Got it 👍 You want this **upgrade plan simplified into clear, easy-to-review points** (no fluff). Here it is, broken down cleanly.

1. Goal (What we are building)

- Convert the **CLI-based Intelligent Bug Detection prototype** into a **large-scale web platform**
- Separate logic into:
  - **Backend API (Python)**
  - **Frontend Dashboard (Web UI)**
- Add **Automated Bug Repair** using LLMs (detect + suggest fixes)

2. User Review / Requirements (Important)

- **New Backend Dependencies**
  - `fastapi`
  - `uvicorn`
- **New Frontend Stack**
  - Node.js
  - Next.js
- **LLM API Key Required**
  - Needed for Automated Bug Repair (e.g., GPT-4)
  - Provide:
    - Input field for API key **OR**
    - Mock / local fallback option

3. Backend Changes (Python + FastAPI)

New File

- `server.py`
  - FastAPI application
  - Exposes REST endpoints

API Endpoints

- `POST /analyze`
  - Input: folder path
  - Runs analysis pipeline asynchronously
  - Returns: `job_id`
- `GET /status/{job_id}`
  - Returns current analysis progress
- `GET /results/{job_id}`

    - Returns final JSON bug report
- `POST /repair`
    - Input: bug description + code
    - Output: suggested fix (via LLM)

Modified File

- `main.py`
    - Refactor `run_pipeline`
    - Must be:
        - Importable
        - Callable without side effects
        - Return structured data (JSON-friendly)

4. Frontend Changes (Next.js Dashboard)

New Folder

- `dashboard/`

Tech Stack

- Next.js
- TailwindCSS
- Framer Motion (animations)
- Recharts (charts & graphs)

Pages & Features

- **Home Page**
    - Select / enter target project path
- **Dashboard Page**
    - Overview metrics:
        - Bugs found
        - Code clones
        - Severity counts
    - **Sunburst Chart**
        - Visualize code clones interactively
    - **Bug Feed**
        - List of bugs with severity badges
    - **File View**
        - Code editor with highlighted issues

    - “Repair” button for each bug

5. Automated Bug Repair (LLM)

- Integrate LLM for fix suggestions
- Options:
  - OpenAI API
  - Local transformer model (if feasible)
  - Or structured prompt generator (fallback)
- Output:
  - Suggested code fix
  - Explanation of change

6. Verification Plan

Automated Testing

- **Backend**
  - Run: `uvicorn server:app`
  - Use `curl` to test:
    - `/analyze`
    - `/status`
    - `/results`
- **Frontend**
  - Run: `npm run dev`
  - Open: `http://localhost:3000`
  - Trigger analysis on sample project

Manual Testing (User Flow)

- Open web dashboard
- Enter project path
- Track progress bar
- View:
  - Charts
  - Bug list
- Click **Repair**
- See LLM-generated fix

If you want, next I can:

- Convert this into a **1-page architecture diagram**
- Turn this into a **project proposal / design document**

- Break it into **Week-wise implementation plan**