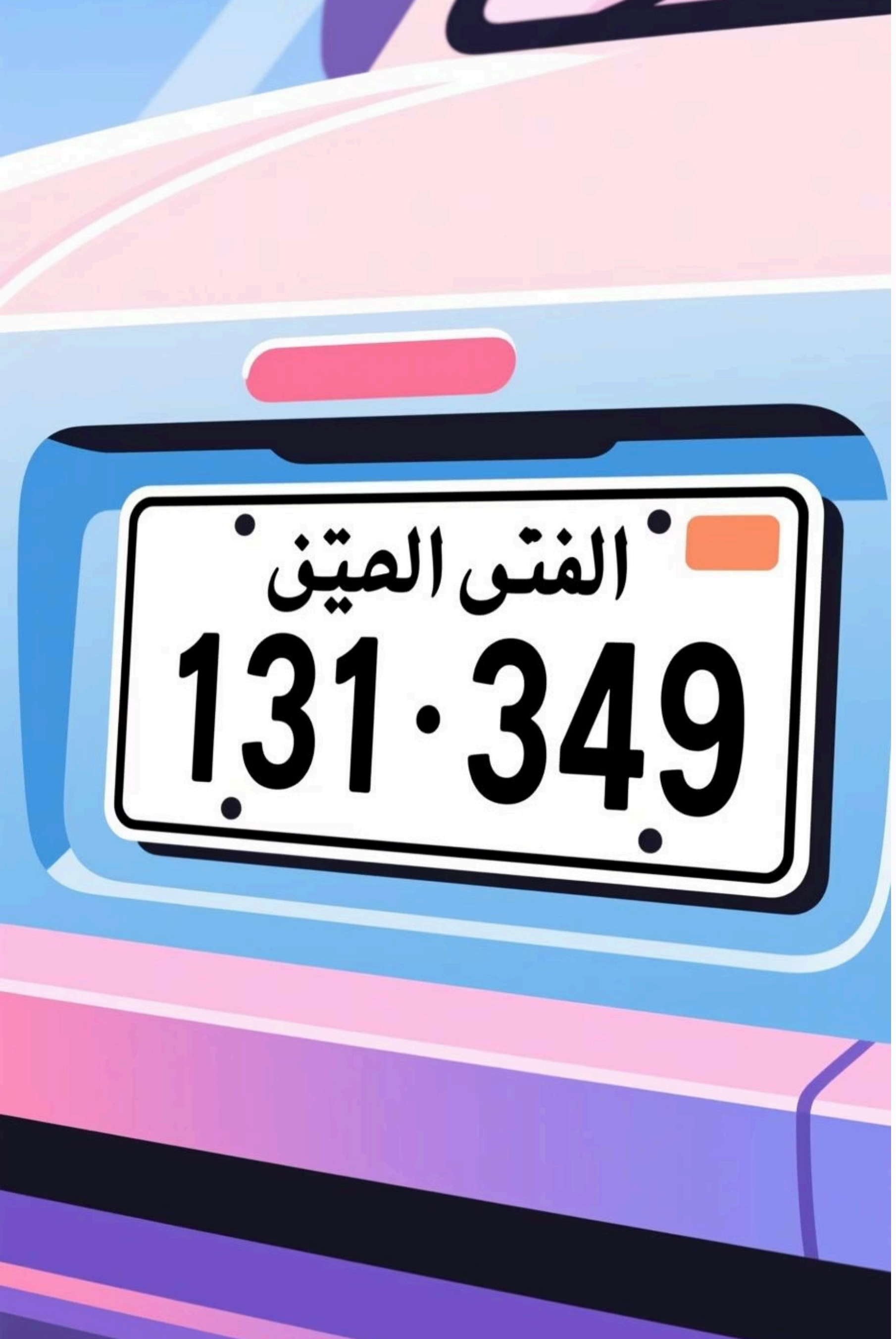


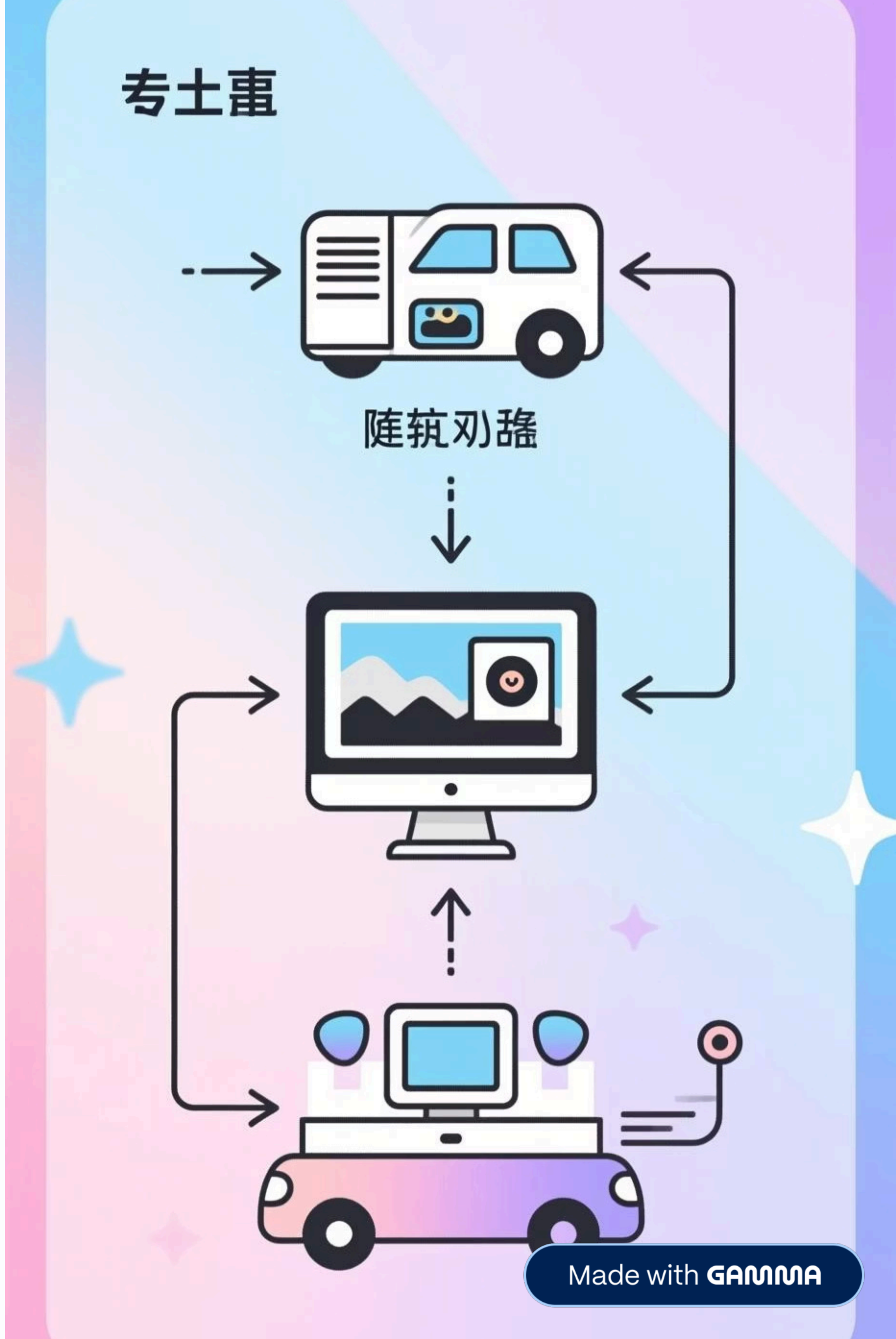
Arabic License Plate Recognition System OCR Smart Compounds Entry.

An automated system that detects and recognizes Arabic license plates from vehicle images, combining object detection and character classification to extract and reconstruct plate text ,Recognition System End-to-End OCR Pipeline Using YOLOv5 and ResNet18. to Use it For Smart Compounds.

Made with GAMMA

Project Overview

This system automates the detection and recognition of Arabic license plates from vehicle images. It combines object detection and character classification to extract and reconstruct plate text. The pipeline is modular, scalable, and ready for real-world deployment.



Made with GAMMA

Core Pipeline Components

1

License Plate Detection

- YOLOv5 model trained on Arabic plate annotations
- Detects and crops plate region from vehicle image

2

Character Cropping

- Manual cropping with filename encoding (plateID-char-xpos)
- Future upgrade: YOLOv5 model for character detection

3

Character Classification

- ResNet18 trained on Arabic characters
- Uses grayscale 64×64 crops and custom classes.txt

4

Text Reconstruction

- Characters sorted by horizontal position
- Reconstructed right-to-left for Arabic format

Made with GAMMA

Achievements So Far

✅ YOLOv5 Model Trained and Integrated

✅ ResNet Classifier Trained and Working

✅ Full Pipeline Implemented and Tested

✅ Character Predictions Accurate and Varied

✅ Debug Images and Plate Text Overlays Generated

Made with GAMMA

Tools & Technologies

Purpose	Tool / Library
Object Detection	YOLOv5 (PyTorch)
Classification	ResNet18 (Torchvision)
Image Processing	OpenCV, PIL
Data Handling	Python, Pathlib, OS
Model Training	PyTorch
Visualization	OpenCV, Matplotlib

Made with GAMMA

Backend Architecture (Expected)

API Layer

FastAPI or Flask – RESTful endpoints for image upload & results

Model Serving

TorchServe or ONNX – Efficient deployment of YOLOv5 and ResNet

Data Storage

SQLite or PostgreSQL – Store plate logs, images, predictions

File Handling

Local or S3 – Save cropped plates and character images

Made with GAMMA

Frontend Architecture (Expected)

UI Framework

React or Vue.js – Interactive dashboard for uploads & results

Styling

Tailwind CSS – Clean, responsive design

Image Preview

HTML5 Canvas – Show detection boxes and plate text

API Integration

Axios or Fetch – Communicate with backend endpoints

Made with GAMMA

Next Steps

→ Train YOLOv5 Model for Character Detection

→ Automate Cropping and Feed Into Classifier

→ Expand and Balance Character Dataset

→ Retrain ResNet Classifier With Improved Data

→ Build Web Interface for Uploads and Results

→ Package and Document the Full System

Made with GAMMA

Time Estimation

Task	Time Estimate
Character Detection Model	5–7 days
Auto-cropping Integration	2–3 days
Dataset Expansion	3–5 days
Model Retraining	2–4 days
Web Interface (Optional)	5–7 days
Documentation & Packaging	2 days

Total Estimated Time: 3–4 weeks

Made with GAMMA

Final Deliverables & Team

Deliverables

- Python-based OCR pipeline
- Trained YOLOv5 and ResNet models
- Character detection module
- Sample input/output images
- Web interface (optional)
- Documentation and proposal

Team Members

- **Ashraf Mohamed** – Lead Developer, Model Trainer
- **Mohammed Momen** – Model Tester, Data Presentation
- **Omar Shapan** – Debugging, Planning, Data Collection
- **Pipeline AI Assistant** – Architecture

Made with GAMMA