### **Project Overview:**

This project focuses on developing an AI-based Python application to detect dents on a car using images. The solution will be built using machine learning models, deployed on AWS, and integrated into a Jenkins CI/CD pipeline for continuous integration and delivery.

### **Technologies Used:**

* **Python** (Flask/Django for web app)
* **TensorFlow/Keras** (for AI model)
* **AWS S3** (for storing images)
* **AWS Lambda** (for event-driven serverless inference)
* **AWS EC2** (for model training and web app deployment)
* **Jenkins** (for CI/CD automation)
* **GitHub** (for version control)

### **1. Project Folder Structure**

The GitHub project will be organized as follows:

bash

Copy code

/dent-detection-app

├── /docs # Documentation folder

│ ├── Architecture\_Diagram.png # High-level architecture diagram

│ ├── Project\_Design.md # Design and project plan documentation

│ └── Security\_Strategy.md # Security documentation

├── /src # Source code of the application

│ ├── /app # Flask/Django app code

│ │ ├── \_\_init\_\_.py

│ │ ├── routes.py

│ │ ├── model.py # Model loading and prediction code

│ │ └── utils.py # Image processing utilities

│ ├── /model # Trained model files

│ │ └── dent\_model.h5 # Model file (TensorFlow/Keras format)

│ └── requirements.txt # Python dependencies

├── /infrastructure # Infrastructure as code (IaC)

│ ├── /terraform # Terraform scripts for AWS infrastructure setup

│ │ ├── main.tf

│ │ ├── variables.tf

│ │ └── outputs.tf

│ └── /cloudformation # CloudFormation templates if needed

├── /jenkins # Jenkins pipeline files

│ ├── Jenkinsfile # Jenkins pipeline script for CI/CD

│ └── scripts/ # Custom scripts (e.g., deployment scripts)

├── /tests # Unit and integration tests

│ ├── test\_app.py

│ └── test\_model.py

├── README.md # Overview of the project

└── LICENSE # Project license file

### **2. AI Model for Dent Detection**

#### **Model Training:**

* **Dataset**: The first step is collecting a dataset of images of cars with and without dents. You can use a publicly available dataset or create one by scraping images from the web or using tools like Open Images Dataset or Kaggle.
* **Model Architecture**:
  + **Pretrained Model**: Use a pretrained model like **ResNet50** or **VGG16** for transfer learning. You can fine-tune the model to detect dents in car images.
  + **Training**: The model can be trained using TensorFlow or Keras in Python. The output layer should be binary: 0 (no dent) and 1 (dent).

Example model code:

python

Copy code

import tensorflow as tf

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense, Flatten, Conv2D, MaxPooling2D

from tensorflow.keras.applications import ResNet50

def create\_model(input\_shape):

base\_model = ResNet50(weights="imagenet", include\_top=False, input\_shape=input\_shape)

base\_model.trainable = False

model = Sequential([

base\_model,

Flatten(),

Dense(128, activation='relu'),

Dense(1, activation='sigmoid')

])

model.compile(optimizer='adam', loss='binary\_crossentropy', metrics=['accuracy'])

return model

# Load and preprocess data, then train the model

# model.fit(train\_data, train\_labels)

After training, save the model as dent\_model.h5 using:

python

Copy code

model.save('dent\_model.h5')

#### **Model Inference in Flask/Django App:**

Once the model is trained, you need to integrate it into the web application (Flask/Django) for inference.

python

Copy code

from tensorflow.keras.models import load\_model

from tensorflow.keras.preprocessing import image

import numpy as np

import os

# Load the trained model

model = load\_model('model/dent\_model.h5')

def predict\_dent(image\_path):

img = image.load\_img(image\_path, target\_size=(224, 224))

img\_array = image.img\_to\_array(img) / 255.0

img\_array = np.expand\_dims(img\_array, axis=0)

prediction = model.predict(img\_array)

return "Dent detected" if prediction[0] > 0.5 else "No dent detected"

### **3. Deploying the Application on AWS**

#### **AWS Infrastructure Setup:**

Use **Terraform** or **CloudFormation** to automate the creation of AWS resources:

* **S3**: To store car images uploaded by users.
* **EC2**: To host the Flask/Django app (for API endpoints).
* **Lambda**: For serverless inference (optional for scalability).
* **RDS**: For user management and other relational data.

#### **Example Terraform Script (main.tf):**

hcl

Copy code

provider "aws" {

region = "us-east-1"

}

resource "aws\_s3\_bucket" "car\_images" {

bucket = "dent-detection-images"

}

resource "aws\_instance" "app\_server" {

ami = "ami-xxxxxxxxxxxxx"

instance\_type = "t2.micro"

tags = {

Name = "DentDetectionApp"

}

user\_data = <<-EOF

#!/bin/bash

cd /var/www/dent-detection-app

python app.py

EOF

}

# More resources (Lambda, RDS, etc.)

### **4. Jenkins CI/CD Pipeline**

#### **Jenkinsfile (Pipeline Script)**

Create a Jenkins pipeline to automate the build, test, and deployment of your AI application.

groovy

Copy code

pipeline {

agent any

environment {

AWS\_REGION = 'us-east-1'

}

stages {

stage('Checkout Code') {

steps {

git 'https://github.com/your-username/dent-detection-app.git'

}

}

stage('Install Dependencies') {

steps {

script {

sh 'pip install -r requirements.txt'

}

}

}

stage('Run Tests') {

steps {

script {

sh 'pytest tests/'

}

}

}

stage('Build Docker Image') {

steps {

script {

sh 'docker build -t dent-detection .'

}

}

}

stage('Deploy to AWS') {

steps {

script {

sh './deploy.sh' // Use a custom script to deploy on EC2 or Lambda

}

}

}

stage('Post-Deployment Tests') {

steps {

script {

sh './health\_check.sh' // Test the health of the app after deployment

}

}

}

}

post {

always {

echo 'Cleaning up resources'

}

success {

echo 'Deployment successful'

}

failure {

echo 'Deployment failed'

}

}

}

### **5. CI/CD Integration**

#### **GitHub Integration:**

* Push the code to GitHub and link it to Jenkins for continuous integration.
* Jenkins will pull the latest changes from the GitHub repository for every commit and trigger the pipeline automatically.

### **6. Security Considerations**

* **Data Protection**: Ensure images and sensitive data are encrypted at rest (S3 encryption) and in transit (SSL/TLS).
* **Access Control**: Use IAM roles to control access to AWS resources.
* **Input Validation**: Validate user inputs to ensure only valid images are uploaded.

### **7. README.md Example**

markdown

Copy code

# AI-Based Car Dent Detection

This project implements an AI-based car dent detection application using Python and deep learning, hosted on AWS with a CI/CD pipeline using Jenkins.

## Features

- Detects dents in car images using a trained deep learning model (TensorFlow/Keras).

- Hosted on AWS with EC2 and Lambda (optional for scalability).

- Continuous integration and deployment using Jenkins.

## Setup Instructions

1. Clone the repository:

```bash

git clone https://github.com/your-username/dent-detection-app.git

Install dependencies:  
bash  
Copy code  
pip install -r requirements.txt

Run the Flask app:  
bash  
Copy code  
python app.py

1. Follow the docs folder for deployment on AWS using Terraform or CloudFormation.

## **Deployment**

* The application is deployed using Jenkins CI/CD pipeline.
* Refer to Jenkinsfile for the complete pipeline configuration.

yaml

Copy code

---

### \*\*8. Deploying on GitHub\*\*

1. Create a new repository on GitHub.

2. Initialize a Git repository locally:

```bash

git init

Add your files:  
bash  
Copy code  
git add .

git commit -m "Initial commit"

Push the code to your GitHub repository:  
bash  
Copy code  
git remote add origin https://github.com/your-username/dent-detection-app.git

git push -u origin master

This approach sets up the AI-based dent detection application on AWS with Jenkins CI/CD for easy deployment and scaling, making it a robust solution for your project.