**Features to Implement in This Project**

**1. Waste Detection & Classification**

* Detect and classify waste from **images** (plastic, paper, glass, metal, organic, cardboard, trash).
* Support **video input** (basic frame sampling to classify waste in videos).
* Return **confidence score** for each prediction.

**2. Personalized Educational Content**

* For every detected item, display:
  + **Eco-friendly disposal/recycling instructions** (region-agnostic but clear).
  + **Short educational tip** (why recycling this matters).
* Provide content in **both text + audio (TTS)** formats.

**3. Interactive Chatbot**

* A chatbot to answer **waste management & recycling queries**.
* Example: *“Can I recycle pizza boxes?” → “Yes, only if clean and grease-free.”*
* Fallback to predefined FAQ + LLM for more flexibility.

**4. Gamification & User Engagement [Optional]**

* **Points & badges system** for users when they sort correctly.
* Local **leaderboard / progress tracker** (simple in prototype, scalable later).
* Daily **eco-challenge tips** to keep users engaged.

**5. User Feedback Loop (Self-Improvement) [Optional]**

* Users can mark model predictions as **Correct / Incorrect**.
* Store misclassified images + corrections to improve dataset.
* Lays foundation for **continuous learning / retraining**.

**6. Voice & Multimodal Support**

* **Text-to-Speech (TTS)** for disposal instructions → accessibility friendly.
* **Voice queries** for chatbot (speech-to-text integration if time permits).

**7. Analytics & Admin Dashboard**

* Track system usage:
  + Number of images classified.
  + Most common waste type detected.
  + Accuracy trends (based on feedback).
* Simple **dashboard** for judges/admins to see real-time metrics.

**8. Deployment & Accessibility**

* **Web app** (React frontend + FastAPI backend).
* Mobile-responsive UI (upload via camera/phone).
* Containerized with **Docker** → deployable on cloud (Google Cloud Run / Runpod).

**🌟 Additional Unique Features (Pitch Value) [Optional]**

* **Roadmap to AR bins** (prototype mockup only).
  + Users point phone → AR highlights which bin to use.
* **Integration with Rewards / Tokens** (future expansion).
  + Users could redeem eco-points for vouchers.

backend/

│── app/

│ │── \_\_init\_\_.py

│ │── main.py # FastAPI entrypoint (create app, include routers, middleware)

│ │── config.py # Settings (env vars, API keys, DB URLs)

│ │── dependencies.py # Shared dependencies (auth, DB session, etc.)

│ │

│ ├── api/ # API routes grouped by feature

│ │ │── \_\_init\_\_.py

│ │ │── v1/ # Versioned API (v1 for hackathon)

│ │ │ │── \_\_init\_\_.py

│ │ │ │── predict.py # /predict (YOLO image/video classification)

│ │ │ │── chatbot.py # /chat (chatbot queries)

│ │ │ │── feedback.py # /feedback (store user feedback + corrections)

│ │ │ │── analytics.py # /analytics (admin dashboard data)

│ │ │ │── gamification.py# /points, /leaderboard

│ │ │ │── tts.py # /tts (generate audio instructions)

│ │

│ ├── core/ # Core utilities (logging, exceptions, middleware)

│ │ │── \_\_init\_\_.py

│ │ │── logging.py

│ │ │── security.py # (if auth needed later)

│ │ │── exceptions.py

│ │

│ ├── models/ # Database models (SQLAlchemy ORM or Pydantic models)

│ │ │── \_\_init\_\_.py

│ │ │── feedback.py # Feedback data model

│ │ │── user.py # User profile / gamification points

│ │ │── leaderboard.py # Leaderboard schema

│ │

│ ├── schemas/ # Pydantic request/response models

│ │ │── \_\_init\_\_.py

│ │ │── predict.py

│ │ │── chatbot.py

│ │ │── feedback.py

│ │ │── analytics.py

│ │ │── gamification.py

│ │ │── tts.py

│ │

│ ├── services/ # Business logic layer

│ │ │── \_\_init\_\_.py

│ │ │── yolo\_service.py # Load YOLOv8/YOLOv12, run inference

│ │ │── chatbot\_service.py # Handle chatbot logic (LLM API + fallback)

│ │ │── feedback\_service.py# Store & retrieve feedback

│ │ │── analytics\_service.py# Collect system metrics

│ │ │── gamification\_service.py# Points, badges, leaderboard logic

│ │ │── tts\_service.py # Generate TTS audio files

│ │

│ ├── utils/ # Helper functions

│ │ │── \_\_init\_\_.py

│ │ │── file\_utils.py # Save images/audio, sanitize filenames

│ │ │── video\_utils.py # Frame extraction for video input

│ │ │── augmentation.py # (future) augment images for retraining

│ │

│ ├── database/ # DB setup (SQLAlchemy/Firebase integration)

│ │ │── \_\_init\_\_.py

│ │ │── session.py # DB session (SQLAlchemy or Firestore client)

│ │ │── migrations/ # (if SQL DB, Alembic migrations)

│ │

│ ├── static/ # For saving generated files (audio, temporary images)

│ │ │── audio/ # TTS mp3/wav files

│ │ │── uploads/ # Uploaded images/videos

│ │

│ └── tests/ # Unit & integration tests

│ │── \_\_init\_\_.py

│ │── test\_predict.py

│ │── test\_chatbot.py

│ │── test\_feedback.py

│ │── test\_gamification.py

│ │── test\_tts.py

│

│── requirements.txt # Python dependencies

│── Dockerfile # Containerization for deployment

│── docker-compose.yml # (Optional, if multi-service setup with DB)

│── .env # Environment variables (DB URL, API keys, secrets)

│── README.md # Project documentation

frontend/

│── public/

│ │── index.html # Main HTML file

│ │── favicon.ico

│

│── src/

│ │── main.jsx # React entrypoint

│ │── App.jsx # Main app component, routes

│ │── index.css # Tailwind base styles

│ │

│ ├── components/ # Reusable UI components

│ │ │── Navbar.jsx

│ │ │── Footer.jsx

│ │ │── UploadCard.jsx # Image/Video upload UI

│ │ │── PredictionCard.jsx # Show classification result + confidence

│ │ │── InstructionCard.jsx# Show disposal instructions

│ │ │── Chatbot.jsx # Chatbot widget

│ │ │── FeedbackButtons.jsx# Correct/Incorrect buttons

│ │ │── PointsBadge.jsx # Gamification points UI

│ │ │── Leaderboard.jsx # Leaderboard UI

│ │ │── DashboardCharts.jsx# Analytics charts (Recharts)

│ │

│ ├── pages/ # Main app pages

│ │ │── Home.jsx # Upload + classification page

│ │ │── Chat.jsx # Dedicated chatbot page

│ │ │── Dashboard.jsx # Analytics dashboard (for judges/admin)

│ │ │── LeaderboardPage.jsx# Leaderboard page

│ │

│ ├── api/ # API interaction layer

│ │ │── axiosClient.js # Pre-configured axios instance

│ │ │── predict.js # Calls /predict endpoint

│ │ │── chatbot.js # Calls /chat endpoint

│ │ │── feedback.js # Calls /feedback endpoint

│ │ │── analytics.js # Calls /analytics endpoint

│ │ │── gamification.js # Calls /points, /leaderboard

│ │

│ ├── context/ # Global state management (if needed)

│ │ │── UserContext.jsx # Track user points, session, etc.

│ │

│ ├── hooks/ # Custom React hooks

│ │ │── usePredict.js # React Query hook for predictions

│ │ │── useChat.js # Hook for chatbot interaction

│ │ │── useFeedback.js # Hook for feedback handling

│ │ │── useGamification.js # Hook for points/leaderboard

│ │

│ ├── assets/ # Static assets (icons, images, mockups)

│ │ │── logo.png

│ │ │── mock-ar.png # For AR roadmap slide/demo

│ │

│ └── utils/ # Utility functions

│ │── formatDate.js

│ │── constants.js # API base URL, class labels, etc.

│

│── .env # Environment variables (API base URL)

│── package.json

│── tailwind.config.js

│── vite.config.js

│── README.md