Skin Disease Detector

Project Overview

The **Skin Disease Detector** is a **Django-based web application** designed to assist users in detecting skin diseases by uploading images of their skin conditions. The system leverages a **machine learning model** for accurate predictions, powered by the Roboflow API. Users can manage their profiles, view prediction history, and access detailed information about different skin diseases.

Features

**User Authentication:** Users can register, log in, and log out.

**Image Upload:** Users can upload images of their skin to receive predictions for possible skin diseases.

**Machine Learning Integration:** Utilizes the Roboflow API for disease classification.

**Profile Management:** Users can upload and update their profile picture and personal details.

**Prediction History:** Keeps a history of past predictions for easy access.

**Disease Information:** Displays detailed information about each predicted skin disease.

Setup and Installation

P**rerequisites:**

Before setting up the project, ensure you have the following installed:

* **Python** (version 3.8 or higher)
* **pip** (Python package installer)
* **Virtual environment tool** (optional but recommended)

Installation Steps:

1. **Clone the repository:**

**Bash -->**

git clone <repository-url>

cd skin-disease-detector

1. **Create and activate a virtual environment:**

**Bash -->**

python -m venv venv

source venv/bin/activate **On Windows:** venv\Scripts\activate

1. **Install project dependencies:**

**Bash -->**

pip install -r requirements.txt

1. **Apply database migrations:**

**Bash -->**

python manage.py migrate

1. **Create a superuser (optional, for admin access):**

**Bash -->**

python manage.py createsuperuser

1. **Run the development server:**

**Bash -->**

python manage.py runserver

1. **Access the application:**

Open your browser and visit [http://127.0.0.1:8000/](http://127.0.0.1:8000/).

Usage

User Registration and Login:

1. **Register a New User:**

Navigate to the registration page to create a new user account. Fill in the required details, including a username, password, and email.

1. **Login:**

Once registered, log in using your credentials to access the prediction feature.

Upload Image for Prediction:

* After logging in, navigate to the **Upload Image** page.
* Choose an image of your skin condition and submit it.
* The application will use the Roboflow API to predict possible diseases based on the image.

View Prediction Results:

* Once the image is uploaded and processed, the predicted skin disease will be displayed with the confidence score.
* Additional information about the disease, such as causes, symptoms, and treatment options, will also be available.

User Profile:

* You can access your profile page to view and update personal information, such as your profile picture and email address.

Prediction History:

* View your previous uploads and their results on the **History** page.

Disease Information:

* The app provides detailed data about various skin diseases, such as **Chickenpox**, **Measles**, **Monkeypox**, **etc.**

Main Components

Models:

**Profile:** Extends the built-in `User` model to add an optional profile picture.

**SkinImage:** Stores uploaded skin images associated with user predictions.

**SkinPrediction:** Stores the prediction results for each uploaded image, linked to the user.

Views:

**User Authentication:** Handles user registration, login, and logout.

**Image Upload and Prediction:** Allows users to upload skin images and get predictions from the machine learning model.

**Profile Management:** Users can view and edit their profile information.

**Prediction History:** Displays a list of past predictions for each user.

**Disease Information:** Provides detailed disease descriptions based on prediction results.

Forms:

**User Registration and Login Forms:** Handle user input for registration and login.

**Profile Update Form:** Allows users to update their profile picture and details.

**Image Upload and Prediction Form:** Handles image submission for disease detection.

Templates:

Templates located in the `detection/templates/detection/` directory render the user interface for various pages, including:

* Login/Registration Pages
* Profile Page
* Image Upload Page
* Prediction Results Page
* History Page
* Disease Information Pages

Static Files:

Located in `detection/static/`, these files include stylesheets (CSS), JavaScript files, and images used for the frontend UI.

Running Tests

To run the Django test suite and ensure everything is functioning as expected, use the following command:

**Bash -->**

python manage.py test

Project Structure

Here's a breakdown of the project structure:

skin-disease-detector/

│

├── detection/ **# Main Django app for skin disease detection**

│ ├── migrations/ **# Database migration files**

│ ├── static/  **# Static files (CSS, images, JavaScript)**

│ ├── templates/ **# HTML templates for rendering views**

│ ├── templatetags/ **# Custom template tags for advanced functionality**

│ ├── admin.py **# Configuration for the Django admin panel**

│ ├── apps.py **# App configuration settings**

│ ├── disease\_data.py **# Data about various diseases**

│ ├── forms.py **# User input forms (registration, image upload)**

│ ├── models.py **# Database models for user profiles, image uploads, and predictions**

│ ├── signals.py **# Django signals for events like user profile updates**

│ ├── tests.py **# Unit tests to ensure functionality**

│ ├── urls.py **# URL routing for the app**

│ └── views.py **# View functions for handling requests and responses**

│

├── media/  **# Uploaded media files (profile pictures, skin images)**

├── skindetector/ **# Project-level configuration**

│ ├── \_\_init\_\_.py

│ ├── asgi.py

│ ├── settings.py **# Django project settings and configuration**

│ ├── urls.py **# Project-level URL routing**

│ └── wsgi.py

│

├── manage.py **# Django management script for running commands**

├── requirements.txt **# List of project dependencies**

├── README.md **# This documentation file**

└── roboflow\_response.json **# Example response from Roboflow API (for testing)p**

Extending the Project

Here are some ways you can extend the functionality of the project:

**Add New Disease Data:** Add more diseases and their respective information in `detection/disease\_data.py`.

**Enhance Machine Learning Integration:** Improve how the app interacts with the Roboflow API by updating the model in `detection/views.py`.

**UI Customization:** Modify templates and static files to improve the user interface.

**Add More Tests:** Write additional unit tests in `detection/tests.py` to improve test coverage.

Contributing

We welcome contributions to this project! If you have ideas for new features or improvements, please **fork** the repository and submit a **pull request** for review.

**Steps for contributing:**

1. Fork the repository.
2. Create a new branch (`git checkout -b feature-name`).
3. Make your changes.
4. Commit your changes (`git commit -am 'Add new feature'`).
5. Push to the branch (`git push origin feature-name`).
6. Open a pull request on GitHub.

License

This project is licensed under the **MIT License**. See the **LICENSE** file for more details.

Contact

For any questions, feedback, or support, please reach out to the project maintainer:

**Email:** [arslansajjad548@gmail.com]

Additional Notes

**Roboflow API Integration:** The project uses Roboflow to predict skin diseases. Ensure you have an API key if you're extending the model.

**Media Files:** Uploaded images and profile pictures are stored in the `media/` directory. Make sure your server is set up to serve media files.

**User Authentication:** This project uses Django's built-in authentication system for user management.