

POLICYSCOPE

The fine print, decoded.

A browser extension that automatically analyzes Terms of Service and Privacy Policies so users know what they're agreeing to.

Understand what you're agreeing to before you click Accept.

Amaan Bukhari

Project Manager / System Architecture

Melvin Berkoh

Backend / Clause Detection

Emmanuel Garcia

UI Design

Sebastian Lazarte

Planning & Structure

Mustafa Bayram

UI/UX Flow

The Problem

Nobody reads Terms of Service.

PolicyScope was built for everyone who clicks Accept anyway.

30+

pages in the
average ToS

<10%

of users read policies
before agreeing

100%

still click
"I Agree"

Our Solution

Well, here's where PolicyScope comes in.

Our solution is a browser extension that automatically analyzes policy pages while users browse.

Instead of requiring users to manually read through large documents, PolicyScope identifies key clauses and presents them in a simplified and organized format.

This allows users to quickly identify important sections: billing policies, subscription conditions, data usage terms, and legal restrictions. All without reading a single page of fine print.

Real-Time Analysis

Scans the page automatically as you browse. No button needed.

Clause Detection

Identifies 16 clause types across billing, data, legal, and more.

Plain-Language Summaries

AI explains each clause in plain English, on demand.

Organized by Category

Results grouped into 8 color-coded categories for fast navigation.

System Architecture

01 Webpage Terms of Service

User visits a ToS or Privacy Policy in Chrome.

02 Text Extraction

parser.js pulls visible text nodes from the DOM.

03 Content Filtering

normalize.js strips noise, nav elements, and irrelevant blocks.

04 Clause Detection Engine

detector.js + rules.js pattern-match against 16 clause types.

05 Category Classification

Clauses mapped into 8 overarching categories.

06 PolicyScope UI Display

popup.js renders results; highlighter.js marks clauses on page.

AI: On Demand Only

The AI backend is NOT part of the main detection pipeline.

It is only called when a user explicitly opens a clause detail view to request a plain-language summary.

This keeps the extension fast, private, and lightweight.

Tech Stack

Extension: Chrome MV3

Detection: Vanilla JS

Backend: Node.js / Express

AI: OpenAI API (on demand)

Technical Implementation

Modular Architecture

Core components separated: parsing, detection, highlighting, popup, backend, and settings. Each module independently debuggable and updatable without touching the others.

Secure Backend

AI summarization runs through a Node.js proxy server. API keys are never exposed inside the browser extension and kept secure server-side.

Reliable Data Handling

Policy text is validated and trimmed before sending to the backend. Oversized requests are handled gracefully. Runtime and API errors never crash the extension.

Safe UI Rendering

All AI-generated summaries are rendered as plain text only. No innerHTML or dynamic HTML execution to prevent any risk of script injection via policy content.

Development Process • SCRUM

Stage 1 Core Workflow

Text extraction and clause detection pipeline.

Stage 2 User Interface

Popup interface and on-page policy highlighting.

Stage 3 Category Expansion

Expanded to 16 clause types across 8 categories.

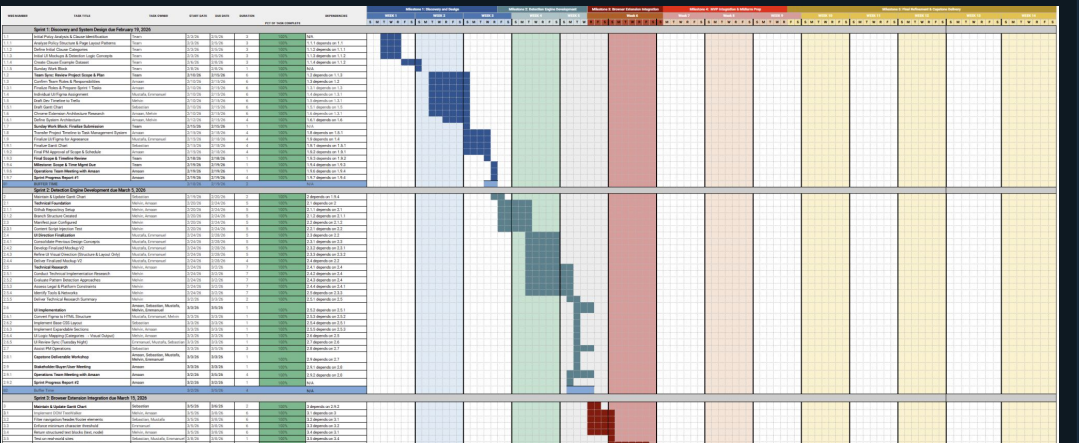
Stage 4 Customization

Settings, highlight styles, day/night theme, UI refinements.

ClickUp Project Management

Task	Assignee	Due Date	Priority	Progress
Initial Policy Analysis & Clause Identification	John Doe	2/15/25	High	1.1
Analyze Policy Structure & Page Layout	John Doe	2/15/25	High	1.1.1
Define Initial Clause Categories	John Doe	2/15/25	High	1.1.2
Initial UI Mockups & Detection Logic C...	John Doe	2/15/25	High	1.1.3
Create Clause Example Dataset	John Doe	2/15/25	High	1.1.4
Setup Initial Backend	John Doe	2/15/25	High	1.1.5
Name Space Review Project Scope & Plan	John Doe	2/15/25	High	1.2
Confirm Team Roles & Responsibilities	John Doe	2/15/25	High	1.3
Finalize Rules & Prepare Sprint 1 Tasks	John Doe	2/15/25	High	1.3.1
Individual UI/UX Design	John Doe	2/15/25	High	1.4
Design Dev Timeline to Trello	John Doe	2/15/25	High	1.5
Design Gantt Chart	John Doe	2/15/25	High	1.5.1
Change Detection Architecture Research	John Doe	2/15/25	High	1.6
Define System Architecture	John Doe	2/15/25	High	1.6.1
Setup Initial Backend	John Doe	2/15/25	High	1.7
Transfer Project Timeline to Task Manager	John Doe	2/15/25	High	1.8
Finalize Gantt Chart	John Doe	2/15/25	High	1.9
Final Gantt Chart	John Doe	2/15/25	High	1.9.1
Final PM Approval of Scope & Schedule	John Doe	2/15/25	High	1.9.2
Final Scope & Timeline Review	John Doe	2/15/25	High	1.9.3
Implement Scope & Timeline Review	John Doe	2/15/25	High	1.9.4
Operations Team Meeting with Review	John Doe	2/15/25	High	1.9.5
Update Progress Report #1	John Doe	2/15/25	High	1.9.7
Monitor & Update Gantt Chart	John Doe	2/15/25	High	2
Technical Foundation	John Doe	2/15/25	High	2.1
GitHub Repository Setup	John Doe	2/15/25	High	2.1.1
Branch Structure Created	John Doe	2/15/25	High	2.1.2
Backend API Configured	John Doe	2/15/25	High	2.2
Backend API Integration Test	John Doe	2/15/25	High	2.2.1

Gantt Chart



Challenges & Solutions

Challenge

Clause Detection Accuracy

Solution

Iterated on regex patterns and keyword rules using real policy pages as test cases. Refined per-category detection logic until false positives dropped significantly.

Challenge

API Key Security

Solution

Moved all AI summarization to a Node.js backend proxy. The API key never touches the browser. This is a hard requirement for any production Chrome extension.

Challenge

Dynamic Page Content

Solution

Many policy pages load text after the initial DOM render. Built the content script to observe DOM changes and re-run extraction on deferred content.

Challenge

False Positives

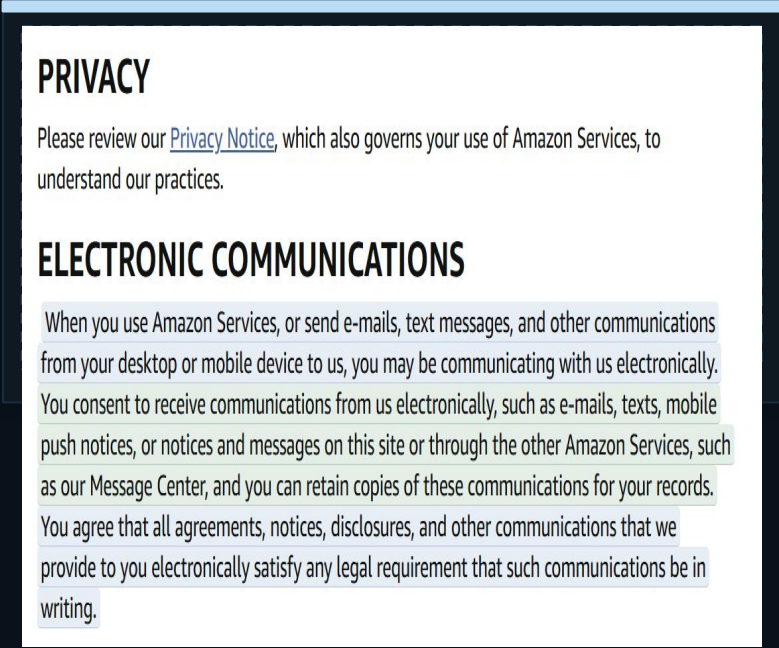
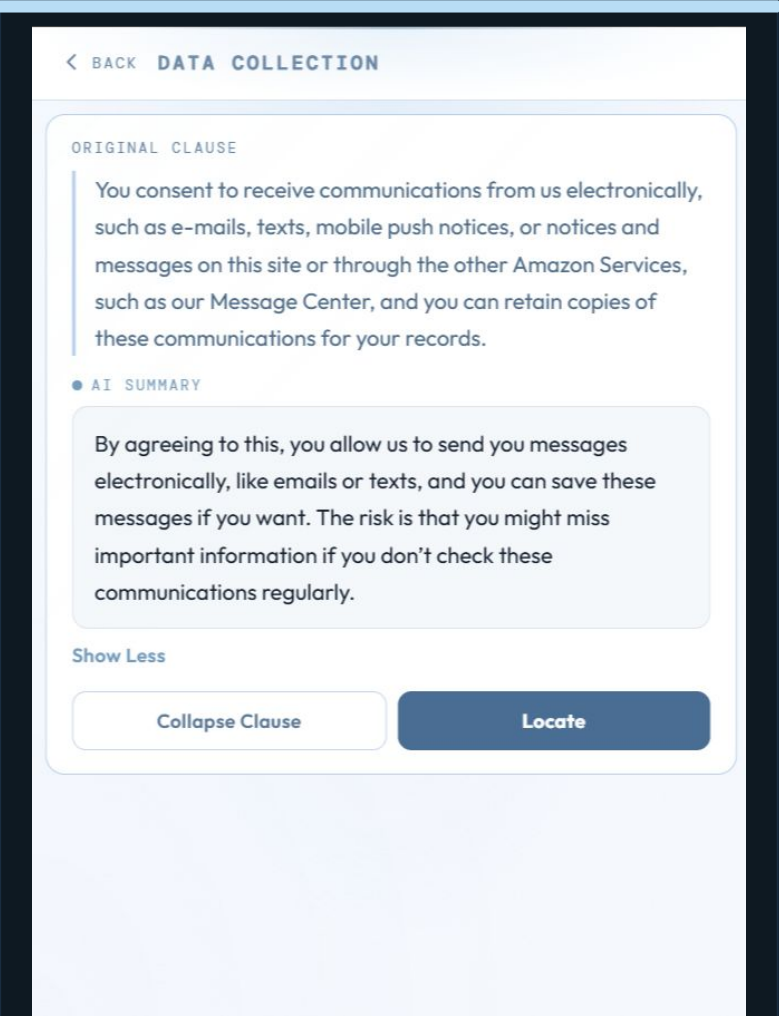
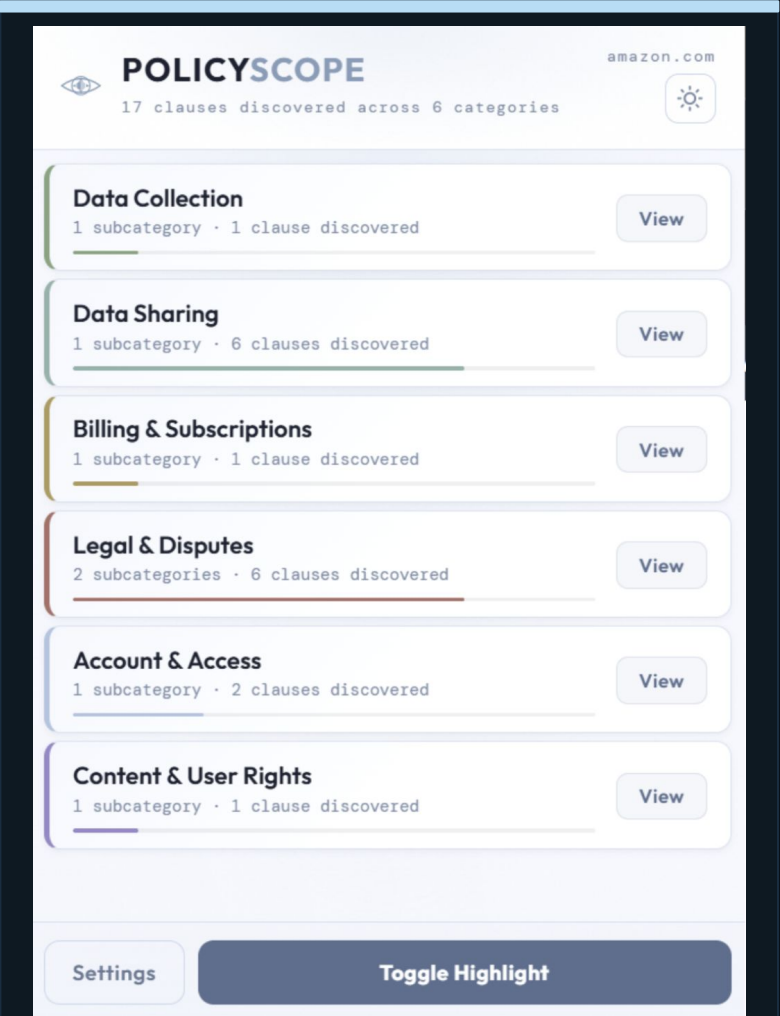
Solution

Early detection matched too aggressively. Tightened matching criteria, added minimum length thresholds, and tested against 10+ major policy pages iteratively.

Design & Usability

Built to feel like a product, not a school project.

Dark / day mode · 5 highlight styles · 8 color-coded categories · slide transitions between views



Testing & Results

4.5/5

Ease of Understanding ToS

4.6/5

Helpfulness for Agreements

96%

Would Recommend or Use

24 respondents · April 2026



What We Learned & What's Next

What We Learned

- **Users don't read policies.**

They need a tool that does it for them and surfaces only what matters.

- **Plain language > completeness.**

Users care more about understanding than seeing every clause.

- **Security is non-negotiable.**

Exposing API keys in a browser extension is a real vulnerability, not a shortcut.

- **UI clarity builds trust.**

A security tool has to look and feel trustworthy and design directly affects adoption.

- **Real users change every decision.**

Designing for actual testers instead of rubric criteria made the product significantly better.

What's Next

Policy Change Alerts

Notify users when a site's ToS updates since their last visit.

Expanded Detection

More clause types across more policy formats and document structures.

Cross-Browser Support

Extend beyond Chrome to Firefox and Edge.

Community Voting

Let users flag and vote on clause importance across the community.

Mobile Support

Explore a mobile companion app for reviewing policies on the go.

Live Demo

PolicyScope analyzing a live Terms of Service page in real time.

POLICYSCOPE

Thank You

See you soon!

Amaan Bukhari · ab2928@njit.edu

Melvin Berkoh · mb2272@njit.edu

Emmanuel Garcia · esg7@njit.edu

Sebastian Lazarte · sl894@njit.edu

Mustafa Bayram · mb2326@njit.edu