UNIPARK
SDDEC25-09

Creating an Automated Parking System Enabled by IoT Technology

Introduction

Problem: Finding a parking spot is currently inefficient, time-consuming, and frustrating.

Solution: UNIPark is a modern day solution leveraging **computer vision** algorithms to help solve these issues.

Intended Users

Parking User:
- Rushed Parker
  - Quick to use with an intuitive interface
- Elderly and Disabled Parker
  - Accessible and simple
- Patient Parker
  - Able to secure their “perfect” spot
- Proactive Parker
  - Able to reserve spot ahead of time
- Forgetful Parker
  - Find car with license plate look-up

Admin User:
- Configure lot specifics (price, reservation durations, etc.)

Design Requirements

Functional Requirements:
- Display capacity of parking lots
- Display capacity within lots
- Incorporate a secure payment portal
- Enable reporting of issues (misparking, blockages, etc)
- Enable saving of spots

Non-Functional Requirements:
- Easy to use interface (for parkers and admins)
- Consistent behavior (reliable app, server, and hardware)
- Quick use of key features
- Visually appealing (app and hardware)
- Modular hardware (deployable)

Technical Details

Frontend
- React Native
- Paper
- Stripe
- Axios
- IOS and Android

Backend
- Java Spring Boot
- MySQL
- Docker Compose
- REST API
- Deployed to Pyrite VM

Hardware
- Arducam 16MP Wide Angle USB Camera
  - Collecting images of the parking lot
- Raspberry Pi 5
  - Checking images for cars
  - Reading license plates
  - Sending collected data to backend
- Docker
  - Automatic deployments

Testing

Hardware Testing
- Latency and Resolution

Connectivity
- Pi and Backend Server

Endpoint
- Ensure Correct REST results and functionality

Frontend
- User testing via app simulation
- Regression testing

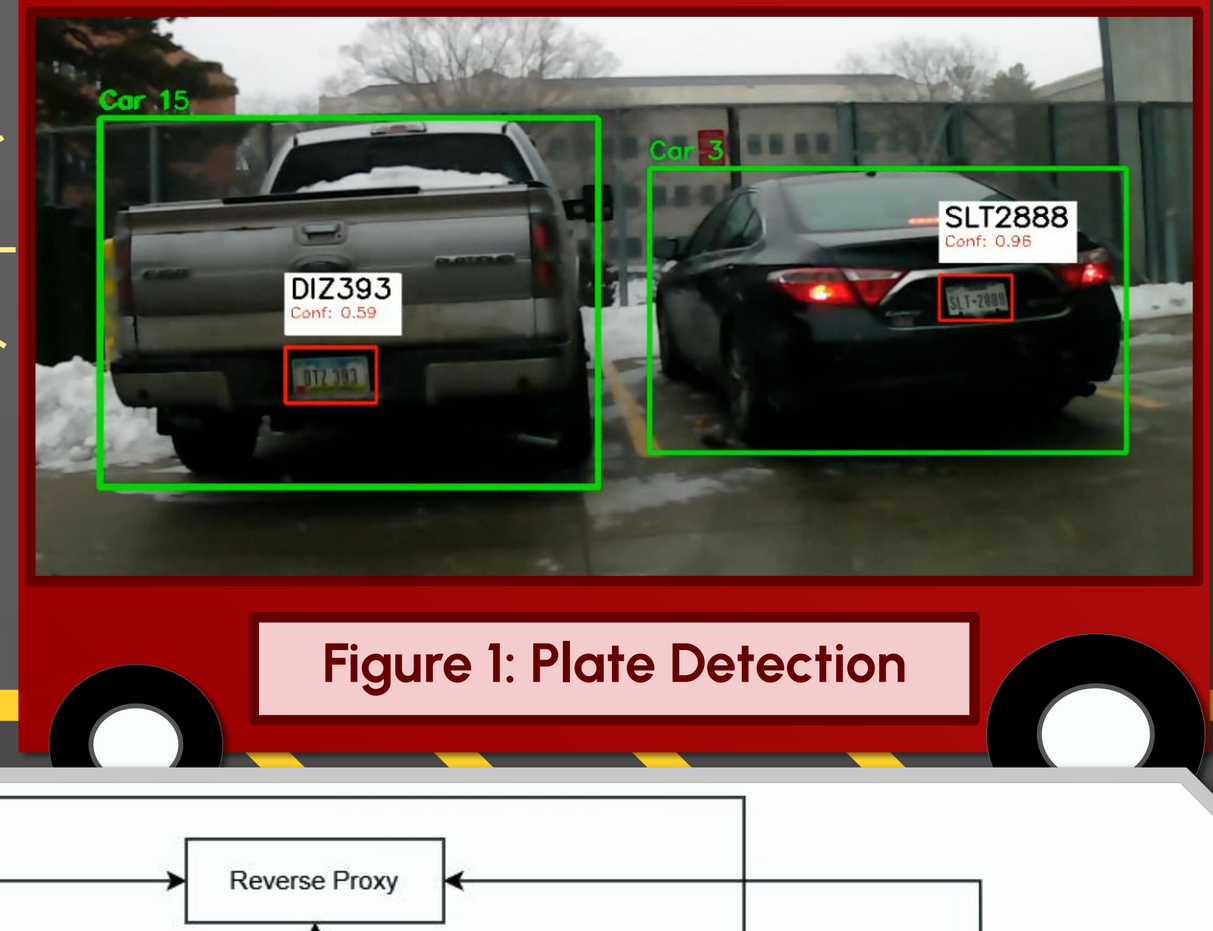

Figure 1: Plate Detection

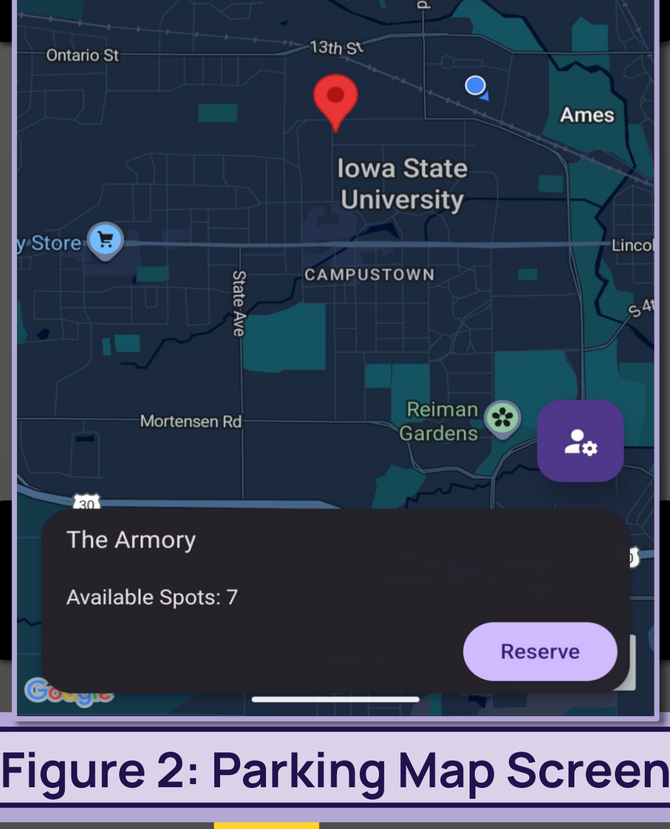

Figure 2: Parking Map Screen

Figure 3: Design Approach