System Architecture Implementation Plan

Becker Books OCR Accelerator

Adam Nelson-Archer

June 25, 2025

Project Documentation
A supplementary system for BookTrakker

System Overview

This document provides supplementary material for the Becker Books Occular Character Recognition (BBOCR) system architecture, focusing on the BookTrakker-specific implementation. The following pages contain the system diagram and a detailed breakdown of each component.

Diagram Structure: Swimlanes

The diagram is structured into five horizontal swimlanes. These represent the major areas of interaction in this project on a high level.

- I. Input Layer
- II. Accelerator Service
- III. Book Trakker
- IV. Data Persistence
- V. External Data Sources

Color Coding: Project Phases

The items are broken into three phases, represented by color , to break the imaging and integration process into easily referenceable parts.

Blue: Represents the "Image to Structured Data" phase.

Orange: Represents the "Search, Enrich, Verify" phase.

Red: Represents the "Finalize and Integrate with the Existing" phase.

Component Shapes

Diamond Boxes: Represent decision points.

Round Boxes (Dotted Outlines): Represent data that is being passed between two functions or models.

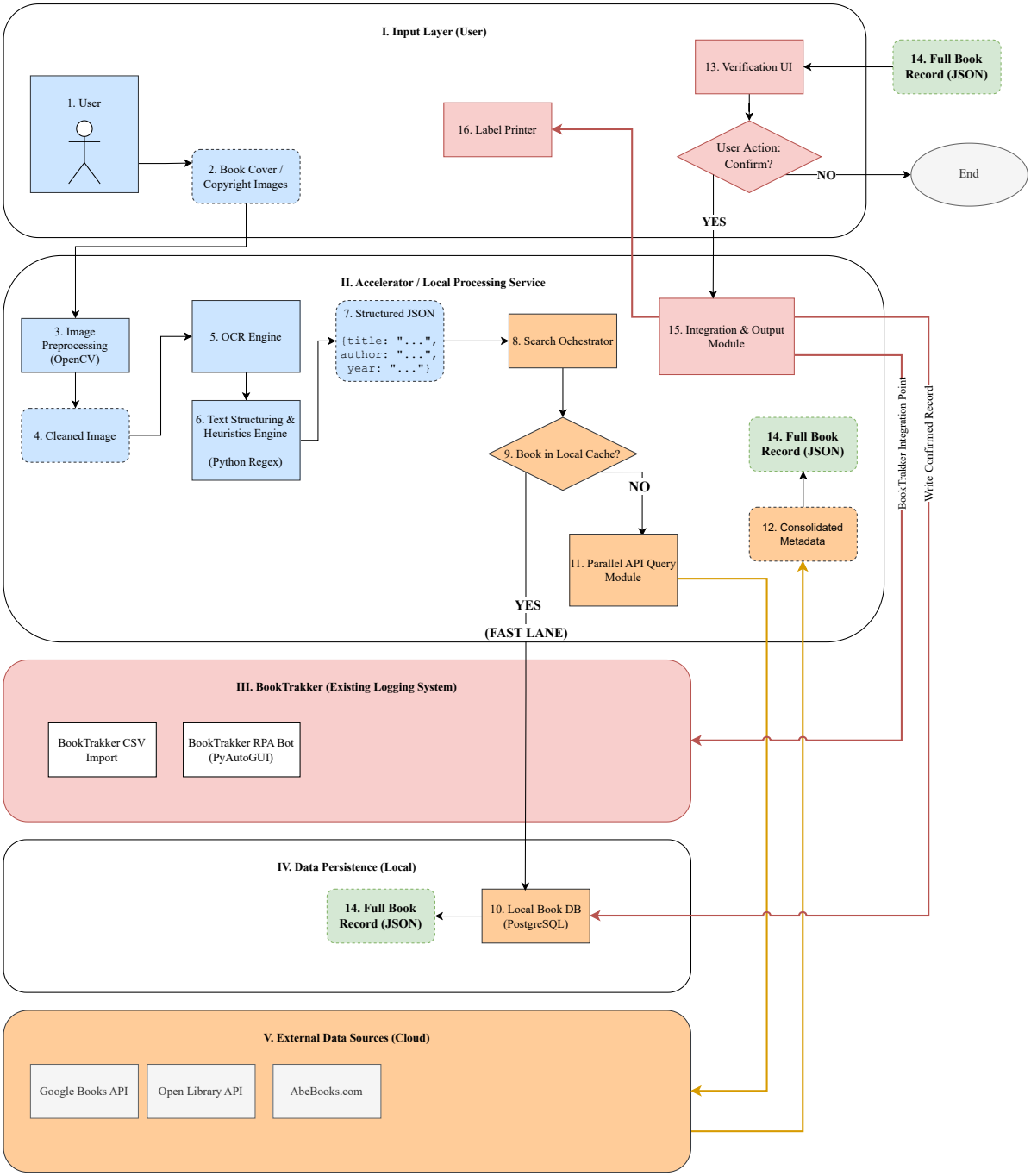

Figure 1: System Architecture Diagram for the BookTrakker OCR Accelerator

Component Breakdown

The following provides context or additional information for each numbered element in the diagram:

1. User: The process begins with a physical book that does not have an ISBN.

2. Book Cover / Copyright Images: The user captures two images using a simple UI.

3. Image Preprocessing: Uses OpenCV (Python) to automatically crop borders, deskew/straighten the image, and enhance contrast.

4. Cleaned Image: Represents the data output from the preprocessing step.

5. OCR Engine: A local engine (likely EasyOCR or Tesseract 5) performs Optical Character Recognition, extracting visible text from both images.

6. Text Structuring & Heuristics Engine: A key 'intelligent' step that parses raw text using rules (e.g., 'longest line at top is title', 'line with © is copyright info') to identify the most probable Title, Author, and Publication Year.

7. Structured JSON: The data output from the heuristics engine.

8. Search Orchestrator: This component manages the data enrichment workflow, checking the local database before querying external APIs.

9. Book in Local Cache?: This is a logical check for the presence of a book in the local cache. If using an API-only approach, the answer is always NO. A YES answer represents a found book in a local database.

10. Local Book DB: A PostgreSQL or SQLite database that stores a full record of every book previously confirmed by the user.

11. Parallel API Query Module: Queries multiple external sources simultaneously to find matching book metadata. It will likely use a fuzzy string matching library (e.g., FuzzyWuzzy) to account for OCR errors.

12. Consolidated Metadata: The data output from the API query module.

13. Verification UI: The 'Human-in-the-Loop' step where a user confirms book data and price before entry into BookTrakker.

14. Full Book Record (JSON): This represents the full, confirmed book data. It is derived from either the local database (10) or the consolidated metadata (12), and is then fed into the verification UI (13).

15. Integration & Output Module: Upon user confirmation, this module performs three parallel actions: writing to the local cache, generating input for BookTrakker, and printing the label.

16. Label Printer: Prints a label to be placed on the physical book, finishing the labeling process in the analog domain.

BookTrakker Integration Methods

The two cards within lane III are the two likely methods for importing data into BookTrakker. The model will be built to output data in a way that BookTrakker can understand.