Alright — I’ll give you **clear developer-style documentation** for:

* **main.py** (detection, saving results, DB updates, image annotation)
* **view\_history.py** (view/clear history)
* **Database schema**
* **Recent updates**
* **README.md** (so you can drop it in your project folder)

**📄 Documentation**

**1. main.py**

**Purpose:**  
Detect number plates from an image using YOLOv5 + EasyOCR, annotate them, and store detections in a central SQLite database on your Desktop.

**Key Features & Flow:**

1. **Model Loading**
   * Loads YOLOv5 custom model from weights/best.pt.
   * Runs detection on the provided image (samples/test.jpg by default).
2. **OCR (EasyOCR)**
   * Adds padding to detected regions for better recognition.
   * Converts detected plate text to uppercase.
3. **Database Handling**
   * Uses a **single persistent SQLite DB** at:
   * C:\Users\<USERNAME>\Desktop\plates.db
   * Table schema:
   * CREATE TABLE IF NOT EXISTS plates (
   * id INTEGER PRIMARY KEY AUTOINCREMENT,
   * plate\_text TEXT,
   * timestamp TEXT,
   * image\_path TEXT
   * )
   * Automatically **adds image\_path column** if missing.
4. **Image Saving**
   * Annotated images saved in output/ folder.
   * Filenames include timestamp:
   * plate\_YYYYmmdd\_HHMMSS.jpg
   * Image path stored in DB alongside detection.
5. **Console Output**
   * Prints detected plate, timestamp, and DB path after every detection.

**2. view\_history.py**

**Purpose:**  
View or clear stored plate detection history without needing to open SQLite Browser.

**Key Features:**

1. **View Mode (default)**
   * Prints DB path first.
   * Displays all records sorted **newest first** by timestamp.
   * Shows:
   * ID | Timestamp | Plate Text | Image Path
   * Allows you to see which annotated image corresponds to each detection.
2. **Clear Mode**
   * Run:
   * python view\_history.py --clear

This **deletes all rows** but **keeps the database file** intact.

**3. plates.db (Database Schema)**

* **id**: Auto-incremented primary key.
* **plate\_text**: Detected plate string (uppercase).
* **timestamp**: Detection time in YYYY-MM-DD HH:MM:SS format.
* **image\_path**: Path to annotated image saved during detection.

**4. Updates Implemented**

✅ Central DB on Desktop for easy access.  
✅ DB auto-schema migration to add image\_path column.  
✅ Annotated image saving with unique filenames.  
✅ Image path stored in DB for linking detections to images.  
✅ History viewer script with --clear option.  
✅ Records sorted newest first in history view.  
✅ Consistent DB path print in both scripts.

**📄 README.md**

# Number Plate Detection App

This project detects number plates from images using YOLOv5 + EasyOCR, annotates them, and stores results in a persistent SQLite database on your Desktop.

## Features

- Detect plates using YOLOv5 and EasyOCR

- Store detections in a central SQLite DB (`plates.db` on Desktop)

- Save annotated images with unique names

- Link each detection to its saved annotated image

- View detection history directly from the terminal

- Clear history without deleting the database file

## Requirements

- Python 3.8+

- PyTorch

- OpenCV

- EasyOCR

- SQLite (built into Python)

- YOLOv5 model weights (`weights/best.pt`)

## Installation

```bash

pip install torch torchvision torchaudio

pip install opencv-python easyocr

Place your YOLOv5 weights in weights/best.pt.

**Usage**

**Detect Plates**

Run:

python main.py

This will:

1. Detect plates from samples/test.jpg
2. Save annotated image in output/
3. Store detection in plates.db on your Desktop

Console output will show:

* Detected plate text
* Timestamp
* DB path

**View History**

python view\_history.py

Shows all detections (newest first) with:

* ID
* Timestamp
* Plate Text
* Annotated Image Path

**Clear History**

python view\_history.py --clear

Deletes all records but **keeps** the plates.db file.

**Database Schema**

id INTEGER PRIMARY KEY AUTOINCREMENT

plate\_text TEXT

timestamp TEXT

image\_path TEXT

**File Overview**

* main.py — Detection, annotation, DB storage
* view\_history.py — View or clear stored detections
* plates.db — Persistent SQLite database stored on Desktop
* output/ — Annotated images
* samples/ — Sample test images
* weights/ — YOLOv5 model weights

---