# AI-Powered Chatbot Order Status Service: End-to-End Code Challenge

## Problem Description

I am creating a custom chat app to interact with my clients. The chatbot must be polite, maintain focus on delivering an excellent customer experience, and provide accurate, specific information about the chosen product. Each team should select a different product for their chatbot to support (e.g., a SaaS analytics tool, an e-commerce platform, or a fleet-management system).

## Part 1: User Stories & MVP Definition

Objective: Write clear, actionable user stories and map them to REST endpoints.

* - Deliverable: `user-stories.md` with 4–5 stories. Each story must include: persona, story statement, benefit, acceptance criteria, and mapped endpoint (e.g., `GET /orders/{id}`).

## Part 2: Project Kick-off & Scaffolding

Objective: Bootstrap your service using an AI assistant.

* - Scaffold a new web-service project (Node.js/Express, Python/Flask, etc.).
* - Generate directory layout, sample routes, and a basic `README.md`.
* - Commit to Git with AI-generated commit messages.
* - Deliverable: Git repo with initial scaffold.

## Part 3: Authentication & Session Management

Objective: Secure your chatbot with user/password login.

* - Implement user registration and login endpoints (`POST /users/register`, `POST /users/login`).
* - Issue a JWT or session cookie on successful login.
* - Protect the `/chat` endpoint so only authenticated users can access it.
* - Deliverable: Auth endpoints plus middleware guarding `/chat`.

## Part 4: Building the Core Chat API

Objective: Implement the `/chat` endpoint and integrate with an LLM API.

* - Endpoint accepts user messages and returns AI-generated replies.
* - Ensure chatbot persona is polite and customer-focused.
* - Handle errors and rate limits gracefully.
* - Deliverable: Working `/chat` endpoint with authentication applied.

## Part 5: Simulating RAG (Knowledge Base Retrieval)

Objective: Augment replies with product context.

* - Create a JSON/YAML knowledge base of FAQs or product documentation for your chosen product.
* - Implement retrieval logic to fetch relevant KB entries at query time.
* - Append retrieved context to the LLM prompt.
* - Deliverable: Code showing retrieval and example contextual chats.

## Part 6: Automated Testing

Objective: Generate and run unit/integration tests.

* - Prompt AI to draft tests that mock the LLM, auth flows, and retrieval logic.
* - Cover positive and negative cases (invalid login, missing auth header, unknown product query).
* - Deliverable: Test suite runnable via `npm test`/`pytest`.

## Part 7: CI/CD Pipeline Setup

Objective: Automate build, test, and deploy.

* - CI config (e.g., GitHub Actions) to checkout code, install deps, run tests, build a Docker image, and push to a registry.
* - Deliverable: `.github/workflows/ci.yml` or equivalent.

## Part 8: Cloud Deployment & Env Configuration

Objective: Deploy your containerized chatbot to the cloud.

* - Use Kubernetes manifests or serverless definitions (AWS Lambda + API Gateway, etc.).
* - Configure environment variables for API keys and KB path.
* - Validate the live `/chat` endpoint URL (authenticated access).
* - Deliverable: Deployment configs and live endpoint link.

## Bonus Extensions

* - Swap JWT for OAuth2 or SSO integration.
* - Implement a front-end chat widget that handles login and chat sessions.
* - Add observability: logs, metrics, and a health check endpoint.

## Delivery Date: August 1

## Deliverable: Code repository url, chatbot repository

## Deliver to: latinamericalearningdevelopment@perficient.com