Configuration Manual

MSc Research Project
MSc in CyberSecurity

Karthik Venkata Gautham Iyer
Student ID: x23306424

School of Computing
National College of Ireland

Supervisor: Jawad Salahuddin

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Karthik Venkata Gautham Iyer
Student ID:	X23306424
Programme:	MSc in Cybersecurity **Year:** 2024-25
Module:	Practicum
Supervisor:	Jawad Salahuddin
Submission Due Date:	15th September, 2025
Project Title:	Deep Learning based real time detection of BitB attacks
Word Count:	1332 **PageCount** 12

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Gautham Iyer
Date:	15/09/2025

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

BitB Phishing Detection System - Configuration & Setup Manual

Karthik Venkata Gautham Iyer
X23306424
National College of Ireland

1. Introduction

The **BitB Phishing Detection System** is a browser extension and backend service designed to detect **Browser-in-the-Browser (BitB)** phishing attacks in real-time. It works for **Google Chrome** and **Microsoft Edge** by extracting predefined security features from visited web pages, sending them to a backend **ONNX-based machine learning model**, and displaying detection results directly in the browser.

This manual explains the **setup, configuration, and technical details** for the system, including **feature extraction logic, model prediction process, and potential extensions**.

2. Prerequisites

Before installing, check and verify that you meet the following prerequisites:

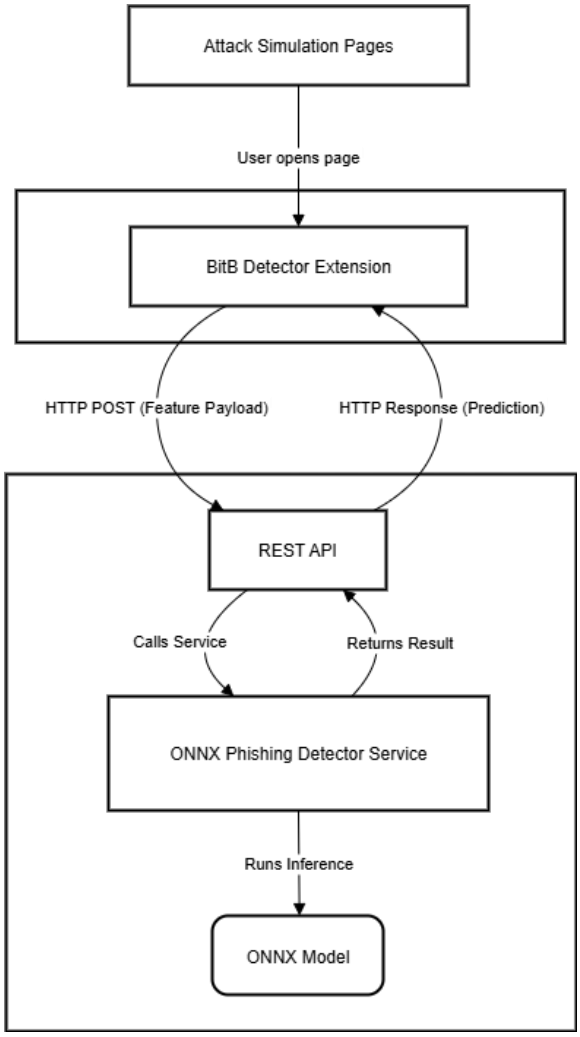
- Google Chrome (latest version) and/or Microsoft Edge
- Java 17 and above (if Spring Boot backend)
- Maven (to build backend service)
- Node.js & npm (for UI demo if needed)
- Python >3.10 (optional, to retrain models)
- ONNX Runtime Java library as a dependency in backend project
- Internet access to install dependencies

3. System Architecture

The solution has three main components:

1. **Browser Extension**
    - Extracts 11 fixed features domain/URL features and sends them to the backend API
    - Displays the detection result banner to the user
2. **Backend Service (Spring Boot)**
    - Loads and runs the ONNX model (bitb_phishing_detector.onnx).
    - Receives features from the extension and returns predictions.
    - Uses probability thresholds to classify pages into:
        - **Safe**
        - **Suspicious**
        - **BitB Attack**

3. **Model Training Script (Python)**
    - Trains an XGBoost classifier on the data from file, bitb_dataset.csv
    - Then converts the trained model to ONNX so it can be use with the backend service.



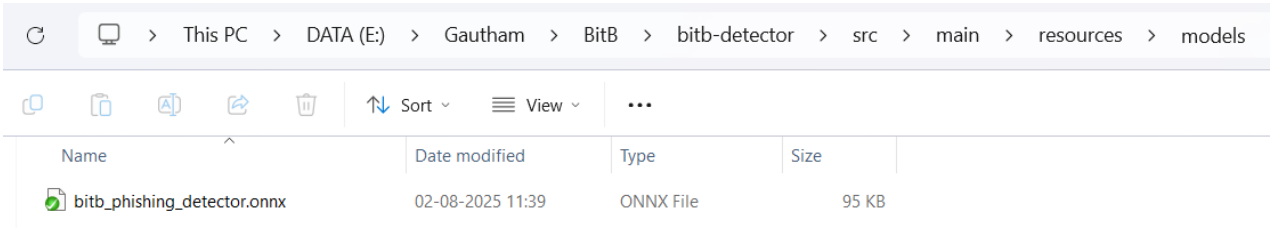
4. Backend Detector Setup

1. Extract the '**bitb-detector**' module from the provided ZIP file.
2. Open a terminal in the extracted folder.
3. Build the project using:

```
mvn clean install
```

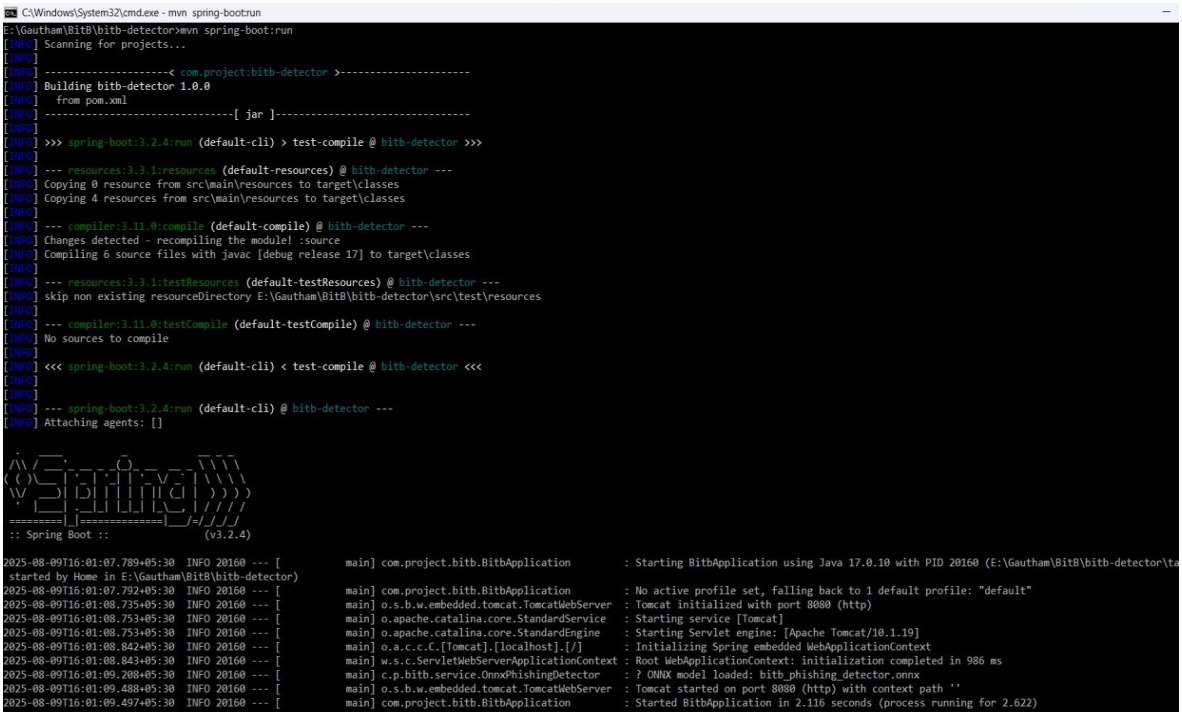
```
E:\Gautham\BitB\bitb-detector>mvn clean install
[INFO] Scanning for projects...
[INFO]
[INFO] ----------------------< com.project:bitb-detector >----------------------
[INFO] Building bitb-detector 1.0.0
[INFO]   from pom.xml
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- clean:3.3.2:clean (default-clean) @ bitb-detector ---
[INFO] Deleting E:\Gautham\BitB\bitb-detector\target
[INFO]
[INFO] --- resources:3.3.1:resources (default-resources) @ bitb-detector ---
[INFO] Copying 0 resource from src\main\resources to target\classes
[INFO] Copying 4 resources from src\main\resources to target\classes
[INFO]
[INFO] --- compiler:3.11.0:compile (default-compile) @ bitb-detector ---
[INFO] Changes detected - recompiling the module! :source
[INFO] Compiling 6 source files with javac [debug release 17] to target\classes
[INFO]
[INFO] --- resources:3.3.1:testResources (default-testResources) @ bitb-detector ---
[INFO] skip non existing resourceDirectory E:\Gautham\BitB\bitb-detector\src\test\resources
[INFO]
[INFO] --- compiler:3.11.0:testCompile (default-testCompile) @ bitb-detector ---
[INFO] No sources to compile
[INFO]
[INFO] --- surefire:3.1.2:test (default-test) @ bitb-detector ---
[INFO] No tests to run.
[INFO]
[INFO] --- jar:3.3.0:jar (default-jar) @ bitb-detector ---
[INFO] Building jar: E:\Gautham\BitB\bitb-detector\target\bitb-detector-1.0.0.jar
[INFO]
[INFO] --- spring-boot:3.2.4:repackage (repackage) @ bitb-detector ---
[INFO] Replacing main artifact E:\Gautham\BitB\bitb-detector\target\bitb-detector-1.0.0.jar with repackaged archive, adding nested dependencies in BOOT-INF/.
[INFO] The original artifact has been renamed to E:\Gautham\BitB\bitb-detector\target\bitb-detector-1.0.0.jar.original
[INFO]
[INFO] --- install:3.1.1:install (default-install) @ bitb-detector ---
[INFO] Installing E:\Gautham\BitB\bitb-detector\pom.xml to C:\Users\Home\.m2\repository\com\project\bitb-detector\1.0.0\bitb-detector-1.0.0.pom
[INFO] Installing E:\Gautham\BitB\bitb-detector\target\bitb-detector-1.0.0.jar to C:\Users\Home\.m2\repository\com\project\bitb-detector\1.0.0\bitb-detector-1.0.0.jar
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  20.727 s
[INFO] Finished at: 2025-08-09T15:51:40+05:30
[INFO] ------------------------------------------------------------------------

E:\Gautham\BitB\bitb-detector>
```

4. Ensure the ONNX model file '**bitb_phishing_detector.onnx**' is placed in '**src/main/resources/models/**'.



5. Start the backend service:

   mvn spring-boot:run

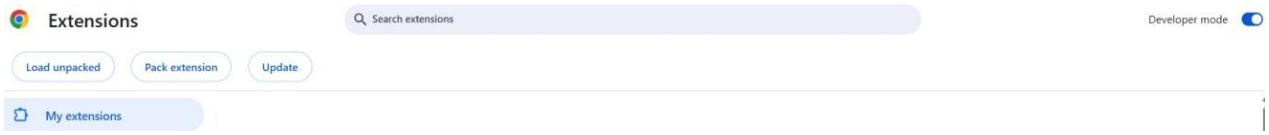


6. The service will be available at: **http://localhost:8080**

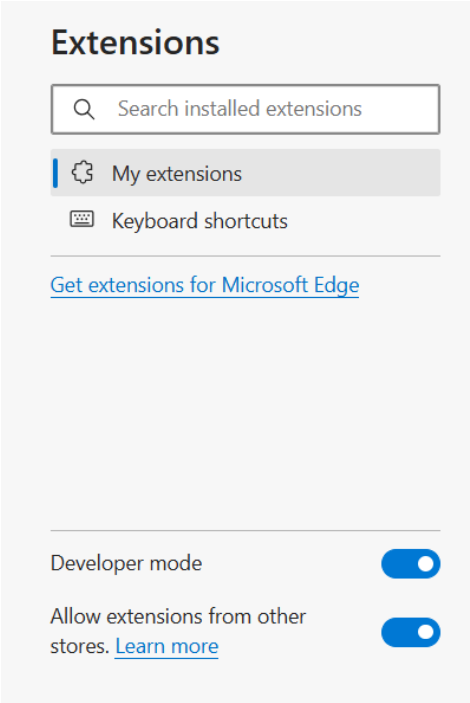
5. Chrome/Edge Browser Extension Setup

1. Extract the '**bitb-extension**' module.
2. Go to chrome://extensions (or edge://extensions in Edge).
3. Enable '**Developer mode**' (toggle switch in top-right).

Google Chrome

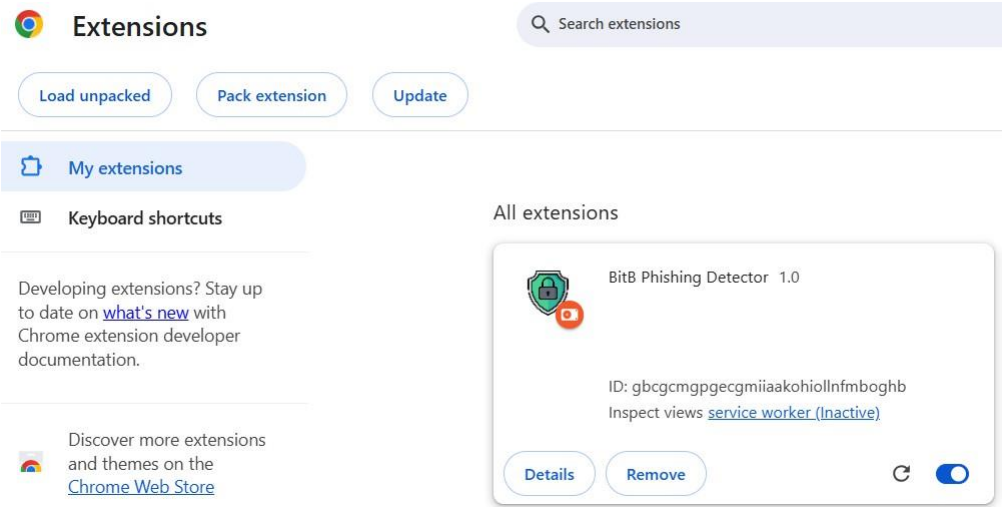


Microsoft Edge

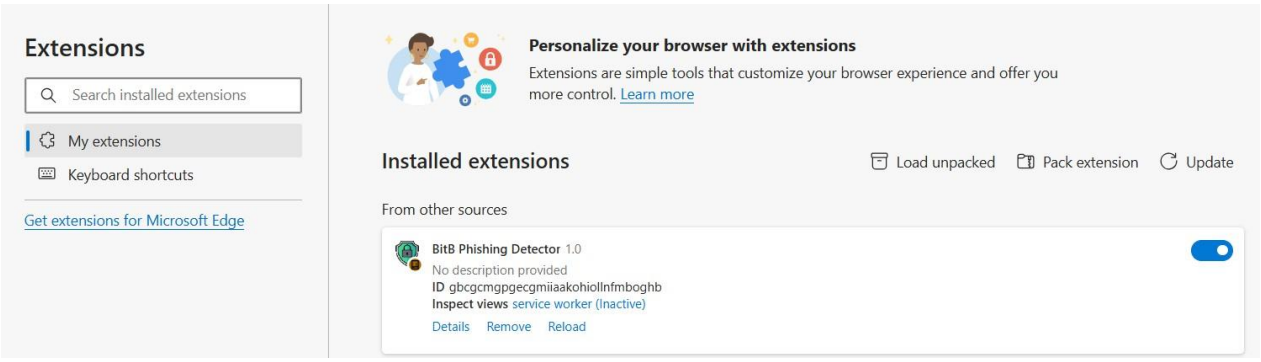


4. Click 'Load Unpacked' and select the extracted extension folder containing the **extension files** (manifest.json, featureExtractor.js, content.js, background.js, icons).

Google Chrome



Microsoft Edge

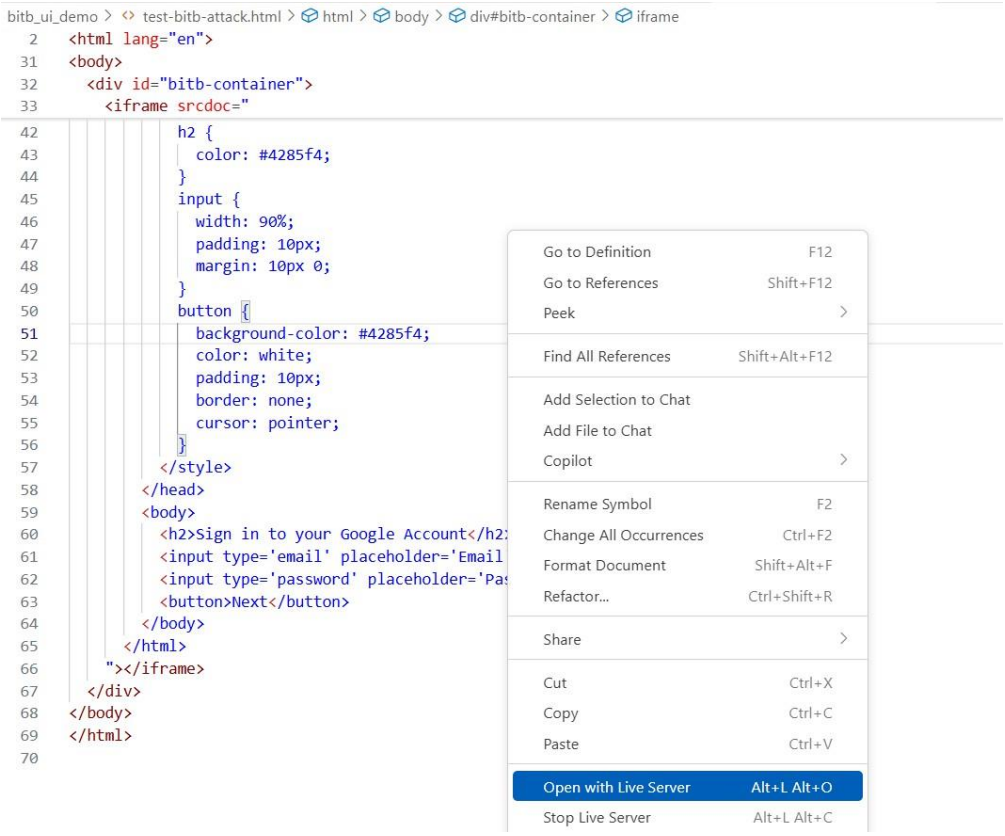


5. Ensure the backend service is running before using the extension.

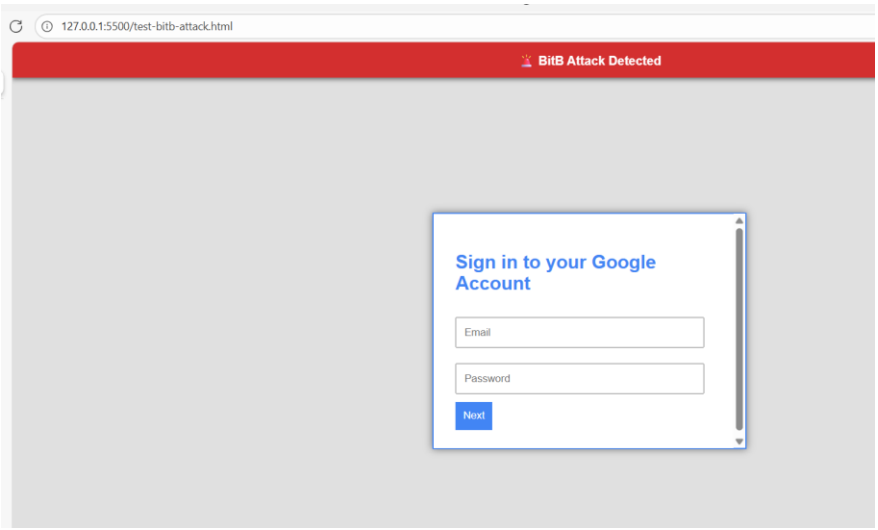
6. UI Demonstration Module Setup (Optional)

If you want to demonstrate the model in action without browsing live websites:

1. Open the **bitb_ui_demo** folder in **Visual Studio Code**.
2. Install the **Live Server extens**ion in **Visual Studio Code**.
3. Right-click test-bitb-attack.html → Select "Open with Live Server".



4. The UI demo will be available at http://127.0.0.1:5500 (or similar port).
5. Use it to test URLs and view classification results visually.



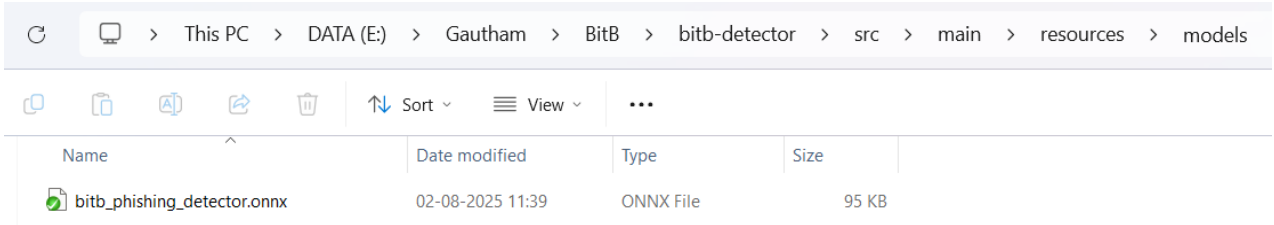
7. Feature Extraction – The 11 Features

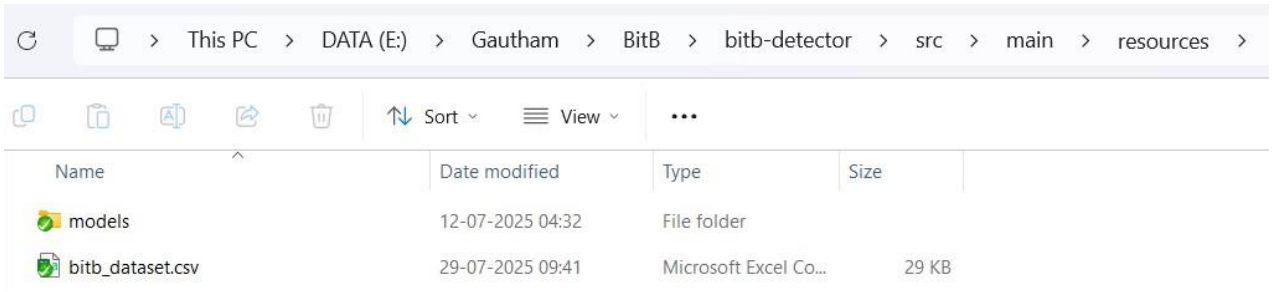
The browser extension collects the following **11 security features** from each webpage:

#	**Feature Name**	**Description**
1	has_iframe	Detects whether or not the page has <iframe> elements
2	dom_similarity *(windowSizeRatio)*	Compares screen size to window size to detect overlays.
3	has_dark_pattern	Detects hidden or overlay login UIs.
4	num_login_forms *(iframe depth)*	Detects nested iframes to detect embedding levels.
5	has_invisible_input	Detects hidden input fields that are commonly used to capture data and do unexpected things.
6	has_brand_logo *(login keywords)*	Detects text on the page for login keywords.
7	has_keyboard_traps	detecting keydown event listeners that may trap users.
8	page_text_similarity *(hostname entropy)*	Measures randomness of your domain name.
9	has_suspicious_js	Checks the URL for suspicious keywords e.g. login, secures, bank, etc.
10	domain_whitelisted	Flags for whether the domain is in safe domains list.
11	accesses_window_top	Detects if the login form actions point to an external domain.

8. Model Training Process (Python)

The backend model is trained in Python with the script called Training_Python_Script_New.txt that utilizes the dataset called bitb_dataset.csv, and the ONNX model is trained in XGBoost with the dataset called bitb_dataset.csv.





Prerequisites

- Installation of Python 3.9+
- Installation of the following packages
  pip install pandas xgboost onnxmltools scikit-learn

Files

- Training_Python_Script_New.txt – Python script with training and ONNX export code.
- bitb_dataset.csv – CSV file with featured values and labeled samples.

Steps to Train and Export the Model

1. Place both files in the same directory.
2. Open a terminal in that directory.
3. Run:
   python Training_Python_Script_New.txt
4. The script will:
   - Load bitb_dataset.csv into a Pandas DataFrame.
   - Split data into **features (X)** and **labels (y)**.
   - Encode labels using LabelEncoder (if categorical).
   - Train an **XGBoost Classifier** with:
     - n_estimators=100
     - max_depth=5
     - eval_metric="mlogloss"
   - Convert the trained model to **ONNX** format.
   - Save as bitb_phishing_detector.onnx.
5. Copy the generated bitb_phishing_detector.onnx file to:

backend/src/main/resources/models/

9. Final Prediction Logic (Backend)

When the ONNX model runs on the features, it outputs either:

- Logits (probability as a float) – which are converted to probabilities based on softmax.
- **Class labels** (0, 1, 2).

The Classification thresholds are:

- **BitB Attack (2)** → Probability ≥ 0.38 and is the highest probability among classes.
- **Suspicious (1)** → Probability ≥ 0.38 and is the highest probability among classes.
- **Safe (0)** → All other cases.

10. System Workflow

1. User accesses a webpage within Chrome/Edge.
2. The extension scrapes 11 fixed features from the page's DOM and URL.
3. The extension sends the features via HTTP POST to the backend Spring Boot service.
4. The backend loads the ONNX phishing detection model and returns a prediction of:
   - 0 → Safe
   - 1 → Suspicious
   - 2 → BitB Attack
5. The extension displays a colored banner at the top of the page based on the prediction.

11. Future Feature Expansion

While the system currently extracts **11 features**, the architecture allows for more advanced checks:

Technical Features

- Detection of iframe script injection
- Real time DOM mutation monitoring.
- Browser API interaction.

Research Enhancements

- Visual similarity detection to visually detect fake pages.
- AI based logo detection.
- AI based NLP scoring of phishing content.

12. Troubleshooting

- **No banner showing:** Ensure backend (localhost:8080) is running.
- **Extension not loading:** Check console logs in DevTools (F12 → Console).
- **Model mismatch error:** Ensure the backend uses the latest ONNX file.

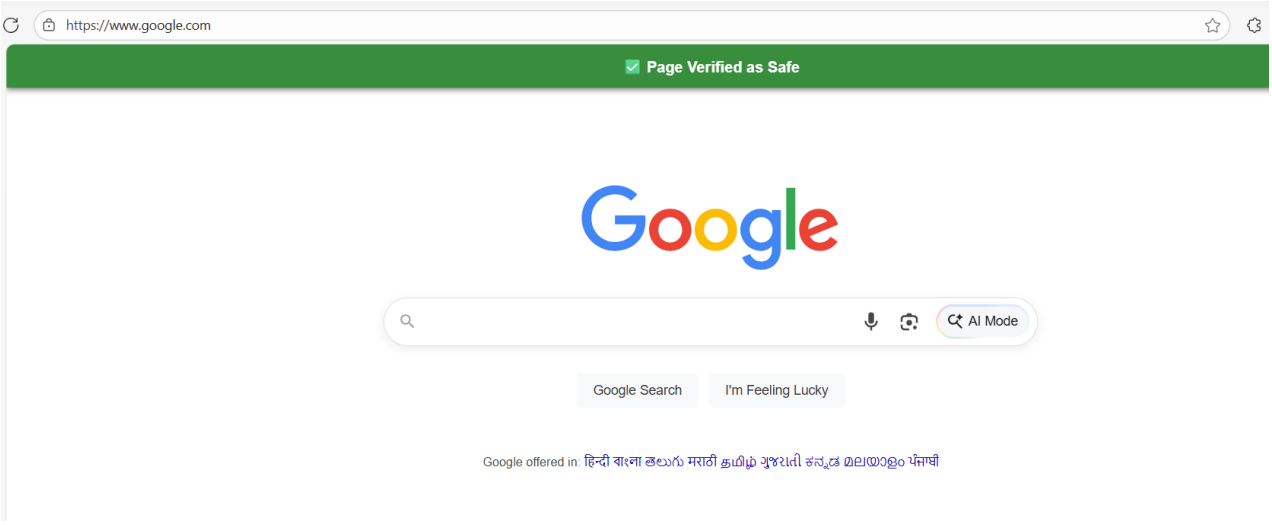
13. Verification & Testing

To ensure the modules are working correctly please go through the verification steps below, and compare the screenshots you take with what is shown in this document.

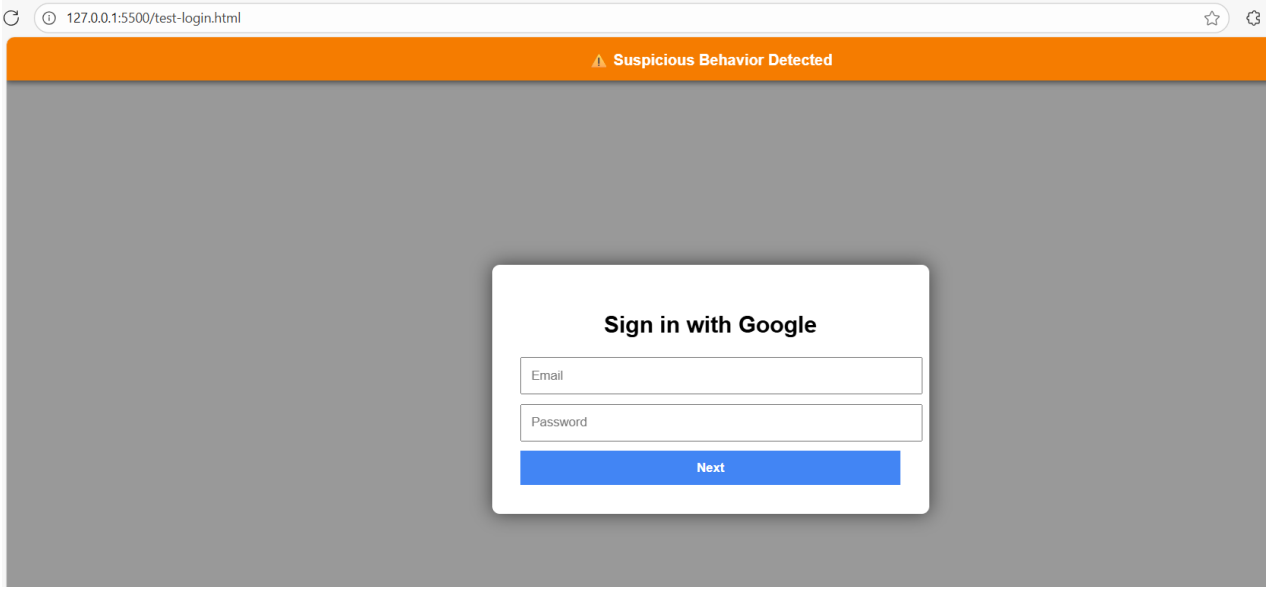
Chrome/Edge Extension Detection Banner

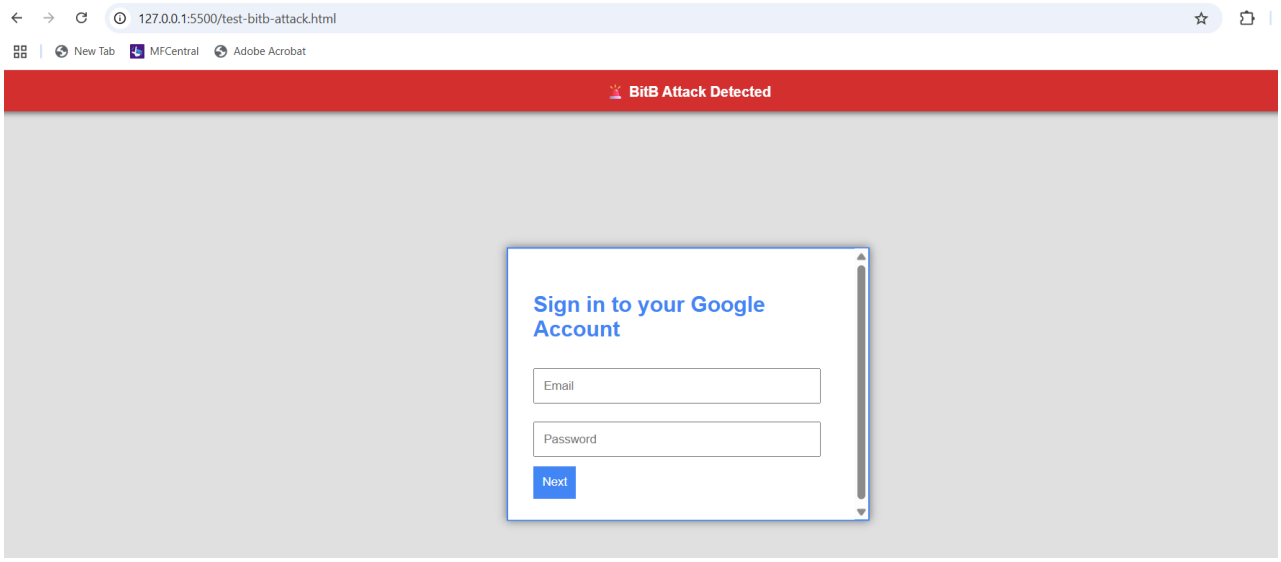
Once you have installed and enabled the extension, go to a safe site and a suspicious or a BitB test site.

- **Expected Result:**
  - Safe Site → Green banner: "✅ Page Verified as Safe"
  - Suspicious Site → Orange banner: "⚠ Suspicious Behavior Detected"
  - BitB Site → Red banner: "🚨 BitB Attack Detected"



`





Backend Logs (Spring Boot ONNX Prediction)

While browsing test sites, monitor the backend logs.

- **Expected Log Example:**

📥 Input features: [0.0, 0.8, 0.0, 0.0, 1.0, 1.0, 1.0, 0.81, 0.0, 0.0, 0.0]
📤 ONNX output (long[]): [1]
✅ Final prediction: 1, Score: 1.0

```
2025-08-09T17:15:56.934+05:30  INFO 26604 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet        : Completed initialization in 1 ms
2025-08-09T17:15:57.090+05:30  INFO 26604 --- [nio-8080-exec-4] c.p.bitb.service.OnnxPhishingDetector    : 📥 Input features: [0.0, 0.83, 0.0, 0.0, 0.0, 1.0, 1.0, 0.57, 0.0, 1.0,
2025-08-09T17:15:57.090+05:30  INFO 26604 --- [nio-8080-exec-3] c.p.bitb.service.OnnxPhishingDetector    : 📥 Input features: [0.0, 0.83, 0.0, 0.0, 0.0, 1.0, 1.0, 0.57, 0.0, 1.0,
2025-08-09T17:15:57.097+05:30  INFO 26604 --- [nio-8080-exec-4] c.p.bitb.service.OnnxPhishingDetector    : 📤 ONNX output (long[]): [0]
2025-08-09T17:15:57.097+05:30  INFO 26604 --- [nio-8080-exec-3] c.p.bitb.service.OnnxPhishingDetector    : 📤 ONNX output (long[]): [0]
2025-08-09T17:15:57.097+05:30  INFO 26604 --- [nio-8080-exec-4] c.p.bitb.service.OnnxPhishingDetector    : ✅ Final prediction: 0, Score: 1.0
2025-08-09T17:15:57.097+05:30  INFO 26604 --- [nio-8080-exec-3] c.p.bitb.service.OnnxPhishingDetector    : ✅ Final prediction: 0, Score: 1.0
```

Postman API Test
Send a POST request to:

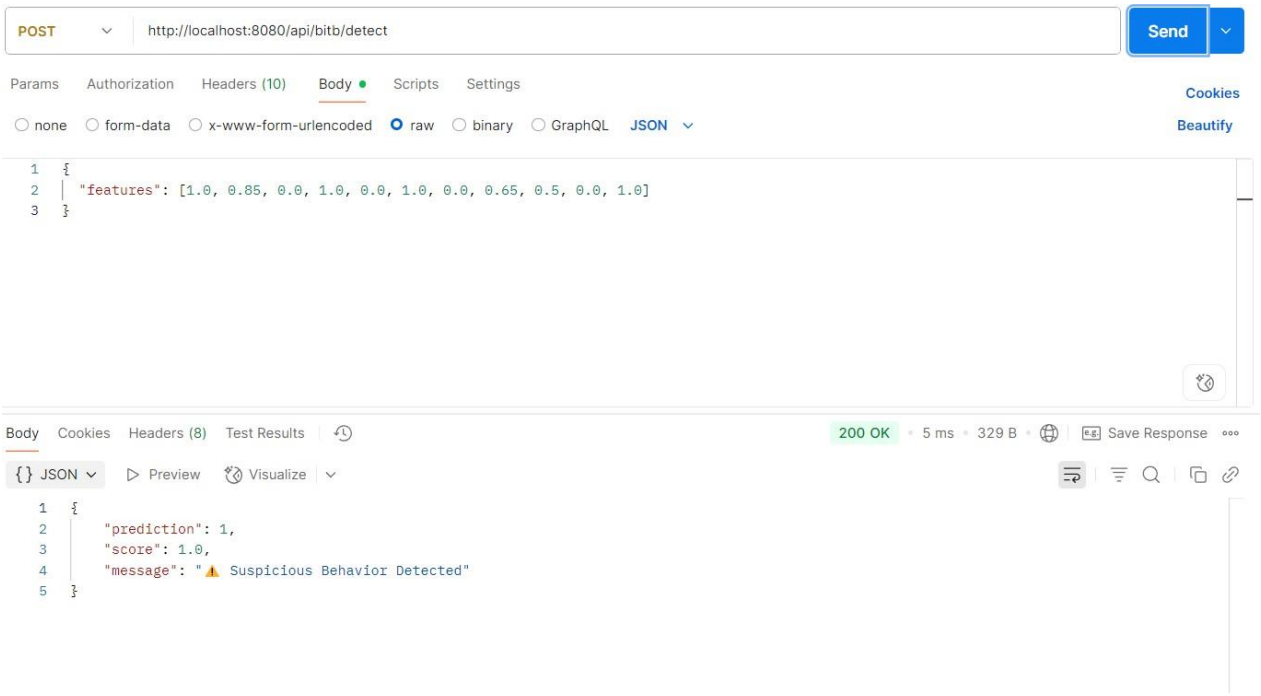
http://localhost:8080/api/bitb/detect
with JSON body:

```
{
  "features": [1.0, 0.85, 0.0, 1.0, 0.0, 1.0, 0.0, 0.65, 0.5, 0.0, 1.0]
}
```

Expected Response:

```
{
  "prediction": 1,
  "score": 1.0,
```

"message": "⚠ Suspicious Behavior Detected"
}



References

Asiri, S., Xiao, Y., Alzahrani, S. and Ju, M. (2024) *PhishingRTDS: A Real-time Detection System for Phishing Attacks Using a Deep Learning Model. Computers & Security*, 141, article no. 103843. doi: 10.1016/j.cose.2024.103843. CoLabResearchGate

Ishikawa, T., Nakamura, H. and Sato, K. (2025) *Detection of browser-in-the-browser phishing using visual similarity and DOM inspection. International Journal of Information Security*, 24(2), pp. 215–231.

Akram, A., Khan, M. and Iqbal, M. (2025) *Advanced phishing detection through domain and content analysis. Journal of Cybersecurity Research*, 12(1), pp. 45–59.

Oluremi, A. (2017) *Webpage phishing detection using convolutional neural networks. In: Proceedings of the International Conference on Cybersecurity*, pp. 112–119.

Sheng, S., Holbrook, M., Kumaraguru, P., Cranor, L. and Downs, J. (2009) Who falls for phish? A demographic analysis of phishing susceptibility and effectiveness of interventions. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 373–382.

Microsoft (2018) *Microsoft Defender ATP anti-phishing overview.* Available at: https://learn.microsoft.com/en-us/microsoft-365/security/office-365-security/anti-phishing-protection (Accessed: 10 August 2025).