UttarAI – Chrome Extension Documentation

---

1. Introduction

UttarAI is a Chrome extension that generates smart, AI-based replies for **Gmail** and **WhatsApp Web** conversations. It captures the context of conversations, sends it to a **Spring Boot-powered AI backend**, and automatically writes an appropriate response in the input box. This saves time and enhances communication efficiency.

---

2. Core Functionality

a. Manifest Configuration (manifest.json)

This file contains essential metadata and configurations for the extension:

- **Permissions**: Includes storage, activeTab, tabs, etc.
- **host_permissions**: Grants access to specific URLs.
- **content_scripts**: Injects JS scripts into specified web pages (e.g., WhatsApp, Gmail).
- **web_accessible_resources**: Allows resources to be injected securely into matching web pages.

---

b. WhatsApp Integration (content.js)

DOM Observer

- A MutationObserver watches the DOM for changes.
- Detects when a new chat is opened by checking if the **input field is rendered**.
- Uses Chrome DevTools to identify class names for targeting HTML elements.

Injecting the AI-Reply Button

- Checks for existing buttons and removes duplicates.
- Locates the **toolbar (input area)** using class selectors and stores it in a variable.

- Calls createButton() to insert a custom AI reply button matching WhatsApp's theme.

Generating Conversation Context

- Captures the **last 10 messages** using class selectors and querySelectorAll().
- Annotates messages using me: or them: to indicate the sender.
- On button click, sends the context in an API call to the Spring Boot backend.
- Injects the AI-generated reply into the input field for user review/editing before sending.

---

c. Gmail Integration

- Similar working as WhatsApp, but only captures the current email content.
- Context sent to backend for reply generation.
- Injects generated reply into Gmail's response area.

---

3. Razorpay Payment Gateway Integration

a. Setup

- Create a Razorpay account and obtain **API key** and **secret key**.

b. Frontend

- View Pricing page lists available subscription plans.
- On plan selection, an API call is made to the backend with the selected plan.

c. Backend Workflow

- Fetches plan details from DB.
- Uses RazorpayClient (added via Maven in pom.xml) to:
  - Create an order with amount, receipt ID, etc.
  - Saves and returns orderId, user info, and payment details to the frontend.

d. Frontend Payment Flow

- Razorpay script is included in index.html.
- A Razorpay.Options object is configured and passed to start the payment.
- Razorpay pop-up handles the payment.

e. Handling Payment Responses

- Razorpay provides success and failure **callbacks**:
    - On **failure**, display an error message.
    - On **success**, Razorpay returns paymentId, signature, and orderId.

f. Verifying Payment

- Frontend sends paymentId, signature, and orderId to the backend.
- Backend uses Razorpay's Utils to verify the payment signature.
- On successful verification:
    - DB is updated with payment details.
    - User's API call quota is increased based on the chosen plan.
- On failure:
    - Payment status is marked accordingly.

---

4. System Workflow Overview

a. User Registration

- Users register via the website.
- Backend stores user info with:
    - Plan: **Free Tier**
    - API Calls: **25** (free quota)

b. Extension Installation & Usage

- User installs and enables the Chrome extension.

- Clicks on the extension icon → Pop-up with login appears.
- On login:
    - A new tab with the login form opens.
    - After successful login, the tab closes and WhatsApp Web opens automatically.

c. Chat Interaction

- When WhatsApp is open:
    - **AI-Reply** button appears next to the input box.
    - On clicking the button:
        - Context is collected.
        - API call is made.
        - AI-generated response is injected into the input field.

d. Token Handling

- JWT token is used for session management.
- If the token is expired:
    - Alert: *"Token expired. Please log in again."*
    - User must re-authenticate to continue.

e. API Call Limit Handling

- Free tier offers **25 API calls**.
- If limit is reached:
    - Alert: *"API call limit reached. Please upgrade your plan."*

f. Upgrading Plan

- Visit the website → Navigate to **View Pricing**.
- Plans are listed for selection.
- Payments handled via **Razorpay** (test mode).

- On successful payment:
    - API quota is updated.
    - Plan is changed in the user's profile.

g. Plan Info

- View Pricing page also displays:
    - **Current plan**
    - **Remaining API calls**

{ Note: ***All payments are currently in test mode. No real money is involved. You can complete the payment process using dummy/test credentials, and your API call quota will be updated according to the selected plan.*}**

Tech Stack

Component	**Technology**
Frontend	Chrome Extension (JavaScript, HTML, CSS)
Backend	Spring Boot (Java, Maven)
Payments	Razorpay API
Authentication	JWT Token-Based
Database	MySQL /