**A. Must-Have Features (Core for Prototype)**

These are non-negotiable — implement these first so you can demo a working pipeline:

**1. OMR Evaluation Core (FastAPI service)**

* Upload scanned/captured OMR sheet (JPG/PNG).
* Preprocess image:
  + Resize + grayscale.
  + Detect corners + warp (perspective correction).
* Extract bubbles using **predefined template coordinates** (JSON per version).
* Detect filled bubbles (using threshold + fill ratio).
* Match detected answers against **answer key** (stored in DB/JSON).
* Compute per-subject scores (20 each × 5 subjects).
* Generate total score.
* Save results to DB and return JSON.

**2. Evaluator Web App (Streamlit)**

* Upload page (drag-and-drop multiple sheets).
* Results table (student ID, per-subject scores, total).
* Export results (CSV/Excel).
* Dashboard (aggregate stats: avg score, per-subject histogram).
* Flag review UI:
  + Show processed OMR sheet overlay.
  + Show flagged questions (ambiguous/no mark).
  + Allow evaluator to correct and re-score.

**3. Database (SQLite/Postgres)**

* Store exam details (exam\_id, answer keys).
* Store student sheet results (answers, scores, flags, file paths).
* Store audit logs for corrections.

**B. Nice-to-Have Features (If time permits)**

These add polish but aren’t strictly required for hackathon demo:

* Multiple sheet versions (detect version code, load template accordingly).
* Ambiguity classifier (tiny ML model for faint marks).
* Authentication for evaluator dashboard.
* Storage of overlay images for transparency.
* Real-time processing progress bar in Streamlit.
* Auto-detect student ID from OMR sheet (OCR/Tesseract).

**C. Features Checklist (mapped to hackathon description)**

✅ **Capture**

* Upload OMR sheets via Streamlit frontend (mobile photo → drag-drop).
* API accepts image and exam metadata.

✅ **Preprocessing**

* Rotation, skew, illumination correction (OpenCV pipeline).
* Perspective rectification using sheet corners.

✅ **Bubble Detection & Evaluation**

* Bubble grid defined via **JSON template** per sheet version.
* Filled detection using adaptive threshold + fill ratio.
* Handle multiple/no marks → flag for review.

✅ **Answer Key Matching**

* Compare detected answers with stored answer key (DB/JSON).
* Compute per-subject (5×20) and total score.

✅ **Result Generation**

* Return JSON with per-question answers, scores, and flags.
* Store in DB with student\_id, exam\_id, sheet\_version.

✅ **Web Application**

* Streamlit UI to upload, view results, and flagged sheets.
* Dashboard with summary stats and export option.
* Manual review interface (change answers → recalc).

✅ **Storage & Audit Trail**

* Store processed overlays and warped sheet images.
* Record corrections (audit logs).

automated-omr-system/

│

├── backend/ # FastAPI backend

│ ├── api/ # API endpoints

│ │ ├── omr.py # Upload & process OMR sheets

│ │ ├── results.py # Fetch & export results

│ │ └── auth.py # Simple user login/auth

│ │

│ ├── core/ # Core config

│ │ └── config.py # DB + app settings

│ │

│ ├── db/ # Database setup

│ │ ├── models.py # All SQLAlchemy models (User, Exam, Result)

│ │ ├── schemas.py # Pydantic schemas

│ │ └── session.py # MySQL session setup

│ │

│ ├── services/ # Business logic

│ │ ├── omr\_service.py # Image preprocessing + bubble detection

│ │ ├── scoring\_service.py # Scoring + answer key matching

│ │ └── export\_service.py # Export results to CSV/Excel

│ │

│ ├── utils/ # Helper functions

│ │ ├── image\_utils.py # Skew correction, perspective fixes

│ │ └── logger.py # Logging setup

│ │

│ └── main.py # FastAPI entry point

│

├── frontend/ # Streamlit frontend

│ ├── pages/

│ │ ├── 1\_Upload\_Sheets.py # Upload scanned OMR sheets

│ │ ├── 2\_Results\_Dashboard.py # Dashboard (subject-wise + total scores)

│ │ ├── 3\_Review\_Flagged.py # Review ambiguous/flagged sheets

│ │ └── 4\_Export\_Reports.py # Export CSV/Excel

│ │

│ ├── utils/

│ │ └── api\_client.py # Call FastAPI endpoints from Streamlit

│ │

│ └── app.py # Streamlit entry point

│

├── data/ # Sample data

│ ├── omr\_samples/ # Example OMR sheets

│ ├── answer\_keys/ # Predefined answer keys

│ └── results\_exports/ # Generated results

│

├── requirements.txt # Python dependencies

├── README.md # Project overview

└── .env # Environment variables (DB creds, secrets)