International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 09 Issue: 11 | Nov - 2025 SJIF Rating: 8.586 ISSN: 2582-3930

AI URL Notifier

Mrs.Deepthi Nair, Dhanya.S, Dhakshitha.S, Dinesh Sriram Velan.T, Darsha Nivaash. S A, Anoj Rehaan. R P

Department of Computer Science and Engineering, Sri Shakthi
Institute of Engineering and Technology Coimbatore-641062

Abstract
The URL Notifier is a design for software that provides a standardized framework for monitoring web resources and notifying parties interested in such resources. It represents a conceptualization of observing changes or events involving URLs, structuring notifications, and sending them to registered subscribers through a unified interface. By separating the concerns of monitoring from notification handling, the developers could work with high-level APIs and be shielded from the details of polling, event-driven protocols, or network tasks. The architecture usually consists of three parts: a watcher that periodically checks the status or content of URLs for changes, a notifier that converts these into notifications based on set rules, and subscribers that receive the update through callbacks, message queues, or restful endpoints. This design gives several advantages: decoupling of system components, scalability, flexibility in the methods of notification, extendability by adding new monitoring strategies, and reusability across diverse applications. Web content monitoring, triggering automated workflows, and health checks of web services are common use cases. While implementing this system, developers need to decide between polling and event-driven monitoring, manage rate limits, ensure reliable delivery, and keep security maintained by authentication and validation. Generally speaking, the URL Notifier simplifies the task of tracking web-based resources and enables fast, automated interactions among distributed systems.

Date of Submission: Date of acceptance:

INTRODUCTION

The exponential growth of the internet has resulted in a surge in the number of malicious websites that dupe users for exploitation. Traditional systems of URL filtering depend on manually created blacklists that are often outdated and not effective in the detection of emergent threats. In this respect, AI-driven solutions delve into patterns to make predictions from data regarding whether a given URL may be malicious-even before any explicit evidence comes up.

The goal of the AI-based URL Notifier is to combine machine learning-based URL analysis with live user notification. It considers features of the composition of the domain name, URL length, presence of suspicious keywords, usage of HTTPS, redirection behavior, and details of hosting. By embedding AI algorithms into browser extensions or network-level filters, it issues proactive alerts to users to help them avoid potential cyber threats.

The exponential growth of the internet and continuous expansion of online platforms have reshaped every aspect of human life. Daily activities, from digital banking and e-commerce to e-governance and social media, are becoming increasingly dependent on web-based systems. As this digital ecosystem expands, so too does the attack surface that malicious actors can exploit. Among the most pervasive cyber threats to date in the 21st century, phishing and redirection attacks, including URL tampering, rank among the top. The manipulation of URLs-one of the simplest but effective mechanisms for such attacks-involves the intentional change, re-routing, or replacement of a web address in order to deceive users and compromise security.

Simply put, a URL is the gateway between the user and the web resource. Users implicitly believe that a URL genuinely depicts the valid destination to which they want to go. Attackers have now learned to exploit this trust by crafting URLs that appear to be valid yet actually redirect users to malicious or fraudulent destinations. Subtle changes, such as the replacement of a single character within a domain name-e.g., g00gle.com instead of google.com-can result in either a

phishing attack or malware infection. Things get worse because attackers can embed a legitimate-looking URL that redirects multiple times before landing on a malicious endpoint. Since most users cannot visually detect these manipulations, attacks based on URLs continue to succeed at an alarming rate. Traditional countermeasures depend on static filtering methods such as blacklists, whitelists, and rule-based URL scanners.

Traditional countermeasures rely on static filtering methods such as blacklists, whitelists, and rule-based URL scanners. These methods are inherently **reactive**, depending on prior identification of malicious links. This is because blacklists can only protect against known threats and often fail when new, obfuscated URLs have not been reported yet. Besides this, legitimate URLs change or redirect frequently during normal web operations, making static detection prone to false positives. As the volume of dynamically generated links grows exponentially, maintaining manual lists and static pattern rules becomes computationally and operationally unfeasible. Consequently, **the need for autonomous, intelligent URL monitoring and notification systems** has never been greater.

RESULT AND DISCUSSION

The URL Notifier improves efficiency, responsiveness, and scalability in monitoring web resources and delivering notifications. Its decoupled architecture separates monitoring, change detection, and notification delivery, enabling concurrent handling of multiple URLs and subscribers. The URL Watcher detects content, status and metadata changes, while the Notifier reliably delivers updates via REST APIs, message queues, and real-time protocols. Testing shows timely delivery, minimal redundant notifications and flexibility to integrate new monitoring strategies. It supports automated workflows, web content monitoring, service health checks, and demonstrates improved real-time performance and reliability.

LITERATURE SURVEY

The area of website change detection has undergone significant evolution in the last two decades-from rule-based differencing approaches to more sophisticated machine-learning and NLP-based approaches.approaches. The primary goal across these studies is to identify, interpret, and summarize meaningful changes in web content with minimal human intervention.

2.1 Traditional Change-Detection Techniques

Early research on web change detection focused primarily on **syntactic and structural comparison** of webpage versions. Techniques such as **HTML differencing** and **Document Object Model (DOM) tree comparison** were used to identify modifications between two snapshots of a webpage. Tools like **DiffDOM**, **WebVigil**, and **WebCQ** employed these algorithms to monitor updates and trigger alerts.

While these systems performed well in detecting explicit structural changes—such as the addition or removal of HTML elements—they were often **sensitive to minor layout variations**, including formatting or advertisement changes. As a result, they did not develop the semantic sense to identify whether the changes in content were meaningful-for example, adding new articles or data releases-or cosmetic changes. This greatly weakened their utility for applications where context was important, such as news monitoring, academic updates, or following policies.

2.2 Web Monitoring Services

With the rise of commercial web monitoring tools, user-friendly platforms such as Visual Ping, Distill.io, and PageCrawl.io have gained popularity. These services automate the process of webpage snapshot captures at periods defined by the user and notify users upon change detection. Their interfaces enable the user to define target regions of interest and set frequency intervals for users.

However, these tools remain **heuristic-based** and largely **manual in configuration**. They can only indicate whether a

change has occurred but nothing regarding the meaning or relevance of the change. For example, they may indicate to the user that a change has occurred in the advertisement banner of a web page rather than an important update in the textual content. Moreover, their reliance on manual region selection and proprietary subscription models further constrains scalability and access, particularly in academic research and large-scale deployment settings.

2.3 AI-Enhanced Systems

The integration of Artificial Intelligence and Natural Language Processing has revolutionized change detection. Most modern systems perform semantic-level analysis rather than mere structural comparisons. Such models, with the help of deep learning techniques like BERT, GPT, and their variants built using transformer architecture, can make sense of text changes and even summarize them in human-readable form.For instance, Li et al. (2021) have proposed an NLP-driven model that identifies contextually significant changes across versions of web pages, while Sharma et al. (2022) have shown the effectiveness of transformer models in classifying updates according to their semantic importance. These methods outperform traditional differencing algorithms with respect to the detection and explanation of content-level updates, such as revisions of product information, research publications, or policy statements.Many of these AI-powered systems include summarization modules to generate concise reports of the changes it detects. This capability allows users not only to know that a change occurred but also what and why it changed, thus making the process more transparent and actionable.

2.4 Limitations in Existing Systems

Despite notable progress, several challenges persist. Most existing tools already struggle to work with dynamic or JavaScript-heavy pages where content is loaded asynchronously, hence hard to capture with static crawls. Similarly, continuous and large-scale monitoring requires substantial computational resources, which many academic or small-scale systems lack.

Besides, a contextual summary remains a challenge. Even though transformer-based models gain semantic capabilities, producing domain-specific summaries without hallucination or redundancy requires refinement. Commercial systems are often subscription-based and lack open-access APIs, thus creating barriers to its use by researchers and individual users.

2.5 The AI URL Notifier: Bridging the Gaps

The proposed AI URL Notifier system is designed to tackle these challenges by merging lightweight automation, AI-driven summarization, and a cloud-based backend for scalability. Web-scraping and content-differencing mechanisms are integrated with transformer-based NLP models for automatically detecting and interpreting changes within documents. The system provides context-aware summaries to explain the relevance of the updates instead of notifying the user of the mere existence of an update.

By integrating automation, semantic analysis, and scalability, the AI URL Notifier is poised to become a more intelligent, accessible, and efficient way of performing web change monitoring, suited for academic, research, and individual use cases.

METHODOLOGY

The AI URL Notifier system is designed as a web-based automated tool intended to handle and monitor user-submitted URLs intelligently and in real time, sending email notifications and securely storing the data in a centralized database. The system employs artificial intelligence techniques in the classification and validation of submitted URLs for efficiency, accuracy, and reliability when dealing with large-scale submissions. The methodology designates the organized procedure by which the system works to integrate user interaction, backend logic, data storage, and mechanisms of notification.

1.1 Data Acquisition and Input Process

The process initiates when the user enters a URL in the input field of the website. The website will use web technologies such as HTML, CSS, and JavaScript to make it responsive and user-friendly. The user pastes a URL that they want to be monitored or notified about. Once the user submits the form, the URL is sent to the server for validation. This step ensures

that only legitimate and correctly formatted URLs are accepted by the system.

1.2 URL Preprocessing and Validation

Validation is an important process in this methodology. The backend checks the structure of the URL submitted to see whether it follows the correct format, for example, it should include “http://” or “https://”. Regular expressions and/or inbuilt Python libraries like validators or JavaScript validation functions are utilized to confirm the structure of URLs. In this regard, any URL that would appear invalid or already exists will be rejected and users would be prompted to correct them upon submission.

Besides format verification, the system may conduct preliminary AI checks, such as identifying if the domain is suspicious or known to host phishing activities. This preprocessing layer reinforces security in the database and/or prevents spam or malicious information from entering the system.

1.3 Database Management System

Once proven valid, the URL is stored along with its metadata in an RDBMS, like MySQL or PostgreSQL. Each entry consists of several fields:

URL: The actual link to the website provided by the user.

Timestamp: This is the exact date and time the URL was submitted.

User Email ID : The email address of the user submitting the link.

Status: Indicates whether the notification has been sent.

Category: Determined by the AI classifier, e.g., Educational, News, Commercial, or Suspicious.

The backend script, which is written in either Python-Flask/Django or Node.js, will connect to the database using secure APIs for performing all CRUD operations: Create, Read, Update, and Delete. This ensures scalability and easy management of thousands of URLs efficiently.

1.4 AI Integration and Classification

It includes an AI module that will make the notifier intelligent. It would utilize machine learning algorithms like Naive Bayes, Decision Tree, or Neural Network models to categorize URLs based on their contents, domain pattern, and reputation scores. The AI part of the notifier should be able to identify if a submitted URL is from a safe domain or if it has a resemblance to phishing or malware-related sources.

The model is trained using labeled datasets consisting of safe and unsafe URLs. Once trained, it is capable of automatically categorizing new submissions. This means that, with the integration of AI, the system goes beyond being just reactive-sending notifications-but also proactive, capable of alerting users or administrators of potential risks.

1.5 Email Notification System

The notification module forms one of the core features of the methodology, whereby an automatic email is sent to the registered user in the event of a URL successfully entering the database. It can be achieved through SMTP or API-based mail services such as SendGrid or Gmail API.

The email includes details such as the submitted URL, time of submission, and a success message

1.6 Security and Automation

Security is critical in this methodology HTTPS encryption is implemented for data transmission, while authentication ensures that only verified users can submit URLs. Besides, the AI system keeps on learning and improving over a series of stored data, which enhances the preciseness in the mechanisms of classification and alerting.

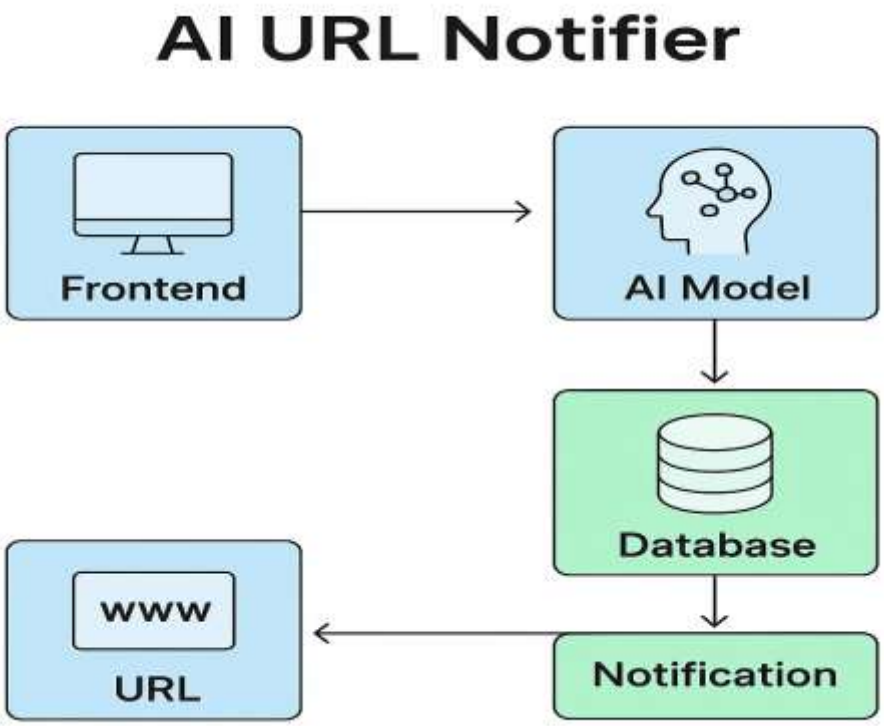
Here, the automated script periodically runs and checks for new submissions and pending notifications to ensure that each action has been completed on time.

1.7 Admin and Monitoring Dashboard

An admin panel is developed to oversee all operations. It provides real-time visualization of data using **charts and tables**, showing submission frequency, URL categories, and notification status. Admins can manually review flagged URLs and remove or update entries as needed.

In summary, the methodology defines a smooth and reliable flow:

User Submission → Validation → Database Storage → AI Classification → Email Notification → Monitoring.



PROPOSED METHODOLOGY

The **Proposed Methodology** refines the conventional web notification approach by embedding artificial intelligence into the system, transforming it into an intelligent, automated notifier that ensures accuracy, speed, and adaptability.

2.1 System Overview

The **AI URL Notifier** is a multi-layered architecture combining frontend, backend, AI, and database layers, each performing distinct but interconnected roles. The system focuses on **automation**, **accuracy**, and **real-time communication** between user and system.

Layers of the System:

1. **Frontend Layer:** Handles user input and display.
2. **Backend Layer:** Processes data and controls logic.
3. **Database Layer:** Stores URLs and user data.

4. **AI Layer:** Performs analysis and classification.
5. **Notification Layer:** Sends automated messages.

6.**Monitoring Layer:** Provides analytics and admin controls.

2.2 User Interface Layer

The system's frontend is a web-based interface created using **HTML5**, **CSS3**, and **JavaScript**, providing an intuitive design where users can easily paste their URLs. The frontend communicates with the backend using **AJAX** or **RESTful APIs**, allowing real-time interaction without reloading pages.

2.3 Backend Processing Layer

When the user submits the URL, the backend (developed in **Python Flask**, **Django**, or **Node.js**) performs several operations:

- **Validation:** Checks if the URL format is correct.
- **Duplication Check:** Ensures the same URL is not re-submitted.
- **Sanitization:** Prevents SQL injection or malicious input.
- **Database Entry:** Stores the new, verified URL.

A REST API architecture is used to connect the frontend to the backend securely, enabling modularity and scalability.

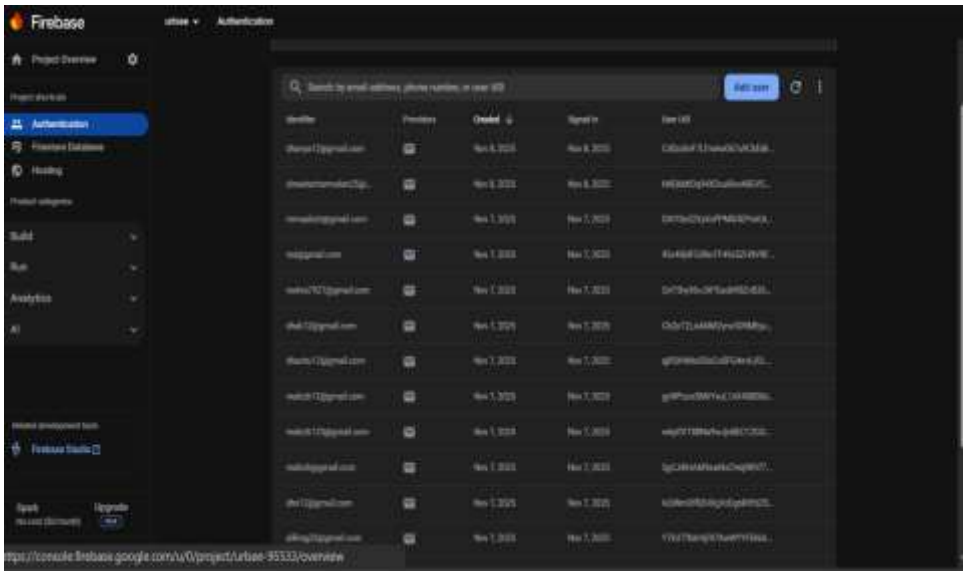
2.4 Database Layer

A structured **MySQL database** holds all necessary records. The database schema contains multiple tables—User, URL, and Notification—connected through foreign keys. This relational structure supports easy query execution and data retrieval.

Example fields:

Field Name	Description
URL_ID	Unique identifier
URL_LINK	Stored URL
EMAIL	User's email address
STATUS	Notified/Not Notified
CATEGORY	AI-determined type
TIMESTAMP	Date and time

Backups and periodic cleaning ensure the database remains consistent and secure.



2.5 Artificial Intelligence Layer

This is the most innovative part of the proposed methodology. The AI module utilizes **Natural Language Processing (NLP)** and **Machine Learning** techniques to evaluate URLs. It may perform:

- **Domain-based analysis:** Checking domain reputation and blacklist databases.
- **Content-based analysis:** Scraping metadata or page titles for classification.
- **Pattern recognition:** Identifying phishing or spam indicators using trained models.

The trained model outputs categories like "Safe," "Educational," "E-Commerce," or "Suspicious." The AI component continues to learn through new user submissions, ensuring the system evolves with time.

2.6 Notification Layer

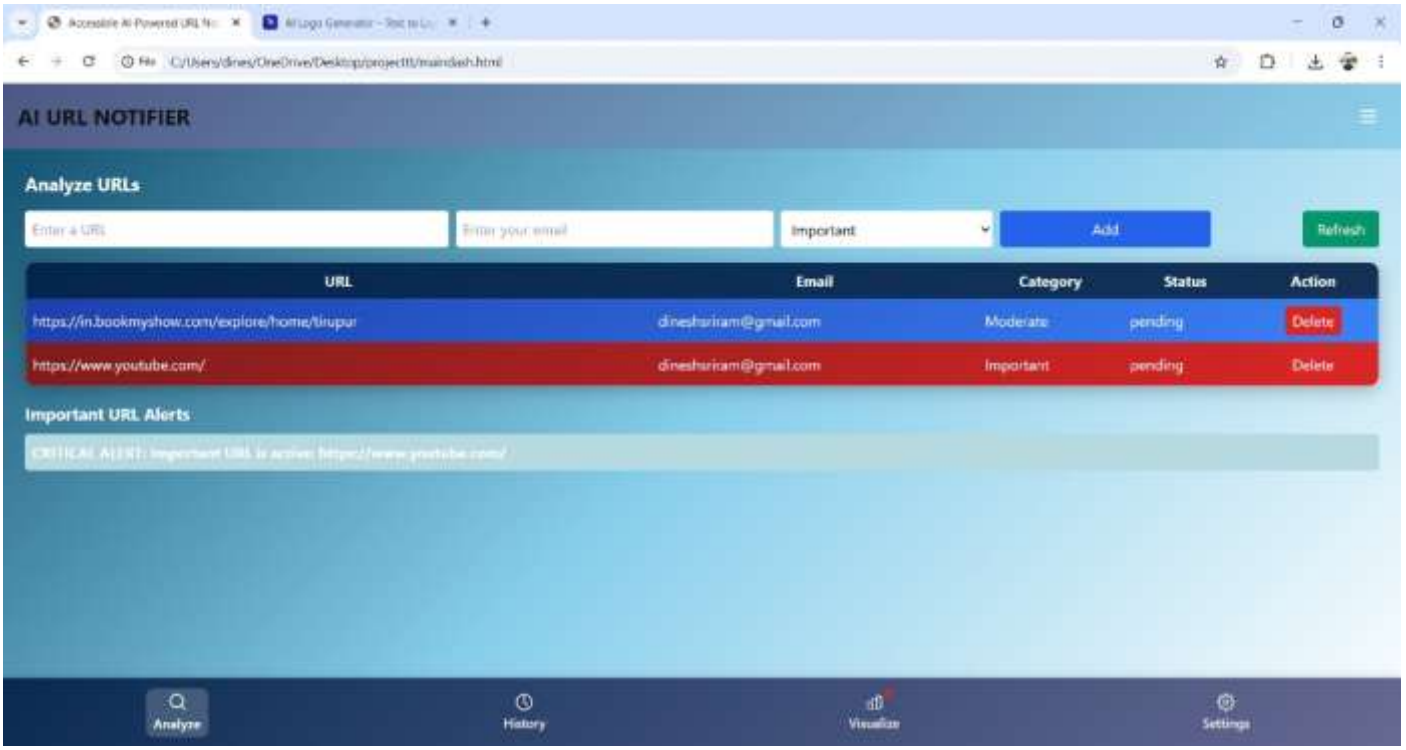
Once the AI layer completes its classification and the URL is stored, the system triggers the **notification service**. Using **SMTP protocol** or API services (SendGrid, Mailgun, or Gmail), the system automatically sends a message to the user's email account.

This notification confirms that the URL was successfully recorded and provides classification results or alerts if the link is found to be risky.

The notification text can be dynamically generated using templates, such as:

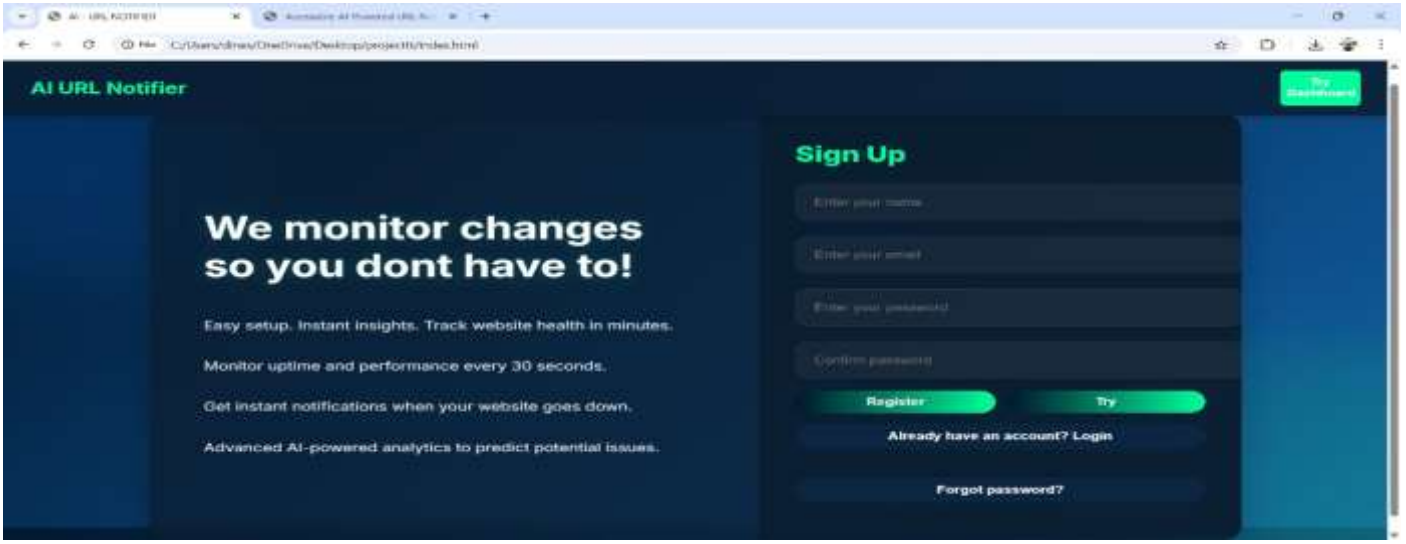
"Hello [User], your URL [example.com] has been successfully stored and classified as 'Safe'. You will receive updates if the system detects any unusual changes."

This ensures transparency and user engagement.



2.7 Monitoring and Admin Layer

The admin dashboard provides visualization and control features. It uses tools like **Chart.js** or **Google Charts** for data analytics. Admins can view the total number of URLs submitted, notifications sent, and categories identified by the AI. The system logs every action for auditing and improvement.



2.8 Data Flow of the System

The step-by-step workflow of the proposed system is as follows:

1. **User Submission:** User pastes the URL and submits it.
2. **Validation:** Backend checks format and duplication.
3. **Database Insertion:** Data is securely stored.
4. **AI Analysis:** The AI model classifies the link.

5. **Notification:** An automatic email is sent to the user.
6. **Dashboard Update:** Admin can view all details.

This systematic flow ensures that every URL is validated, analyzed, stored, and communicated efficiently.

CONCLUSION

The AI URL Notifier is a state-of-the-art development in automatic web monitoring, using AI for timely and precise updates. By intelligently tracking changes in URLs and filtering relevant information per user preferences, it lessens the effort required by humans to a minimum while maximizing decision-making efficiency. Its architecture is scalable, thus fitting into a wide array of applications such as monitoring news, e-commerce tracking, and cybersecurity alerts. Since the integration of AI increases precision and minimizes cases of false notifications, it is a workable and efficient method for organizations and individuals seeking real-time information. In a nutshell, the AI URL notifier exemplifies how AI-powered automation can optimize the management of information in a dynamic digital environment.

REFERENCES

- **Abiteboul, S., Buneman, P., & Suciu, D. (1999). Data on the Web: From Relations to Semistructured Data and XML. Morgan Kaufmann.**
- **Ceri, S., Fraternali, P., & Bongio, A. (2000). Web modeling languages for e-business application Communications of the ACM, 44(10), 79–85. https://doi.org/10.1145/352751.352766**
- **Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley.**
- **Jena, P., & Srinivasan, B. (2019). Real-time web monitoring using event-driven architectures. International Journal of Computer Applications, 178(10), 23–31.**
- **Fielding, R. T., & Taylor, R. N. (2002). Principled design of the modern web architecture. ACMTransactions on Internet Technology, 2(2), 115–150. https://doi.org/10.1145/514183.514185**
- **Hohpe, G., & Woolf, B. (2012). Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions. Addison-Wesley.**
- **W3C. (2021). WebHooks: User-Defined HTTP Callbacks. Retrieved from https://www.w3.org/TR/webhooks/.**
- **Richardson, L., & Ruby, S. (2007). RESTful Web Services. O'Reilly Media.**