Project Report

Project Title:

AI-Based Vehicle Identification and

Parking Management System for Toyota Dealership

Group Members:

Mohtishim Fareed

Aqib Syed

Saad Amjad

Submitted To:

Dr.Syed M.Hamedoon

Assistant Professor

Project Description:

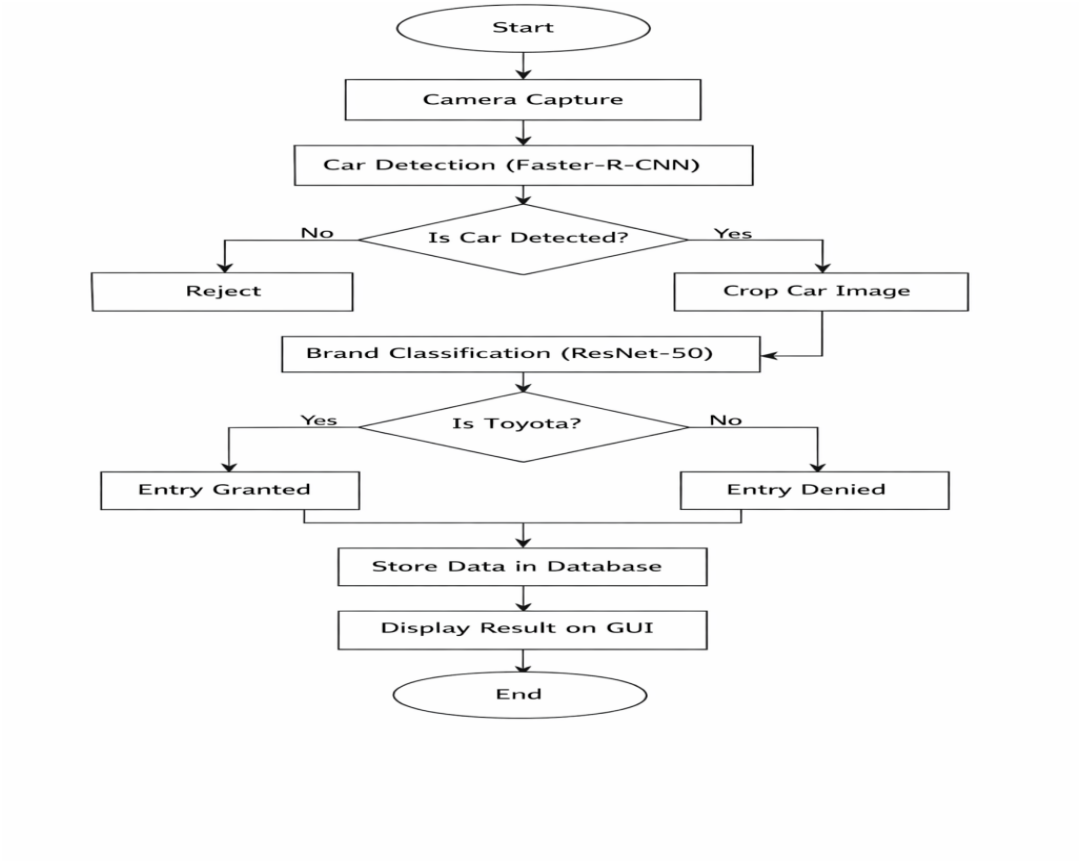
This project presents an AI-based vehicle identification and parking management system designed specifically for a Toyota dealership. The system uses computer vision and deep learning techniques to automatically detect vehicles and identify whether the vehicle belongs to the Toyota brand**.**

A live camera captures vehicle images at the parking entry. The system first detects the presence of a vehicle using a Faster R-CNN object detection model. After detection, a deep learning classifier (ResNet-50) is used to identify the vehicle brand. If the detected vehicle is Toyota, entry is granted; otherwise, access is denied.

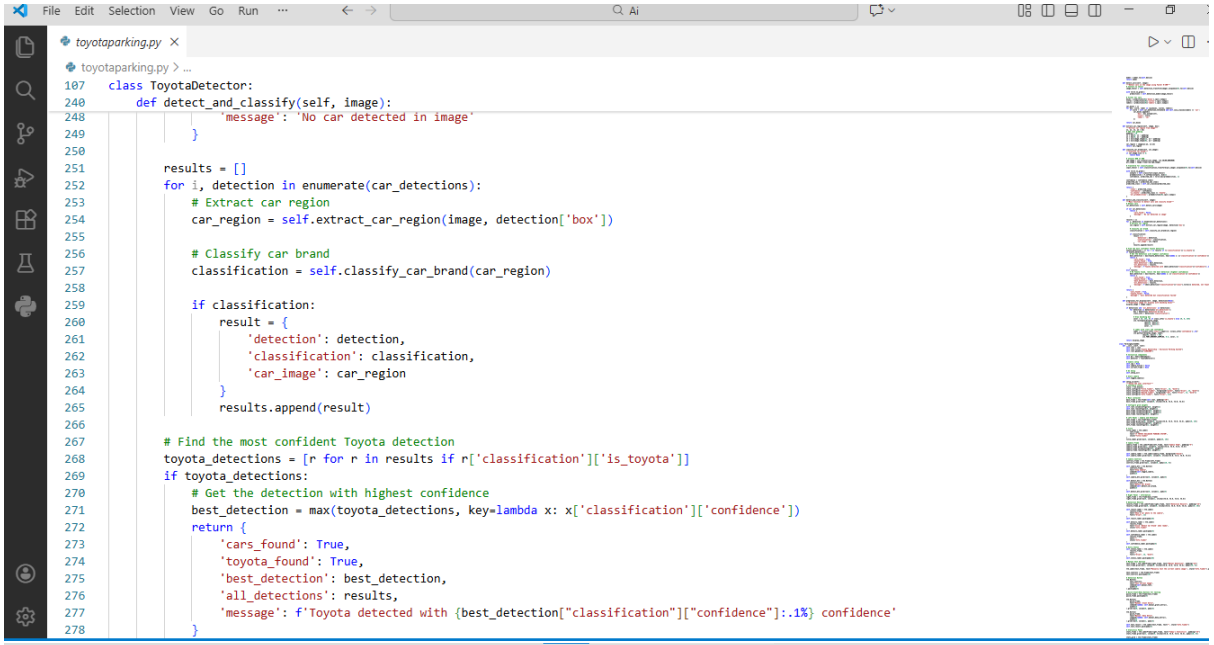
All entry and exit records are stored in a SQLite database, including timestamps, confidence scores, and access status. A Tkinter-based graphical user interface (GUI) provides real-time camera feed, detection results, manual override options, and daily parking statistics.

This system improves security, automation, and efficiency by eliminating manual vehicle verification and ensuring that only Toyota vehicles are allowed inside the dealership parking area.

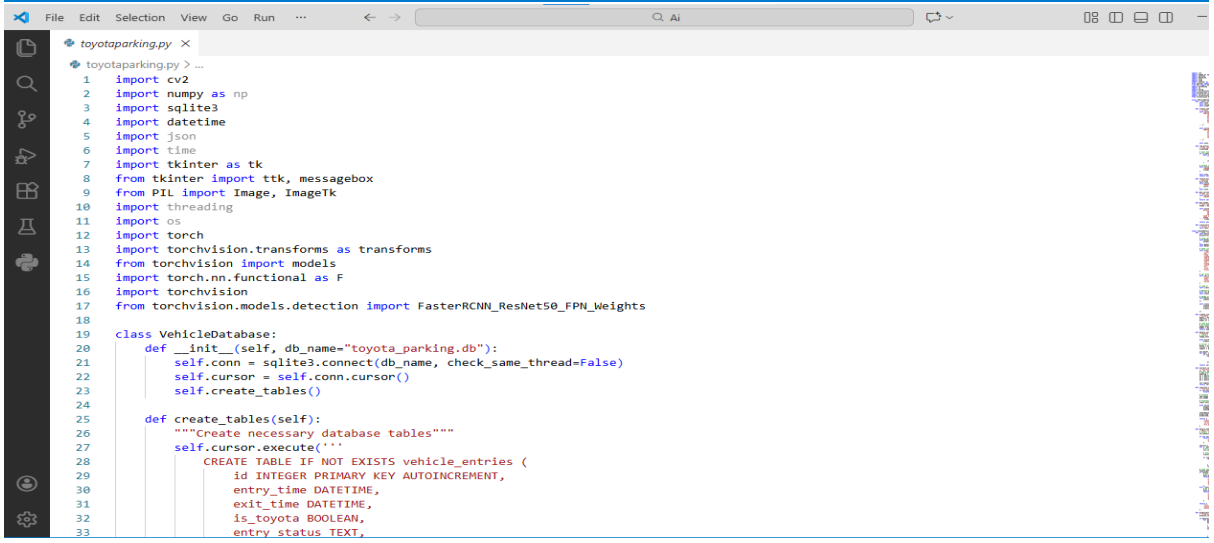
Project Flow Diagram:



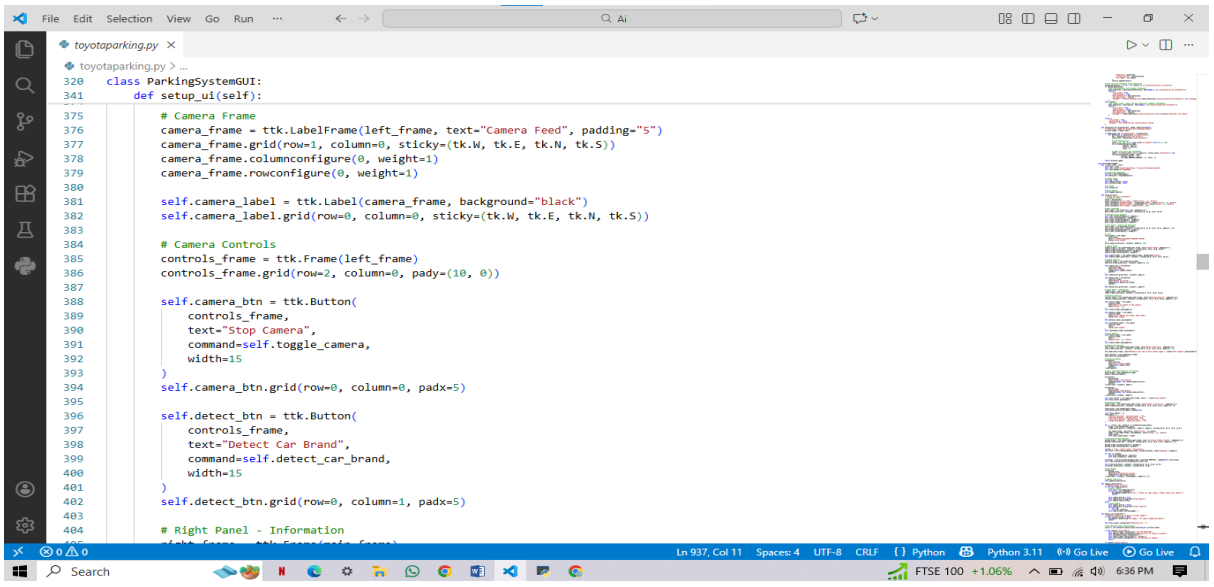
Project Code: ScreenShots



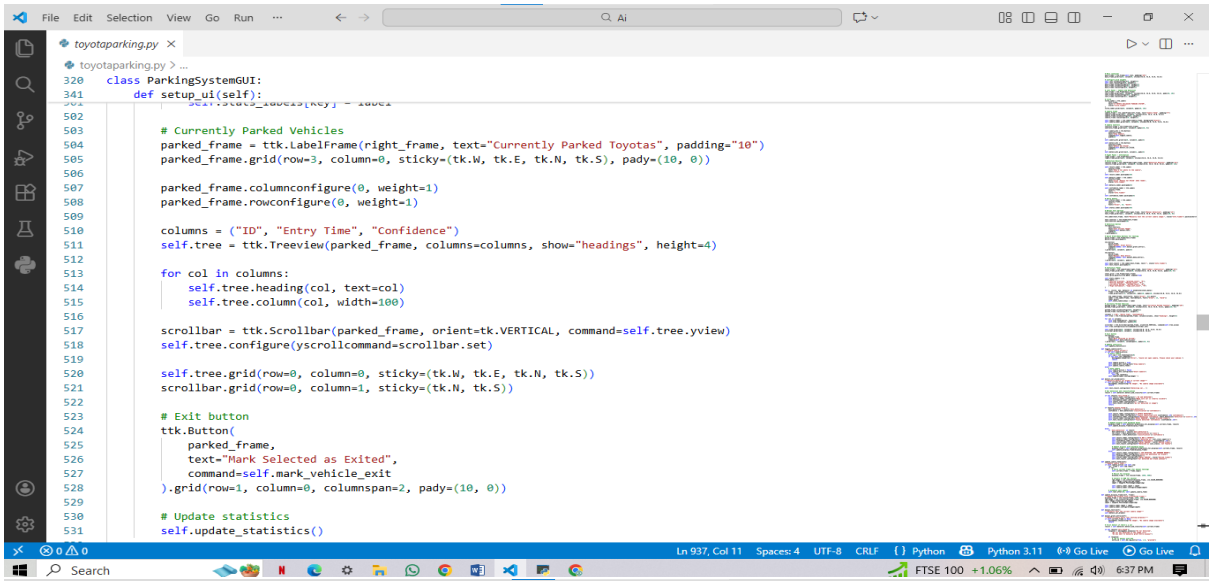
```
class ToyotaDetector:
    def detect_and_classify(self, image):
                'message': 'No car detected in image'
            }

        results = []
        for i, detection in enumerate(car_detections):
            # Extract car region
            car_region = self.extract_car_region(image, detection['box'])

            # Classify car brand
            classification = self.classify_car_brand(car_region)

            if classification:
                result = {
                    'detection': detection,
                    'classification': classification,
                    'car_image': car_region
                }
                results.append(result)

        # Find the most confident Toyota detection
        toyota_detections = [r for r in results if r['classification']['is_toyota']]
        if toyota_detections:
            # Get the detection with highest confidence
            best_detection = max(toyota_detections, key=lambda x: x['classification']['confidence'])
            return {
                'cars_found': True,
                'toyota_found': True,
                'best_detection': best_detection,
                'all_detections': results,
                'message': f'Toyota detected with {best_detection["classification"]["confidence"]:.1%} confidence'
            }
```



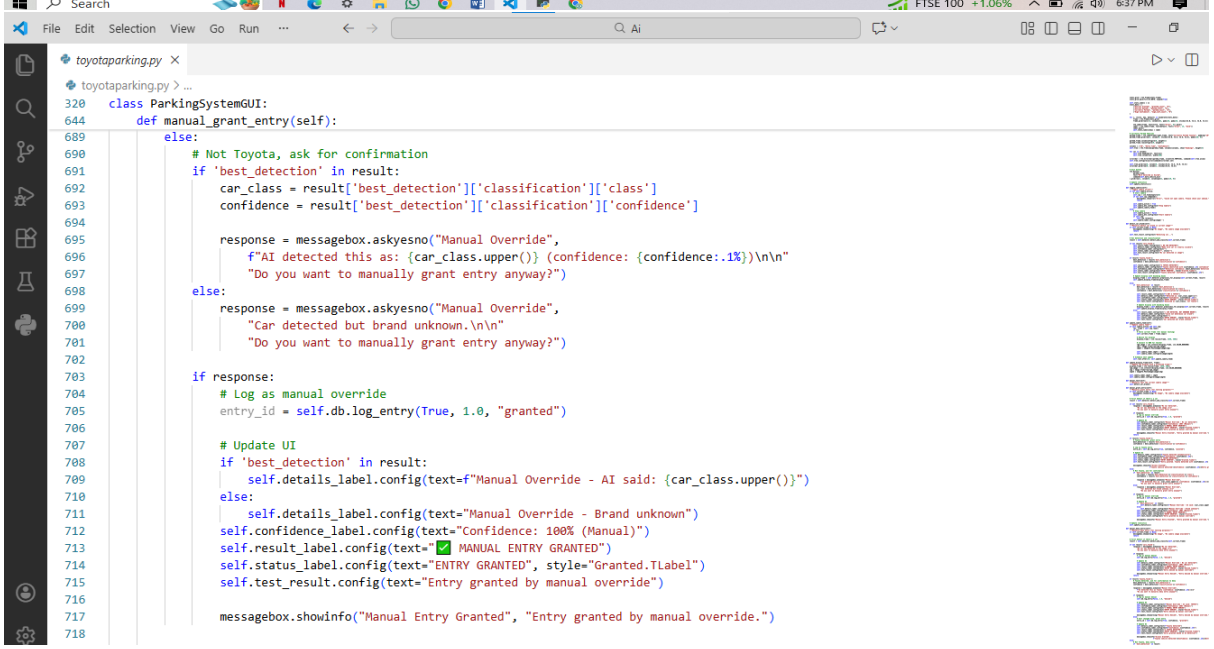
```
import cv2
import numpy as np
import sqlite3
import datetime
import json
import time
import tkinter as tk
from tkinter import ttk, messagebox
from PIL import Image, ImageTk
import threading
import os
import torch
import torchvision.transforms as transforms
from torchvision import models
import torch.nn.functional as F
import torchvision
from torchvision.models.detection import FasterRCNN_ResNet50_FPN_Weights

class VehicleDatabase:
    def __init__(self, db_name="toyota_parking.db"):
        self.conn = sqlite3.connect(db_name, check_same_thread=False)
        self.cursor = self.conn.cursor()
        self.create_tables()

    def create_tables(self):
        """Create necessary database tables"""
        self.cursor.execute('''
            CREATE TABLE IF NOT EXISTS vehicle_entries (
                id INTEGER PRIMARY KEY AUTOINCREMENT,
                entry_time DATETIME,
                exit_time DATETIME,
                is_toyota BOOLEAN,
                entry_status TEXT,
```



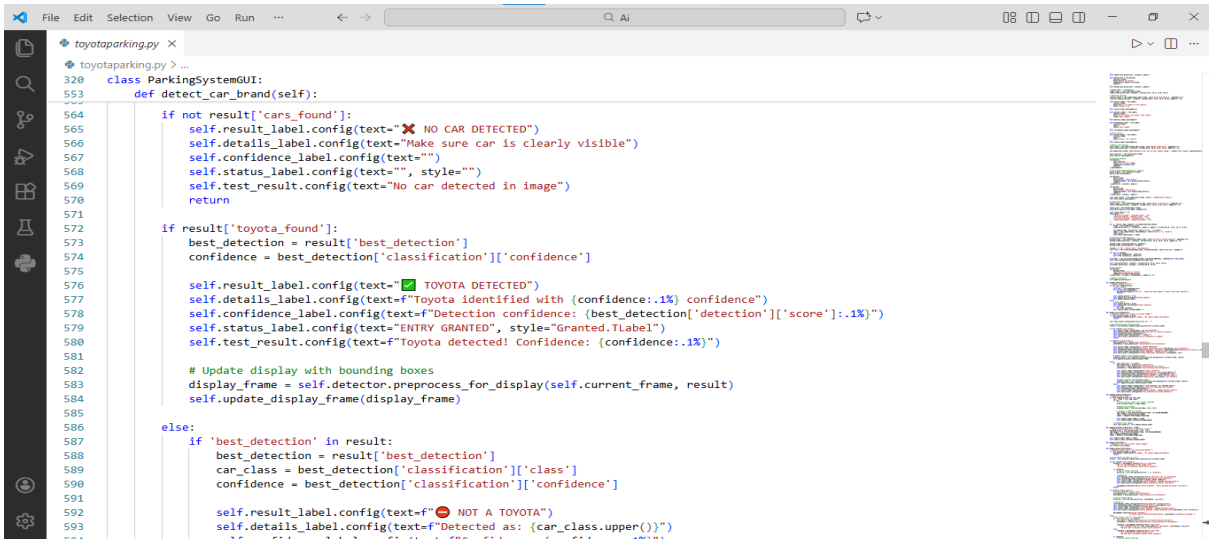
```
class ParkingSystemGUI:
    def setup_ui(self):
        # Camera Frame
        camera_frame = ttk.LabelFrame(left_frame, text="Camera Feed", padding="5")
        camera_frame.grid(row=1, column=0, sticky=(tk.W, tk.E, tk.N, tk.S))
        camera_frame.columnconfigure(0, weight=1)
        camera_frame.rowconfigure(0, weight=1)

        self.camera_label = ttk.Label(camera_frame, background="black")
        self.camera_label.grid(row=0, column=0, sticky=(tk.W, tk.E, tk.N, tk.S))

        # Camera Controls
        controls_frame = ttk.Frame(left_frame)
        controls_frame.grid(row=2, column=0, pady=(10, 0))

        self.camera_btn = ttk.Button(
            controls_frame,
            text="Stop Camera",
            command=self.toggle_camera,
            width=15
        )
        self.camera_btn.grid(row=0, column=0, padx=5)

        self.detect_btn = ttk.Button(
            controls_frame,
            text="Detect Car Brand",
            command=self.detect_car_brand,
            width=15
        )
        self.detect_btn.grid(row=0, column=1, padx=5)

        # Right Panel - Information
```



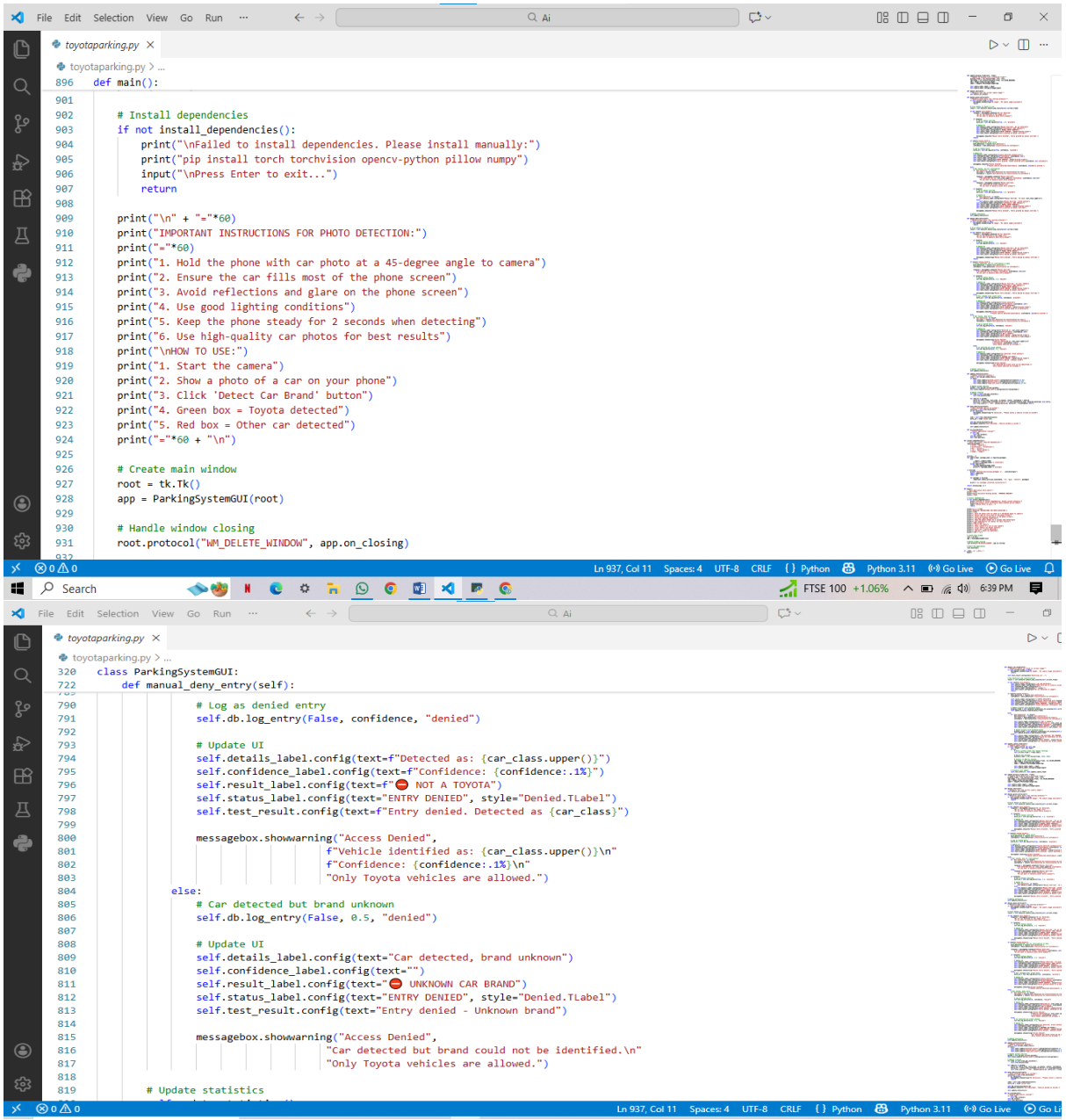
```
toyotaparking.py > ...
320  class ParkingSystemGUI:
341      def setup_ui(self):
502
503          # Currently Parked Vehicles
504          parked_frame = ttk.LabelFrame(right_frame, text="Currently Parked Toyotas", padding="10")
505          parked_frame.grid(row=3, column=0, sticky=(tk.W, tk.E, tk.N, tk.S), pady=(10, 0))
506
507          parked_frame.columnconfigure(0, weight=1)
508          parked_frame.rowconfigure(0, weight=1)
509
510          columns = ("ID", "Entry Time", "Confidence")
511          self.tree = ttk.Treeview(parked_frame, columns=columns, show="headings", height=4)
512
513          for col in columns:
514              self.tree.heading(col, text=col)
515              self.tree.column(col, width=100)
516
517          scrollbar = ttk.Scrollbar(parked_frame, orient=tk.VERTICAL, command=self.tree.yview)
518          self.tree.configure(yscrollcommand=scrollbar.set)
519
520          self.tree.grid(row=0, column=0, sticky=(tk.W, tk.E, tk.N, tk.S))
521          scrollbar.grid(row=0, column=1, sticky=(tk.N, tk.S))
522
523          # Exit button
524          ttk.Button(
525              parked_frame,
526              text="Mark Selected as Exited",
527              command=self.mark_vehicle_exit
528          ).grid(row=1, column=0, columnspan=2, pady=(10, 0))
529
530          # Update statistics
531          self.update_statistics()
```



```
toyotaparking.py > ...
320  class ParkingSystemGUI:
644      def manual_grant_entry(self):
689          else:
690              # Not Toyota, ask for confirmation
691              if 'best_detection' in result:
692                  car_class = result['best_detection']['classification']['class']
693                  confidence = result['best_detection']['classification']['confidence']
694
695                  response = messagebox.askyesno("Manual Override",
696                      f"AI detected this as: {car_class.upper()} (confidence: {confidence:.1%})\n\n"
697                      "Do you want to manually grant entry anyway?")
698              else:
699                  response = messagebox.askyesno("Manual Override",
700                      "Car detected but brand unknown.\n\n"
701                      "Do you want to manually grant entry anyway?")
702
703              if response:
704                  # Log as manual override
705                  entry_id = self.db.log_entry(True, 1.0, "granted")
706
707                  # Update UI
708                  if 'best_detection' in result:
709                      self.details_label.config(text=f"Manual Override - AI said: {car_class.upper()}")
710                  else:
711                      self.details_label.config(text="Manual Override - Brand unknown")
712                  self.confidence_label.config(text="Confidence: 100% (Manual)")
713                  self.result_label.config(text="✅ MANUAL ENTRY GRANTED")
714                  self.status_label.config(text="ENTRY GRANTED", style="Granted.TLabel")
715                  self.test_result.config(text="Entry granted by manual override")
716
717                  messagebox.showinfo("Manual Entry Granted", "Entry granted by manual override.")
718
```



```
toyotaparking.py > ...
320  class ParkingSystemGUI:
553      def detect_car_brand(self):
564          if not result['cars_found']:
565              self.result_label.config(text="❌ NO CAR DETECTED")
566              self.details_label.config(text="Make sure car is clearly visible")
567              self.confidence_label.config(text="")
568              self.status_label.config(text="", style="")
569              self.test_result.config(text="No car detected in image")
570              return
571
572          if result['toyota_found']:
573              best_detection = result['best_detection']
574              confidence = best_detection['classification']['confidence']
575
576              self.result_label.config(text="✅ TOYOTA DETECTED")
577              self.details_label.config(text=f"Toyota identified with {confidence:.1%} confidence")
578              self.confidence_label.config(text=f"Detection confidence: {best_detection['detection']['score']:.1%}")
579              self.status_label.config(text="ENTRY GRANTED", style="Granted.TLabel")
580              self.test_result.config(text=f"Toyota detected! Confidence: {confidence:.1%}")
581
582              # Update display with bounding boxes
583              display_frame = self.detector.preprocess_for_display(self.current_frame, result)
584              self.update_display_frame(display_frame)
585
586          else:
587              if 'best_detection' in result:
588                  best_detection = result['best_detection']
589                  car_class = best_detection['classification']['class']
590                  confidence = best_detection['classification']['confidence']
591
592                  self.result_label.config(text=f"⛔ NOT A TOYOTA")
593                  self.details_label.config(text=f"Detected as: {car_class.upper()}")
```

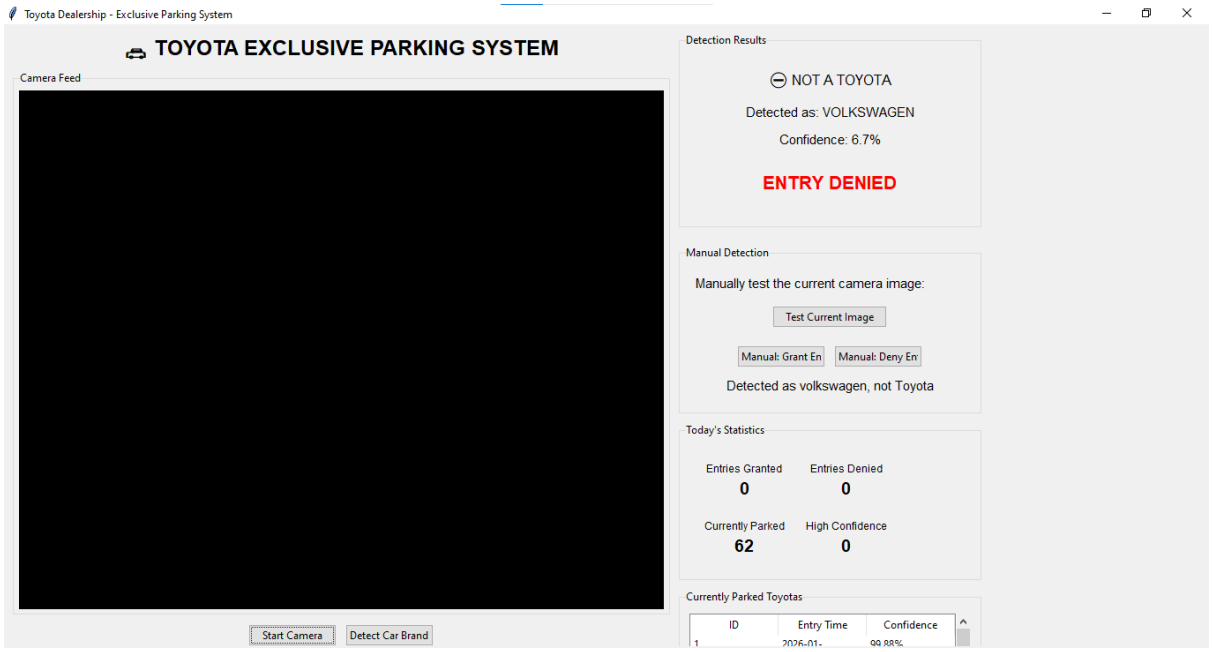


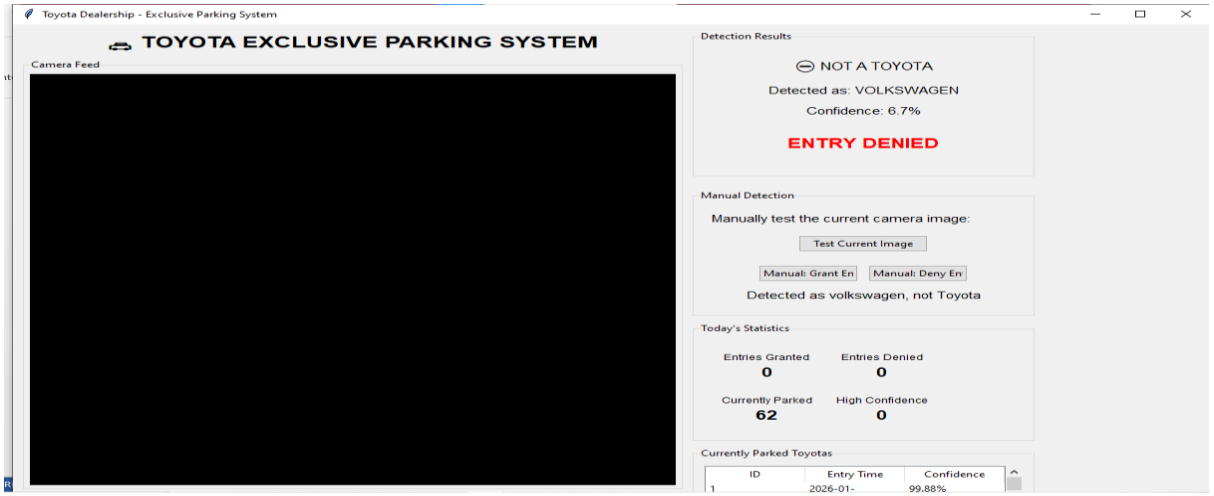
```
toyotaparking.py
toyotaparking.py > ...
896  def main():
901
902      # Install dependencies
903      if not install_dependencies():
904          print("\nFailed to install dependencies. Please install manually:")
905          print("pip install torch torchvision opencv-python pillow numpy")
906          input("\nPress Enter to exit...")
907          return
908
909      print("\n" + "="*60)
910      print("IMPORTANT INSTRUCTIONS FOR PHOTO DETECTION:")
911      print("="*60)
912      print("1. Hold the phone with car photo at a 45-degree angle to camera")
913      print("2. Ensure the car fills most of the phone screen")
914      print("3. Avoid reflections and glare on the phone screen")
915      print("4. Use good lighting conditions")
916      print("5. Keep the phone steady for 2 seconds when detecting")
917      print("6. Use high-quality car photos for best results")
918      print("\nHOW TO USE:")
919      print("1. Start the camera")
920      print("2. Show a photo of a car on your phone")
921      print("3. Click 'Detect Car Brand' button")
922      print("4. Green box = Toyota detected")
923      print("5. Red box = Other car detected")
924      print("="*60 + "\n")
925
926      # Create main window
927      root = tk.Tk()
928      app = ParkingSystemGUI(root)
929
930      # Handle window closing
931      root.protocol("WM_DELETE_WINDOW", app.on_closing)
```

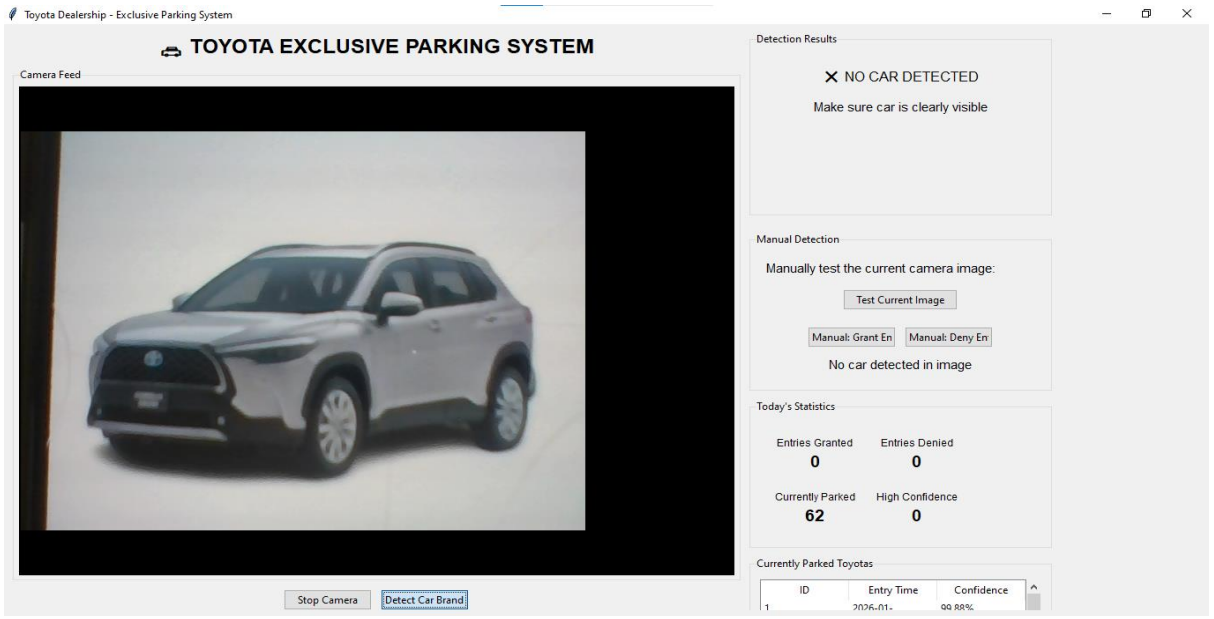


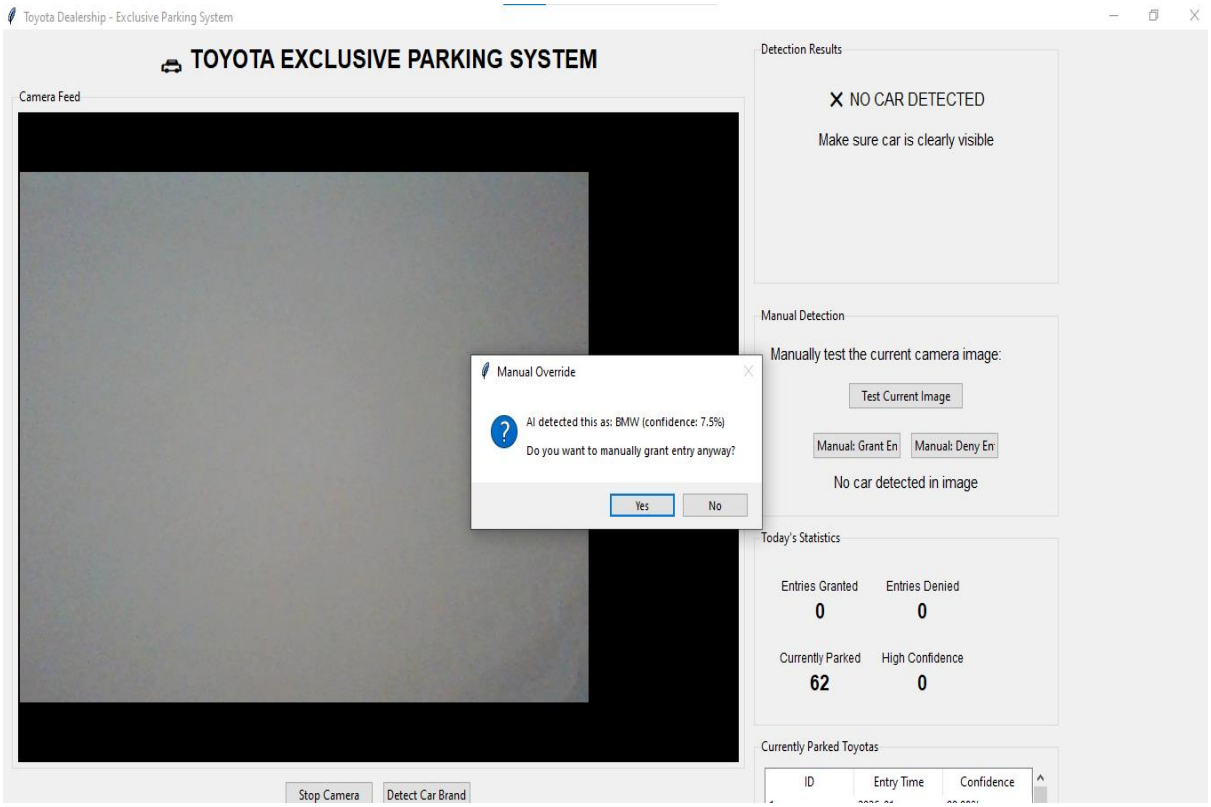
```
toyotaparking.py
toyotaparking.py > ...
320  class ParkingSystemGUI:
722      def manual_deny_entry(self):
790                  # Log as denied entry
791                  self.db.log_entry(False, confidence, "denied")
792
793                  # Update UI
794                  self.details_label.config(text=f"Detected as: {car_class.upper()}")
795                  self.confidence_label.config(text=f"Confidence: {confidence:.1%}")
796                  self.result_label.config(text=f"⛔ NOT A TOYOTA")
797                  self.status_label.config(text="ENTRY DENIED", style="Denied.TLabel")
798                  self.test_result.config(text=f"Entry denied. Detected as {car_class}")
799
800                  messagebox.showwarning("Access Denied",
801                                         f"Vehicle identified as: {car_class.upper()}\n"
802                                         f"Confidence: {confidence:.1%}\n"
803                                         "Only Toyota vehicles are allowed.")
804              else:
805                  # Car detected but brand unknown
806                  self.db.log_entry(False, 0.5, "denied")
807
808                  # Update UI
809                  self.details_label.config(text="Car detected, brand unknown")
810                  self.confidence_label.config(text="")
811                  self.result_label.config(text="⛔ UNKNOWN CAR BRAND")
812                  self.status_label.config(text="ENTRY DENIED", style="Denied.TLabel")
813                  self.test_result.config(text="Entry denied - Unknown brand")
814
815                  messagebox.showwarning("Access Denied",
816                                         "Car detected but brand could not be identified.\n"
817                                         "Only Toyota vehicles are allowed.")
818
819          # Update statistics
```

Output Screenshots:


Toyota Dealership - Exclusive Parking System
TOYOTA EXCLUSIVE PARKING SYSTEM
Camera Feed
Start Camera
Detect Car Brand
Detection Results
NOT A TOYOTA
Detected as: VOLKSWAGEN
Confidence: 6.7%
ENTRY DENIED
Manual Detection
Manually test the current camera image:
Test Current Image
Manual: Grant En
Manual: Deny En
Detected as volkswagen, not Toyota
Today's Statistics
Entries Granted
0
Entries Denied
0
Currently Parked
62
High Confidence
0
Currently Parked Toyotas
ID
Entry Time
Confidence
1
2026-01-
99.88%


Toyota Dealership - Exclusive Parking System
TOYOTA EXCLUSIVE PARKING SYSTEM
Camera Feed
Detection Results
NOT A TOYOTA
Detected as: VOLKSWAGEN
Confidence: 6.7%
ENTRY DENIED
Manual Detection
Manually test the current camera image:
Test Current Image
Manual: Grant En
Manual: Deny En
Detected as volkswagen, not Toyota
Today's Statistics
Entries Granted
0
Entries Denied
0
Currently Parked
62
High Confidence
0
Currently Parked Toyotas
ID
Entry Time
Confidence
1
2026-01-
99.88%


Toyota Dealership - Exclusive Parking System
TOYOTA EXCLUSIVE PARKING SYSTEM
Camera Feed
Stop Camera
Detect Car Brand
Detection Results
NO CAR DETECTED
Make sure car is clearly visible
Manual Detection
Manually test the current camera image:
Test Current Image
Manual: Grant En
Manual: Deny En
No car detected in image
Today's Statistics
Entries Granted
0
Entries Denied
0
Currently Parked
62
High Confidence
0
Currently Parked Toyotas
ID
Entry Time
Confidence
1
2026-01-
99.88%



Scope of our Project:

- Real-time vehicle detection using AI
- Toyota brand identification
- Automated entry/exit logging
- Confidence-based decision making
- Manual override (grant/deny) feature
- Graphical interface with live camera feed
- Daily parking statistics

Git Hub Repo Link:

https://github.com/mohtishim1555/Ai-Project-and-Lab-Manual.git