# SeerSync : Browser Native Intelligent Communication

Google Chrome Built-in AI Challenge 2025

Vidhi Kothari
vidhikothari1@gmail.com
Pace University
New York, USA

Rajpreet Singh
rajpreet.singh@tum.de
TU Munich
Bavaria, Germany


**Figure 1: Browser Built-it Communication Extension**

## ABSTRACT

Users switch between messaging platforms while researching or working, losing context and productivity. Real-time collaboration requires constant context-switching between the browser and external tools. **SeerSync** is a Chrome extension that brings seamless, intelligent messaging directly into the browser workflow. It enables users to communicate with contacts without leaving their current context, while leveraging AI to enhance message quality in real-time using **Gemini**. Built with React and a background service worker, it integrates messaging and processing APIs seamlessly, offering context-menu quick-send, persistent history with URL tracking, OAuth authentication, and a frictionless *Select -> Right-click -> Send* workflow.

In: Google Chrome Built-in AI Challenge (Submitted by: Vidhi and Rajpreet) DevPost Hackathon.


SeerSync eliminates productivity loss from context-switching, reduces communication friction, and brings enterprise-grade message enhancement to casual browser-based communication—all without leaving your workflow.

## KEYWORDS

datasets, neural networks, gaze detection, text tagging

## 1 INTRODUCTION

Modern web-based workflows fragment user attention across multiple applications. While users research, write, and collaborate online, communication happens in separate platforms like email, messaging apps, chat tools—forcing constant context-switching that disrupts productivity and loses critical information. The browser itself becomes merely a content consumption tool rather than a communication hub. Additionally, asynchronous communication often suffers from quality issues. Messages lack polish, context, or clarity where users struggle to convey tone across different audiences and language barriers complicate collaboration. Existing

solutions either lock users into dedicated platforms or require deliberate context export both friction points that delay or prevent communication. We present **SeerSync**, a Chrome extension that integrates intelligent, real-time messaging into the browser's native interface. By enabling in-context communication with AI-powered message enhancement, **SeerSync** removes friction from two critical workflows which includes sharing information and composing messages. Users can select text, right-click, and send to contacts with optional AI processing to proofread, summarize, translate, or refine tone without leaving their browser. The system automatically captures URLs and maintains persistent message history, turning the browser into a collaborative communication platform. This approach reduces cognitive load, preserves context, and leverages AI to improve message quality at the point of composition. We demonstrate the feasibility of this integration through a working prototype and discuss its implications for browser based productivity tools.

# 2 ARCHITECTURE

## 2.1 System Overview

**SeerSync** is composed of three interconnected layers: a Chrome extension frontend built with React, a background service worker for asynchronous event handling, and external APIs for message persistence and AI processing. The system operates within browser security constraints, communicating with backend services through standard HTTP protocols.

## 2.2 Frontend Layer

The user interface renders as a popup extension with three distinct views managed through React state. The authentication view presents a Google OAuth login interface, the contacts view displays saved contacts with action buttons for chat, add, and invite operations, and the chat view shows message history with a real-time message input panel. Message enhancement is accessed through a process menu triggered by an emoji button, offering five transformation options: proofread for grammar correction, summarize for content condensation, translate for language conversion, rewrite for tone refinement, and generate for context-aware composition. Users select a processing type, receive a preview of the transformed message, and confirm before sending. The interface uses Tailwind CSS for styling with careful attention to scroll persistence, ensuring that chat views maintain user scroll position across new message arrivals rather than jumping to the top on every update.

## 2.3 Background Service Worker

The background worker operates independently of the popup lifecycle, handling persistent browser events and asynchronous operations. Upon extension installation, the worker generates a context menu structure dynamically populated with user contacts. When a user right-clicks on selected text, the worker intercepts the event, captures the selection along with the current page URL and title, and routes this data appropriately. The worker maintains an in-memory cache of contacts and user identifiers to enable instant menu generation without incurring backend latency on each right-click. For message delivery through context menus, the worker applies selected AI processing before submission, eliminating the need for user confirmation in the quick-send workflow. This separation of concerns allows messaging operations to complete even when the popup window is closed.

## 2.4 API Integration

Message processing leverages the Gemini 2.5 Flash API, which accepts original text, processing type, and optional page context (the first 500 characters of page content) to generate refined messages. Processing prompts are dynamically constructed to ensure output clarity and brevity, with specific instructions for each transformation type. The custom backend manages user authentication, contact relationships, and message persistence through a RESTful API. Authentication maps Google OAuth identifiers to internal user IDs, enabling seamless user identification across sessions. The messages endpoint stores and retrieves conversation history between user pairs, while the contacts endpoint manages relationship operations including add, delete, and invitation workflows. All API communication occurs over HTTPS with standard JSON serialization.

## 2.5 State Management and Data Flow

React hooks manage all local UI state, including message lists, contact lists, user input fields, and UI visibility toggles. Message history is fetched on-demand when a user opens a chat and cached in component state throughout the session. The system implements one-time scroll-to-bottom behavior using a ref-based flag that triggers only when opening a new chat, preventing unwanted scroll jumps when new messages arrive. When a user sends a message, the system transmits it to the backend, refreshes the local message history, and optionally generates a mock AI response for demonstration purposes. The context menu workflow follows a parallel path: capture text, apply optional processing via Gemini, submit to backend, and update the sender's message view. Both workflows maintain eventual consistency, allowing users to see sent messages reflected in their chat history after backend confirmation.

## 2.6 Security and Privacy

User authentication relies on Google OAuth, eliminating the need for password management. API requests include user identifiers only after successful authentication. Message content is transmitted over HTTPS, and the extension never stores credentials or sensitive data in local storage. The background worker operates with minimal permissions, accessing only the active tab's URL and selected text when explicitly triggered by user action.

# 3 USE CASES AND FEATURES

## 3.1 Consumer Use Cases

**SeerSync** enables seamless communication for everyday web workflows. Students researching assignments can instantly share relevant passages with study groups without context loss, with optional summarization for quick peer review. Remote workers coordinating across time zones can send page excerpts to colleagues with professional tone refinement, maintaining async communication quality. Content creators collaborating on research can highlight key findings, auto-capture source URLs, and share refined notes

with co-authors directly from the browser. Social media managers monitoring conversations can quickly respond to audience inquiries with grammar-checked messages, reducing response time from minutes to seconds. Language learners studying foreign content can translate selected passages to peers while maintaining conversation flow, reducing friction in multilingual collaboration.

### 3.2 Enterprise Use Cases

In enterprise settings, **SeerSync** transforms knowledge sharing and compliance workflows. Legal teams reviewing contracts can highlight critical clauses, share them with colleagues with auto-captured document URLs, and ensure consistent communication tone across sensitive discussions. Research organizations conducting competitive intelligence can rapidly disseminate market insights with automatic summarization, reducing information mismatch for stakeholders. Customer support teams handling tickets can draft professional responses with grammar enhancement and tone adjustment, ensuring brand consistency while reducing ticket resolution time. Compliance officers monitoring communications can ensure all outbound messages meet regulatory tone and clarity standards through mandatory AI review before sending. Cross-functional product teams can accelerate feedback cycles by sharing design mockups, research findings, or technical documentation with instant, in-context messaging rather than context-switching to email.

### 3.3 Core Features

The extension provides five AI-powered message transformations: proofread corrects grammar and spelling in real-time, summarize condenses lengthy passages to essential points, translate converts content across languages, rewrite adapts tone for professional or casual contexts, and generate creates new content from brief prompts. Right-click context menu integration enables one-click sharing from any webpage with automatic URL capture for reference. Persistent message history with clickable links maintains conversation context and enables easy reference to shared resources. Contact management supports add, invite, and delete operations, with Google OAuth for secure authentication. Mock AI responder generates contextual replies for demonstration, enabling single-user testing and feature showcase without requiring multiple accounts.

## 4 ENGINEERING CHALLENGES AND DESIGN TRADE-OFFS

Managing asynchronous message flow across browser contexts required careful state synchronization between the independent background worker and popup. Scroll position management presented a critical UX challenge—automatically scrolling on new messages caused jarring behavior during user interaction, so we implemented a one-time scroll flag triggered only on chat open. API efficiency demanded trade-offs: calling Gemini for every preview preview incurred latency, while caching created inconsistency. We adopted a preview-on-demand model where processing occurs only when explicitly selected, reducing API calls without sacrificing functionality. Context menu generation required cache invalidation whenever contacts update to prevent stale items. We prioritized friction reduction over feature completeness, deferring offline messaging, encryption, and rich media to focus on core select-and-send workflow and message enhancement. The five AI options balance utility against cognitive load; we excluded sentiment analysis and tone detection to maintain simplicity. We chose polling-based refresh over WebSockets to simplify backend requirements, accepting slight latency as acceptable for a productivity tool. The mock responder trades realism for demo capability, and in-memory state storage—though ephemeral per session—complies with browser sandbox restrictions while the backend maintains permanent records.

## 5 LEARNING AND INSIGHTS

Several unexpected insights emerged during development. First, scroll management revealed that users prioritize predictability over automation—forcing scroll-to-bottom on every message update created frustration despite seeming helpful. This taught us that UX intuitions often fail; small behavioral choices deeply impact user experience. Second, the right-click context menu proved more powerful than anticipated for adoption—it eliminated the mental friction of "open extension, find contact, type message." This single interaction pattern became the extension's defining feature, suggesting that reducing interaction steps matters more than feature richness. Third, integrating Gemini API revealed that AI quality varies dramatically by use case: summarization excels on technical content but struggles with ambiguous text, while proofreading is consistently reliable. This taught us that AI is not a universal solution—careful prompt engineering and use-case matching are essential. Fourth, we discovered that mock responders, while artificial, unlock critical functionality for solo testing and demonstration. This highlighted the value of accepting "good enough" solutions for non-core features. Finally, managing state across browser contexts exposed the complexity hidden beneath Chrome's apparent simplicity—the extension model, while powerful, requires deliberate architectural thinking about where logic lives. This reinforced that architectural decisions early in development have an enormous impact on maintainability and feature velocity.

## 6 CONCLUSION AND FUTURE WORK

**SeerSync** demonstrates that in-browser messaging with AI enhancement reduces communication friction in web-based workflows. By embedding select-and-send capability directly in the browser and offering intelligent message refinement at point-of-composition, we eliminate context-switching while improving message quality. The architecture balances simplicity with functionality, proving that a focused MVP can deliver meaningful productivity gains. Future work should explore WebSocket-based real-time messaging to eliminate polling latency, implement end-to-end encryption for sensitive communications, and extend AI processing to include custom tone profiles and sentiment analysis. Integration with browser sync could enable persistent message history across devices, while support for rich media (images, documents) would broaden use cases. Expanding Gemini's capabilities to context-aware suggestions based on conversation history could further reduce user effort. Finally, analytics on message processing patterns would reveal which transformations provide highest value, guiding feature prioritization for production deployment.