



VIT<sup>®</sup>  
BHOPAL

# EPICS (DSN3099 - Engineering Projects in Community Service)

## REVIEW 2

GROUP 268



**VIT**<sup>®</sup>  
BHOPAL

# Bahaar.AI

**SUPERVISOR :**

**DR. BHUMIKA CHOKSI** (Assistant Professor)

**REVIEWER 1 :**



**DR. AMIT KUMAR SINGH** (Assistant Professor)

**REVIEWER 2 :**

**DR. SONJOY PAN** (Assistant Professor)



# GROUP MEMBERS

- 21BSA10113 Mayank Choubey
  - 21BSA10133 Vansh Thakur
  - 21BSA10122 Suraj Sasmal
  - 21BSA10105 Imran Wani
  - 21BSA10075 Anurag Prasad
  - 21BAI10463 Rajvardhan Singh
  - 21BCE11069 Pranav Nehe
  - 21BCE10983 Harshal Wakchaure
  - 21BCG10011 Jyoti Sharma
  - 21BCG10084 Kanishk Arvind Singh
- 
- 

# PROJECT OBJECTIVE

This project aims to create a system using Convolutional Neural Networks (CNNs) to identify and treat plant diseases quickly and accurately. By taking pictures of plant leaves, the system can analyze them statistically to detect diseases like those affecting leaves, fruits, and buds. Early identification helps prevent spread, saving crops and resources. Currently, identifying diseases manually is time-consuming and costly. Our system offers a faster, more efficient solution.

By combining image processing and leaf image classification, we train the CNN to recognize diseases, aiding farmers and professionals in timely diagnosis and treatment. This technology is crucial for the agricultural industry, where diseases can significantly impact crop production.



# SOFTWARE REQUIREMENTS

- **STREAMLIT:** Streamlit is a Python library designed to simplify the process of building interactive web applications. It allows developers to create data-driven applications quickly and easily using simple Python scripts.
- Streamlit provides a high-level API for building web interfaces, making it accessible to developers with varying levels of expertise. With Streamlit, developers can create dynamic dashboards, visualizations, and other web-based applications without needing to write HTML, CSS, or JavaScript code.
- It offers features such as widgets, layout options, and built-in components for common tasks like data visualization and user input handling. Streamlit's simplicity and flexibility make it a popular choice for prototyping, data exploration, and deploying machine learning models as web applications.



# Streamlit

# SOFTWARE REQUIREMENTS

- **TENSORFLOW-CPU:** TensorFlow is a powerful open-source machine learning framework developed by Google. It provides a comprehensive ecosystem of tools and libraries for building, training, and deploying machine learning models across a variety of platforms.
- The "tensorflow-cpu" requirement specifies that the project will use TensorFlow with CPU support, meaning that computations will be performed using the CPU (Central Processing Unit) rather than the GPU (Graphics Processing Unit). This can be advantageous in environments where GPU resources are limited or unavailable. While CPU-based computation may be slower compared to GPU-based computation for certain tasks, TensorFlow's CPU support allows developers to run machine learning models on a wider range of hardware configurations, making it more accessible for development and deployment in diverse environments.



TensorFlow

# SOFTWARE REQUIREMENTS

- PILLOW: Pillow is a Python Imaging Library (PIL) fork that provides extensive support for opening, manipulating, and saving various image file formats.
- It offers a wide range of image processing functionalities, including resizing, cropping, filtering, enhancing, and converting images.
- Pillow is widely used in Python applications for tasks such as preprocessing images before feeding them into machine learning models, generating thumbnails, creating visualizations, and performing basic image editing operations. With its user-friendly API and extensive documentation, Pillow simplifies the process of working with images in Python, making it an essential library for projects involving image processing, computer vision, and multimedia applications.



pillow

# SOFTWARE REQUIREMENTS



- **SQLALCHEMY:** SQLAlchemy is a popular SQL toolkit and Object-Relational Mapping (ORM) library for Python.
- It provides a high-level, Pythonic interface for interacting with relational databases, allowing developers to perform database operations using Python objects and methods.
- SQLAlchemy abstracts away the complexities of database management and SQL queries, making it easier to work with databases in Python applications. It supports a wide range of database systems, including SQLite, PostgreSQL, MySQL, and Oracle, making it a versatile choice for database-driven applications. SQLAlchemy's ORM capabilities enable developers to define database models as Python classes, with each class representing a table in the database.
- This allows for seamless integration between application code and database schemas, simplifying tasks such as data retrieval, manipulation, and persistence. With features such as query building, transactions, connection pooling, and schema migration, SQLAlchemy provides a comprehensive toolkit for building robust and scalable database-driven applications in Python.

SQLA







# LITERATURE REVIEW

- Plant disease detection using Convolutional Neural Networks (CNNs) has emerged as a promising approach in agricultural research. This methodology leverages deep learning techniques to automatically identify and classify diseases in plants based on image analysis.
  - Researchers have explored the application of CNNs in plant disease detection, demonstrating its effectiveness in various studies. These studies typically involve the development of CNN models trained on datasets of plant images, with the aim of accurately classifying diseases. By utilizing CNNs, researchers have achieved high levels of accuracy in disease identification, enabling timely intervention and treatment to mitigate crop losses.
  - The success of CNNs in plant disease detection lies in their ability to extract meaningful features from images and learn complex patterns associated with different diseases. This enables the models to distinguish between healthy and diseased plants with a high degree of accuracy, even in cases where the symptoms may be subtle or difficult to detect with the naked eye.
- 
- 



# LITERATURE REVIEW



- Additionally, researchers have emphasized the importance of preprocessing techniques and feature extraction methods in enhancing the performance of CNN models for plant disease detection. By carefully curating training datasets and optimizing model architectures, researchers have been able to improve the robustness and generalization capabilities of CNN-based disease detection systems.
  - Overall, the literature on plant disease detection using CNNs highlights the potential of this approach to revolutionize agricultural practices. By providing automated, efficient, and accurate solutions for disease identification, CNNs offer valuable tools for farmers and agricultural professionals to monitor and manage crop health effectively. However, ongoing research is needed to address challenges such as dataset diversity, model generalization, and real-world applicability, ensuring the continued advancement of CNN-based plant disease detection systems.
- 
- 



# LITERATURE REVIEW



J. Kolli, D. M. Vamsi, and V. M. Manikandan, "Plant Disease Detection using Convolutional Neural Network," presented at the 2021 IEEE Bombay Section Signature Conference (IBSSC), Gwalior, India, 2021, pp. 1-6, doi: 10.1109/IBSSC53889.2021.9673493.



- This paper presents a study on the application of Convolutional Neural Networks (CNNs) for automated plant disease detection. The authors developed a CNN model trained on a dataset of images containing healthy and diseased plant leaves.
  - The study likely involved collecting and preprocessing a dataset of plant images, including steps such as image acquisition, labeling, and normalization. The authors may have utilized techniques such as data augmentation and transfer learning to improve the model's performance.
  - Evaluation of the CNN model probably included metrics such as accuracy, precision, recall, and F1 score. The authors likely compared their model's performance against existing methods or benchmarks to demonstrate its effectiveness.
  - The paper likely discusses potential applications of the developed CNN model, such as real-time disease detection in agricultural fields, enabling farmers to take timely actions to prevent crop losses.
- 
- 




# LITERATURE REVIEW





G. Shrestha, Deepshikha, M. Das, and N. Dey, "Plant Disease Detection Using CNN," presented at the 2020 IEEE Applied Signal Processing Conference (ASPCON), Kolkata, India, 2020, pp. 109-113, doi: 10.1109/ASPCON49795.2020.9276722.

- This paper presents another approach to using CNNs for plant disease detection. The authors may have focused on specific aspects such as feature extraction, model architecture, or dataset composition to improve the accuracy and efficiency of their CNN model.
  - Similar to the first paper, the evaluation of the CNN model's performance likely involved rigorous testing and comparison with other methods or benchmarks.
  - The paper may also discuss practical considerations for deploying the CNN-based disease detection system, such as computational requirements, scalability, and usability in real-world agricultural settings.
  - Additionally, the authors might have explored future directions for research, including potential enhancements to the CNN model, integration with other technologies, or application in related domains such as precision agriculture or crop management.
- 
- 





# • Factors Affecting the Feasibility of Disease Detection Systems in Agriculture

- **Resource Cost & Access:** Affordable hardware, software, and reliable internet are crucial, especially for resource-limited areas.
  - **Power & Connectivity:** Consistent power and internet access are essential for system functionality, particularly in rural settings.
  - **User Skills & Adoption:** Farmer and extension worker literacy, digital skills, and openness to new technologies are vital for successful implementation.
  - **Crop & Disease Complexity:** System needs to handle diverse crops, diseases, and pests with varying symptoms across different environments.
  - **Data Quality & Ethics:** High-quality, diverse data with accurate labels is needed for training reliable machine learning models, while adhering to ethical and legal considerations.
  - **Model Accuracy & Explainability:** The system needs to deliver accurate and interpretable results for effective disease management and user trust.
  - **Integration & Compatibility:** Seamless integration with existing agricultural practices ensures smooth adoption and complements existing workflows.
- 
- 





# MODULE DESCRIPTION

- Image Acquisition and Preprocessing
  - Disease Classification and Reporting
  - CNN Model Development and Training
  - System Integration and Deployment
- 
- 



# IMAGE ACQUISITION AND PREPROCESSING





- Streamlit is utilized in this module to create a user-friendly interface for image acquisition. Users can upload images directly through the Streamlit application, which serves as the image source.
  - TensorFlow-CPU is employed for image preprocessing tasks such as resizing and normalization. These operations ensure that the images are prepared appropriately before being fed into the CNN model.
  - Pillow library is used for image manipulation during preprocessing. It provides functionalities for tasks like resizing, cropping, and adjusting image quality, ensuring that the images are standardized for analysis.
  - Data augmentation techniques, if implemented, can be integrated using TensorFlow-CPU to augment the dataset and improve the robustness of the CNN model
- 
- 



# DISEASE CLASSIFICATION AND REPORTING

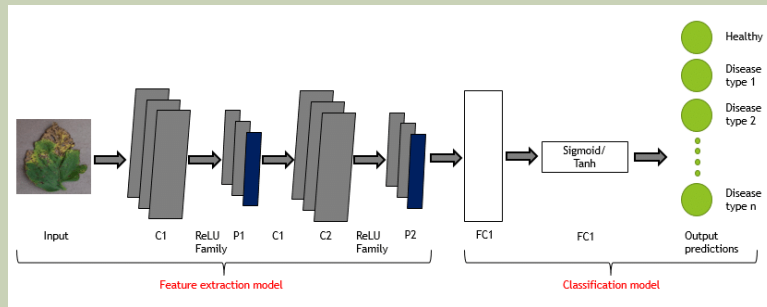


- Streamlit is leveraged again in this module to display the classification results and report them to the user in a clear and understandable format through the web application.
  - TensorFlow-CPU is utilized to execute the trained CNN model for disease classification. The model analyzes the preprocessed images and makes predictions about the presence of diseases.
  - Pillow may be used in this module for any additional image processing tasks that are required before or after classification, such as generating visualizations or enhancing image quality.
  - SQLAlchemy can be integrated into the reporting system for storing and managing classification results in a database, enabling users to access historical data and track disease patterns over time.
- 
- 



# CNN MODEL DEVELOPMENT AND TRAINING



- TensorFlow-CPU is the primary library used for developing and training the CNN model. It provides extensive support for building and training deep learning models, including CNNs, on CPU hardware.
- Pillow may be used during model development for visualizing training data, inspecting images, and debugging preprocessing pipelines.
- SQLAlchemy is not directly involved in this module but may be used to manage datasets, experiment logs, or model checkpoints in a database if required for tracking training progress and results.



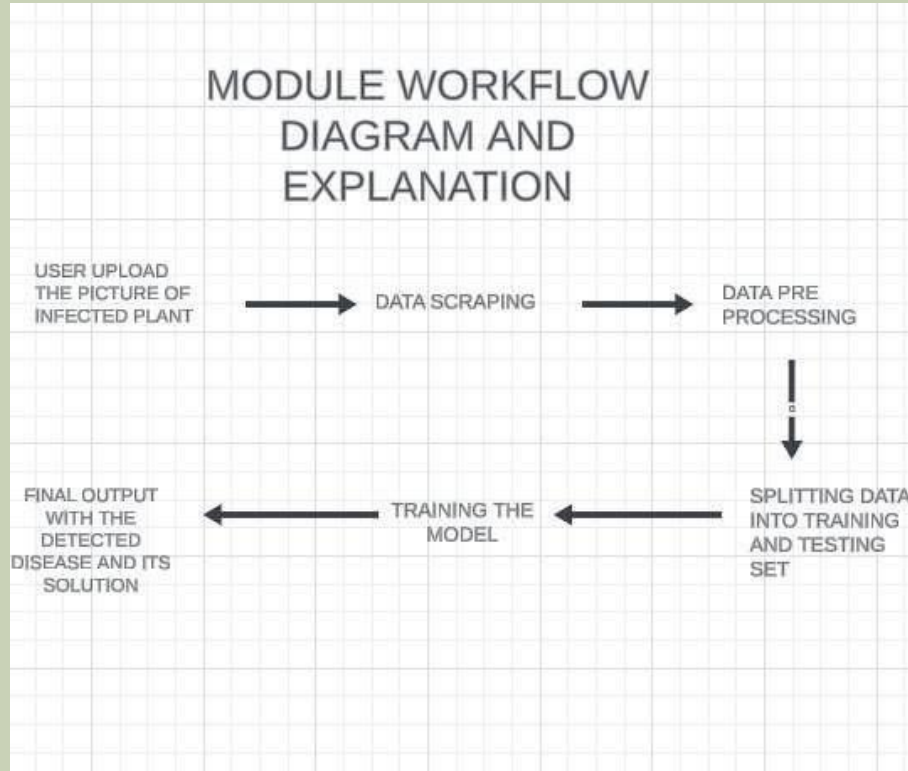


# SYSTEM INTEGRATION AND DEPLOYMENT



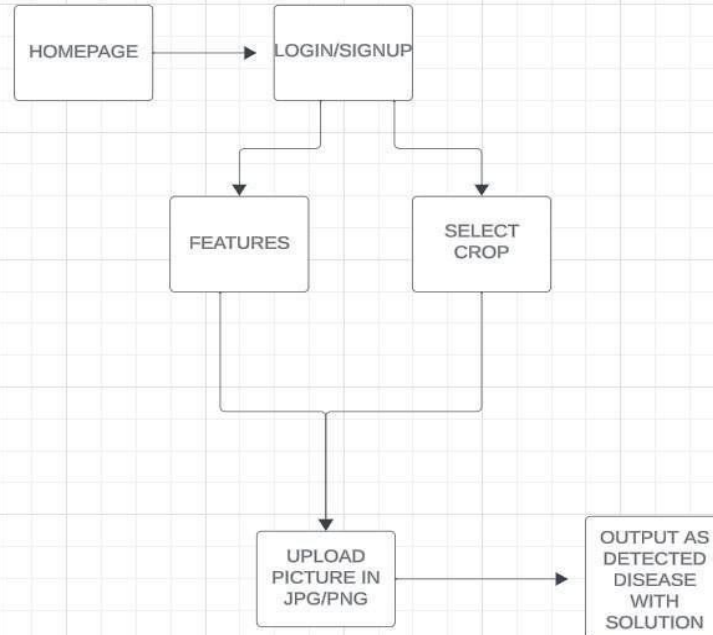
- Streamlit is crucial for deployment in this module, serving as the platform for hosting the user interface and deploying the complete plant disease detection system as a web application.
  - TensorFlow-CPU remains essential for executing the trained CNN model within the deployed system, ensuring that disease detection can be performed in real-time.
  - Pillow may be used during deployment for any on-the-fly image processing tasks required to handle user-uploaded images within the Streamlit application.
  - SQLAlchemy is not directly involved in deployment but may be utilized for managing any database interactions or backend operations required by the deployed system, such as user authentication, data storage, or logging.
- 
- 

# MODULE WORKFLOW

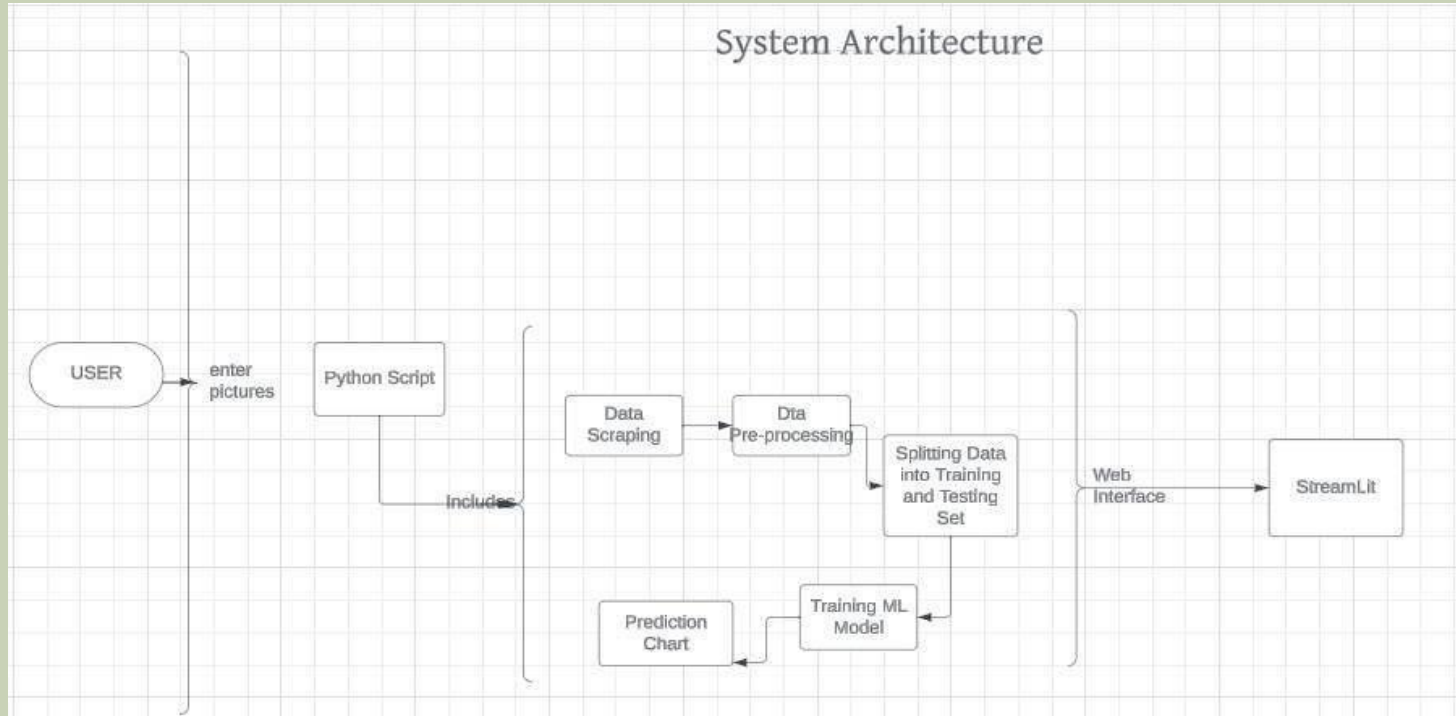


# TASKFLOW

## MODULE TASKFLOW DIAGRAM





# ARCHITECTURE





# OUTCOME & CONCLUSION



- **Improved Disease Diagnosis:** Deep learning boosts accuracy in identifying plant diseases from images.
  - **Smart Disease Detection Proposal:** Combining computer vision and machine learning for efficient disease spotting in agriculture.
  - **AI System for Tomato Disease Detection:** An AI system identifies tomato leaf diseases using image-based machine learning.
  - **K-means Clustering for Disease Detection:** Using K-means clustering alongside environmental monitoring for spotting disease patterns.
  - **Real-time Crop Disease Prediction:** Introducing deep learning for instant detection of crop diseases.
  - **Enhanced Disease Identification:** Combining machine learning with thermal and visible light images for better accuracy in disease spotting.
- 
- 

# TIMELINE

## PHASE 1



- Start data preprocessing and exploratory analysis.
- Set up infrastructure for model development.

## PHASE 2



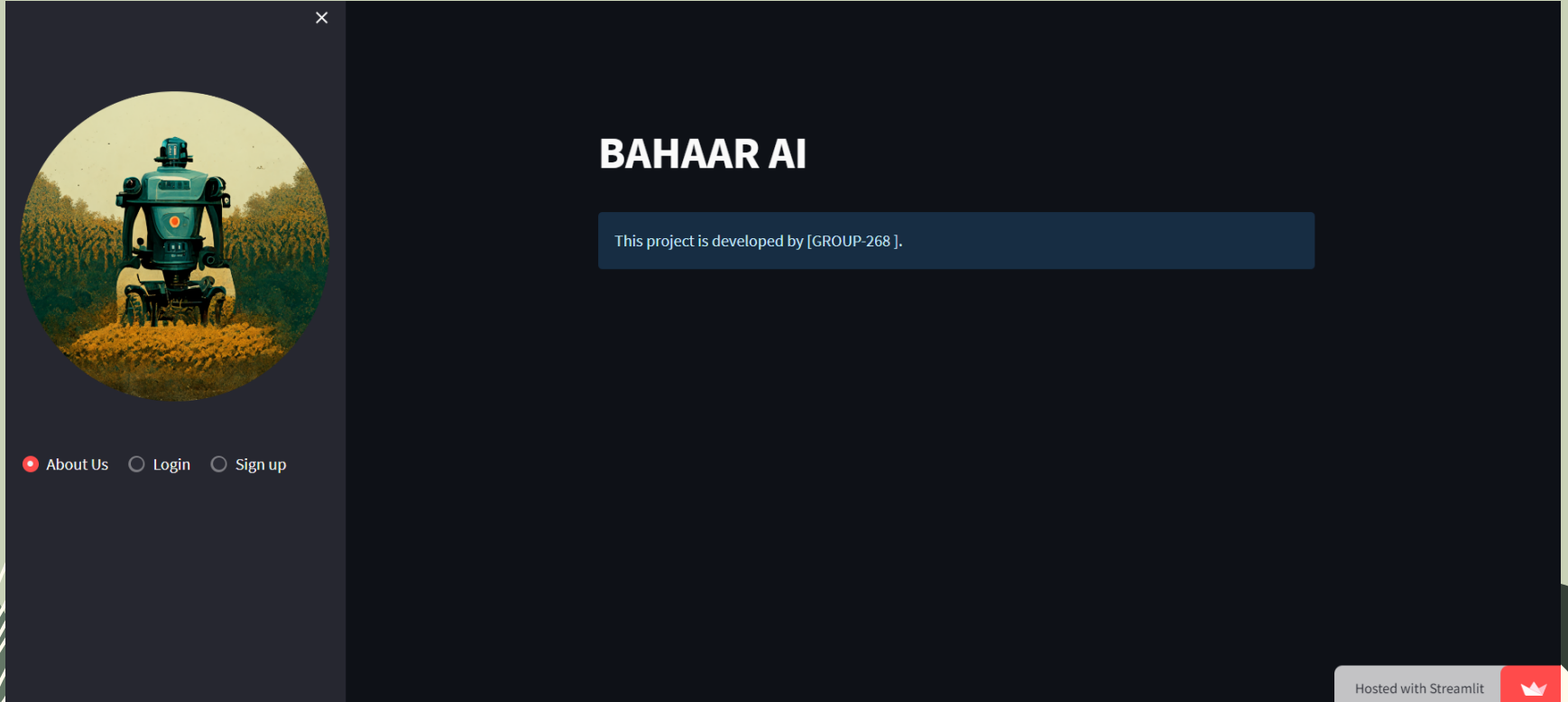
- Develop and train the CNN-based deep learning model.
- Test the model and create a mobile application.
- Pilot rollout with local agricultural communities for feedback.

## PHASE 3



- Deploy the refined system to additional regions.
- Provide training and support for users.
- Ensure smooth adoption in agricultural communities.

# PILOT RUN (SCREENSHOT)







**THANK YOU!**