Question 1 – Dispute Handler

Explanation to the different Node.js files in the folder and their functionality:

* AIClassifier.js – Sets up server, run configurations, requests from local host. Essentially the main file to run as it creates objects of the other 2 files within it.
* DisputeClassifier.js – The AI/ML logic to classifying disputes based on a learning model.
* Training-data.js – A file used to retrieve data from the mentioned source.
* PriorityAssigner.js – Redirecting and labelling disputes.
* mockDB.js – The low-code database to store intermediate data.
* Index.html – A simple UI to accept user inputs.
* Styles.css – A css file adding décor to the UI

How to run:

-- 1) Clone repo with git clone and cd into directory

-- 2) Create folder if it doesnt already exist

mkdir dispute-handler

cd dispute-handler

-- 3) Initialize node project with npm init -y

-- 4) Install dependencies as given in the .txt file or run

npm install express body-parser cors natural @tensorflow/tfjs node-fetch@2 csv-parser

-- 5) Start the server/ run file with node AIClassifier.js

-- 6) Open local host http://localhost:3000 in a browser

# Question 1.3 a & b:

My approach to building the workflow:

Essentially customers submit transaction disputes through a web interface. The system captures essential details (customer ID, amount, reason, timestamp) via a RESTful. Upon submission, the data flows through an ML-based classifier (built with TensorFlow.js and trained on CFPB data) that categorizes the dispute into types like "Fraud," "Billing Error," or "Service Issue." The classifier analyzes the dispute text using natural language processing, while the priority assignment module evaluates customer history and dispute type to determine urgency. The API then returns a comprehensive response including dispute category, priority level, recommended actions, and team assignments, which is displayed to the user in real-time through the interface.

1.3 a) A low-code tool can be used for the front-end which makes it more time friendly and UI-enhanced. Tools like Microsoft Power Platform, Bubble.io, etc are easy to use and implement.

AI service integration - I would also integrate the interface with a trusted LLM API service like OpenAI (provided it is acceptable within company in terms of privacy and cost). This would provide detailed dispute analysis and classification. It could also give users examples and situation based reasoning in case of High or Low level frauds.

Both the low-code and AI-integration would require little to almost no-code since it is essentially prompt engineering with API calls. However, it would incur additional cost to the company.

1.3 b) With increased cases of bank frauds globally, the prototype can use AI-Assistance for providing users urgent-care based on the dispute level. Assistance for next-steps for the users based on compliance and law. Assistance for a chatbot to interact with disputed users. AI can also optimize the code can incorporate other data sources to improve classification. Provide resources to users to help curb bank traffic.