eGain
2025 - Functional Presentation

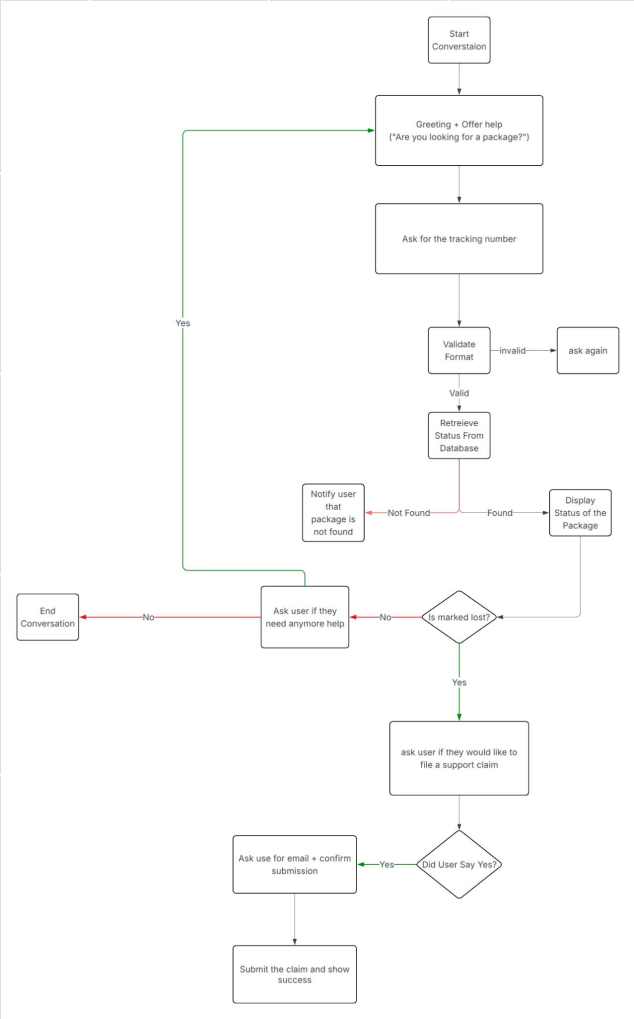
Project kickoff presentation

By Andrey Neudachin

Contents

eGain - Functional Portion

Flow Chart



System Design & Architecture

Tech Stack:

1. **Backend:** FastAPI (Python) + Uvicorn ASGI server
2. **Frontend:** Next.js 16 + React 19 + TypeScript + Tailwind CSS 4
3. **Communication:** REST API with JSON

Key Design Decisions:

- Conversational State Machine Pattern
- Session-Based Architecture
- Component-Based UI

Conversational State Machine Pattern

4 states:

awaiting_tracking →

awaiting_claim_confirmation →

awaiting_claim_email → claim_created

Maintains conversation context per session

Natural dialogue flow for package tracking → claim filing

Session-Based Architecture

Server-side session management with 1-hour TTL

In-memory storage for rapid prototyping

Session validation on all protected endpoints

Component-Based UI

Modular React components: ChatInterface (container)
→ ChatInput + MessageBubble (presentational)

Clean separation of concerns

Reusable, testable components

Technical Implementation

11 REST endpoints (/chat/start, /chat/message, /chat/track, /chat/claim, etc.)

Validation: Regex-based tracking number (^[A-Z]{2}\d{9}$) + email validation

Mock Database: 8 sample packages with statuses (in_transit, delivered, lost)

Challenges Faced

Problem 1

Needed to remember tracking number when asking follow-up questions (claim confirmation, email)

Solution

Implemented state machine with context object that persists tracking_number across conversation steps

Challenges Faced

Problem 2

Needed persistent sessions but no database requirement

Solution

Built custom CookieStore class with in-memory dict + expiration logic (1-hour TTL)

Challenges Faced

Problem 2

Parsing user responses like "yes", "yeah", "sure" vs "no", "nope"

Solution

Lowercase normalization + keyword checking (if "yes" in user_input.lower())

Company Name

Future Improvements

Database Implementation

Add PostgreSQL/MongoDB for session + claims storage

Implement Redis for session caching (faster lookups)

Database migrations support with tools like Alembic

Natural Language Processing

Integrate LLM (GPT-4/Claude) for better intent recognition

Handle typos, ambiguous queries, multi-intent messages

Handle typos, ambiguous queries, multi-intent messages

Enhanced Features

File upload for claim photos (damaged package evidence)

Email notifications when claim status changes

Multi-language support

Package delivery notifications via webhooks

Claim status tracking page (not just chatbot)

Thank you