MICHIGAN STATE UNIVERSITY

Project Plan Presentation

Seller Agent Management Platform (SAMP)

The Capstone Experience

Team Amazon

Ziad Bakki
Jiwoo Jeong
Ethan Tunney
Meet Patel
Daniel Berezovsky
Tyler Nguyen

Department of Computer Science and Engineering
Michigan State University

Fall 2025

From Students…
…to Professionals

Project Sponsor Overview

- Online global marketplace.
- Founded by Jeff Bezos.
- Founded in 1994.
- Amazon was almost named Cadabra.

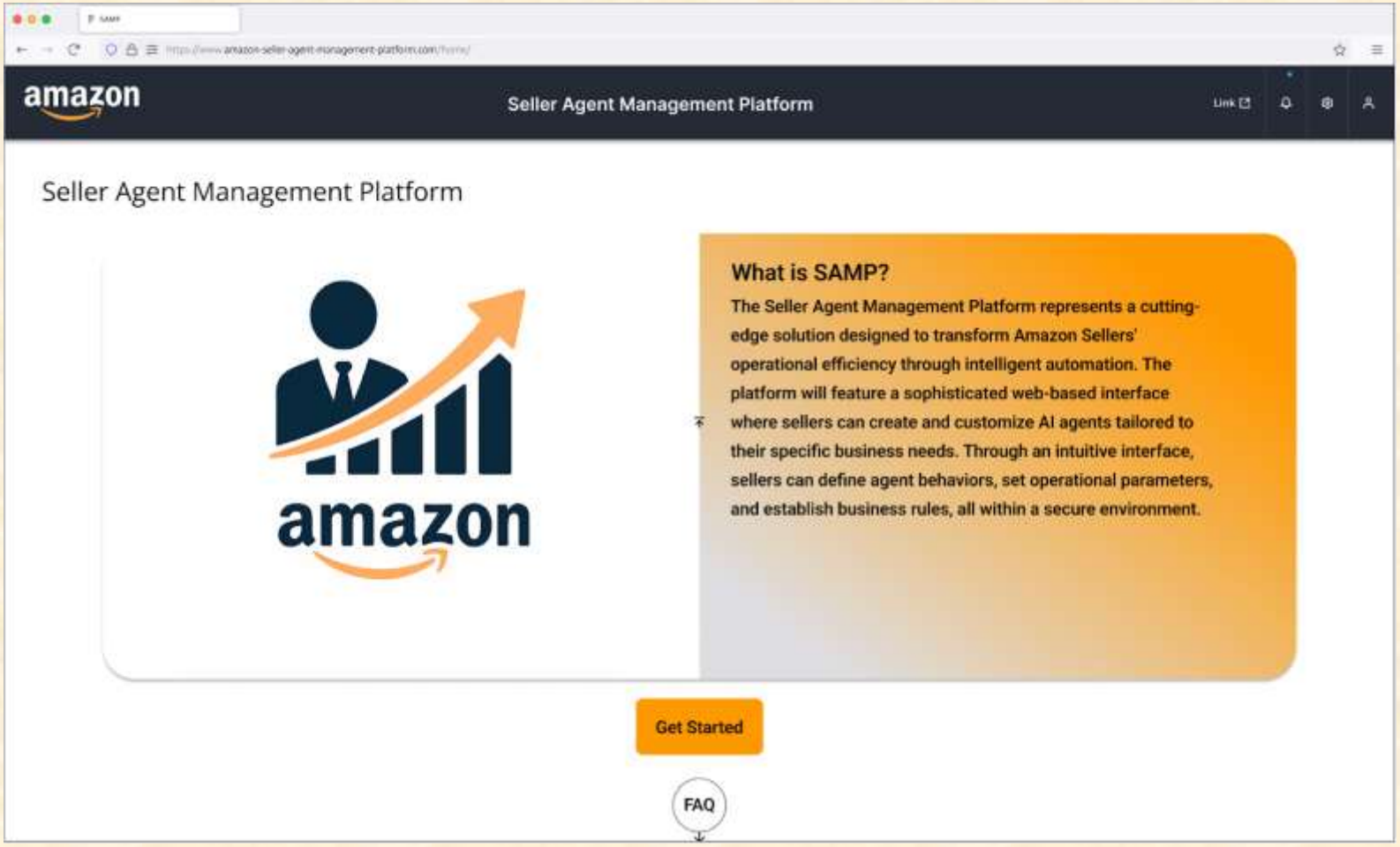
Project Functional Specifications

- Automate manual seller tasks using agents.
- Use Slack to ask for approvals and feedback.
- Implement web-based interface to configure agents.
- Be robust, scalable, and secure (e.g. queues, authentication).

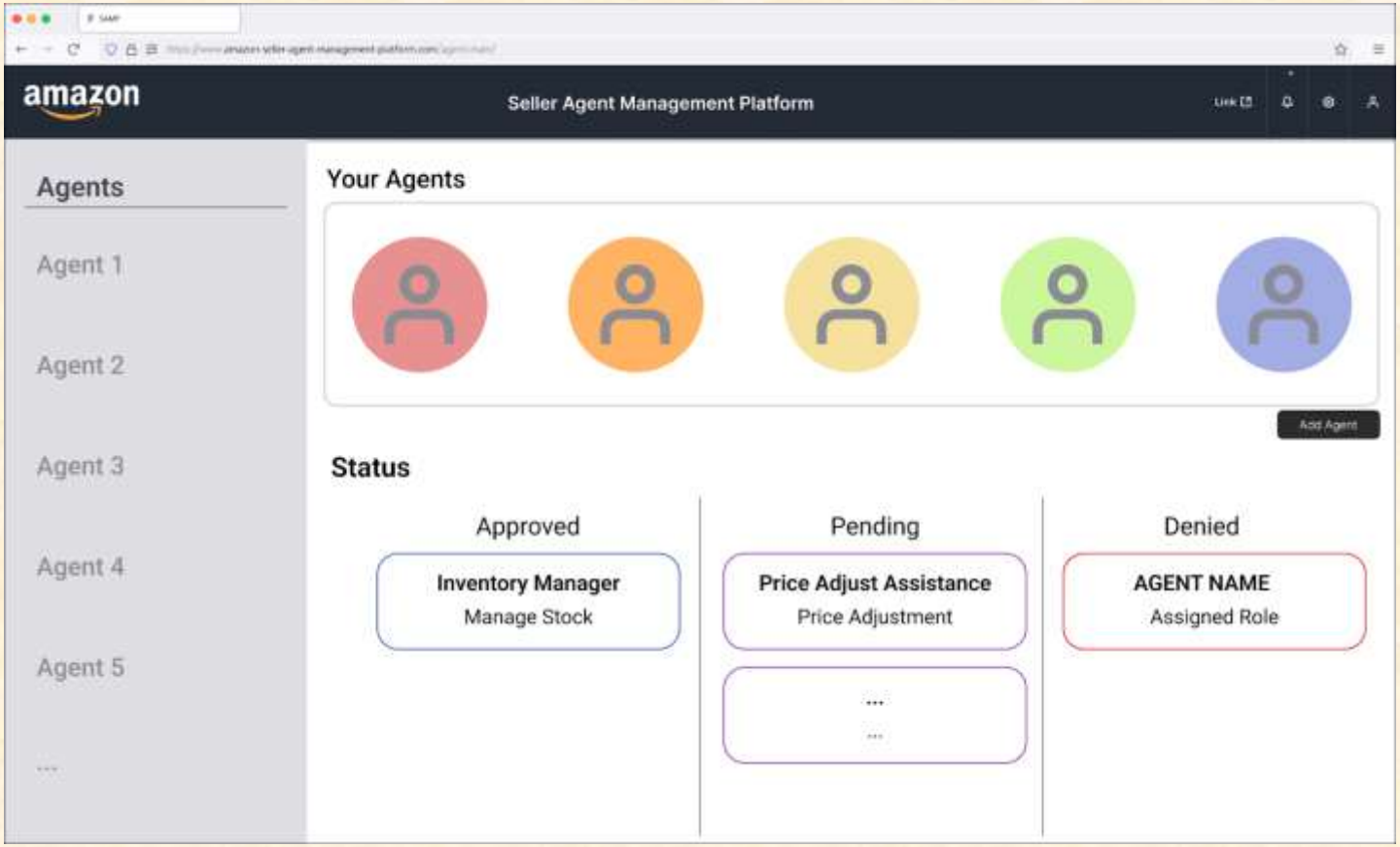
Project Design Specifications

- SAMP web application that accelerates and simplifies the Amazon Selling environment.
- Allow sellers to configure the AI Agent to operate simple yet important tasks such as responding to buyer messages and managing inventory.

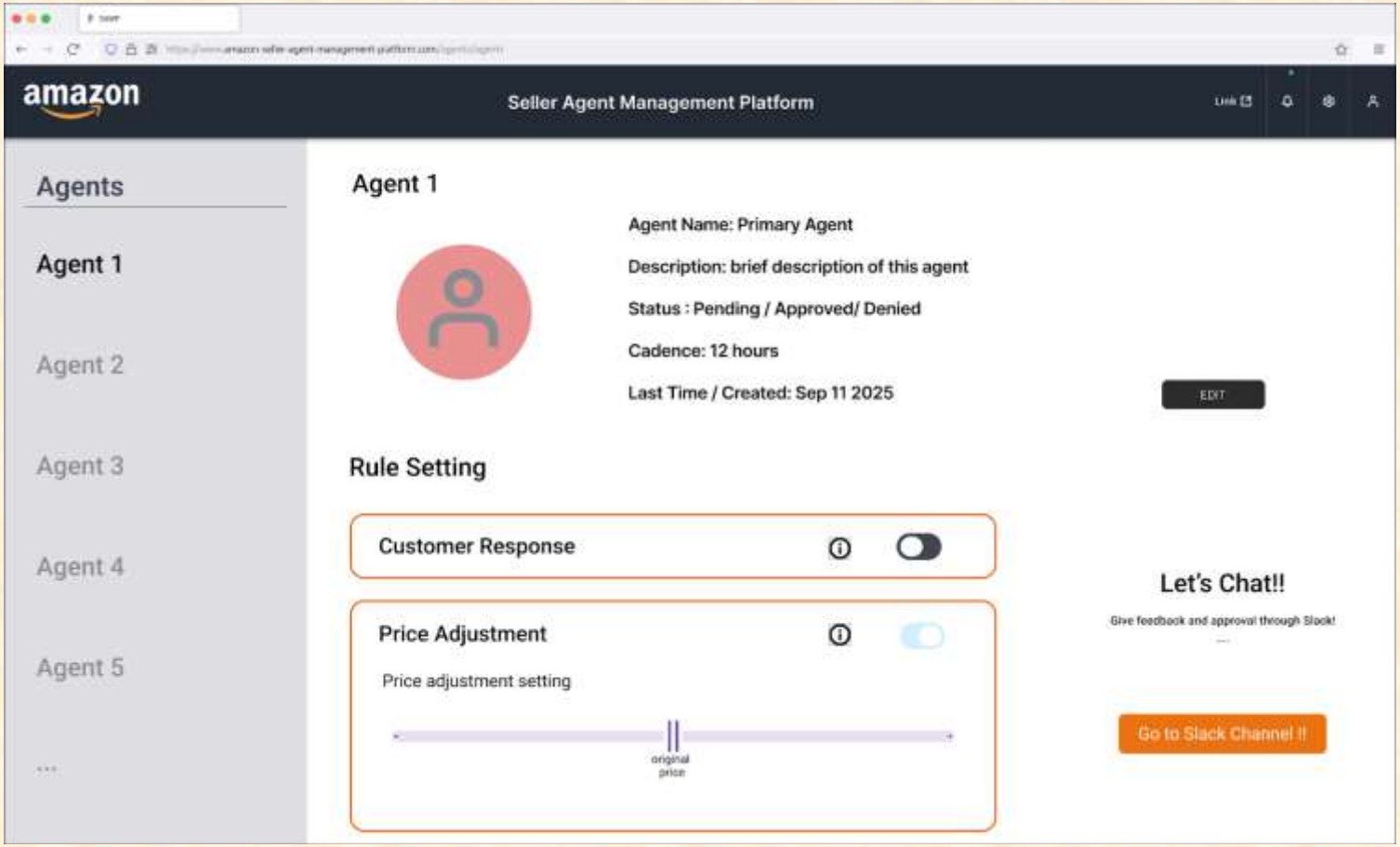
Screen Mockup: Home Page



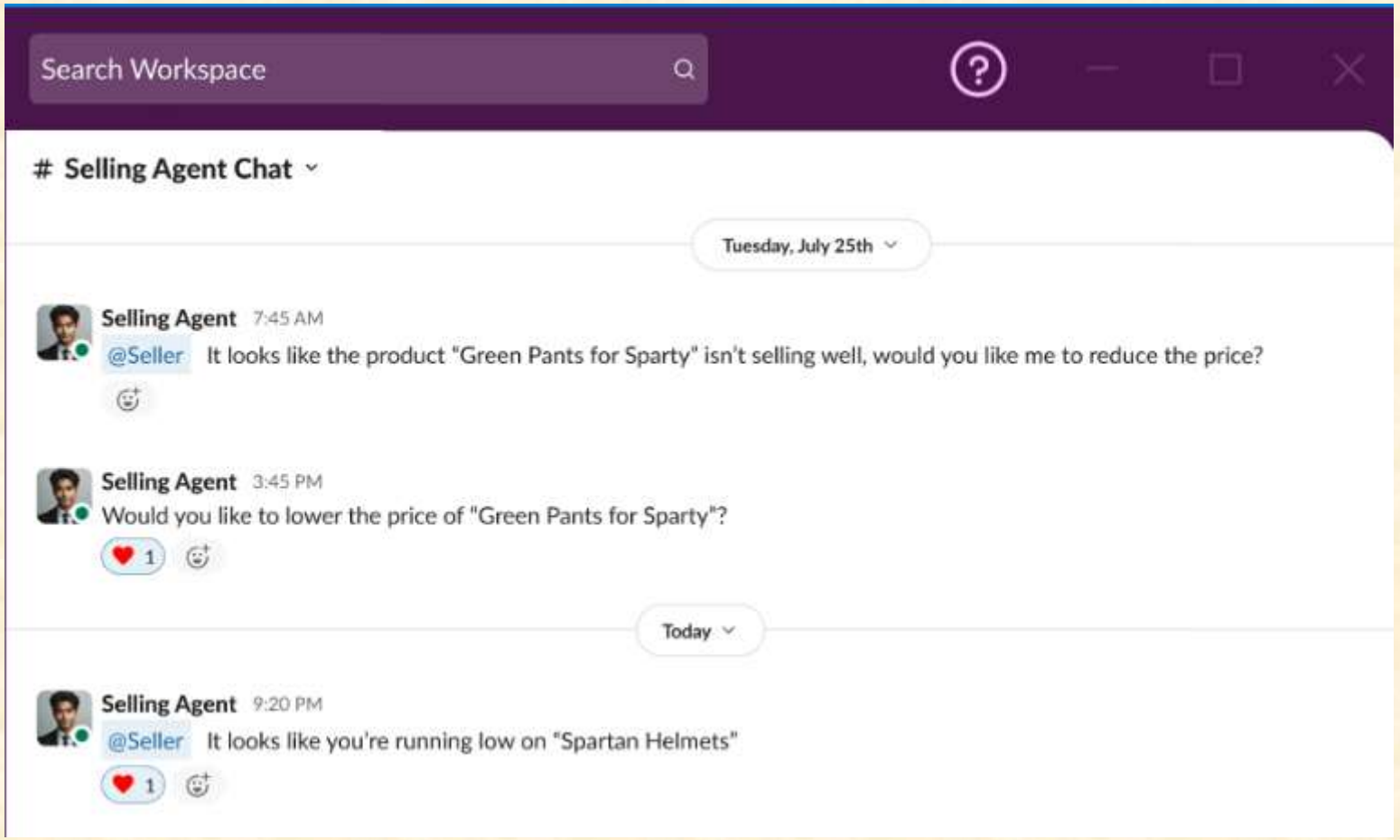
Screen Mockup: Agent Page - Main



Screen Mockup: Agent Managing Page



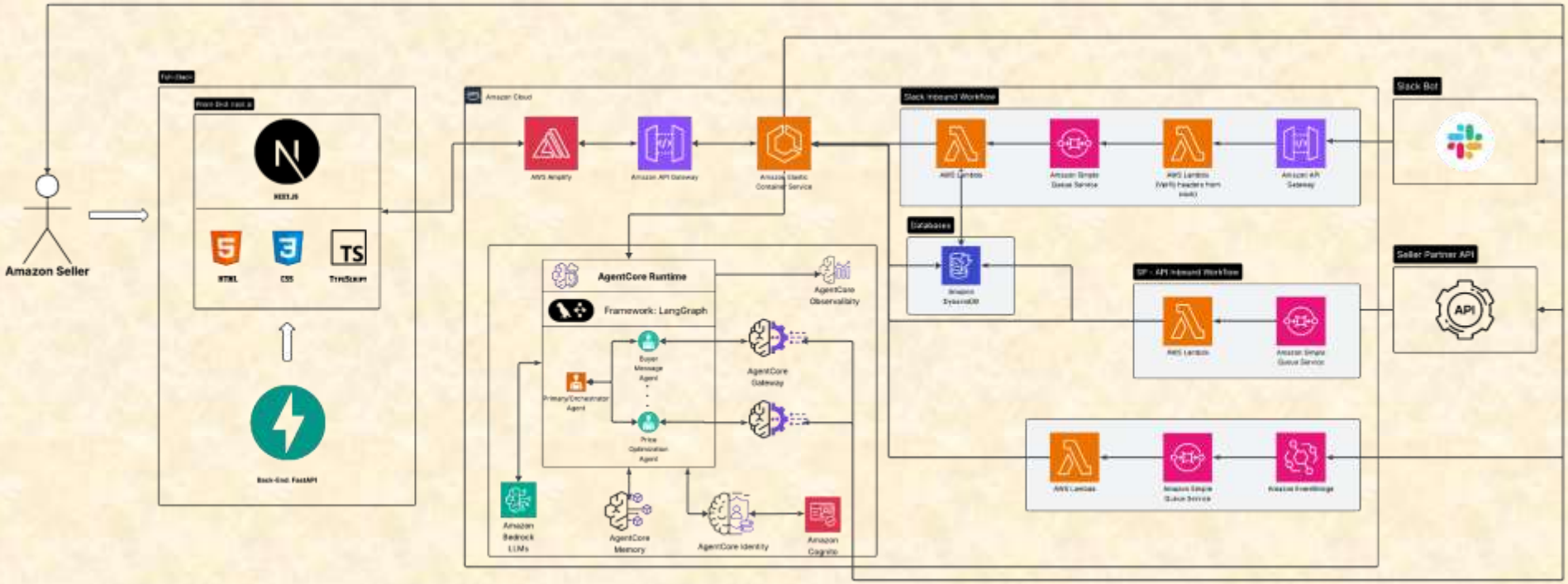
Screen Mockup: Example Slack Chat



Project Technical Specifications

- Web-based interface, written with NextJS framework and FastAPI backend.
- Authentication
  - Frontend to backend + Slack authentication through AWS Cognito with LWA (Login with Amazon).
  - SP-API Authentication through one-time auth flow and storing tokens.
- Using Slack Events API for inbound messages and Slack Web API for outbound messages.
- Agents
  - Using AWS AgentCore Runtime to host our agents.
  - Written in LangGraph with supervisor architecture.
  - Using Slack for human-in-the-loop approval.
  - SP-API functionality provided through multiple AgentCore Gateways as MCP servers.

Project System Architecture



Project System Components

- Software Platforms / Technologies
  - **Amazon API Gateway**
  - **AWS Lambda**
  - **AWS AgentCore Gateway**
  - **AWS AgentCore Memory**
  - **AWS Cognito**
  - **Login With Amazon (LWA) / OAuth2.0**
  - **Seller Partner API**
  - **Slack API**
  - **Next.js**
  - **Python**
  - **FastAPI**
  - **LangGraph**
  - **AWS Amplify**
  - **AWS Elastic Container Service (ECS)**
  - **AWS Simple Queue Service (SQS)**
  - **AWS Agentcore**
  - **Amazon DynamoDB**
  - **Amazon EventBridge**
  - **AWS CDK**
  - **AWS Bedrock**

Project Risks

- **Rate Limits on SP-API**
  - To prevent too much load on their servers, Amazon along with most services implements some form of rate limits.
  - **Mitigation**: Use a form of backoff in case we encounter something like this, for example, exponential backoff.
- **Guardrails/Human-In-The-Loop Approval**
  - Since AI is non-deterministic, giving it unlimited access to a Seller's Account is not acceptable.
  - **Mitigation**: User can choose approval settings for related groups of tools.
- **Authentication**
  - Need authentication to contact the Seller Partner API from the MCP server and from the backend, we need it to also send messages to Slack, we need it to verify the user on Slack, and we need it on the frontend to allow the user to configure business rules and agents.
  - **Mitigation**: Use AWS Cognito with LWA (Login With Amazon) for user authentication into the app/backend and the SP-API auth flow to connect to the seller's account and store the access and refresh tokens encrypted in DynamoDB.
- **Robust Design/Fault Tolerance**
  - When designing a product, it is important that it doesn't break regardless of what happens.
  - **Mitigation**: Added queues in all of our ingestion pipelines. These queues store messages until they are able to be processed by our backend, so even in high load periods, we will not drop requests. We also have DLQs (Dead Letter Queues) to store events that caused errors so they can be retried at a later point in time.

Questions?