AMOS #2 Backup Data Aanalyzer software-architecture

Technology Stack

Frontend:

- Angular (TypeScript)

Backend:

- Node.js with NestJS (TypeScript)
- Python with Flask (for specialized backend functionalities, e.g., AI integrations or specialized APIs)

Database:

- PostgreSQL

Monorepo Management:

- **NX** will be used to manage all apps (Frontend, Node backend, Flask backend) within a single monorepo.

Containerization

- **Docker containers** for each component (Frontend, Node backend, Flask backend, and Database).
- **Docker Compose** (docker-compose up) will be used to launch the entire application with a single command. Each component will have its own service in the Compose setup.

AI Integration

Option 1:

- A dedicated Docker container for AI, which receives data via API from the backend (Node or Flask), processes it, and returns the results. Python and TensorFlow are potential tools for implementing and running the model.

Option 2:

- A custom-trained AI model hosted on an Nvidia GPU instance (e.g., AWS server).
- Communication with the backend (Node or Flask) would also occur via API.

Further Research:

- Additional options and configurations will be added here as the research on AI integration progresses, to ensure flexibility and scalability in AI functionality.

Deployment and Environments

- The web app will be hosted online with two separate environments:
    - **DEV environment**: For development and testing.
    - **PROD environment**: For production use.