جامعة أنطاكية السورية الخاصة
Antioch Syrian Private University

PLANTER

Prepared by: Jawdat Al-Husien

supervisied by : prof. Ahmad Khadour

2024-2025

Planter:

Introduction and objective

Project Name: "Planter" – A Technology-Driven Platform for Plant Health and Knowledge Sharing

1. General Introduction

Overview

Planter is an innovative digital ecosystem that combines machine learning with community-driven knowledge sharing to empower farmers, gardeners, and plant enthusiasts. The platform addresses critical agricultural challenges through:

- Instant plant disease diagnosis via AI
- Crowdsourced educational resources
- Gamified learning experiences
- Collaborative knowledge exchange

2. Key Components

2.1 AI-Powered Plant Disease Diagnosis

Functionality: Users upload leaf images for real-time analysis using convolutional neural networks (CNNs).

Technical Details:

- Model: Fine-tuned MobileNetV2 architecture (transfer learning)
- Dataset: 2,000 labeled plant disease images (diverse species/disease types)
- Augmentation: Rotation, zoom, brightness variation, and flipping
- Training: Two-phase approach (frozen base → fine-tuned layers)
- Output: Disease classification with confidence score

Innovation: Democratizes access to diagnostic tools for smallholder farmers through lightweight, mobile-friendly architecture.

2.2 Blogging Platform

Features:

- Article publishing system with expert verification

- Reporting a post or a comment
- Reports are admin monitored for assurance
- Admin ability to remove inappropriate posts

2.3 Trivia & Fun Facts

Interactive education through bite-sized content:

- Historical plant uses (e.g., willow bark → aspirin)
- Record-breaking specimens (e.g., oldest living trees)
- Real-time AI generated facts

2.4 Gamified Learning

- **Quizzes**: 400 random questions across difficulty levels
- **Rewards**: Scoring system
- **Partnerships**: University collaborations for academic validation (future goal)

2.5 Role-Based Access Control

Role	Permissions
Admins	Content moderation, account management + Core features
Analysts	Model auditing, diagnosis verification + Core features
Normal Users	Core features

3. Target Audience

Primary:

- Smallholder farmers
- Urban gardeners
- Agricultural students

Secondary:

- Botanists
- Environmental NGOs (non- governmental organizations)

4. Problem Statement

Traditional Challenges:

- Slow diagnosis via lab tests (weeks for results)
- Limited access in rural areas
- Misinformation proliferation online

Planter's Solutions:

- Instant AI diagnosis (≤10 second inference time)
- Verified crowdsourced knowledge
- Gamified education to combat information gaps

5. Vision

Long-Term Goals (2030):

- Diagnose 50+ crop species
- 1 million active users
- Mobile AR integration for field scanning
- Universities collaboration
- Government partnerships for agricultural extension

6. Development Phases

Phase	Name	Description	Git Tracking
0	Research & Planning	Studied academic papers on CNN-based detection and gamification; mapped user workflows (image upload → diagnosis → recommendations)	N/A
1	Model	Trained CNN model (MobileNet) on	Branch:

	Development	2,000+ images; dataset preprocessing and hyperparameter tuning	`feature/model-training`
2	Model Testing	Validated ≥78% accuracy; optimized inference speed ≤10 seconds	Tag: `v1.0-model`
3	Core Upload Functionality	Built UI for image uploads with unique ID storage	Merged: `feature/image-upload`
4	Authentication System	Implemented login/registration with email verification; role-based access logic	Branch: `feature/auth`
5	Blogging Platform	Developed CRUD for posts/comments/likes/reports;	Commits: `feature/blog-organic-farming`
6	Web Scraper for Trivia	Created Python scripts to scrape Qwen AI ;	Branch: `dev/web-scraper`
7	Quiz Framework	Designed quiz engine with question banks and scoring logic	Branch: `feature/quiz-engine`
8	Notifications System	Implemented real-time alerts via WebSocket for quiz results, blog updates, and diagnosis status	Branch: `feature/notifications`
9	Role-Based Governance	Defined admin/analyst roles for content moderation and AI model auditing	Branch: `feature/role-control`
10	Frontend Development	Built responsive UI/UX with mobile-first design principles; {HTML ,CSS ,JS} implementation	Branch: `frontend`
11	Initial Software Testing	Conducted unit/integration tests for core features (image upload, login, etc.)	Hotfix: `hotfix/testing-phase1`

12	AJAX/JSON Enhancements	Integrated AJAX for dynamic content (live comments and likes ..etc....) ; performance optimization	Branch: `feature/ajax-optimization`
13	Final Testing & Polishing	fixed edge cases	Merged all

ML Model Development Details:

```
Base Model: MobileNetV2 (pretrained weights)
Freezing Strategy: Phase 1 - frozen base, Phase 2 - last 50 layers
trainable
Augmentation: 8 transformation techniques
Regularization: L2 + Dropout (0.5/0.7)
Optimizer: Adam (1e-5 → 1e-6 learning rate decay)
```

7. Project Objectives

1. **Instant Disease Diagnosis** : Use smart technology to help farmers/gardeners spot plant diseases early and get solutions fast.
2. **Community Knowledge Hub** : Connect plant lovers to share tips, ask questions, and learn through blogs, quizzes, and real time generated trivia facts.
3. **Trusted & Engaging Tools** : Verify AI predictions with experts, and use fun quizzes to keep users learning and involved.

Objective	Implementation Details
Rapid Diagnosis	2,000-image dataset + MobileNetV2 (78% val accuracy)
Knowledge Sharing	Moderated blogging platform
Gamification	400+ quiz questions + scoring system

Accessibility	Mobile-first design + all screen sizes friendly

Environmental Impact:

- Crop loss reduction .
- Planets health knowledge sharing via blogging.

8. Technical Innovations

Model Training Pipeline:

1. **Data Preparation**:
    a. 2,000 images across 6+ classes
    b. 80/20 train/validation split
    c. Class weighting for imbalance
2. **Training Process**:

```
Phase 1: Top layers training (20 epochs)
Phase 2: Fine-tuning (20 epochs, LR=1e-6)
Metrics: Accuracy (80%), Precision (85%), Recall (86%)
```

Planter:

System analysis and design

Planter – System Analysis and Design

1. Introduction

Planter is a multi-component Django-based web application designed to provide users with tools to:

- Manage user accounts securely
- Create, read, comment, report, and moderate blog content
- Upload plant leaf images for diagnosis using machine learning

The system consists of three modules:

1. **Accounts WebApp**: Handles authentication, roles, bans, verification
2. **Blogs WebApp**: Enables blogging with reporting and moderation
3. **Leaf Identifier WebApp**: Diagnoses plant diseases from uploaded images

2. System Requirements

Functional Requirements

Accounts

- User registration and login
- Email verification
- Password reset
- Role assignment (admin, analyst, normal user)
- Strike system and banning mechanism

Blogs

- CRUD operations for blog posts

- Commenting/Liking
- Reporting blogs/comments
- Admin deletes inappropriate posts
- Notifications users

Leaf Identifier

- Upload leaf image
- Display diagnosis result with confidence score
- Store results in database
- Analyst validation of diagnoses
- Quizzing system integrated with the scoring system
- Generate trivia about plants

Non-functional Requirements

- Secure authentication and role-based access
- Fast and accurate image diagnosis
- Scalable and maintainable design
- Responsive UI/UX

3. System Functions (Processes)

Module	Function ID	Function Name	Description
Accounts	F01	Register User	Registers new users; sends verification email
	F02	Verify Email	Verifies user's email via code
	F03	Login User	Authenticates user using credentials
	F04	Reset Password	Allows users to reset passwords
	F05	Ban User	Bans user after 3 reports

Blogs	F06	Create Blog	Users can write and publish blogs
	F07	Edit/Delete Blog	Users can edit/delete their own blogs
	F08	Report Post	Users can report blogs/comments
	F09	Delete Post	Admin deletes inappropriate blog posts
	F10	Send Notification	Sends notifications (ban, report status)
Leaf_ identifier	F11	Upload Image	Uploads plant leaf image for diagnosis
	F12	View Diagnosis Result	Displays diagnosis with confidence level
	F13	Save Diagnosis Result	Saves diagnosis in DB for review
	F14	Validate Diagnosis	Analyst validates accuracy of diagnosis
	F15	Quizzing	Quizzing with a scoring system where a user can choose difficulty
	F16	Trivia	Realtime generated trivia via web scraping or local LLM

4. Data Dictionary

Data Item	Description	Format / Type
UserID	Unique identifier for a user	Integer (Auto-increment)
Username	Unique username	String
Email	Email address	String

Password	Hashed password	String (encrypted)
Role	User role (user/admin/analyst)	Enum
StrikeCount	Number of valid reports against user	Int
IsBanned	Whether user is currently banned	Boolean
BanExpiry	Timestamp when ban expires	DateTime (nullable)
BlogID	Unique ID for each blog	Integer
Title	Title of blog	String
Content	Body of the blog	Text
Likes	Number of likes on a post	ManyToManyField
ReportID	Unique ID for each report	Integer
Reason	Reason for report	Text
Status	Status of report (pending/approved)	Enum
LeafID	Unique ID for leaf diagnosis	Integer
ImgURL	Path to uploaded leaf image	String
Disease	Predicted disease name	String
Confidence	Prediction confidence (0–100%)	Float
UploadedAt	Timestamp when image was uploaded	DateTime
Validated	Whether analyst validated diagnosis	Boolean

5. Entity Relationship Diagram (ERD Summary)

Entities:

- User
- UserProfile
- EmailVerification
- BlogPost
- Comment
- Report
- Notification
- Leaf
- Score

Key Relationships:

- One User has one UserProfile
- One User can have many BlogPosts
- One BlogPost can have many Comments
- One Report can target a BlogPost or Comment
- One Leaf represents a diagnosis result
- One UserProfile can be associated with multiple Leaf records

6. Use Case Diagram

Actors:

- Normal User
- Admin
- Analyst

Use Cases:

- Register/Login
- Verify Email
- Reset Password
- Write/Edit/Delete Blog
- Comment on Blog
- Report Post or Comment
- Delete Post (Admin only)
- Ban User (Admin only)
- Upload Leaf Image
- View Diagnosis Result

- Validate Diagnosis (Analyst only)
- Receive Notifications

7. System Design Overview

a) Modular Architecture

Module	Features
Account Management	Registration, login, verification, banning
Blog Management	Blog CRUD, comments, reporting, notifications
Leaf Diagnosis	Image upload, diagnosis display, saving results
Admin Panel	Ban users, delete posts, view reports
Analyst Dashboard	Review and validate leaf diagnoses

b) Technology Stack

- Backend: Python (Django)
- Frontend: HTML/CSS/JS or React.js
- Database: PostgreSQL or SQLite
- Image Processing: TensorFlow/Keras
- Authentication: Django Auth + JWT
- File Storage: AWS S3 or local media storage
- Notifications: In-app notifications using model

8. Sample Screen Layouts

Leaf Diagnosis Result Page

```
Diagnosis Result
Uploaded Image: [image_preview.jpg]
Predicted Disease: Bacterial Spot
Confidence: 95%
Timestamp: 2025-04-05 14:30
```

9. Role-Based Permissions

Normal User

☑Can register, login, reset password

☑Can create, edit, delete their own blog posts

☑Can comment on blogs

☑Can report blogs/comments

☑Can upload leaf images and view diagnosis

❌Cannot ban other users

❌Cannot validate diagnoses

❌Cannot delete other users' content

Admin

☑All normal user permissions

☑Can delete any blog post

☑Can ban users after 3 approved reports

☑Can manage reported content

❌Cannot validate leaf diagnoses

Analyst

☑All normal user permissions

☑Can log in

☑Can view all saved leaf diagnoses

☑Can validate whether diagnosis was accurate

❌Cannot interact with users blogs (only as a normaluser)

✗Cannot ban users

Planter:

Implementation and overview

Planter Implementation: Functionality Overview and Demonstration

Note : In Django, `views.py` serves as the backend hub where code awaits user requests. Each function within this file defines a specific feature, processing interactions like form submissions, data retrieval, or dynamic page rendering.

The project is structured into three distinct web applications , each designed to handle a unique role:

1. Accounts
2. Blogs
3. Leaf Identifier

Below is a detailed breakdown of each app's functionality:

1. Accounts App

Purpose : Manages user authentication and account management.
Key Features :

- User Registration : Allows new users to create accounts with validated input fields (e.g., email, password).
- Login/Logout : Secure session management for authenticated users.

2. Blogs App

Purpose : Facilitates content creation and community interaction through blog posts.
Key Features :

- Post Creation : Users can draft and publish blog entries with rich text formatting and images.
- Edit/Delete : Owners may modify or remove their posts.
- Blog Feed : Displays a dynamic list of public blogs from all users, organized by date or category.
- Commenting System : Allows readers to engage via comments (optional enhancement).

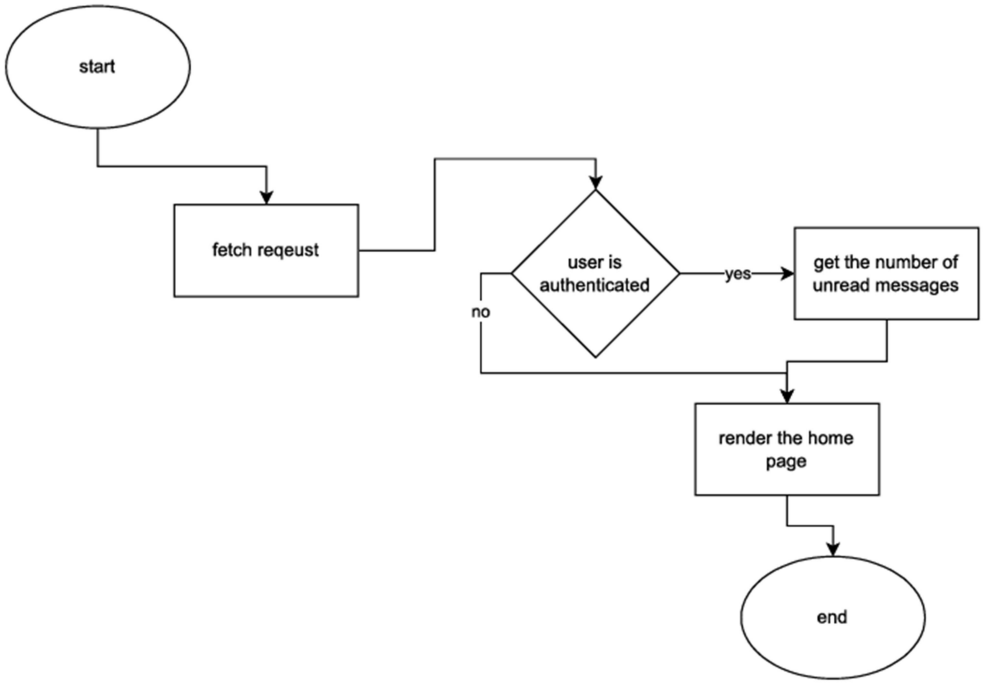
3. Leaf Identifier App

Purpose : Combines educational tools and machine learning for plant-related diagnostics.
Key Features :

- Interactive Quizzes : Tests users' knowledge on plant species, care, or environmental impact.
- Trivia Facts : Displays bite-sized educational content about flora.
- Image Upload & Diagnosis :
    - Users upload plant images for analysis.
    - Integrates a machine learning model to identify plant species or detect diseases.
    - Returns results with confidence and diagnosis.

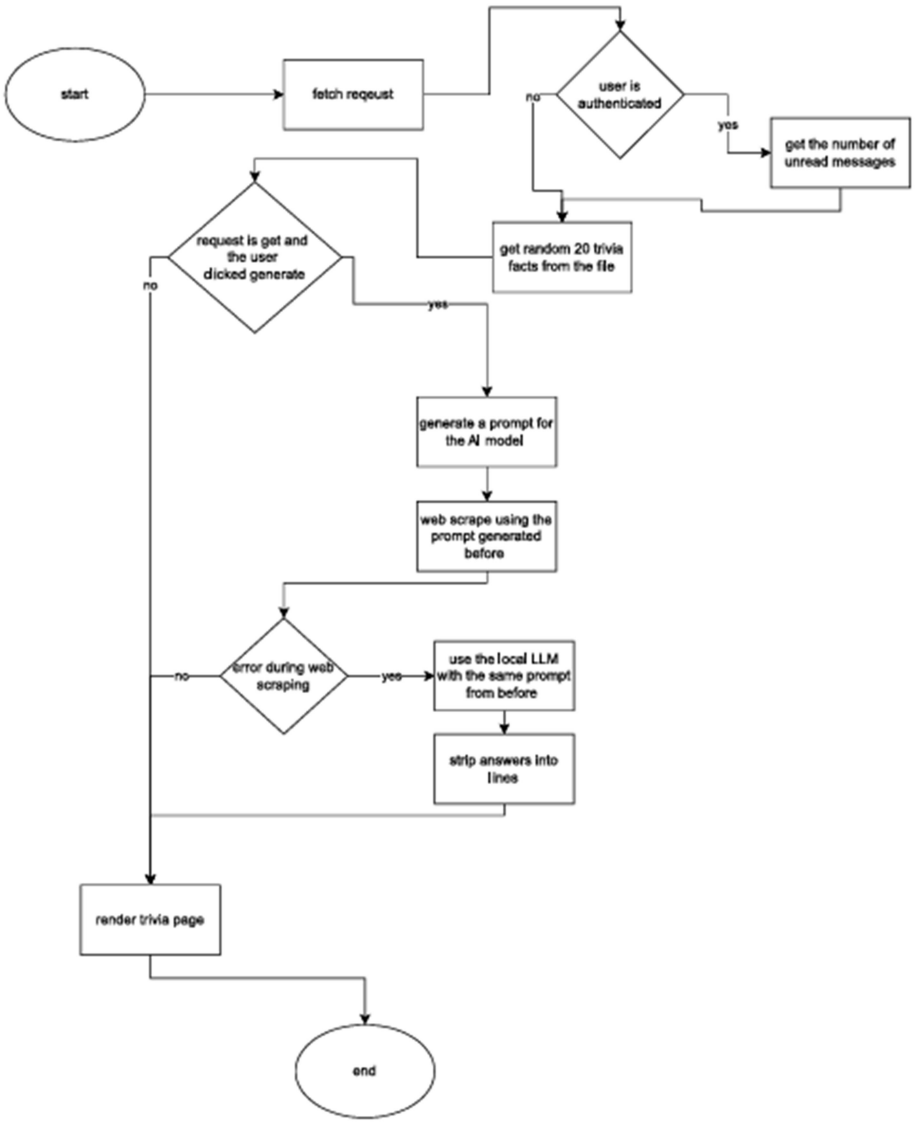
Starting with Leaf Identifier app

Note: each page gets the notifications so we wont be mentioning that in the demonstration of the function and what it does which same goes for the login decorator.



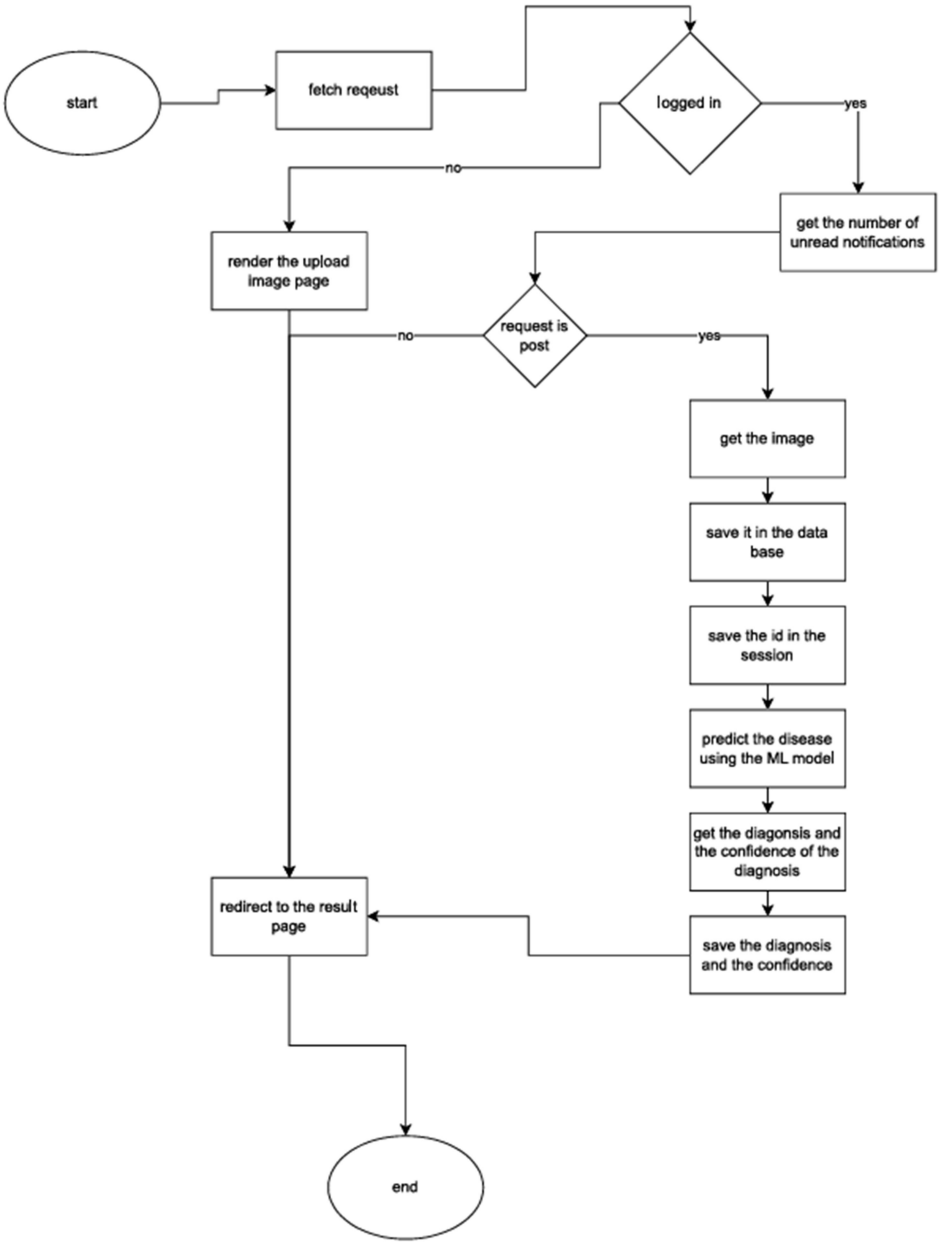
Home view :this is where the user is redirected upon logging in and is considered the base view of the webapp

It renders the home page whenever the user visits the website and if they are logged in it shows the notifications.

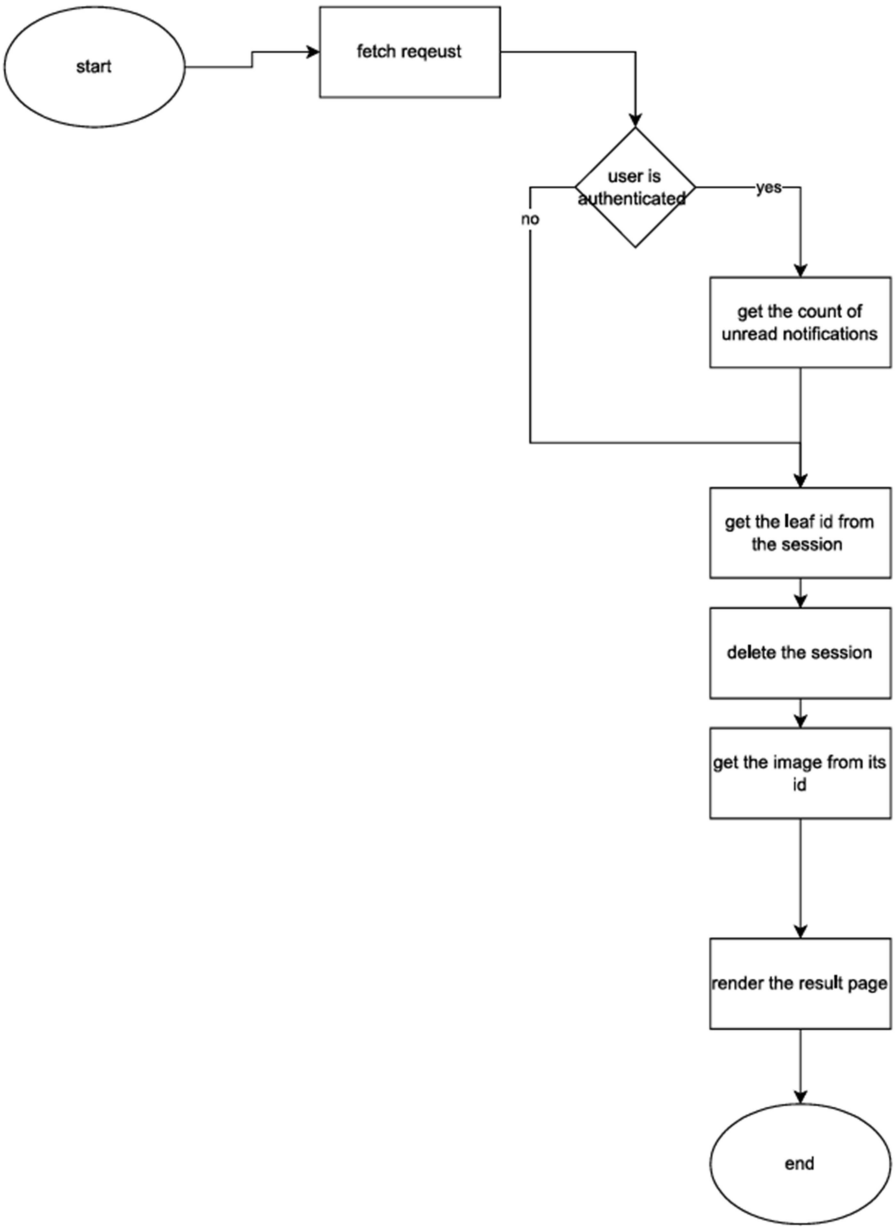


Trivia view: this is responsible of generating real-time trivia facts about plants using Qwen scraper or Zypher AI in case Qwen failed ,it also generates random trivia which are stored in the code before to appear before the AI generated facts be ready for the user .

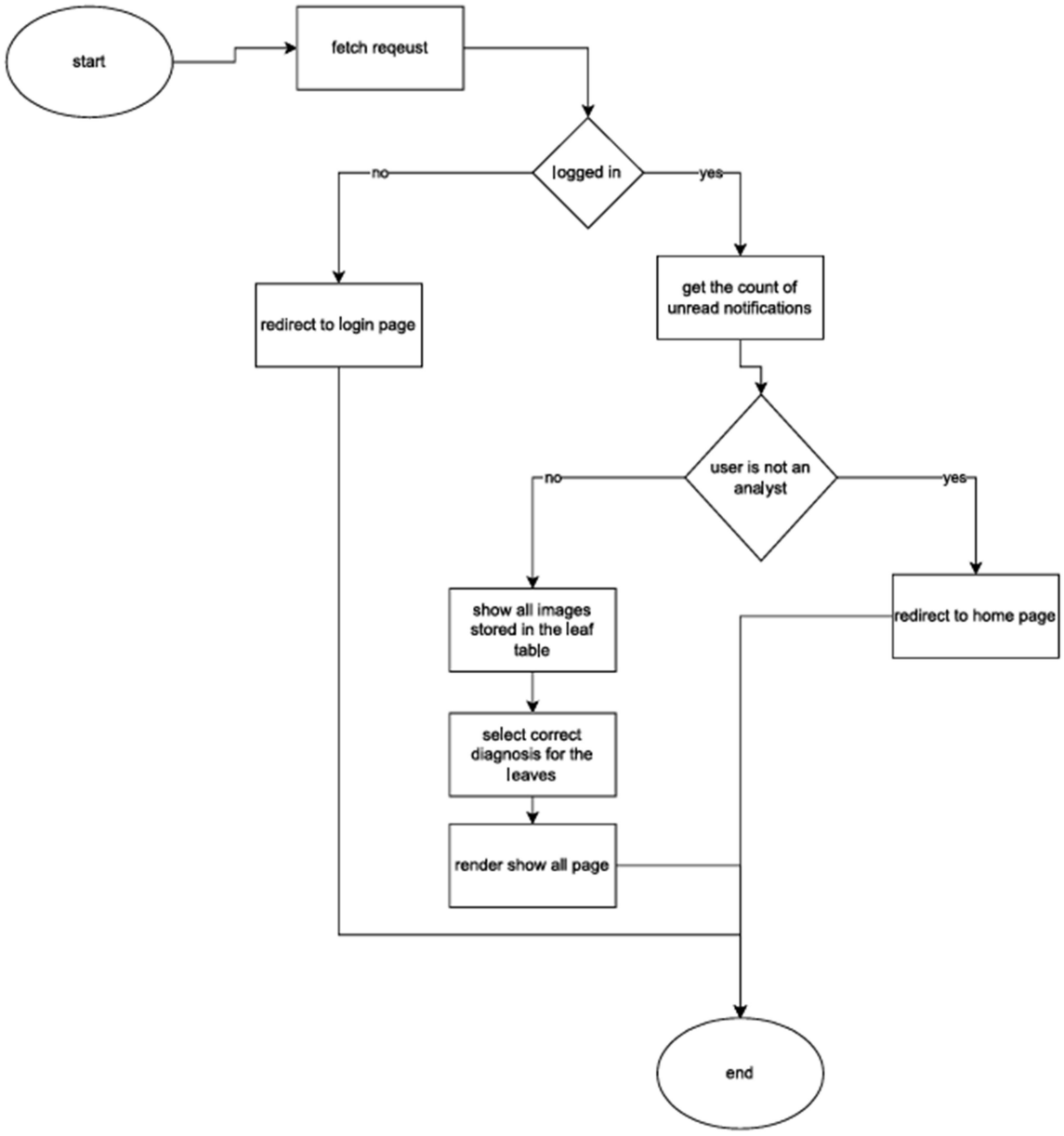
Note: to get rid of deterministic answers it uses a prompt generator to create random prompts for before feeding it to the AI model.



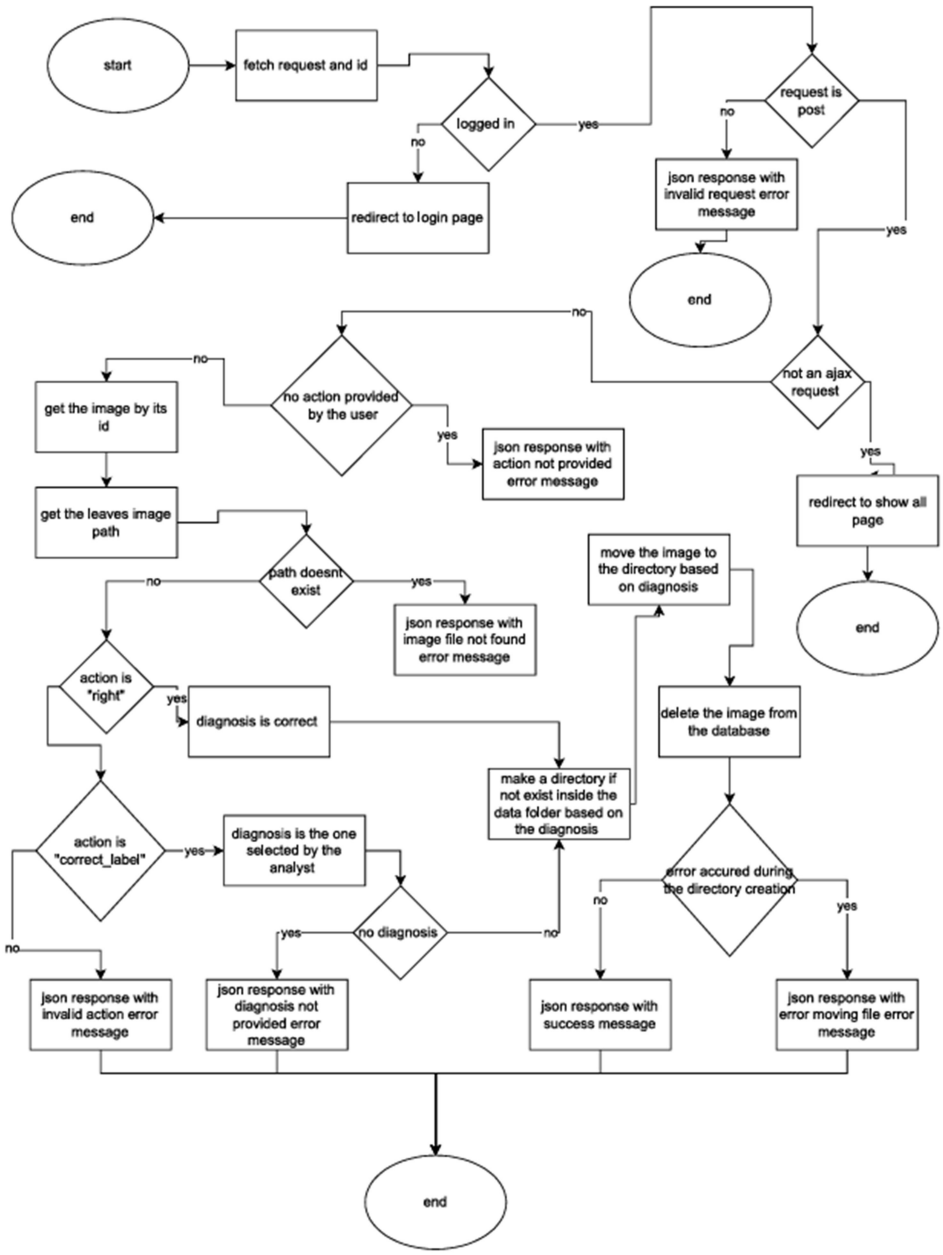
Upload view : is where the user can use the ML model to see the leaf Diagnose , this function stores the ID of the image in the session so it can be used in the result page in addition to storing the image and the results of the prediciton in the DB , after predicting it redirects the user to the result page where they can see the diagnosis and the confidence .



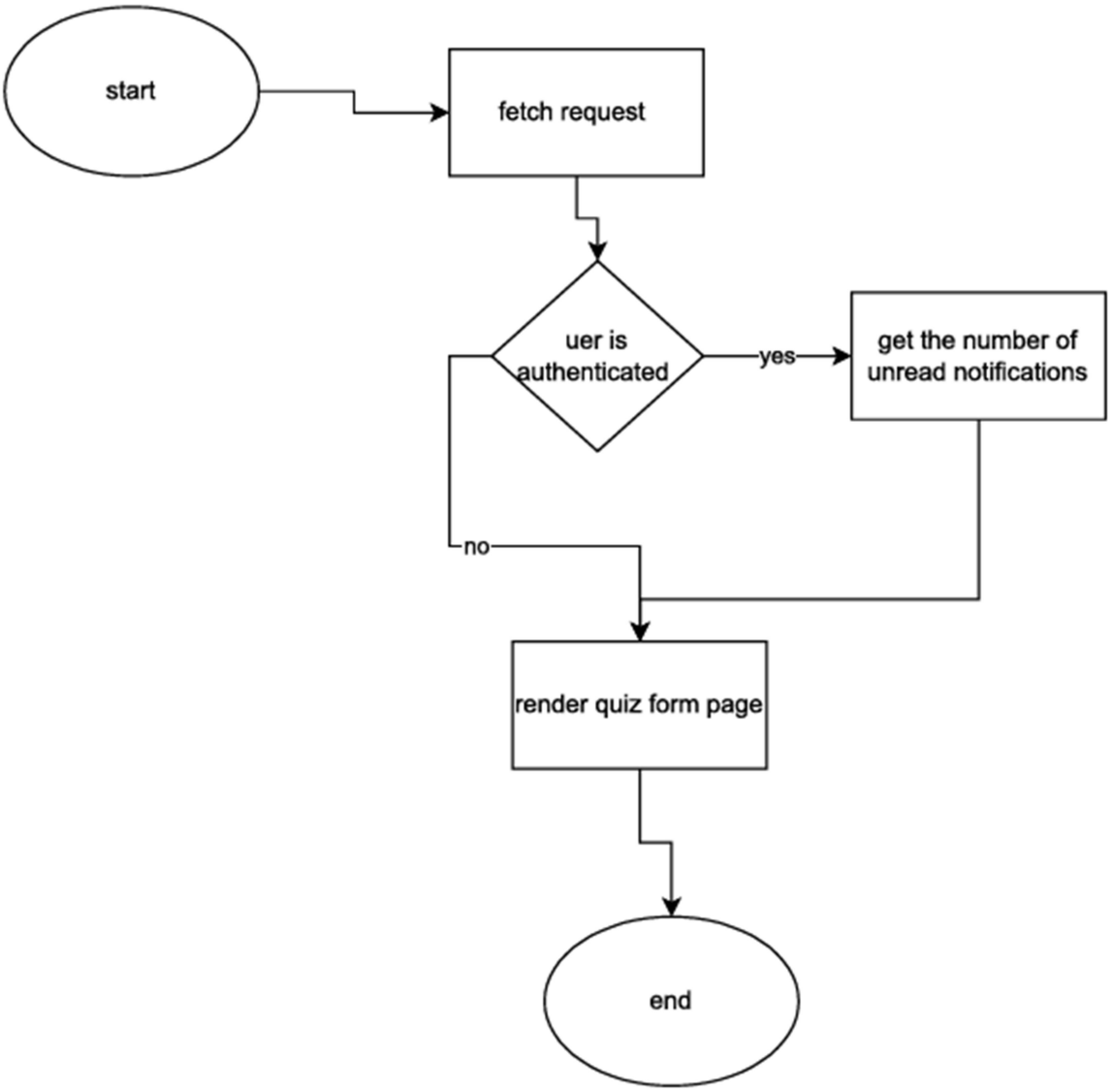
Result view: it uses the session id to get the leaf after uploading it where the session is deleted afterward and the diagnosis and confidence are restored from the Database.



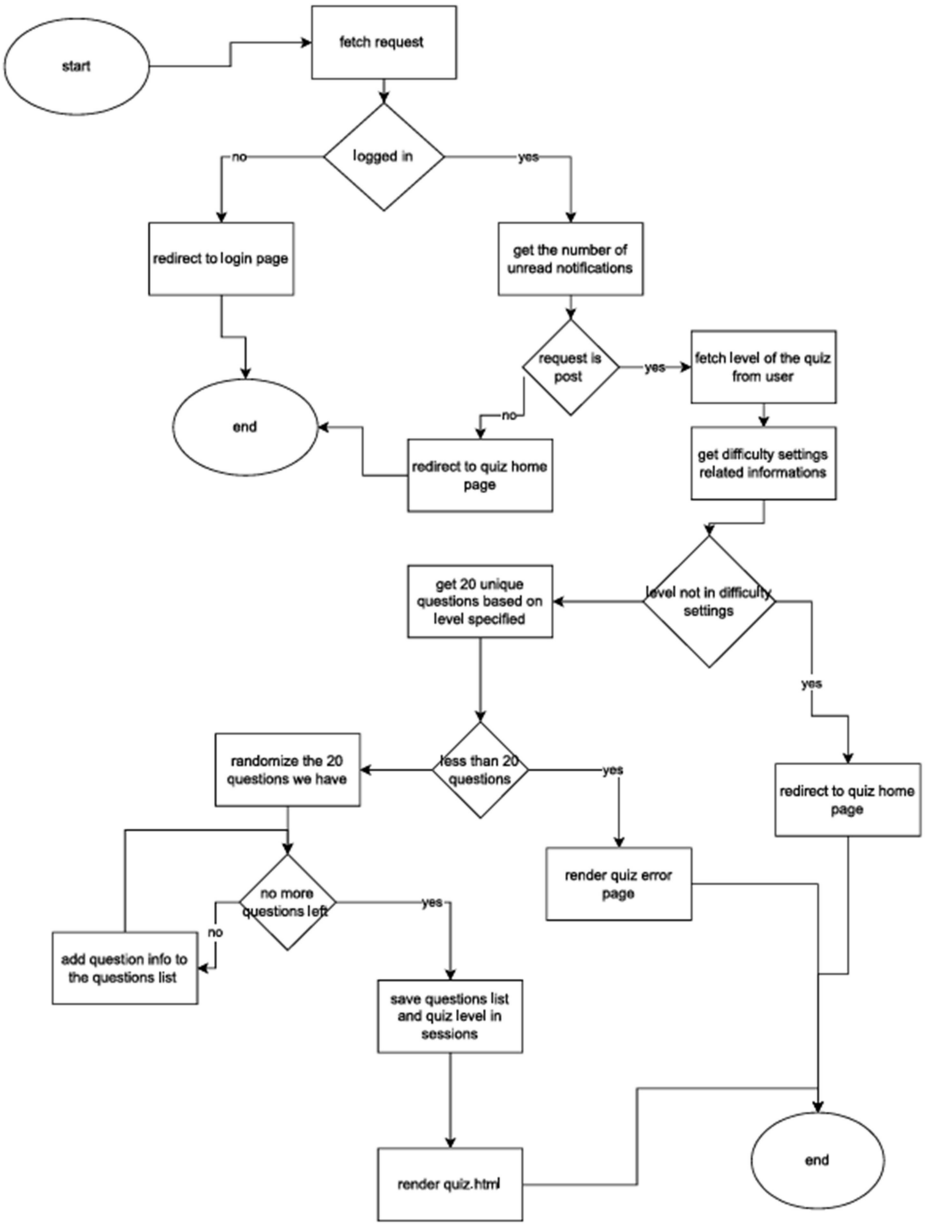
Show_all view: this view redirects any request if its not from an analyst and is where the analyst can do their job , the uploaded leaf images and their diagnosis are viewed by the analyst and marked as correctly labeled or if the label needs to corrected before moving the images to the training folder.



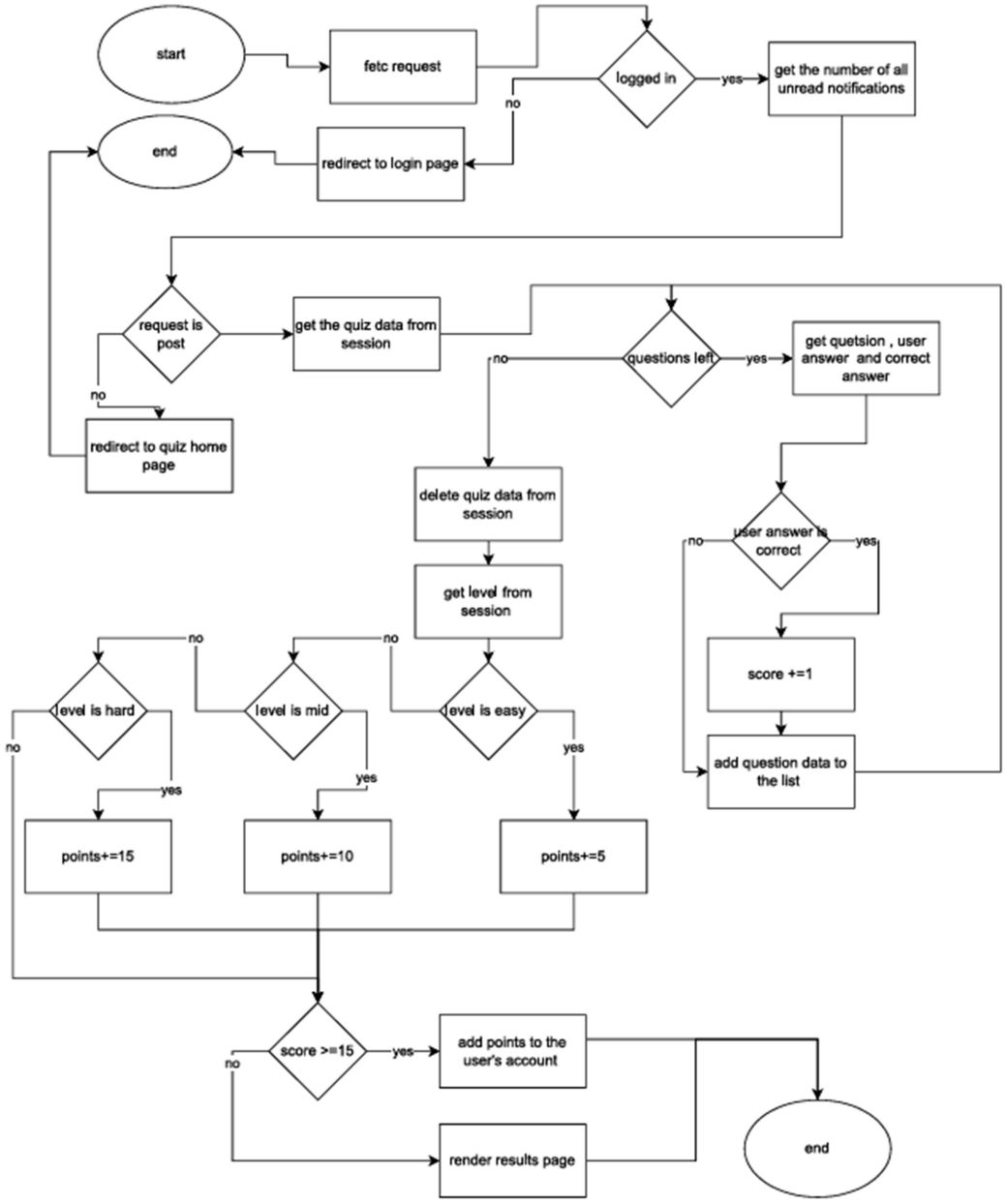
Move_image view : this view moves the images to another folder (creates it if doesnt already exist) based on the diagnosis provided by the analyst (if correctly labeled moved to the label's folder , if not move to the corrected label's folder) then deletes the image from the database and returns a json response for interactivity (no need to reload the page).



Quiz_home view : is just like home but for the quizzes so a user can set difficulty and start the quiz no more no less.



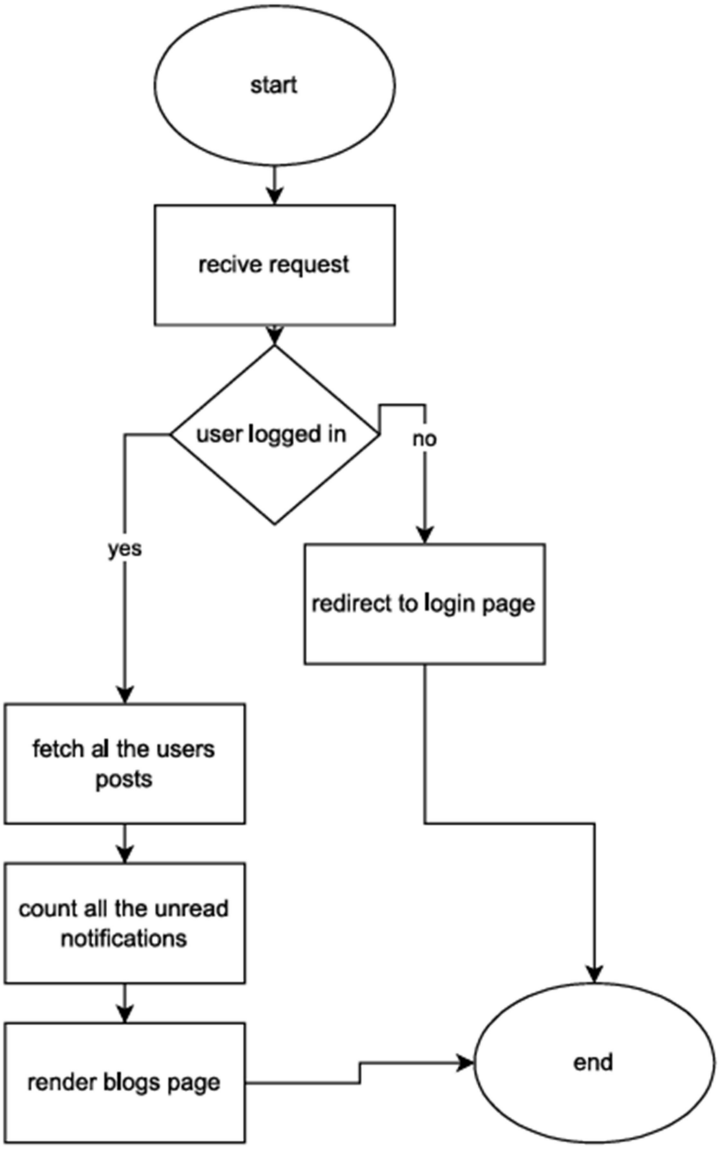
Run_quiz view: its starts by preparing the quiz's questions based on difficulty level chosen by user , it puts questions in a set to assure uniqueness where it samples 20 questions of all the questions in the set , it then prepares the questions for the user by grabbing each question with the user's answer and the correct answer for the questions and stores them in the session with the level of the quiz before using them in the process_quiz view.



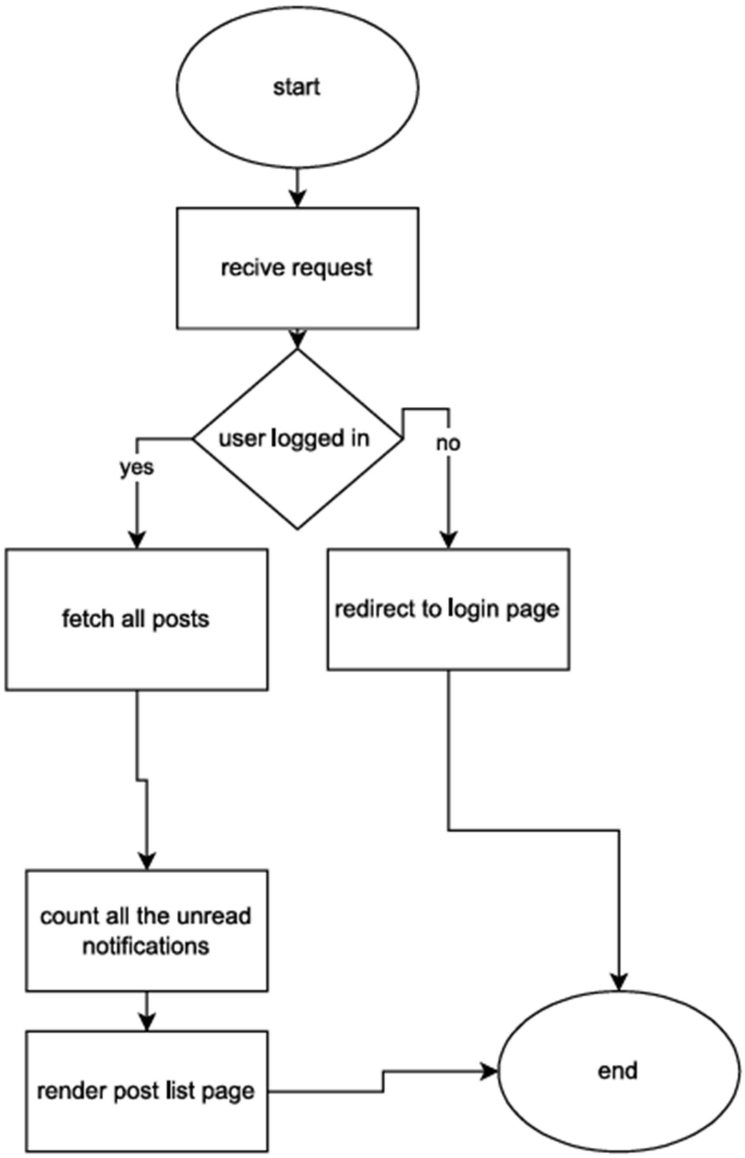
Process_quiz view: this view is responsible of processing ,scoring and giving points for the user based on their performance in the quiz , it starts by grabbing the quiz data from the session , then it iterates through the list of questions and checks which question's answer is similar to the correct answer and adds 1 score to the score counter , then it adds the question ,user's answer, correct answer and the result (correct/wrong) to a list,

It then deletes quiz data from the session, and if the user got 15 or more correct answers it adds points based on the difficulty level to the user's account score and saves it .

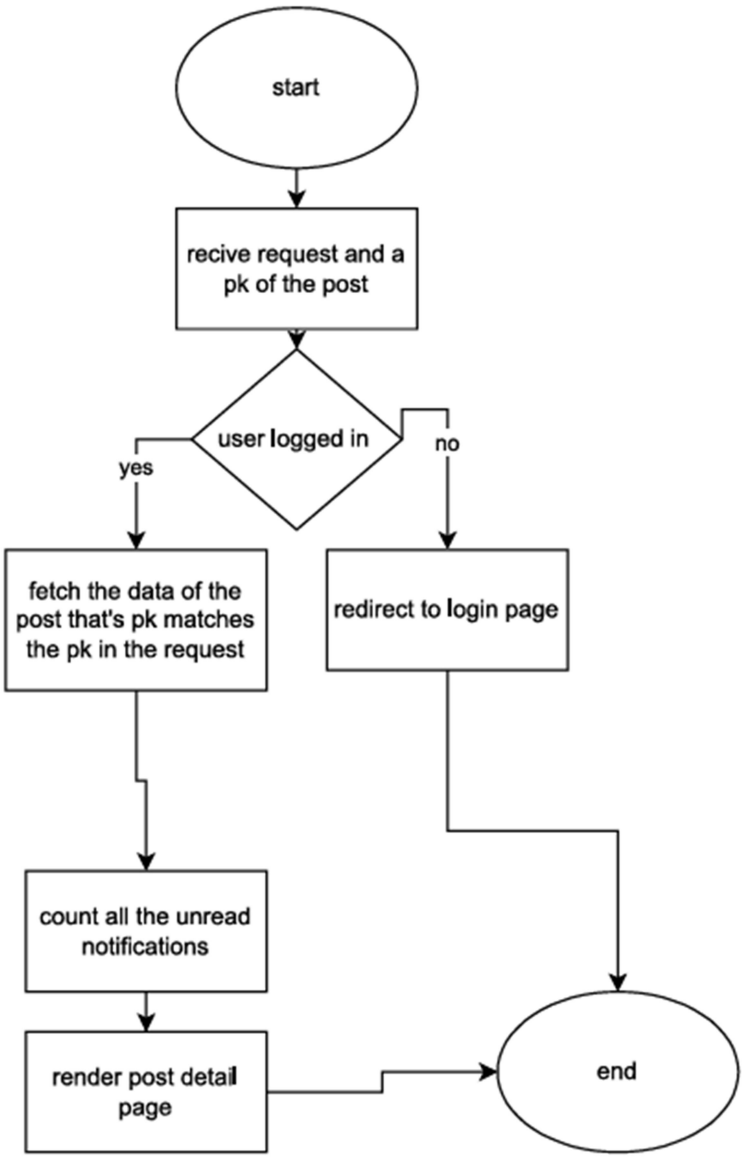
Blogs app



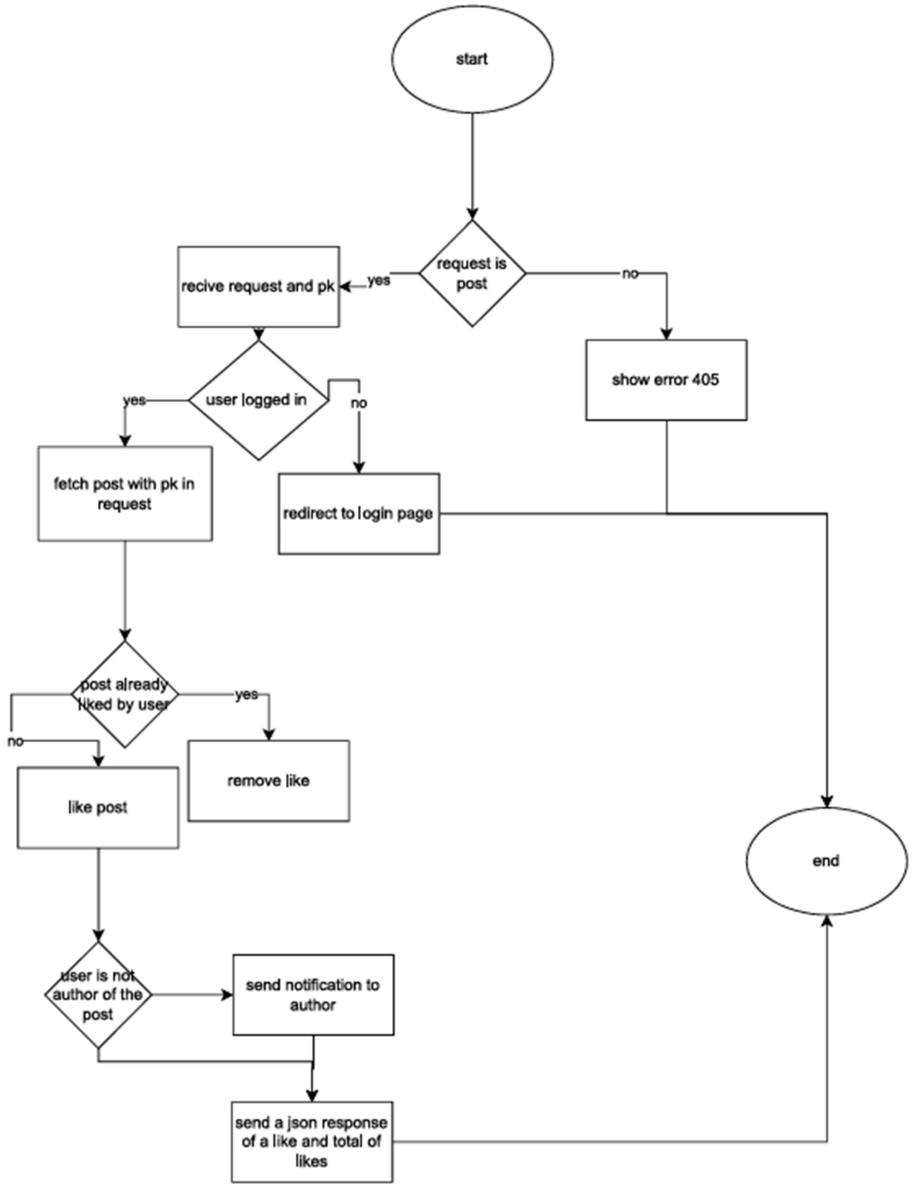
Blogs view: shows the user all their posts (if any exist).



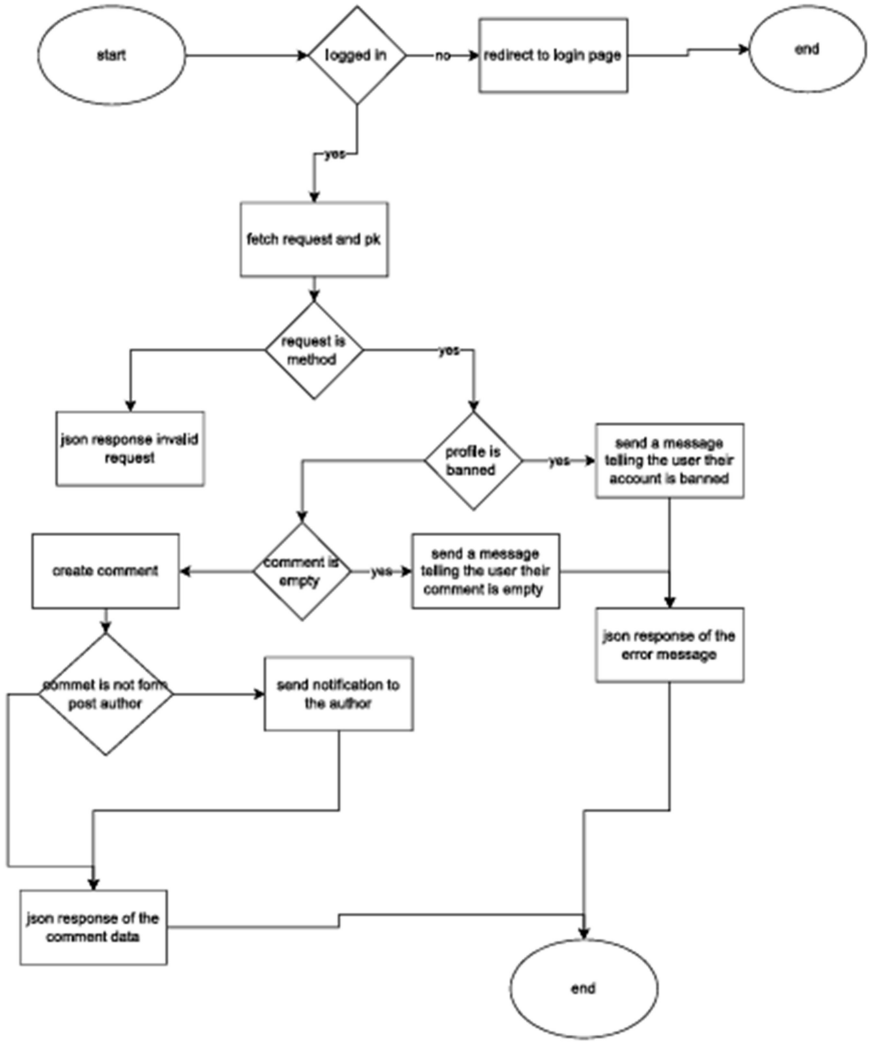
Post_list view: shows all posts ordered chronologically



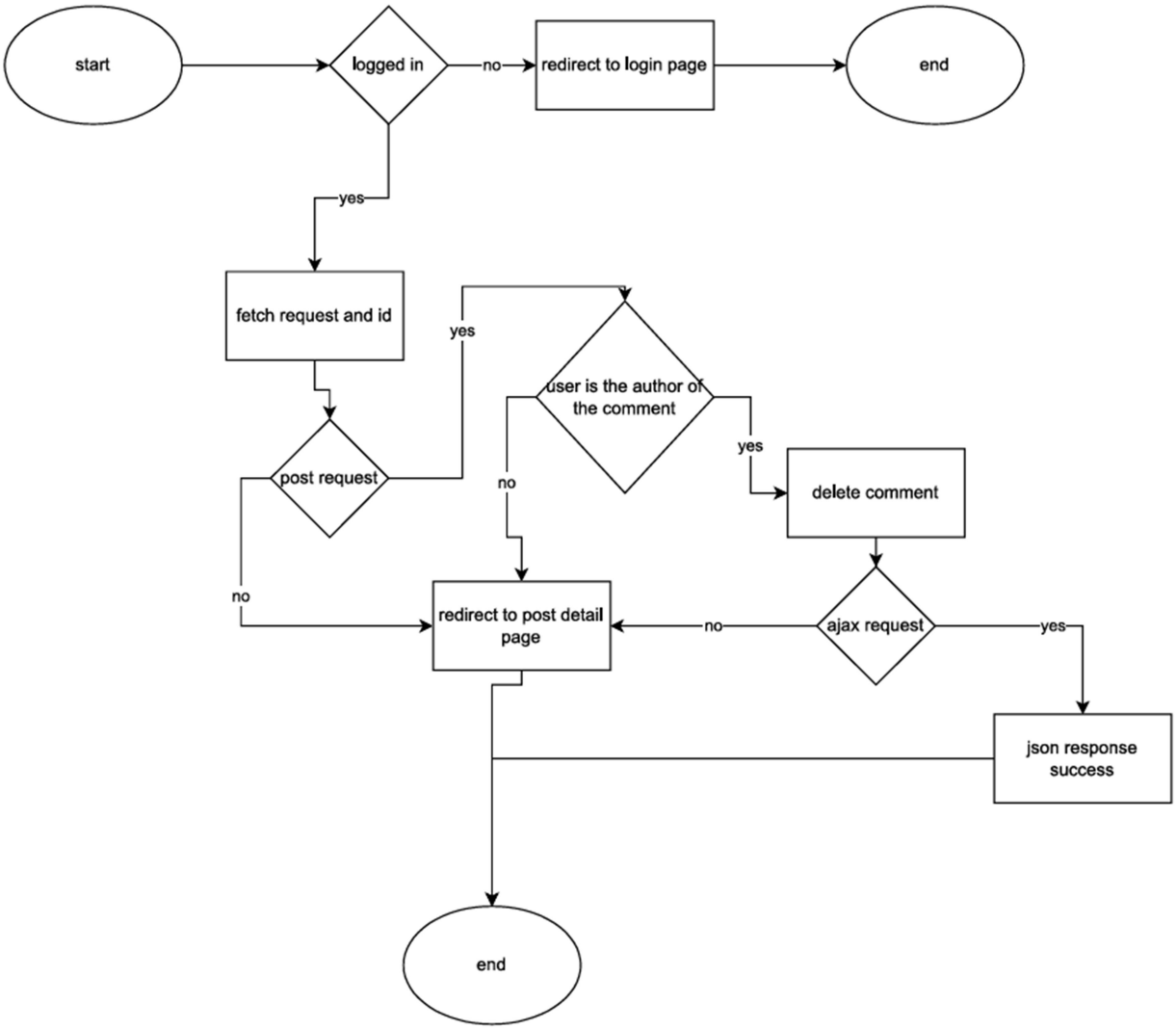
Post_detail view: show the details of the post



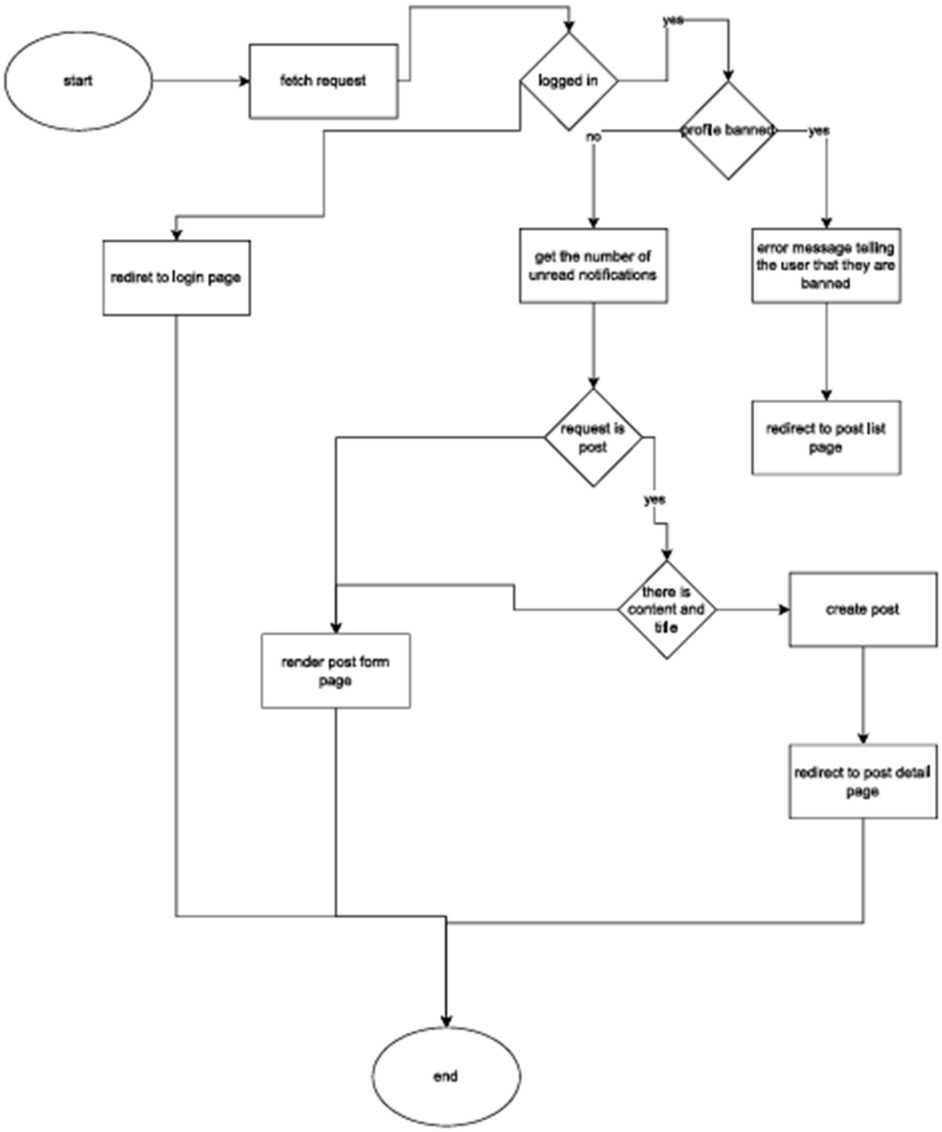
Like_post view: it allows the user to like a post if not already liked, otherwise it removes the like from the post , it also returns a json response for responsivity (no reload upon action).



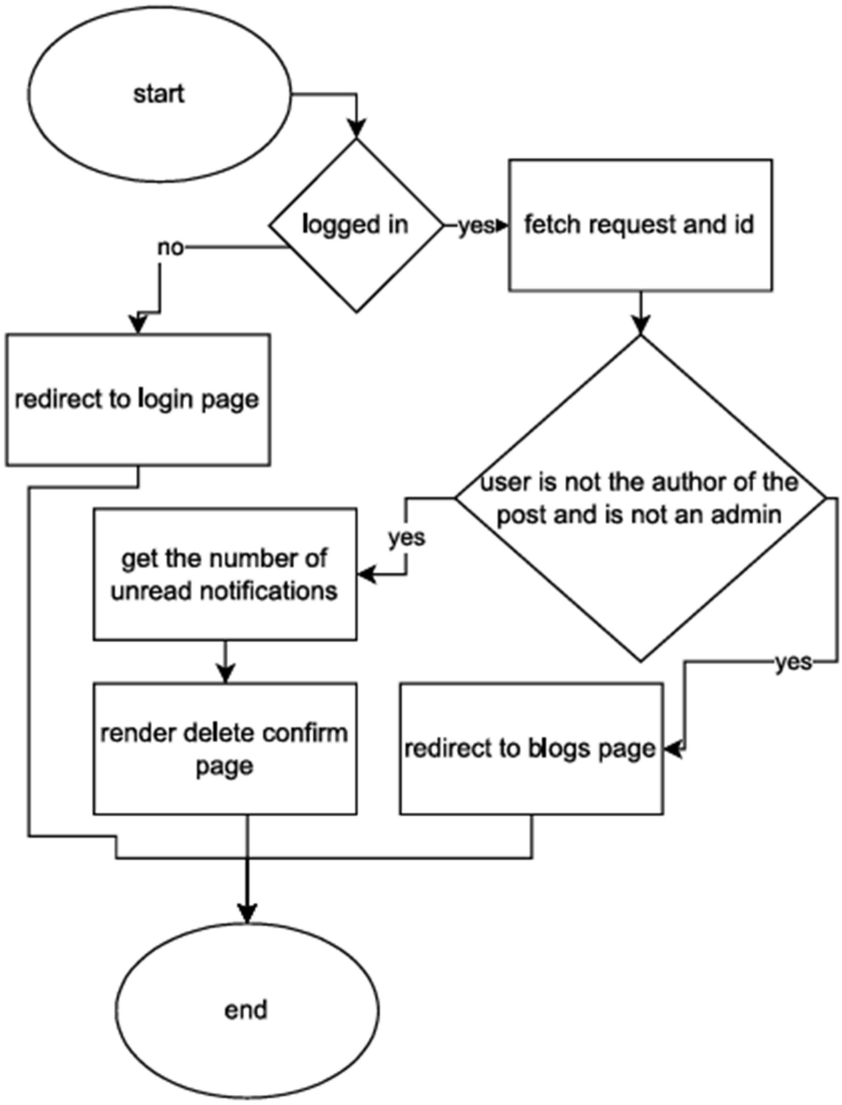
Add_comment view: prohibits the user from adding a comment in case they are banned ,otherwise it adds a comment and if the user is not the post author it send a notification to the post author , in addition to using a json response for responivity (no reload upon action)



Del_comment view: allows the user to remove their comment from a post and uses a json response for responsivity (no reload upon aciton)

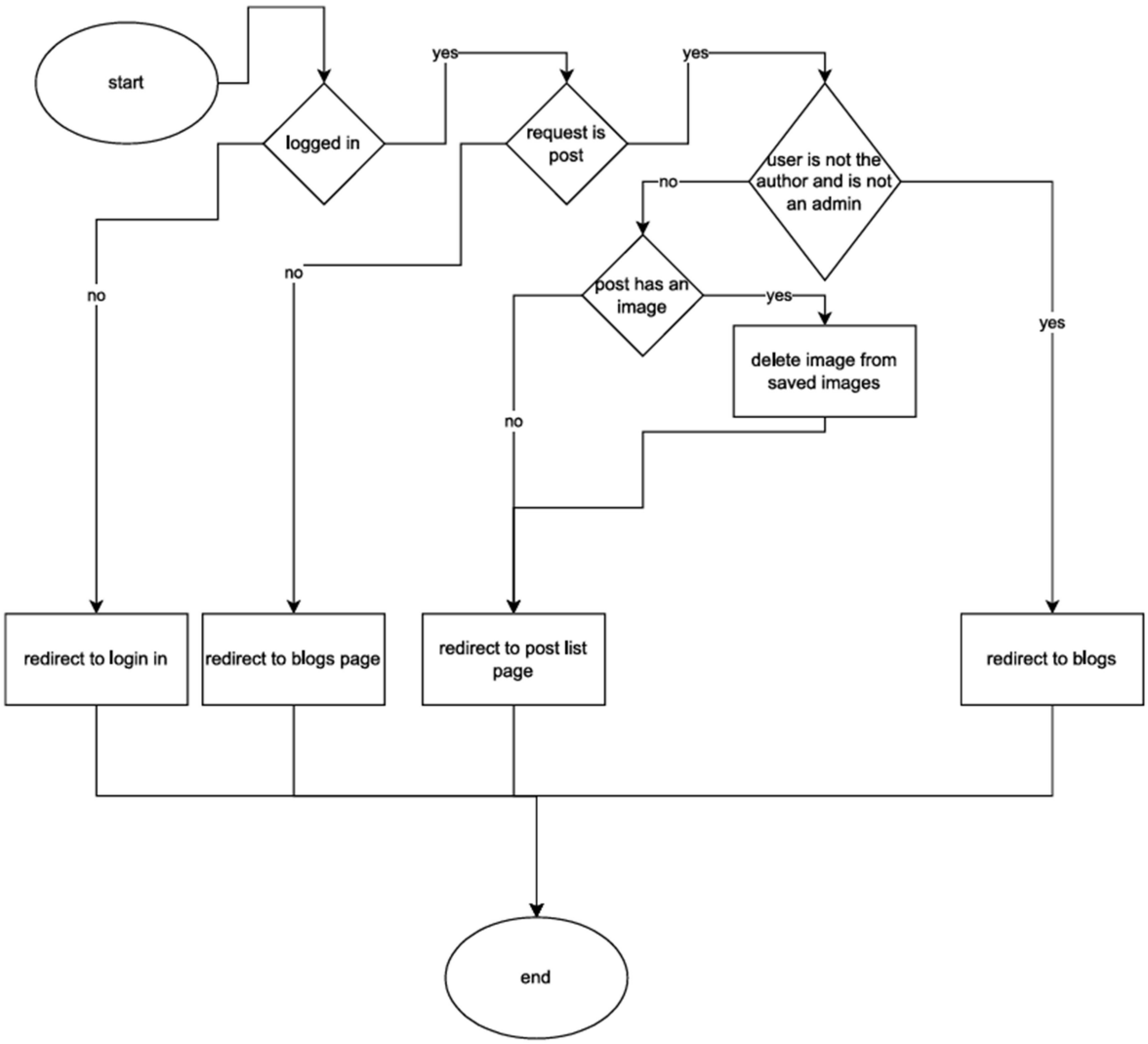


Create_post view: it prohibits the user from creating a post if they were banned , otherwise it allows them to create a post with content and title for the post , and optionally an image

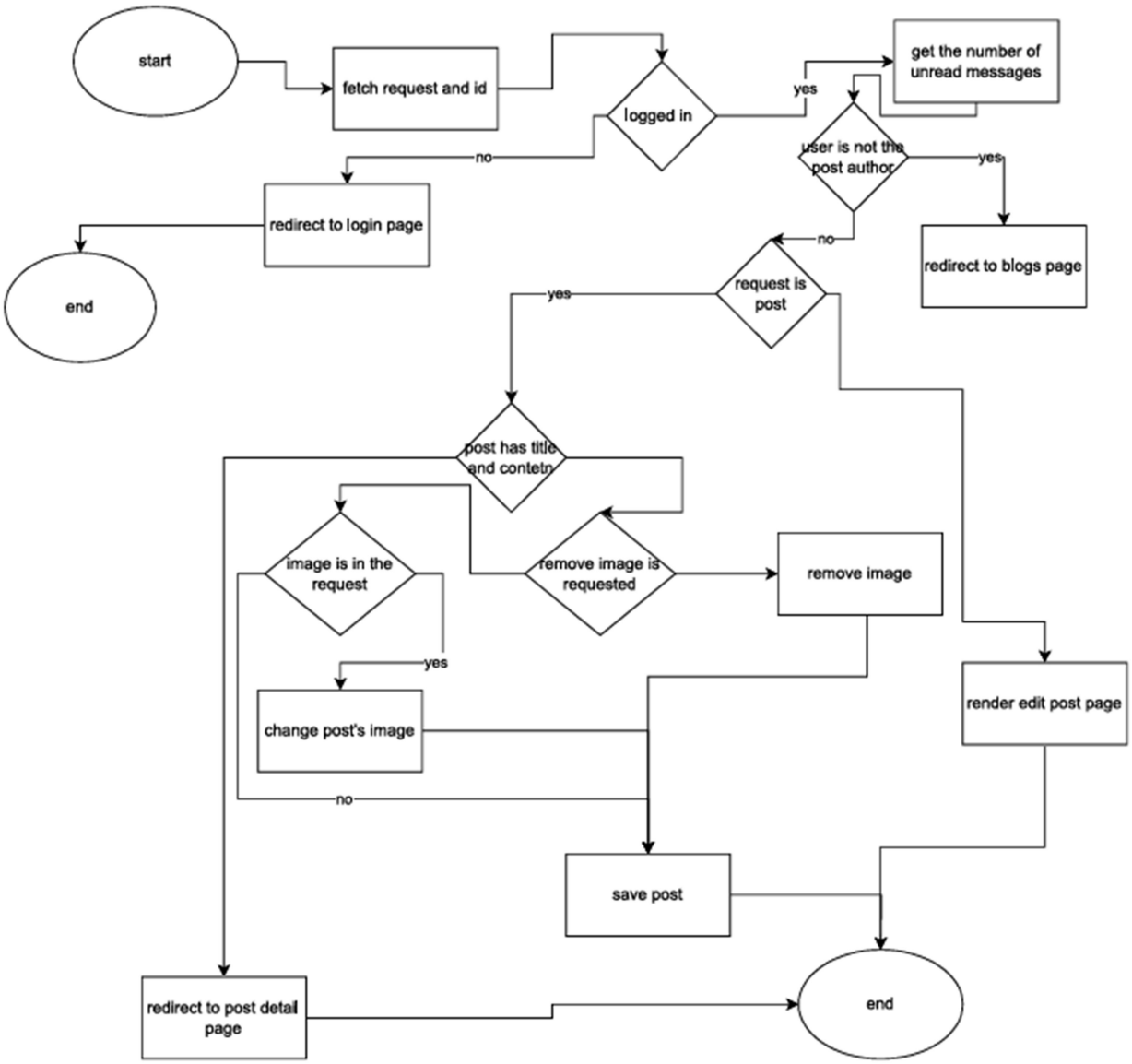


Del_confirm view : renders the delete confirmation page for the post (using the post's ID)

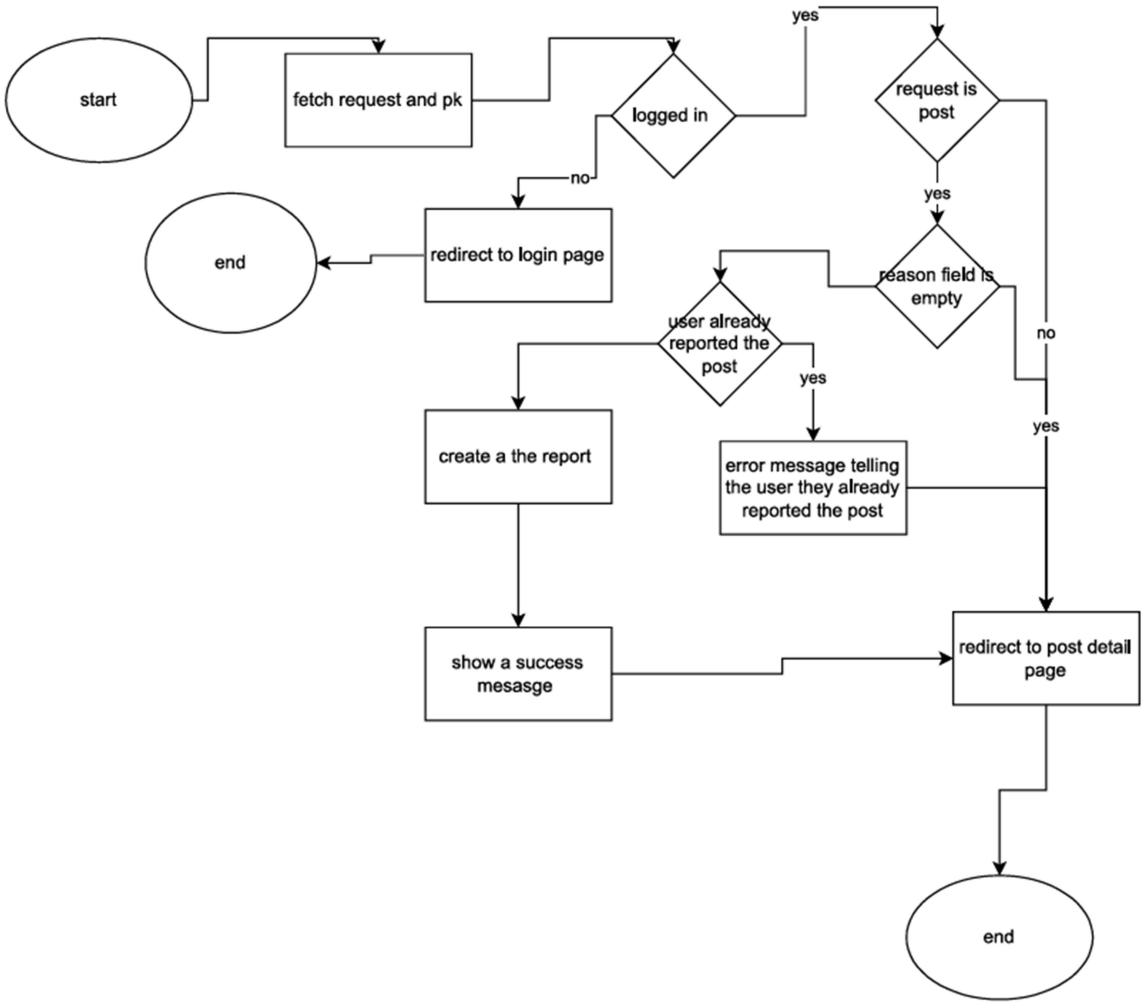
Where the user can delete the post (or the admin)



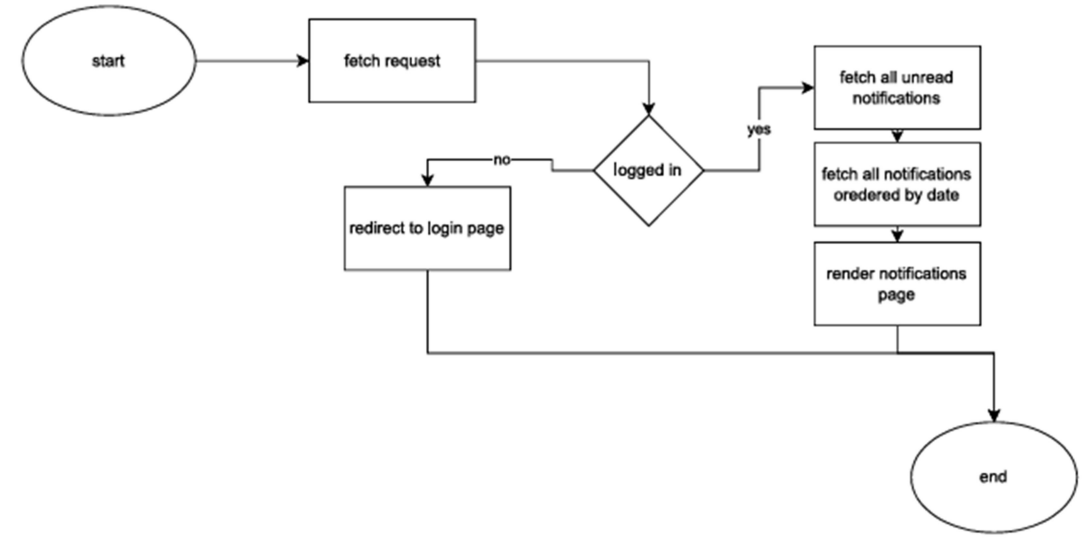
Del_post view: if the user is the post author or is an admin they can delete the post (and the image so it doesnt stay on the sever)

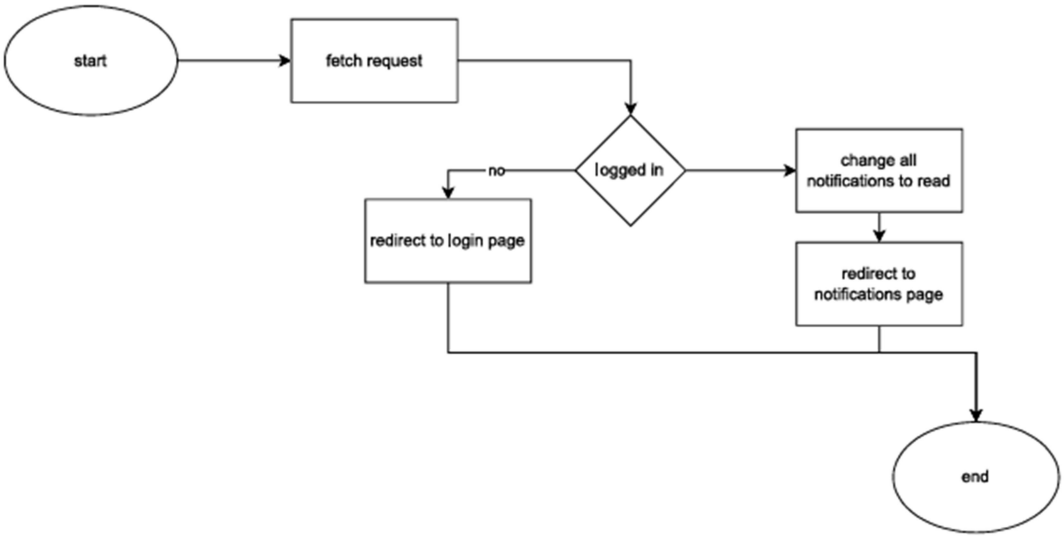


Edit_post view: the post author can edit the post that they have already posted where they can change title, content or change or remove the post's image



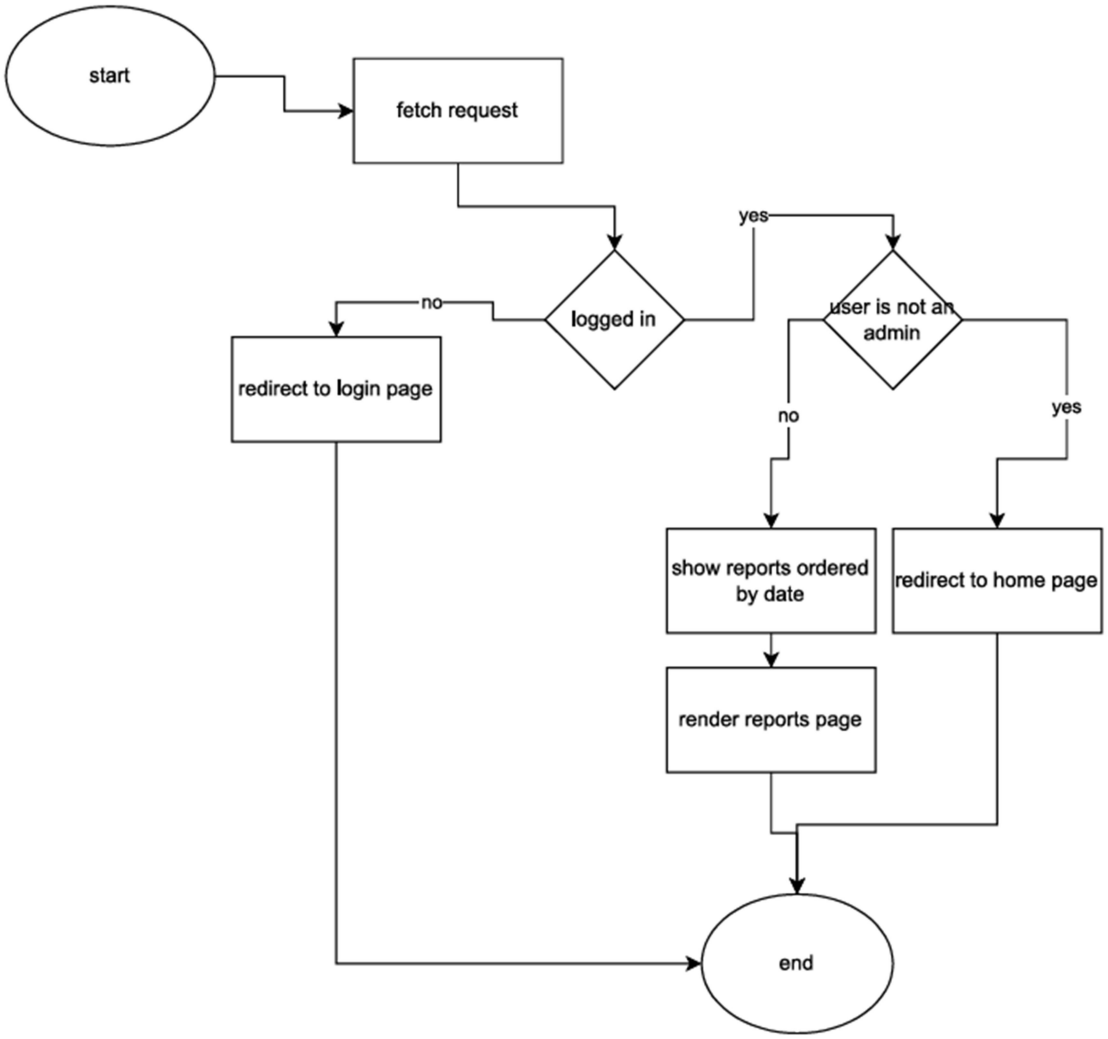
Report_post & report_comment views : they both do exactly the same thing except that one will grab a post and the other will grab a comment if the user didn't already report the post/comment they can send a report where the admin will inspect the report request.



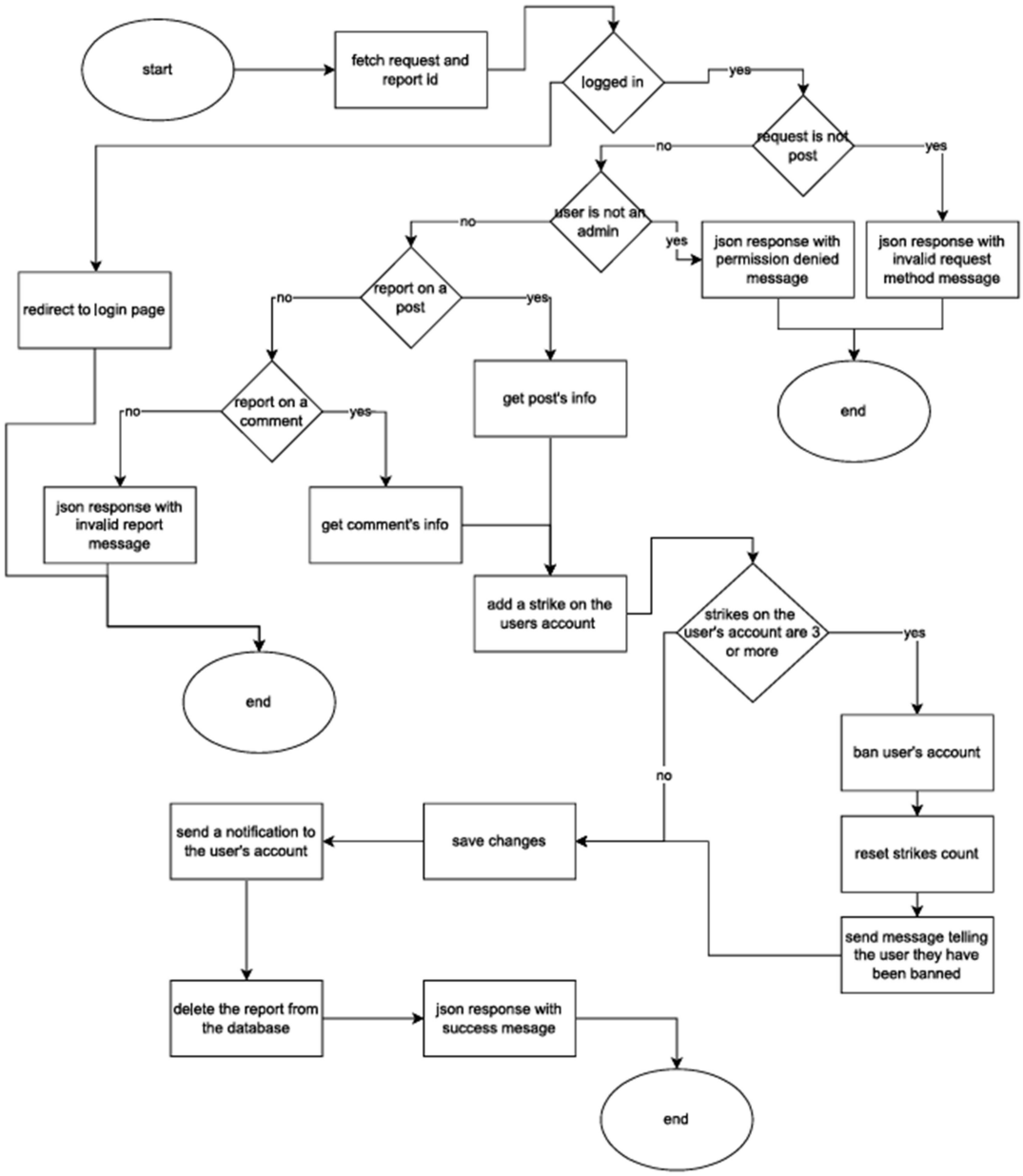


Notifications view (the top one) : will fetch all the user's notifications in a chronological order.

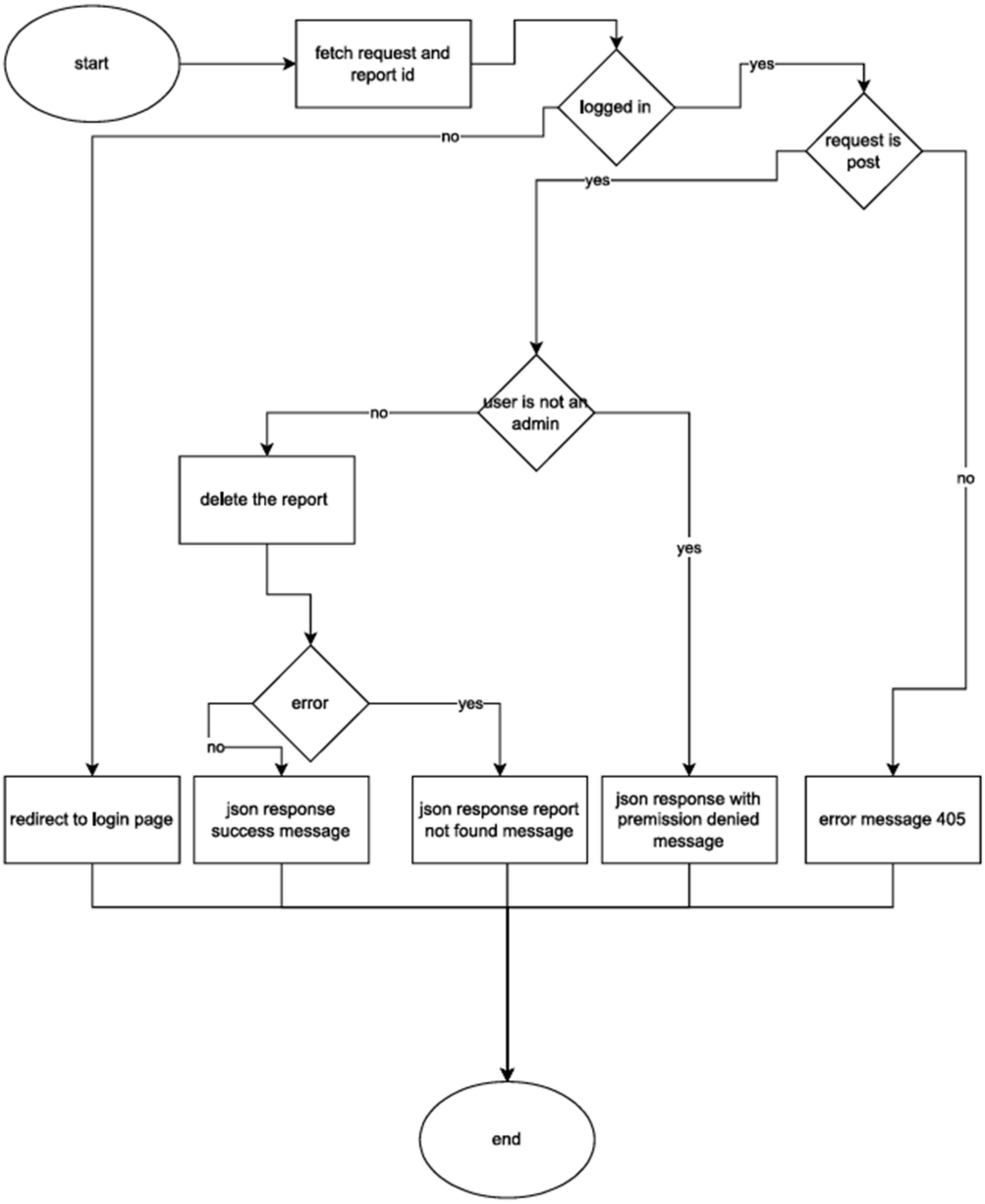
Mark_all_Read view (the bottom one) : will change the state of all unread messages to read.



View_reports view: admin based view(only admin can see it) where it fetch's all the reports for the admin to inspect them , where reports can be marked as (send report/ignore).



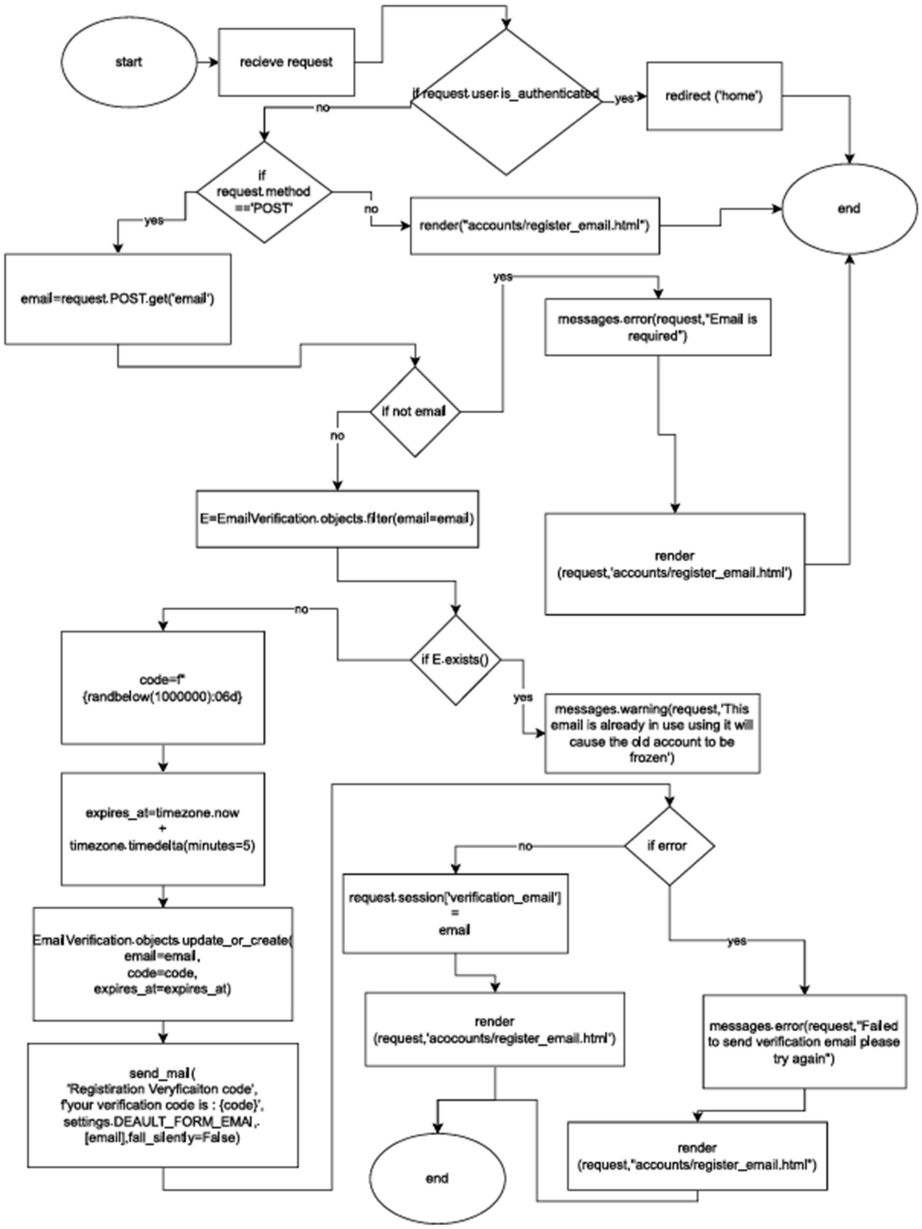
Send_report view: lets the admin send a notification to the reported user's account telling them their post/comment have been reported and sets a strike on the account ,
if a user got 3 strikes they get a 10 mins ban where they get prohibited from some functionalities.



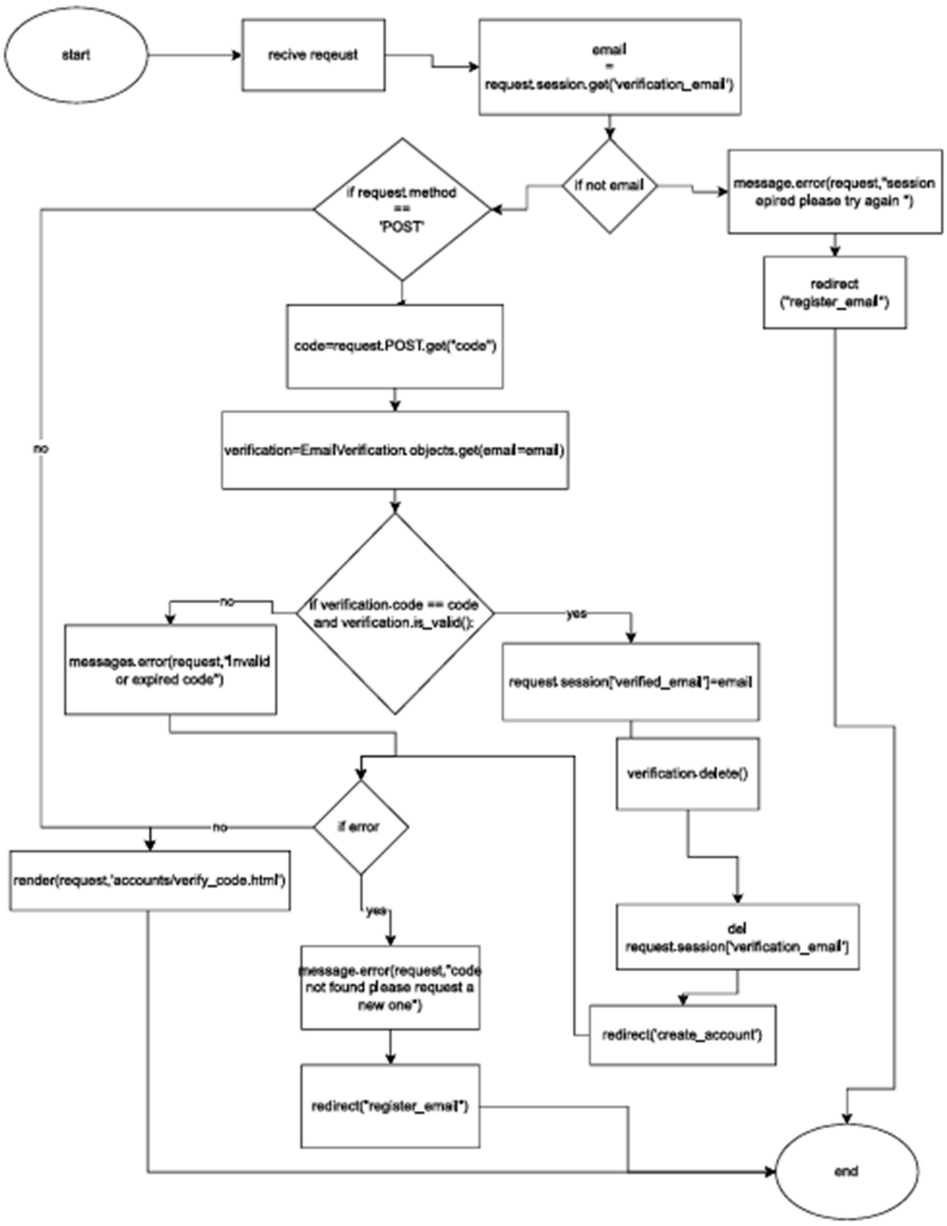
Ignore_report view: deletes the report without doing any action against it (in case the user didn't violate any terms of use).

Accounts app

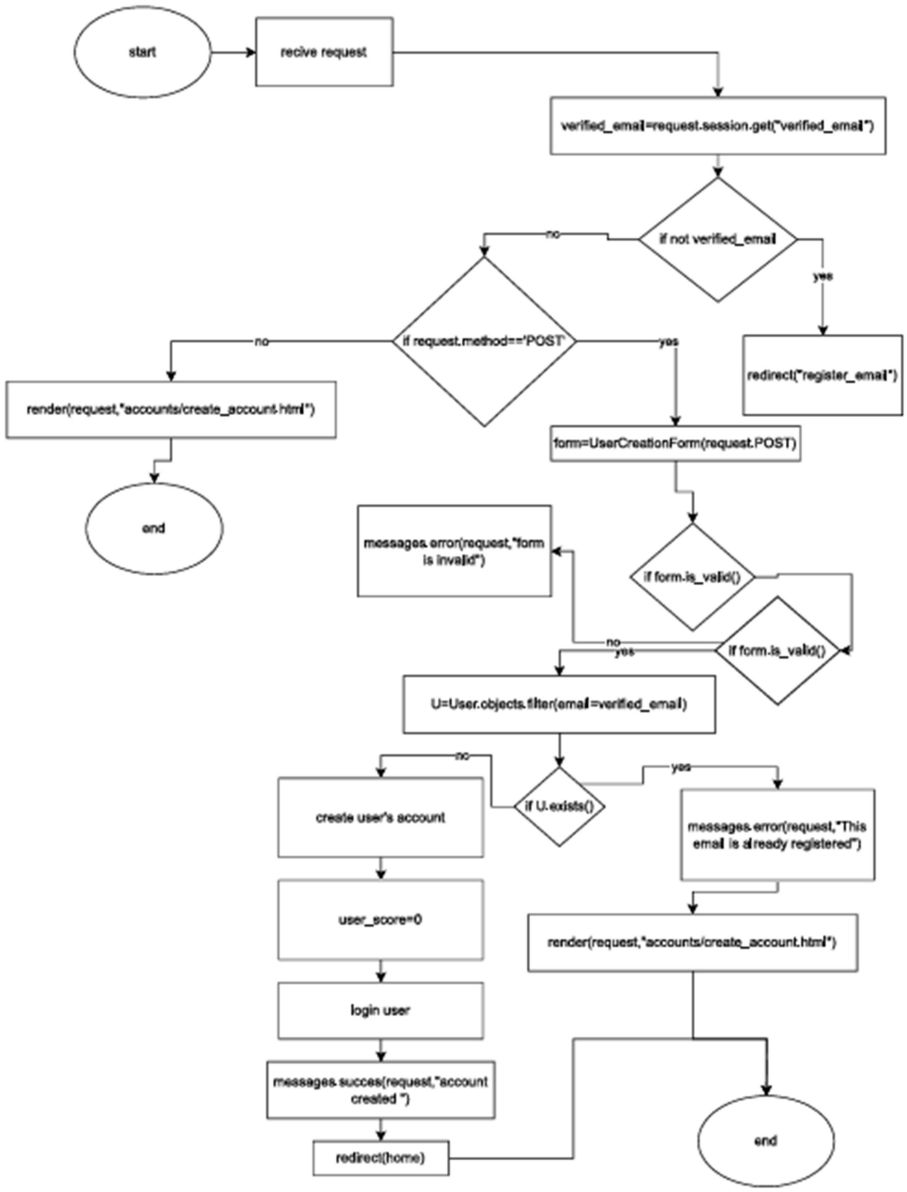
Manages user's accounts (login and registrations)



Register_email_view view: after the user uses their email , this view checks if the email is already used and if so, it displays a warning message telling the user that the email is in use ,then it sends a code to the user's email for verification, afterwards it stores the user's email in the session for usage in the verify_code_view.



Verify_code_view view: allows the user to go to account creationg if the code matches the one sent to their email.

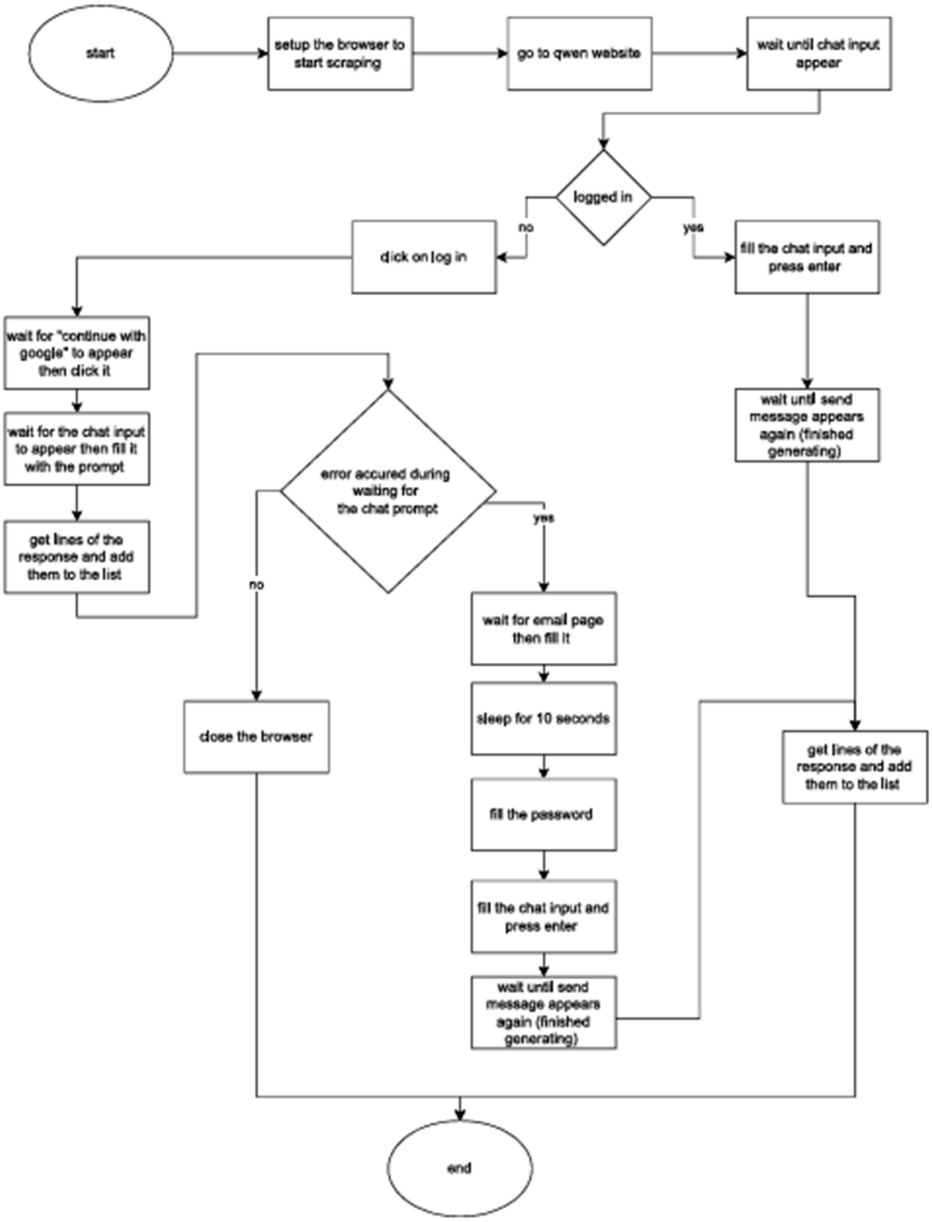


Create_account_view view: allows the user to create an account with a unique username

And if the password and password2 match , after creating the account it creates a score for the user with an initail value of 0 for future use (for the quizzing system) , after creating it authenticates the user (login the user) and redirects them to the home page.

QwenScraper

This is used to web scrape qwen ai so it fetch's the trivia facts for the user.



If the user is logged in fill the chat input and scrape the output , if the user is not logged in ,

Login the user and after that fill the chat input and scrape the output.

For the ML model:

1. Core Components

Libraries :

TensorFlow/Keras : For building/training the neural network.

ImageDataGenerator : Applies data augmentation (rotations, flips, brightness) and loads images in batches.

MobileNetV2 : Pretrained CNN (ImageNet weights) used as a feature extractor.

Scikit-learn : Computes class weights (to handle imbalanced datasets) and evaluation metrics (precision, recall, confusion matrix).

Seaborn/Matplotlib : Visualizes training metrics and confusion matrix.

2. Key Techniques

Transfer Learning :

Uses MobileNetV2's pre-trained weights, freezing most layers initially.

Unfreezes the last 50 layers for fine-tuning in Phase 2.

Regularization :

Dropout : Prevents overfitting (50% and 70% dropout rates).

Batch Normalization : Stabilizes training.

L2 Regularization : Applied to dense layers to penalize large weights.

Label Smoothing : Softens labels in the loss function (CategoricalCrossentropy(label_smoothing=0.1)) to improve generalization.

Class Weights : Balances training via compute_class_weight (inverse frequency of classes).

3. Training Workflow

Phase 1 :

Trains only the newly added top layers (dense/BatchNorm/dropout) with a low learning rate (1e-5).

Uses early stopping and learning rate reduction to avoid overfitting.

Phase 2 :

Unfreezes deeper MobileNetV2 layers and fine-tunes with an even lower learning rate (1e-6).

Saves the best model based on validation accuracy.

4. Prediction Pipeline

predict_image() :

Loads and preprocesses an image (resizes, normalizes).

Returns predicted class and confidence score.

6. Configuration Details

Image Size : 224×224 (standard for MobileNet).

Batch Size : 16 (small for limited GPU memory).

Seed : Ensures reproducibility (seed=39).

Paths : Requires train and validation directories with class-labeled subfolders.

□Project Tools Overview

In this project, I utilized a combination of programming languages, frameworks, libraries, and custom tools to develop a dynamic and functional web application. Below is a breakdown of the tools used:

1. Backend Development: Python & Django

- **Python** : The primary programming language used for both backend logic and utility scripts.
- **Django** : A high-level Python web framework that enabled rapid development, clean design, and secure handling of database operations, URL routing, and views.

Key Features Used in Django :

- ORM (Object Relational Mapper) with SQLite3 (default database)
- Template Language (DTL - Django Template Language)
- Views, URLs, and Forms
- Built-in admin panel for managing data

2. Frontend Development: HTML, JavaScript & Tailwind CSS

- **HTML** : Used to structure the content and layout of the web pages.
- **JavaScript** : Added interactivity and handled client-side logic, especially for AJAX requests.
- **Tailwind CSS** : A utility-first CSS framework that allowed for rapid UI development with responsive and modern designs.

3. AJAX Integration

- Implemented AJAX (Asynchronous JavaScript and XML) to allow asynchronous communication between the frontend and backend without reloading the page.
- This improved user experience by enabling partial updates and real-time interactions.

4. Utilities & Libraries

A variety of built-in and third-party Python libraries were used for different functionalities:

module	purpose
`secrets`	Generating secure tokens and keys
`random`	Creating random values (e.g., prompts)
`os`&`shutil`	File system manipulation (create, move, delete)
`tensorflow`	AI/ML-related features or image predictions

5. Database: SQLite3

- Used SQLite3 , Django's default database engine.
- Suitable for small-scale applications and local development environments.
- Easy setup and maintenance without needing external database servers.

6. Custom Tools Developed

I also created some internal tools to assist with development and testing:

a. **QwenScraper**

- A web scraping tool designed to extract data from Qwen AI.
- Built using Python (built with playwright).
- Helped gather real-time generated data.

b. **PromptGen**

- A prompt generation tool that creates various types of prompts automatically.
- Useful for testing AI response behaviors or populating datasets.
- Uses randomness to create diverse prompts (solved AI's deterministic answers).

c. **Zypher (LLM)**

- Used in case QwenScraper failed for any reason ,to generate real-time data .

☑Why These Tools Were Chosen

- Django provides a solid foundation for full-stack apps with minimal configuration.
- Tailwind CSS allows for fast and consistent styling without writing custom CSS.
- Using AJAX improves performance and user interaction.
- Custom tools like Qwen Scraper and PromptGen helped automate tasks and reduce manual effort during development.
- QwenScraper was the choice for real-time data gathering
- Zypher is the replacement if QwenScraper failed
- PromptGen solved deterministic answers problem

Planter:

Results

Project Results: Planter – A Technology-Driven Platform for Plant Health

Overview

Planter successfully launched as a digital ecosystem that bridges technology, agriculture, and education. By combining AI-powered plant disease diagnosis with community-driven knowledge sharing, the platform empowers farmers, gardeners, and plant enthusiasts to protect crops, learn sustainably, and collaborate globally.

Key Achievements

- AI-Powered Plant Disease Diagnosis
- Developed an AI tool that identifies plant diseases using a smartphone photo of a leaf.
- Achieved 78% accuracy in diagnosing diseases across 6+ plant species, with results delivered in ≤10 seconds .
- Reduced reliance on slow, lab-based testing (which can take weeks) by providing instant, accessible solutions for rural and smallholder farmers.
- Built a scalable model trained on 2,000+ labeled images , ensuring adaptability to diverse crops and environments.
- Community Knowledge Hub
- Created a moderated blogging platform where users share verified tips, success stories, and challenges in plant care.
- Implemented a reporting and moderation system to combat misinformation, ensuring content remains accurate and trustworthy.
- Fostered a global community of farmers, students, and experts collaborating to solve agricultural problems.
- Gamified Learning Tools
- Launched an interactive quiz system with 400+ questions across difficulty levels, encouraging users to learn about plant health, history, and sustainability.
- Introduced bite-sized trivia facts (e.g., "Willow bark inspired aspirin!") to make plant science engaging and accessible.
- Tracked user progress with scores and rewards, promoting continuous learning and engagement.
- User-Centric Design
- Built a mobile-first, responsive platform usable on all devices, ensuring accessibility for rural communities with limited tech resources.
- Introduced role-based access (admin, analyst, normal user) to balance open collaboration with expert oversight.
- Impact on Agriculture and Education

For Farmers :

- Enabled early disease detection, potentially reducing crop losses and improving food security.
- Provided cost-effective, online tools for smallholder farmers who lack access to labs or agronomists.

From another perspective:

- Interactive experience, attracting students, hobbyists, and educators.
- Fast, reliable and free AI disease diagnosis for farmers, Gardeners and plant enthusiastic.
- Safe space to learn about anything and only about plants .
- Created a hub for plant safety and education.

Conclusion

Planter has proven that technology can democratize plant health management and education. By combining AI, gamification, and community collaboration, the platform addresses critical gaps in agriculture while fostering a global network of learners and growers. With iterative improvements, Planter is poised to become a cornerstone of sustainable farming and environmental advocacy worldwide.

Planter:

Prospective view

Planter – Strategic Vision & Technical Roadmap

1. Technical Innovations & Scalability

a. Advanced AI/ML Enhancements

Model Optimization:

Transition from MobileNetV2 to lightweight architectures like EfficientNet-Lite or Vision Transformers (ViTs) for higher accuracy and faster inference on edge devices (e.g., mobile phones).

Multimodal Inputs:

Combine image analysis with text-based symptom descriptions (e.g., user-reported symptoms like leaf discoloration) using multimodal neural networks .

Real-Time Feedback Loop:

Allow users to submit ground-truth labels (e.g., confirmed lab results) to retrain the model iteratively, improving accuracy over time.

b. AR/VR Integration

Augmented Reality (AR) Field Scanning:

Develop a mobile AR feature where users scan crops in real time, overlaying disease hotspots or treatment recommendations on the camera feed.

Example: Highlight infected areas on a leaf and provide localized solutions.

Virtual Plant Clinics:

Host AR/VR-based expert consultations for complex cases, connecting farmers with agronomists globally.

c. IoT Sensor Integration

Environmental Monitoring:

Partner with IoT sensor providers to integrate weather, soil moisture, and humidity data into diagnosis workflows.

Example: Predict fungal infections based on local humidity spikes.

Automated Alerts:

Use sensor data to trigger proactive alerts such as:

"High risk of powdery mildew in your area—apply fungicide now."

d. Multilingual & Global Expansion

Localized AI Models:

Train region-specific models for crops like rice (Asia), maize (Africa), or wheat (Europe) to improve accuracy for local diseases.

Language Support:

Expand the platform to include Arabic, Spanish, Hindi, and Swahili , targeting smallholder farmers in underserved regions.

Translate blogs, quizzes, and trivia into these languages using AI-powered tools .

2. User Engagement & Community Growth

a. Gamification & Social Features

Leaderboards & Challenges

Introduce competitive quizzes with global leaderboards and seasonal challenges (e.g., "Identify 10 diseases in a week").

Collaborative Learning

Add a "Community Diagnosis" feature where users vote on uncertain cases, fostering collective problem-solving.

User-Generated Content

Allow users to submit plant care tips, DIY remedies, or success stories, which analysts can verify and publish as official guides.

b. Monetization & Partnerships

Premium Features (still free)

Offer advanced analytics (e.g., crop yield predictions) or offline diagnosis capabilities .

Partnerships with Agri-Businesses

Collaborate with fertilizer/pesticide companies to recommend products directly after diagnosis (with ethical disclosures).

NGO/Government Programs

Partner with organizations like FAO or local agricultural departments to deploy Planter in rural training programs.

c. Educational Ecosystem

University Collaborations

Certify quizzes and trivia with agricultural universities, offering micro-credentials for users completing modules.

3. Sustainability & Environmental Impact

a. Climate Resilience Tools

Disease Outbreak Prediction

Use historical climate data and AI to forecast regional disease outbreaks (e.g., late blight during rainy seasons).

Carbon Footprint Tracking

Integrate a tool for users to track the environmental impact of their farming practices (e.g., reduced pesticide use via early diagnosis).

b. Open-Source Contributions

Open Dataset Sharing

Release anonymized diagnosis data to the research community to accelerate global plant health studies.

Developer Community

4. Strategic Roadmap

2025–2026

Launch multilingual support (Arabic, Spanish)

Integrate IoT sensors for environmental data

2027–2028

Deploy AR-based field scanning

Partner with 5+ universities for gamified content validation

2029–2030

Reach 1 million active users

Expand to 50+ crop species

Develop NGO/government dashboards for policy-making

5. Long-Term Vision

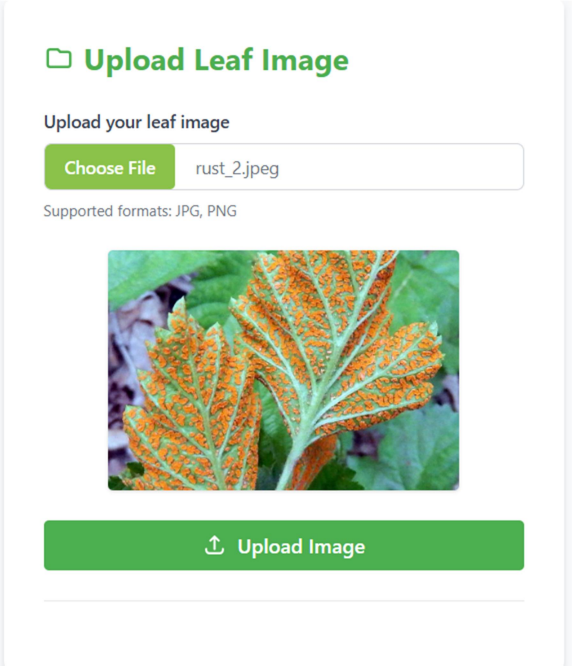
Planter aims to become the global standard for plant health diagnostics , bridging the gap between AI innovation and grassroots agricultural needs. By 2030, it could:

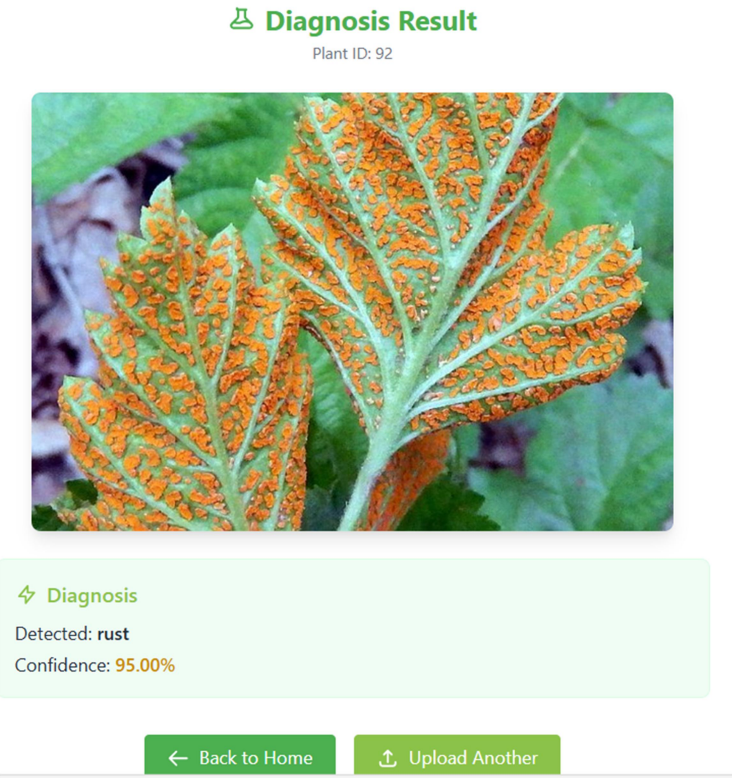
Reduce crop losses by in partner regions

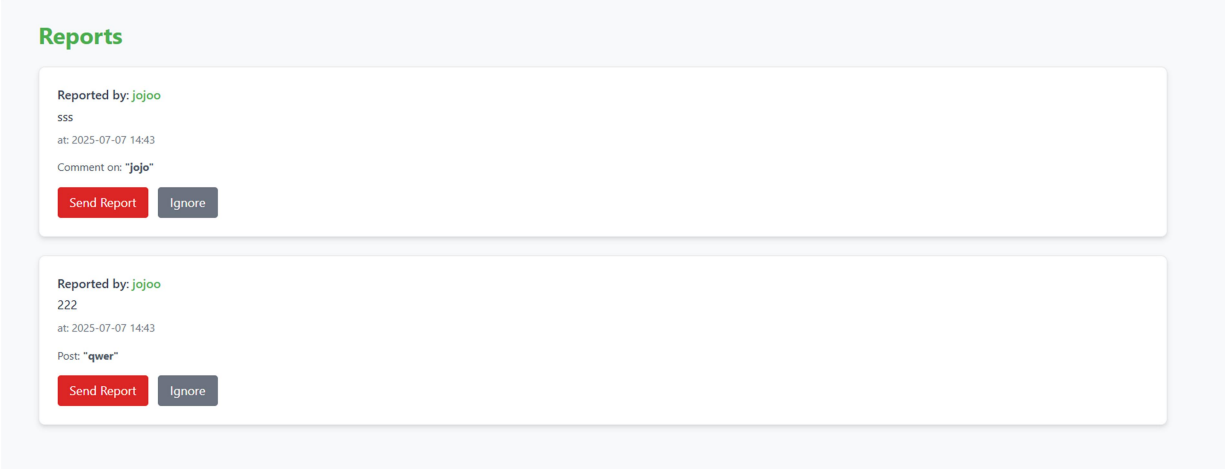
Empower farmers with instant, affordable diagnostic tools

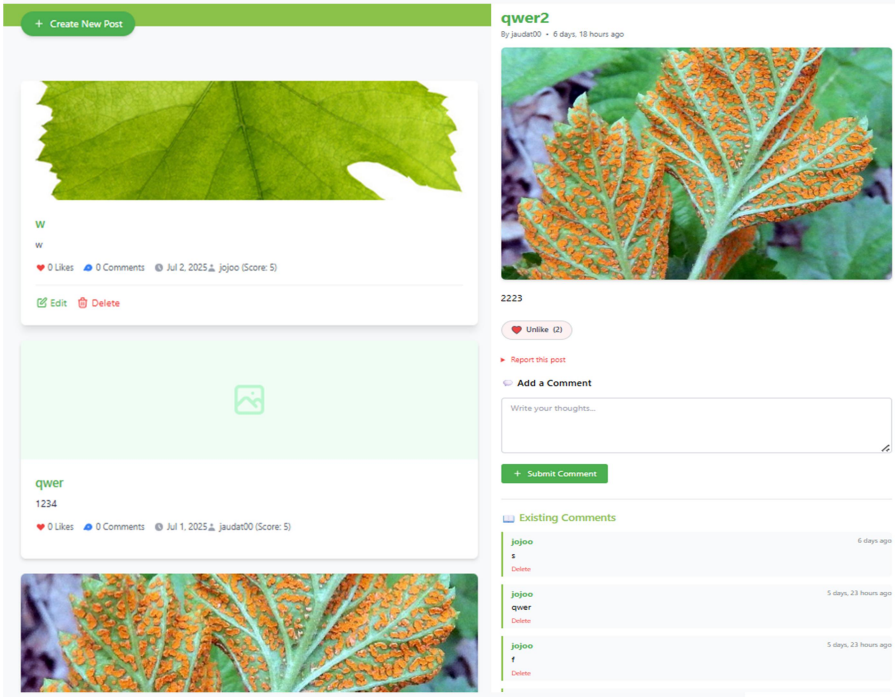
Foster a global community of plant enthusiasts driving sustainable farming practices

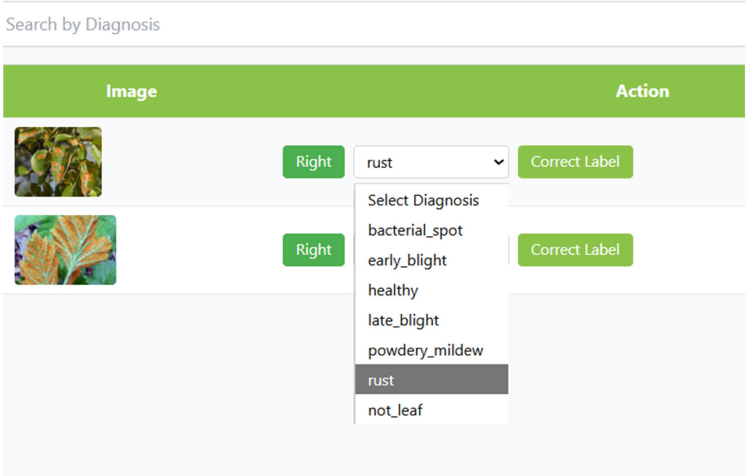
User interfaces

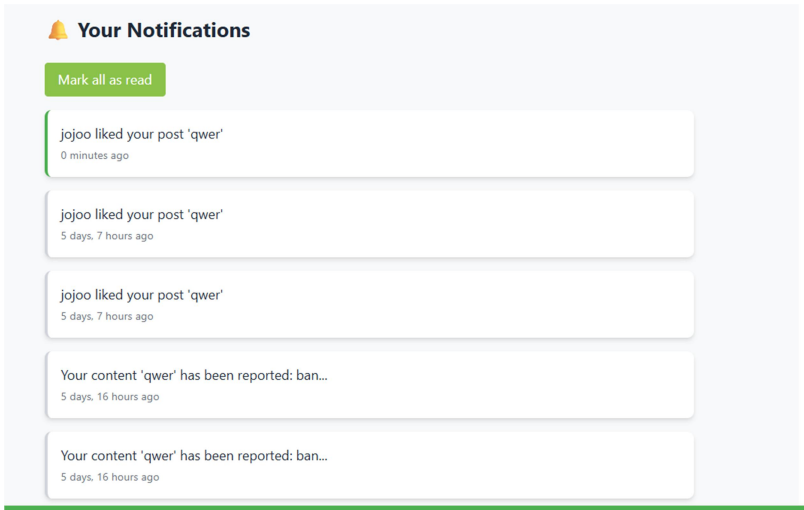


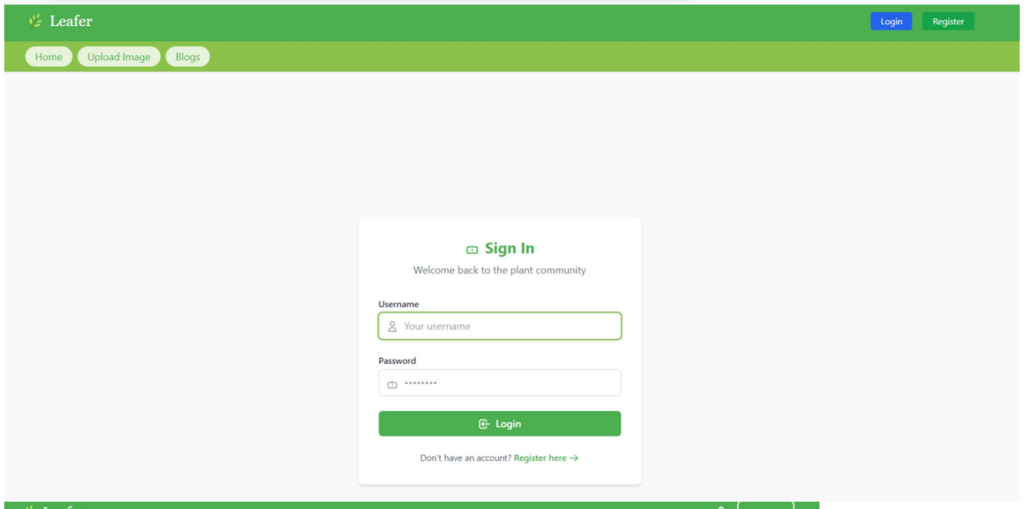


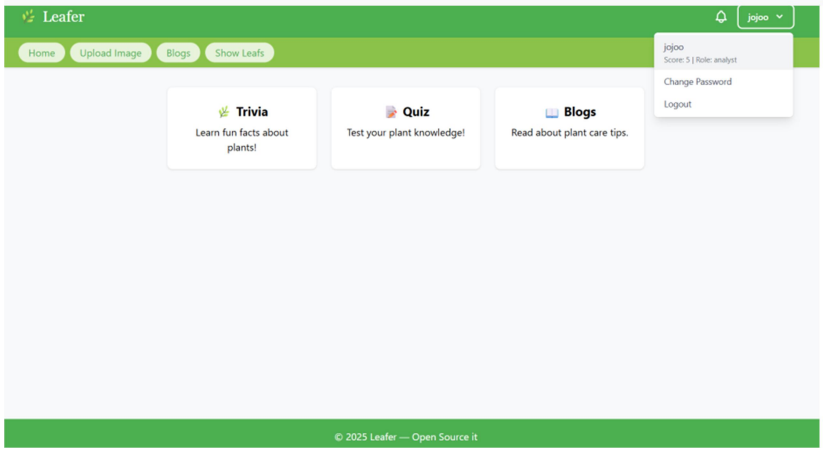

Reports
Reported by: jojoo
sss
at: 2025-07-07 14:43
Comment on: "jojo"
Send Report
Ignore
Reported by: jojoo
222
at: 2025-07-07 14:43
Post: "qwer"
Send Report
Ignore

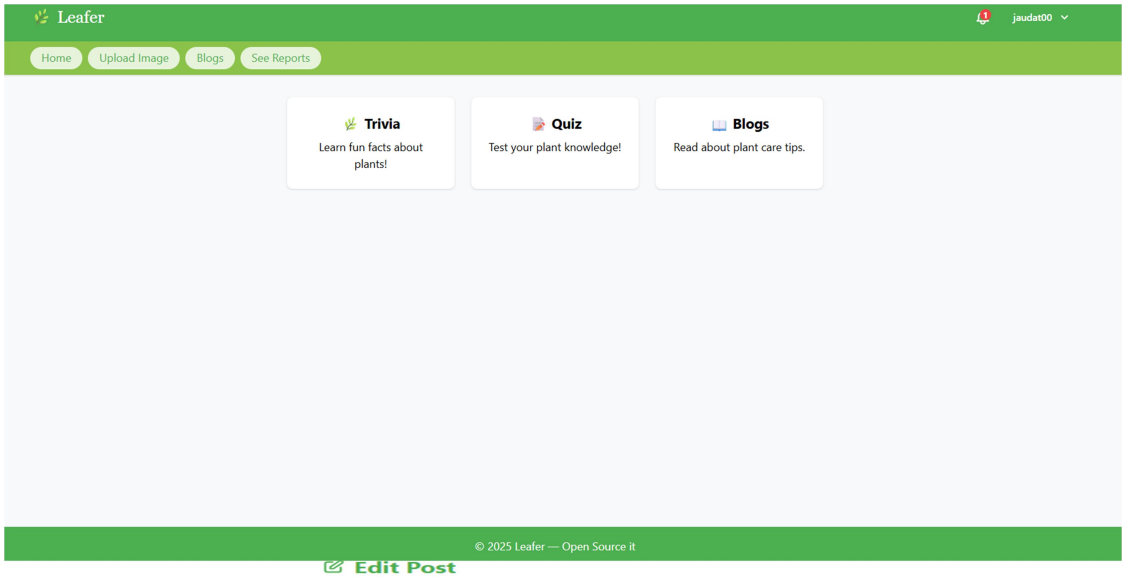

Create New Post
w
w
0 Likes
0 Comments
Jul 2, 2025
jojoo (Score: 5)
Edit
Delete
qwer
1234
0 Likes
0 Comments
Jul 1, 2025
jaudat00 (Score: 5)
qwer2
By jaudat00 • 6 days, 18 hours ago
2223
Unlike (2)
Report this post
Add a Comment
Write your thoughts...
Submit Comment
Existing Comments
jojoo
6 days ago
s
Delete
jojoo
5 days, 23 hours ago
qwer
Delete
jojoo
5 days, 23 hours ago
r
Delete

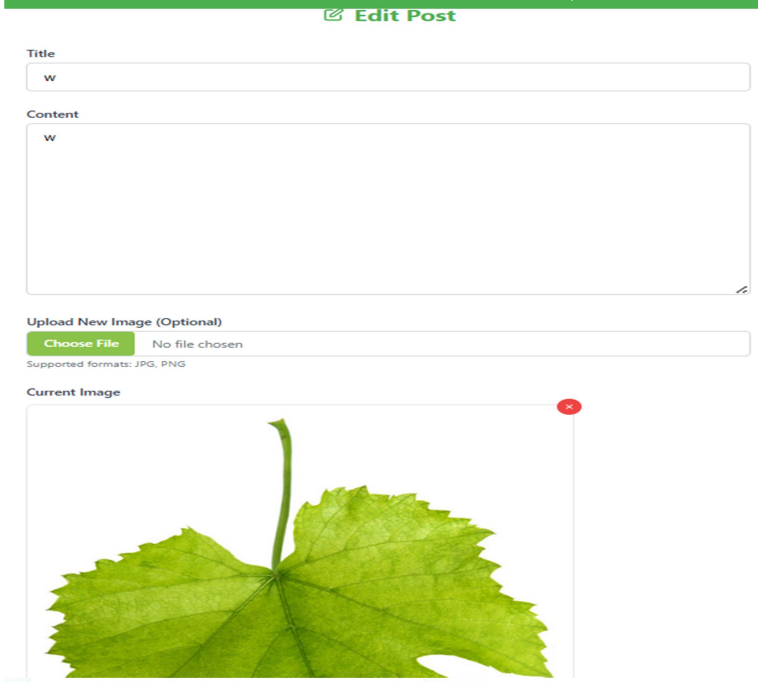

Search by Diagnosis
Image
Action
Right
rust
Correct Label
Select Diagnosis
bacterial_spot
early_blight
healthy
late_blight
powdery_mildew
rust
not_leaf
Right
Correct Label

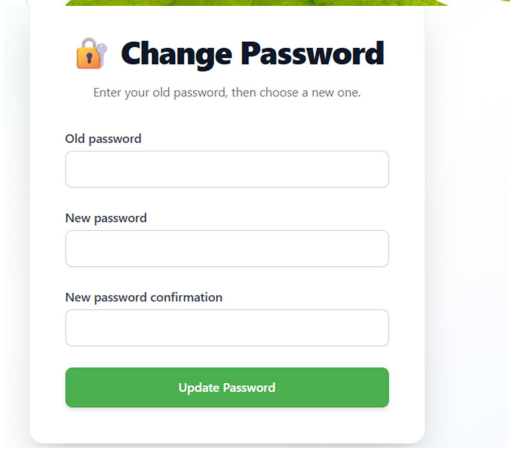

Your Notifications
Mark all as read
jojoo liked your post 'qwer'
0 minutes ago
jojoo liked your post 'qwer'
5 days, 7 hours ago
jojoo liked your post 'qwer'
5 days, 7 hours ago
Your content 'qwer' has been reported: ban...
5 days, 16 hours ago
Your content 'qwer' has been reported: ban...
5 days, 16 hours ago

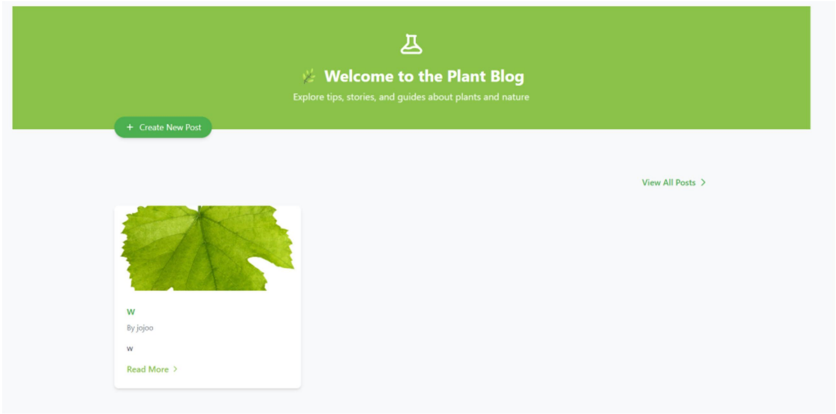

Leafer
Login
Register
Home
Upload Image
Blogs
Sign In
Welcome back to the plant community
Username
Your username
Password
Login
Don't have an account? Register here →

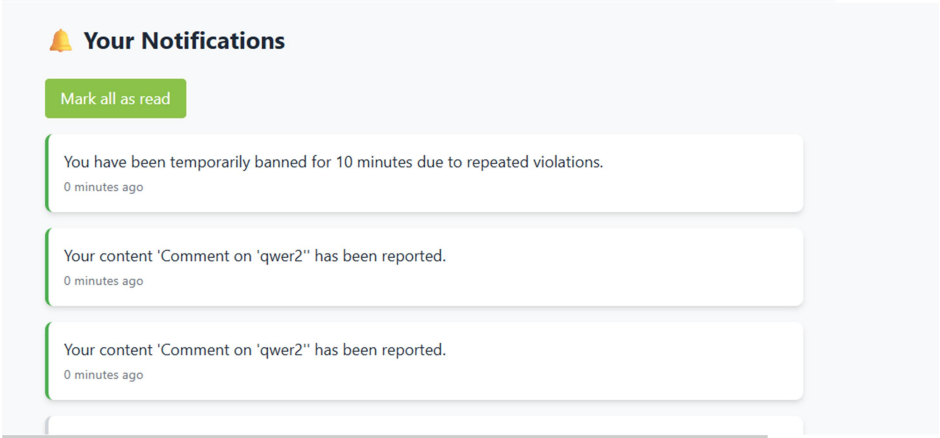

Leafer
jojoo
Home
Upload Image
Blogs
Show Leafs
jojoo
Score: 5 | Role: analyst
Change Password
Logout
Trivia
Learn fun facts about plants!
Quiz
Test your plant knowledge!
Blogs
Read about plant care tips.
© 2025 Leafer — Open Source it

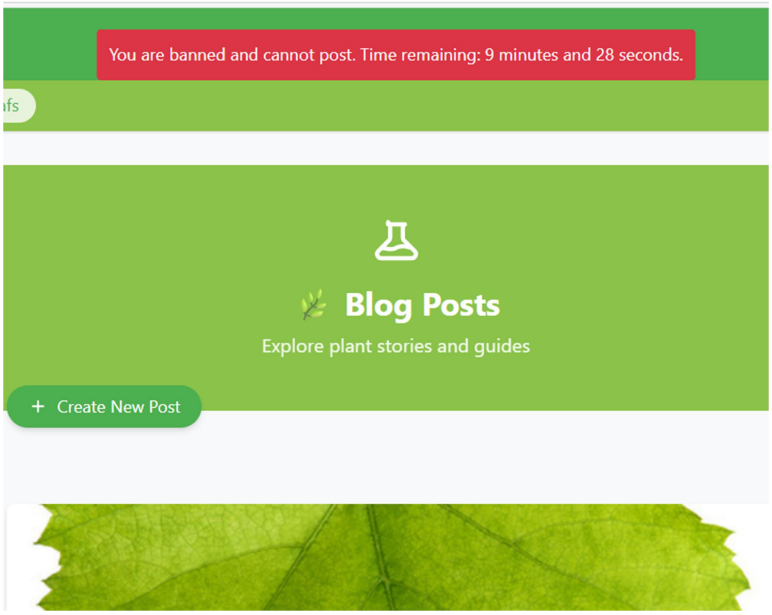

Leafer
jaudat00
Home
Upload Image
Blogs
See Reports
Trivia
Learn fun facts about plants!
Quiz
Test your plant knowledge!
Blogs
Read about plant care tips.
© 2025 Leafer — Open Source it


Edit Post
Title
w
Content
w
Upload New Image (Optional)
Choose File
No file chosen
Supported formats: JPG, PNG
Current Image


Change Password
Enter your old password, then choose a new one.
Old password
New password
New password confirmation
Update Password


Welcome to the Plant Blog
Explore tips, stories, and guides about plants and nature
+ Create New Post
View All Posts >
W
By jojoo
w
Read More >


Your Notifications
Mark all as read
You have been temporarily banned for 10 minutes due to repeated violations.
0 minutes ago
Your content 'Comment on 'qwer2'' has been reported.
0 minutes ago
Your content 'Comment on 'qwer2'' has been reported.
0 minutes ago


You are banned and cannot post. Time remaining: 9 minutes and 28 seconds.
Blog Posts
Explore plant stories and guides
+ Create New Post

References

Django documentation

Django documentation | Django documentation | Django

Kaggle data leaves data set (VPN needed)

leaves dataset | Kaggle

Tensorflow python documentation (VPN needed)

All symbols in TensorFlow 2 | TensorFlow v2.16.1

Index

Introduction and objective 2
General Intro 3
Key components 3
Audience 4
Development Phases 5
problem statment 5
Vision 5

Project Objectives 7
Technical innovations 8

System analysis and design 9
Intro 10
System Requirements 10
System functions 11
Data dictionary 12
ERD summary 13

Use case diagram 14
System design overview 15
Role base premissions 16

Implementation and overview 18
Start 20
Finish 53

Results 54
Key achievements 55

Prospective view 57
Community 59
Technically 59
Environmental impact 60
Strategic roadmap 60
Long term vision 61

User interfaces 62
Finish 66
References 67