# Vaxtor Server V1 User Guide

## 1. Overview

Vaxtor Server V1 is a Flask-based web application that provides license plate recognition functionality. The system uses SQLite as its database and is designed to store and manage license plate information along with associated vehicle details (make and model). The application exposes RESTful APIs for plate recognition and data management.

Key Features include:

* License plate recognition and storage
* Vehicle make and model tracking
* RESTful API endpoints
* SQLite database for data persistence
* CORS support for cross-origin requests

## 2. Installation and Setup

### 2.1 Prerequisites

* Python 3.8 or higher
* pip (Python package manager)
* Virtual environment (recommended)

### 2.2 Installation Steps

1. Clone the Repository

* git clone [repository-url]
* cd [repository-directory]

1. Create and Activate Virtual Environment

* For Windows:
  + python -m venv venv
  + venv\Scripts\activate
* For Linux/Mac:
  + python3 -m venv venv
  + source venv/bin/activate

1. Install Dependencies

* pip install -r requirements.txt

1. Database Setup

* The system uses SQLite, which will be automatically created when the application first runs
* The database file (alpr.db) will be created in the project root directory

## 3. Running the System

1. Start the system

* python main.py

1. Access the application

* The server runs on <http://localhost:5000>

1. Default Configuration

* Host: 0.0.0.0
* Port: 5000
* Debug Mode: Enabled

## 4. API Endpoints

The system provides two main endpoints:

### Web Interface Endpoint

* + URL: /home/platerecognizer
  + Method: GET
  + Description: Displays a web interface showing all license plate records
  + Response: Renders an HTML template with all plate records
  + Usage: Access this endpoint in a web browser to view the plate recognition dashboard

1. **Latest Record API**
   * URL: /platerecognizer-record-latest
   * Method: GET
   * Description: Retrieves the most recent license plate record
   * Response format: {"id": "integer", "plate": "string", "make": "string", "model": "string"}
   * Error Response: Returns 404 if no records are found
   * Usage: Use this endpoint to get the latest plate recognition result

## 5. Troubleshooting & Common Issues

1. **Database Issues**

* Problem: Database not created
* Solution: Ensure the application has write permissions in the project directory
* Solution: Check if SQLite is properly installed

1. **Server Issues**

* Problem: Port 5000 already in use
* Solution: Change the port number in main.py to an available port
* Solution: Find and close the process using port 5000
* Problem: Module not found errors
* Solution: Ensure all dependencies are installed correctly
* Solution: Verify virtual environment activation

1. **API Issues**

* Problem: CORS errors
* Solution: Verify the client is using the correct origin
* Solution: Check if flask\_cors is properly configured

1. **General Troubleshooting Steps**

* Check the application logs for error messages
* Verify all dependencies are installed correctly
* Ensure the virtual environment is activated
* Check file permissions in the project directory
* Verify network connectivity and port availability