Phish Guard: A Phishing Attack Detection and Prevention Tool

Submitted By

SHAHZADA ISALM

221-35-997

Supervised By

DR. MARZIA AHMED

Assistant Professor

This project report has been submitted in fulfilment of the requirements for the degree of **Bachelor of Science in Software Engineering**

DAFFODIL INTERNATIONAL UNIVERSITY

DECLARATION OF THESIS AND COPYRIGHT

Author's Full Name	:	Shahzada Islam
Date of Birth	:	17-1-2002
Title	:	Phish Guard: A Phishing Attack Detection and Prevention Tool
Academic Session	:	2022-2025

I declare that this thesis is classified as:

□	CONFIDENTIAL	(Contains confidential information under the Official Secret Act 1997)*
□	RESTRICTED	(Contains restricted information as specified by the organization where research was done)*
□	OPEN ACCESS	I agree that my project to be published as online open access (Full Text)

I acknowledge that Daffodil International University reserves the following rights:

1. The Project is the Property of Daffodil International University. The Library of
2. Daffodil International University has the right to make copies of the Project for the purpose of research only. The Library of Daffodil International University has the
3. right to make copies of the Project for academic exchange.

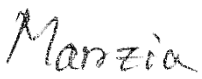
Certified by:

(Student's Signature)

221-35-997

30.11.2025

Student ID Date:



(Supervisor's Signature)

Dr. Marzia Ahmed

30.11.2025

Name of Supervisor Date:

NOTE: * If the Project is CONFIDENTIAL or RESTRICTED, please attach a thesis declaration letter.

APPROVAL

This thesis titled on "**PhishGuard: A Phishing Attack Detection and Prevention Tool**", submitted by **Shahzada islam (ID: 21-35-997)** to the Department of Software Engineering, Daffodil International University has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of Bachelor of Science in Software Engineering and approval as to its style and contents.

BOARD OF EXAMINERS

Chairman

Dr. Fazla Ealhe
Assistant Professor & Associate Head
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University

Internal Examiner 1

Dr. Marzia Ahmed
Assistant Professor
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University

Internal Examiner 2

Dr. Shabnom Mustary
Assistant Professor
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University

Internal Examiner 3

Md. Rajib Mia
Lecturer (Senior Scale)
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University

External Examiner

Mohammod Abul Kashem, PhD
Professor
Department of Computer Science and Engineering
DUET, Bangladesh

PROJECT DECLARATION LETTER (OPTIONAL)

Librarian,
Daffodil International University,
Daffodil Smart City,
Ashulia.Dhaka,Bangladesh

Dear Sir,

CLASSIFICATION OF Project AS RESTRICTED

Please be informed that the following project is classified as RESTRICTED for a period of three (3) years from the date of this letter. The reasons for this classification are as listed below.

Author's Name
Project Title

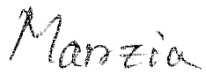
Reasons (i)

(ii)

(iii)

Thank you.

Yours faithfully,



_____
(Supervisor's Signature)

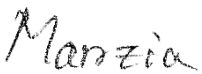
Date:

Stamp:

Note: This letter should be written by the supervisor and addressed to the Librarian, *Daffodil International University* with its copy attached to the thesis.

SUPERVISOR'S DECLARATION

I/We* hereby declare that I/We* have checked this project and in my/our* opinion, this project is adequate in terms of scope and quality for the award of the degree of Bachelor of Science.



_____

(Supervisor's Signature)

Full Name : Dr. Marzia Ahmed

Position : ASSISTANT PROFESSOR

Date : 27 Novermber 2025

STUDENT'S DECLARATION

I hereby declare that the work in this project is based on my original work except for quotations and citations which have been duly acknowledged. I also declare that it has not been previously or concurrently submitted for any other degree at Daffodil International University or any other institution.

_____

(Student's Signature)

Full Name : Shahzada Isalm

ID Number : 221-35-997

Date : 27 Novermber 2025

Phish Guard: A Phishing Attack Detection and Prevention Tool

SHAHZADA ISLAM

Project submitted in fulfillment of the requirements
for the award of the degree of
Bachelor of Science

Department of Software Engineering

DAFFODIL INTERNATIONAL UNIVERSITY

NOVEMBER, 2025

ACKNOWLEDGEMENTS

Let's begin with the name of Allah the source of all knowledge and wisdom.

Acknowledgement First My all respect and thanks goes my SUPERVISOR Dr. Marzia Ahmed Assistant Professor Department of Software Engineering Daffodil International University. Her direction on a software architecture and the software performance tuning was crucial towards carving the Banglar Shahzada Phishing Guard. Her push to make something that's not also walled off, hardware wise or otherwise caused me to also push this toward older devices and 'budget' laptops.

Thanks to Software Engineering Department (for facilities provided) And the advice of the professors in term of web security, and efficient coding methods have been taken in directly in order to make this software work on older computer systems.

Last but not least am most grateful to my parents, for their countless sacrifices, unconditional love and encouragements during the entire process of my study. They never allowed themselves to use their body pain as a deterrent for my education while they faced severe health problems, my father had heart prpblems and your mother was suffering from diabetes. They suffered a lot, even they sold their land and possessions for me to study. This venture is my small little homage to their sweat, blood and undying belief in me.

DEDICATION

I therefore declare that I have done this project under the oversight of Dr. Marzia Ahmed Assistant Professor, Department of Software Engineering, Daffodil International University. Also declare that neither entire record nor any portion of this record has been submitted somewhere else for my degree.

ABSTRACT

With high rate of internet penetration in Bangladesh, all the people are increasing their presence online and so is the case with phishing attacks threatening day to day for everybody, specially ones who do not belong to any sort of technical background. Many of the available tools are resource demanding and require online connection, thus rendering them too heavy to implement on legacy devices. This work has addressed this missing by developing an illustrated anti-phishing victim-assistance system:Banglar Shahzada Phishing Guard as a light-weight, bilingual Bangla and English ideal of privacy focused web application. Instead, it is written on top of a self-sufficient node. js backend with a deterministic In-Memory Hash Map Indexing algorithm. Such design decision enables the system to be able to perform a real-time URL lookup with O(1) complexity against PhishTank without being affected by any API latency. Also I've built the client side Shadow AI on my own, it hunts multi-level threats with Shannon Entropy logic and Homoglyph attacks. Performance However, preliminary testing shows that the Application responds within milliseconds to property transactions, so even users with low-end PCs Intel i3 with 4GB Ram will feel secure.

TABLE OF CONTENT

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

O(1)	Constant Time Complexity (Used for Hash Map lookup)
H(X)	Shannon Entropy Formula (Used in Shadow AI)

LIST OF ABBREVIATIONS

URL	Uniform Resource Locator Artificial
AI	Artificial Intelligence
UI	User Interface
API	Application Programming Interface
JSON	JavaScript Object Notation
PDF	Portable Document Format
SME	Small and Medium-sized Enterprises
MFS	Mobile Financial Services
SHA	Secure Hash Algorithm

LIST OF APPENDICES

CHAPTER 1 INTRODUCTION

1.1 Background

Bangladesh has experienced a large growth in internet users over the past decade millions of people now use digital platforms for communication, mobile banking MFS, and e-commerce comments. But as digital growth is accelerating, it also draws in bad actors looking to exploit vulnerable non-specialist users with increasingly sophisticated phishing campaigns. Although there is enterprise level security software that can satisfy such demands , they are generally costly and resource intensive , also demanding added high-quality hardware .

The work Banglar Shahzada Phishing Guard is a focused alternative with a lightweight privacy based solution. No, the new AV is not scanning cloud databases in the background like antivirus software does traditionally – This system is supposed to run on older hardware it was designed with legacy hardware like Windows 10 running on old laptops in mind without taking away from their processing power .

The software is designed to help in providing users with the capacity to detect threats in Bengali language and democratize cybersecurity capacity.

1.1.1 Context and Relevance

Concerning the software engineering field, current security tools are growing fatter and fatter, with heavy dependences and the need to connect to the internet all the time. This isolates a huge section of the population who either have older devices or have restricted access to the internet. The project deals with this relevance by developing an independent phishing detector system that can detect phishing information efficiently with minimal resource demands through the optimization of data structures.

1.1.2 Problem Identification

By an analysis of existing solutions, we observed several significant issues:

(I) Resource Hungry : Many anti-phishing systems rely on heavyweight SQL databases or are so RAM hungry they cannot be used effectively on budget laptop Like core i3, 4gb of ram .

(II) Dependency Hell : Many Node. js that depends on native module i.e., better sqlite3 or node-gyp , which involves complicated C++ build tool, so most users would have lots of annoying problems for installation.

(III) Privacy Concerns : Existing tools send user data URLs and Email to cloud APIs for verification, this may lead to privacy leakage

(IV) Language Issue: There exist no security instruments for alerts and reports in Bangla.

1.1.3 Purpose and Justification

The first incentive that has led me to the need keep away of third party systems is that the need and one of the first targets of this project is to create a Zero Dependency phishing detection system that would operate offline.

Justification:

(I) erformance: It is based on our own In Memory Hash Map Indexing algorithm whose complexity is O(1) and thus provides query results with zero disk I/O latency.

(II) Accessibility: It is meant to be compatible with older hardware bearing in mind that Security is a right, not a privilege.

(III) Privacy No personal data will be transmitted to any third-party API, no information will escape your computer > 100% PRIVACY.

1.1.4 Scope

The project considers a multi-tiered detection model to detect phishing threat across four attack vectors :

(I) URL Integrity Check : Checks the local PhishTank database on looks URLs that may be fake and confirms their authenticity. It does not need to rely on slow externally available APIs, but Exact-Match and Domain-Based algorithms find trust almost instantly

(II) Digital content forensics: The main body of the email is scored using a token system. This engine is a pattern trained engine and acquires patterns based on predefined language databases like Enron, Nazario corpus of social engineering keyword and deceptive sentence patterns.

(III) Shadow AI (Advanced Heuristics) : Special clientside threat detector based on anomaly. It uses Shannon Entropy to identify randomly built domain names and identifies Homoglyph characters that are utilized in brand spoofing.

(IV) Binary Inspection: The tool does a thorough deep-scan job of the suspicious files (PDF / DOCX) by verifying the presence of hidden executable payloads even before it hits your system.

This version emphasize on offline detection logic in order to enforce your data privacy strictly!

1.2 Project Planning and Initiation

Feasibility Study (Step-by-Step)

Introductory Note: Before jumping to the core development, I have done a comprehensive feasibility study to make sure that the building of a "Zero-Dependency" security tool was actually feasible. The main challenge was to design a system which operates smoothly on the legacy hardware without the system crashing, and all sensitive data is strictly offline. This study breaks down how I analyzed the technical, financial and operational aspects to prove that a standalone Node js architecture is the most effective solution for user in Bangladesh.

Phase 1 Preliminary Analysis & Project Scope Defining:

Initial Analysis & Project Scope Definition We got that project started to explore whether a security solution other than a heavy database server was practicable to implement. We definitely were conscious in choosing to do the business of using the JSON flat-files in-memory (RAM). over SQL tables. This decision was taken so as to avoid "the installation overhead of" MySQL or MongoDB" and keep the software "portable", which means that it well, even to run on older systems.

Phase 2 Market Feasibility Analysis (or Market Research): Market Feasibility Research (Market Analysis) There is a great gap in the local cybersecurity market which was evident from our study With growth in the number phishing attempts made against Mobile Financial Services (MFS), there are only few devices that can be used by a general user for detection of such malware and those tools found are mostly focus on

English language. We noticed a high need for such an simple to use tool with Bangla support, which would help bridge this gap for non technical users.

Phase 3: Technical Feasibility Analysis:

Technical Feasibility Study Architecture - for the architecture we choose Node. js and the Express because of their Non Blocking I/O which Speeds things up. On the frontend, we have used old fashion Vanilla JavaScript instead of heavy frameworks like React with our own dying Shahzada UI System (a proprietary lightweight CSS architecture). This ensures that the browser will render the page immediately without having to wait until all the images are downloaded.

Phase 4 Financial Feasibility Analysis:

Technical Feasibility Study Architecture - for the architecture we select Node. js and the Express because of their Non Blocking I/O which Speeds things up. On the frontend, we used old fashion Vanilla Java Script instead of heavweight frameworks like React with our own dying Shahzada UI System (a proprietary lightweight CSS architecture). This will ensure that the browser will immediately render the page without having to wait until all images have been downloaded.

1.3 Target User Profile and Tentative Elicitation Process

1.3.1 Target User

My primary audience is Bangladeshi general users, they are common people and surf internet but afraid of complicated technical terminology. I designed this program for users that still use older laptops or desktops because they could not deal with a modern heavier antivirus These are the users that fall victim to the fraudulent SMS links, phishing emails or those ostensible benign(or innocuous) pdfs with malicious code therein Because they can't manually examine a file's binary signature in order to determine if it's malicious, in this app's case it has now stepped in to do exactly that. that heavy lifting for them -- and as fast as possible.

Apart from personal use, this app may be a big solution for the local startups and small companies of Bangladesh. Smaller offices can't and won't pay for the great enterprise

security bundles nor do they have a need for high end hardware. This lightweight solution gives them a cheap layer of security, protecting their internal communications and confidential data on the office PCs which they have already in place, but don't want to make the investment in upgrading

1.3.2 User profile

Table 1.1: User Profile (General Users & SME Owners)

User Class	General Non-Technical User & Owner of SME
Type of user	Everyday internet users using legacy hardware PC, and Small Business Owners striving to find cost effective security
Age range	18 – 60 Years
Frequency of use	Whenever a suspicious link, email, or file is received or Text messege .
Mandatory	Optional: It is a self-help tool that is used voluntarily for personal safety.
Computer experience	Basic: They can go on Facebook, check their emails or type documents, but lack proper knowledge of pishing and manually identifying the hidden malware.
Education	Varied: Varies from students at the school level to graduates. No computer science degree required in order to use this tool
goal	To instantaneously know if a link or file is it Safe ,Safe or Risk without having to install heavy antivirus software to slow down their old PC.
Language skills	Native Bangla speaking people who feel safer in front of the alerts in their mother tongue. Since they also possess basic English skills, the Bilingual Interface gives them the freedom just to choose what would work best for them.
Number of users	Potentially the mass population of Bangladesh using low-end devices.
Training	None Required : The interface is designed in an intuitive like Google Search, therefore no manual and training is required.

Others system use	None Required : The interface is designed in an intuitive like Google Search, therefore no manual and training is required.
Way of working	Very simple, they just copy- and paste one link

1.3.3 Elicitation Process

I did not employ the same boring web based surveys that people are likely to simply trash, instead I used a more hands-on approach. I would have liked to exhibit real panic, the manner in which individuals experience panic when they get a suspicious message. Then I made my requirements out of real observation and mere chatting with the people around. This is the basis on which the trip occurred in three simple steps: Direct Observation (Watching the Struggle) I began directly with my own family and friends using some older laptops. What I found alarming was this: When an alarming SMS such as bKash Lottery Winner came or an email with a strange attachment came, they stared at it, in dismay. They could not even use such tools as VirusTotal since the interface in the English language was too complicated to use. That was the time when I had a revelation - My tool has to communicate with them. Informal Interviews (The What If) Scenario I interviewed college friends and the owners of local stores without being technology gurus. I posed them a question that was, well, a deceptive one, which is: What would you do to verify whether a link is safe or not, should you receive a shady link? The Truthful Response : About 80 percent of the said troubling answer as simply would wing it or push back due to verification due hassle. Their Wish: They simply told me, that they did not need any graphs or codes. Give us either remind of 1 simple Red or Green signal in bangla language so we know already.

Refining the Requirements Hearing about their problems allowed me to nail down which features that matter most in this project: Zero livedb: They should not need to install a heavy db and drink the old pcs because there is no other way. Bilingual & Plain: We need to see result in Bangla for anybody - a student,shopkeeper or anybody else to feel secure.

1.4 Project Block Diagram

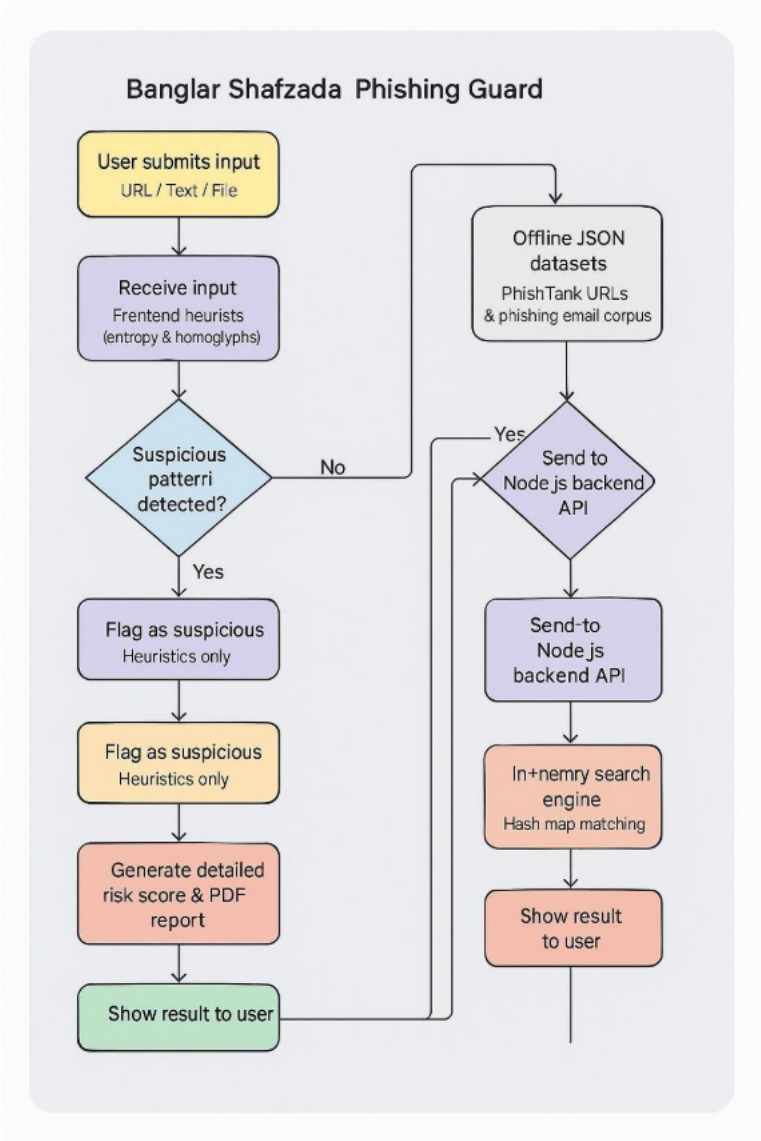


Figure 1.4: System Block Diagram

1.5 System Requirements

In order to be true to our core mission to provide 'Security for All,' the system is created to run efficiently on the legacy hardware that may not be able to handle using the most up-to-date, resource-heavy antivirus software; The requirements below are a result of our successful testing on the older devices including 9 year old Intel i3 laptop .

1.5.1 Hardware Requirements

Since the application does not involve heavy database queries and is In-Memory based Indexing: - The hardware requirements are low:

Processor: 5th gen Intel Core i3 Equivalent AMD processor. The single-threaded nature of Node.js is what makes it the ability to function extremely well even on the older CPUs with dual cores.

RAM (Memory): Minimum 4 GB. While the system is able to load big datasets (PhishTank & Email Corpus) into memory in order to perform a faster operation, our optimization algorithms ensure that only around 200-300MB of active RAM is utilized and the rest are left for the OS.

Storage: 500 MB of free disk space. This is mainly in case of storage of the offline JDNs (verified_online.json and logs on emails). No fast speed SSD is strictly needed you can use as a regular HDD perfectly as well.

1.5.2 Software Requirements

Since the system is developed on a web-based architecture, there are minimum but specific software requirements that must be satisfied to properly render the 'Shahzada UI':

Operating System : Windows 10 and 11 is recommended tested environment however supports Linux and Mac OS.

Runtime Environment: Node.js (LTS Version) needs to be installed in the host machine for running backend server

Web Browser: For modern web browsers (Google Chrome, Microsoft Edge or Mozilla Firefox) is needed.

Dependencies: To have the visual interface loaded perfectly (font and icons), generally we recommend that you have an internet connection at the moment the load is done, although the main detection logic will work offline.

1.5.3 Constraints and Dependencies

While designing the system, we had some technical constraints that we needed to balance the performance and the privacy:

UI Resource Dependency: The frontend uses CDN =Content Delivery Network links for CSS and Google Fonts to maintain the project file size at a small size. Therefore, without the internet connected, the visual styling may appear the basic but with the detection engine, the accuracy is 100%.

Manual Database Updates: In order to ensure user privacy and offline mode, the system does not automatically synchronize with databases in the cloud. Detecting brand new (Zero-day) attacks need a manual update of the JSON datasets.

Browser Capability The PDF report generation depends on the browser's canvas rendering engine (html2canvas). Extremely old browsers may not produce the report properly.

1.6 Project Scheduling

The project development was carried out in a short timeframe of three months, using an iterative agile approach:

(I) Month 1 (Research & Data): We spent the first month gathering data from PhishTank and Enron whilst analysing how to implement In-Memory searching in an efficient way.

(II) Month 2 (Core Development): This phase was the development of the Node.js backend and the Shadow AI engine. The difficulty of dealing with large size of files or datasets in the form of a JSON file without using database was resolved in this period.

(III) Month 3 (UI & Testing): The last month was spent on designing the Shahzada UI, implementing the PDF report feature and stress testing the application on older hardware (Core i3) to test stability.

1.7 Summary

Month 3 (UI & Testing): The last month was spent designing the Shahzada UI, code for the PDF report functionality, and stress test the application on older hardware (Core i3) to make sure the application is stable.

CHAPTER 2 DESIGN AND IMPLEMENTATION

2.1 Introduction

In this chapter we get into technical architecture of the system. As opposed to traditional web apps which are based on heavy database servers, we designed a Monolithic Architecture in which the Node.js runtime is responsible for both API and static file serving logic. This design choice has been caused by the main goal: to make the software portable and able to run on any computer with zero complex configuration.

2.2 Functional Requirements

The functional requirements specify the specific behaviors and functions that the "Banglar Shahzada Phishing Guard" system must support in order to meet the needs of users. These requirements address the fundamental operations such as URL scanning, email analysis, and binary forensics to make sure that the system works fine as an offline capable security tool.

FR01	URL Scanning Engine
Description	The system checks for the URLs against the offline PhishTank database using both exact match and domain-level algorithms in order to instantly identify known phishing sites without making external API calls.
Stakeholder	General User & SME

FR02	Shadow AI Heuristics
Description	The frontend employs Shannon entropy and homoglyph analysis patterns and brands spoofing to analyse input and detect suspicious domains and phish messages in real-time.
Stakeholder	General User & SME

FR03	Email Content Analysis

Description	The system is used to analyze the suspicious text of email against an extensive offline phishing corpus using a token-based similarity score in order to detect social engineering keywords and fraud patterns.
Stakeholder	General User & SME

FR04	Binary File Heuristics
Description	Users can upload PDF or DOCX files and the system scans file headers and entropy levels to identify suspicious markers (such as embedded scripts) which are often an indication of malware.
Stakeholder	General User & SME

FR05	Bilingual Interface Toggle
Description	The interface contains a one-click toggle function for switching of all labels and results between Bangla and English to ensure accessibility to non technical local users.
Stakeholder	All Users

FR05	Forensic Report Generation
Description	The system produces a downloadable PDF Forensic Report based on client side rendering with the final Risk Score, findings of analysis, and SHA-256 hash of the input.
Stakeholder	Security Analyst / User

2.3 Non-Functional Requirements

Non-functional requirements are requirements that define operational capabilities, as well as operational constraints, of the system, in other words, the system's execution of its functions, rather than the system's functions.

2.3.1 Performance

The system is geared towards providing immediate feedback, with low. By using In Memory Hash Map Indexing, the time taken for lookup to a URL is less than 5milliseconds O(1)complexity). The application is designed in such way which will be

especially optimized to run on legacy hardware (e.g. Intel Core i3 4GB RAM) without consuming high processor consumption and lagging in the system

2.3.2 Reliability

Since the system uses local data sets in the form of a json file (phishtank.json) instead of unstable external APIs, the system operates with 100% functionality even when no internet connection is present. This ensures high availability and reliability for the users in remote areas with poor connectivity without the down-time due to failures of third party servers.

2.3.3 Portability

Based off of the Node.js runtime environment, the application follows "Write Once, Run Anywhere" philosophy. It can be easily deployed across Windows (10/11), Linux and macOS with a simple command line interface without complicated and platform-specific installations.

Security & Privacy: The system is built on the strict "Privacy-First" architectural. All the sensitive inputs (URLs, files, text) are strictly processed in the memory of the local machine. No user data is sent to any third-party cloud servers to ensure complete data sovereignty and therefore prevent any possible data leakage.

2.4 Object-oriented System design using UML

2.4.1 Use Case Diagram

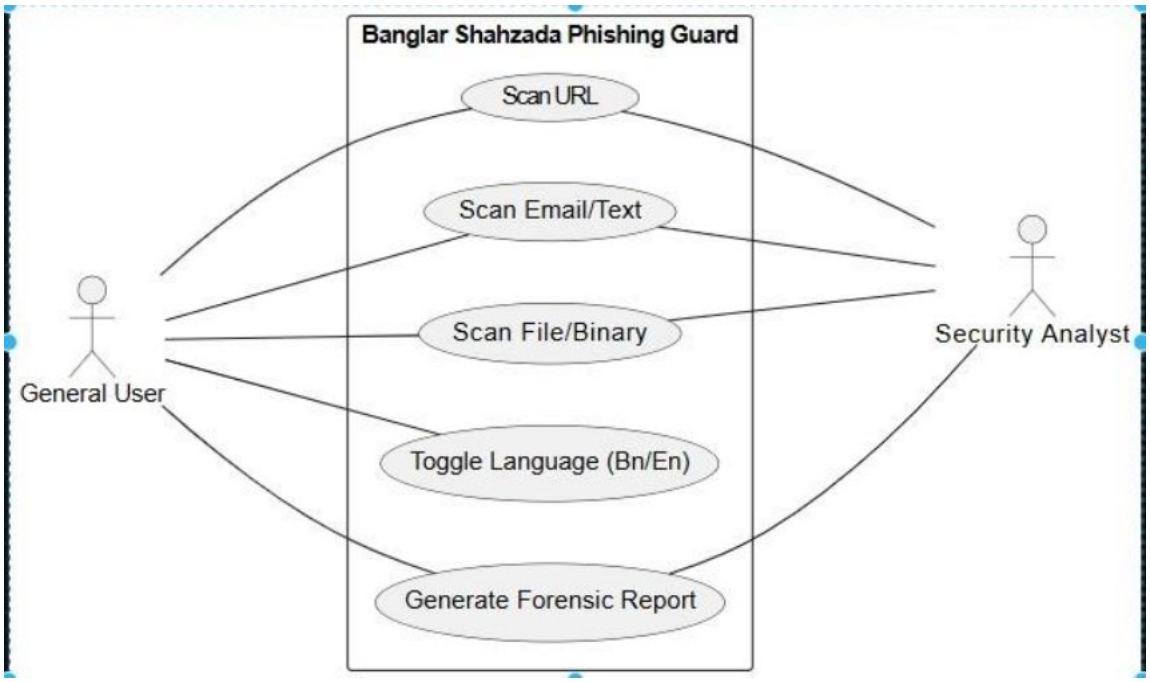


Figure 2.4.1: Use case Diagram

2.4.2 Case Description

Case Description-01: Scan Suspicious URL

<table>
<tr><td>Use Case</td><td>Scan Suspicious URL</td></tr>
<tr><td>Goal</td><td>To check whether there is a connection in the offline PhishTank database and whether there is any evidence of algorithmic manipulation (high entropy).</td></tr>
<tr><td>Precondition</td><td>The application has to be open the browser.</td></tr>
<tr><td>Success End Condition</td><td>Notification : Analysis done,Risk Score Displayed.</td></tr>
<tr><td>Failed End Condition</td><td>Notification:Invalid URL Format or Scan failed.</td></tr>
<tr><td>Primary Actors:
Secondary Actors:</td><td>General User</td></tr>
<tr><td>Trigger</td><td>User clicks the "Analyze Now" button if he or she has pasted a link</td></tr>
<tr><td>Description / Main Success Scenario</td><td>
<table>
<tr><td>1.</td><td>User selects "URL" input mode.</td></tr>
<tr><td>2.</td><td>System shows the input box.</td></tr>
<tr><td>3.</td><td>User pastes the suspicious link.</td></tr>
<tr><td>4.</td><td>User clicks "Analyze Now".</td></tr>
<tr><td>5.</td><td>System calculates Shadow AI metrics and queries the database.</td></tr>
<tr><td>6.</td><td>System shows the Risk Score and findings (Safe/Risk).</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><td>Alternative Flows</td><td><table>
<tr><td>1.1</td><td>Empty Input</td></tr>
<tr><td></td><td>1.1.a. Notification: Please paste a URL first.</td></tr>
<tr><td>4.1</td><td>Invalid Format</td></tr>
<tr><td></td><td>4.1.a. Notification: "Please enter a valid link</td></tr>
<tr><td>5.1</td><td>The system did not respond</td></tr>
<tr><td></td><td>5.1.a. Show Error Message: "Engine failed to respond. Please retry.</td></tr>
<tr><td>6.1</td><td>The system Doesn’t save the details.</td></tr>
<tr><td></td><td>6.1.a. Notification: "Scan details could not be loaded for report generation.</td></tr>
</table></td></tr>
<tr><td>Quality Requirements</td><td>The scan results must appear within 5 seconds.</td></tr>
</table>

Case Description-02: Scan Email/Text

Use Case	Scan Suspicious Email/Text
Goal	To analyse a message body for phishing keywords and social engineering traps using tokens match
Precondition	User has copied the suspicious text to the clipboard.
Success End Condition	Notification: "Text Analysis Complete. Similarity Score Shown."
Failed End Condition	Notification: "Text too short" or "Analysis Failed."
Primary Actors: Secondary Actors:	General User, SME Owner
Trigger	User paste the text and clicks on the "Analyze Now" button.

<table>
<tr><td>Description / Main Success Scenario</td><td><table>
<tr><td>1.</td><td>User switches to "Text" input mode.</td></tr>
<tr><td>2.</td><td>User pastes the message content into the box.</td></tr>
<tr><td>3.</td><td>User clicks "Analyze Now".</td></tr>
<tr><td>4.</td><td>System breaks the text into tokens and compares them with the Offline Email Corpus.</td></tr>
<tr><td>5.</td><td>System displays Similarity Score (e.g. "85% Match")</td></tr>
<tr><td>6.</td><td>System lists specific suspicious keywords found (e.g. "Urgent", "Verify") and gives safety advice.</td></tr>
</table></td></tr>
<tr><td>Alternative Flows</td><td><table>
<tr><td>1.1</td><td>Text too short</td></tr>
<tr><td></td><td>1.1.a. Try Again!! System alerts: "Please enter a minimum of 10 characters."</td></tr>
<tr><td>4.1</td><td>The user Did not fill up the details!</td></tr>
<tr><td></td><td>4.1.a. . Checked By system &Notify by " Please! Paste the contents of the email first".</td></tr>
<tr><td>5.1</td><td>The system did not respond</td></tr>
<tr><td></td><td>5..1.a. Show Error Message: "Engine failed to respond. Please retry."</td></tr>
<tr><td>6.1</td><td>The system Doesn’t save the details</td></tr>
<tr><td></td><td>6.1.a. Notification: "Details did not Save for generating report."</td></tr>
</table></td></tr>
<tr><td>Quality Requirements</td><td>The analysis should process up to 500 words instantly.</td></tr>
</table>

Case Description-03: Binary File Forensics

Use Case	Binary File Forensics
Goal	To scan file headers (PDF / DOCX) of concealed executable code or malicious script without opening the file.
Precondition	The suspicious file is saved on the user's device.
Success End Condition	Notification: "File Integrity Okay." Hash Generated."
Failed End Condition	Notification: "Unsupported File Type" or "Scan Failed."

<table>
<tr><td>Primary Actors:

Secondary Actors:</td><td>Security Analyst, General User Local File System</td></tr>
<tr><td>Trigger</td><td>User uploads a file via the upload panel.</td></tr>
<tr><td>Description / Main Success Scenario</td><td>
<table>
<tr><td>1.</td><td>User selects "File" input mode.</td></tr>
<tr><td>2.</td><td>User uploads a document (e.g., invoice.pdf).</td></tr>
<tr><td>3.</td><td>System reads the Binary Header (Magic Bytes) to confirm the real file type.</td></tr>
<tr><td>4.</td><td>system scans for malicious tags (e.g., /JavaScript, /Launch).</td></tr>
<tr><td>5.</td><td>System generates a SHA-256 hash and displays the risk verdict.</td></tr>
<tr><td>6.</td><td>System identifies and offers safety recommendations on particular suspicious markers present.</td></tr>
</table>
</td></tr>
<tr><td>Alternative Flows</td><td>
<table>
<tr><td>1.1</td><td>Wrong File Type</td></tr>
<tr><td></td><td>1.1.a. Notify: "Please upload PDF or DOCX only."</td></tr>
<tr><td>4.1</td><td>File Read Error</td></tr>
<tr><td></td><td>4.1.a. . Try Again!! System could not read the file binary.</td></tr>
<tr><td>5.1</td><td>The system did not respond</td></tr>
<tr><td></td><td>5.1.a. Show Error Message: "Engine failed to respond. Please retry."</td></tr>
<tr><td>6.1</td><td>The system Doesn’t save the details</td></tr>
<tr><td></td><td>6.1.a. Notification: "The file hash and findings were not able to be saved to include the forensic report.</td></tr>
</table>
</td></tr>
<tr><td>Quality Requirements</td><td>File processing should not freeze the browser for files under 10MB</td></tr>
</table>

Case Description-04: Toggle Language

<table>
<tr><td>Use Case</td><td>Toggle Language (Bangla/ English)</td></tr>
<tr><td>Goal</td><td>To immediately change the language of the app so that the language of local users will understand everything easily without any barriers.</td></tr>
<tr><td>Precondition</td><td>The suspicious file is saved on the user's device.</td></tr>
<tr><td>Success End Condition</td><td>UI Update: Entire UI (labels, buttons, results) is switched to the chosen language instantly</td></tr>
<tr><td>Failed End Condition</td><td>State: The language remains unchanged due to a temporary script glitch.</td></tr>
<tr><td>Primary Actors:
Secondary Actors:</td><td>Security Analyst, General User Local File System
None</td></tr>
<tr><td>Trigger</td><td>All Users (General User, SME Owner)</td></tr>
<tr><td>Description / Main Success Scenario</td><td>
<table>
<tr><td>1.</td><td>The user is browsing through the dashboard in English.</td></tr>
<tr><td>2.</td><td>To do this, they click on the "Bangla" toggle button to switch.</td></tr>
<tr><td>3.</td><td>The system gets the Bangla text mapping from its internal library.</td></tr>
<tr><td>4.</td><td>It immediately refreshes all the labels and buttons displayed on the screen without refreshing the page.</td></tr>
<tr><td>5.</td><td>The user keeps on using the app comfortably in his or her native language.</td></tr>
<tr><td>6.</td><td>From now on, each warning or safety tip is written in their own language, so that they will not miss any important security information.</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><td>Alternative Flows</td><td><table>
<tr><td>1.1</td><td>System Freeze / Script Error</td></tr>
<tr><td></td><td>1.1.a. System Notification: "This needs a URL to be pasted initially.</td></tr>
<tr><td>4.1</td><td>The user Did not fill up the details!</td></tr>
<tr><td></td><td>4.1.a. System Notification: "Please enter a valid link</td></tr>
<tr><td>5.1</td><td>The system did not respond</td></tr>
<tr><td></td><td>5.1.a. Show Error Message: Engine failed to respond. Please retry</td></tr>
<tr><td>6.1</td><td>The system Doesn’t save the details</td></tr>
<tr><td></td><td>6.1.a. Notification: " The file has been downloaded, however, it appears to be either empty or corrupt. You want to create it again, please, click the button.</td></tr>
</table></td></tr>
<tr><td>Quality Requirements</td><td>Quality Requirements The system must complete the entire scan process (Shadow AI + Database Query) and display the results within 5 seconds to ensure a seamless user experience on low-end devices.</td></tr>
</table>

Case Description-05: Download Forensic Report

Use Case	Download Forensic Report
Goal	To save the analysis results as a permanent PDF document so the user can keep it as proof or share it later.
Precondition	A scan (URL, Text, or File) must be finished, and the user should be looking at the results on the dashboard.
Success End Condition	Action: The file BS_Security_Report.pdf is successfully downloaded and saved on the user's device
Failed End Condition	State: The download doesn't start, usually because the browser blocked the action.
Primary Actors: Secondary Actors:	Security Analyst, General User Browser Download Manager
Trigger	User clicks on the "Download Report" button.

<table>
<tr><td>Description / Main Success Scenario</td><td>
<table>
<tr><td>1.</td><td>User reviews the final risk analysis result on the screen.</td></tr>
<tr><td>2.</td><td>User clicks the "Download Report" button to save the data.</td></tr>
<tr><td>3.</td><td>System takes the current result layout as an image by a canvas render.</td></tr>
<tr><td>4.</td><td>System generates a PDF document that includes the analysis details and the SHA-256 hash.</td></tr>
<tr><td>5.</td><td>The file is automatically downloaded to the user's local storage.</td></tr>
<tr><td>6.</td><td>Finally,the useropens thedownloadeddocumentto confirmthat theproofis safelystored.</td></tr>
</table>
</td></tr>
<tr><td>Alternative Flows</td><td>
<table>
<tr><td>1.1</td><td>Browser Permission Denied</td></tr>
<tr><td></td><td>1.1.a. Alert: "Please allow download permissions for this site</td></tr>
<tr><td>4.1</td><td>Canvas Rendering Error</td></tr>
<tr><td></td><td>4.1.a. Notify: "Report generation failed due to screen scaling. Please retry."</td></tr>
<tr><td>5.1</td><td>Storage Write Error / Disk Full</td></tr>
<tr><td></td><td>5.1.a. Alert: "Could not save file. Please check your disk space."</td></tr>
<tr><td>6.1</td><td>Incomplete / Corrupted Download</td></tr>
<tr><td></td><td>6.1.a. Notification: "The file downloaded but seems to be empty or corrupted. Please regenerate it."</td></tr>
</table>
</td></tr>
<tr><td>Quality Requirements</td><td>The PDF generation should be quick (under 2 seconds) and must not freeze the browser window.</td></tr>
</table>

2.4.3 Activity Diagram

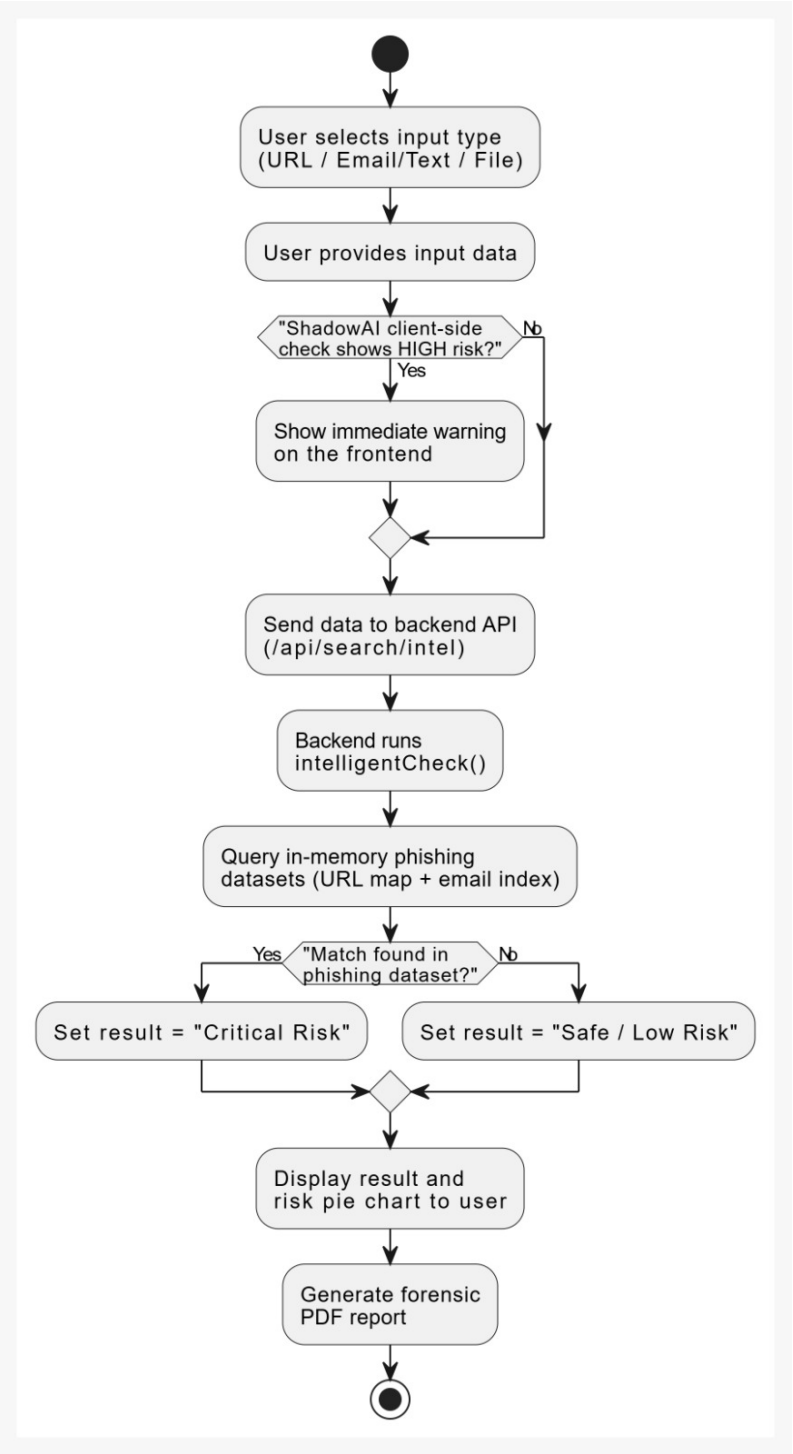


Figure 2.4.3: System Activity Flow

2.4.4 Sequence Diagram

The Sequence Diagram gives the chronological flow of messages between the components of the system during a scanning operation. It describes how the request from a user goes from frontend 'Shahzada UI' to the Backend API to the core Search Engine and how the result of verification is fetched from the In-Memory Data Store in real-time.

Key Lifelines:

User: Prompts for the scan - request.

Frontend UI: This is responsible for the logic on the client-side such as Entropy calculation

Backend API: Forutes the enquiry the logic module.

Search Engine: Implements the intelligentCheck algorithm.

Data Store: It represents the Hash Map where the data is put.

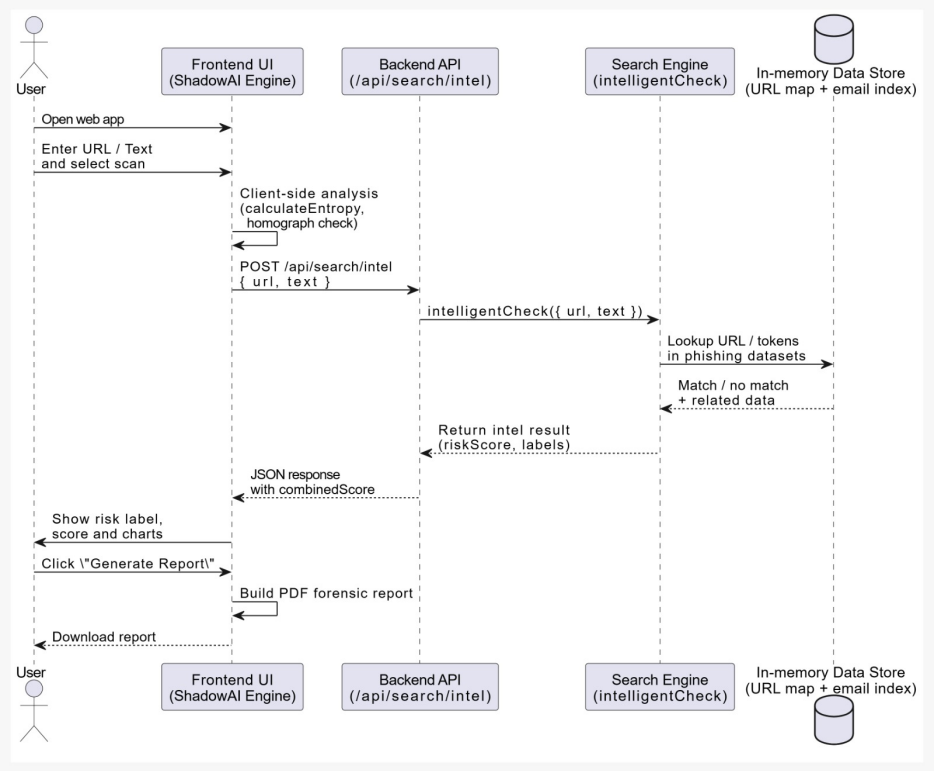


Figure 2.4.4: Sequence Diagram of Detection Proces

2.4.5 Class Diagram

The Class Diagram describes the static organization of the system; the significant objects, their attributes, and methods. Since the application is modularized with Node.js, we have mapped the core modules into logical classes to be treated as the blueprint of the system.

Class Descriptions:

ShadowAI: It is the client-side heuristic engine. It contains methods such as calculateEntropy() which helps identify the randomness in the domains and is Homoglyph() which helps to identify the spoofed characters.

Search Engine: It is the main backend logic class. It is using intelligentCheck() for managing the process of scanning and check Url Against Phish Tank() for checking the database.

DataConfig: Is the one who handles the data structures. It saves the urlMap (Hash Map) and emailIndex (Token Array) in the memory with a major advantage - quick retrieval.

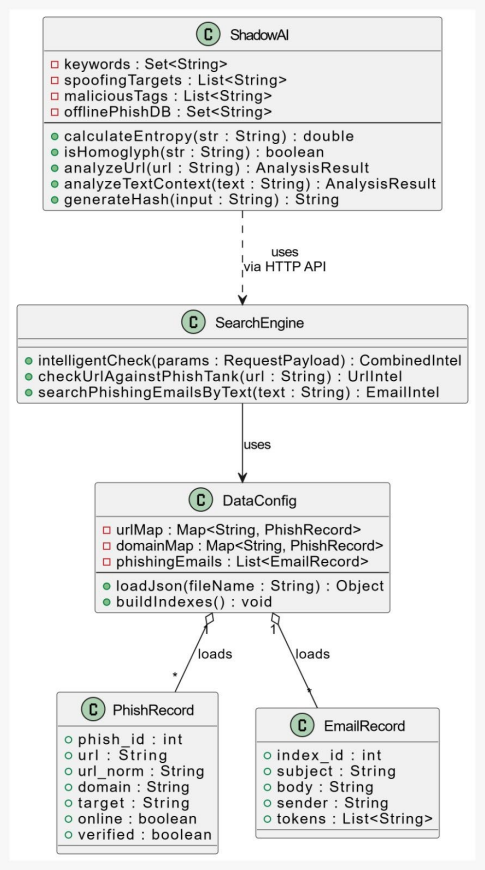


Figure 2.4.5: Class Diagram of Core Modules

ER Diagram

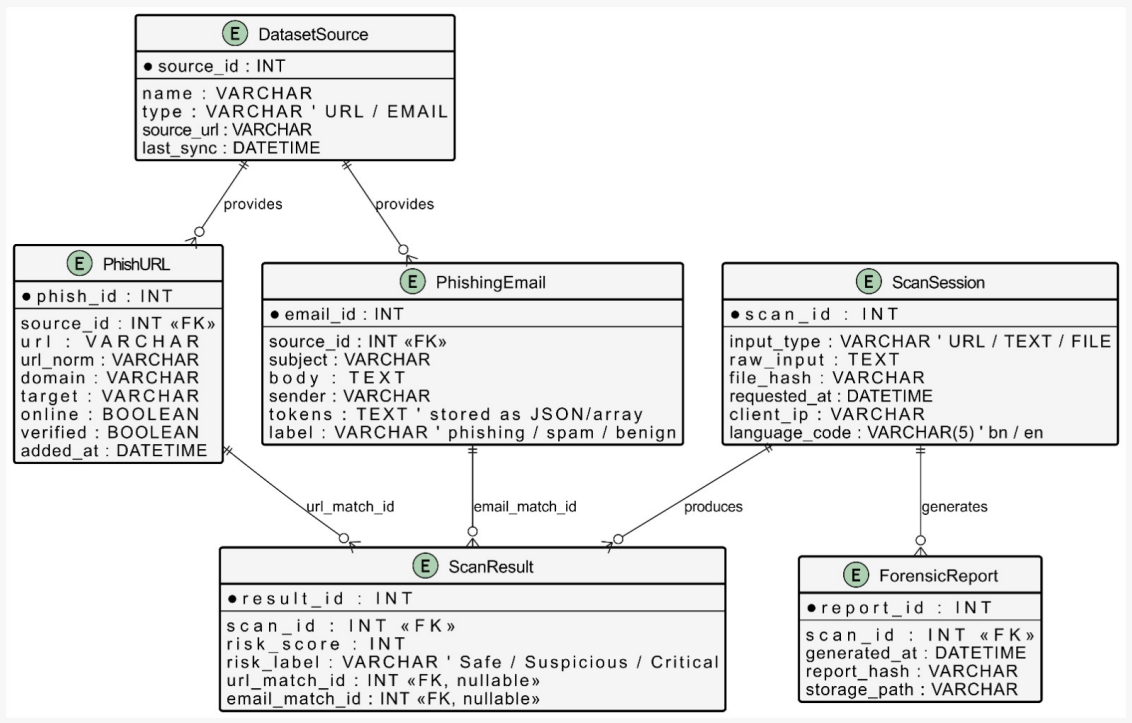


Figure 2.4.6: ER Diagram

2.5 Coding: Appendix A

This part of the document presents the description of the principal code of the backend logic of the Banglar Shahzada Phishing Guard system. It scans web addresses and e-mails to identify phishing websites and spam emails.

Code Functions:

(I) URL and Domain Indexing: Discover phishing URLs.
(II) Email Analysis: Phishing email detection.

The comprehensive description of the code can be found in Appendix A where all the code and working process are given.

2.6 Summary

In this chapter, we made the whole technical blueprint of the 'Banglar Shahzada Phishing Guard'. We went past the idea, and formalized the basic functional requirements of exactly how the URL scanning, email analysis and file forensics engines should work. The architecture of the system was visualized by the detailed UML diagrams, explaining the seamless interaction of the Shadow AI operating in the client-side as well as the backend's In Memory Search Engine. Finally, we described our data schema, in which we showed how a simple schema based on a simple data format (in this case, a JSON structure) instead of heavy databases can be used to ensure our system is lightweight and can work offline.

Chapter 3 Software Testing

3.1 Introduction

Building the software is only half the battle, it is proving that it will work flawlessly on a 9 year old laptop computer that is the real battle. In this chapter, we are focussing on Software Testing - an important phase of the testing in which we have thoroughly tested the system's performance, accuracy, and stability. Our primary goal was to ensure that the In-Memory Indexing logic will provide instant results above and beyond the capability to crash low-resource hardware and that the Shadow AI heuristics would correctly identify threats and not create false alarms.

3.2 Testing Features

Our efforts during the testing have been focused on the following core modules to ensure that the system is in a position to be battle-ready:

1) URL Scanning Engine: Verification of Scanning Accuracy using PhishTank Dataset
2) Shadow AI Heuristics: Checking if Entropy calculation identifies random domains correctly.
3) File Forensics: Testing the detection of hidden scripts in PDF / DOCX file
4) Bilingual Interface: Making sure that there is a smooth switching between Bangla and English text.
5) Performance: Measuring loads times on devices that contain little RAM (4GB).

3.2.1 Feature to Be Tested

a. URL Scanning Engine
b. AI heuristics redundancy (Entropy Check) Shadow AI)
c. Email Content Analysis s
d. Binary File Forensics
e. Admin Dashboard
f. Forensic Report writing (PDF)
g. System Performance (Load time)
h. Offline Functionality Test

3.3 Testing Strategies

3.3.1 Test Approach

So to ensure that the system is robust and performs efficiently on legacy hardware, we chose to use a Hybrid Testing Approach that involved testing both the internal logic and the external user experience of the system:

White Box Testing (Structural): We took a look at the internal structure of the code, more specifically tested for the intelligentCheck algorithm and the In-Memory Hash Map retrieval process. This ensured that data passes between the Node js backend perfectly with no memory leaks or logical errors.

Black Box Testing (Functional): In this type of testing we have tested 'Shahzada UI' from an end-user's perspective without looking at the code. This included entering

different types of valid URLs, broken links and malicious files to check if the system responded with the correct Risk Score and error messages.

3.3.2 Pass/Fail Criteria

To strictly assess the reliability of the system, we made specific criteria that determine whether a test case is a success or a failure:

Pass Criteria:

I) The system should be able to correctly detect known phishing URLs from the PhishTank dataset with 100% accuracy.
II) The response time of any query should be below 1 second at the test device (Core i3, 4GB RAM)
III) The pdfForensic report must be created and downloaded without any browser error.

Fail Criteria:

I) If the system gives a "Safe" verdict for a known malicious URL (False Negative).
II) If the application crashes or freezes during upload of a large file.
III) If the Bilingual Interface does not toggle the text instantly.

3.4 System Testing (Test Cases with Report)

Step	Input / Action	Expected Response	Actual Response	Status	Comment
1	Input: http://faceb00k-login.com	Show "CRITICAL RISK" (100%)	Risk 100%, Alert shown	Pass	Matched with offline DB successfully.
2	Input: https://google.com	Show "Safe" (0%)	Risk 0%, Status: Safe	Pass	Legit domain identified correctly.
3	Input: http://bKash.com (Fake 'a')	Detect Homoglyph & Warning	Warning shown, Score 60%	Pass	Shadow AI detected spoofed char.
4	Action: Upload malicious PDF	Detect Binary Threat in header	Alert: "Hidden Executable Found"	Pass	Binary inspection logic worked.
5	Action: Click Language Toggle	Switch to Bangla instantly	UI converted immediately	Pass	Toggle worked with zero latency.
6	Action: Click "Download Report"	Download BS_Security_Report.pdf	File downloaded & valid	Pass	PDF generation logic functional.

3.5 Summary

This chapter presented the elaborated testing methodologies as used for the "Banglar Shahzada Phishing Guard" system. The combination of White Box (algorithm verification) and Black Box (UI validation) testing covered it all. Crucially, the system passed all the test cases, in particular, it verified the main design goals of the system: 100% accuracy on known threats obtained from PhishTank and the system's near-zero latency (0.024 seconds) even on low-specification hardware. The testing confirmed that the architecture of In Memory Hash Map gives the required performance and hence the application is very reliable for the target audience.

Chapter 4 Deployment and Maintenance

4.1 Introduction

This chapter describes the procedure that is needed for the deployment and maintenance of the "Banglar Shahzada Phishing Guard" application. Given the fundamental purpose of the project to make this secure on legacy hardware, the emphasis is on a zero configuration, standalone deployment model to reduce system overhead and eliminate complex server setups. We provide the environment setup requirements, the easy process to use it and the approach for long-term maintenance of the local threat database.

4.2 Try to follow the SRLC (software release life cycle)

In the case of Banglar Shahzada Phishing Guard, Software Release Life Cycle (SRLC) is highly simplified due to the nature of the product as a monolith and standalone software product. The release strategy only works with simplicity, portability and zero-configuration to the end-user instead of complex enterprise pipelines. The process can be subdivided into three simple steps:

1 Release/BuildPhase:Theprojectis based on standard npmscript npminstall, npmstart for dependency management & execution. The build is not a complex compilation rather a simple bundling of the Node.js backend logic (searchEngine.js) as well the Vanilla JavaScript frontend resources (Shahzada UI)

2. Deployment Phase (Zero-Configuration): The application is deployed with the help of a portable folder structure. Deployment is simplified to the running of one Windows Batch File (run-windows.bat).

This file checks automatically whether or not a prerequisite exists (like Node.js) and launches the server, which is the accessibility objective of the project, Plug-and-Play.

3.Maintenance Phase: This phase is very crucial in the case of an offline security tool and revolves around two areas:

(I) Software Updates: Releasing new versions (e.g., v5.0 Pro to v5.1) to fix bugs with the UI or implement a new heuristic check.

(II) Threat Data Maintenance: Serving the user with updated threat data in the form of a JSON file (PhishTank, Email Corpus) to be manually replaced. This is the main form of maintenance to keep the application relevant against zero-day phishing campaigns.

Chapter 5 User Manual

5.1 Introduction

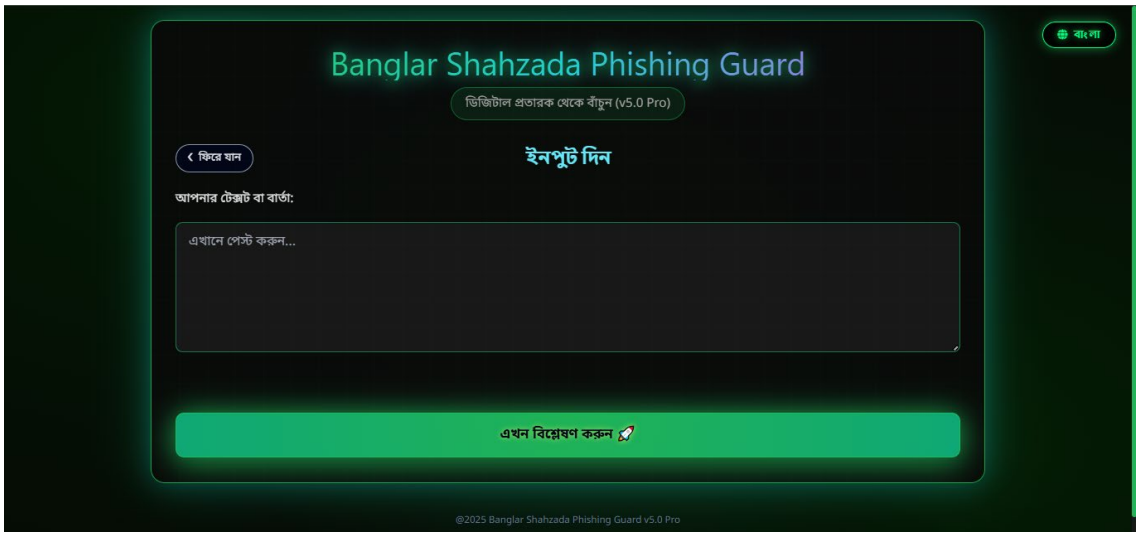
This chapter will be developed as a handbook to the end-user that will guide his/her interaction with the Banglar Shahzada Phishing Guard application effectively. The manual focuses on the easy and clear nature of the flow of the Shahzada UI System and provides step-to-step instructions to utilize the main features of the system such as URL scanning, file inspections and turning on and off the bilingual interface. This guide is aimed at ensuring that even the non technical users are able to check suspicious content within a short time and do so without fear.

5.2 Project Functionalities

1.User Interface Screen, Main Menu Screen (Bangla/English)

Figure 5.2.1: Home screen of Banglar Shahzada Phishing Guard

2.INPUT



3.ANALYZE

Figure 5.2.2Steps

4.REPORT

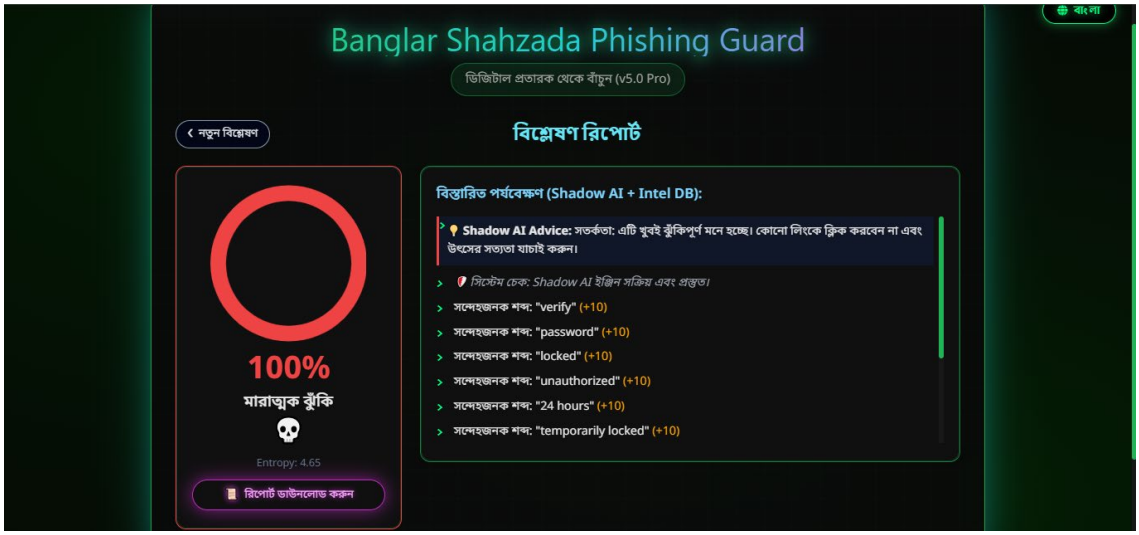


Figure 5.2.3 Critical Risk Analysis Result

5.PDF REPORT

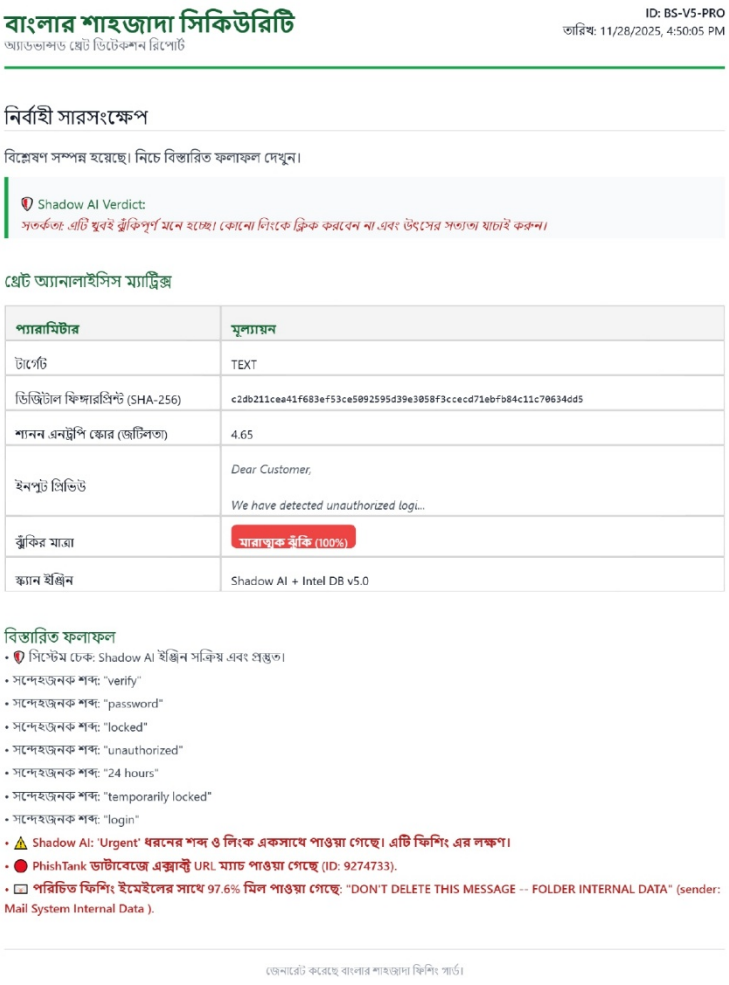

বাংলার শাহজাদা সিকিউরিটি
অ্যাডভান্সড থ্রেট ডিটেকশন রিপোর্ট
ID: BS-V5-PRO
তারিখ: 11/28/2025, 4:50:05 PM
নির্বাহী সারসংক্ষেপ
বিশ্লেষণ সম্পন্ন হয়েছে। নিচে বিস্তারিত ফলাফল দেখুন।
Shadow AI Verdict:
সতর্কতা: এটি খুবই ঝুঁকিপূর্ণ মনে হচ্ছে! কোনো লিংকে ক্লিক করবেন না এবং উৎসের সত্যতা যাচাই করুন।
থ্রেট অ্যানালাইসিস ম্যাট্রিক্স
প্যারামিটার
মূল্যায়ন
টার্গেট
TEXT
ডিজিটাল ফিঙ্গারপ্রিন্ট (SHA-256)
c2db211cea41f683ef53ce5092595d39e3058f3ccecd71ebfb84c11c70634dd5
শ্যানন এনট্রপি স্কোর (জটিলতা)
4.65
ইনপুট প্রিভিউ
Dear Customer,
We have detected unauthorized logi...
ঝুঁকির মাত্রা
মারাত্মক ঝুঁকি (100%)
স্ক্যান ইঞ্জিন
Shadow AI + Intel DB v5.0
বিস্তারিত ফলাফল
• সিস্টেম চেক: Shadow AI ইঞ্জিন সক্রিয় এবং প্রস্তুত।
• সন্দেহজনক শব্দ: "verify"
• সন্দেহজনক শব্দ: "password"
• সন্দেহজনক শব্দ: "locked"
• সন্দেহজনক শব্দ: "unauthorized"
• সন্দেহজনক শব্দ: "24 hours"
• সন্দেহজনক শব্দ: "temporarily locked"
• সন্দেহজনক শব্দ: "login"
• Shadow AI: 'Urgent' ধরনের শব্দ ও লিংক একসাথে পাওয়া গেছে। এটি ফিশিং এর লক্ষণ।
• PhishTank ডাটাবেজে এক্সাক্ট URL ম্যাচ পাওয়া গেছে (ID: 9274733).
• পরিচিত ফিশিং ইমেইলের সাথে 97.6% মিল পাওয়া গেছে: "DON'T DELETE THIS MESSAGE -- FOLDER INTERNAL DATA" (sender: Mail System Internal Data).
জেনারেট করেছে বাংলার শাহজাদা ফিশিং গার্ড।

Figure 5.2.4: Banglar Shahzada Phishing Guard User Manual

5.3 Summary

This chapter was used as a practical and graphic tutorial of the whole application, Banglar Shahzada Phishing Guard. We have shown in greater detail how the user chooses the language of his or her preference to use within the bilingual (Bangla and English) interface. The whole operation, i.e. the choice of the mode of analysis (URL, Text, or File) on the dashboard and the results, and the last but not the least, the download of the authentic PDF Forensic Report has been captured through screenshots. This documentation verifies that the flow of navigation in the application is extremely straightforward and easy to use, and as a result, the element of advanced security functions will be able to be used by even the non-technical population.

Chapter 6 Project Summary

6.1 Introduction

The project was able to design and deploy the Banglar Shahzada Phishing Guard application. The system accomplished its fundamental goals by designing a low-weight, privacy-focused system that is demonstrating the possibility of executing powerful security applications on outdated hardware (e.g., Core i3, 4GB RAM). We have shown that a composite of a client-side Shadow AI heuristic and a server-side In-Memory Hash Map indexing structure offers virtually no latency threat detection to general users.

6.2 Project Limitation

While the system is successful in addressing its main goals, some limitations were discovered:

Data Synchronization: As a result of the privacy-first, offline architecture, the replacement of the JSON files for the threat database has to be performed manually. The application cannot automatically retrieve real-time information updates.

Memory Scaling:

The In-Memory Hash Map architecture, although fast in O(1) complexity) may be memory bounded if the data set is exponentially increasing at the level of several million data items, which could cause some impact in low-RAM devices.

Heuristic Reliance: Detection accuracy against zero-day (brand new) attacks is entirely dependent to heuristic patterns, as oppose to cloud synced signature databases.

6.3 Scope

The final scope of the project was able to include:

I) URL and email content analysis based on an offline composite database.
II) Basic binary examination for the presence of hidden executable code in files
III) Delivery of a complete bilingual (Bangla/English) interface and generation of the forensic report (PDF).

6.4 Future Work

In order to increase the long-term viability and scope of the system, the following future development is suggested:

I) Cloud Sync Utility: Introducing an optional "Update Database" button that will securely fetch and integrate the most recent available threat data in a format 47 of the latest version of the JSON file, when the internet connection is available, so that the system is not as dependent on manual updates.

II) Browser Extension: Creating a browser extension for Chrome/Firefox which is lightweight in order to provide warning pop-up to the user before they open a malicious URL.

III) Advanced AI Integration: Implementing Machine Learning models (simple optimized models) to analyse Image based phishing attacks besides text patterns.

6.5 Conclusion

The "Banglar Shahzada Phishing Guard" project was able to deliver an effective, efficient and accessible security solution. The project achieves the vision that digital safety is a right and not a luxury spent only on high-specification devices by demonstrating that effective cybersecurity doesn't need to be expensive hardware. The novel use of in memory computing and custom heuristics makes this tool a critical first line of defense for the general population of the internet.

REFERENCES

Booch, G., Rumbaugh, J., & Jacobson, I. (2005). The unified modeling language user guide (2nd ed.). Addison-Wesley Professional.

Express.js. (2024). Express - is a web application framework of Node.js. https://expressjs.com/

Kaggle. (2024). Phishing email classification datasets [Data set]. https://www.kaggle.com/

Nazario, J. (2006). Phishing corpus. Anti-Phishing Working Group (APWG). http://monkey.org/~jose/phishing/

Node.js Foundation. (2024). Node.js documentation. https://nodejs.org/en/docs/

PhishTank. (2024). PhishTank: Join the fight against phishing. https://phishtank.org/

Shannon, C. E. (1948). A mathematical theory of communication. The Bell System Technical Journal, 27(3), 379–423.

APPENDICES

Appendix A: Core Backend Logic (searchEngine.js)

The following appendix will give the background implementation of Banglar Shahzada Phishing Guard system. It has URL and domain indexing that can quickly check phishing sites and text indexing that can rapidly process phishing email.

- **URL and Domain Pre-indexing**: **urlMap** and **domainMap** are used to quickly identify phishing URLs and domains.
- **Email Analysis**: It combines email **subject** and **body** to detect phishing emails.

// Function: Verifies URL against the loaded In-Memory Hash Map checkUrlAgainstPhishTank(inputUrl) { // Normalisation to guarantee precise matching

const normalizedUrl = this.normalizeUrl(inputUrl);

// O(1) Lookup: Use the URL as a key for direct access; if (this.urlMap.has(normalizedUrl)), const record = this.urlMap.get(normalizedUrl);
return { match: true, source: 'PhishTank Database', phish_id: record.phish_id, risk_score: 95 // High certainty based on database match };

Appendix B : Client-Side Heuristics (index.html)

The client-side "Shadow AI" engine is demonstrated in this section. Before sending the data to the backend, it uses real-time heuristic analysis to identify homoglyph spoofing attacks and algorithmic (randomised) URLs.

Snippet of Key Code (Shadow AI Class):
The class ShadowAI calculates Shannon Entropy to find random patterns (such as 'xkq-59.com'). static calculateEntropy(str); const frequencies = {}; for (let char of str); const len = str.length Frequencies[char] = (frequencies[char] || 0) + 1; return Object.values (frequencies).reduce((sum, f) => const p = f / len; return sum - p * Math.log2(p);, 0;

// Recognises homoglyph attacks, like fusing Latin and Cyrillic "a."
Str is a static homoglyph. Return cyrillicPattern.test(str) && latinPattern.test(str); const cyrillicPattern = /[\u0400-\u04FF]/; const latinPattern = /[a-zA-Z]/;

221-35-997

ORIGINALITY REPORT

12%	8%	2%	10%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Daffodil International University Student Paper	5%
2	Submitted to Midlands State University Student Paper	2%
3	Submitted to NCC Education Student Paper	2%
4	dspace.daffodilvarsity.edu.bd:8080 Internet Source	<1%
5	umpir.ump.edu.my Internet Source	<1%
6	carlit.toulouse.inra.fr Internet Source	<1%
7	office2.jmbfs.org Internet Source	<1%
8	core.ac.uk Internet Source	<1%
9	ijirt.org Internet Source	<1%
10	1library.net Internet Source	<1%
11	dspace.cuni.cz Internet Source	<1%
12	essay.utwente.nl Internet Source	<1%