# **Development Choices**

**1**. **Tech Stack:**

* **Frontend:** React.js with Material-UI (MUI) for a modern, responsive UI.
* **Backend:** Node.js with Express.js for scalable API development.
* **Database:** PostgreSQL for relational data management and data integrity.
* **Authentication:** JSON Web Tokens (JWT) used to secure access to protected routes.
* **Validation & Forms:** Form validation on both client and server for robustness.

**2. Libraries/Tools Used:**

* axios: For HTTP requests from frontend to backend and to Hugging Face API.
* dotenv: To manage environment variables securely.
* Yup+formik: For clean and declarative input validation.
* pg: PostgreSQL client for Node.js.
* huggingface/inference: For AI-based claim description classification.

**3. Design Patterns**

* **Modular Structure:** Backend and frontend separated cleanly.
* **Controller-Service Pattern:** Keeps route logic clean and reusable.
* **Environment-based Configuration:** All credentials and secrets are externalized.

**4. Improvements if Given More Time**

* Add unit & integration tests using Jest and Supertest.
* Improve UX with loading states, better form error handling.
* Add fine-grained role-based access control (Admin,, User).
* Set up CI/CD pipelines (e.g., GitHub Actions or Jenkins).

## **5. AI Integration**

* Model Used: facebok/bart-large-mnli from Hugging Face.

## **How It Works:**

* The model is prompted with a claim description and asked to classify it against predefined labels:
* insurance claim, scam, incomplete, unclear, detailed
* It returns confidence scores for each label.
* The system uses thresholds to determine whether the claim is valid enough to be submitted.

## **Limitations:**

* The model is zero-shot, meaning it tries to match new text to the provided labels without training — it's not always perfect.
* Confidence scores are context-sensitive and may misclassify niche claim types.
* The AI might favor more verbose and detailed descriptions.

## **Future Improvements:**

* Use advanced models from Open AI using real insurance claim data.
* Use fine-tuning to adapt the model to insurance-specific context.
* Add a feedback loop where rejected claims are reviewed by admins to retrain the model over time.
* Add natural language feedback suggestions (e.g., “Please mention the incident date and location.”)