



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:

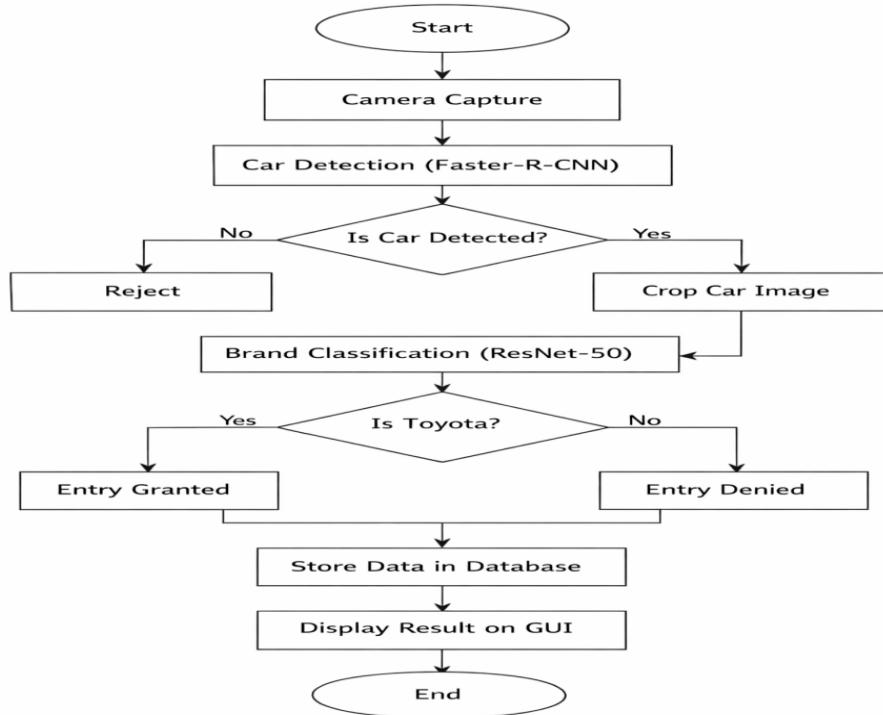
This project presents an AI-based vehicle identification and parking management system designed specifically for a Toyota dealership. The system uses computer vision and deep learning techniques to automatically detect vehicles and identify whether the vehicle belongs to the Toyota brand.

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

```

File Edit Selection View Go Run ... < > Q Ai
toyotaparking.py > ...
187     class ToyotaDetector:
188         def detect_and_classify(self, image):
189             if not image:
190                 return {'message': 'No car detected in image'}
191
192             results = []
193             for i, detection in enumerate(car_detections):
194                 # Extract car region
195                 car_region = self.extract_car_region(image, detection['box'])
196
197                 # Classify car brand
198                 classification = self.classify_car_brand(car_region)
199
200                 if classification:
201                     result = {
202                         'detection': detection,
203                         'classification': classification,
204                         'car_image': car_region
205                     }
206                     results.append(result)
207
208             # Find the most confident Toyota detection
209             toyota_detections = [r for r in results if r['classification']['is_toyota']]
210             if toyota_detections:
211                 # Get the detection with highest confidence
212                 best_detection = max(toyota_detections, key=lambda x: x['classification']['confidence'])
213             return {
214                 'cars_found': True,
215                 'toyota_found': True,
216                 'best_detection': best_detection,
217                 'all_detections': results,
218                 'message': f'Toyota detected with {best_detection["classification"]["confidence"]:.1%} confidence'
219             }
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278

```

```

File Edit Selection View Go Run ... < > Q Ai
toyotaparking.py > ...
1 import cv2
2 import numpy as np
3 import sqlite3
4 import datetime
5 import json
6 import time
7 import tkinter as tk
8 from tkinter import ttk, messagebox
9 from PIL import Image, ImageTk
10 import threading
11 import os
12 import torch
13 import torchvision.transforms as transforms
14 from torchvision import models
15 import torch.nn.functional as F
16 import torchvision
17 from torchvision.models.detection import FasterRCNN_ResNet50_FPN_Weights
18
19 class VehicleDatabase:
20     def __init__(self, db_name="toyota_parking.db"):
21         self.conn = sqlite3.connect(db_name, check_same_thread=False)
22         self.cursor = self.conn.cursor()
23         self.create_tables()
24
25     def create_tables(self):
26         """Create necessary database tables"""
27         self.cursor.execute('''
28             CREATE TABLE IF NOT EXISTS vehicle_entries (
29                 id INTEGER PRIMARY KEY AUTOINCREMENT,
30                 entry_time DATETIME,
31                 exit_time DATETIME,
32                 is_Toyota BOOLEAN,
33                 entry_status TEXT,
34             )
35         ''')
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
229
230
231
232
233
234
235
236
237
238
239
239
240
241
242
243
244
245
246
247
248
249
249
250
251
252
253
254
255
256
257
258
259
259
260
261
262
263
264
265
266
267
268
269
269
270
271
272
273
274
275
276
277
278
279
279
280
281
282
283
284
285
286
287
288
289
289
290
291
292
293
294
295
296
297
298
299
299
300
301
302
303
304
305
306
307
308
309
309
310
311
312
313
314
315
316
317
318
319
319
320
321
322
323
324
325
326
327
328
329
329
330
331
332
333
334
335
336
337
338
339
339
340
341
342
343
344
345
346
347
348
349
349
350
351
352
353
354
355
356
357
358
359
359
360
361
362
363
364
365
366
367
367
368
369
369
370
371
372
373
374
374
375
376
377
378
379
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
404
405
406
407
408
409
409
410
411
412
413
414
414
415
416
416
417
418
418
419
419
420
421
422
423
423
424
425
425
426
427
427
428
428
429
429
430
431
432
432
433
433
434
434
435
435
436
436
437
437
438
438
439
439
440
440
441
441
442
442
443
443
444
444
445
445
446
446
447
447
448
448
449
449
450
450
451
451
452
452
453
453
454
454
455
455
456
456
457
457
458
458
459
459
460
460
461
461
462
462
463
463
464
464
465
465
466
466
467
467
468
468
469
469
470
470
471
471
472
472
473
473
474
474
475
475
476
476
477
477
478
478
479
479
480
480
481
481
482
482
483
483
484
484
485
485
486
486
487
487
488
488
489
489
490
490
491
491
492
492
493
493
494
494
495
495
496
496
497
497
498
498
499
499
500
500
501
501
502
502
503
503
504
504
505
505
506
506
507
507
508
508
509
509
510
510
511
511
512
512
513
513
514
514
515
515
516
516
517
517
518
518
519
519
520
520
521
521
522
522
523
523
524
524
525
525
526
526
527
527
528
528
529
529
530
530
531
531
532
532
533
533
534
534
535
535
536
536
537
537
538
538
539
539
540
540
541
541
542
542
543
543
544
544
545
545
546
546
547
547
548
548
549
549
550
550
551
551
552
552
553
553
554
554
555
555
556
556
557
557
558
558
559
559
560
560
561
561
562
562
563
563
564
564
565
565
566
566
567
567
568
568
569
569
570
570
571
571
572
572
573
573
574
574
575
575
576
576
577
577
578
578
579
579
580
580
581
581
582
582
583
583
584
584
585
585
586
586
587
587
588
588
589
589
590
590
591
591
592
592
593
593
594
594
595
595
596
596
597
597
598
598
599
599
600
600
601
601
602
602
603
603
604
604
605
605
606
606
607
607
608
608
609
609
610
610
611
611
612
612
613
613
614
614
615
615
616
616
617
617
618
618
619
619
620
620
621
621
622
622
623
623
624
624
625
625
626
626
627
627
628
628
629
629
630
630
631
631
632
632
633
633
634
634
635
635
636
636
637
637
638
638
639
639
640
640
641
641
642
642
643
643
644
644
645
645
646
646
647
647
648
648
649
649
650
650
651
651
652
652
653
653
654
654
655
655
656
656
657
657
658
658
659
659
660
660
661
661
662
662
663
663
664
664
665
665
666
666
667
667
668
668
669
669
670
670
671
671
672
672
673
673
674
674
675
675
676
676
677
677
678
678
679
679
680
680
681
681
682
682
683
683
684
684
685
685
686
686
687
687
688
688
689
689
690
690
691
691
692
692
693
693
694
694
695
695
696
696
697
697
698
698
699
699
700
700
701
701
702
702
703
703
704
704
705
705
706
706
707
707
708
708
709
709
710
710
711
711
712
712
713
713
714
714
715
715
716
716
717
717
718
718
719
719
720
720
721
721
722
722
723
723
724
724
725
725
726
726
727
727
728
728
729
729
730
730
731
731
732
732
733
733
734
734
735
735
736
736
737
737
738
738
739
739
740
740
741
741
742
742
743
743
744
744
745
745
746
746
747
747
748
748
749
749
750
750
751
751
752
752
753
753
754
754
755
755
756
756
757
757
758
758
759
759
760
760
761
761
762
762
763
763
764
764
765
765
766
766
767
767
768
768
769
769
770
770
771
771
772
772
773
773
774
774
775
775
776
776
777
777
778
778
779
779
780
780
781
781
782
782
783
783
784
784
785
785
786
786
787
787
788
788
789
789
790
790
791
791
792
792
793
793
794
794
795
795
796
796
797
797
798
798
799
799
800
800
801
801
802
802
803
803
804
804
805
805
806
806
807
807
808
808
809
809
810
810
811
811
812
812
813
813
814
814
815
815
816
816
817
817
818
818
819
819
820
820
821
821
822
822
823
823
824
824
825
825
826
826
827
827
828
828
829
829
830
830
831
831
832
832
833
833
834
834
835
835
836
836
837
837
838
838
839
839
840
840
841
841
842
842
843
843
844
844
845
845
846
846
847
847
848
848
849
849
850
850
851
851
852
852
853
853
854
854
855
855
856
856
857
857
858
858
859
859
860
860
861
861
862
862
863
863
864
864
865
865
866
866
867
867
868
868
869
869
870
870
871
871
872
872
873
873
874
874
875
875
876
876
877
877
878
878
879
879
880
880
881
881
882
882
883
883
884
884
885
885
886
886
887
887
888
888
889
889
890
890
891
891
892
892
893
893
894
894
895
895
896
896
897
897
898
898
899
899
900
900
901
901
902
902
903
903
904
904
905
905
906
906
907
907
908
908
909
909
910
910
911
911
912
912
913
913
914
914
915
915
916
916
917
917
918
918
919
919
920
920
921
921
922
922
923
923
924
924
925
925
926
926
927
927
928
928
929
929
930
930
931
931
932
932
933
933
934
934
935
935
936
936
937
937
938
938
939
939
940
940
941
941
942
942
943
943
944
944
945
945
946
946
947
947
948
948
949
949
950
950
951
951
952
952
953
953
954
954
955
955
956
956
957
957
958
958
959
959
960
960
961
961
962
962
963
963
964
964
965
965
966
966
967
967
968
968
969
969
970
970
971
971
972
972
973
973
974
974
975
975
976
976
977
977
978
978
979
979
980
980
981
981
982
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
990
991
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000

```



The screenshot shows a Python IDE interface with the following details:

- File Menu:** File, Edit, Selection, View, Go, Run, ...
- Search Bar:** Search term "Ai".
- Toolbar:** Includes icons for file operations, search, and other common functions.
- Code Area:** Displays the `toyotaparking.py` script. The code is a Tkinter-based GUI for managing Toyota parking. It includes a label frame for currently parked vehicles, a treeview for vehicle data, and buttons for exiting and updating statistics.
- Right Panel:** Shows a detailed code navigation or search results panel with multiple entries related to the current file.
- Status Bar:** Indicators for Line 937, Column 11, Spaces: 4, UTF-8, CRLF, Python 3.11, and Go Live.

The screenshot shows a Python IDE interface with the following details:

- Title Bar:** The title bar displays "toyataparking.py" and "File Edit Selection View Go Run ...".
- Toolbar:** A standard Windows-style toolbar is visible at the top.
- Status Bar:** The status bar shows "FTSE 100 +1.06% 6:57 PM".
- Code Area:** The main area contains Python code for a "ParkingSystemGUI" class. The code handles manual override requests based on AI detection results. It uses `messagebox.askyesno` to prompt the user and `self.db.log_entry` to log entries. It also updates UI labels and status messages.
- Right Panel:** A large panel on the right side displays a detailed log or history of the application's operations, showing numerous entries related to parking grants and detections.

```
File Edit Selection View Go Run ...
toyataparking.py > ...
320     class ParkingSystemGUI:
321         def manual_grant_entry(self):
322             if 'best_detection' in result:
323                 car_class = result['best_detection']['classification']['class']
324                 confidence = result['best_detection']['classification']['confidence']
325
326                 response = messagebox.askyesno("Manual Override",
327                     f"AI detected this as: {car_class.upper()} (confidence: {confidence:.1%})\n\n"
328                     "Do you want to manually grant entry anyway?")
329             else:
330                 response = messagebox.askyesno("Manual Override",
331                     "Car detected but brand unknown.\n\n"
332                     "Do you want to manually grant entry anyway?")
333
334             if response:
335                 # Log as manual override
336                 entry_id = self.db.log_entry(True, 1.0, "granted")
337
338                 # Update UI
339                 if 'best_detection' in result:
340                     self.details_label.config(text=f"Manual Override - AI said: {car_class.upper()}")
341                 else:
342                     self.details_label.config(text="Manual Override - Brand unknown")
343                     self.confidence_label.config(text="Confidence: 100% (Manual)")
344                     self.result_label.config(text="✓ MANUAL ENTRY GRANTED")
345                     self.status_label.config(text="ENTRY GRANTED", style="Granted.TLabel")
346                     self.test_result.config(text="Entry granted by manual override")
347
348             messagebox.showinfo("Manual Entry Granted", "Entry granted by manual override.")
```

The screenshot shows a Python IDE interface with the following details:

- Title Bar:** File, Edit, Selection, View, Go, Run, ...
- Search Bar:** Ai
- Left Sidebar:** Includes icons for file operations like Open, Save, Find, and a tree view of the project structure.
- Code Editor:** The main area displays the `toyotaparking.py` script. The code is written in Python and defines a `ParkingSystemGUI` class with methods for detecting car brands and updating display frames.
- Right Sidebar:** Shows a detailed list of code analysis results, including metrics like LOC, NLOC, and various code quality indicators (e.g., 100% coverage, 0 bugs).

```
File Edit Selection View Go Run ...
Ai
toyotaparking.py ×
toyotaparking.py > ...
320 class ParkingSystemGUI:
553     def detect_car_brand(self):
554         if not result['cars_found']:
555             self.result_label.config(text="❌ NO CAR DETECTED")
556             self.details_label.config(text="Make sure car is clearly visible")
557             self.confidence_label.config(text="")
558             self.status_label.config(text="", style="")
559             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="🔴 NOT A TOYOTA")
593             self.details_label.config(text=f"Detected as: {car_class.upper()}")
```

```
File Edit Selection View Go Run ... < > 🔍 Ai
```

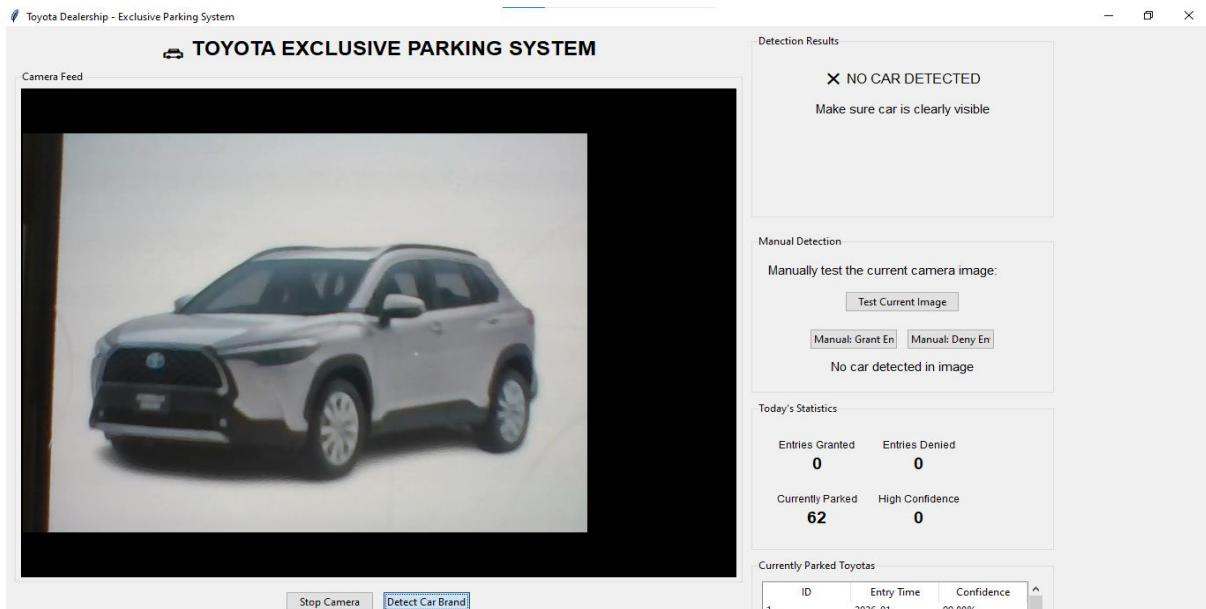
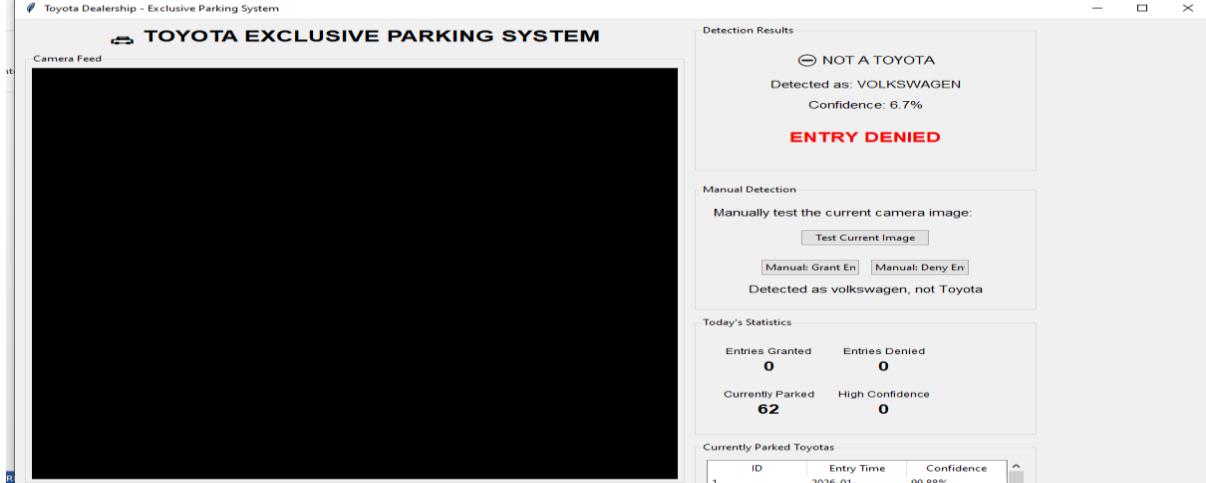
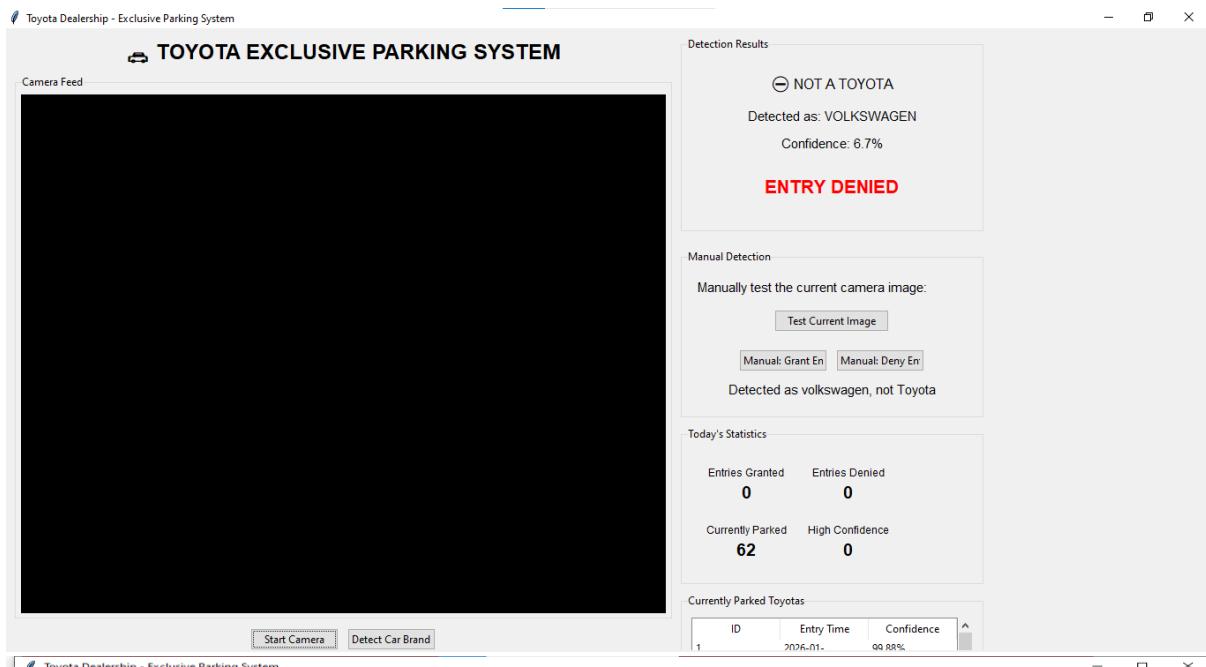
```
toyotaparking.py ×
toyotaparking.py > ...
896 def main():
897
898     # Install dependencies
899     if not install_dependencies():
900         print("\nFailed to install dependencies. Please install manually:")
901         print("pip install torch torchvision opencv-python pillow numpy")
902         input("\nPress Enter to exit...")
903         return
904
905     print("\n" + "="*60)
906     print("IMPORTANT INSTRUCTIONS FOR PHOTO DETECTION:")
907     print("="*60)
908     print("1. Hold the phone with car photo at a 45-degree angle to camera")
909     print("2. Ensure the car fills most of the phone screen")
910     print("3. Avoid reflections and glare on the phone screen")
911     print("4. Use good lighting conditions")
912     print("5. Keep the phone steady for 2 seconds when detecting")
913     print("6. Use high-quality car photos for best results")
914     print("\nHOW TO USE:")
915     print("1. Start the camera")
916     print("2. Show a photo of a car on your phone")
917     print("3. Click 'Detect Car Brand' button")
918     print("4. Green box = Toyota detected")
919     print("5. Red box = Other car detected")
920     print(" "*60 + "\n")
921
922     # Create main window
923     root = tk.Tk()
924     app = ParkingSystemGUI(root)
925
926     # Handle window closing
927     root.protocol("WM_DELETE_WINDOW", app.on_closing)
928
929
930
931
932
```

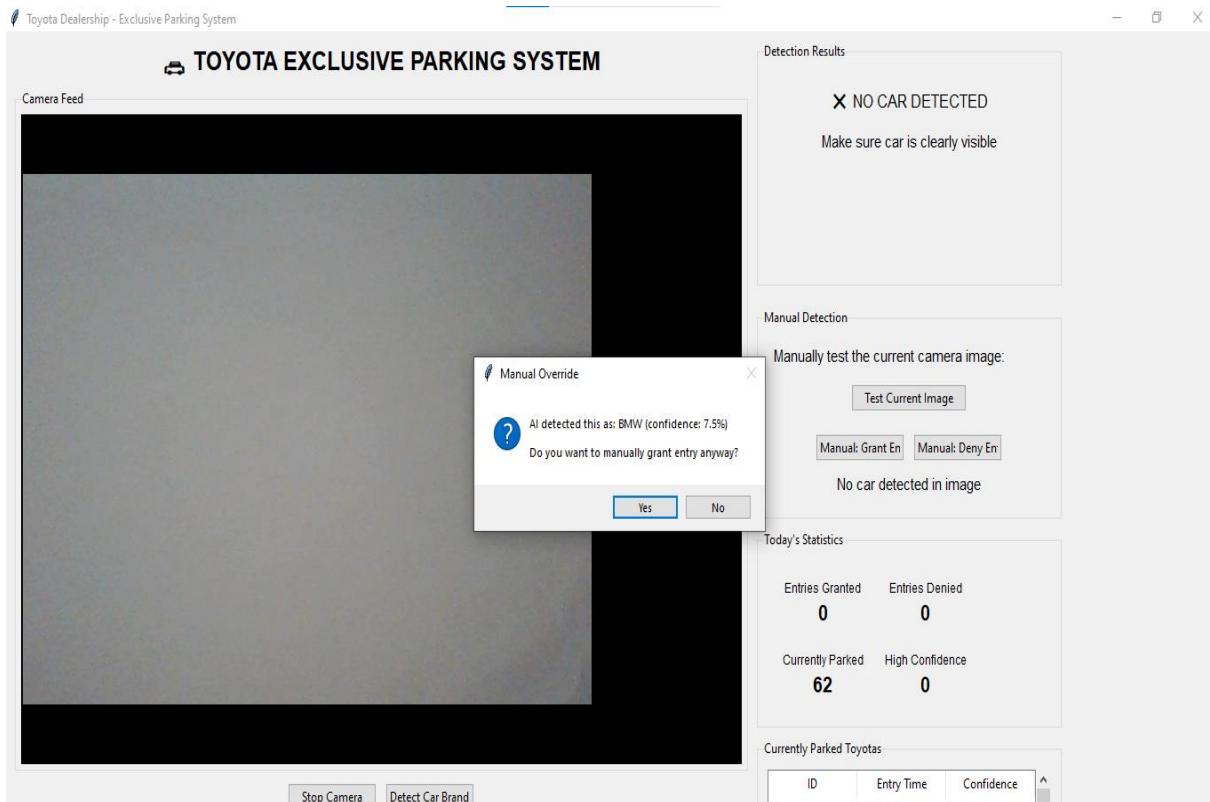
The screenshot shows a Python IDE interface with a code editor and various toolbars. The code editor displays a file named `toyotaparking.py`. The script implements a parking system GUI using Tkinter. It includes logic to log entries, update UI labels, and show message boxes for denied or unknown entries. A sidebar on the right shows a preview of the application's user interface.

```
Ln 937, Col 11 Spaces: 4 UTE-8 CRLF [ ] Python FTSE 100 +1.06% Python 3.11 Go Live Go Live  
File Edit Selection View Go Run ... < > Q Ai
```

```
toyotaparking.py x
toyotaparking.py > ...
320 class ParkingSystemGUI:
722     def manual_deny_entry(self):
723
724         # Log as denied entry
725         self.db.log_entry(False, confidence, "denied")
726
727         # Update UI
728         self.details_label.config(text=f"Detected as: {car_class.upper()}")
729         self.confidence_label.config(text=f"Confidence: {confidence:.1%}")
730         self.result_label.config(text=f"⚠ NOT A TOYOTA")
731         self.status_label.config(text="ENTRY DENIED", style="Denied.TLabel")
732         self.test_result.config(text="Entry denied. Detected as {car_class}")
733
734         messagebox.showwarning("Access Denied",
735                               f"Vehicle identified as: {car_class.upper()}\n"
736                               f"Confidence: {confidence:.1%}\n"
737                               "Only Toyota vehicles are allowed.")
738
739     else:
740
741         # Car detected but brand unknown
742         self.db.log_entry(False, 0.5, "denied")
743
744         # Update UI
745         self.details_label.config(text="Car detected, brand unknown")
746         self.confidence_label.config(text="")
747         self.result_label.config(text="⚠ UNKNOWN CAR BRAND")
748         self.status_label.config(text="ENTRY DENIED", style="Denied.TLabel")
749         self.test_result.config(text="Entry denied - Unknown brand")
750
751         messagebox.showwarning("Access Denied",
752                               f"Car detected but brand could not be identified.\n"
753                               "Only Toyota vehicles are allowed.")
754
755     # Update statistics
```

Output Screenshots:





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>