A

Project Report

on

Personal Firewall

Developed By

Nagati Upender [2405102140020]

Nandana Unni [2405102140021]

P Nikhila Sree [2405102140030]

as

Partial Fulfillment of Semester III of Master of Computer Applications / Master of Science in information Technology

for A.Y. 2025 - 2026

Under the Guidance of

Prof. Mini Bhola

Submitted To

Department of MCA/MS.c.IT

Faculty of IT & Computer Science

PARUL University





CERTIFICATE

This is to certify that Mr./Ms. Student Name, Enrollment No. \_\_\_\_\_\_\_\_\_\_\_\_\_\_ and Mr./Ms. Student Name, Enrollment No. \_\_\_\_\_\_\_\_\_\_\_\_\_\_ students of Master of Computer Applications and/or Master of Science in Information Technology has satisfactorily completed the Minor Project on “Project Title” at Faculty of IT & Computer Science, Parul University as partial fulfillment of MCA and/or M.Sc. (IT) Semester III.

**Seat No. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date of Submission: \_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Internal Guide Head of Department Dean - FITCS**

Faculty of IT & Computer Science

PARUL University

Vadodara

|  |  |  |
| --- | --- | --- |
| **no** | **index** | **p.no** |
| **1** | **Introduction** |  |
| **2** | **Project Profile**  2.1 Project Definition  2.2 Project Description  2.3 Problem Statement  2.4 Proposed System & Features   * 1. Tools & Technology Used   2. Outcomes |  |
| **3** | **Requirement Analysis**  3.1 Feasibility Study  3.2 Users of the System  3.3 Modules  3.4 Process Model   * 1. Hardware & Software Requirements   2. Use Cases |  |
| **4** | **System Design**  4.1 Use Case Scenarios  4.2 Diagrams  4.2.1 UML / DFD   * + 1. Entity Relationship Diagram   4.3 Data Dictionary |  |
| **5** | **Implementation**  5.1 Frontend (Chrome Extension)  5.2 Backend (Fast API & Database)   * 1. Database Schema   2. Integration |  |
| **6** | **Features of Personal Firewall Pro**  6.1 Keyword Filtering  6.2 HTTPS Enforcement  6.3 Trusted Domains   * 1. Logging System   2. Password Protection for Firewall Toggle |  |
| **7** | **Testing, Results & Demonstration**  7.1 Test Cases and Results  7.2 How the Project Works (Execution Flow)  7.3 System Demonstration (Screenshots)  7.4 Advantages and Performance |  |
|  | **conclusion** |  |

**1.Introduction:**

The rapid growth of the internet has made access to information easier than ever, but at the same time it has also increased exposure to harmful, inappropriate, and unsafe online content. Many websites contain adult material, gambling platforms, malicious links, and other unwanted resources that may cause security risks or negatively impact users. Ensuring safe browsing, especially for children, students, and workplace environments, has therefore become a necessity.

This project, Personal Firewall Pro, is a browser extension combined with a backend service that provides safe browsing by filtering and blocking access to harmful or unwanted websites. It allows users to enforce keyword-based filtering, HTTPS-only browsing, and trust checking through a defined list of trusted domains. The extension integrates with a backend server that manages keyword rules, trusted domains, and stores logs of blocked attempts for monitoring.

To make the project more professional, additional features such as password-protected settings, logging, and the ability to update keyword lists dynamically from the backend have been implemented. Unlike traditional firewalls that operate at the network layer, this solution focuses specifically on browser-level filtering, making it lightweight, easy to deploy, and effective for everyday users.

**Objectives:**

**The main objectives of this project are:**

1. To develop a browser extension that blocks unwanted content using keywords and domain-based filtering.
2. To ensure secure browsing by enforcing HTTPS-only connections.
3. To integrate a backend system for dynamic management of rules and storing access logs.
4. To provide password protection for enabling/disabling firewall settings.
5. To support easy monitoring through a log system that tracks blocked attempts.
6. To ensure the solution is lightweight, user-friendly, and customizable.

**2.Project Profile:**

**2.1 Project Definition**

This project is a Personal Firewall Chrome Extension that blocks access to unwanted websites (adult, gambling, crime, etc.) and enforces safe browsing rules like HTTPS-only and trusted domains. It ensures a safe and distraction-free browsing environment.

**2.2 Project Description**

The project integrates a Chrome Extension (frontend) with a FastAPI backend and SQLite database.

The extension monitors user browsing activities.

Keywords and domains stored in the backend database are used to filter requests.

If a request matches the unwanted list, the site is blocked and a log entry is created.

Features like trust checking and firewall toggle give users more control.

**2.4 Problem Statements**

Children and employees can access adult or gambling websites easily.

Browsers don’t provide strong keyword/domain-based blocking.

No simple, integrated solution to log and monitor blocked sites.

Lack of password-protected control, making it easy to disable filters.

Is lightweight, user-friendly, and customizable.

Prevents unauthorized disabling by using password protection.

**2.6 Proposed System & Features**

The proposed system is a Chrome Extension with backend support:

Keyword and Domain Blocking – blocks unwanted websites.

HTTPS-only browsing – ensures safe browsing.

Trust Checker – allows only trusted domains.

Password Protection – prevents unauthorized access to settings.

Logging System – records blocked attempts for review.

User-friendly UI – toggle switches for firewall, HTTPS, trust check.

**2.8 Outcomes**

A working browser firewall extension with backend database support.

A secure browsing environment with no access to adult, crime, or gambling sites.

Easy-to-use interface for toggling security features.

A logging system for monitoring browsing attempts.

**2.9 Tools & Technology Used**

Frontend: Chrome Extension (JavaScript, HTML, CSS)

Backend: Fast API (Python)

Database: SQLite (lightweight local DB)

IDE/Tools: VS Code

Platform: Windows 11

**3.Requirement Analysis**

**3.1 Feasibility Study**

Before building the system, we must check if the project is possible and useful.

We look at:

Technical Feasibility – Do we have the right tools (Chrome extension, Python, database)?

Operational Feasibility – Will users (students, parents, or institutions) actually use it?

Economic Feasibility – Does it save time/money compared to existing methods?

**3.2 Users of the System**

**End Users** – Students or general internet users who browse safely.

**Administrator** – Person who manages keywords, trusted sites, and can view blocked logs.

**Parents/Institutions** – Can lock the firewall with a password to ensure safety.

[Admin] ---> Manage Keywords, Trusted Domains

[User] ---> Browsing ---> Firewall Extension ---> Block/Allow

**3.3 Modules**

Break the project into smaller parts:

1. Keyword Filtering Module – Blocks websites with adult/crime/betting words.
2. HTTPS-Only Module – Ensures browsing only on secure (HTTPS) sites.
3. Trust Checker Module – Allows only trusted domains, blocks others.
4. Backend & Database Module – Stores rules and logs (Python + SQLite).
5. User Interface Module – Chrome extension popup for toggles & password lock.

**3.4 Process Model**

This project follows the Incremental Development Model:

Step 1: Build simple keyword blocking.

Step 2: Add HTTPS-only rule.

Step 3: Add trust checker.

Step 4: Add backend logging & password lock.

(Incremental Model):

[Step 1] Keyword Filter

+

[Step 2] HTTPS Only

+

[Step 3] Trust Checker

+

[Step 4] Backend & Password

3.5 Hardware & Software Requirements

Hardware:

Laptop/Desktop with at least 4GB RAM, 2GHz processor.

Storage: 500MB free space.

Software:

OS: Windows / Linux

Browser: Google Chrome

Programming: Python (FastAPI, SQLite3), JavaScript (Chrome Extension)

**3.6 Use Cases**

Use cases describe **how a user interacts with the system**.

**UC1:** User tries to open an adult site → Firewall blocks it.

**UC2:** User visits a secure site (<https://google.com>) → Allowed.

**UC3:** User visits unknown HTTPS site → Blocked if not in trusted list.

**UC4:** Admin views blocked logs → Can see history.

**UC5:** Admin changes password → Locks/unlocks firewall.

**4. System Design**

**4.1 Data Flow Diagram (DFD)**

A DFD shows how data moves in the system.

**Explanation:**

The user browses a website.

The request goes through the Firewall Extension.

Firewall checks:

Keyword filtering

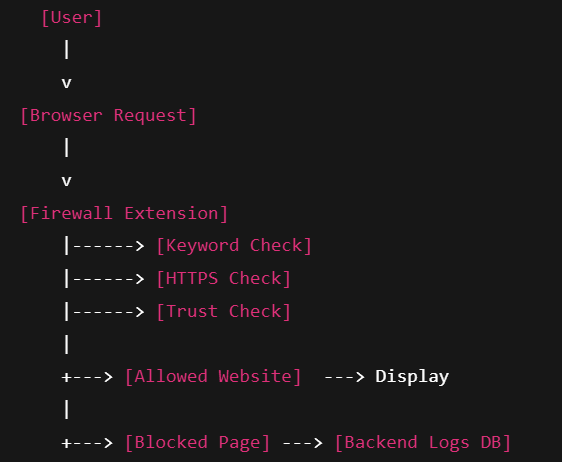
HTTPS check

Trusted domain check

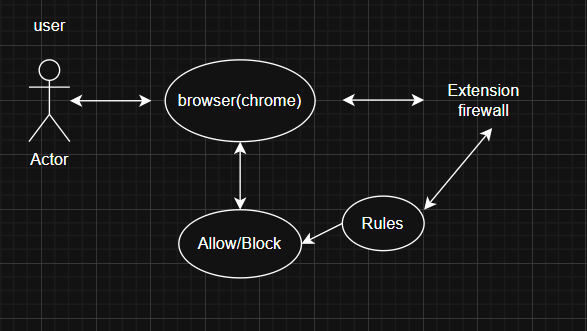
If blocked → “Blocked Page” is shown and Log saved in backend.

If safe → Website loads normally.

(DFD Level 1):



**4.2 Use Case Diagram:**

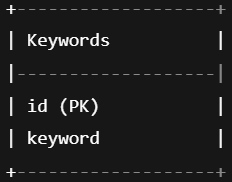
****

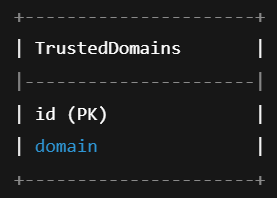
**4.3 Entity Relationship Diagram (ERD)**

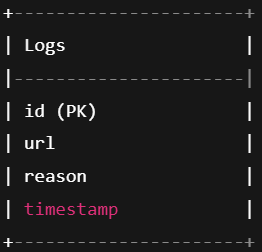
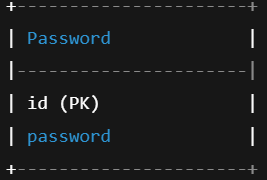
ERD shows how your database tables are related.

Tables in DB (SQLite):

1. firewall\_keywords → stores blocked keywords.
2. firewall\_trusted\_domains → stores trusted websites.
3. firewall\_logs → stores blocked URLs with reason + timestamp.
4. firewall\_password → stores admin password

****

****

** **

1. **Implementation**

**5.1 Frontend (Chrome Extension)**

The frontend is built as a Chrome Extension.

It provides a popup UI with buttons/toggles:

Firewall ON/OFF

Keyword filter ON/OFF

HTTPS-only browsing

Trust checker

A password lock system is included so only authorized users can disable the firewall.

It uses HTML, CSS, JavaScript and connects to the backend via API calls.

A blocked.html page is shown whenever a restricted website is accessed.

Chrome Extension Popup UI → Background.js → Browser

**5.2 Backend (Fast API & Database)**

The backend is built using Fast API (Python).

It provides API endpoints such as:

/rules → fetches list of keywords and trusted domains

/logs → stores logs of blocked sites

The backend acts as the control center

where rules are defined and logs are stored.

Communication with frontend happens over http://127.0.0.1:8000.

Frontend Extension ⇄ Backend API ⇄ Database

**5.3 Database Schema**

1. firewall\_keywords – stores blocked words.
2. firewall\_trusted\_domains – stores safe websites.
3. firewall\_logs – stores browsing logs (url, reason, timestamp).
4. firewall\_password – stores admin password (for locking/unlocking firewall).

Small ER Diagram (already created earlier) can be reused here.

**5.4 Integration**

The frontend and backend are integrated through REST API calls.

Workflow:

* 1. User toggles firewall in popup → state saved in Chrome storage.
  2. Background.js fetches rules from backend (/rules).
  3. If a site matches keyword/HTTP rule → site blocked → log sent to backend (/logs).
  4. Admin can later check logs via backend API or database.

Browser Request → Extension Filter → Backend Rule Check → Block/Allow Decision → Log Saved

**6.Features of Personal Firewall Pro**

**6.1 Keyword Filtering**

Blocks websites containing specific unwanted keywords such as adult, crime, or betting content.

The keyword list is stored in the backend database, and new keywords can be added easily.

Prevents access even if the site is HTTPS, by scanning the URL for keywords.

**6.2 HTTPS Enforcement**

Forces browsing only through secure HTTPS connections.

Automatically blocks unsafe http:// websites.

Protects against insecure data transfer and possible attacks.

Example: http://example.com will be blocked, while https://example.com is allowed.

**6.3 Trusted Domains**

Allows access only to pre-approved safe websites.

Trusted domains include educational, news, and official sites like Google, Wikipedia, and BBC.

Users cannot add fake websites because trust is managed in the backend.

Example: https://wikipedia.org opens normally, but https://randomsite.com is blocked.

**6.4 Logging System**

Every blocked request is recorded in the backend database.

Each log contains:

Website URL

Reason for blocking (keyword/HTTP/untrusted site)

Timestamp

Helps admin track browsing activity and detect repeated unsafe access attempts.

Example: Logs show attempts like “Blocked

http://badsite.com - Reason: Insecure HTTP”.

* 1. **Password Protection for Firewall Toggle**

Firewall toggle is **locked with a password** to prevent unauthorized changes.

Default password: admin123 (can be changed in the extension).

Ensures that only an **authorized admin** can disable/enable the firewall.

Example: If someone tries to turn off the firewall, they must enter the password

**7.Testing, Results & Demonstration**

**7.1 Test Cases and Results**

Different scenarios were tested to check the accuracy of the firewall.

Test Case Examples:

* 1. Searching for adult keywords → Website blocked.
  2. Opening HTTP websites → Blocked with warning page.
  3. Opening trusted domains like Google/Wikipedia → Allowed.
  4. Opening unknown HTTPS sites (with trust check ON) → Blocked.
  5. Trying to turn off firewall without password → Denied.

All test cases worked as expected after bug fixes.

**7.2 How the Project Works (Execution Flow)**

**step-by-step workflow** of the project:

**Starting the Backend**

Open CMD and navigate to the backend folder.

Run:

python -m uvicorn backend:app --reload

This starts the FastAPI server at [**http://127.0.0.1:8000**](http://127.0.0.1:8000).

**Loading the Extension in Browser**

Open Chrome → chrome://extensions/.

Enable Developer Mode → Load Unpacked → Select extension folder.

The extension icon appears on the browser

**Communication Between Extension and Backend**

When you enable firewall toggle, the extension requests rules from the backend.

Rules (keywords + trusted domains) are applied dynamically.

Blocked attempts are logged into firewall.db by the backend.

**Browsing with Firewall ON**

If user opens a blocked site → redirected to blocked.html.

If site is trusted → page loads normally.

If site is unknown (with trust check ON) → blocked.

**Checking Logs**

Visit [**http://127.0.0.1:8000/logs**](http://127.0.0.1:8000/logs) in browser.

Shows all blocked sites with reasons + timestamps.

**7.3.System Demonstration (Screenshots)**

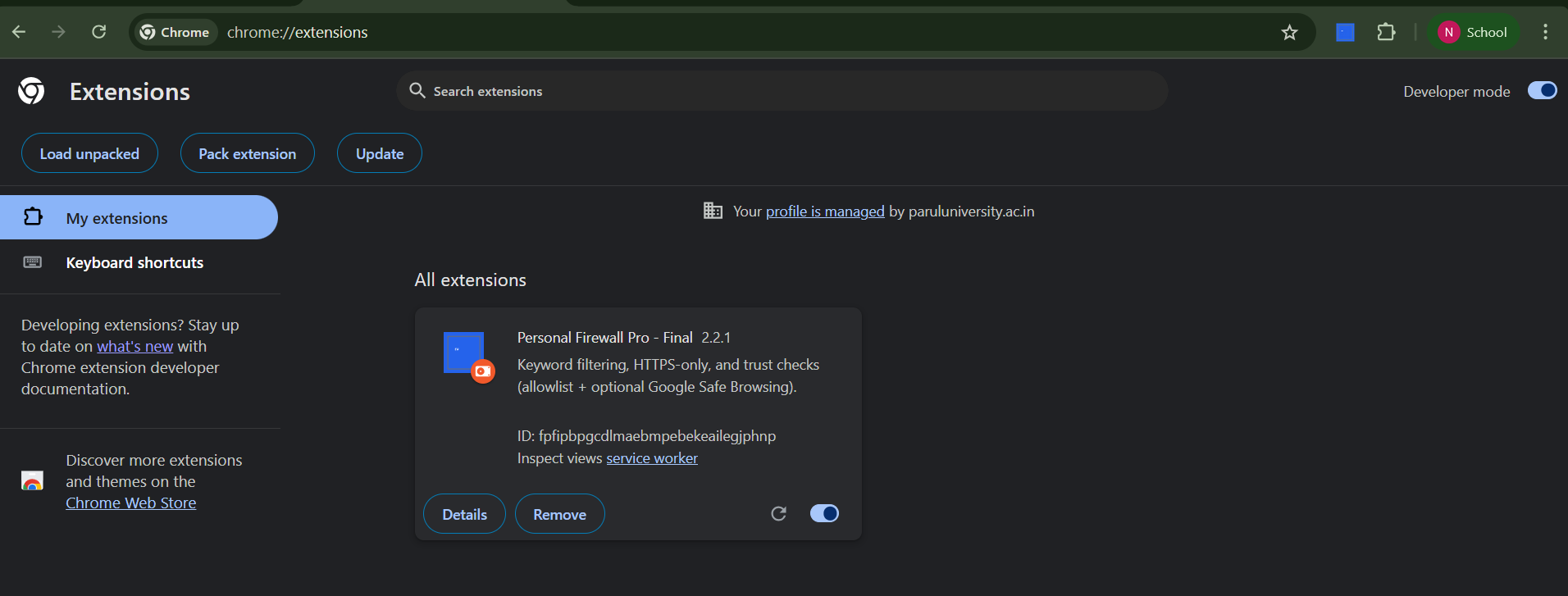
This section explains how the Personal Firewall Pro extension works step by step, with screenshots to show each stage of the workflow.

Step 1: Loading the Extension into Chrome

Open Chrome → Extensions → Manage Extensions.

Enable Developer Mode.

Click Load Unpacked and select the folder personal\_firewall\_pro

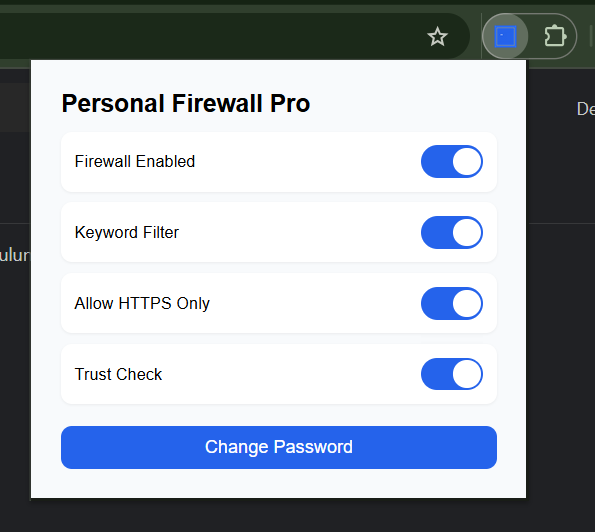


**Step 2: Opening the Extension Popup**

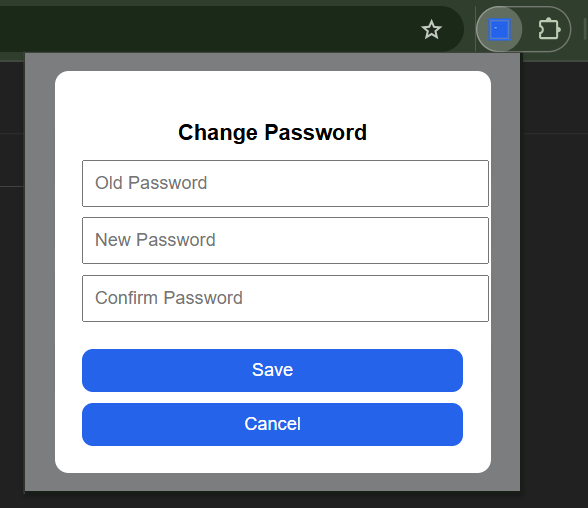
Click on the extension icon.

A popup appears showing toggles:

* + Firewall Enabled
  + Keyword Filter
  + HTTPS Only
  + Trust Check



There is also a **Change Password** button.

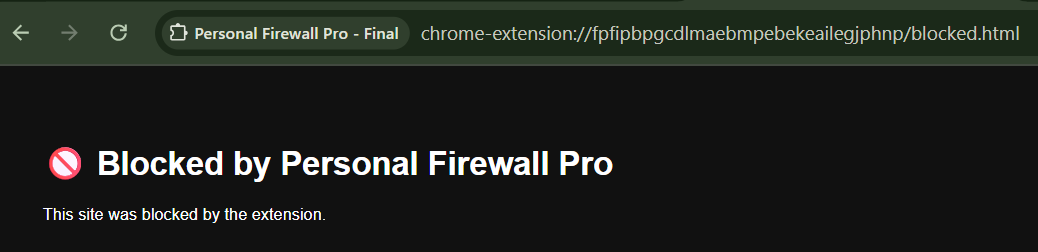


**Step 3: Blocking a Keyword Website**

Try to search for or open a blocked keyword site like adult words or unnecessary words it will block

The system immediately redirects to the **Blocked Page (blocked.html)**.

Screenshot: Blocked page displayed in browser.



**Step 4: HTTPS Enforcement**

**Points:**

Blocks all HTTP sites automatically.

Redirects to blocked page when HTTP site is opened.

Forces only secure HTTPS browsing.

One of the major security features of Personal Firewall Pro is its ability to enforce **HTTPS-only browsing**. This means that whenever a user tries to open a website using the insecure http:// protocol (for example, http://example.com), the firewall intercepts the request and immediately redirects the user to a blocked warning page. This protects the user from unsafe, unencrypted websites that could expose sensitive data to hackers. By ensuring that all browsing takes place over **HTTPS**, the firewall guarantees that the connection is encrypted and secure, reducing the risk of data theft or man-in-the-middle attacks.

**Step 5: Trusted Domains Check**

**Points:**

Works only when Trust Check is enabled.

Allows access to pre-approved websites (Google, Wikipedia, BBC, etc.).

Blocks all untrusted HTTPS sites (e.g., randomsite.com).

Prevents malicious or fake websites from loading.

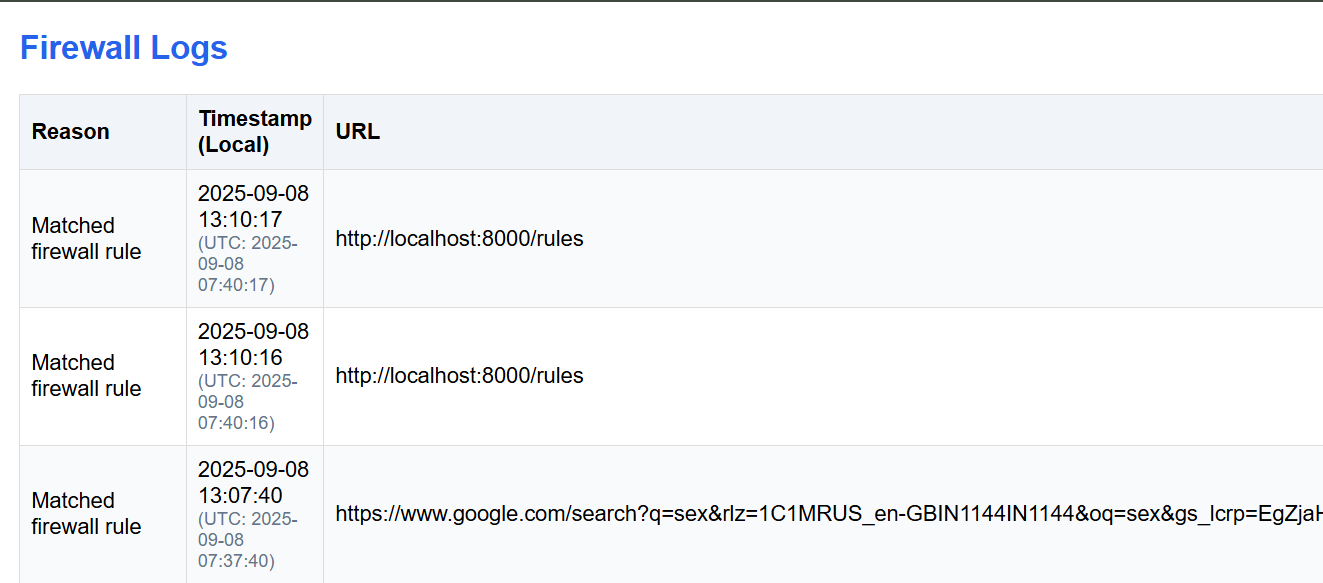
The **Trusted Domains Check** adds an extra layer of safety by allowing users to browse only those websites that are included in the **trusted allowlist**. These trusted sites are managed in the backend database and typically include safe and reliable platforms such as google.com, wikipedia.org, and bbc.com. When Trust Check is ON, any website outside this allowlist is automatically blocked, even if it uses HTTPS. For example, visiting wikipedia.org will be allowed, but trying to open an unknown site like randomsite.com will be blocked and redirected to the warning page. This feature ensures that even **HTTPS websites, which can sometimes be fraudulent**, do not bypass the firewall unless they are verified and trusted.

**Step 6: Viewing Logs in Backend**

Open the backend logs at http://127.0.0.1:8000/logs/html

All blocked attempts are recorded with URL, reason, and timestamp.

Screenshot: Backend logs page.



**7.3 Advantages and Performance**

Ensures safe browsing by blocking harmful and adult content.

Works in real-time (blocking happens instantly).

Lightweight and doesn’t slow down browsing.

Can be extended with more keywords, trusted sites, or APIs.

**Conclusion**

The Personal Firewall Pro successfully demonstrates how browsing can be made safer by combining a Chrome extension frontend with a Python FastAPI backend. The system enforces keyword filtering, HTTPS-only browsing, trusted domain checks, and activity logging, all while protecting the firewall toggle with a password. Through testing, it was observed that unsafe or inappropriate websites were effectively blocked, while trusted sites were allowed without issues. This project not only provides a practical solution for safe browsing but also highlights how backend–frontend communication and database logging can enhance user security

THE END