SHIELDING USERS FROM DIGITAL DECEPTION: AI-POWERED DETECTION OF FRAUDULENT COURSES AND BUSINESS SCAMS

[1]T.M. Mohammed Marzuk, [2]Dr.S. Latha

[1]Master's Degree Student, [2] Director,
[1]Department of Computer Science,
[1]Dr.MGR Educational and Research Institute, Chennai, India,
[2]Centre for Cyber Forensics and Information Security, University of Madras, India

Abstract : In the world of internet, Digital Marketing has a wide potential to promote any kinds of products and services including educational and informative media. Do you ever check the legitimacy of such advertisements or news? Whether those so-called pop-ups and ads are really posted by trusted sources and companies that has proper credentials. Especially in India, where the amount of people having access to internet is very high, possibility of selling such fake courses and business is also very high due to abundance of youngsters who got very less knowledge about how a business works and on need of quick money due to poverty. These scammers majorly promote their work through social media influencers or they themselves be an influencer to exploit the trust built between them and the audience [11]. People often trust and don't check the legitimacy of the things these influencers advertise due to the face value of them and this is the way most of the big scale online scams occur using social media. In this paper we majorly focus on AI Powered Extension framework capable of detecting malicious Fake Phishing Sites that disguises themselves as legitimate sites. It is capable of using the entire page content to detect presence of Fake colleges, Universities and Educational Institutions in the list of official UGC and AICTE Websites. It can also take User input and classify them as AI generated or Human along with Fact Checking.

IndexTerms **- Fake Institutions, Domain Validation, Phishing Sites, Fact Check and Course Scams**

I. INTRODUCTION

Introduction of Computers and Internet has brought up a revolution in many sectors including Education and Marketing. Use of web and social media to access media and make use of it for business isn't anything new. It is a wide technique being practised for a long time and its introduction dates back to the 90s. Digital Marketing blooming happened in 2000s with many varieties of methods like email marketing, e-commerce marketing, influencer marketing, social media marketing and so on [2]. Use of internet to provide online education was started in 1984 by University of Toronto with the course titled "Women and Computers in Education" via computers that were interlined to form a basic network [3]. In 1989, the University of Phoenix started to conduct both Bachelorette and Masters course through the internet [4]. Both Digital Marketing and Online Education paved the way to influence the increase of Online courses. Online courses provide the ability to attend a course from major institutions and universities from all over the world without being present in the location. Skill based and Professional courses or any other type of learning can be achieved by online courses and accessing study material is particularly easy. Online Community and Discussion Forum in educational platforms can be rather helpful in addressing any issues and to clear doubts that may have arisen in classes. Even though, there are many useful and verified educational platforms offering courses from institutes and universities, there are also a very high number of fake platforms and websites in names of colleges offering fake courses and courses without any credentials. Some of these websites chooses their domain name with names of real renowned colleges and universities to spoof their identity to fool internet users and sell courses among them. India is a country in which education is being considered as a way to improve their lives by many lower- and middle-class people. From having only 36% of literacy rate in 1980, India has improved its literacy rate exponentially up to 77% in the year 2023 [5]. Indian Government has made it compulsory for children to undergo education up to 8th grade (Primary and Middle School). Parents urge their children to pursue higher education to open up more choices in the working sector. Students use internet as a medium to learn new skills and offering fake courses to those students will turn them into victims of cybercrime. Most of these people may not realize to the end that the course they completed was a fake one until later. Fake courses aren't the only thing to be careful about but there is also usage of social media by culprits to promote business and products without having proper permission and documents to do so [8]. Some of these products sold might cause medical, physical and emotional harm. Example for such incident is when a famous Tamil youtuber PORCHEZHIYAN (Saapatu Raaman YouTube channel) got arrested for prescribing medicine without proper qualification during corona period [6]. There are more youtubers like him who still continue to prescribe medicine without proper qualification which ranges from normal fever, cold medicine to fertility drugs causing major problems to women. Many Phishing Domains and URLs are used to propagate pages that are capable of downloading malware which create a vulnerability or an attack surface which can be later exploited by an attacker and spoof legitimate sites like business domains, e-commerce platforms, mail platforms, login pages, bank webpages and educational sites to fool users to enter sensitive information like Credentials, Bank Account details, Transaction Details, Business Secrets.

II. RESEARCH METHODOLOGY

2.1 Objectives

To Create a Browser Extension which prevent users from getting Phished and fooled by malicious attackers and consequences of being a victim of such cases

Deciding and gathering details on features required for the Extension

- Domain, URL Matching from Dynamic Blocklist
- Web Page Content Scanning to Match Fake Institutions
- Fact Checking and Possible AI Detection from User Inputs
- User Friendly Secure Interface with clear warning banners

2.2 Design Phase

2.2.1 Workflow Architecture for Domain Matching and Fake Institutions

Threat Intelligence Dataset:
There are Threat intelligence sites and communities that updates Phishing and Malicious Domains, URLs Periodically on their databases which are available public in various data formats. They were collected, de-duplicated and merged along with manual lists using the python script and then organized in separate Json files based on their sources and manifest. Json (GitHub) which tells the extension if there are files available to download.

Figure 2.1: Threat Intelligence Dataset

Content.js Scripts (Blocklist and UGC, AICTE)
Manifest. Json (Extension) lists permissions needed for extension to work properly. The Background. Js refreshes the Blocklists and stores it in IndexedDb and local browser storage during Installation and every 24 hours at UTC 1:15. Storing in IndexedDB helps to process faster and local storage acts as a fallback. Content_blocklist.js and Content_ugc_aicte.js is injected into webpages

Warning Banner
Every page is analysed to match domains and page content with blocklist and if a match was found, it will trigger a warning banner to indicate that site or page content is malicious

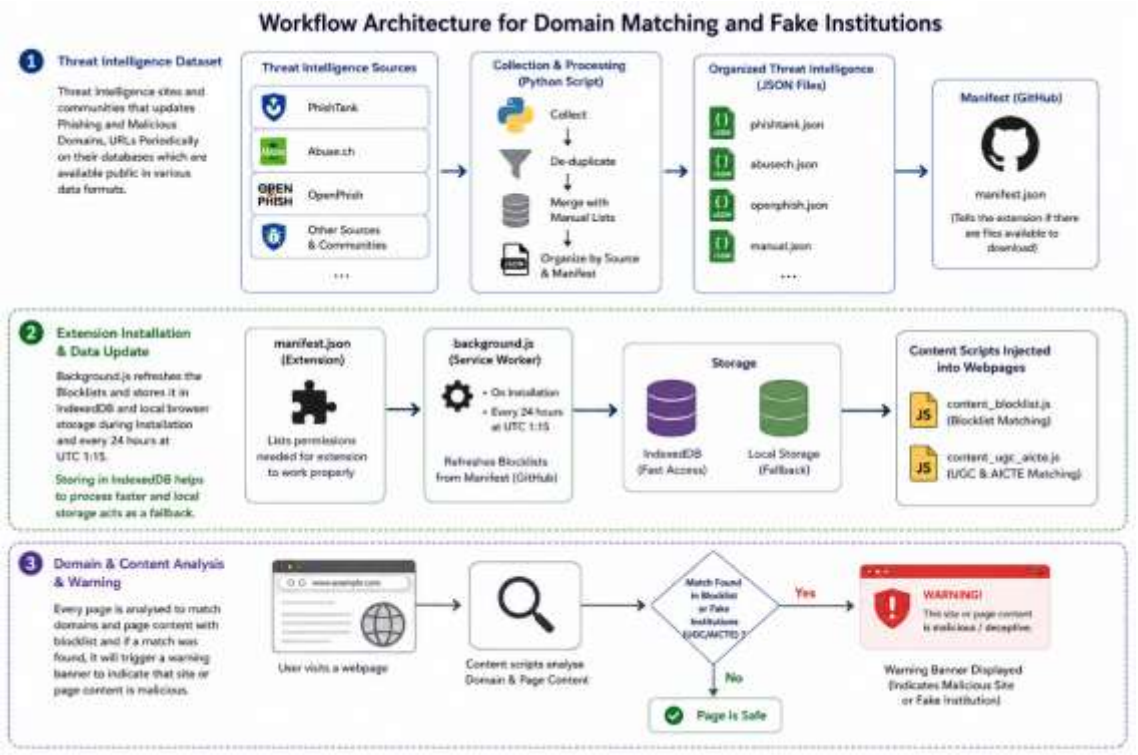

Figure 2.1: Workflow for Domain and Fake Institutions

2.2.2 Workflow For Fact Checking

Popup.js acts as an Intermediary between users and Backend, when the user inputs a text and chooses the fact check option, the popup.js connects to proxy server with the request. The Proxy server again forwards the request to backend using API Connection to Hugging Face Space which has AI Models in it which uses Google custom search API to send evidences regarding the input. The Evidence is sent back by Hugging face to proxy server which is returned to Browser. The Browser caches this result to fast retrieve if the fact check was again used for same input. The Results are showed to the user in a User-Friendly HTML Web Page.

Figure 2.2: Workflow for Fact Check

2.3 Development Phase

2.3.1 GitHub

manifest.js
It acts as list that the update_and_spli_blocklist.py should use to process for all the blocklist sources, Manifest. Json has two data structures in it

- File: The Name of the Json file where the domain, URL dataset will be stored
- Source: The Name of the site or Community in which the dataset was taken from

Manifest. Json is used by Update_and_split_blocklist.py when it starts execution and it knows which file to access when updating a specific source. When update_and _split_blocklist.py needs an object from urlhaus, it uses manifest. Json to learn that it will be available in urlhaus.json so adding a new source requires adding a file and source data to the manifest. Json

Custom.json – It is also added to manifest. Json to refer to source and file that are added to extension by the users.

Update_blocklist.yml
This is a GitHub Actions Workflow script which is responsible for dynamic delivery of blocklist to the extension. The script uses cron schedule (0 1 * * *) to run this yml script at 01:00 AM UTC every day. It also has a manual run workflow option which can be used for testing and debugging. This script is used to create a latest version virtual machine of Ubuntu environment and copies all the GitHub codes to the environment. It installs all the necessary libraries and dependencies that will be needed to run the update_and_split_blocklist.py script. After all the necessities are satisfied, it will run the update_and_split_blocklist.py which will generate blocklists from multiple sources. The update_blocklist.yml will finally push those blocklist files to the GitHub using a secure token and authentication bot so changes to GitHub repositories can only done by a trusted entity. The extension uses the GitHub updated blocklists for its matching processes.

update_and_split_blocklist.py
This is also an automation script that is used to run a series of steps. It will start with downloading libraries and dependencies it needs to complete all of its goals. It needs several libraries to read Json files, clean URL and domains, read HTML files and even Pdf files to collect data from such files to generate blocklists. It will take an entire URL and clean it to all lowercase and regular domains. Collect blocklist feeds from HTML, txt files and even from Pdf files. It is capable of avoiding same input record from a same source multiple time. The script merges all blocklist feeds including manual list and custom user input list as a single blocklist. Json file along with their source details. The script records data and its respective sources from single blocklist. Json and make a copy so they are recorded as data from their respective source in their own Json files (e.g.: urlhaus. Json). This record is stored in manifest. Json automatically so extension will know which source blocklists are present in which files so it can be downloaded for use.

2.3.2 Extension Folder

manifest. Json

manifest. Json present in Extension folder is not same as the one in GitHub. Here manifest. Json is considered as an identity of the extension. It specifies the versions, name and icons of the extension. It acts like a blueprint that represents what are the resources that the extension will need to run and process information properly

EduShield (Extension Name) uses manifest version 3, which is the latest version with service worker-based structure and has new strict security policies

Manifest. Json offers permission to access all URL in browser to match it with blocklist using content scripts. It dictates background.js as service worker, gradio. Js to connect to backend and popup.html as user interface file for the extension popup.

EduShield requests permissions using the manifest. Json file which includes

- Storage: To read and write web browser's local cache with blocklist data
- Tabs: To give access to tab information like URL and Domains
- Scripting: Permission required to run scripts on webpages
- Alarms: To run scheduled refresh of blocklists
- Unlimited storage: To avoid chrome browser limit of 5Mb to store local storage data
- Notifications: To Show popup warnings on the web pages itself
- Downloads: Access to download manager of browser to export allowlist files

Background. Js

Background. Js is the service worker of the EduShield extension which is a type of JavaScript that is responsible for tasks that occurs in the background and not specific to any pages. It resembles a communication system for the entire extension. It works only when a process occurs, waits for some time to receive any others and goes back to dormant state when there is no task assigned in a certain time.

During installation of extension, background. Js activates its listener that communicates with other parts of the extension. It uses mergeAndStore() function to fetch manifest. Json (GitHub) to check which files have blocklists present in it, it visits those files to receive and store blocklists from GitHub in browser's local storage. It uses fetchAndStore_UGC_AI () function to get lists of fake institutions. Background.js not only saves blocklists in local storage but also in IndexedDB which doesn't have storage limitations of 5 to 10Mb like local storage. IndexedDB files are stored in two methods, the blocklists gathered each day is stored in their Date as file names and as master file. The master file gets merged with daily blocklist and current blocklists. When the daily blocklists reaches its 30th day, it will get dumped. In the case of master blocklists file, even if an entry has received before 30 days but it has been seen again in another daily blocklist which is less than 30 days, the entry will still exist. Just like GitHub yml script, the background. Js also has a schedule to get updated blocklists and store every day. Custom lists added by users are also stored in local browser storage with help of background. Js. It saves the blocklists in many small chunks that are easy to store and manage instead of a single blocklists that has a large size and can't be handled due to browser size limit. Listener function in the script is used to send commands and messages to other extension parts like get blocklist, refresh blocklist, get allowlist and add to allowlist, etc.

Content_blocklist. Js

Content_blocklist. Js is a self-invoking function (Function that runs automatically without being called to run). It also safe helper functions to avoid potential runtime errors. This script is the first line of defence present in the extension. This script runs in content of a web pages so it can inject itself inside web pages and show warning banner whenever needed. The checkDomain() function is the primary code function that takes the URL from the current web page and checks it against the allowlist URL, users had added. If Allowlist doesn't have that URL, the script runs it with latest blocklists to find an exact match and fuzzy match (Typo squatting detection). The showBlocklistWarning() function is used to create and display a warning banner with specific colours and styles. Some modern websites don't need a full refresh to load new content to web pages which are known as Single Page Applications so the extension uses to MutationObserver to watch for changes in the web pages so it can automatically recheck the URL If it has changed.

The Banner has three options along with the warning:

- Learn More: Displays the reason for warning and source that flagged the Domain or URL
- Dismiss: Hides the warning banner until refresh of the same web page
- Allow: Send the current Domain to allowlist so the user won't see warning banner again for that web page

Content_ugc_aicte. Js

Content_ugc_aicte. Js script also use some exact features of content_blocklist. Js which includes self-invoking function, safe helpers and SPA (Single Page Application) support to watch for changes in the Page content instead of URLs. The loadManuallists() function directly connects to GitHub repository to collect manual_ugc.json and manual_aicte.json and cache it in local storage. The extractTextFromNode() function is used to retrieve texts from Document Object Model (DOM) which includes direct HTML elements and shadow DOMs which are used by modern sites to fetch content to web pages. The getPageText() function takes the text from extractTextFromNode() function and cleans it, splits into individual lines and remove extra spaces. The checkManualNames() function reviews each text line to match it with the text from UGC or AICTE list names. The showWarning() function shows a warning banner if a match was found

The warning banner shows a learn more option which opens a new fixed size popup that lists the text that got flagged and its respective source (UGC or AICTE).

Popup.Js

Popup. Js is the control centre for all of the extension's user interface. It is responsible for taking user actions in the extension popup and converting to commands for rest of the extension parts which includes display of data, manage user inputs and fact check initiation. The renderAll() function fetches details regarding locally stored allowlists, custom domains and pass it to renderlist() function which creates a visual interface of those lists in the popup. Popup. Js uses background. Js to fetch those details using render functions from local storage. Popup. Js has several listeners to manage allowlist, it has functions like add-to-add new allowlist using text, import to add allowlist using txt files and export to download the already present allowlist as a txt file. The clear options is used to remove individual or all allowlists. For the custom feeds options, it has a file upload support and url support to extract feeds. The popup.js is used to trigger popup.html to show notifications of total blocklists and added/removal of blocklists using custom feeds and feed URL. The show more options open a new popup that shows all the allowlist or custom feeds added. The runFactCheck() function is used to provide a series of functions such as input box with character count support and disables run fact check button when it reaches the character limit of 1750 chars. It shows a spinner clock while loading and sends the input to local proxy. If a connection can't be created, it shows error to connect warnings.

Popup.Html and CSS

Popup.html is used to provide a structure and organise sections in a user-friendly manner. Popup.css is responsible for providing colours, patterns and styles to each section and buttons present in it. Html uses <section> and <h1> to separate sections with their heading and provide a logical view. Popup.html is responsible for showing total merged blocklist and count of new blocklist added using custom feeds and feed URL. It also shows notifications for removal. These notifications are displayed for 3 seconds and fades away. The sections in the html includes allowlist, custom feeds, Feed URLs and AI Fact Check. Each display and interactive buttons have their unique id (allowInput, addAllowBtn and factCheckInput) to define their styles. The appearance and display of all elements are defined by CSS by defining them like primary-btn, secondary-btn and clear-btn with each having their own colours, 3d pressing looks, hover effects and spinner clock icons to indicate loading or processing. The <script type="module" src="popup.js"></script> tag at end of the html script links the sections and components to popup. Js so when those sections are accessed by user, the popup.js will be able to convert them to commands.

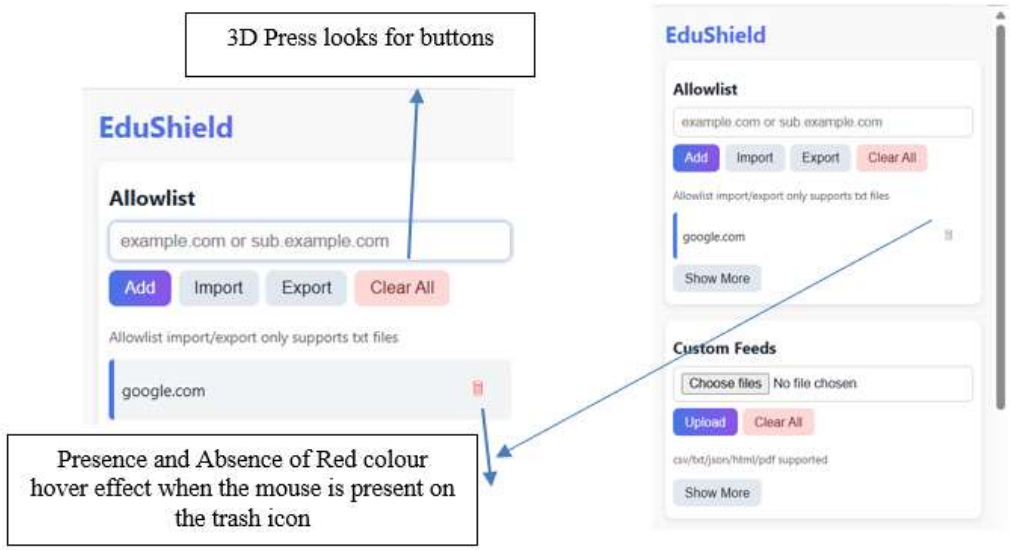

Figure 2.3 Extension Popup (First Half)

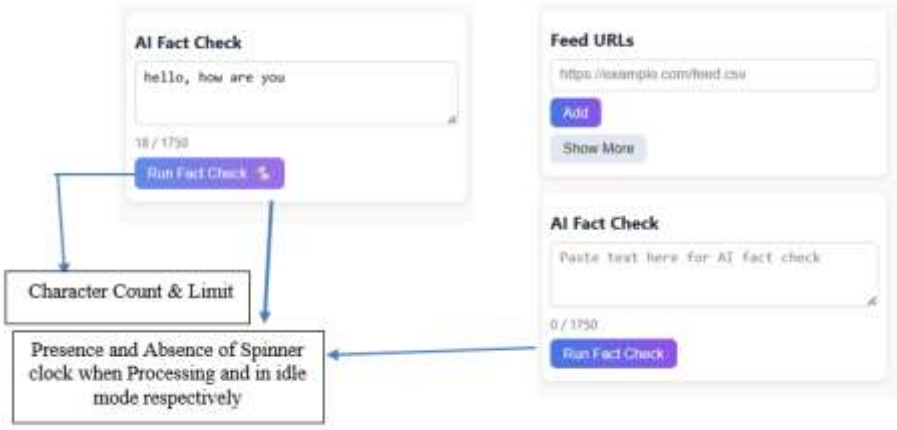

Figure 2.4: Extension Popup (Second Half)

Listview

Listview.html creates a basic interface structure for a separate full popup that is used to serve as a container that will be used by listview. Js to update with content. Linkview.html uses popup.css as a reference to make styles of the new popup similar to the primary extension popup. Listview.html uses unique elements with id such as listTitle, listContainer and backBtn that can be applied based on correct loaded list. Both listview.html and listview. Js is used to show details already added by user in sections of Allowlist, Custom Feeds and Feed URLs. Show more button on the primary extension popup under sections of allowlist, custom feeds and feed URL triggers the listview popup. Listview.Js uses URL based data fetching which can be used to fetch multiple data with a single script. The script fetches the details of the list and renders it inside the list container created by listview.html. It adds trash button which can be used to remove inputs present on the list and saves the updated list. Back option is used to return back to primary extension popup. In Allowlist section, the primary popup can show up to 5 list entries and if more present show more needs to be used to see all of them. In other sections like custom feeds and feed URL, amount of list entries on primary extension is lower than allowlist value of 5 entries.

Factcheckview.html and Factcheckview.Js

Factcheckview.html is similar to to listview.html. Its job is to create of structure of space that will be used to factcheckview.Js with content. Factcheckview.html has a simple structure with a heading and a body where the content will be pasted by factcheckview.js, until then it will continue to show “waiting for fact-check results”. HTML create a space with fixed width and length so long results needed to be viewed by scrolling the page. <script src="factcheckview.js"></script> tag at the end of html connects the space with factcheckview.Js. The Js uses DOMContentLoaded listener to make sure that entire html page has been loaded before updating results to the page. The waitForResult() function is used to wait but check if entire result has reached the browser’s local storage from the proxy server. The renderData() function is used to analyse the received data and recognize the type of data received (Array, Objects or primitive). It creates a html based on the structure the data type needs. It creates section for Results of fact checking which are Claims, AI Detection and Google Evidences.

2.3.3 Local Proxy Server

Factcheck_proxy.py is built upon the lightweight flask framework, it acts as an intermediary between browser extension and the backend by routing requests and results back and forth. An Intermediary server is needed cause chrome browser security policies doesn’t allow extensions to make API calls or run scripts from external sources but when API is received from another source, the browser accepts it as legible. The localhost address 127.0.0.1 and Port 5000 is used as the proxy server for this extension which has flask and gradio client installed to make API calls to Hugging Face Space Backend URL. The factcheck() function is used to receive incoming requests. The script also checks the content type to male sure that it has a string input. API Name /predict is the name of the API running on the backend URL so the input is sent there for further processing.

2.3.4 Backend (Hugging Face Space)

Requirements.txt

- Gradio – This is a library used to create a web interface to use for the AI backend
- Transformers – Main Engine that gives access to AI Models
- Torch – Foundation framework that is used by Transformers to create blocks for running neural networks to run AI Models
- Sentence-Transformers – This library is used as add on for AI Models to make them understand sentences and their meanings better
- Requests – This is a python library used to make HTTP requests
- Feedparser – This library allows AI to understand Feeds like news and blogs
- Huggingface Hub – This is used to download AI Models, Datasets and managing them to communicate them with Huggingface Spaces

App.py

App.py is the backend python code hosted on cloud platform Hugging Face Space to use as Public facing API. The Script loads MoritzLaurer/DeBERTa-v3-base-mnli model for Zero Shot Classification (Not trained in those categories but will work using its

language processing Capabilities) to classify text into categories (factual claim, opinion, Personal Anecdote and others). It also loads roberta-base-openai-detector model to differentiate the user input as AI-Generated or Human. Roberta-base-openai-detector model is often called as Discriminator for its abilities. Google/embeddinggemma-300m model is used for semantic matching which helps the backend to understand the meaning of user inputs. The Script uses ThreadPoolExecutor to take multiple sentences and split those texts from each sentence parallel to increase processing speed. safe_split_text(text) functions are used to split entire sentences to line of text using period(.) but it makes sure not to split periods between numbers to avoid breaking decimals ($1.5). The Split texts are referred to as claims and are classified into categories as factual claim, opinion to prioritize the texts with highest label to be sent for analysis. Claims are sent to AI Model for AI detection. The Script sends the user input texts to Google Custom Search using API calls in two modes which are keywords based and semantic based. They retrieve top 3 title and evidence that better suits the input which provides citable evidence related to input claims in each mode. Keyword method matches important words from user input to match with google custom search API and Semantic tries to understand the meaning of input so to retrieve more accurate results.

The API Name of the Backend is Predict and it has two Parallel Analysis modes. Full text analysis sends the entire input text for AI Detection and Fact Checking. Claim Extracted Analysis sends each claim as separate input for AI Detection and Fact Checking.

III. RESULT

Features of the extension were Validated by visiting a number of malicious sites and sites that have mentioned the names of fake institutions present in the UGC and AICTE lists to check the display of warning Banners along with their sources. Allow and dismiss options properly added the URL to allowlist and removed the banner respectively. Popup was validated using adding allowlists using text, importing and exporting the current entries in the allowlist. Custom feeds section was checked by uploading files to add malicious domains. Add Feed URL option loaded all the Domains present in added feeds. Clear all option removed all entries present in each section. Fact Checking is validated by using texts with multiple claims and it was properly classified and separated as claims to get detected as AI generated or Human. It provided Evidences from Google Custom Search API to fact check the user's input. Functions of Indexeddb to store blocklists of each day and merging new inputs with master blocklists can be validated using background service worker of the extension using verifying code that allows to see the current day by day blocklists and master list counts along with the ability to verify that all inputs in daily blocklists are present in master blocklist.

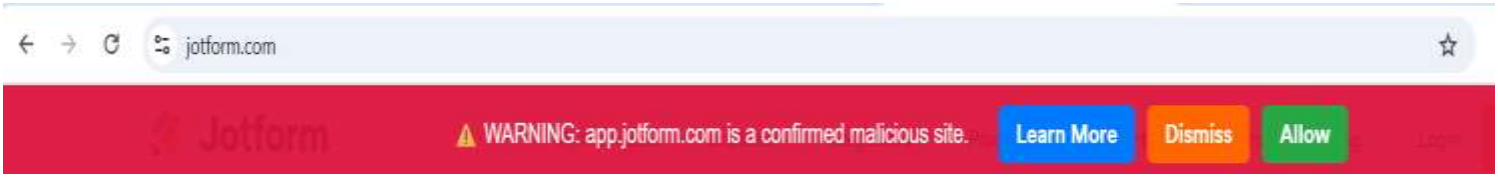

Figure 3.1: Blocklist URL warning

Figure 3.2: UGC&AICTE Warning

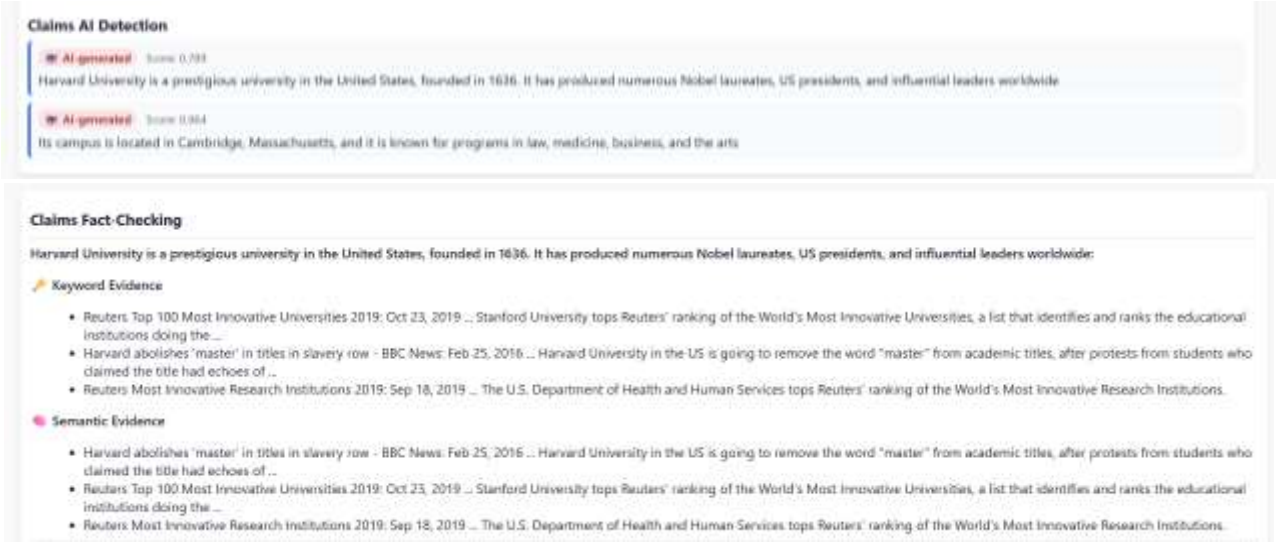

Figure 3.3: AI Detection and Fact Check

IV. SUGGESTIONS

In Its current state, EduShield extension can only store up to 30 days' worth of blocklist in its IndexedDB storage and then it will get dumped cause it can only handle up to a certain amount of blocklist. Its performance threshold can be improved to hold blocklist for much longer time periods while making sure that it is still fast enough to query and alert the users with warnings. Along with UGC and AICTE, more datasets can be added and managed for fake Educational Institutions [12]. It is recommended to create a community to support with maintain blocklists and to avoid potential false positives. EduShield extension uses pre trained AI Models for categorising input (MoritzLaurer/DeBERTa-v3-base-mnli model), detect them as AI Generated or Human (Roberta-base-openai-detector model) and for Semantic Matching(Google/embeddinggemma-300m model). AI Models specifically trained for these features can provide more accurate results [13]. Right now, the fact check component checks the Google custom search API using keywords so it can only provide evidences related to the keywords not exact evidences and even Semantic matching isn't entirely accurate(which can be irrelevant sometimes). Employing specifically trained AI Model can help to understand and summarize text sentences as Multiple single line to retrieve exact evidences. Google Custom Search API only offers 100 queries in its free tier so premium tier and multiple evidence check sources can be added along with local evidence fallback sources.

V. CONCLUSIONS

Phishing and malicious sites are a go to method for attackers to reach a wide range of potential victims in both short and prolonged time. Phishing attacks always can't be identified by looking for differences in content, grammar mistakes and such. More than 40% of phishing using mails are made from verified email domains [1]. In 2024 itself, the estimated loss of capital around the world is said to be at 17.4 billion US Dollars [1]. General Public are targeted for phishing attacks for up to 26% but they can't identify every attack and protect themselves from being a victim of these crimes [1]. The EduShield extension uses multiple datasets from various threat intelligence sites and communities to create a dynamic blocklists which updates itself every day and prevents from phishing, scam and malicious sites. It can also protect people from Fake educational institutions flagged by UGC and AICTE by scanning the page content for mention of these names. User Input text can be analysed by the extension to categorise them as Potential AI Generated or Human along with Trust Score. This multi-feature extension can run current URL against blocklist to flag those domains in real time providing threat security.

REFERENCES

[1] Elad, B. (2025b, June 24). Phishing statistics by demographic, healthcare, industry and country (2025). Sci-Tech Today. https://www.sci-tech-today.com
[2] Wikipedia contributors. (2025, September 16). Digital marketing. Wikipedia. https://en.wikipedia.org/wiki/Digital_marketing
[3] Sarkar, S. (2020, May 16). A brief history of online education. Adamas University. https://adamasuniversity.ac.in
[4] Tikkanen, & Amy. (2016, January 27). University of Phoenix | History, Overview, & Facts. Encyclopedia Britannica. https://www.britannica.com
[5] Statista. (2025, July 10). Literacy rate in India 1981-2023, by gender. https://www.statista.com/statistics
[6] Correspondent, S. (2021, May 28). Popular YouTuber held for quackery in Kallakurichi. The Hindu. https://www.thehindu.com
[7] India, T. O. (2025, August 8). Doctor duped in online phishing scam, loses Rs 82k. The Times of India. https://timesofindia.indiatimes.com
[8] Kumari, A. (2025, July 10). Assets Worth Over Rs 200 Crore Attached In Manav Bharti University Fake Degree Scam. Www.ndtv.com; NDTV. https://www.ndtv.com
[9] Kannan, R., Kumar, S., & Mehta, P. (2025). Automated trust evaluation systems for digital platforms. International Journal of Information Security and Digital Trust, 12(2), 45–58.
[10] Gupta, A., Sharma, N., & Verma, R. (2025). AI detection in academia: How Indian universities can safeguard academic integrity. Journal of Educational Technology and Policy, 18(1), 22–34.
[11] Olan, F., Jayawickrama, U., Arakpogun, E., & Dwivedi, Y. K. (2024). Fake news on social media: The impact on society. Information Systems Frontiers, 26(3), 567–582. https://doi.org/10.1007/s10796-023-10345-2
[12] Lebopa, K., Mokoena, T., & Dlamini, S. (2024). Information verification system to prevent academic fraud by employees. Journal of Information Assurance and Security, 19(1), 91–103.
[13] Alnabhan, M., & Branco, P. (2024). Fake news detection using deep learning: A systematic literature review. IEEE Access, 12, 134210–134225. https://doi.org/10.1109/ACCESS.2024.3389123
[14] D'Souza, R., & Mehra, S. (2024). Integrating public datasets for online verification and fraud prevention. Journal of Cybersecurity and Digital Governance, 7(2), 101–115.
[15] Zadrozny, B. (2020, February 21). She wanted a "freebirth" with no doctors. Online groups convinced her it would be OK. NBC News. https://www.nbcnews.com