Project Title: Identree — Tree Species Identification Using Leaf Images

Overview

Identree is an AI-powered web application designed to identify tree species from leaf images using **deep learning and computer vision**. The project leverages **Convolutional Neural Networks (CNNs)** for image-based classification and integrates a **Flask backend**, **Angular frontend**, and **PostgreSQL database** to deliver a seamless, user-friendly interface.
This system contributes to **biodiversity monitoring** and **environmental research** by providing an automated and scalable solution for tree species identification.

System Design and Architecture

The system follows a **modular architecture**, integrating a machine learning model with a dynamic web application.

Architecture Components:

- **Frontend:** Angular framework for responsive, component-based UI.
- **Backend:** Flask (Python) serving REST APIs, hosting the trained CNN model.
- **Database:** PostgreSQL for managing users, quiz data, and user activity.
- **Machine Learning:** TensorFlow/Keras CNN trained on a curated dataset of leaf images, refined through **data augmentation** and **transfer learning (ResNet)**.
- **Deployment:** The model and web app were containerized and hosted on a **cloud platform** for accessibility and scalability.

Workflow:

1. User uploads a leaf image.
2. Image preprocessing (resizing, normalization) occurs on the backend.
3. The trained CNN model predicts the tree species.
4. Flask returns a JSON response containing:
    - Common name
    - Scientific name
    - Confidence percentage
    - External links (Wikipedia, tree nursery, map location)
5. Angular frontend presents the result in an intuitive interface.

Implementation

Machine Learning Layer

- Built a **custom CNN** model from scratch, followed by a **CNN-ResNet hybrid** for transfer learning.
- Techniques applied:
    - **Batch Normalization**, **Dropout Regularization**, and **Early Stopping** to prevent overfitting.
    - **Data Augmentation** for improving model generalization (rotation, flipping, scaling).
    - **Hyperparameter tuning** using grid search and validation accuracy metrics.
- Achieved high accuracy and reliability across species classes, even under varied lighting and background conditions.

Backend (Flask)

- Flask app loads the trained model and exposes REST endpoints for prediction, user activities, and trivia handling.
- Image data handled via io.BytesIO streams and preprocessed using **Pillow (PIL)** and **OpenCV**.
- Flask integrates securely with PostgreSQL via SQLAlchemy ORM.
- Error-handled endpoints manage failed uploads and invalid inputs gracefully.

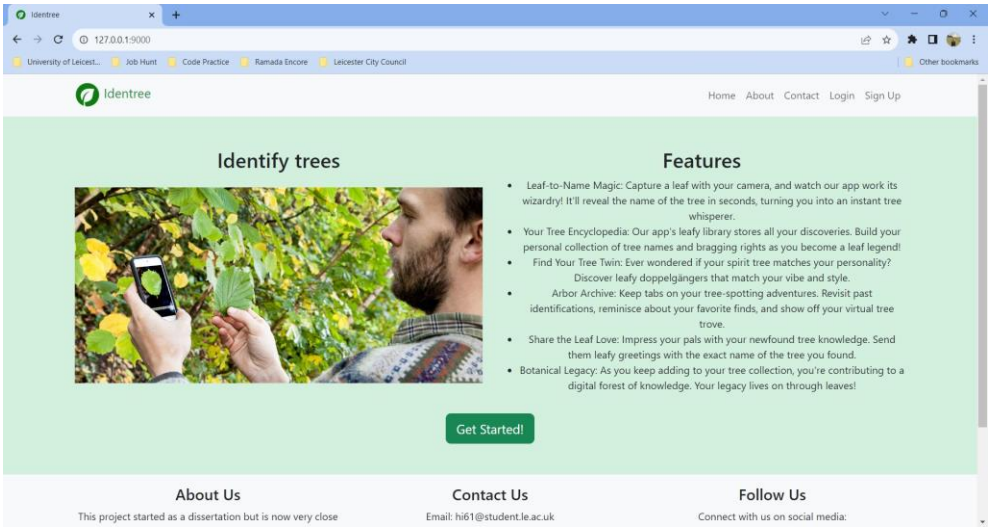
Frontend (Angular)

- Developed using a **mobile-first responsive design** with **Bootstrap** and **Font Awesome**.
- Includes authentication (register/login), image upload, trivia portal, diary view, and mapping functionality.
- Fetches backend predictions and dynamically renders identification results and external resources.

Web Application Components

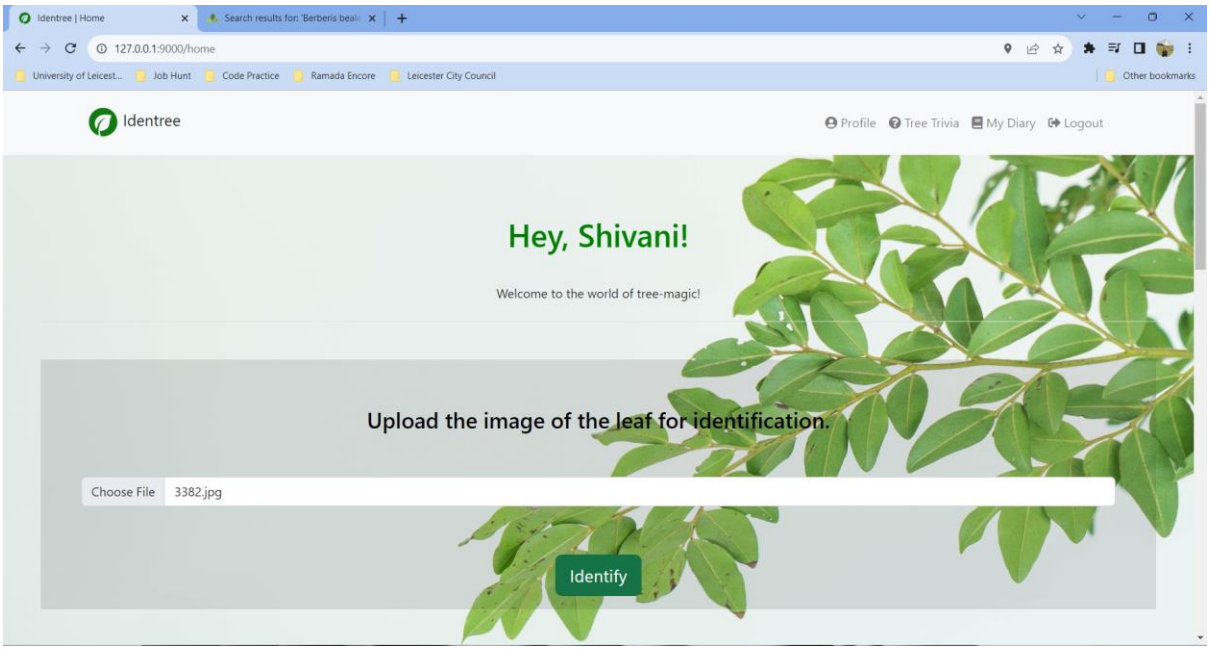
The web app comprises of the following:

- Index Page: Also known as homepage or landing page, serves as a main entry point for Identree web app. Its primary purpose is to provide users an overview of the website's content, navigation options, contact, social media links and other information to begin user interaction on the app.



- Register: It is the registration page that allows user to sign up or create an account with details such as name, email, username, and password. The form has validations to fill all required fields and correct format of details.

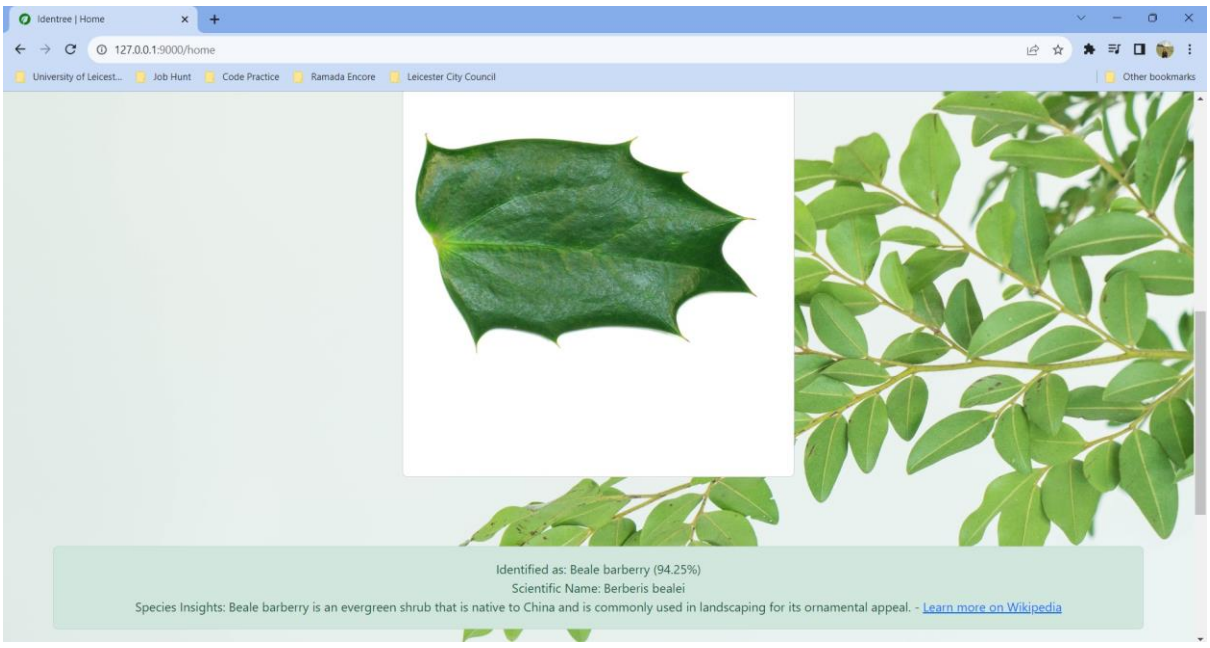
- Login: After registration, a user can login or access to their accounts, validations are added to check if the username/password are valid or invalid and allows access to only valid users.
- Homepage: The main page for tree identification, where a user uploads an image of the leaf and triggers identification.



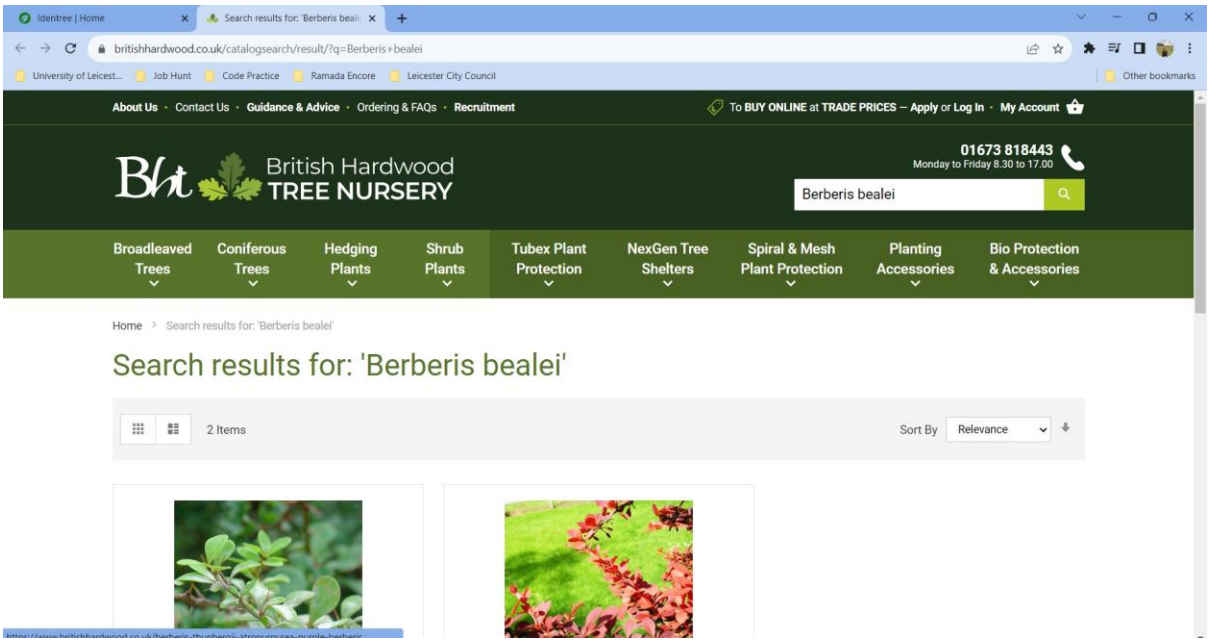
Users interact with the application by uploading an image, which is then pre-processed to prepare it for model input. The loaded machine learning model predicts the species in the image and generates a JSON response containing key information such as the species' common name, scientific name, an interesting fact about the

species, and a percentage indicating the model's confidence in its prediction. This JSON response is validated with a class JSON from the model to ensure correct class labels. Finally, the validated response is presented to the user on the web page, offering a user-friendly display of the identified species and relevant details, enhancing the user experience, and providing accurate information based on the uploaded image.

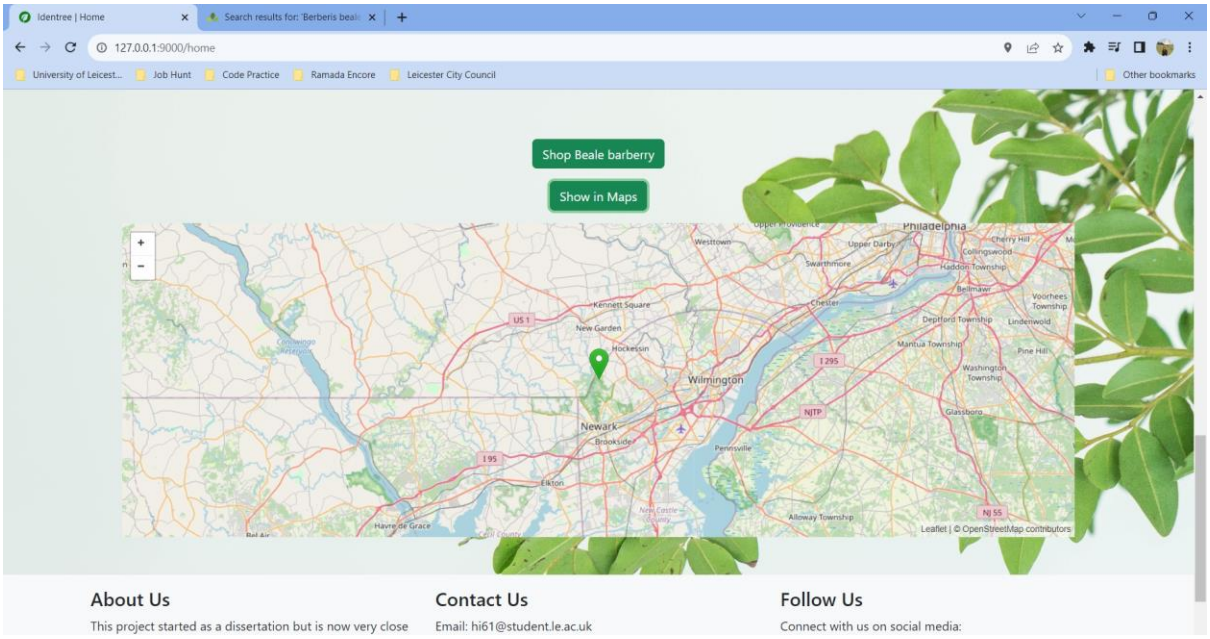
```python
    # Make predictions using the loaded_model
    predictions = loaded_model.predict(image)
    # Determine the predicted class index with the highest confidence score
    predicted_class_index = np.argmax(predictions)
    predicted_class_label = class_info[str(predicted_class_index)]
    predicted_scientific_name = label_details[predicted_class_label]['scientific_name']
    predicted_other_info = label_details[predicted_class_label]['other_info']

    # Calculate the confidence score for the predicted class
    confidence_score = predictions[0][predicted_class_index] * 100
    # Format the confidence score as a percentage with two decimal places
    confidence_percentage = "{:.2f}%".format(confidence_score)

    # Construct the classification response with all the information
    response = {
        'class_label': predicted_class_label,
        'scientific_name': predicted_scientific_name,
        'other_info': predicted_other_info,
        'confidence_percentage': confidence_percentage
    }

    # Log the user's activity in the user_activity table
    if 'loggedin' in session:
        user_id = session['id']

        cursor = db_pool.getconn().cursor(cursor_factory=psycopg2.extras.DictCursor)
        cursor.execute("INSERT INTO user_activity (user_id, image_path, result) VALUES (%s, %s, %s)",
                       (user_id, image_path, predicted_class_label))
        cursor.connection.commit()
        db_pool.putconn(cursor.connection)

    return jsonify(response)
```



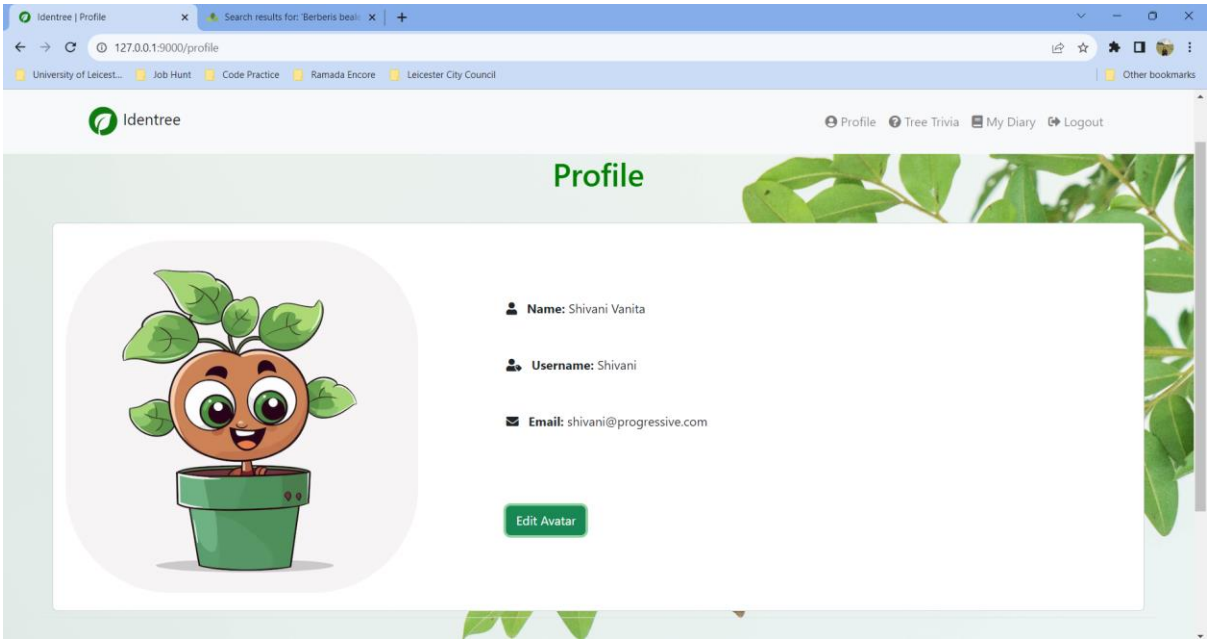
- External links: Wikipedia and Shop at Nursery
  In addition to this the user can also access Wikipedia from the link and has a button to shop the resultant species from a tree nursery



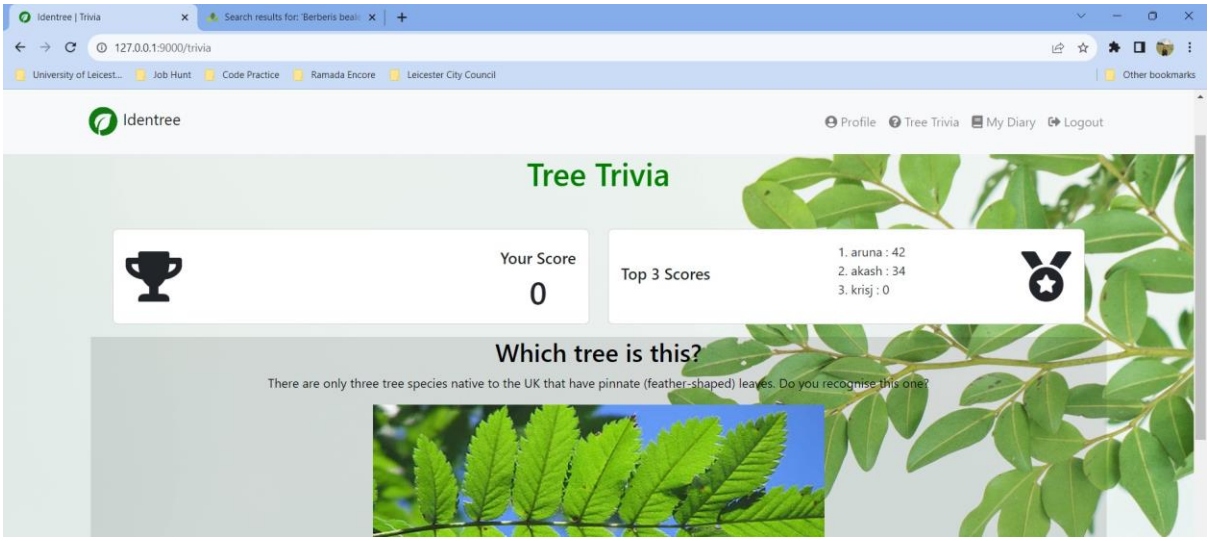
- Map: Location of the tree species
  For the resulting species, the map button also opens up the location of that species and points to the nearest location of user's current location.



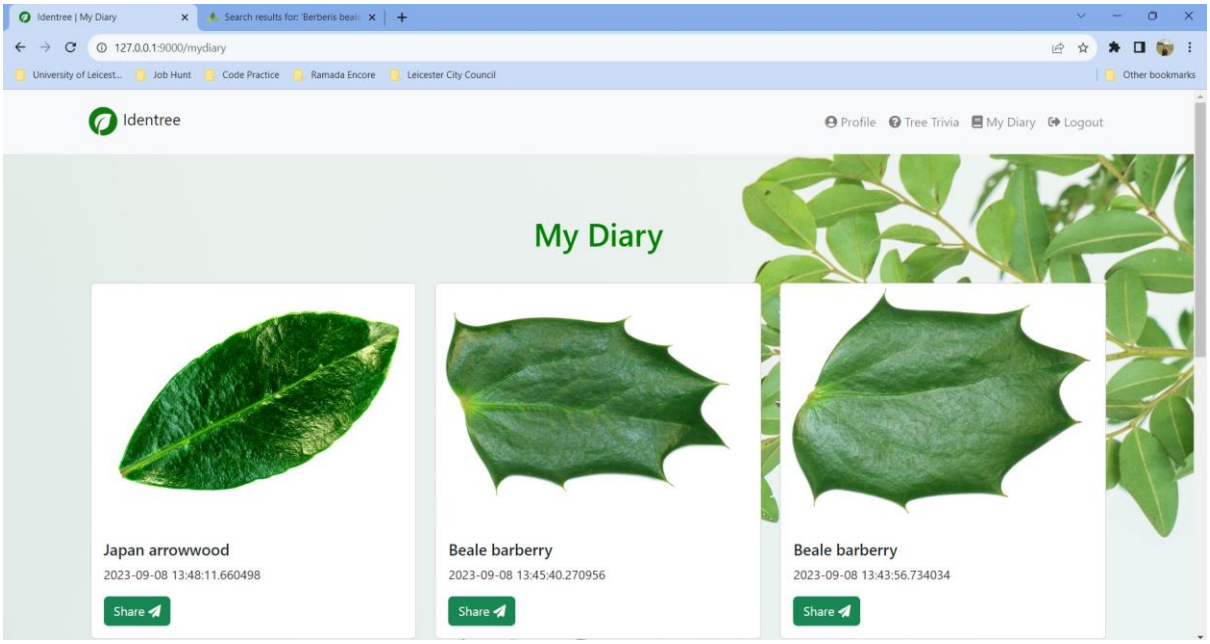
- Profile
  The user can access their profile and check their details.

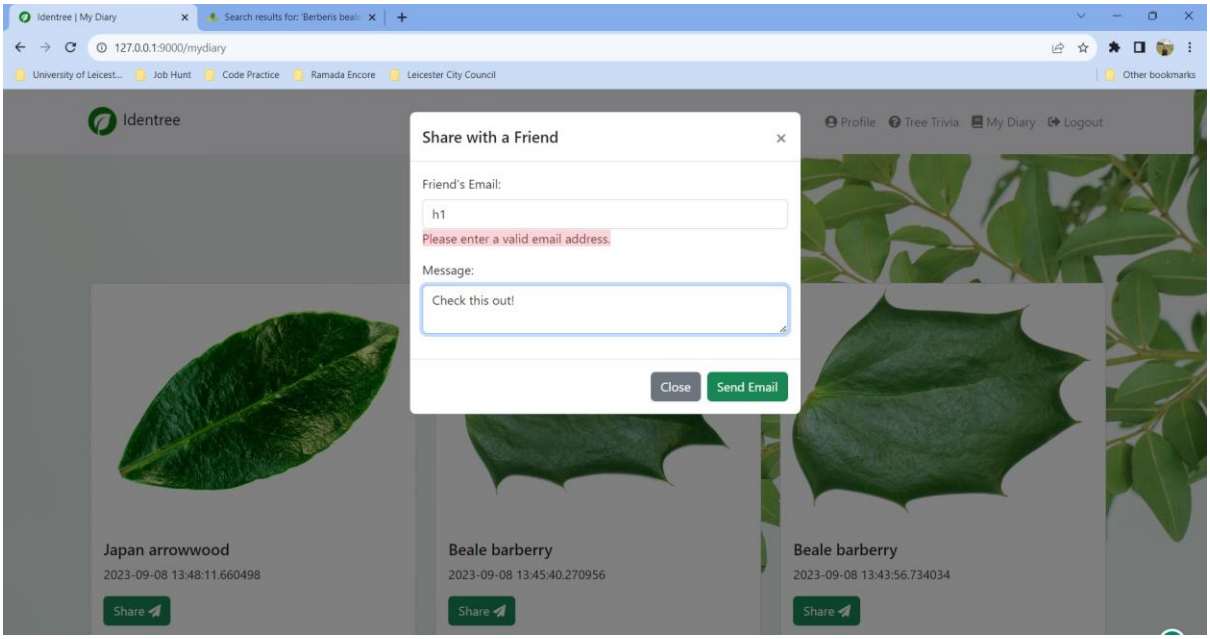


- Tree Trivia
  A user can also enjoy an interactive tree-trivia where a random question pops on the trivia screen and if the user answers correct, the score table updates and the top three scorers are also displayed during the quiz.



- My Diary
  The user can access the images they assessed on the web app and are stored in the diary which can also be shared to friend via email.



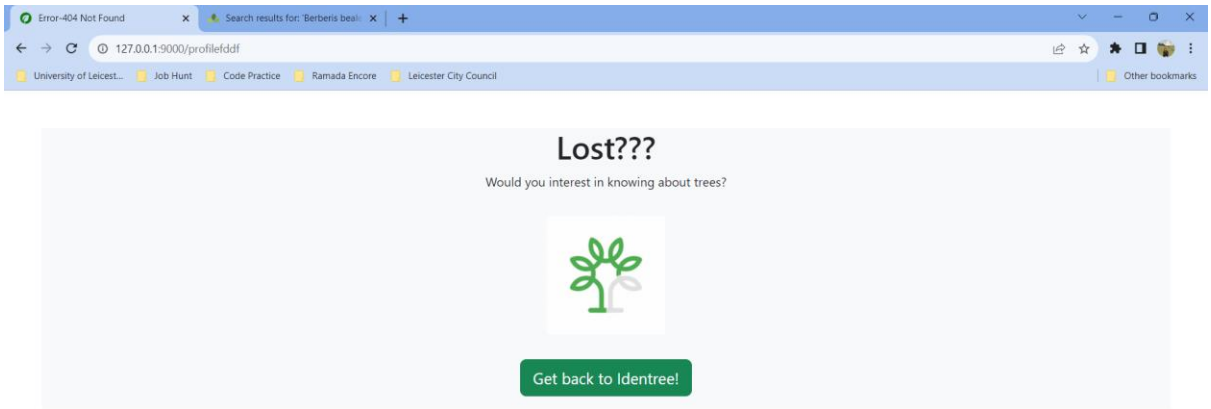


- Logout

The session is closed at the logout of user and the pages other than index, login, and register, no page would be accessible.

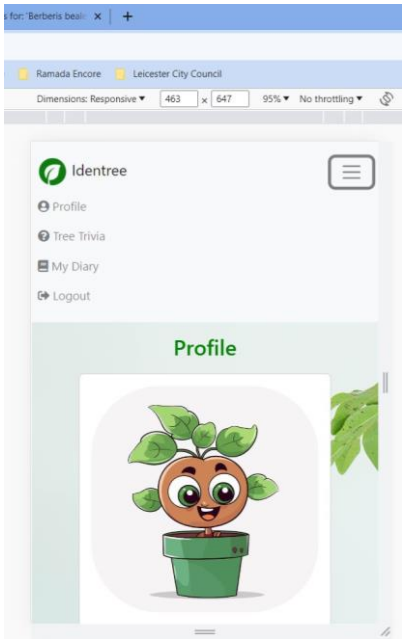
- Error Handling

Error handling in done in various relevant functionalities in the application and even handles 404 error with a custom page.



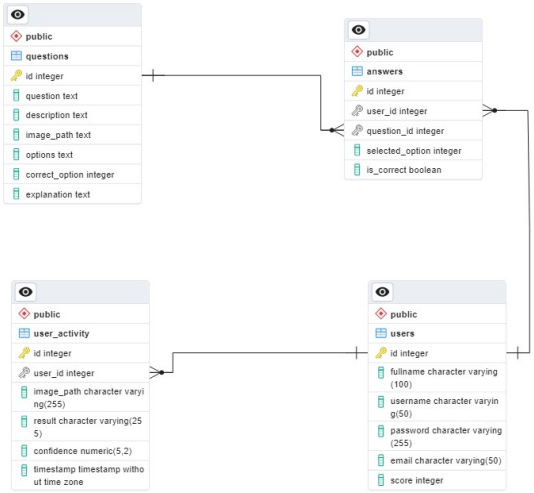
- Responsive

The application has been developed using mobile first approach to be responsive and adaptable on devices of all sizes using libraries such as bootstraps and font-awesome along with adding custom media queries for some elements.



- Database to handle user interactions
  In this database structure, there are several key tables. The "users" table primarily stores user account information, including usernames and potentially additional user details, which users can access via a profile route. The "questions" table holds data related to trivia questions, encompassing the question text, answer options, and correct answers. User scores are influenced by their performance on these questions. The "answers" table is responsible for tracking the answers selected by users on the trivia portal. Lastly, the "user_activity" table appears to be associated with a feature related to tree identification, possibly allowing users to document and access tree-related data in a diary-style format on a dedicated page within the application. This database structure supports various functionalities, including user management, trivia gameplay, score tracking, and user-generated content for tree identification.

  The Entity-Relationship Diagram of the same is as shown :



- **Index Page:** Entry point providing app overview, navigation, and contact links.
- **Register / Login:** Secure user authentication with validation and session management.
- **Homepage:** Upload leaf images for identification and receive model-based predictions.
- **External Links:** Redirects to Wikipedia and a tree nursery for related species.
- **Map:** Displays nearest geographical occurrence of the identified tree species.
- **Profile:** Shows user details and activity.
- **Tree Trivia:** Interactive quiz module with score tracking and leaderboard.
- **My Diary:** Stores user-uploaded images and results; supports sharing via email.
- **Logout:** Ends session; restricts internal page access post-logout.
- **Error Handling:** Custom 404 page and form-level validation.

Testing

Key test cases included:

Testing functionality:

- User without login – access internal pages
  User enter manual link of an internal page – Fail
- User register
  - Invalid email address – Cannot submit
  - Correct email address, but one field empty – Cannot submit
  - Correct email, all fields filled – Submit success
- User login
  - Incorrect username, correct password – Cannot login
  - Correct username, incorrect password – Cannot login
  - Correct username, correct password – Login success
- Image upload for identification
  - Upload image with .webp extension, cannot upload – Error thrown
  - Upload image with .jpeg extension – image uploaded and identified
- Tree Trivia answer submission
  - Click submit button without choosing any option – Cannot submit
  - Submit after choosing an option – Gets correct/incorrect answer
- Email to a friend
  - Enter invalid email – send button disabled, cannot send
  - Valid email – email sent successfully

All test cases executed successfully, ensuring robust and user-friendly interaction.

Software and Tools

Category	**Tools / Libraries**
Deep Learning	TensorFlow, Keras
Image Processing	OpenCV, Pillow (PIL)
Data Handling	NumPy, Pandas, scikit-learn
Visualization	Matplotlib, Seaborn
Web Framework	Flask (Python)
Frontend	Angular, Bootstrap, Font Awesome
Database	PostgreSQL
Version Control	Git / SVN
Cloud Deployment	Azure / Google Colab for model training

Key Contributions

- Designed and trained CNN and CNN-ResNet models for accurate tree species identification.
- Implemented Flask REST API for model integration and data communication.
- Developed a responsive Angular-based web app with authentication, trivia, and diary features.
- Integrated database-driven user management and gamified engagement through quizzes.
- Deployed and tested system across environments ensuring stability and performance.

Conclusion and Future Scope

Identree demonstrates the power of combining **AI**, **web technologies**, and **data analytics** for environmental applications.
The system accurately identifies tree species from leaves, contributing to ecological research and public awareness.

Future Enhancements:

- Integration with **mobile app** for real-time image capture.
- Expansion of dataset to include global species diversity.
- Implementation of **ensemble models** for improved precision.
- Use of **cloud storage** for secure image management.
- Addition of **object detection** for identifying pests or diseases.