Parking Availability System

Design Plan Draft

Erik Meurrens, Benjamin Simonson, Ryan Jalloul, Evan Tobon, Samer Khatib

Introduction

Purpose / Need

Live parking data is only available at a select few locations on the University of Florida (UF) campus, making it difficult for those who must drive to campus to find parking during busy periods, as it is impossible to check real-time availability beforehand. The purpose of this project is to provide real-time, reliable and accessible parking capacity tracking information for individuals who wish to park at the parking facilities on campus. This information would be beneficial for off-campus commuters and visitors to campus, reducing the stress, frustration, and confusion drivers can feel trying to find parking, as well as the congestion that can occur at parking facilities, during busy traffic periods on campus. Our project, the Parking Availability System (PAS), was inspired to help the UF community, alleviating the symptoms of the lack of easily available parking resources for the campus and promoting quicker and more reliable transportation to and from campus, by offering a way to access parking availability information in real time via PAS.

Domain & Prior Art

Currently, there are existing implementations that provide onsite observability for garage capacities. The solution currently employed by the University of Florida is the Genetec AutoVu system that uses an automatic license plate recognition service to identify and read license plates and count vehicles that are currently in the facility. While effective in license plate identification, this solution lacks a virtual view of parking facility capacities and current availability. This omission inconveniences drivers searching for available parking when none exists. Additionally, the current system comes with a significant cost burden and is facilitated via subscription services that may go underutilized. Specifically, this system maintains a "$6000 servicing cost, $11,000-$8,000 per lane, and $2,495 per camera" [1]. In contrast, our project will provide a much more affordable solution to the AutoVu as it will only require equipment and installation fees. Our project will also provide virtual observability into parking facility current availability to streamline an individual's search into finding a parking spot.

Impact & Risk Assessment

This project will benefit the campus community by providing commuters and visitors a mobile application to reference parking resources on campus and view real-time parking availability. This will have an overall benefit to campus by alleviating the current lack of easily accessible parking resources, thus reducing the stress, frustration, and confusion drivers feel trying to find parking on campus. Additionally, this would make it easier for drivers to plan their routes, promoting better traffic flow and less congestion on campus as drivers will know where they will be parking ahead of time. By lessening congestion and reducing the drivers' frustration of finding parking, PAS will enable safer driving around campus, improving the safety of pedestrians and drivers alike.

However, one potential drawback, given that this system requires a video feed transmitted to an object detection model in the cloud, is concerns about personal privacy. While these parking facilities are public spaces, we guarantee that PAS will only collect video data in order to identify cars entering and exiting parking facilities and no more than what is necessary for no other purpose. With this in mind, we believe that the benefit to the community outweighs the privacy considerations.

Statement of Work

Board Software

A Raspberry Pi 3 Model B+ will handle object detection for vehicles, data streaming, and updating availability counter in the database. Equipped with a camera module, it will process a video stream at a minimum of 10 frames per second to detect vehicles and license plates. The Raspberry Pi will select an image frame with a high confidence containing the license plate, stream the data to AWS for license plate recognition. Cars are detected and determined to either be leaving or entering the facility, adjusting the counter accordingly.

The Raspberry Pi will use a Wi-Fi module to connect to the “UFDevice” SSID, ensuring a connection to the internet to allow seamless data transfer to AWS for real-time updates. In the event of a lost connection, image frames will be cached until the connection is re-established. Additionally, allowing the board onto UF Wi-Fi would allow us to SSH into the board to program it live.

All the module initialization and usage will be established by programming the board using Python and will use the AWS IoT Core connectivity libraries.

Object Detection Model

Utilizing the preexisting You Only Look Once (YOLOv11) model by Ultralytics, we will make use of transfer learning to train the yolo11x model to specifically detect vehicles and license plates. This model will be run on the Raspberry Pi 3 to detect vehicles and send image snapshots to our dedicated remote processing servers to conduct the optical character recognition (OCR) using the Google Tesseract model and device registration within our databases through the Backend API.

The data used to train the transfer learning model will be trained on an open-sourced license plate dataset found on Hugging Face [2]. By using the pretrained YOLOv11 model and open-sourced dataset, we save extensive amounts of capacity in the project while maintaining a high level of accuracy in detection.

Backend API

Once the OCR model completes its recognition and has extracted the necessary data from the image sent from the Raspberry Pi 3, we register the information into a PostgreSQL database using a RESTful API written in Golang. The API implements CRUD functions to CREATE a new vehicle object stored in the database, READ a vehicle object that is stored in the database, UPDATE a new vehicle object as a replacement for an existing object in the database, and DELETE a vehicle object in the database.

Making use of a RESTful API to connect the detection pipeline to our frontend application standardizes what service to call to access data within the database. The frontend application will make use of the GET and DELETE functions within the API to update different pages with data from the database, and the detection pipeline will make use of the POST, PUT, and DELETE functions to add and update vehicles within the database, as well as deleting vehicles from the database once they exit the garage.

Hardware Development

- Our system is intended to maintain a persistent power source.
    - If no persistent source is available, the system is targeted to 24 hours.
    - Batteries – were thinking 2 10k mah batteries when rotated every day gies us a total amount of power of about 40 hours but this means a constant Maintenace.
    - Power over Ethernet was another option that needs to be considered but we are unsure of how this would work or if the power would be strong enough
- **Raspberry Pi**
    - **2.4-5GHz for wireless LAN capability and ~300 MB/s throughput.**
      This allows the Raspberry Pi to connect to the internet or a local network for remote monitoring, sending data to a server, or triggering alerts. The throughput is sufficient for transferring image data, car counts, or other logs efficiently over a network.
    - **1.4Hz 64-bit quad-core processor, dual-band wireless LAN.**
      The processing power is essential for handling the image processing algorithms, such as object detection and counting. It ensures that the Raspberry Pi can process video frames, detect cars, and count them in real-time.
        - **Bluetooth 4.2**
          You could use Bluetooth to connect to nearby devices, such as mobile phones or tablets, for monitoring car counts or other project-specific functionalities.
        - **1 GB LPDDR2 SDRAM**
          This memory may be sufficient for handling small to medium-sized image processing tasks, storing video frames temporarily, and running the car counting software alongside the operating system. Testing will happen to see if this is enough in the future.
        - **4 USB 2.0 ports**
          These ports can be used to connect peripherals, such as sensors or you could connect external storage for saving video recordings or logs.
        - **HDMI**
          The HDMI port allows you to connect a display for setting up, debugging, or visualizing the car detection and counting in real-time.
        - **Camera Port**

This port connects directly to the camera module, such as the OV7670, which will be used to capture images or video of the garage entrance where the cars are being counted.

        - **5V/2.5A DC power input**

- Camera Module
    - OV7670 – VGA (640x480)
      he resolution of 640x480 is modest but should be enough for simple car detection tasks. Image processing algorithms like edge detection or object tracking will be applied to this video feed to count cars.
- Ultrasonic Sensor
    - The ultrasonic sensor can be used to enhance the accuracy of the car counting system by detecting when a vehicle is close to the entrance. This could trigger the camera to take images or help confirm a vehicle's presence when counting.

All Together, these components allow for an efficient, low-cost solution for counting cars in a garage using image processing on a Raspberry Pi.

Mobile Application

A mobile application will be developed to provide an intuitive user interface, enabling users to search for parking facilities and view real-time availability. Built using the Flutter framework and Dart, the app will integrate the Google Maps API to display nearby parking facilities. Each facility will show its current availability, retrieved from an SQL database updated by an edge device at the garage. This approach ensures users can easily locate available parking in close proximity, streamlining the search process for parking spots.

Core Features
- Car object detection
- License plate recognition
- Video stream
- Wireless edge device
- Real-time Parking availability determination accessible via mobile app

Secondary Features
- Tracklog containing license plate information for cars that have entered and left
- Enforce legal parking in licensed zones

Milestones and Tasks:

Pre-Alpha Build Milestone		**Target Date: 10/25/24**
Features/Tasks:	**Assignee(s):**	**Target Date:**
Identify and obtain remaining hardware modules to configure raspberry pi (RPi)	Ryan	10/18/24
Configure software backbone to RPi (OS, frameworks/libraries, programs)	Evan, Ben, Erik	10/18/24

Connect RPi to UF WiFi	Ben	10/18/24
Configure remote access to RPi for development	Erik	10/18/24
Connect camera module hardware to RPi	Ryan	10/18/24
Configure AWS and SQL database	Samer	10/18/24
Complete Backend API	Samer	11/15/24
Establish connection from RPi to AWS and SQL database	Ben, Evan	10/25/24
Establish connection from App to SQL database	Ben, Erik	10/25/24
Design Prototype Milestone		**Target Date: 11/22/24**
Features/Task:	**Assignee(s):**	**Target Date:**
Obtain or create camera driver library on RPi	Evan	11/01/24
View video feed from camera through RPi	Ben, Ryan	11/08/24
Provisional machine learning model on RPi	Samer, Erik	11/08/24
Provisional machine learning model on AWS	Samer	11/08/24
View processed video stream from RPi remotely	Ben	11/15/24
Configure global parking facility counter	Ben, Samer	11/15/24
Set up ALPR photo/frame transmission from RPi to AWS	Evan, Ben	11/15/24
Test machine learning models	Samer	11/22/24
Test databases	Erik, Ben, Samer	11/22/24
Confirm real-time app updates	Erik	11/22/24
Benchmark device prototype	Ryan, Erik	11/22/24
Design housing for pi hardware	Ryan	11/15/24

Deliverable Artifacts

Hardware Module: Raspberry Pi, Camera, and Physical Housing

- The board and attachments will facilitate the capture and the transmission of necessary data. This data will consist of capturing the car and license plate frames, transmitting car and license plate frames, and benchmarking car entry via a sensor. The frames will be transmitted by the board to the AWS module for further feature extraction and analysis.
- Maintenance will depend on whether the module has persistent power supply, and the lifespan of the attached components will depend on usage. Assuming no persistent power supply, batteries will need to be changed and recharged every 24 hours.

Yolov11 Car Detection Model

- This model will oversee detecting and sending the information regarding the detection of a car from the Hardware Module to the AWS Module for further processing. This model will be run entirely on the Raspberry Pi and will be the first step in the pipeline of registering a new car entering the garage or removing a car that has just exited the garage.
- No maintenance will need to be done once the model is properly finetuned and trained on a specific dataset to detect only cars and license plates.

Backend API

- This backend API will be a RESTful API that acts as a middleman between the frontend Parking Availability Application and the Hardware Module. By implementing CREATE, READ, UPDATE, and DELETE (CRUD) functions, the API will allow access to the PostgreSQL Database without having to directly write or read from it.
- Minimal maintenance will be needed once the API is completed, as the only changes required will be potential updates and changes to the functions as the scope of the project changes.

AWS Module

- The AWS module will receive the frame binary data from the board and build an image from it. The model will then confirm that a vehicle has been detected and has entered or exited the parking facility. Optical Character Recognition will be run to identify the characters of the license plate and assign it to that vehicle. If a vehicle enters or exits the facility a global counter will be incremented or decremented respectively which will convey the total number of vehicles in the facility. The license plate will be assigned to the vehicle that has entered the facility and removed when the vehicle exits the facility.
- Maintenance will include observing costs accrued by different AWS services and apply changes to ensure that the model is maintaining its performance at the minimal potential financial impact.

PostgreSQL Database

- This database will be accessible by both the AWS module and the board through the backend API. The board will write the benchmark entry data of cars to this database and the AWS module will assign the interpreted license plate to the car that has entered the parking facility.
- No maintenance will need to be done besides cleaning left over data, if any, and validating the performance of the model and the sensor as need be.

Parking Availability Application

- This application will be developed to allow users to enhance observations in facility availability. It will allow users to search for different parking facilities on campus to observe available parking options on campus.
- Maintenance will include ensuring that the application is available to mobile user markets, ensuring that identified bugs are resolved in a timely manner, and that recommended enhancements to the application are considered and enacted.

Programming Interface Repository

- This deliverable will contain all the source code for all the artifacts. It will maintain a concurrent instance of all aspects of this project.
- Maintenance will include ensuring that the most up to date version of all code is pushed and available to this repository. It might also include issues of any known problems or bugs, and any ideas that are planned to be enacted upon.

Mockups

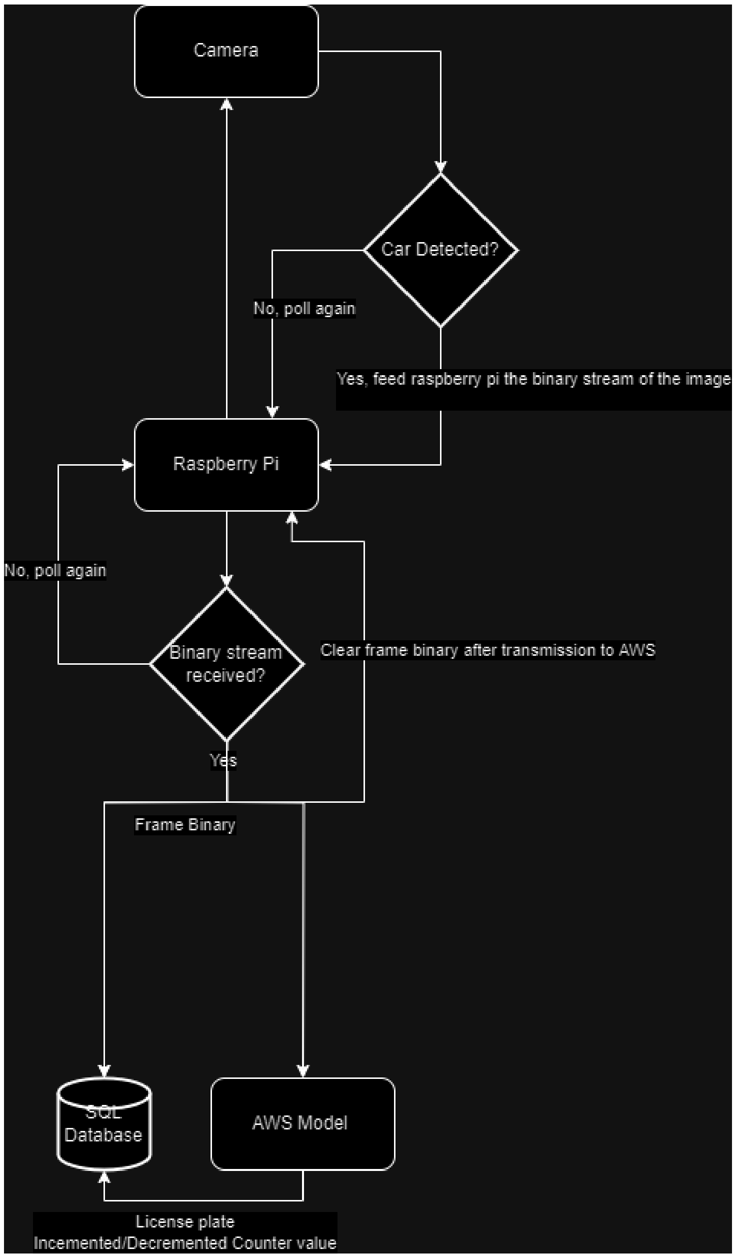

Camera
Car Detected?
No, poll again
Yes, feed raspberry pi the binary stream of the image
Raspberry Pi
No, poll again
Binary stream received?
Clear frame binary after transmission to AWS
Yes
Frame Binary
SQL Database
AWS Model
License plate
Incemented/Decremented Counter value

Figure 1. System Architecture Diagram

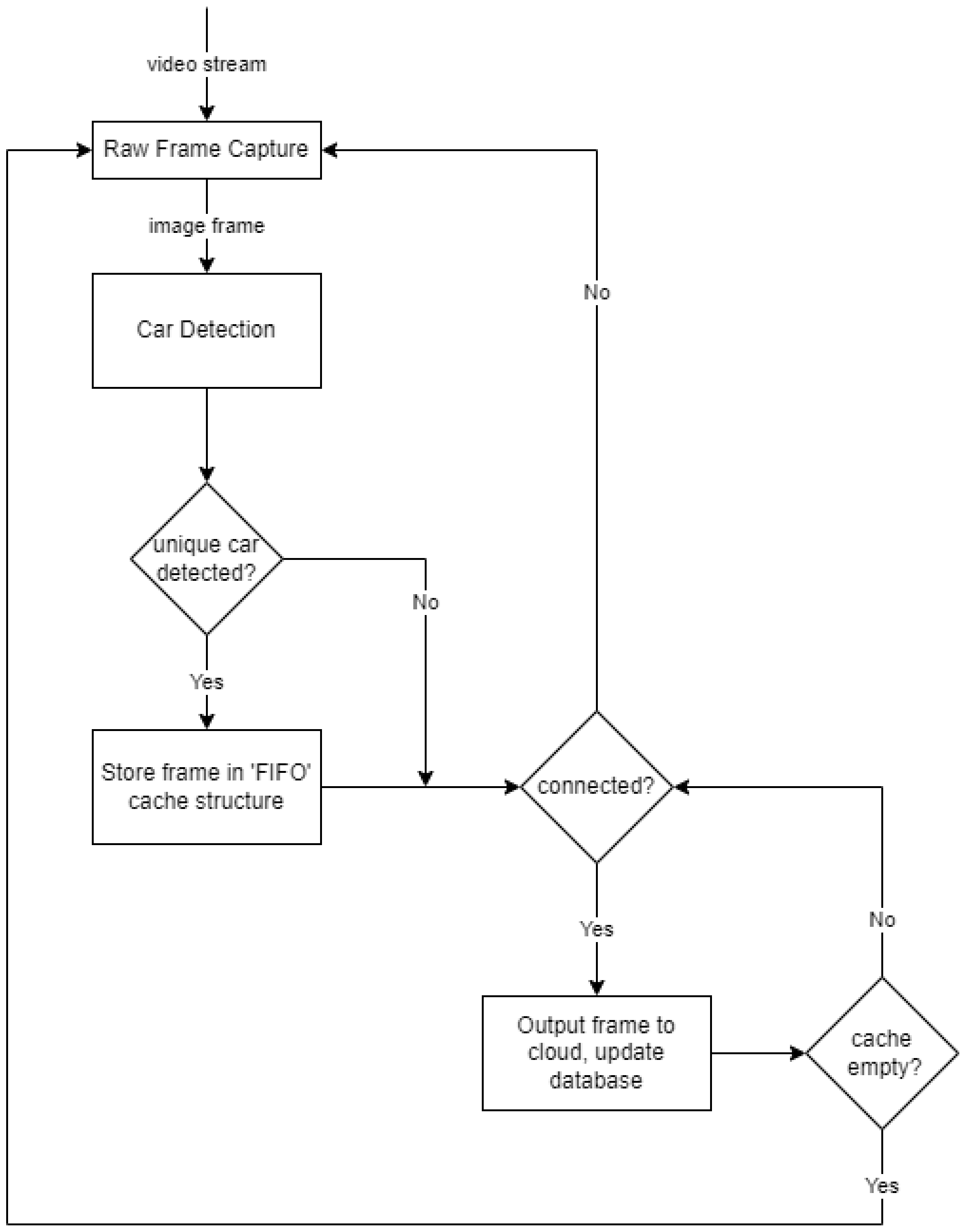
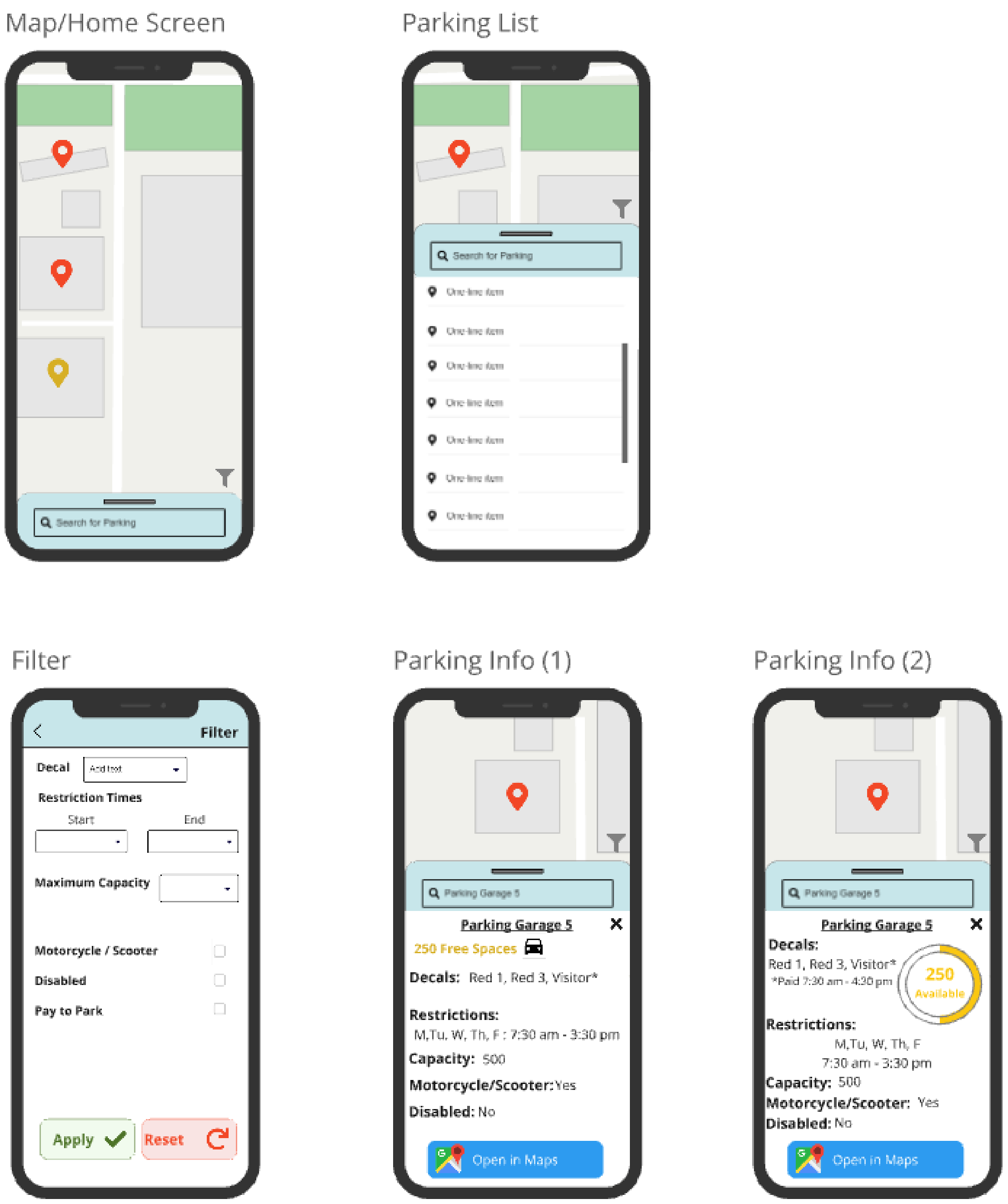


Figure 2. Board Program Flowchart



250 Free Spaces
Decals: Red 1, Red 3, Visitor*
Restrictions: M,Tu, W, Th, F : 7:30 am - 3:30 pm
Capacity: 500
Motorcycle/Scooter: Yes
Disabled: No

Decals: Red 1, Red 3, Visitor*
*Paid 7:30 am - 4:20 pm
Restrictions: M,Tu, W, Th, F 7:30 am - 3:30 pm
Capacity: 500
Motorcycle/Scooter: Yes
Disabled: No

Figure 3. Mobile App High-Fidelity Wireframes

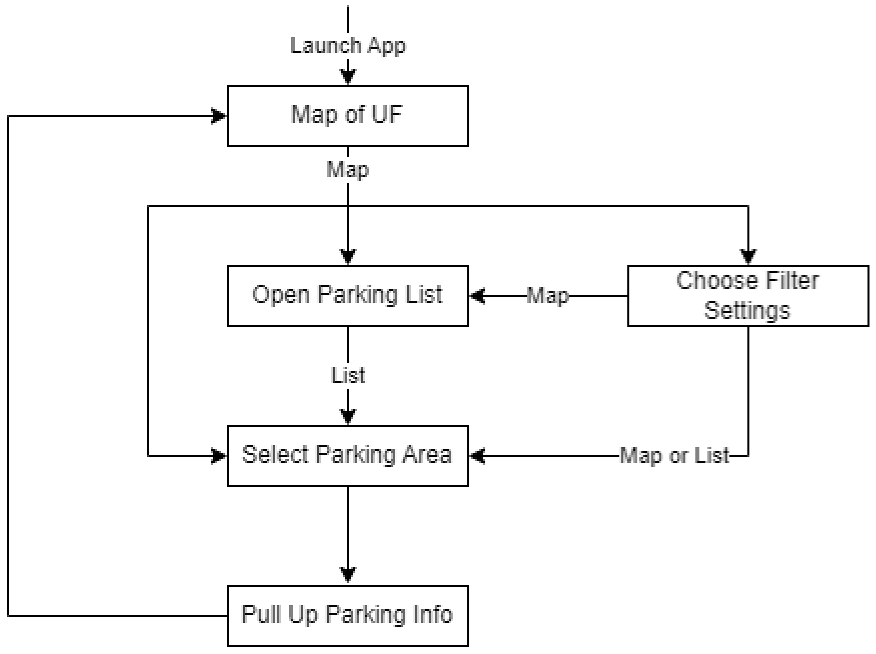


Figure 4. Mobile App User Flow Diagram

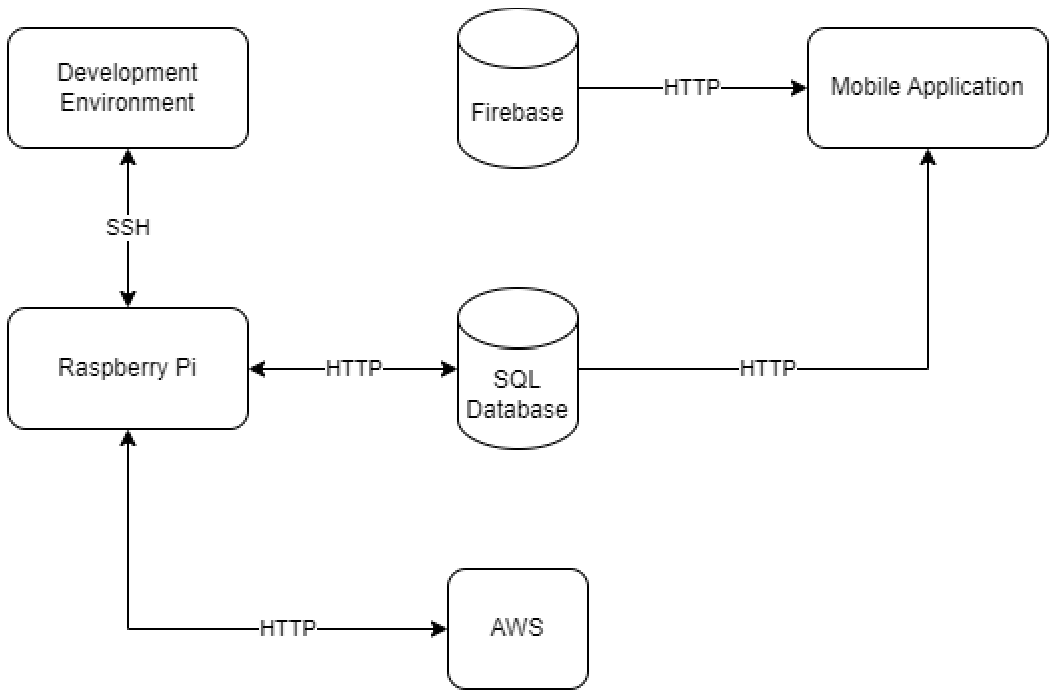


Figure 5. System Networking Diagram

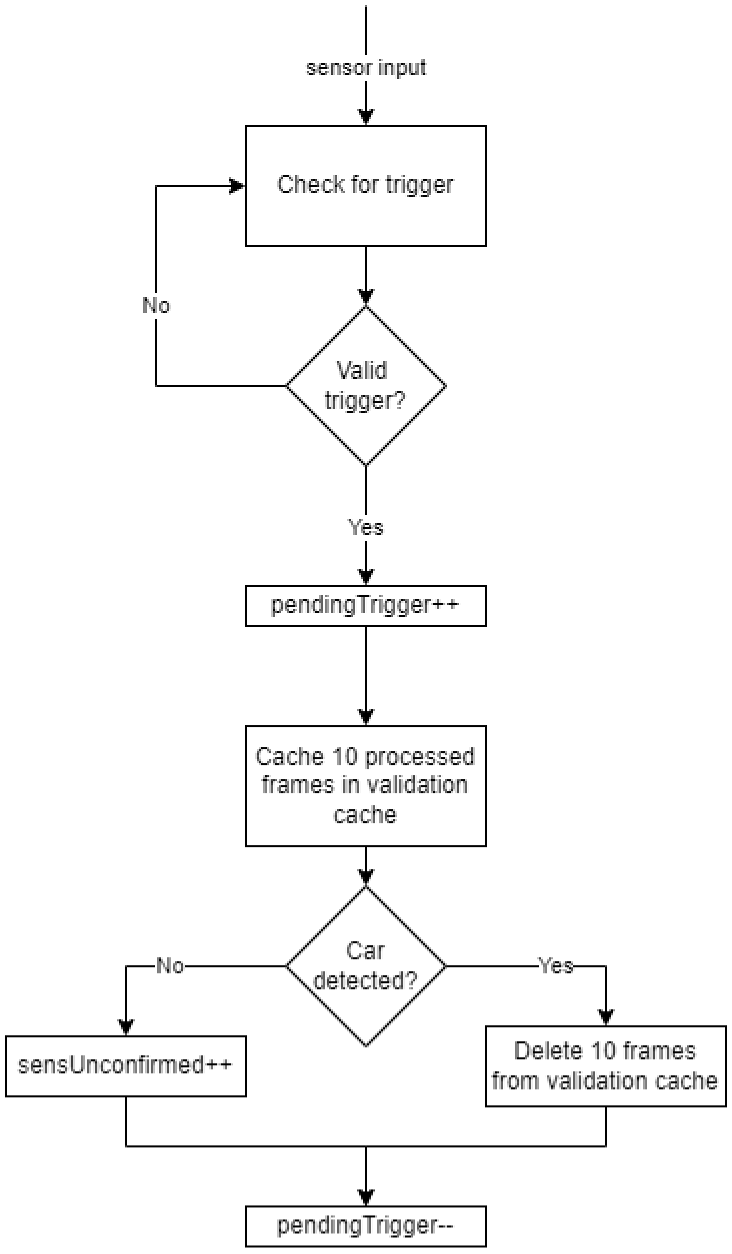


Figure 6. Plan to Validate System Performance Using Sensor

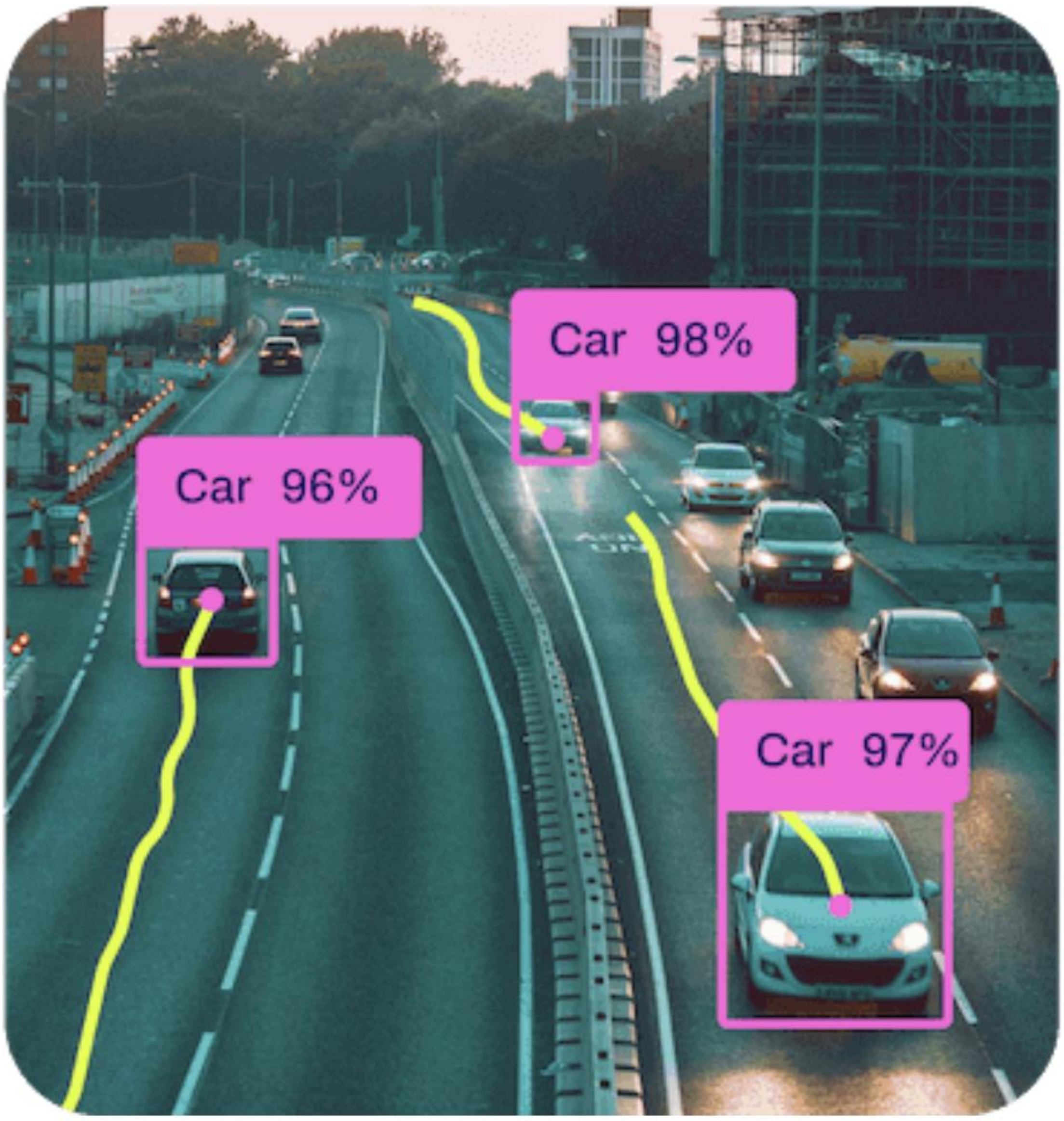


Figure 7. YOLOv11x model detecting vehicles

References

[1]L. Du and S. Washburn, “SMART PARKING SYSTEM ON UF CAMPUS,” Apr. 2019. Available: https://fora.aa.ufl.edu/docs/38/2018-2019/SmartParkingProposalLiliDuScottWashburn.pdf

[2] A. Startups, ‘Vehicle Registration Plates Dataset’, *Roboflow Universe*. Roboflow, Jun-2022.