**Title of the Experiment:**  
**Simulation of AI Website Detection and Blocking using Chrome Extension and Gemini API**

**Submitted by:**  
**Name:** Archit Deshpande, Ayush Dutta, Ujwal Nandanwar, Sahil Deshmukh  
**Roll No:** A1-30, A1-26, A1-5, A1-66.  
**Branch / Semester:** CSE III Semester

**1. Aim / Objective**  
To simulate and implement a Chrome extension that detects and blocks AI-related websites using the Google Gemini API.

**2. Theory / Conceptual Background**  
This experiment demonstrates how browser extensions can integrate with AI models to automate content filtering. The extension uses the Gemini API to analyse website content and determine if it is AI-related. If identified, the site is blocked and redirected to a custom page.

**Key Concepts:**

* Chrome Extension Architecture: Includes manifest files, background scripts, and popup interfaces.
* Gemini API: A generative AI model used to classify website content.
* Content Filtering: Blocking or redirecting access based on classification.
* Local Storage: Used to persist blocked site data.

**Workflow Diagram:**  
User visits site → background.js triggers Gemini API → Response: AI-related? → If yes → redirect to blocked.html

**3. Software / Simulation Tool Used**

* Chrome Browser v118 or above
* Google Gemini API
* Visual Studio Code

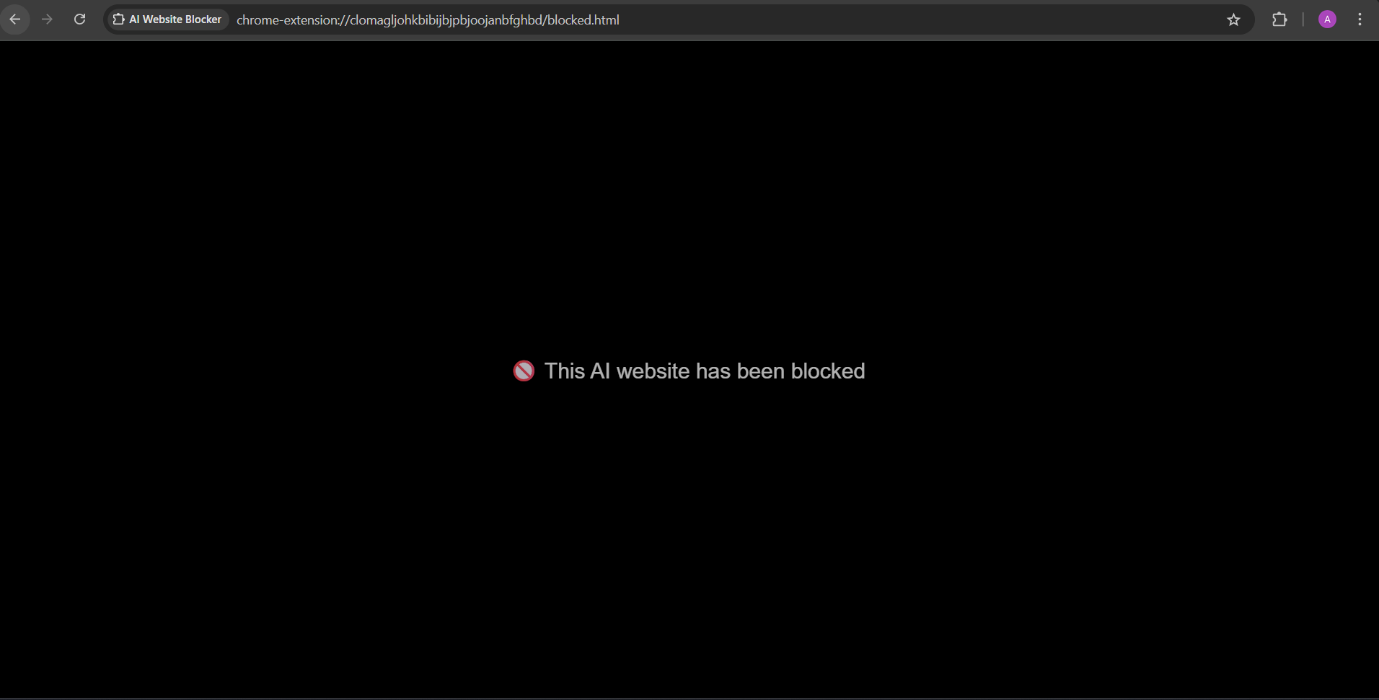
**4. Hardware / Software Requirements**

**Software:**

* Operating System: Windows 10
* Chrome Browser
* Gemini API access
* Code Editor (VS Code)

**5. Procedure / Steps Performed**

* Clone the GitHub repository: <https://github.com/DuttaAyush/ai_blocker>
* Generate a Gemini API key from Google AI Studio
* Replace YOUR\_API\_KEY\_HERE in background.js with your actual key
* Open chrome://extensions/ in Chrome
* Enable Developer Mode and click Load unpacked
* Select the project folder
* Visit websites to test auto-blocking
* Use the popup to manually add/remove blocked sites

**6. Simulation Diagram / Screenshot**  


**7. Observations**

* AI-related websites were successfully detected and blocked
* Manual site blocking worked via popup interface
* Blocked sites redirected to a black screen
* Local storage retained blocked site list across sessions

**8. Result / Output**  
Successfully implemented a Chrome extension that uses Gemini API to detect and block AI-related websites, with manual override and persistent storage.

**9. Conclusion**  
This experiment demonstrated how browser extensions can integrate with AI models to automate content filtering. It highlighted the use of Gemini API for real-time classification and Chrome’s extension APIs for user interaction and redirection.

**10. Viva Questions**

1. What is the role of the Gemini API in this extension?

Ans: The Gemini API is used to analyse the content of visited websites and determine whether they are AI-related. When a user navigates to a site, the extension sends the URL or page content to the Gemini API, which returns a classification. If the response indicates the site is AI-related, the extension redirects the user to a custom page.

1. How does Chrome’s background script interact with web pages?

Ans: When a user visits a new site, the background script intercepts the URL, sends it to the Gemini API for analysis, and based on the result, either allows access or redirects the tab to a blocked page.

1. How is local storage used in this project?

Ans: When a user manually adds or removes a site from the block list via the popup interface, the changes are saved using Chrome’s “chrome.storage.local” API.

1. What are the limitations of using AI for website classification?

Ans: **Accuracy**: AI models may misclassify websites due to ambiguous or limited content.

**Latency**: Sending requests to external APIs like Gemini can introduce delays.

**Privacy**: Sending URLs or content to an external API may raise privacy concerns.

**Adaptability**: AI models may not keep up with rapidly changing web content or new AI-related domains.

**Cost**: Frequent API calls may incur usage limits or costs depending on the provider.