Chapter 1
Introduction

Introduction: -

The rapid growth of Artificial Intelligence (AI) and Machine Learning (ML) has significantly transformed various industries, particularly in the field of computer vision. Computer vision enables machines to interpret and analyze visual data in a manner similar to human perception. Among the many applications of computer vision, image classification has become one of the most widely researched and implemented areas. Image classification involves training computational models to automatically recognize and categorize objects within digital images. One practical and challenging application of image classification is dog breed identification.

Dog breed identification is a complex problem due to the large number of dog breeds and the subtle visual differences among them. Globally, there are more than 300 recognized dog breeds, each with unique physical characteristics such as size, fur texture, ear shape, color patterns, and facial structure. However, many breeds share highly similar visual traits, making manual identification difficult even for experienced breeders or veterinarians. Variations in lighting conditions, camera angles, background environments, and occlusions further increase the complexity of accurately classifying dog breeds from images. Therefore, developing an automated and intelligent system for dog breed recognition becomes an important and valuable task.

In conclusion, the Dog Breed Identification using Transfer Learning project highlights the effectiveness of modern deep learning techniques in solving complex image classification problems. By leveraging pre-trained models and fine-tuning them for a specific task, it is possible to achieve high accuracy while reducing training time and computational requirements.

The project not only demonstrates technical proficiency in machine learning and web deployment but also provides a scalable solution that can be extended to other animal classification or object recognition systems.

DATE	28-02-2026
TEAM ID	LTVIP2026TMIDS90703
PROJECT NAME	Dog breed identification using transfer learning
MAXIMUM MARKS	2 MARKS

Chapter – 2
Indentation Phase

2.1 problem statement

Dog breed identification is a challenging task due to the presence of a large number of dog breeds with subtle visual differences. Many breeds share similar physical characteristics such as fur colour, body structure, facial features, and size, making it difficult to accurately distinguish between them using traditional image processing techniques. Manual identification often requires expert knowledge and experience, and even then, it may lead to incorrect classification, especially when images are captured under varying lighting conditions, different angles, or complex backgrounds.

Traditional machine learning approaches rely heavily on handcrafted feature extraction methods such as colour histograms, edge detection, and texture analysis. These techniques require domain expertise and often fail to generalize effectively when dealing with large-scale image datasets. Furthermore, training deep neural networks from scratch demands a vast amount of labeled data and significant computational resources, which may not always be feasible in real-world applications.

Another major challenge lies in handling high intra-class similarity and inter-class variation. Intra-class similarity refers to different breeds appearing visually similar to one another, while inter-class variation refers to the same breed appearing different due to pose, age, grooming style, or lighting conditions. These variations reduce classification accuracy and increase model complexity.

By leveraging pre-trained deep learning models and fine-tuning them for dog breed classification, the project aims to provide a reliable automated solution capable of accurately predicting dog breeds from input images.

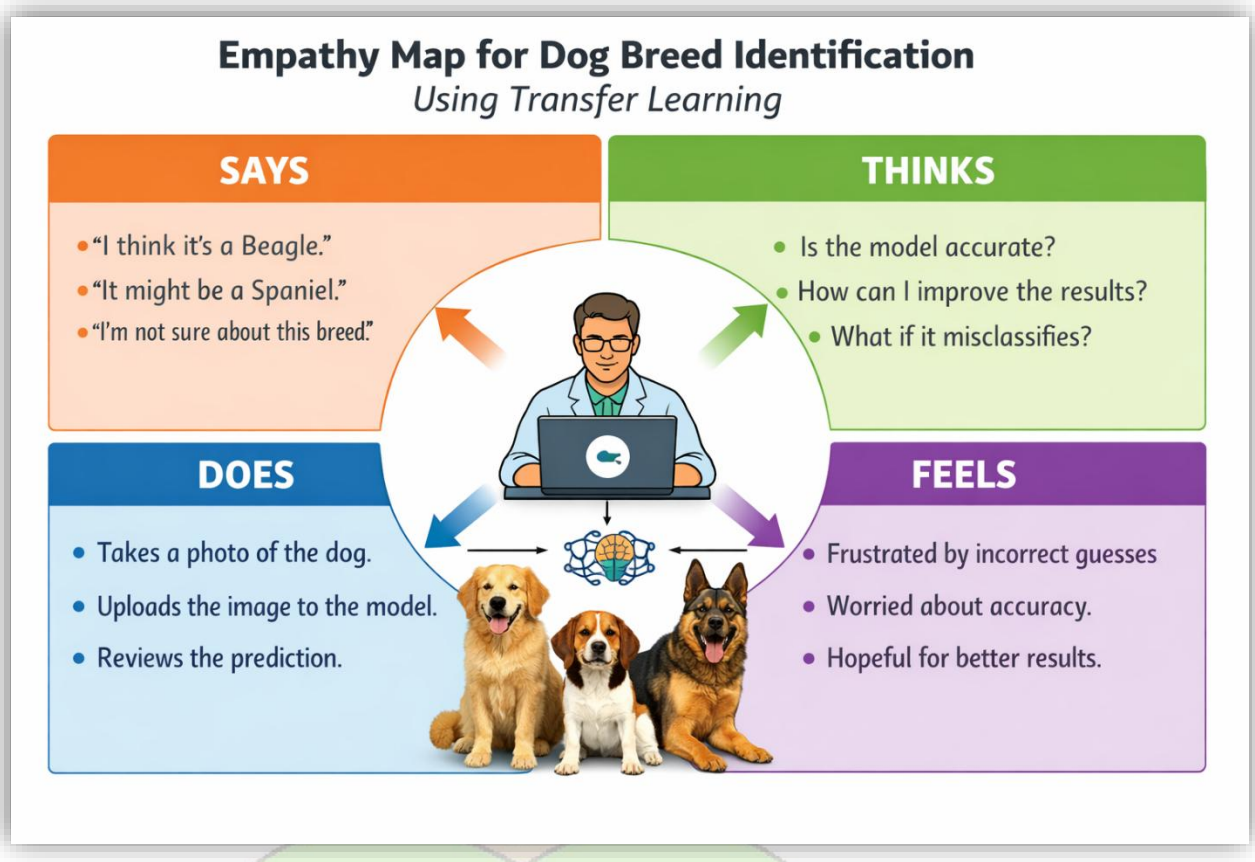
DATE	28-02-2026
TEAM ID	LTVIP2026TMIDS90703
PROJECT NAME	Dog breed identification using transfer learning
MAXIMUM MARKS	4 MARKS

2.2 Empathy map canvas

In the ideation phase of the Dog Breed Identification using Transfer Learning project, an empathy map was developed to better understand the needs, behaviors, emotions, and challenges of potential users. The primary target users include pet owners, veterinarians, animal shelter staff, dog breeders, and general individuals who wish to identify dog breeds accurately. Understanding user perspectives helps in designing a system that is practical, user-friendly, and capable of addressing real-world problems effectively.

From the users' perspective, many individuals express uncertainty when trying to identify a dog's breed. They often say that it is difficult to determine the exact breed because many dogs share similar physical characteristics. Some users mention that the dog may appear to be a mixed breed, making identification even more complicated. In cases where breed information is required for medical, adoption, or registration purposes, users emphasize the importance of having accurate identification. They also express a desire for a simple and quick solution that does not require expert consultation.

Overall, the empathy map analysis clearly indicates that users require an accurate, reliable, and user-friendly dog breed identification system. By leveraging transfer learning and deep learning techniques, the proposed solution aims to address these user needs effectively.



Dog breed identification using transfer learning

DATE	28-02-2026
TEAM ID	LTVIP2026TMIDS90703
PROJECT NAME	Dog breed identification using transfer learning
MAXIMUM MARKS	4 MARKS

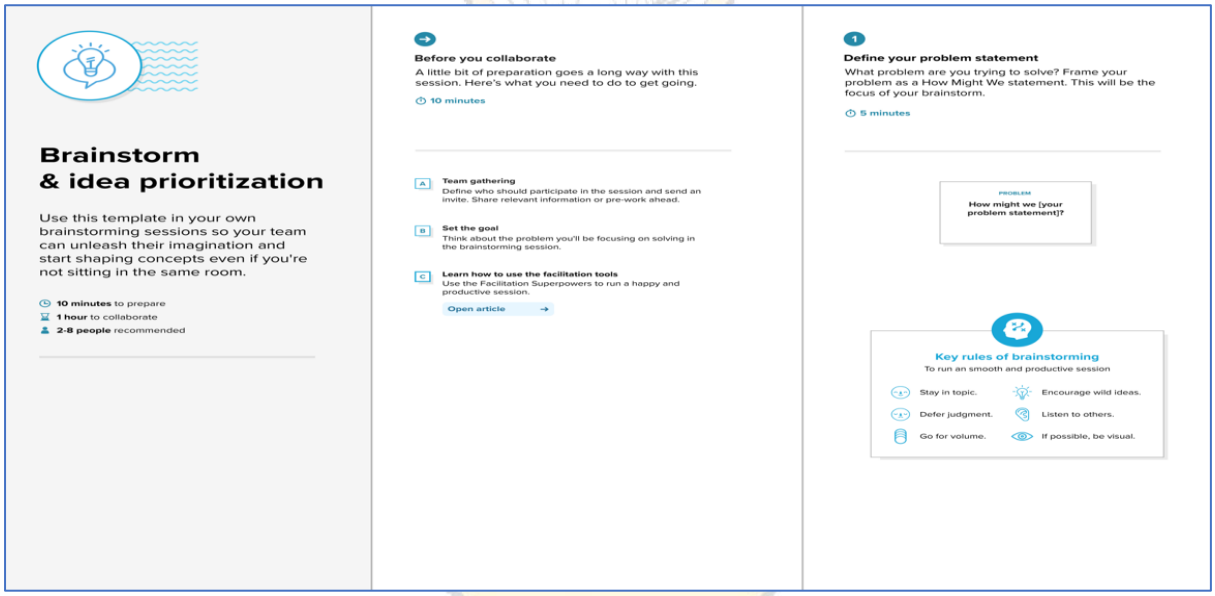
2.3 – Brainstorming

Brainstorming provides a free and open environment that encourages everyone within a team to participate in the creative thinking process that leads to problem solving. Prioritizing volume over value, out-of-the-box ideas are welcome and built upon, and all participants are encouraged to collaborate, helping each other develop a rich amount of creative solutions.

Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

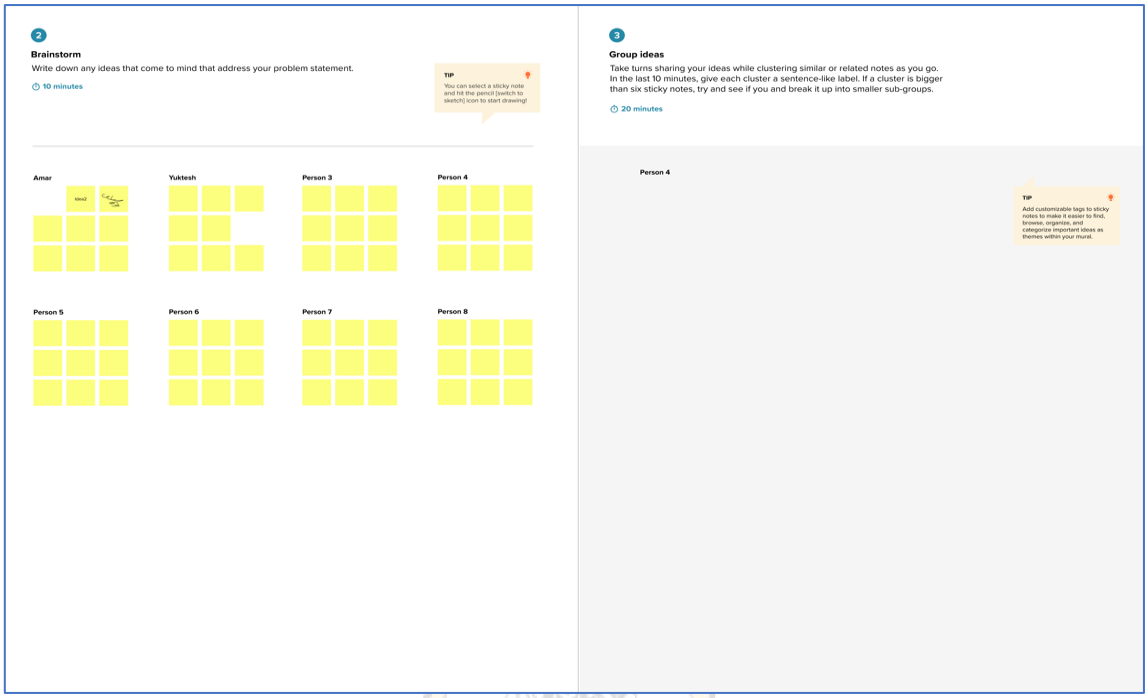
Reference: https://www.mural.co/templates/brainstorm-and-idea-prioritization

Step-1: Team Gathering, Collaboration and Select the Problem Statement



Step-2: Brainstorm, Idea Listing and Grouping

Dog breed identification using transfer learning



Step-3: Idea Prioritization

DATE	28-02-2026
TEAM ID	LTVIP2026TMIDS90703
PROJECT NAME	Dog breed identification using transfer learning
MAXIMUM MARKS	2 MARKS

Chapter 3
Requirement Analysis

3.1 Custom journey map

In the Requirement Analysis phase of the Dog Breed Identification using Transfer Learning project, the Customer Journey Map was developed to understand how users interact with the system from the initial stage of recognizing a need to the final outcome of obtaining breed identification results. This journey helps identify user expectations, potential painpoints, and opportunities for system improvement.

The journey begins with the awareness stage, where the user realizes the need to identify a dog's breed. This situation may arise when a pet owner adopts a new dog, when a veterinarian requires breed information for medical assessment, or when an individual encounters a stray dog and wishes to know its breed. At this stage, the user may feel curious or uncertain. They may attempt to manually compare images online or seek advice from experts, but often experience confusion due to the similarity between breeds. This creates the motivation to search for a technological solution.

Overall, the Customer Journey Map reveals that users prioritize simplicity, speed, and accuracy throughout their interaction with the dog breed identification system. From awareness to post-experience, each stage highlights functional and non-functional requirements such as intuitive designrobust image preprocessing, efficient model performance, accurate classification, and user satisfaction. By analyzing the customer journey, the project ensures that the developed system

DATE	28-02-2026
TEAM ID	LTVIP2026TMIDS90703
PROJECT NAME	Dog breed identification using transfer learning
MAXIMUM MARKS	2 MARKS

3.2 - Solution Requirement

In the Requirement Analysis phase of the Dog Breed Identification using Transfer Learning project, solution requirements define the technical and functional expectations that the proposed system must fulfill to address user needs effectively. These requirements ensure that the system not only solves the identified problem but also operates efficiently, accurately, and reliably in real-world scenarios.

The primary solution requirement is the development of an intelligent image classification system capable of accurately identifying dog breeds from uploaded images. The system must utilize deep learning techniques, specifically transfer learning, to leverage pre-trained Convolutional Neural Network (CNN) models. By using pre-trained models trained on large datasets such as ImageNet, the system can reduce training time while maintaining high accuracy. The solution must allow fine-tuning of the final layers to adapt the model specifically to the selected dog breed dataset.

Another essential requirement is image preprocessing capability. The system must automatically resize images to a standard input size compatible with the selected pre-trained model architecture. It should normalize pixel values and, if necessary, apply data augmentation techniques during training to improve generalization and reduce overfitting. The preprocessing pipeline must handle different image qualities, lighting conditions, and orientations to ensure consistent performance.

DATE	28-02-2026
TEAM ID	LTVIP2026TMIDS90703
PROJECT NAME	Dog breed identification using transfer learning
MAXIMUM MARKS	3 MARKS

3.3 - Data Flow Diagram

In the Requirement Analysis phase, the Data Flow Diagram (DFD) is used to represent the logical flow of data within the Dog Breed Identification system. The DFD illustrates how input data moves through different processes, how it is transformed, and how the final output is generated. It provides a clear understanding of system functionality without focusing on implementation details.

At a high level (Level 0 – Context Diagram), the system consists of one primary external entity, which is the user. The user interacts with the system by uploading an image of a dog. The system processes the image using a trained deep learning model and returns the predicted dog breed as the output. The main data flow begins with the image input from the user and ends with the breed prediction result displayed on the user interface.

In Level 1 DFD, the system is divided into multiple processes to represent internal operations. The first process is Image Upload and Validation. In this stage, the user uploads an image through the web interface. The system checks whether the file format is valid and ensures that the image is suitable for processing. If validation fails, an error message is returned; otherwise, the image proceeds to the next stage.

DATE	28-02-2026
TEAM ID	LTVIP2026TMIDS90703
PROJECT NAME	Dog breed identification using transfer learning
MAXIMUM MARKS	3 MARKS

3.4 - Technology Stack

In the Requirement Analysis phase, defining the technology stack is essential to ensure that the proposed solution is technically feasible, scalable, and efficient. The technology stack for the Dog Breed Identification using Transfer Learning project consists of programming languages, frameworks, libraries, development tools, and deployment components that collectively support model development, training, and web integration.

The primary programming language used in this project is Python. Python is widely preferred in machine learning and deep learning applications due to its simplicity, readability, and extensive ecosystem of libraries. Its rich set of scientific and AI-related packages enables efficient implementation of complex algorithms with minimal code complexity.

For deep learning model development, TensorFlow is utilized as the core framework. TensorFlow provides a powerful and flexible platform for building and training neural networks. It supports large-scale numerical computations and GPU acceleration, which significantly improves model training performance. On top of TensorFlow, Keras is used as a high-level API to simplify the process of designing and training Convolutional Neural Networks (CNNs). Keras enables easy implementation .

DATE	28-02-2026
TEAM ID	LTVIP2026TMIDS90703
PROJECT NAME	Dog breed identification using transfer learning
MAXIMUM MARKS	5 MARKS

Chapter 4
Project Design

4.1 - Problem Solution Fit

In the Project Design phase, establishing a clear Problem–Solution Fit is essential to ensure that the proposed system effectively addresses the identified user needs and challenges. The Dog Breed Identification using Transfer Learning project was designed after carefully analyzing the difficulties users face in accurately identifying dog breeds and aligning those challenges with an appropriate technological solution.

The core problem identified is the difficulty in accurately distinguishing between a large number of dog breeds that often share similar physical characteristics. Manual identification methods, such as visual comparison through online searches or consulting experts, are time-consuming, inconsistent, and dependent on human expertise.

Users frequently experience confusion due to subtle variations in fur patterns, facial structure, size, and color. Additionally, variations in image quality, lighting conditions, and camera angles further complicate accurate breed recognition. Traditional machine learning approaches that rely on handcrafted feature extraction methods lack robustness and fail to generalize effectively in diverse real-world conditions.

DATE	28-02-2026
TEAM ID	LTVIP2026TMIDS90703
PROJECT NAME	Dog breed identification using transfer learning
MAXIMUM MARKS	5 MARKS

4.2 - Proposed Solution

The proposed solution for the Dog Breed Identification using Transfer Learning project is to develop an intelligent, automated image classification system capable of accurately identifying dog breeds from digital images. The system is designed using deep learning techniques, specifically transfer learning, to ensure high accuracy, reduced training time, and efficient deployment.

The solution is built upon a pre-trained Convolutional Neural Network (CNN) architecture that has been previously trained on large-scale image datasets such as ImageNet. These pre-trained models have already learned general visual features such as edges, textures, shapes, and object patterns. Instead of training a neural network from scratch, the proposed system reuses these learned features and fine-tunes the final layers to classify specific dog breeds. Architectures developed by the Visual Geometry Group, such as VGG models, are suitable choices due to their strong feature extraction capabilities.

The system workflow begins with the user uploading a dog image through a web-based interface. The uploaded image undergoes preprocessing steps, including resizing to a standard input size, normalization of pixel values, and formatting compatible with the CNN model. These preprocessing steps ensure that the image data is consistent with the model's training requirements and improves

The proposed solution also emphasizes scalability and maintainability. Additional dog breeds can be incorporated into the dataset, and the model can be retrained or fine-tuned further to improve performance. The architecture supports enhancements such as displaying confidence scores, providing breed information, or extending the system to mobile platforms.

In conclusion, the proposed solution leverages transfer learning and deep learning technologies to build a robust, accurate, and efficient dog breed identification system. By combining advanced image classification techniques with a user-friendly web interface, the system effectively addresses the limitations of manual breed identification methods and provides a reliable AI-driven solution for real-world applications.

DATE	28-02-2026
TEAM ID	LTVIP2026TMIDS90703
PROJECT NAME	Dog breed identification using transfer learning
MAXIMUM MARKS	5 MARKS

4.3 - Solution Architecture

The Solution Architecture of the Dog Breed Identification using Transfer Learning project defines the overall structural design of the system, describing how different components interact to deliver accurate breed predictions. The architecture follows a modular and layered approach, ensuring scalability, maintainability, and efficient performance.

At a high level, the system architecture consists of four primary layers: the Presentation Layer, Application Layer, Model Layer, and Data Layer. Each layer performs a specific function while interacting seamlessly with the others to ensure smooth system operation.

The Presentation Layer represents the user interface of the system. This layer is developed using the Flask web framework and is responsible for handling user interactions. Through a web page, users can upload an image of a dog for breed identification. The interface is designed to be simple and intuitive, allowing users to interact with the system without technical knowledge. Once the image is uploaded, it is sent to the backend server for further processing.

In conclusion, the solution architecture is designed to integrate deep learning capabilities with a web-based application framework effectively. It ensures high accuracy through transfer learning, smooth data handling through preprocessing modules, and user accessibility through a web interface.

DATE	28-02-2026
TEAM ID	LTVIP2026TMIDS90709
PROJECT NAME	Dog breed identification using transfer learning
MAXIMUM MARKS	1 MARKS

Chapter – 5

Project Planning & Scheduling

5.1 - Project Milestones & Tasks

1 Data Collection

Objective: Gather high-quality, relevant data required for the project.

Tasks:

- Identify reliable data sources (databases, APIs, web scraping, surveys, etc.)
- Collect structured and unstructured data
- Ensure data relevance and completeness
- Store data in a centralized database or storage system
- Maintain data documentation for reference

Deliverables:

- Raw dataset
- Data source documentation
- Data storage setup

---

2 Data Pre-Processing

Objective: Clean and prepare the data for model training and analysis.

Tasks:

- Remove duplicates and handle missing values
- Handle outliers and inconsistent data
- Data normalization or scaling
- Feature engineering and selection
- Encode categorical variables
- Split dataset into training and testing sets

Deliverables:

- Cleaned dataset
- Feature-engineered dataset
- Preprocessing scripts

---

3 Model Building

Objective: Develop and train a machine learning model.

Tasks:

- Select appropriate algorithm(s)
- Train model using training dataset
- Hyperparameter tuning
- Model validation and evaluation
- Compare performance metrics
- Finalize best-performing model

Deliverables:

- Trained model
- Evaluation report (Accuracy, Precision, Recall, F1-score, etc.)
- Saved model file

---

4 API Integration

Objective: Expose the trained model via an API for external access.

Tasks:

- Develop REST API endpoints
- Integrate model with backend framework (Flask / FastAPI / Django)
- Implement request validation
- Add error handling
- Test API responses
- Secure API (authentication if required)

Deliverables:

- Functional API
- API documentation (Swagger/Postman collection)
- Deployment-ready backend

---

5 Web Integration

Objective: Integrate API into a web-based user interface.

Tasks:

- Design user-friendly frontend interface
- Connect frontend to backend API
- Display model predictions dynamically
- Implement input validation
- Perform end-to-end testing
- Deploy web application

Deliverables:

- Fully functional web application
- User interface design
- Live deployment link (if applicable)

Dog breed identification using transfer learning

DATE	28-02-2026
TEAM ID	LTVIP2026TMIDS90703
PROJECT NAME	Dog breed identification using transfer learning
MAXIMUM MARKS	2 MARKS

5.2 - Sprint Delivery Plan

- **Phase 1: Live Sessions (Week 1–6)**

Objective: Build strong foundational knowledge and prepare interns for real-time project development.

Sprint 1 (Week 1–2): Fundamentals & Tools

Focus Areas:

- Introduction to Internship Program
- Programming Fundamentals (Python / Relevant Tech Stack)
- Git & GitHub
- Development Environment Setup
- Basics of Databases

Deliverables:

- Setup development environment
- GitHub repository creation
- Mini practice assignments

Sprint 2 (Week 3–4): Data & Backend Foundations

Focus Areas:

- Data Handling (Pandas / Data Structures)
- Data Cleaning Techniques
- Introduction to APIs
- Backend Basics (Flask / FastAPI)
- SQL & Database Integration

Deliverables:

- Data preprocessing assignment
- Basic API development task
- Database connectivity demo

🟦 **Sprint 3 (Week 5–6): Machine Learning & Deployment Basics**

Focus Areas:

- Machine Learning Fundamentals
- Model Training & Evaluation
- REST API Integration with Model
- Introduction to Web Integration
- Deployment Overview

Deliverables:

- Simple ML Model
- Model evaluation report
- API with working prediction endpoint

🔹 **Phase 2: Project Work (Week 7–15)**

Objective: Apply learned skills to build a complete end-to-end project.

🟩 **Sprint 4 (Week 7–8): Project Planning & Data Collection**

Activities:

- Finalize project topic
- Define problem statement
- Collect dataset
- Perform initial data analysis (EDA)

Deliverables:

- Project proposal document
- Dataset documentation
- EDA report

🟩 Sprint 5 (Week 9–10): Data Preprocessing & Feature Engineering

Activities:

- Clean dataset
- Handle missing values & outliers
- Feature engineering
- Data transformation
- Train-test split

Deliverables:

- Cleaned dataset
- Preprocessing pipeline
- Feature documentation

🟩 Sprint 6 (Week 11–12): Model Development & Optimization

Activities:

- Train multiple models
- Hyperparameter tuning
- Model comparison
- Performance evaluation

Deliverables:

- Best performing model
- Evaluation metrics report
- Saved model artifact

🟩 Sprint 7 (Week 13–14): API Development & Integration

Activities:

- Develop REST API
- Integrate trained model
- Implement validation & error handling
- Test API endpoints

Deliverables:

- Functional API
- API documentation
- Backend deployment

🟩 Sprint 8 (Week 15): Web Integration & Final Deployment

Activities:

- Develop frontend interface
- Connect frontend with API
- End-to-end testing
- Deployment & final presentation

Deliverables:

- Fully functional web application
- Deployment link
- Final project presentation
- Internship completion report

📊 Final Outcome

By the end of 15 weeks, interns will have:

- Strong technical foundation
- Real-time project experience
- A complete end-to-end deployed project
- Industry-ready portfolio project

DATE	28-02-2026
TEAM ID	LTVIP2026TMIDS90703
PROJECT NAME	Dog breed identification using transfer learning
MAXIMUM MARKS	2 MARKS

5.3 - Project Progress Tracking

1. Zoho Cliq Workspace Structure

- **Channels Setup**

Create the following channels for organized communication:

📌 **1. #announcements**

- Official updates
- Sprint start/end notifications
- Deadlines & evaluation updates
- Meeting schedules

📌 **2. #project-discussion**

- Technical queries
- Implementation discussions
- Code review discussions
- Issue troubleshooting

📌 **3. #daily-updates**

- Daily progress reports
- Blockers
- Completed tasks

📌 **4. #resources**

- Shared datasets
- Documentation links
- Recorded session links
- API documentation

📌 5. #team-specific-channels (if multiple teams)

Example:

- #team-alpha
- #team-beta

🗓️ 2. Sprint-Based Tracking Method

Each sprint will follow a structured reporting cycle.

🟢 Daily Progress Update Format (Posted in #daily-updates)

Every intern must post:

Format:

Date:

Sprint:

Tasks Completed:

Tasks In Progress:

Blockers (if any):

Plan for Tomorrow:

🔵 Weekly Sprint Review (Every Weekend)

Mentor will post:

- Sprint Goals
- Completed Milestones
- Pending Tasks
- Risk Areas
- Next Week Targets

📋 3. Task Tracking System

Option 1: Zoho Cliq Tasks Feature

Use built-in task management in Zoho Cliq:

- Assign tasks to interns
- Set deadlines
- Track completion status
- Add task priority (High / Medium / Low)

Task Status Workflow:

- 🟡 **To Do**
- 🔵 **In Progress**
- 🟢 **Completed**
- 🔴 **Blocked**

Option 2: Zoho Projects Integration (Optional)

For advanced tracking, integrate with:

- Zoho Projects

Use:

- Kanban Board
- Gantt Chart
- Milestone tracking
- Automated reminders

📊 4. Sprint Progress Monitoring Dashboard

Track the following metrics weekly:

- % Tasks Completed
- API Development Progress
- Model Accuracy Improvement
- Deployment Readiness

- Bug Count
- Attendance in Live Sessions

Mentor shares a weekly progress summary in:
👉 **#announcements channel**

📈 **5. Milestone Tracking Structure**

Milestone	Week	Status	Owner	Remarks
Data Collection	**Week 7–8**	☐	**Team**	
Data Preprocessing	**Week 9–10**	☐	**Team**	
Model Building	**Week 11–12**	☐	**Team**	
API Integration	**Week 13–14**	☐	**Team**	
Web Integration	**Week 15**	☐	**Team**	

🚨 **6. Escalation Process**

If blocker > 24 hours:

- **Post in #project-discussion**
- **Tag mentor**
- **If unresolved → Schedule quick call via Zoho Cliq**
- **Update resolution summary in channel**

🏆 **7. Performance Evaluation Criteria**

Evaluation will be based on:

- Daily update consistency
- Sprint milestone completion
- Code quality
- Participation in discussions
- Final project delivery
- Timely submissions

Weekly Workflow Summary

Day	Activity
Monday	Sprint planning post
Tue–Thu	Development & daily updates
Friday	Progress review
Saturday	Sprint demo
Sunday	Feedback & planning

Final Outcome

Using Zoho Cliq ensures:

- Clear communication
- Accountability
- Structured sprint tracking
- Real-time problem resolution
- Professional workflow management

Dog breed identification using transfer learning

DATE	28-02-2026
TEAM ID	LTVIP2026TMIDS90703
PROJECT NAME	Dog breed identification using transfer learning
MAXIMUM MARKS	1 MARKS

5.4 - Team Management Tools for Agile Planning

🧩 What is Jira?

Jira is an Agile project management and issue-tracking tool developed by Atlassian. It supports Scrum and Kanban methodologies, helping teams plan, track, and release software efficiently.

🏗️ Jira Project Structure for Internship

1️⃣ Project Creation

Create a project with:

- Project Name: Internship Capstone Project
- Template: Scrum (Recommended)
- Project Type: Software Development

📌 Issue Types Configuration

Define standard issue types:

- 🟢 Epic – Major project phases
- 🔵 Story – Feature or functionality
- 🟡 Task – Smaller implementation steps
- 🔴 Bug – Errors or defects
- ⚪ Sub-task – Breakdown of tasks

🗂️ Suggested Epics (Based on Your Milestones)

Epic	Description
Data Collection	**Dataset gathering & validation**

Data Preprocessing	Cleaning & feature engineering
Model Development	ML training & evaluation
API Integration	Backend & model API
Web Integration	Frontend & deployment

Sprint Planning Structure

- **Sprint Duration**
  - 2 Weeks per Sprint
  - Total: 4–5 Sprints (Project Phase)

- **Sprint Workflow**
  1. Create Sprint in Backlog
  2. Add Stories/Tasks to Sprint
  3. Assign Tasks to Interns
  4. Set Priority (High/Medium/Low)
  5. Estimate Story Points
  6. Start Sprint

Workflow Status Configuration

Customize workflow:

- To Do
- In Progress
- In Review
- Done
- Blocked

This ensures transparent tracking of task movement.

📊 Agile Boards in Jira

1️⃣ Scrum Board

Used for:

- Sprint planning
- Daily standups
- Tracking sprint progress

2️⃣ Kanban Board (Optional)

Used for:

- Continuous workflow
- API & bug tracking

📈 Agile Reports for Monitoring

Jira provides built-in reports:

- 📊 Burndown Chart – Sprint progress tracking
- 📈 Velocity Chart – Team performance
- 📋 Sprint Report – Completed vs pending work
- 🐞 Bug Report – Defect tracking

Mentors can review reports weekly to evaluate progress.

👥 Role-Based Access

Define permissions:

Role	Responsibilities
Project Admin	Configure board & workflow
Scrum Master	Sprint planning & review
Developer (Intern)	Task implementation
Reviewer	Code & feature review

🧠 **Daily Standup Format (Using Jira Board)**

Each intern updates:

- What was completed yesterday
- What will be done today
- Any blockers

Tasks must be moved across workflow stages accordingly.

🔐 **Integration Capabilities**

Jira can integrate with:

- GitHub (code tracking)
- CI/CD tools
- Slack or Zoho Cliq (notifications)
- Confluence (documentation)

🗓️ **Example Sprint Breakdown (2 Weeks)**

Sprint Goal: Complete Data Preprocessing

Planned Stories:

- Clean missing values
- Outlier detection
- Feature scaling
- Train-test split

Expected Outcome:
Clean dataset ready for model training.

✅ **Benefits of Using Jira**

- Structured Agile Planning
- Clear Accountability
- Transparent Sprint Tracking

- Performance Analytics
- Real-time Status Visibility
- Industry-standard project management tool

📌 **Final Outcome**

By implementing Jira for Agile planning, the internship project will follow:

- Scrum-based sprint execution
- Organized backlog management
- Efficient task delegation
- Measurable progress tracking
- Professional software development workflow

DATE	28-02-2026
TEAM ID	LTVIP2026TMIDS90703
PROJECT NAME	Dog breed identification using transfer learning
MAXIMUM MARKS	2 MARKS

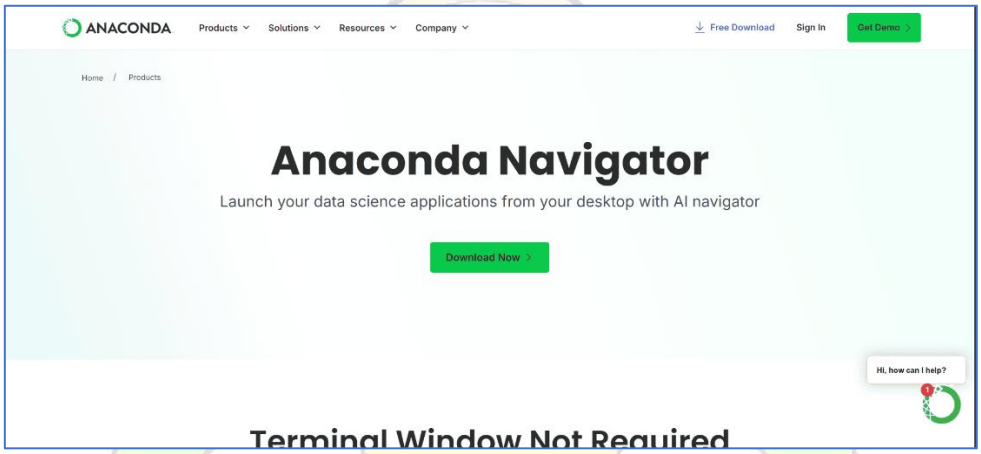
Chapter – 6

Pre-Requisites

→ **How to install Anaconda Navigator**

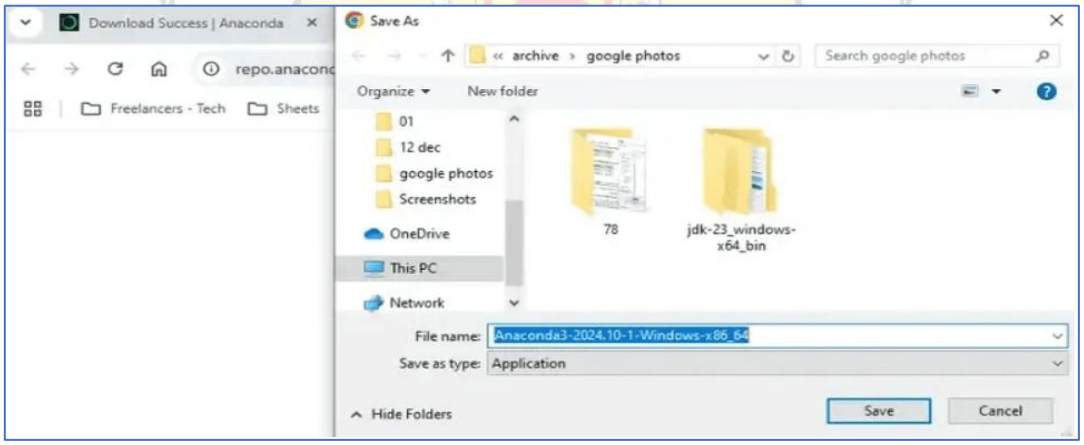
Step - 1: Open this link ***https://www.anaconda.com/products/navigator***

Step - 2: Click on **Download** button



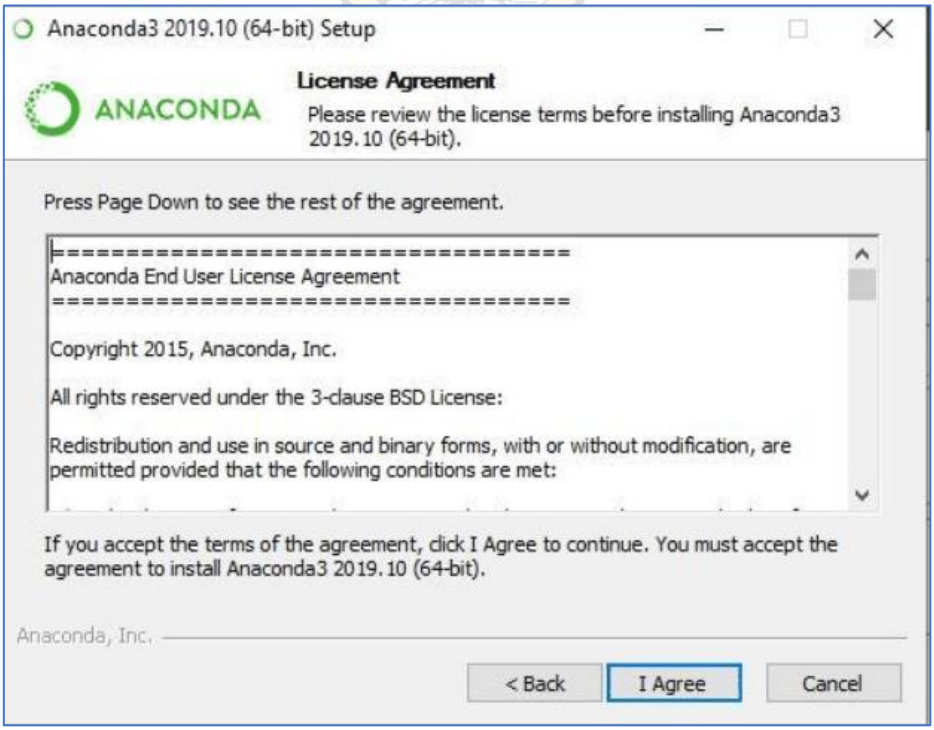
Step - 3: After **downloading** the **Anaconda Navigator** double click on the file

Step - 4: Select the **Windows Installer**

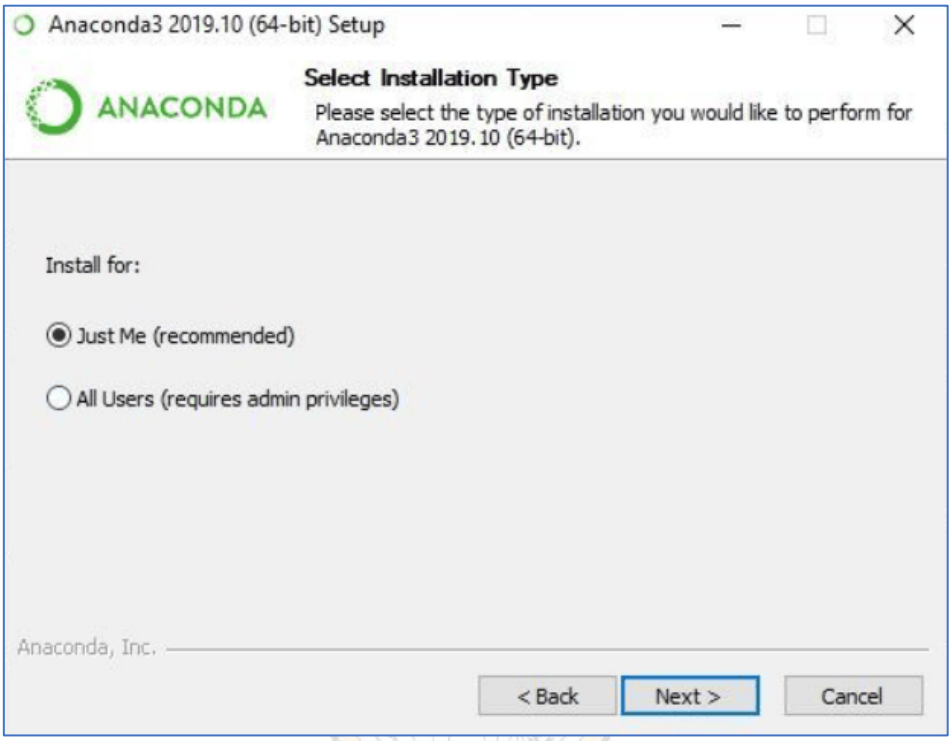


Step - 5: Begin the **installation** process

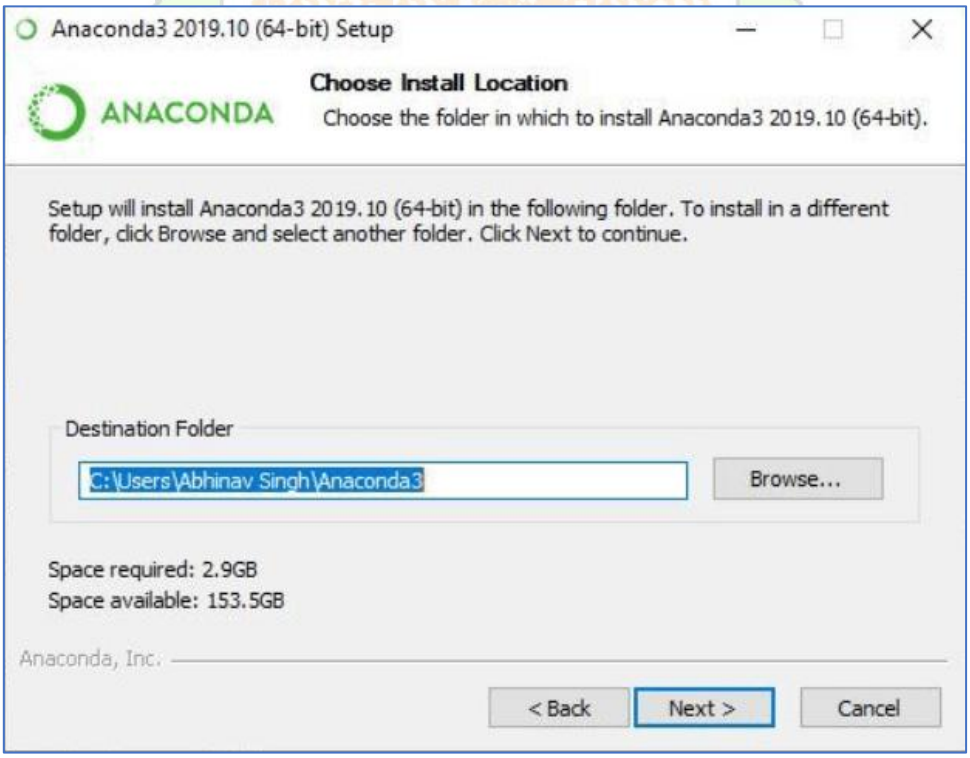
Step - 6: Getting through license agreement



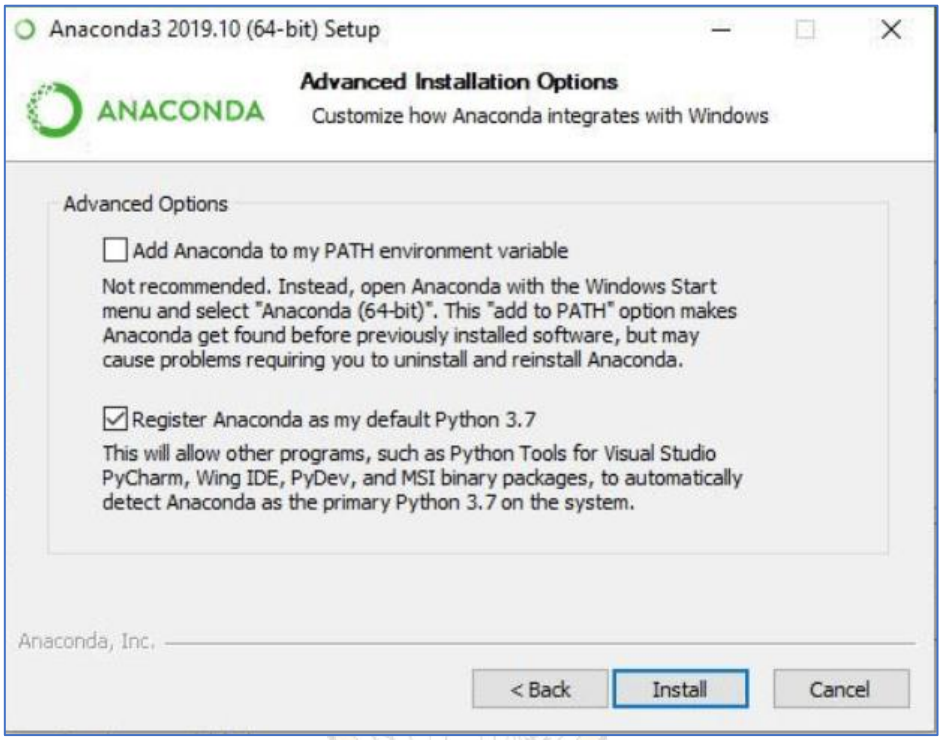
Step - 7: Select Installation Type



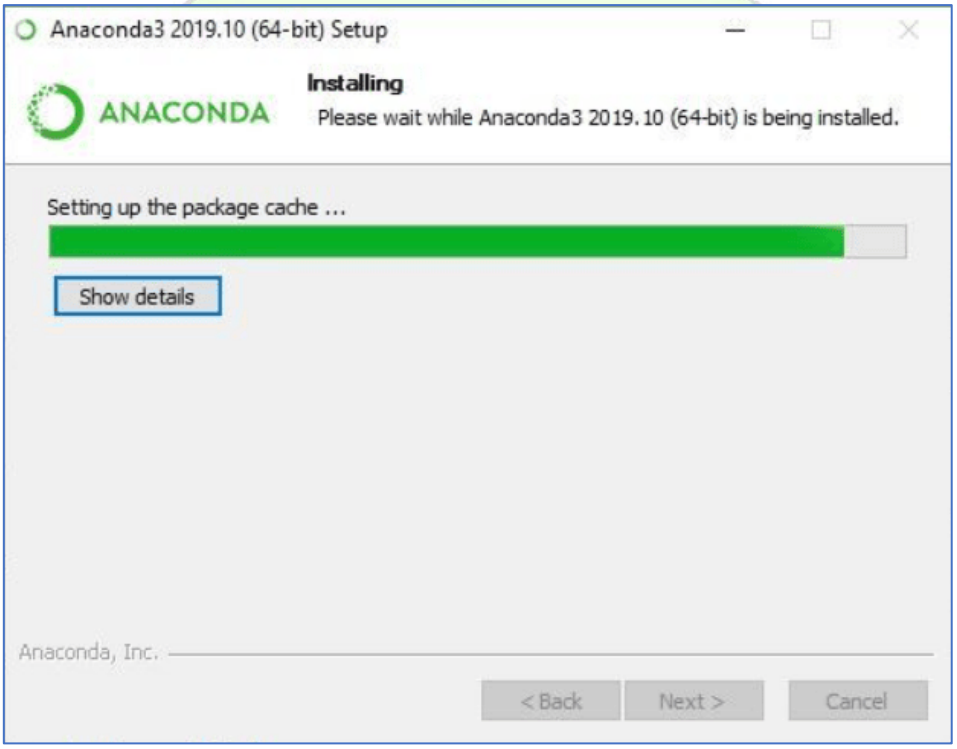
Step - 8: Select the path where you wish to install the file extractor and click "Next" to proceed ahead.



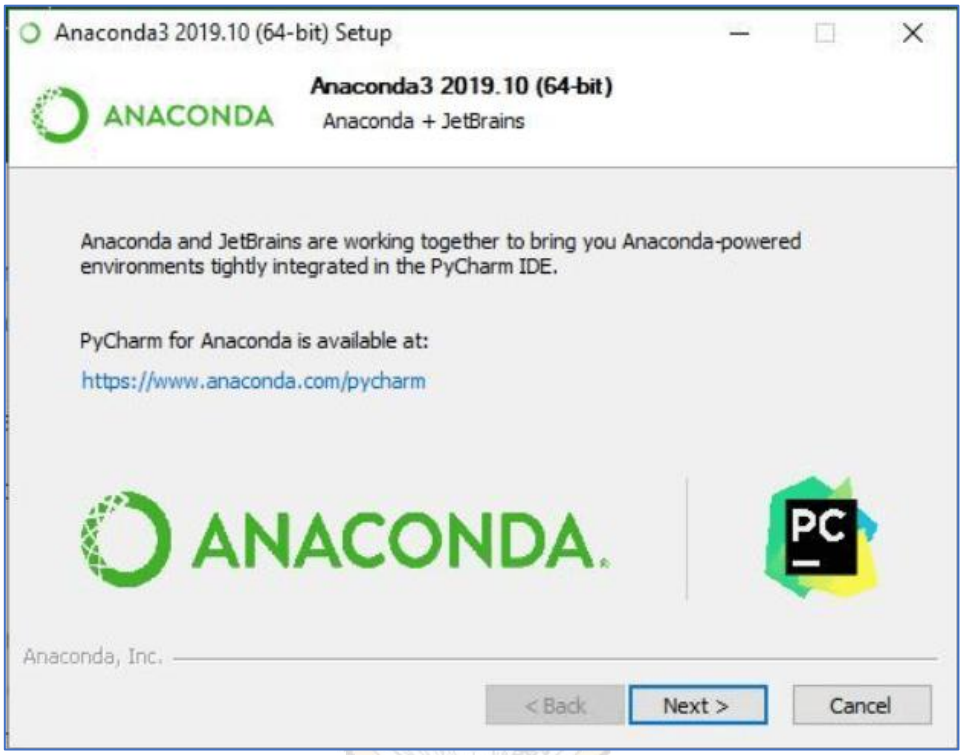
Step - 9: Advance installation options



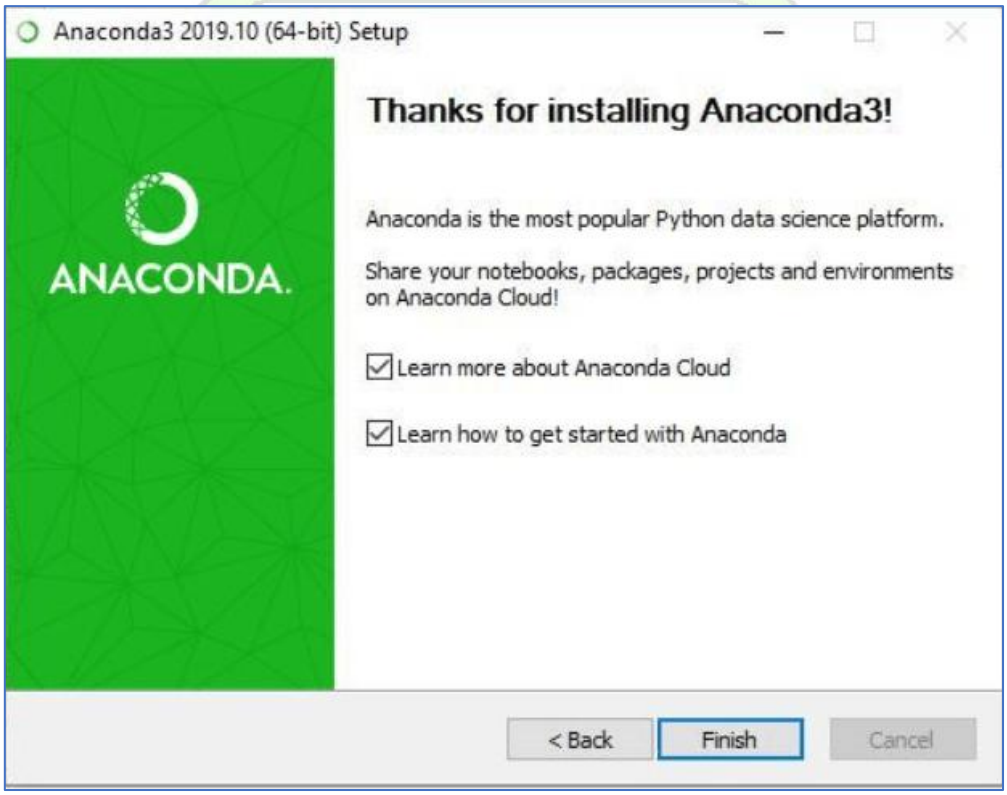
Step - 10: Click **Install** to start the **Anaconda Installation** process.



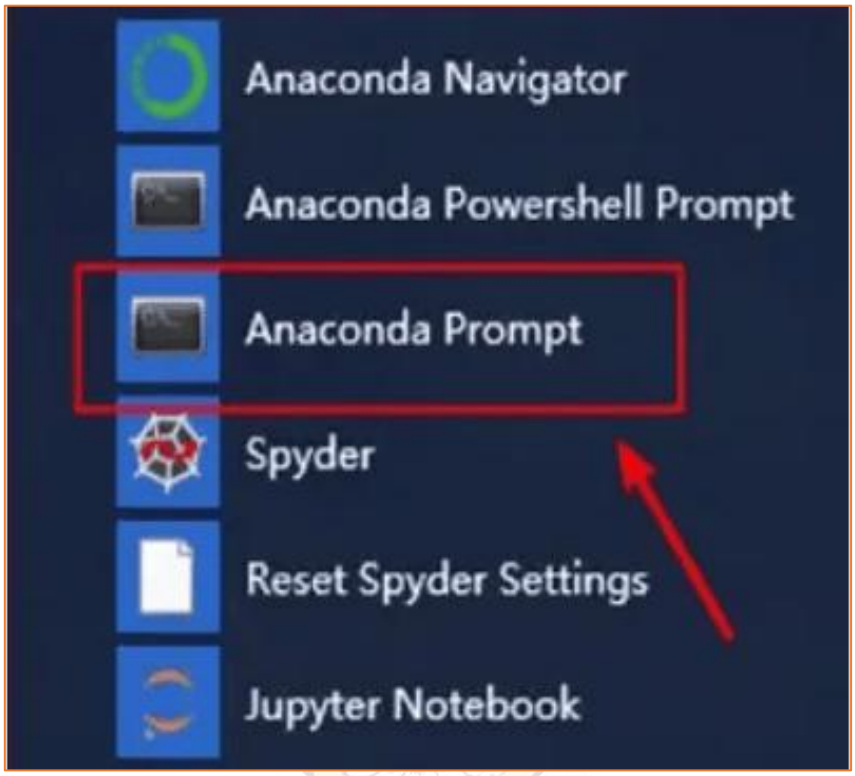
Step - 10: Recommendation to install **PyCharm**



Step - 11: Once the installation gets complete, click Finish to complete the process.



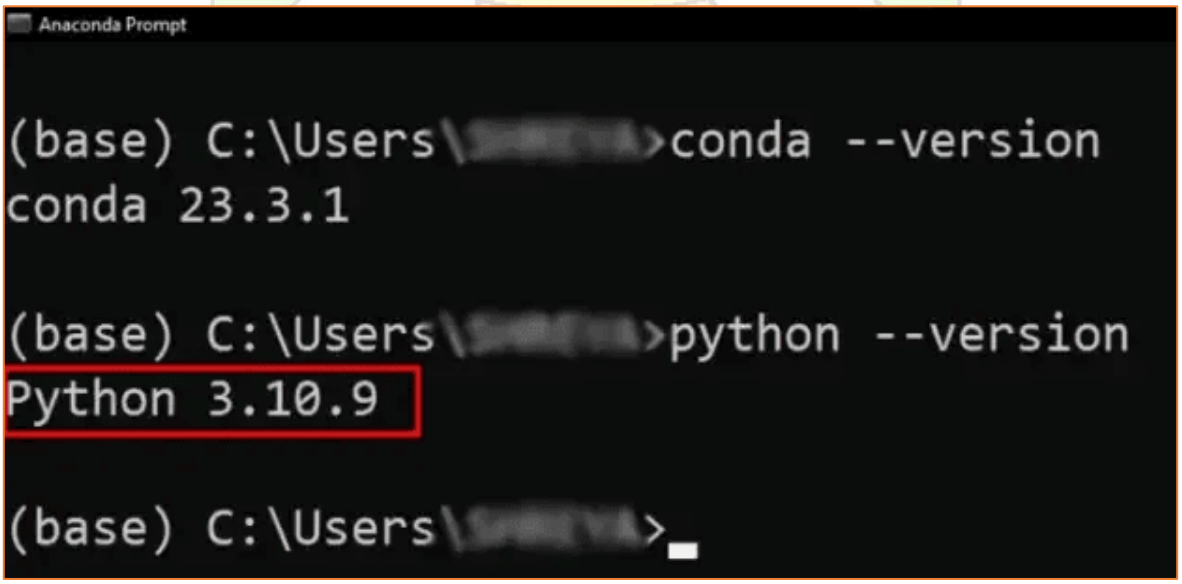
Step - 12: Click on the Start Menu and search for "Anaconda Prompt" and click to open it.



Step - 13: Run the Program to check for the Anaconda Version

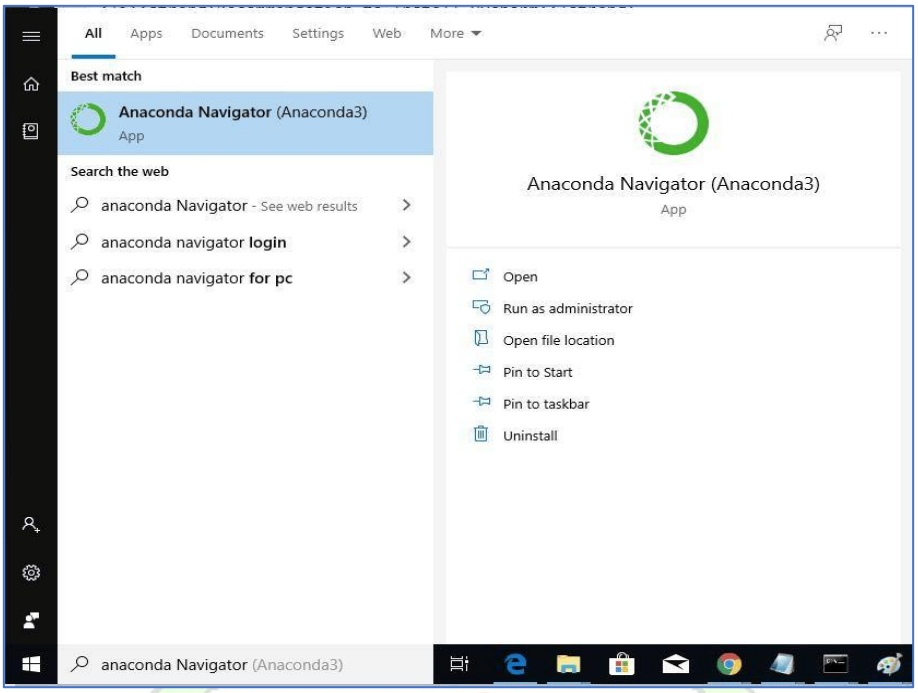
Type the following command to check for the installed version of conda:

Conda –version

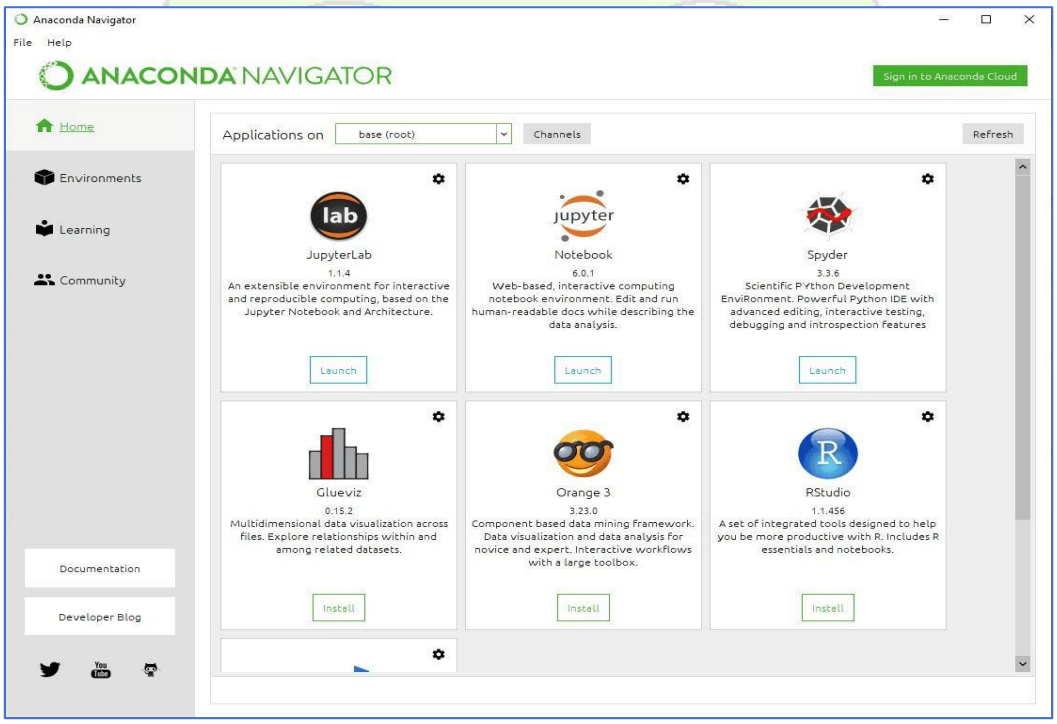


Step - 13: Access **Anaconda Navigator**

Once the installation process is done, Anaconda can be used to perform multiple operations. To begin using Anaconda, search for Anaconda Navigator from the Start Menu in Windows PC.



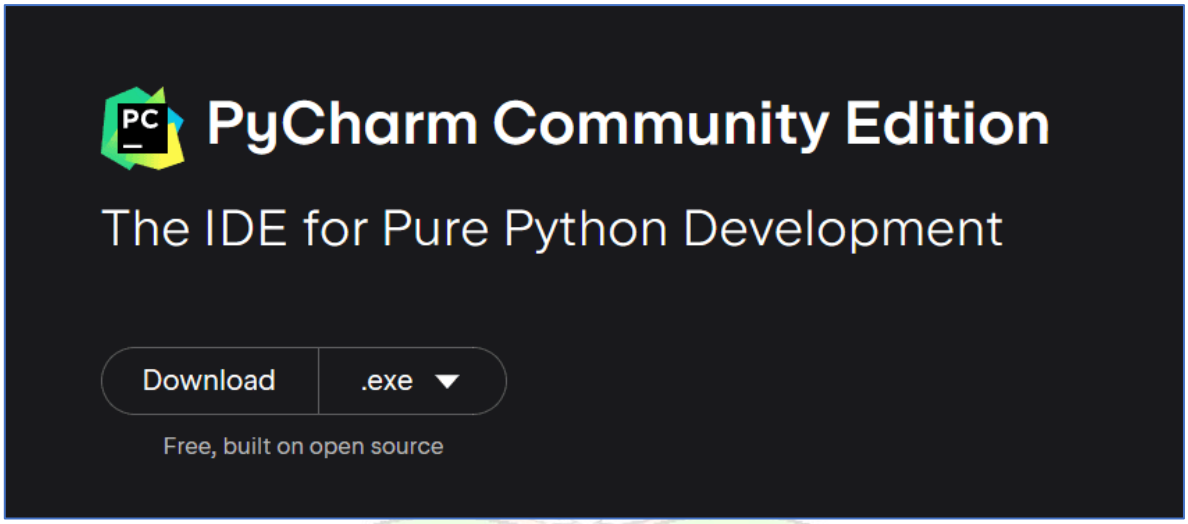
Step - 14: Explore **Navigator & Features**



How to Install PyCharm

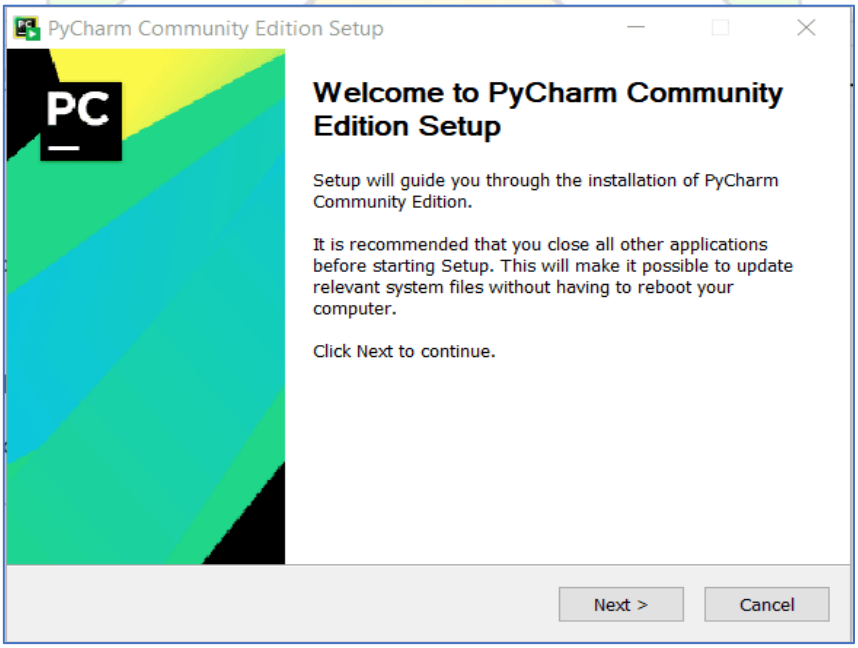
Step 1: Download **PyCharm**

- Go to the JetBrains PyCharm download page
- Choose either the Community edition (free) or the Professional edition (paid).
- Download the installer according to your preference.



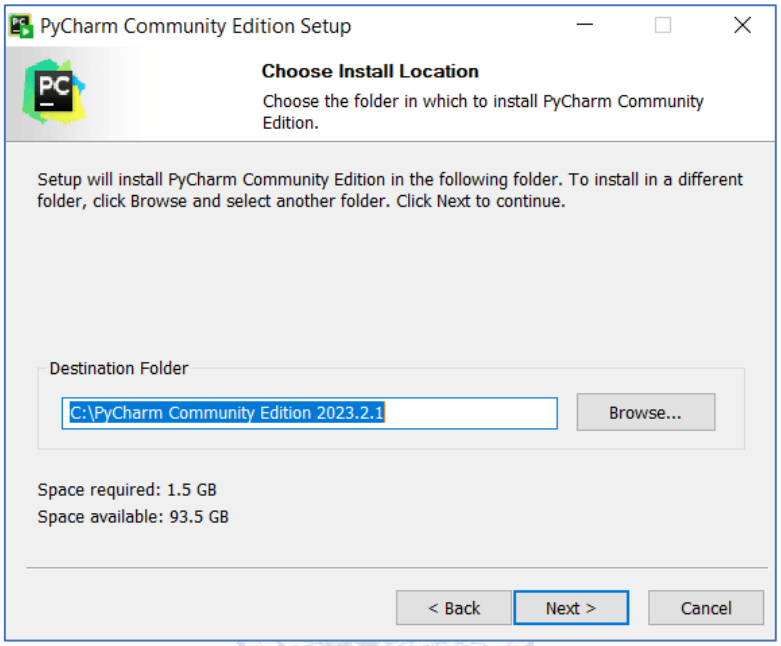
Step - 2: Run the Installer

- Locate the downloaded installer file and double-click it to run.
- Follow the on-screen instructions.



Step - 3: Choose Installation Location

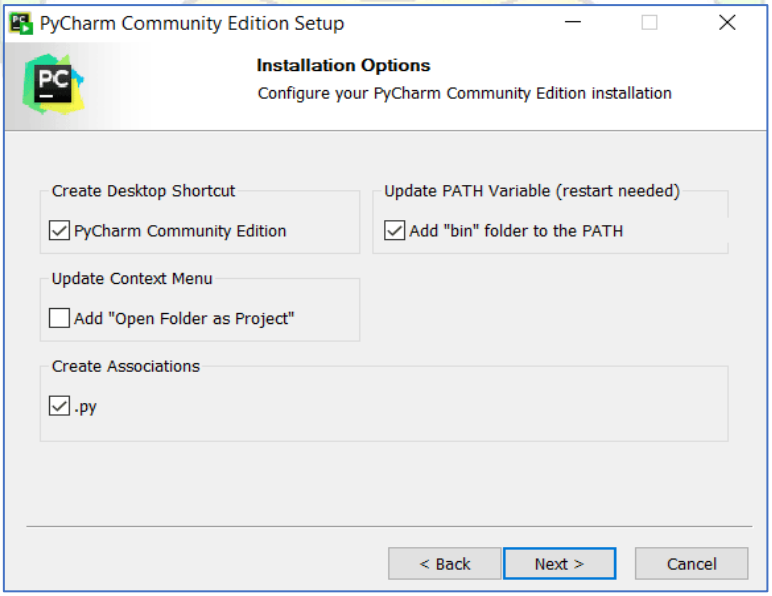
- Select the folder where you want to install PyCharm.



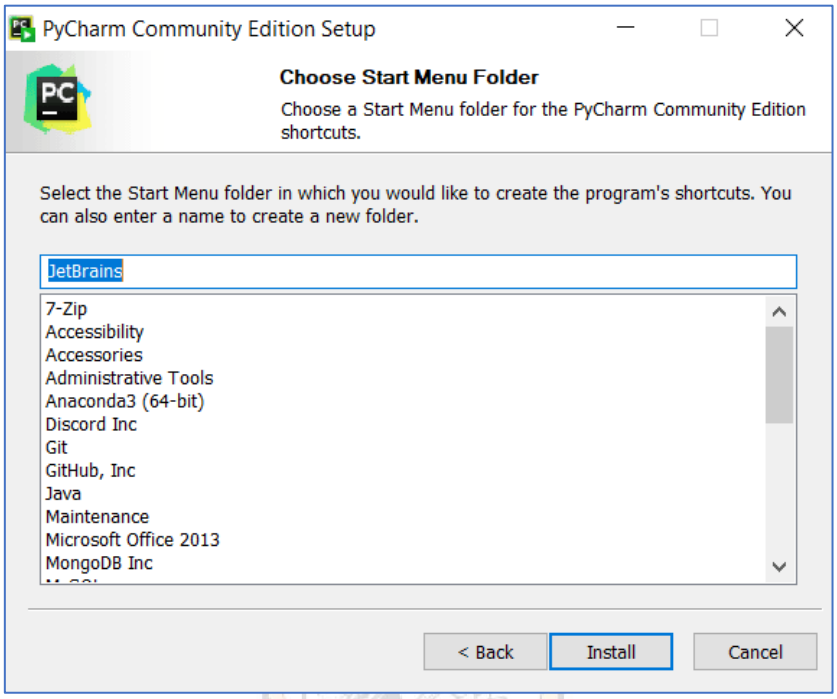
Step - 4: Select Installation Options

You can select the following options according to your preference:

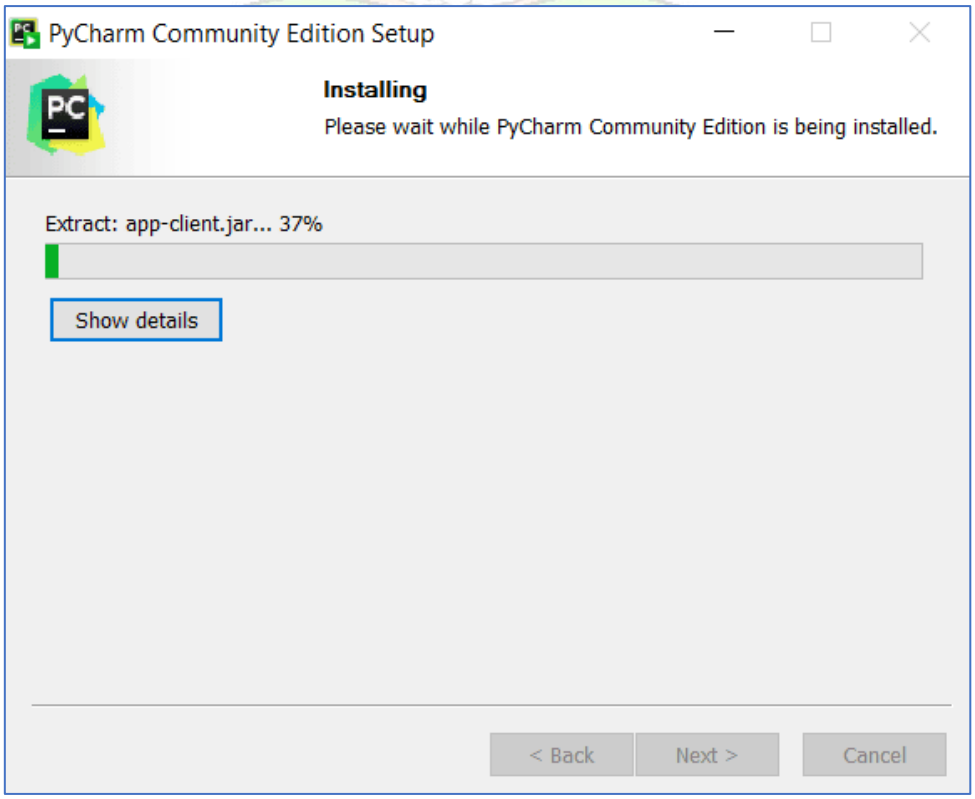
- Create Desktop Shortcut - adds a shortcut on your desktop.
- Add PyCharm to PATH - allows you to run PyCharm from the command line.
- Associate .py files with PyCharm - makes PyCharm the default editor for Python files.



- Choose folder name where your project will be saved as default.

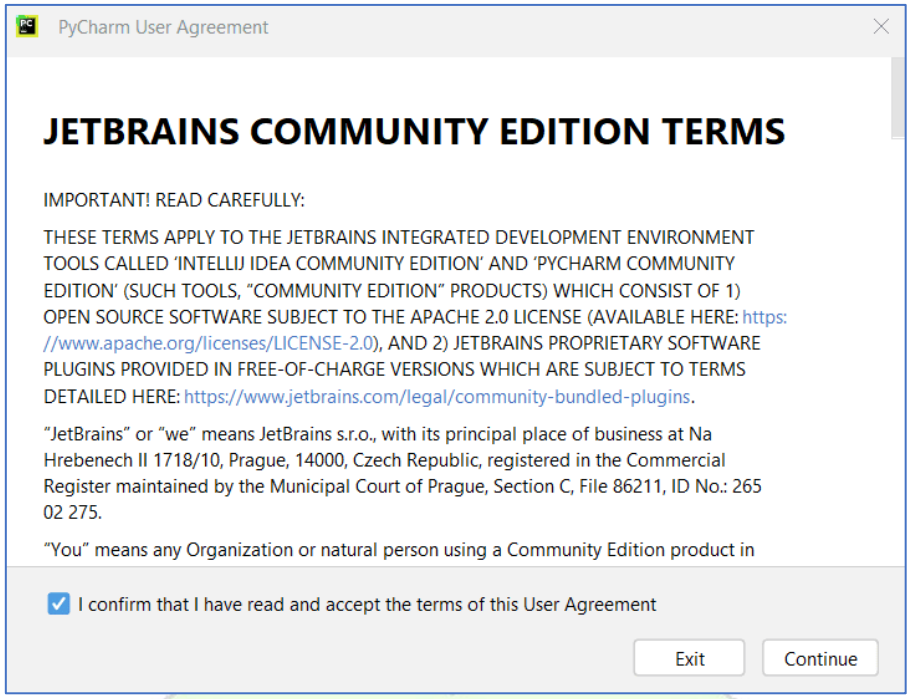


- Now it will download the PyCharm.



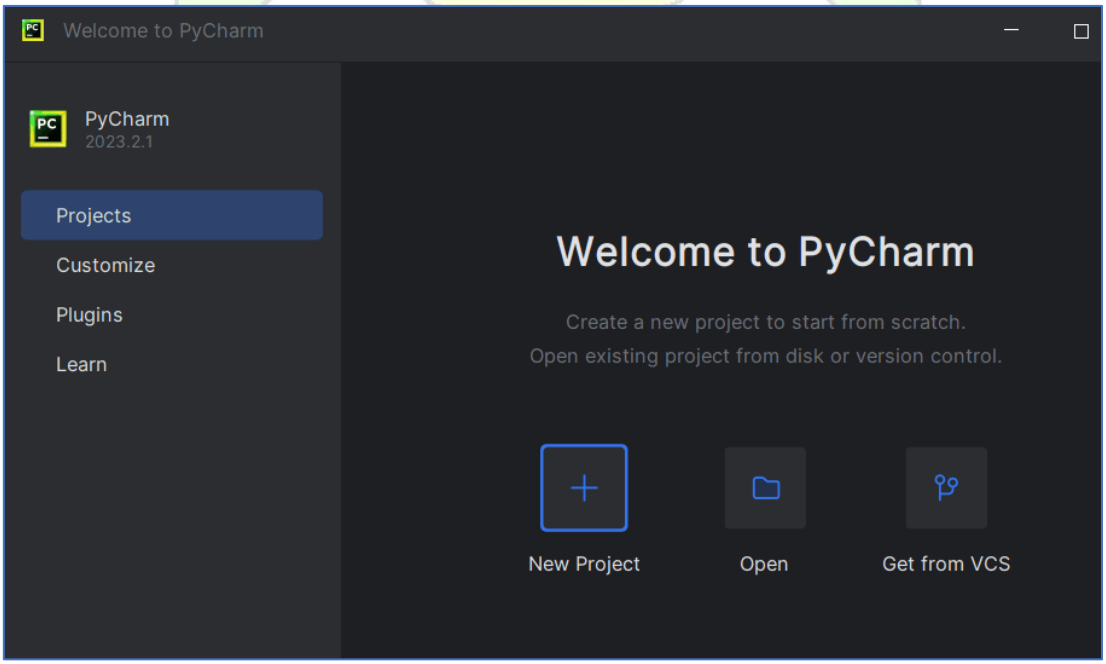
Step 5: Complete Installation

- Accept the license agreement and continue.
- PyCharm will be installed.



Step 6: Launch PyCharm

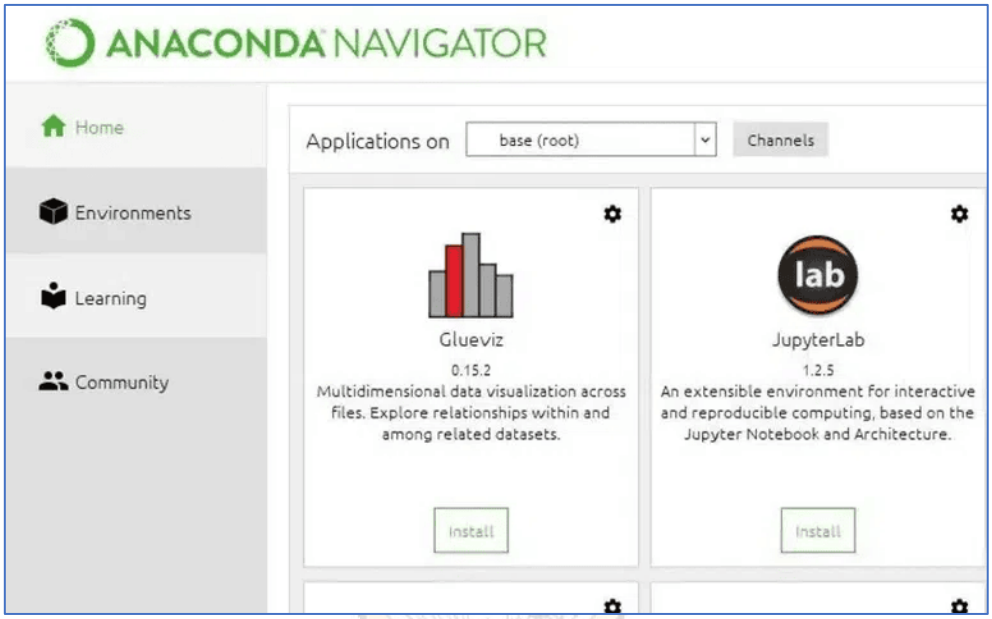
- Once installed, launch PyCharm either from the Start menu or the desktop shortcut.



How to open Jupyter Notebook

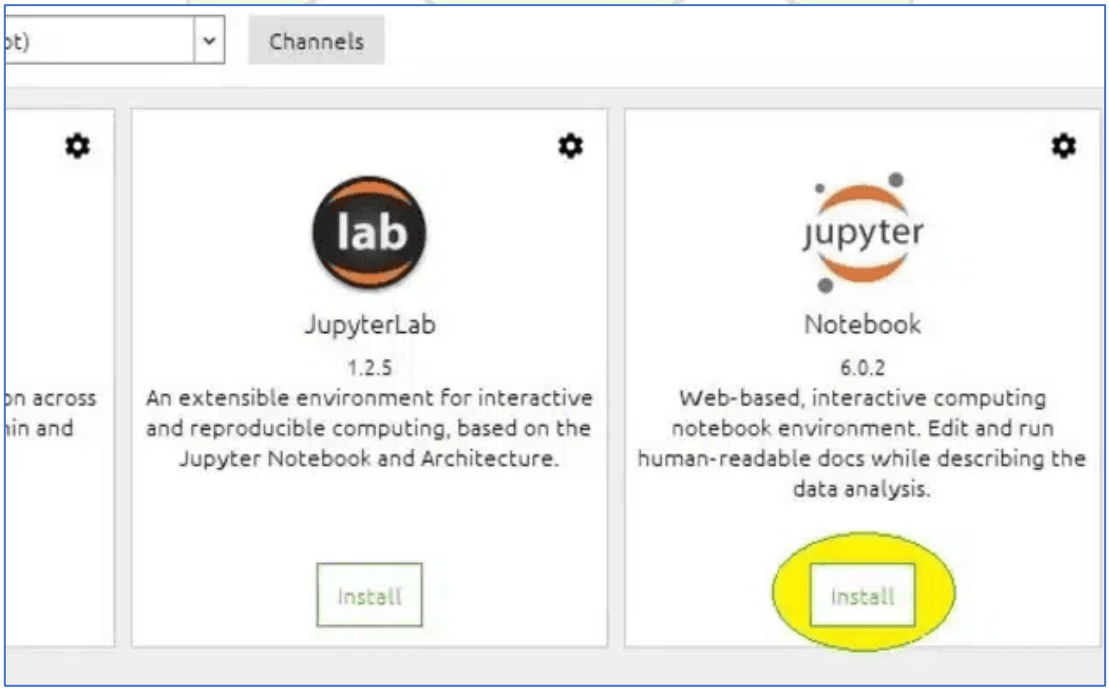
Step 1: Go to Anaconda Navigator

Firstly, launch anaconda and click on the Install Jupyter Notebook Button.



Step 2: Install Jupyter Notebook

Search for Jupyter Notebook and click on the Install button to begin with the installation process.

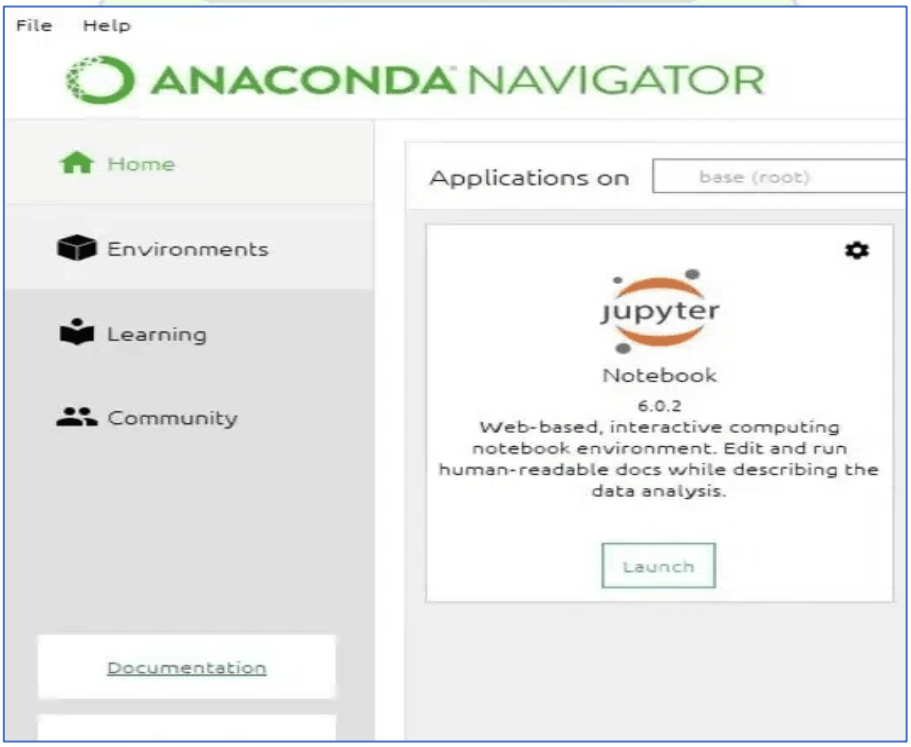


Step 3: Load Packages

Once the installation is complete, it will start loading the packages that comes along with it and click to finish the Installation.

Step 4: Launch Jupyter Notebook

Now, click on Launch button to start the Jupyter Notebook.



How to install Packages

If you are using PyCharm IDE, you can install the packages through the command prompt and follow the same syntax as above.

Type “pip install numpy” and click enter.

```
(.venv) PS D:\LT Python Project> pip install numpy
Requirement already satisfied: numpy in c:\users\hp\appdata\local\programs\python\python313\lib\site-packages (2.2.3)

[notice] A new release of pip is available: 25.1.1 -> 26.0.1
[notice] To update, run: python.exe -m pip install --upgrade pip
(.venv) PS D:\LT Python Project>
```

Type “pip install pandas” and click enter.

```
(.venv) PS D:\LT Python Project> pip install pandas
Requirement already satisfied: pandas in c:\users\hp\appdata\local\programs\python\python313\lib\site-packages (2.2.3)
Requirement already satisfied: numpy>=1.26.0 in c:\users\hp\appdata\local\programs\python\python313\lib\site-packages (from pandas) (2.2.3)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\hp\appdata\local\programs\python\python313\lib\site-packages (from pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in c:\users\hp\appdata\local\programs\python\python313\lib\site-packages (from pandas) (2025.1)
Requirement already satisfied: tzdata>=2022.7 in c:\users\hp\appdata\local\programs\python\python313\lib\site-packages (from pandas) (2025.1)
Requirement already satisfied: six>=1.5 in c:\users\hp\appdata\local\programs\python\python313\lib\site-packages (from python-dateutil>=2.8.2->pandas) (1.17.0)

[notice] A new release of pip is available: 25.1.1 -> 26.0.1
[notice] To update, run: python.exe -m pip install --upgrade pip
(.venv) PS D:\LT Python Project>
```

Type “pip install matplotlib” and click enter.

```
(.venv) PS D:\LT Python Project> pip install matplotlib
Requirement already satisfied: matplotlib in c:\users\hp\appdata\local\programs\python\python313\lib\site-packages (3.10.1)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\hp\appdata\local\programs\python\python313\lib\site-packages (from matplotlib) (1.3.1)
Requirement already satisfied: cycler>=0.10 in c:\users\hp\appdata\local\programs\python\python313\lib\site-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\hp\appdata\local\programs\python\python313\lib\site-packages (from matplotlib) (4.56.0)
Requirement already satisfied: kiwisolver>=1.3.1 in c:\users\hp\appdata\local\programs\python\python313\lib\site-packages (from matplotlib) (1.4.8)
Requirement already satisfied: numpy>=1.23 in c:\users\hp\appdata\local\programs\python\python313\lib\site-packages (from matplotlib) (2.2.3)
Requirement already satisfied: packaging>=20.0 in c:\users\hp\appdata\local\programs\python\python313\lib\site-packages (from matplotlib) (24.2)
Requirement already satisfied: pillow>=8 in c:\users\hp\appdata\local\programs\python\python313\lib\site-packages (from matplotlib) (11.1.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\hp\appdata\local\programs\python\python313\lib\site-packages (from matplotlib) (3.2.1)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\hp\appdata\local\programs\python\python313\lib\site-packages (from matplotlib) (2.9.0.post0)
Requirement already satisfied: six>=1.5 in c:\users\hp\appdata\local\programs\python\python313\lib\site-packages (from python-dateutil>=2.7->matplotlib) (1.17.0)

[notice] A new release of pip is available: 25.1.1 -> 26.0.1
[notice] To update, run: python.exe -m pip install --upgrade pip
(.venv) PS D:\LT Python Project>
```

Type “pip install scikit-learn” and click enter

```
(.venv) PS D:\LT Python Project> pip install scikit-learn
Requirement already satisfied: scikit-learn in c:\users\hp\appdata\local\programs\python\python313\lib\site-packages (1.7.2)
Requirement already satisfied: numpy>=1.22.0 in c:\users\hp\appdata\local\programs\python\python313\lib\site-packages (from scikit-learn) (2.2.3)
Requirement already satisfied: scipy>=1.8.0 in c:\users\hp\appdata\local\programs\python\python313\lib\site-packages (from scikit-learn) (1.16.2)
Requirement already satisfied: joblib>=1.2.0 in c:\users\hp\appdata\local\programs\python\python313\lib\site-packages (from scikit-learn) (1.5.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in c:\users\hp\appdata\local\programs\python\python313\lib\site-packages (from scikit-learn) (3.6.0)

[notice] A new release of pip is available: 25.1.1 -> 26.0.1
[notice] To update, run: python.exe -m pip install --upgrade pip
(.venv) PS D:\LT Python Project>
```

Type “pip install Flask” and click enter.

```
(.venv) PS D:\LT Python Project> pip install Flask
Requirement already satisfied: Flask in c:\users\hp\appdata\local\programs\python\python313\lib\site-packages (3.1.2)
Requirement already satisfied: blinker>=1.9.0 in c:\users\hp\appdata\local\programs\python\python313\lib\site-packages (from Flask) (1.9.0)
Requirement already satisfied: click>=8.1.3 in c:\users\hp\appdata\local\programs\python\python313\lib\site-packages (from Flask) (8.2.1)
Requirement already satisfied: itsdangerous>=2.2.0 in c:\users\hp\appdata\local\programs\python\python313\lib\site-packages (from Flask) (2.2.0)
Requirement already satisfied: jinja2>=3.1.2 in c:\users\hp\appdata\local\programs\python\python313\lib\site-packages (from Flask) (3.1.6)
Requirement already satisfied: markupsafe>=2.1.1 in c:\users\hp\appdata\local\programs\python\python313\lib\site-packages (from Flask) (3.0.2)
Requirement already satisfied: werkzeug>=3.1.0 in c:\users\hp\appdata\local\programs\python\python313\lib\site-packages (from Flask) (3.1.3)
Requirement already satisfied: colorama in c:\users\hp\appdata\local\programs\python\python313\lib\site-packages (from click>=8.1.3->Flask) (0.4.6)

[notice] A new release of pip is available: 25.1.1 -> 26.0.1
[notice] To update, run: python.exe -m pip install --upgrade pip
(.venv) PS D:\LT Python Project>
```

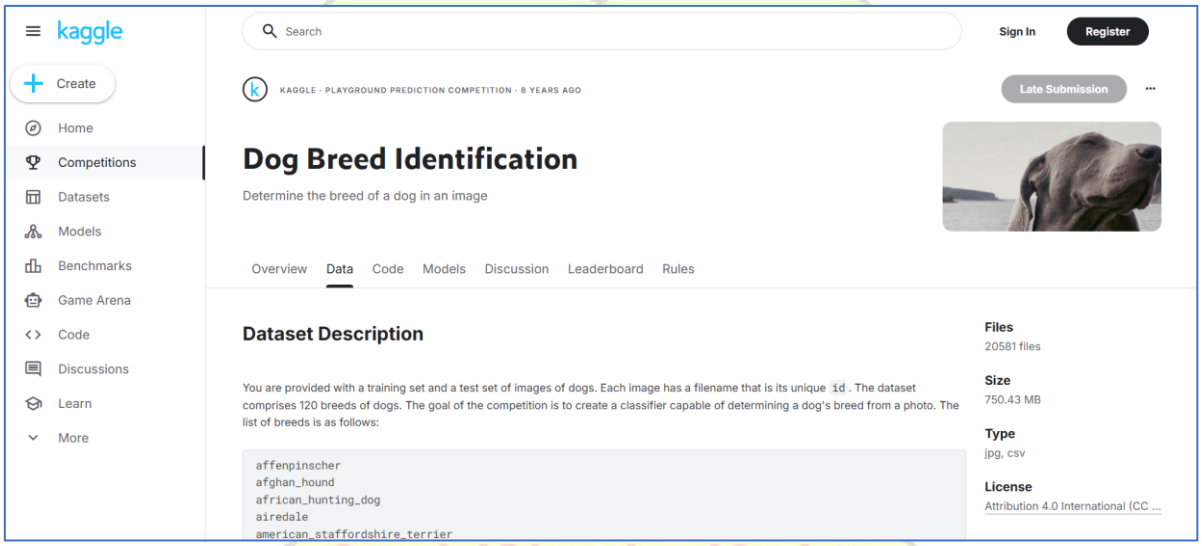
DATE	28-02-2026
TEAM ID	LTVIP2026TMIDS90703
PROJECT NAME	Dog breed identification using transfer learning
MAXIMUM MARKS	4 MARKS

6.2 - Data Collection

In this milestone First, we will collect images of Dog Breeds then organized into subdirectories based on their respective names as shown in the project structure. Create folders of types of Dog Breeds that need to be recognized. In this project, we have collected images of 20 types of Images like affenpinscher, beagle, appenzeller, basset, bluetick, boxer, cairn, doberman, german_shepherd, golden_retriever, kelpie, komondor, leonberg,

mexican_hairless, pug, redbone, shih-tzu, toy_poodle, vizsla, whippet they are saved in the respective sub directories with their respective names.

DownloadtheDataset- https://www.kaggle.com/competitions/dog-breed-identification/data?select=train



In Image Processing, we will be improving the image data that suppresses unwilling distortions or enhances some image features important for further processing, although perform some geometric transformations of images like rotation, scaling, translation, etc.

Dog breed identification using transfer learning

DATE	28-02-2026
TEAM ID	LTVIP2026TMIDS90703
PROJECT NAME	Dog breed identification using transfer learning
MAXIMUM MARKS	3 MARKS

6.3 - Data Pre-Processing

1: Organizing the Images into Different Classes.

```
[ ]  import os
     import shutil
     import sys

[ ]  dataset_dir = '/content/train'
     labels = pd.read_csv('/content/labels.csv')

[ ]  import os

     def make_dir(x):
         if os.path.exists(x)==False:
             os.makedirs(x)

     base_dir = './subset'
     make_dir(base_dir)

[ ]  train_dir = os.path.join(base_dir, 'train')
     make_dir(train_dir)

[ ]  breeds = labels.breed.unique()
     for breed in breeds:
         # Make folder for each breed
         _ = os.path.join(train_dir, breed)
         make_dir(_)

         # Copy images to the corresponding folders
         images = labels[labels.breed == breed]['id']
         for image in images:
             source = os.path.join(dataset_dir, f'{image}.jpg')
             destination = os.path.join(train_dir, breed,f'{image}.jpg')
             shutil.copyfile(source, destination)
```

The Images should be organized based on the Image id's. So that Training the Model will be simpler. For each image ID (image_id) in the current breed, the source path is formed by combining the dataset_dir and the image filename (f'{image_id}.jpg'). The destination path is formed by combining the breed_folder and the same image filename. shutil. copyfile () is used to copy the image from the source path to the destination path.

2: Import the ImageDataGenerator library.

Image data augmentation is a technique that can be used to artificially expand the size of a training dataset by creating modified versions of images in the dataset.

The Keras deep learning neural network library provides the capability to fit models using image data augmentation via the ImageDataGenerator class.

Let us import the ImageDataGenerator class from tensorflow Keras.

```
#import image datagenerator library
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

3: Configure ImageDataGenerator class

ImageDataGenerator class is instantiated and the configuration for the types of data augmentation. There are five main types of data augmentation techniques for image data; specifically:

- Image shifts via the width_shift_range and height_shift_range arguments.
- The image flips via the horizontal_flip and vertical_flip arguments.
- Image rotations via the rotation_range argument
- Image brightness via the brightness_range argument.
- Image zoom via the zoom_range argument.

```
train_datagen = ImageDataGenerator(rescale=1./255, shear_range=0.2,zoom_range=0.2,horizontal_flip=True)
```

4: Apply ImageDataGenerator functionality to Trainset and Test set:

Let us apply ImageDataGenerator functionality to Train set and Test set by using the following code. For Training set using flow_from_directory function.

This function will return batches of images from the subdirectories affenpinscher, beagle, appenzeller, basset, bluetick, boxer, cairn, doberman, german_shepherd, golden_retriever, kelpie, komondor, leonberg, mexican_hairless, pug, redbone, shih-tzu, toy_poodle, vizsla, whippe, together with labels 0 to 19.

Arguments:

- directory: Directory where the data is located. If labels are "inferred", it should contain subdirectories, each containing images for a class. Otherwise, the directory structure is ignored.
- batch_size: Size of the batches of data which is 32.
- target_size: Size to resize images after they are read from disk.
- class_mode:

  'int': means that the labels are encoded as integers (e.g. for sparse_categorical_crossentropy loss).

  'categorical' means that the labels are encoded as a categorical vector (e.g. for categorical_crossentropy loss).

- 'binary' means that the labels (there can be only 2) are encoded as float32 scalars with values 0 or 1 (e.g. for binary_crossentropy).
- None (no labels).

```
[ ]  datagen = ImageDataGenerator()
     generator = datagen.flow_from_directory(
         dataset,
         target_size=(224, 224),  # Adjust target size as needed
         batch_size=32,
         class_mode='categorical',
         shuffle=False,  # Ensure order is maintained for class indices
         classes=selected_classes  # Specify the selected classes
     )

     Found 1683 images belonging to 20 classes.
```

```
test_datagen = ImageDataGenerator(rescale=1./255)
```

We notice that 1683 images belong to 20 classes for training purposes.

Dog breed identification using transfer learning

DATE	28-02-2026
TEAM ID	LTVIP2026TMIDS90703
PROJECT NAME	Dog breed identification using transfer learning
MAXIMUM MARKS	4 MARKS

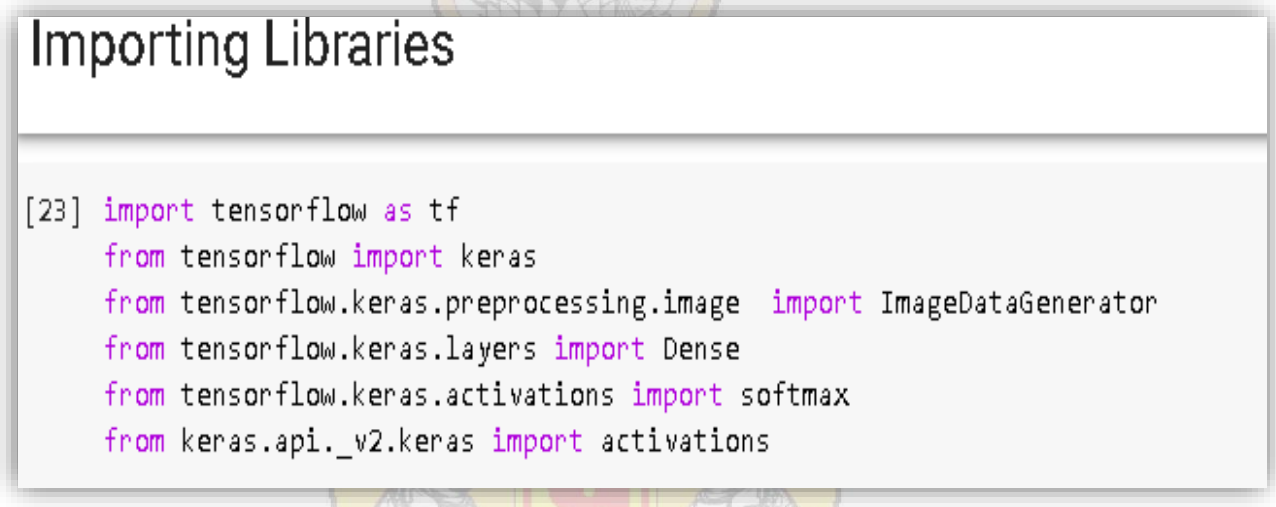
Model Building

Now it's time to build our Convolutional Neural Networking using vgg19 which contains an input layer along with the convolution, max-pooling, and finally an output layer.

Link: https://thesmartbridge.com/documents/spsaimldocs/CNNflow.pdf

1: Importing the Model Building Libraries

Importing the necessary libraries



2: Importing the VGG19 model:

To initialize the VGG19 model, the weights are usually pre-trained on the ImageNet dataset, which is a large-scale dataset of images belonging to 1,000 different categories. These pre-trained weights can be downloaded from the internet, and they can be used as a starting point to fine-tune the model for a specific task, such as object recognition or classification.

```
from tensorflow.keras.applications.vgg19 import VGG19
from tensorflow.keras.layers import Dense, Flatten
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
```

3: Initializing the model:

- The model will be initialized with the pre-trained weights from the ImageNet dataset, and the last fully connected layer will be excluded from the model architecture.
- The loop that follows freezes the weights of all the layers in the VGG19 model by setting

  `i.trainable=False` for each layer in the model. This is done to prevent the weights from being updated during training, as the model is already pre-trained on a large dataset.
- Finally, a `Flatten()` layer is added to the output of the VGG19 model to convert the output tensor into a 1D tensor.
- The resulting model can be used as a feature extractor for transfer learning or as a starting point for building a new model on top of it.

```
[ ] Image_size=[224,224]

[ ] sol=VGG19(input_shape=Image_size + [3] , weights='imagenet' , include_top = False)

[ ] for i in sol.layers:
      i.trainable = False

[ ] y=Flatten()(sol.output)
```

4: Adding Fully connected Layers

- For information regarding CNN Layers refer to the link Link: https://victorzhou.com/blog/intro-to-cnns-part-1/
- As the input image contains three channels, we are specifying the input shape as (128,128,3).
- We are adding a output layer with activation function as “softmax”.

```
[ ]  from keras.api._v2.keras import activations
     final = Dense(20, activation = 'softmax')(y)
```

A dense layer is a deeply connected neural network layer. It is the most common and frequently used layer. The number of neurons in the Dense layer is the same as the number of classes in the training set.The neurons in the last Dense layer, use softmax activation to convert their outputs into respective probabilities.

Understanding the model is a very important phase to properly use it for training and prediction purposes. Keras provides a simple method, summary to get the full information about the model and its layers.

```
vgg19_model.summary()

Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 224, 224, 3)]     0
 block1_conv1 (Conv2D)       (None, 224, 224, 64)      1792
 block1_conv2 (Conv2D)       (None, 224, 224, 64)      36928
 block1_pool (MaxPooling2D)  (None, 112, 112, 64)      0
 block2_conv1 (Conv2D)       (None, 112, 112, 128)     73856
 block2_conv2 (Conv2D)       (None, 112, 112, 128)     147584
 block2_pool (MaxPooling2D)  (None, 56, 56, 128)       0
 block3_conv1 (Conv2D)       (None, 56, 56, 256)       295168
 block3_conv2 (Conv2D)       (None, 56, 56, 256)       590080
 block3_conv3 (Conv2D)       (None, 56, 56, 256)       590080
 block3_conv4 (Conv2D)       (None, 56, 56, 256)       590080
 block3_pool (MaxPooling2D)  (None, 28, 28, 256)       0
 block4_conv1 (Conv2D)       (None, 28, 28, 512)       1180160
 block4_conv2 (Conv2D)       (None, 28, 28, 512)       2359808
 block4_conv3 (Conv2D)       (None, 28, 28, 512)       2359808
 block4_conv4 (Conv2D)       (None, 28, 28, 512)       2359808
 block4_pool (MaxPooling2D)  (None, 14, 14, 512)       0
 block5_conv1 (Conv2D)       (None, 14, 14, 512)       2359808
```

5: Configure The Learning Process

- The compilation is the final step in creating a model. Once the compilation is done, we can move on to the training phase. The loss function is used to find errors or deviations inthe learning process. Keras requires a loss function during the model compilation process.
- Optimization is an important process that optimizes the input weights by comparing the prediction and the loss function. Here we are using adam optimizer.
- Metrics are used to evaluate the performance of your model. It is similar to the loss function, but not used in the training process.

```
vgg19_model.compile(optimizer = 'adam' , loss = 'categorical_crossentropy' , metrics = ['Accuracy'])
```

6: Train The model

Now, let us train our model with our image dataset. The model is trained for 6 epochs and after every epoch, the current model state is saved if the model has the least loss encountered till that time. We can see that the training loss decreases in almost every epoch till 30 epochs and probably there is further scope to improve the model.

fit_generator functions used to train a deep learning neural network.

Arguments:

- steps_per_epoch: it specifies the total number of steps taken from the generator as soon as one epoch is finished and the next epoch has started. We can calculate the value of steps_per_epoch as the total number of samples in your dataset divided by the batch size.
- Epochs: an integer and number of epochs we want to train our model for.
- validation_data can be either:
    - an inputs and targets list
    - a generator
    - an inputs, targets, and sample_weights list which can be used to evaluate the loss and metrics for any model after any epoch has ended.
- validation_steps: only if the validation_data is a generator then only this argument can be used. It specifies the total number of steps taken from the generator before it is

stopped at every epoch and its value is calculated as the total number of validation data points in your dataset divided by the validation batch size.

```
[ ]  vgg19_model.fit(generator, epochs = 6)

     Epoch 1/6
     53/53 [==============================] - 1353s 25s/step - loss: 126.8239 - Accuracy: 0.1218
     Epoch 2/6
     53/53 [==============================] - 1352s 26s/step - loss: 30.5051 - Accuracy: 0.6049
     Epoch 3/6
     53/53 [==============================] - 1336s 25s/step - loss: 3.8852 - Accuracy: 0.9008
     Epoch 4/6
     53/53 [==============================] - 1333s 25s/step - loss: 1.1688 - Accuracy: 0.9584
     Epoch 5/6
     53/53 [==============================] - 1332s 25s/step - loss: 1.6495 - Accuracy: 0.9548
     Epoch 6/6
     53/53 [==============================] - 1287s 24s/step - loss: 0.2561 - Accuracy: 0.9857
     <keras.callbacks.History at 0x7f66172dd540>
```

7: Save the Model

The model is saved with .h5 extension as follows.

An H5 file is a data file saved in the Hierarchical Data Format (HDF). It contains multidimensional arrays of scientific data.

```
[ ]  vgg19_model.save("dogbreed.h5")
```

8: Test The model

Evaluation is a process during the development of the model to check whether the model is the best fit for the given problem and corresponding data.

```
[ ] import numpy as np
    from keras.preprocessing import image
    from keras.applications.vgg16 import preprocess_input
    from tensorflow.keras.preprocessing.image import load_img, img_to_array

[ ] img_path ='/content/test/000621fb3cbb32d8935728e48679680e.jpg'

[ ] img = load_img(img_path, target_size=(224, 224))
    x = img_to_array(img)
    x = preprocess_input(x)
    preds = vgg19_model.predict(np.array([x]))

    1/1 [==============================] - 1s 1s/step

[ ] class_names=['affenpinscher','beagle','appenzeller','basset','bluetick','boxer','cairn','doberman','german_shepherd','golden_retriever','kelpie','komondor','leonberg',

[ ] class_names[np.argmax(preds)]
```

Taking an image as input and checking the results

By using the model we are predicting the output for the given input image. The predicted class index name will be printed here.

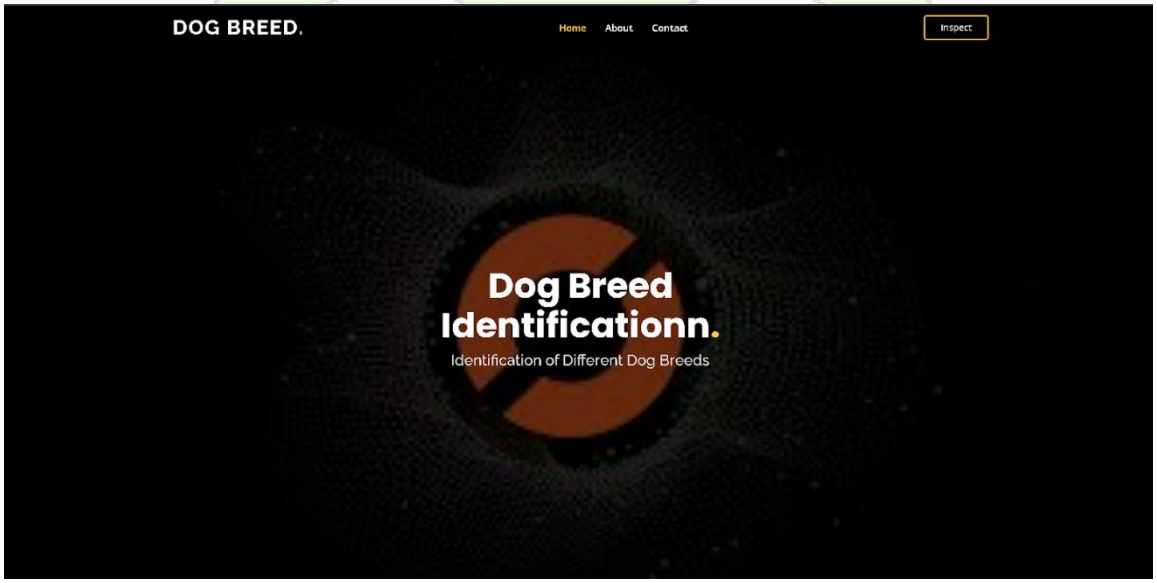
DATE	28-02-2026
TEAM ID	LTVIP2026TMIDS90703
PROJECT NAME	Dog breed identification using transfer learning
MAXIMUM MARKS	4 MARKS

6.5 - Application Building

1 : Create HTML Pages

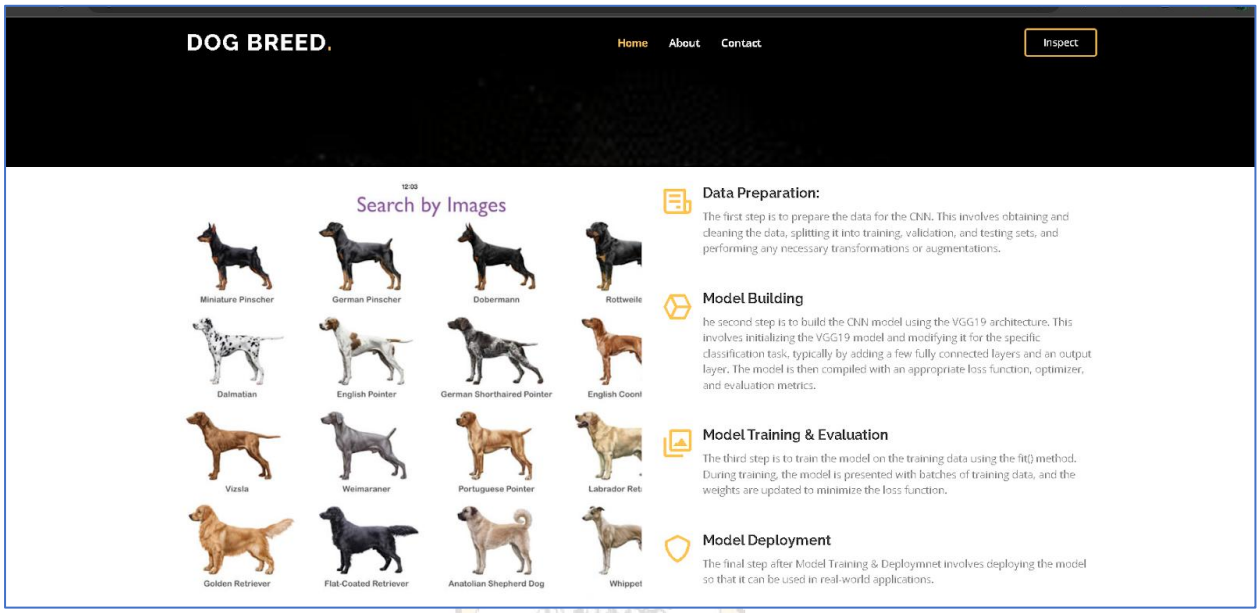
- We use HTML to create the front end part of the web page.
- Here, we have created 3 HTML pages- index.html, predict.html, and output.html
- home.html displays the home page.
- index.html displays an introduction about the project
- upload.html gives the emergency alert For more information regarding HTML https://www.w3schools.com/html/
- We also use JavaScript-main.js and CSS-main.css to enhance our functionality and view of HTML pages.
  - Link :CSS , JS

index.html looks like this

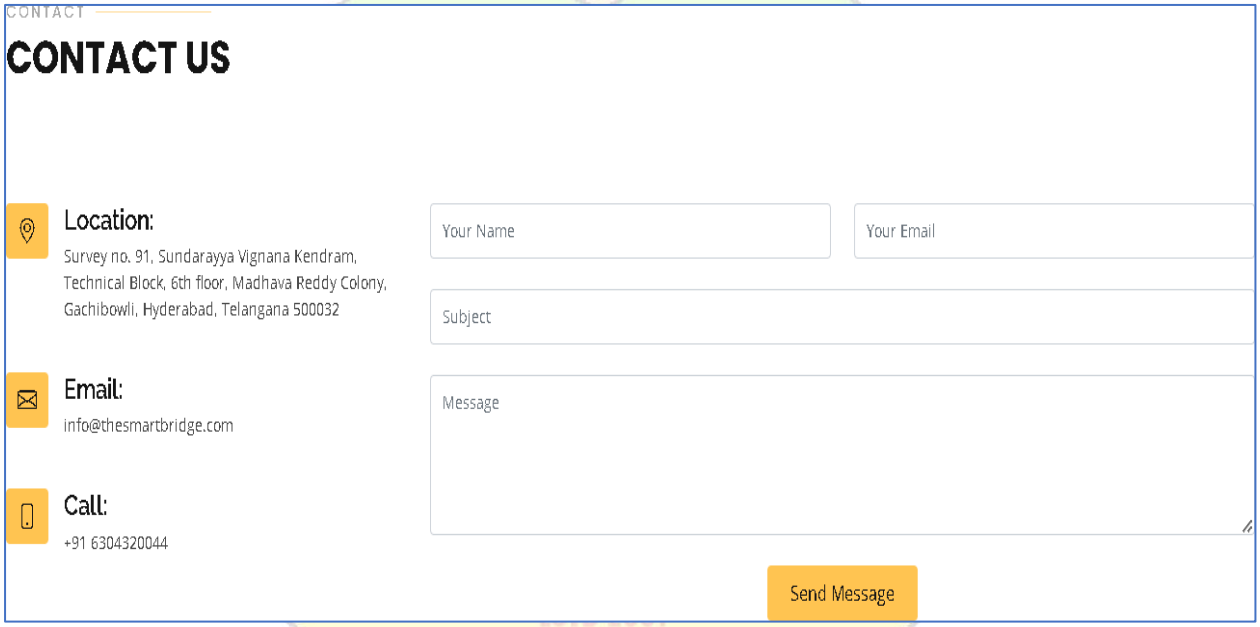


Dog breed identification using transfer learning

About Section: -



Contact Us: -



2: Build python code

Building Python Code

1: Importing Libraries

The first step is usually importing the libraries that will be needed in the program.

Importing the flask module in the project is mandatory. An object of the Flask class is our WSGI application. Flask constructor takes the name of the current module (name) as argument Pickle library to load the model file.

```
import numpy as np
import os
import tensorflow as tf
from PIL import Image
from flask import Flask, render_template,request,jsonify,url_for,redirect
from tensorflow.keras.preprocessing.image import load_img,img_to_array
```

2: Creating our flask application and loading our model by using load_model method

```
app=Flask(__name__)
model = tf.keras.models.load_model('dogbreed.h5')
```

3: Routing to the html Page

Here, the declared constructor is used to route to the HTML page created earlier.

In the above example, '/' URL is bound with index.html function. Hence, when the home page of a web server is opened in the browser, the html page will be rendered. Whenever you browse an image from the html page this photo can be accessed through POST or GET Method.

```
@app.route('/')
def index():
    return render_template("index.html")

@app.route('/predict')
def predict():
    return render_template("predict.html")
```

Showcasing prediction on UI:

```
@app.route('/output',methods=['GET','POST'])
def output():
    if request.method=='POST':
        f=request.files['file']
        basepath=os.path.dirname(__file__)
        filepath=os.path.join(basepath,'uploads',f.filename)
        f.save(filepath)
        img=load_img(filepath,target_size=(224,224))
        # Resize the image to the required size

        # Convert the image to an array and normalize it
        image_array = np.array(img)
        # Add a batch dimension
        image_array = np.expand_dims(image_array, axis=0)
        # Use the pre-trained model to make a prediction
        pred=np.argmax(model.predict(image_array),axis=1)
        index=['affenpinscher','beagle','appenzeller','basset','bluetick','boxer','cairn','doberman','german_shepherd','golden_retriever',
        prediction = index[int(pred)]
        print("prediction")
        return render_template("output.html",predict = prediction)
```

Here we are defining a function which requests the browsed file from the html page using the post method. The requested picture file is then saved to the uploads folder in this same directory using OS library. Using the load image class from Keras library we are retrieving the saved picture from the path declared. We are applying some image processing techniques and then sending that preprocessed image to the model for predicting the class. This returns the numerical value of a class (like 0 to 19.) which lies in the 0th index of the variable preds. This numerical value is passed to the index variable declared. This returns the name of the class. This name is rendered to the prediction variable used in the html page.

Finally, Run the application

This is used to run the application in a local host.

```
if __name__ == '__main__':
    app.run(debug=False,threaded = False)
```

3: Run the application

- Open the anaconda prompt from the start menu.
- Navigate to the folder where your app.py resides.
- Now type “python app.py” command.
- It will show the local host where your app is running on http://127.0.0.1.5000/
- Copy that local host URL and open that URL in the browser. It does navigate me to where you can view your web page.
- Enter the values, click on the predict button and see the result/prediction on the web page.

Then it will run on localhost: 5000

```
PS F:\Smart_Internz\Dog_Breed_prediction> python -u "f:\Smart_Internz\Dog_Breed_prediction\app.py"
 * Serving Flask app 'app' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 580-415-876
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
127.0.0.1 - - [11/Jul/2023 12:15:29] "GET / HTTP/1.1" 200 -
```

Navigate to the localhost (http://127.0.0.1:5000/)where you can view your web page.

DATE	28-02-2026
TEAM ID	LTVIP2026TMIDS90703
PROJECT NAME	Dog breed identification using transfer learning
MAXIMUM MARKS	4 MARKS

Chapter 7

7. Functional and Performance Testing

7.1 - Functional and Performance Testing

Testing is a critical phase in the development of the Dog Breed Identification using Transfer Learning system. It ensures that the application performs according to the specified requirements and delivers accurate, reliable, and efficient results. Functional testing verifies whether each component of the system operates correctly, while performance testing evaluates the system's speed, stability, and scalability under different conditions.

Functional Testing

Functional testing focuses on validating the core functionalities of the system to ensure that it behaves as expected. In this project, functional testing begins with verifying the image upload feature. The system must accept valid image formats such as JPG or PNG and reject unsupported file types with appropriate error messages. This ensures secure and proper data handling.

The testing process also verifies edge cases, such as uploading extremely small images, large images, or images that do not contain a dog. These tests ensure that the system remains stable and provides meaningful feedback to users.

Performance Testing

Performance testing evaluates how efficiently the system operates under different workloads and conditions. One of the primary performance metrics in this project is model accuracy. The trained model is evaluated using validation and test datasets to measure classification accuracy. Metrics such as precision, recall, and F1-score may also be analyzed to assess model effectiveness.

Response time is another critical performance factor. The system must process uploaded images and return predictions within a few seconds to maintain a smooth user experience. Testing ensures that the prediction pipeline—including preprocessing and inference—operates efficiently without unnecessary delays.

Finally, stress testing ensures that the system does not crash under extreme conditions, such as repeated uploads or high server load. This confirms the robustness and reliability of the deployed web application.

Conclusion

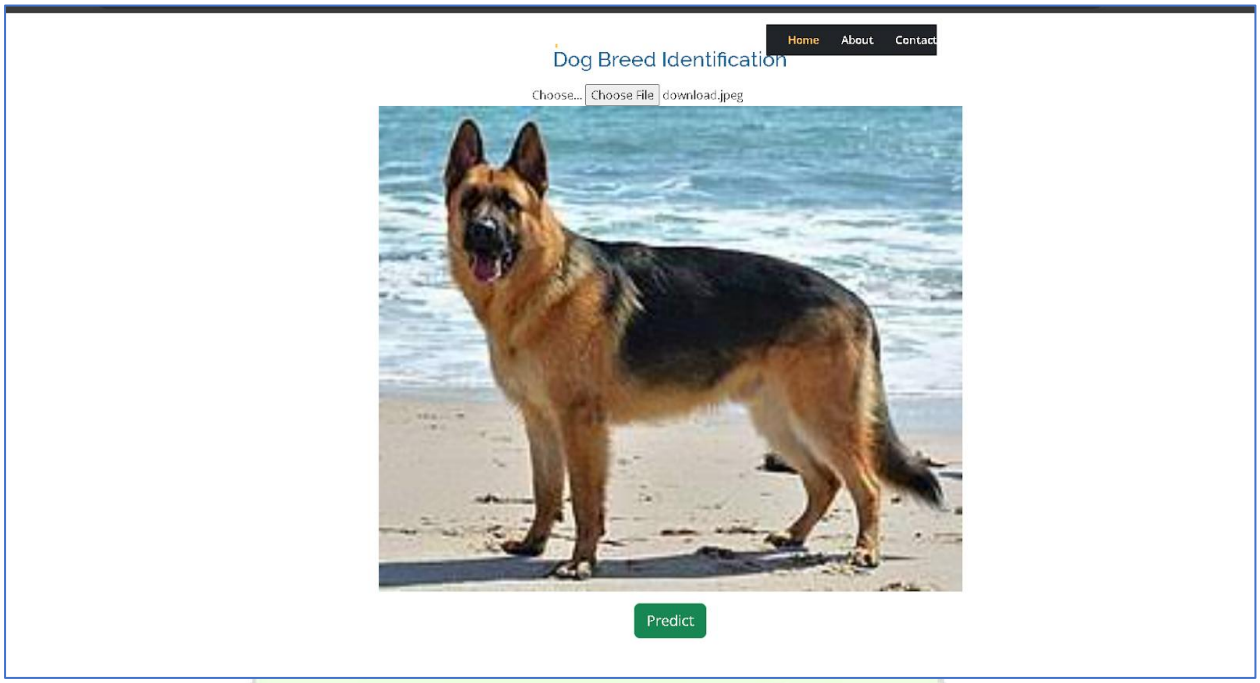
Functional and performance testing together ensure that the Dog Breed Identification system is accurate, reliable, and user-friendly. Functional testing verifies correct operation of each module, while performance testing ensures speed, efficiency, and stability. By conducting thorough testing, the project guarantees that the implemented solution meets both technical requirements and user expectations.

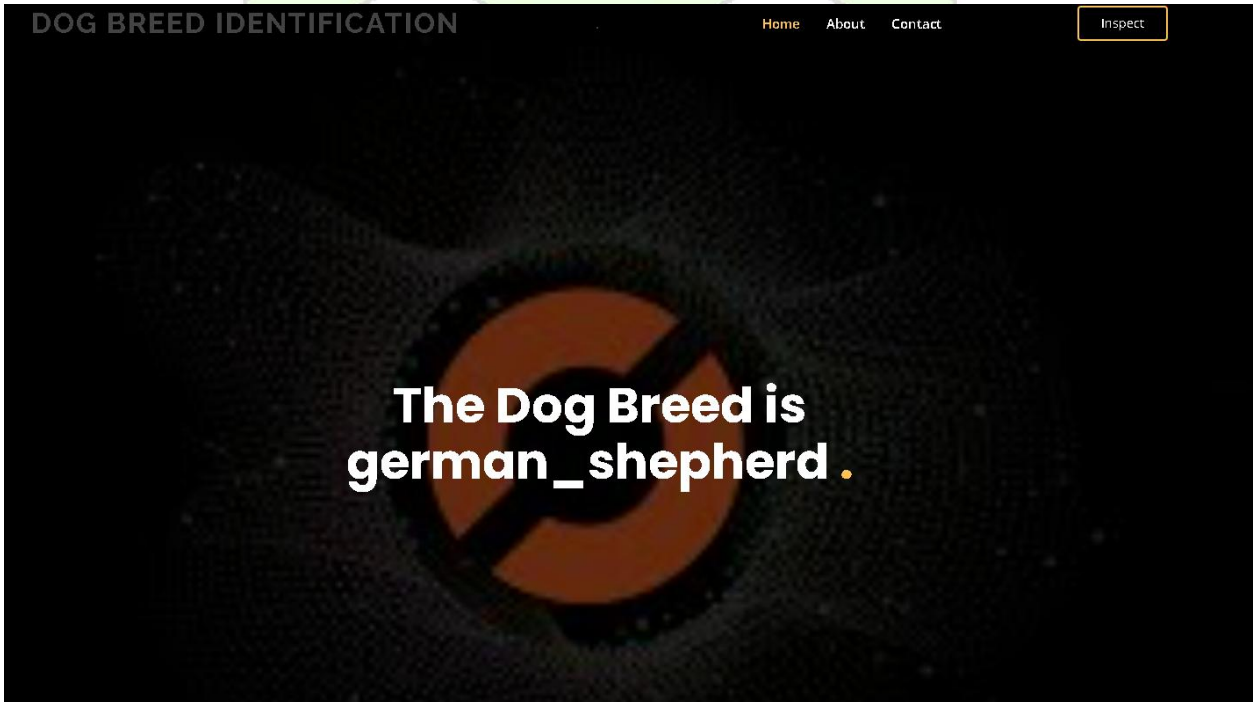
Chapter 8

Results

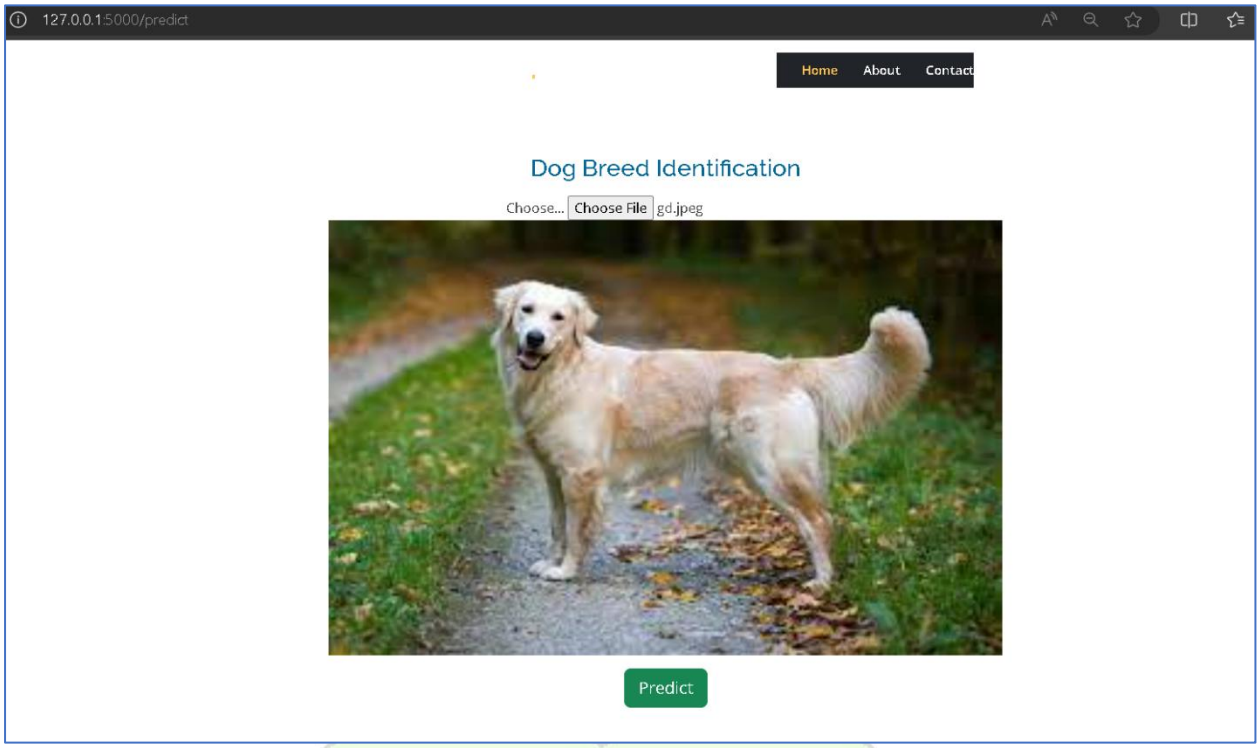
8.1 - Output Screens

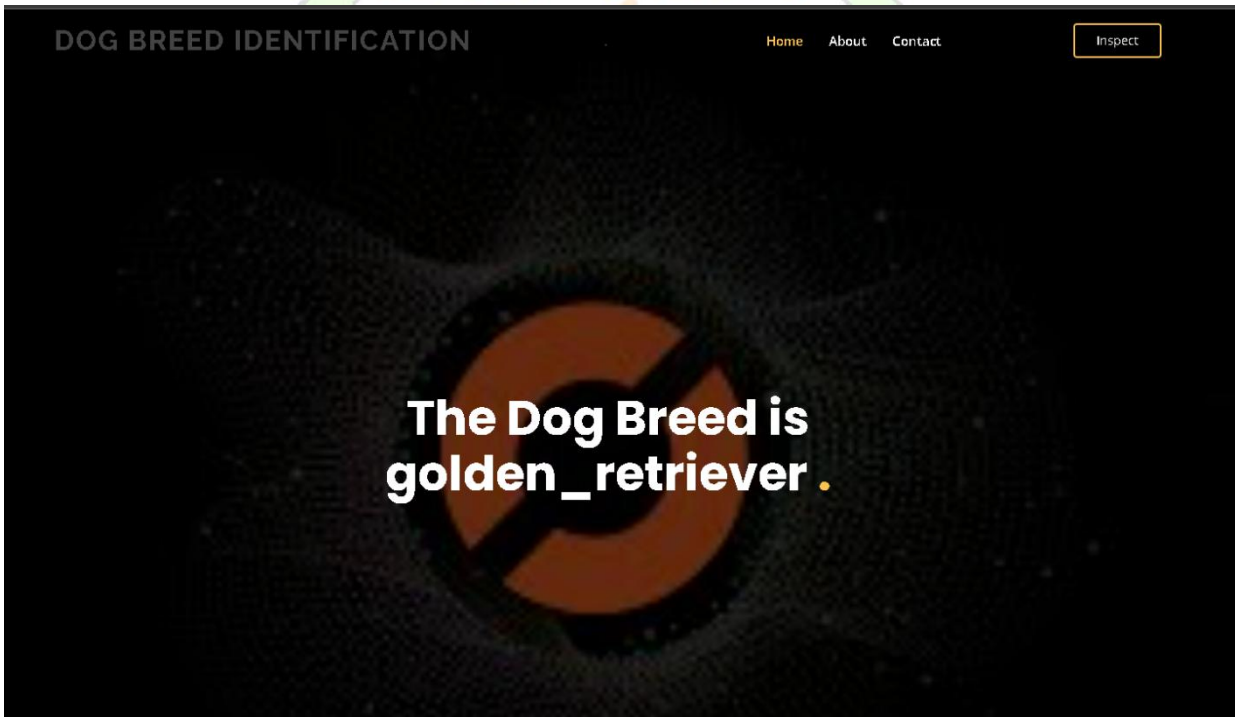
Output 1:





Output 2:


127.0.0.1:5000/predict
Home
About
Contact
Dog Breed Identification
Choose...
Choose File
gd.jpeg
Predict


DOG BREED IDENTIFICATION
Home
About
Contact
Inspect
The Dog Breed is
golden_retriever .

Chapter 9

Advantages and Disadvantages

9.1 – Advantages

• High classification accuracy due to the use of deep learning and transfer learning techniques.

• Reduced training time since pre-trained models are used instead of training from scratch.

• Efficient feature extraction using pre-trained models trained on large datasets such as ImageNet.

• Ability to handle large multi-class datasets with many dog breeds.

• Improved performance even with limited training data.

• User-friendly web interface for easy image upload and prediction.

• Fast prediction response suitable for real-time applications.

• Scalable architecture allowing addition of new dog breeds in the future.

• Reduced computational cost compared to building a deep learning model from scratch.

• Automatic feature learning without the need for manual feature engineering.

• Can be integrated into mobile or cloud-based applications.

• Supports practical applications in veterinary clinics, pet adoption platforms, and animal rescue systems.

• Provides consistent and reliable results compared to manual identification methods.

• Easy deployment using lightweight frameworks such as Flask.

• Flexible system design that allows further enhancements and optimization.

9.2 – Disadvantages

• Model accuracy highly depends on the quality and size of the training dataset.

• Similar-looking dog breeds may lead to misclassification.

• Performance may decrease if the uploaded image has poor lighting, low resolution, or unclear background.

• Transfer learning models may not perfectly adapt if the dataset differs significantly from datasets such as ImageNet.

• Requires computational resources (CPU/GPU) for training and efficient deployment.

• Large pre-trained models increase storage requirements.

• Prediction accuracy may reduce when dealing with mixed-breed dogs.

• Overfitting may occur if the model is not properly regularized or trained with sufficient data augmentation.

• Requires periodic retraining to improve performance with new data.

• Web application performance may slow down under heavy user traffic.

• Dependency on external deep learning frameworks and libraries.

• Initial setup and environment configuration can be complex for beginners.

Chapter 10

Conclusion

10.1 - Conclusion

The Dog Breed Identification using Transfer Learning project successfully demonstrates the practical application of deep learning techniques in solving real-world image classification problems. By leveraging transfer learning and pre-trained Convolutional Neural Networks (CNNs), the system effectively overcomes the challenges associated with manual breed identification and traditional machine learning approaches.

One of the major strengths of this project is the efficient use of pre-trained models originally trained on large-scale datasets such as ImageNet. Instead of building a model from scratch, transfer learning enables faster training, improved feature extraction, and higher classification accuracy. This approach significantly reduces computational cost while maintaining robust performance.

Overall, this project showcases the power of artificial intelligence in automating complex visual recognition tasks. It combines innovation, efficiency, and practicality, making it suitable for applications in veterinary services, pet adoption systems, animal rescue organizations, and educational research. The implementation of transfer learning proves to be a smart and resource-efficient solution for large-scale image classification problems.

In conclusion, the Dog Breed Identification system stands as a scalable, accurate, and user-centric AI solution that demonstrates both strong technical foundation and real-world applicability. It represents a successful integration of machine learning, computer vision, and web deployment technologies, paving the way for future advancements in intelligent image recognition systems.

Chapter 11

Future Scope

11.1 - Future Scope

• Increase the dataset size to improve model accuracy and generalization.

• Include more dog breeds to make the system more comprehensive.

• Fine-tune advanced deep learning architectures beyond models trained on ImageNet for better performance.

• Implement ensemble learning techniques to reduce misclassification.

• Develop real-time dog breed detection using live camera input.

• Create a mobile application version for better accessibility.

• Deploy the system on cloud platforms to handle large-scale user traffic.

• Add breed information details such as characteristics, lifespan, and health conditions.

• Introduce mixed-breed prediction with multiple probability outputs.

• Improve image preprocessing to handle low-quality or blurred images.

• Integrate the system with veterinary clinics and pet adoption platforms.

• Enhance user interface design for better user experience.

• Implement continuous model retraining using new datasets.

• Add multilingual support for global users.

• Expand the system to classify other animals in the future.

Chapter 12

Appendix

12.1 - Source Code

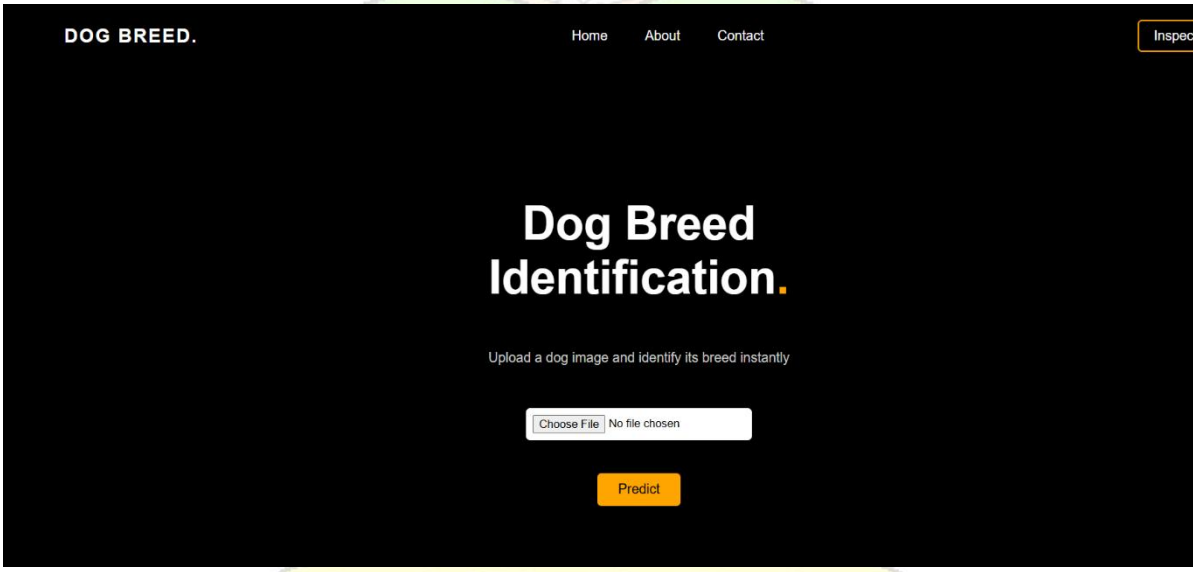
Index.html

```
<!DOCTYPE html>
<html>
<head>
    <title>Dog Breed Identification</title>
    <meta name="viewport" content="width=device-width, initial-
scale=1">

    <style>
        body {
            margin: 0;
            font-family: Arial, Helvetica, sans-serif;
            background: #000;
            color: white;
        }

        /* Navbar */
        .navbar {
            display: flex;
            justify-content: space-between;
            align-items: center;
            padding: 20px 80px;
        }

        .logo {
            font-size: 22px;
            font-weight: bold;
            letter-spacing: 2px;
        }

        .nav-links a {
            color: white;
            text-decoration: none;
            margin: 0 20px;
        }

        .nav-links a:hover {
            color: orange;
        }

        .inspect-btn {
            border: 2px solid orange;
```

```
    padding: 8px 18px;
    border-radius: 5px;
    color: white;
    text-decoration: none;
}

.inspect-btn:hover {
    background: orange;
    color: black;
}

/* Hero Section */
.hero {
    height: 85vh;
    display: flex;
    justify-content: center;
    align-items: center;
    flex-direction: column;
    text-align: center;
}

.hero h1 {
    font-size: 55px;
}

.hero span {
    color: orange;
}

.hero p {
    color: #ccc;
    margin-bottom: 30px;
}

/* Upload Box */
.upload-box {
    margin-top: 20px;
}

input[type="file"] {
    padding: 8px;
    background: white;
    color: black;
    border-radius: 5px;
}

.btn {
    margin-top: 20px;
    padding: 10px 25px;
```

```
            background: orange;
            border: none;
            border-radius: 5px;
            font-size: 16px;
            cursor: pointer;
        }

        .btn:hover {
            background: darkorange;
        }

        img {
            margin-top: 20px;
            max-width: 400px;
            border-radius: 10px;
        }
    </style>

    <!-- Image Preview Script -->
    <script>
        function previewImage(event) {
            const reader = new FileReader();
            reader.onload = function() {
                const output = document.getElementById('preview');
                output.src = reader.result;
                output.style.display = "block";
            }
            reader.readAsDataURL(event.target.files[0]);
        }
    </script>

</head>

<body>

<!-- Navbar -->
<div class="navbar">
    <div class="logo">DOG BREED.</div>

    <div class="nav-links">
        <a href="/">Home</a>
        <a href="#">About</a>
        <a href="#">Contact</a>
    </div>

    <a href="#" class="inspect-btn">Inspect</a>
</div>

<!-- Hero Section -->
```

```
<div class="hero">
    <h1>Dog Breed <br> Identification<span>.</span></h1>
    <p>Upload a dog image and identify its breed instantly</p>

    <!-- Upload Form -->
    <form action="{{ url_for('output') }}" method="POST"
enctype="multipart/form-data">
        <div class="upload-box">
            <input type="file" name="file" accept="image/*"
onchange="previewImage(event)" required>
        </div>

        <img id="preview" style="display:none;"/>

        <br>
        <button type="submit" class="btn">Predict</button>
    </form>
</div>

</body>
</html>
```

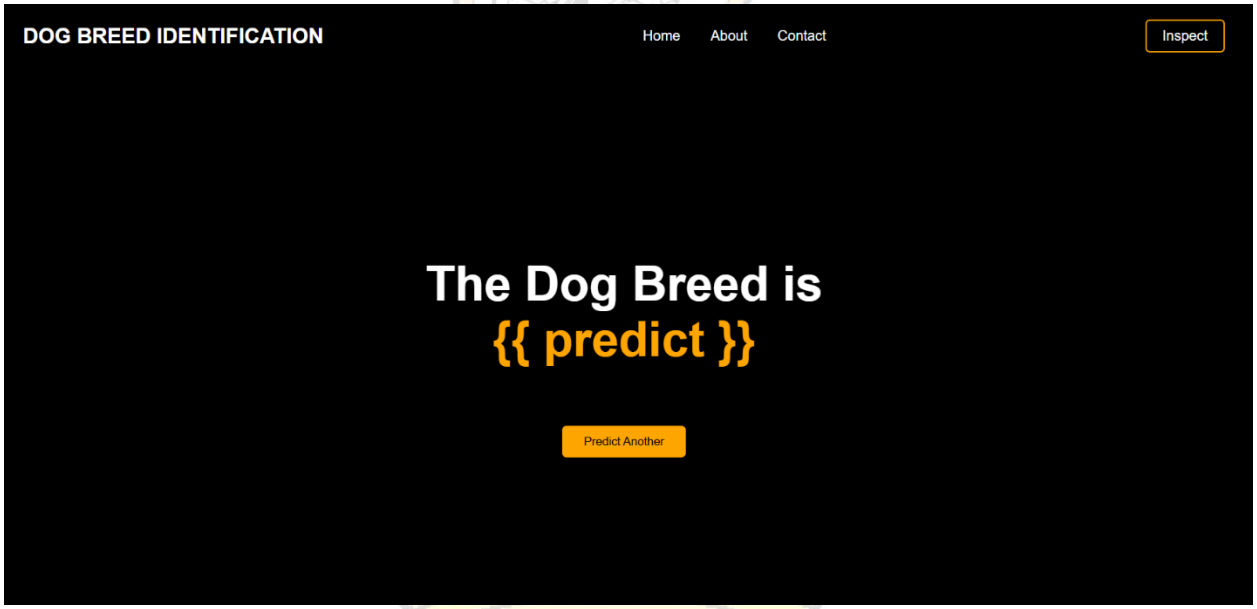


Predict.html

```
<!DOCTYPE html>
<html>
<head>
    <title>Dog Breed Identification</title>
    <meta name="viewport" content="width=device-width, initial-
scale=1">

    <style>
```

```
        body {
            margin: 0;
            font-family: Arial;
            background: #f4f4f4;
            text-align: center;
        }

        .navbar {
            background: black;
            color: white;
            padding: 20px;
        }

        .navbar a {
            color: white;
            margin: 0 20px;
            text-decoration: none;
        }

        .container {
            margin-top: 40px;
        }

        input[type="file"] {
            margin: 20px;
        }

        .btn {
            background: green;
            color: white;
            padding: 10px 25px;
            border: none;
            border-radius: 5px;
            cursor: pointer;
        }

        .btn:hover {
            background: darkgreen;
        }

        img {
            margin-top: 20px;
            max-width: 500px;
        }
    </style>
</head>

<body>
```

```
<div class="navbar">
    <a href="/">Home</a>
    <a href="/">About</a>
    <a href="/">Contact</a>
</div>

<div class="container">
    <h2>Dog Breed Identification</h2>

    <form action="{{ url_for('output') }}" method="POST"
enctype="multipart/form-data">
        <input type="file" name="file" required>
        <br>
        <button type="submit" class="btn">Predict</button>
    </form>
</div>

</body>
</html>
```

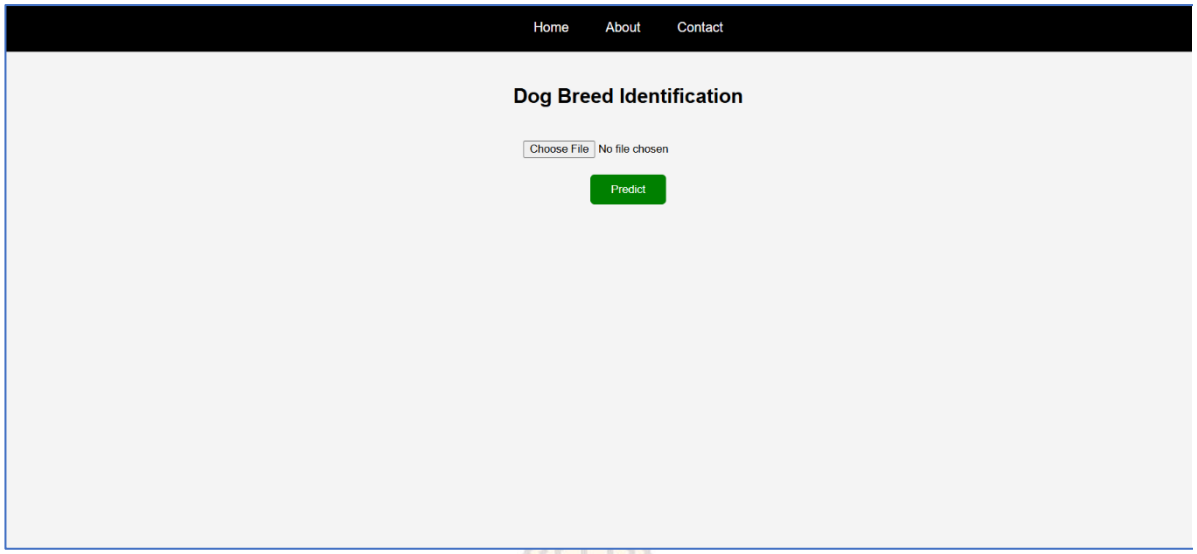


Output.html

```
<!DOCTYPE html>
<html>
<head>
    <title>Dog Breed Identification</title>
    <meta name="viewport" content="width=device-width, initial-
scale=1">

    <style>
```

```
body {
    margin: 0;
    font-family: Arial;
    background: #000;
    color: white;
    text-align: center;
}

.navbar {
    display: flex;
    justify-content: space-between;
    padding: 20px 80px;
    align-items: center;
}

.nav-links a {
    color: white;
    margin: 0 15px;
    text-decoration: none;
}

.nav-links a:hover {
    color: orange;
}

.inspect-btn {
    border: 2px solid orange;
    padding: 8px 18px;
    border-radius: 5px;
    color: white;
    text-decoration: none;
}

.inspect-btn:hover {
    background: orange;
    color: black;
}

.result {
    height: 80vh;
    display: flex;
    justify-content: center;
    align-items: center;
    flex-direction: column;
}

.result h1 {
    font-size: 55px;
}
```

```
        .result span {
            color: orange;
        }

        .back-btn {
            margin-top: 30px;
            padding: 10px 25px;
            background: orange;
            border: none;
            border-radius: 5px;
            cursor: pointer;
        }

        .back-btn:hover {
            background: darkorange;
        }
    </style>
</head>

<body>

<div class="navbar">
    <h2>DOG BREED IDENTIFICATION</h2>

    <div class="nav-links">
        <a href="/">Home</a>
        <a href="/">About</a>
        <a href="/">Contact</a>
    </div>

    <a href="{{ url_for('predict') }}" class="inspect-
btn">Inspect</a>
</div>

<div class="result">
    <h1>
        The Dog Breed is <br>
        <span>{{ predict }}</span>
    </h1>

    <a href="{{ url_for('predict') }}">
        <button class="back-btn">Predict Another</button>
    </a>
</div>

</body>
</html>
```



12.2 - Dataset Link

Download the Dataset –

https://www.kaggle.com/competitions/dog-breed-identification/data?select=train

QR Code

12.3 - GitHub and Project Demo Link

Project Demolink:

https://drive.google.com/file/d/1Btw21uWX1zrerOuQOOdrnHT98BEi5vaN/view?usp=drive_link

Project demo link QR code

Github link:

chitti4569/LTVIP2026TMIDS90703-Polisetty-Sanjay-Kumar

Github QR code: