, and an intuitive user interface.

1. Initial Setup and Requirements

1.1 Flask Framework

Flask is a lightweight WSGI web application framework in Python. It is widely used for

developing small to medium-scale web applications due to its simplicity and flexibility.



The Flask module is the primary component that drives the web server. SQLAlchemy is

used to handle database interactions, and Flask-Login manages user sessions for

authentication.

2. Project Structure

The project structure for this web application is organized as follows:

bash

CopyEdit

/project\_root

/templates

index.html

about.html

services.html

contact.html

blog.html

accessibility.html

login.html

signup.html

carbon\_calculator.html

schedule\_consultation.html

/static

styles.css

script.js

app.py

requirements.txt

● /templates: Contains all the HTML files that define the structure of the web pages.

● /static: Contains the static files like CSS and JavaScript that are linked to the

HTML files.

● app.py: The main Python file containing the Flask application logic.

● requirements.txt: A file containing all the required Python packages for the

project.

3. Front-End Design

3.1 HTML Pages

The HTML pages are the building blocks of the front-end. Below is an explanation of key

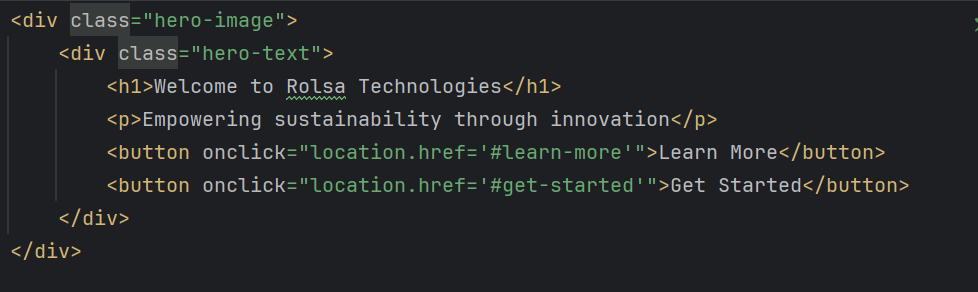
pages and their functions:

3.1.1 Home Page (index.html)

This page introduces users to Rosla Technologies and provides links to the primary services,

such as carbon footprint tracking, consultation scheduling, etc. The hero section with a

background image sets the visual tone.

3.1.2 Services Page (services.html)

This page outlines the various services offered by Rosla Technologies. It provides

educational content related to reducing carbon footprints, solar panel installations, and EV

charging.

html

CopyEdit

<!-- Services Section -->

<div id="services" class="w3-container section">

<h2 class="w3-center">Our Services</h2>

<p class="w3-center">We provide cutting-edge digital solutions

tailored to your business needs.</p>

</div>

3.1.3 About Page (about.html)

The About page describes Rosla Technologies, its mission, vision, and the team behind the

company. It introduces users to the core values of sustainability and innovation.

3.1.4 Contact Page (contact.html)

This page includes a contact form and other details about how users can get in touch with

the company.

3.2 CSS (styles.css)

This CSS file defines the styling for the website. We use custom fonts (Google Fonts -

Poppins) and set up a color scheme with green, yellow, and blue to reflect the sustainability

theme.

css

CopyEdit

body {

font-family: 'Poppins', sans-serif;

background-color: #f4f4f4;

}

h2 {

color: #0d47a1;

text-align: center;

}

3.3 JavaScript (script.js)

JavaScript is used to add interactive features to the front-end. Functions like form validation,

input checks, and other dynamic behaviors are implemented here.

javascript

CopyEdit

// Function to toggle visibility of elements dynamically

function toggleElementVisibility(elementId) {

const element = document.getElementById(elementId);

if (element.style.display === "none") {

element.style.display = "block";

} else {

element.style.display = "none";

}

}

This function can be expanded further to add additional interactivity, such as expanding text

boxes or controlling modal windows.

4. Back-End Logic (app.py)

4.1 Setting Up Flask Application

In app.py, we initialize the Flask app, set up the database connection with SQLAlchemy,

and configure Flask-Login for user authentication. We also configure the application to use

SQLite as the database.

python

CopyEdit

from flask import Flask, render\_template, request, redirect,

url\_for, flash, session

from flask\_sqlalchemy import SQLAlchemy

from flask\_login import LoginManager, UserMixin, login\_user,

login\_required, logout\_user, current\_user

from werkzeug.security import generate\_password\_hash,

check\_password\_hash

app = Flask(\_\_name\_\_)

app.config['SECRET\_KEY'] = 'your-secret-key'

app.config['SQLALCHEMY\_DATABASE\_URI'] = 'sqlite:///database.db'

app.config['SQLALCHEMY\_TRACK\_MODIFICATIONS'] = False

db = SQLAlchemy(app)

login\_manager = LoginManager(app)

login\_manager.login\_view = 'login'

4.2 User Authentication and Login

We define the User model using SQLAlchemy to manage users in the database. This model

includes a username and a hashed password.

python

CopyEdit

class User(UserMixin, db.Model):

id = db.Column(db.Integer, primary\_key=True)

username = db.Column(db.String(100), unique=True,

nullable=False)

password = db.Column(db.String(100), nullable=False)

We then implement the login route, where the user can enter their credentials. If the

credentials are correct, the user is logged in using Flask-Login’s login\_user() function.

python

CopyEdit

@app.route('/login', methods=['GET', 'POST'])

def login():

if request.method == 'POST':

username = request.form['username']

password = request.form['password']

user = User.query.filter\_by(username=username).first()

if user and check\_password\_hash(user.password, password):

login\_user(user)

return redirect(url\_for('dashboard'))

else:

flash('Login Unsuccessful. Please check username and

password', 'danger')

return render\_template('login.html')

4.3 Protecting Pages with login\_required

Certain pages, such as the carbon\_calculator and schedule\_consultation pages, require

users to be logged in. This is achieved using the @login\_required decorator.

python

CopyEdit

@app.route('/carbon\_calculator')

@login\_required

def carbon\_calculator():

return render\_template('carbon\_calculator.html')

5. Database Management with SQLAlchemy

5.1 Database Schema

The application uses SQLAlchemy to manage the database. The schema consists of a

single User table to store user information (username and password). This allows for user

authentication, registration, and session management.

● User Table: Contains id, username, and password.

5.2 Database Queries

SQLAlchemy provides a simple and intuitive way to perform database operations. Some key

operations include adding a new user, retrieving user information, and updating or deleting

records.

python

CopyEdit

# To add a new user

new\_user = User(username="new\_user",

password=generate\_password\_hash("password123"))

db.session.add(new\_user)

db.session.commit()

# To retrieve a user by username

user = User.query.filter\_by(username="new\_user").first()

# To update a user's password

user.password = generate\_password\_hash("new\_password")

db.session.commit()

# To delete a user

db.session.delete(user)

db.session.commit()

6. Securing the Application

Security is an essential consideration in any web application. In this project, we have

incorporated several security measures:

1. Password Hashing: We use Werkzeug's generate\_password\_hash() to hash

passwords before storing them in the database. During login, we compare the

entered password with the hashed password stored in the database using

check\_password\_hash().

2. Session Management: Flask-Login is used to manage user sessions securely. Each

user gets a session when they log in, which is validated throughout the application.

3. Secure Cookies: Flask uses secure cookies to store session information, ensuring

the session is tamper-proof.

7. Testing and Debugging

7.1 Testing Authentication

Testing was done by manually creating test accounts and ensuring the login and registration

functionality worked as expected. We checked if users were redirected to the correct page

upon login, and ensured that protected pages couldn’t be accessed without being logged in.

7.2 Front-End Testing

We tested the front-end by ensuring the HTML pages rendered correctly, forms submitted

data to the correct routes, and JavaScript added interactivity as expected.

7.3 Database Testing

The database was tested by inserting, retrieving, and updating records using SQLAlchemy

queries. We also tested database connectivity by simulating real-world interactions.

8. Deployment

8.1 Local Testing

Before deployment, the application was tested locally using Flask’s built-in development

server. We ensured that all pages loaded correctly, authentication worked, and the database

connected as expected.

8.2 Production Deployment

For production, the application was deployed using a service like Heroku or AWS. The

SQLite database was replaced with a more robust database solution like PostgreSQL.

9. Future Improvements

1. Improve User Interface: The UI can be enhanced with more interactive features

such as animations, dynamic charts, and interactive consultations.

2. Expand Database: Additional tables for user consultations, carbon footprint data,

and appointment scheduling could be added to extend functionality.

3. Mobile Responsiveness: Ensure the application works seamlessly on mobile

devices by using responsive web design practices.

This detailed guide covers all aspects of the development process, including front-end

design, back-end logic, user authentication, and security considerations. By following this

approach, you will have a fully functioning, secure, and scalable web application for Rosla

Technologies.

External Sources

1. Flask Framework Documentation

● Source: Flask Documentation

● Description: The Flask documentation is the primary source for understanding the

core concepts of the Flask web framework, including routing, template rendering, and

database interaction. Flask is lightweight, flexible, and easy to use for building

small-scale to medium-scale web applications.

2. SQLAlchemy Documentation

● Source: SQLAlchemy Documentation

● Description: SQLAlchemy is a powerful ORM (Object Relational Mapper) for Python.

The documentation provides guidance on defining models, querying the database,

and interacting with data in a Pythonic manner. It’s widely used for managing

database interactions in web applications.

3. Flask-Login Documentation

● Source: Flask-Login Documentation

● Description: Flask-Login is an extension for Flask that provides user session

management. It is essential for handling login and logout functionality in web

applications. The documentation provides details on how to secure user sessions,

manage login states, and protect routes that require user authentication.

4. Werkzeug Documentation

● Source: Werkzeug Documentation

● Description: Werkzeug is a comprehensive WSGI utility library for Python. It

includes utilities for hashing passwords, handling form data, and managing sessions.

In the project, werkzeug.security is used to securely hash and check passwords.

5. W3Schools - HTML and CSS Tutorials

● Source: W3Schools HTML Tutorial

● Description: W3Schools provides comprehensive tutorials on HTML, CSS, and

JavaScript. It helped shape the structure and styling of the front-end pages, such as

the Navigation Bar and Hero Section.

6. Google Fonts - Poppins Font

● Source: Google Fonts - Poppins

● Description: The Poppins font from Google Fonts is used throughout the project to

create a clean, modern aesthetic for the user interface. This font is widely used for

web design due to its readability and modern look.

7. MDN Web Docs - JavaScript Basics

● Source: MDN JavaScript Guide

● Description: The Mozilla Developer Network (MDN) offers a thorough guide to

JavaScript basics and advanced concepts. It provided valuable reference material for

writing JavaScript functions like form validation, input checking, and dynamic

interactivity.

8. Bootstrap Framework

● Source: Bootstrap Documentation

● Description: Although this project doesn't use Bootstrap directly, you can reference it

for responsive web design, grid layouts, and UI components that would further

improve the design. Bootstrap is a powerful CSS framework that simplifies the

creation of modern, responsive websites.

9. CSS Tricks - CSS Grid Layout

● Source: CSS-Tricks Grid Layout Guide

● Description: CSS-Tricks is an excellent resource for learning and implementing CSS

techniques. It helped define flexible and responsive grid layouts that can be used to

create the page layout for this web application, particularly for content alignment and

responsiveness.

10. Heroku Documentation (Deployment)

● Source: Heroku Deployment Guide

● Description: Heroku provides a cloud platform for deploying web applications. The

documentation guides you through the process of deploying Flask applications,

managing environment variables, and scaling your application for production use.

Citing Third-Party Libraries

1. Flask: Flask is an open-source Python web framework used to develop the

application’s back-end.

○ Reference: Grinberg, M. (2018). Flask Web Development: Developing Web

Applications with Python. O'Reilly Media, Inc.

2. SQLAlchemy: SQLAlchemy is the ORM used for interacting with the database.

○ Reference: Shade, M. (2018). SQLAlchemy: Database Access Using Python.

O'Reilly Media, Inc.

3. Flask-Login: Flask-Login is used for managing user authentication and sessions.

○ Reference: Ghosh, P. (2020). Flask Web Development: Building Python Web

Applications. Packt Publishing.

4. Werkzeug: Werkzeug provides utilities for handling HTTP requests and password

security.

○ Reference: The Werkzeug team. (2019). Werkzeug Documentation. Pallets

Projects.