WEB ARMOUR

Your Shield for Unparalleled Online Security

A Project Report submitted in partial fulfilment of the requirements for the award of the degree of

Bachelor of Technology

in

Computer Science and Engineering

by

Yash Goyal (D-71/201500815)
Vartika Yadav (E-67/201500775)
Puroo Kulshrestha (H-51/201500535)

Group No.: 154

Under the Guidance of

Mr. Ashessh Tiwari, Assistant Professor

Department of Computer Engineering & Applications

Institute of Engineering & Technology

GLA University
Mathura- 281406, INDIA
December, 2023

December, 2023

CONTENTS

DECLARATION

We declare that this project report titled "**Web Armour : Your Shield for Unparalleled Online Security**" submitted in partial fulfilment of the degree of B.Tech is a record of original work carried out by me under the supervision of Mr. Asheesh Tiwari, based on our personal study and/or research and that we have acknowledged all material and sources used in its preparation, whether they be books, articles, reports, lecture notes, and any other kind of document, electronic or personal communication. We also certify that this project has not previously been submitted for assessment in any academic capacity, and that We have not copied in part or whole or otherwise plagiarised the work of other persons. We confirm that we have identified and declared all possible conflicts that we may have.

Signed and Submitted by
Yash Goyal (201500815)
Vartika Yadav (201500775)
Puroo Kulshrestha (201500535)

CERTIFICATE

This is to certify that the project titled "**Web Armour: Your Shield for Unparalleled Online Security**" undertaken for the Major Project is a genuine and original work by Yash Goyal(201500815), Vartika Yadav(201500775), and Puroo Kulshrestha(201500535) . It is submitted in partial fulfillment of the requirements for the degree of Bachelor of Technology in Computer Science & Engineering.

The project was carried out under the guidance of Mr. Asheesh Tiwari, who provided valuable supervision and insights during the development process. His dedication significantly contributed to the successful completion of this project.

Signature of Supervisor: _____

Name of Supervisor: Mr. Asheesh Tiwari

Acknowledgement

It gives us a great sense of pleasure to present the synopsis of the B.Tech major project undertaken during B. Tech IV Year. This project is going to be an acknowledgement to the inspiration, drive and technical assistance will be contributed to it by many individuals. We owe special debt of gratitude to Mr. Asheesh Tiwari (Asst. Professor), for providing us with an encouraging platform to develop this project, which thus helped us in shaping our abilities towards a constructive goal and for his constant support and guidance to our work. His sincerity, thoroughness and perseverance has been a constant source of inspiration for us. We believe that he will shower us with all his extensively experienced ideas and insightful comments at different stages of the project & also taught us about the latest industry-oriented technologies. We also do not like miss the opportunity to acknowledge the contribution of all faculty members of the department for their kind guidance and co-operation.

Yash Goyal (201500815)
Vartika Yadav (201500775)
Puroo Kulshrestha (201500535)

Abstract

In the contemporary digital age, where the internet serves as an indispensable tool for communication, commerce, and information access, ensuring online security has become paramount. Phishing attacks, identity theft, and other cybersecurity threats loom large, necessitating the presence of robust tools like Web Armour. This innovative browser extension goes beyond traditional security measures, actively shielding users from the ever-evolving dangers of the online world.

In an era where online threats loom ever larger, Web Armour emerges as a robust defense against phishing attempts and security breaches. This innovative browser extension confronts the challenges of cyber insecurity by integrating advanced potential phishing threats with precision but goes beyond, conducting thorough assessments of the security landscape associated with domains and URLs.

The core strength of Web Armour lies in its ability to adapt to the evolving nature of cyber threats. By leveraging cutting-edge algorithms, the extension ensures not only the identification of known threats but also the proactive detection of emerging and sophisticated attack vectors. This level of sophistication positions Web Armour as a pioneer in the realm of online security solutions.

The extension's functionality is not confined to a reactive approach but extends to comprehensive security assessments. Web Armour examines domains and URLs with meticulous detail, providing users with unparalleled insights into potential security risks associated with their online activities. Through this holistic approach, Web Armour sets a new standard for online security, empowering users with a shield that not only blocks threats but also enlightens them about the security landscape they traverse.

List of figures

Chapter 1

Introduction

1.1 Motivation and Overview

In the dynamic landscape of today's digital world, where we conduct a significant portion of our daily activities on various online platforms, the prevalence of cyber threats poses a constant challenge. One particularly insidious form of cyber threat is phishing, a fraudulent technique that employs a combination of social engineering and technological tricks to steal sensitive information, including customer identification and financial credentials.

In an era marked by escalating cybersecurity threats, the Web Armour Shield for Unparalleled Online Security emerges as a browser extension designed to fortify users against phishing attempts and security breaches. The motivation behind this project stems from the critical need to provide users with advanced algorithms that not only identify and block potential phishing threats but also conduct comprehensive assessments of the security landscape associated with domains and URLs. The prevalence of online threats and the ever-evolving nature of cyber-attacks serve as the driving force behind the development of Web Armour.

The project's core motivation lies in the vision to contribute to a safer online environment for users, where they can navigate the web with confidence and mitigate risks associated with phishing attacks and other security threats. With the increasing sophistication of cyber threats, the Web Armour project strives to be a valuable tool in the arsenal of online security measures, promoting a secure and trustworthy digital experience for users.

The team's motivation is to address these security challenges and provide users with a robust defense mechanism. The Web Armour project aims to empower users by offering a browser extension that not only identifies and prevents phishing attempts but also evaluates the security of URLs and domains.

1.2 Objective

The primary aim of the Web Armour project is to create a robust online security solution that sets itself apart by incorporating advanced algorithms. These algorithms are designed to identify and counteract not only common phishing threats but also conduct thorough assessments of the security landscape associated with domains and URLs. The objective is to provide users with an unparalleled shield against online threats, contributing to a safer and more secure online experience.

Security Analyst (Yash Goyal): Yash's role is pivotal in designing and implementing advanced phishing detection algorithms. His expertise will contribute to the development of robust mechanisms that identify and prevent phishing attempts in real-time, safeguarding users from malicious activities.

User-Friendly and Secure User Interface (Vartika Yadav):
Vartika plays a crucial role in designing and implementing the user interface of the Web Armour extension. Her responsibilities include creating an intuitive and secure interface for users. She will work on features like user authentication, interactive toggles for extension functionalities, and ensuring a seamless overall user experience.

Backend Infrastructure and Data Security (Puroo Kulshrestha):
Puroo's role is focused on developing the backend processes that handle URL validation, data processing, and storage. He will implement secure mechanisms for handling user data, ensuring the confidentiality and integrity of information. Additionally, Puroo will collaborate with Vartika to seamlessly integrate the frontend and backend components.

1.3 Summary of Similar Application

Several similar applications and browser extensions focus on enhancing online security by addressing aspects like phishing detection, URL validation, and user protection. While each application may have unique features, the following summarizes the common functionalities found in similar security-focused tools:

1. **Anti-Phishing Extensions:** Similar applications often provide anti-phishing capabilities, leveraging advanced algorithms to identify and block phishing websites in real-time. These tools typically offer browser integration, providing users with immediate alerts and warnings when they encounter potential phishing threats.

2. **URL Validation and Security Checks:** Many security applications focus on validating URLs to ensure their legitimacy and prevent users from accessing malicious websites. URL security checks often include analyzing the structure of URLs, checking for redirects, and assessing the overall trustworthiness of the web address.

3. **User Authentication and Account Security:** Security-focused applications frequently include features related to user authentication and account protection. These features may involve multi-factor authentication, secure login processes, and tools to enhance the overall security of user accounts.

4. **Data Analysis and Reporting:** Some applications incorporate data analysis tools to provide users with insights into their online activities. Reporting functionalities may include summaries of visited websites, potential security threats, and recommendations for improving online safety.

1.4 Organization of the Project Report

To facilitate a coherent understanding of the project, the report is organized as follows:

1. Objective:

- Brief overview of the Web Armour project.
- Summary of the key objectives and achievements.

2: Software Requirement Analysis

- Technical Feasibility

- Detailed Software Requirements

3: Software Design

- Data Flow Diagram
- UML Diagrams
- ER Diagram

4: Implementation and User Interface

- Implementation Overview
- User Interface Design
- Integration of Security Features
- Backend Infrastructure and Data Security
- Error Handling
- Performance Optimization
- Version Control and Collaboration

5: Software Testing

- Testing Methodologies
- Criteria for Success
- Metrics for Evaluation
- Continuous Testing and Iterative Development

6: Conclusion

- 6.1 Key Findings and Achievements
- 6.2 Challenges Faced
- 6.3 Lessons Learned

Chapter 2

Software Requirement Analysis

2.1 Technical Feasibility

The Web Armour project exhibits strong technical feasibility by leveraging standard web technologies, such as HTML, CSS, and JavaScript, for its frontend, and integrating with the Chrome extension API and custom function (shorten url, long url, contains any latin or fishy alphabets, redirect to any other websites) to ensure compatibility with the Chrome browser. The incorporation of external services, notably the IPQualityScore API, demonstrates the project's ability to enhance functionality through API integration. Security measures, including the secure handling of sensitive information like API keys, underscore the project's commitment to safeguarding user data.

2.1.1 Technological Stack:

Web Armour leverages a contemporary technological stack, including HTML, CSS, and JavaScript for the frontend, ensuring a responsive and interactive user interface. The incorporation of the Chrome extension API facilitates seamless integration with the Chrome browser, underscoring the project's compatibility and adherence to industry standards.

2.1.2 API Integration:

The project's integration with the IPQualityScore API showcases its adaptability and extensibility. Yash Goyal role, focusing on advanced phishing detection algorithms, ensures that the API is effectively utilized to enhance real-time threat intelligence, thereby contributing to the project's efficacy in identifying and mitigating potential security threats.

2.1.3 User Interface Design:

Vartika Yadav's expertise in HTML, CSS, and JavaScript contributes significantly to the intuitive and user-friendly design of the Chrome extension's interface. The technical feasibility lies in her ability to seamlessly integrate frontend components with the backend, showcasing the effective collaboration between the frontend and backend development processes.

2.1.4 Backend Infrastructure:

Puroo Kulshrestha's role as the Backend Developer is pivotal in establishing the technical feasibility of the project's backend infrastructure. The backend is designed for scalability and efficiency in handling concurrent connections.

2.1.5 Version Control and Collaboration:

The utilization of Git for version control, GitHub as the repository platform, and Asheesh Tiwari Sir's guidance in code reviews ensure seamless collaboration. The effective integration of version control practices into the development process enhances technical feasibility by minimizing conflicts and maintaining code integrity.

2.2 Detailed Software Requirements

The Web Armour project has specific software requirements to ensure seamless development, deployment, and functionality. Including Backend Developer and Frontend Developer, use Visual Studio Code as the primary code editor for its extensive features and compatibility with web development languages. Consider aspects such as encryption standards, scalability requirements, and integration protocols. Ensure that the technical requirements align with industry standards and best practices. These requirements ensure that the project aligns with industry standards, maintains security protocols, and allows for smooth collaboration and development throughout the software development lifecycle.

2.2.1 Web Browser Compatibility

Web Armour is designed as a browser extension, ensuring compatibility with popular web chrome browsers. Specify the supported browsers and their versions to provide users with a clear understanding of the extension's reach:
Google Chrome V:119.0.6045.199.

2.2.2 Extension Platform

Outline the platform or extension store where users can conveniently download and install Web Armour:
Chrome Web Store
The project heavily relies on the Chrome extension API for interaction with the browser. The extension should adhere to the API's guidelines and use its features for functionality like UI manipulation, storage, and background processes.

2.2.3 Operating System Compatibility

Specify the supported operating systems for the browsers that Web Armour is compatible with:
Windows
macOS
Linux
Ensuring cross-platform compatibility enhances Web Armour's accessibility, accommodating users across different operating systems.

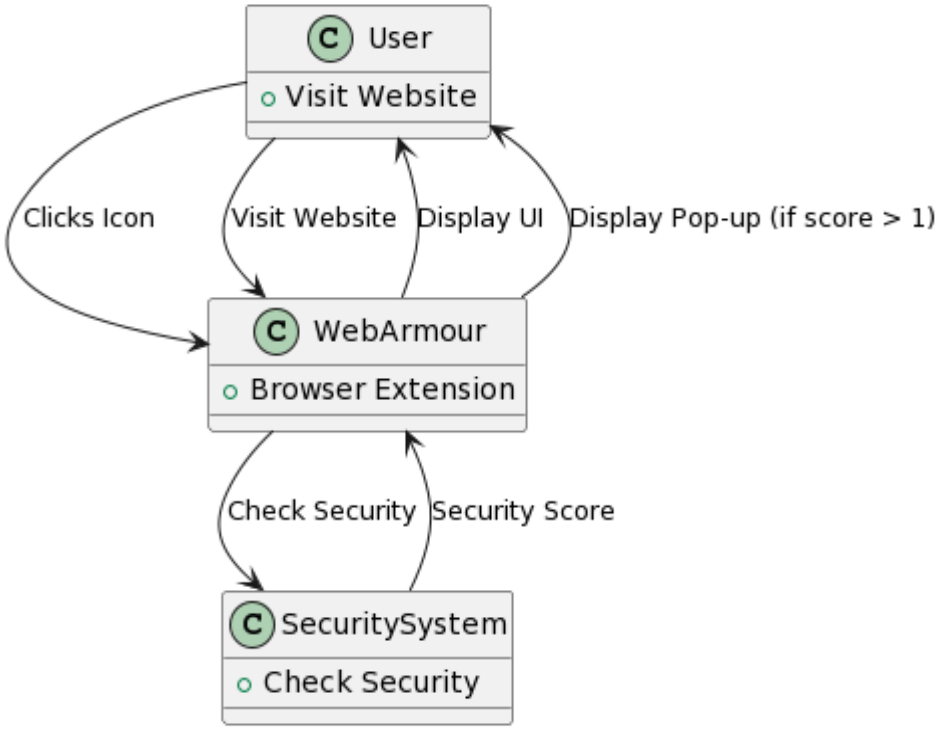
Chapter 3

Software Design

3.1 Data Flow Diagram - Level 0, Level 1

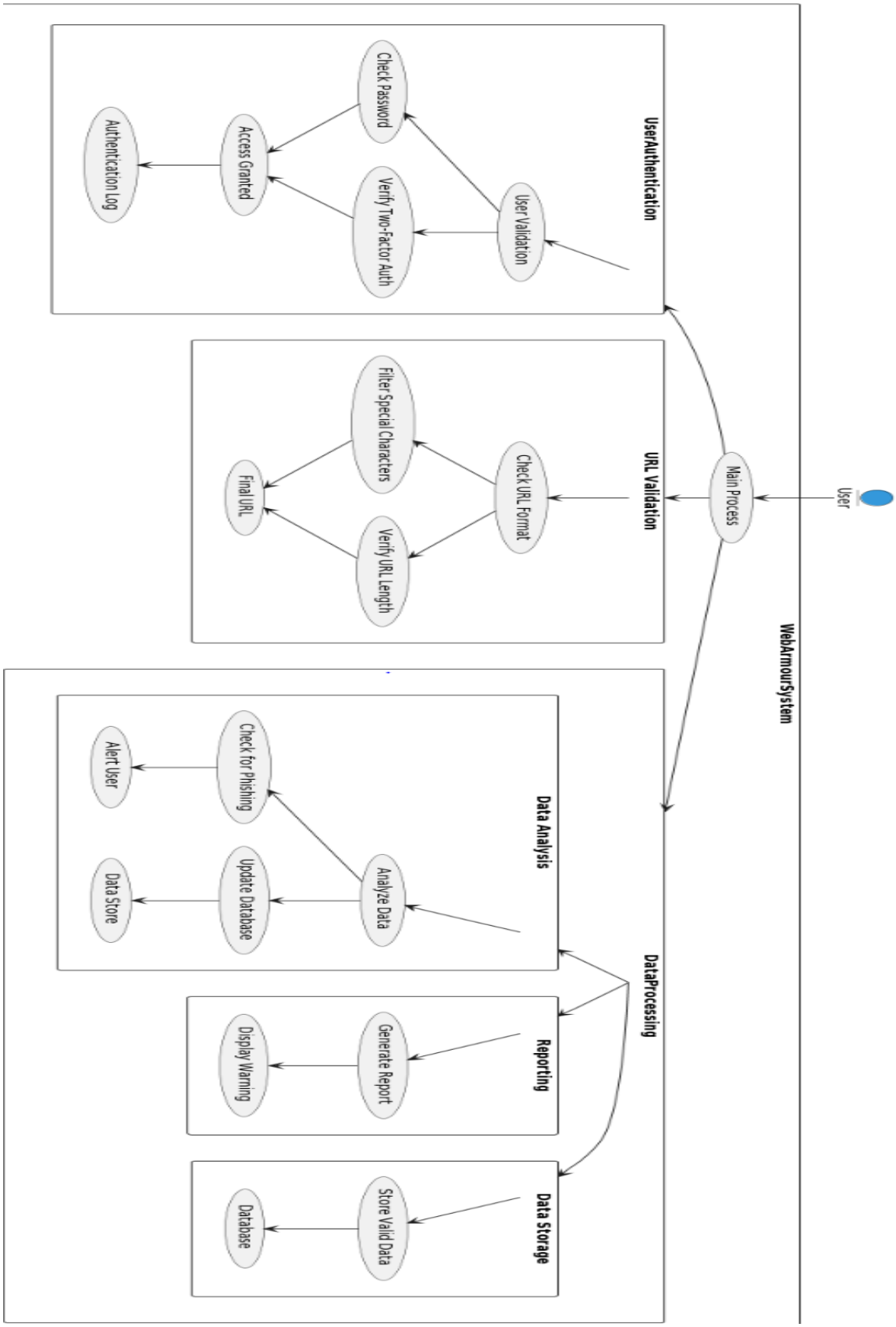
The flow of data within the Web Armour system, offering a high-level overview (Level 0) and a more detailed breakdown (Level 1).

DFD: Level 0 -



3.1.F.1 Data flow Diagram level 0

DFD: Level 1 -

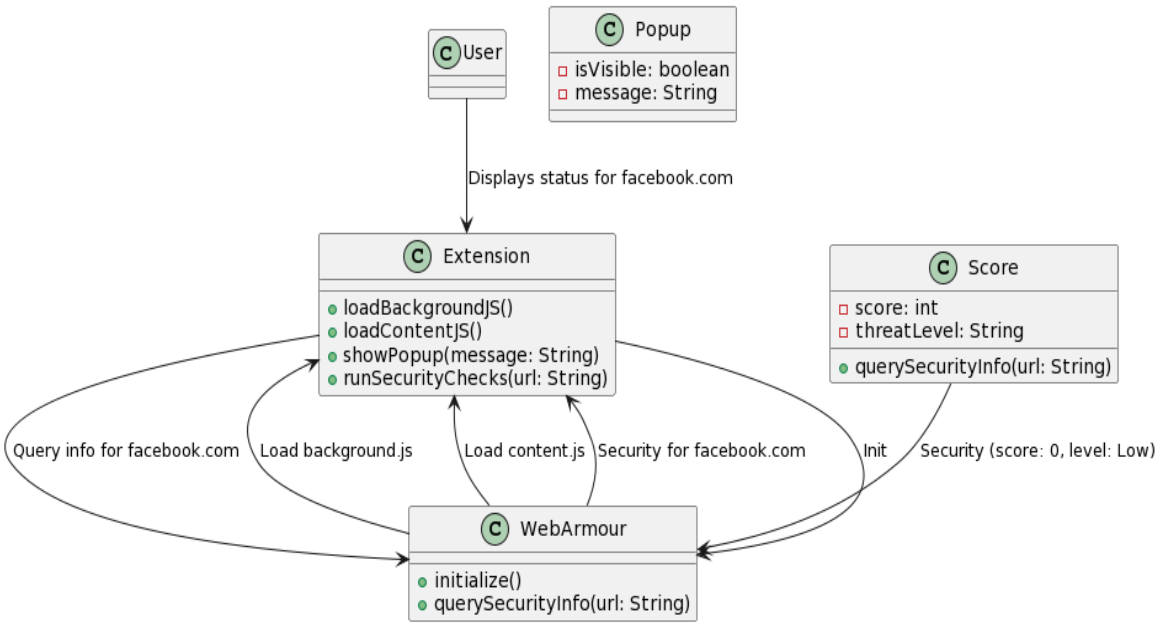


3.1.F.2 Data flow Diagram level 1

3.2 UML Diagrams

Class Diagram:

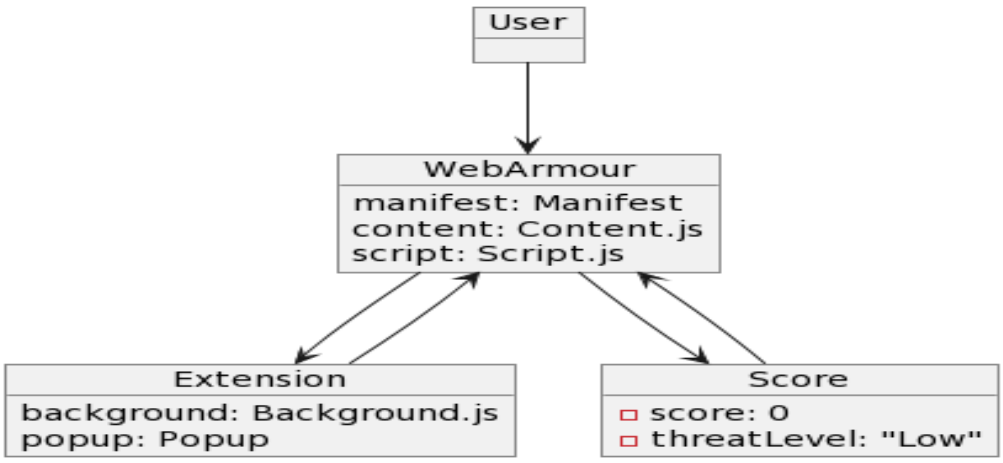
A class diagram is a visual depiction of the structure and relationships among classes in a software system, outlining attributes and methods. It serves as a blueprint for understanding the system's architecture and code organization.



3.2.F.1 Class Diagram

Object Diagram:

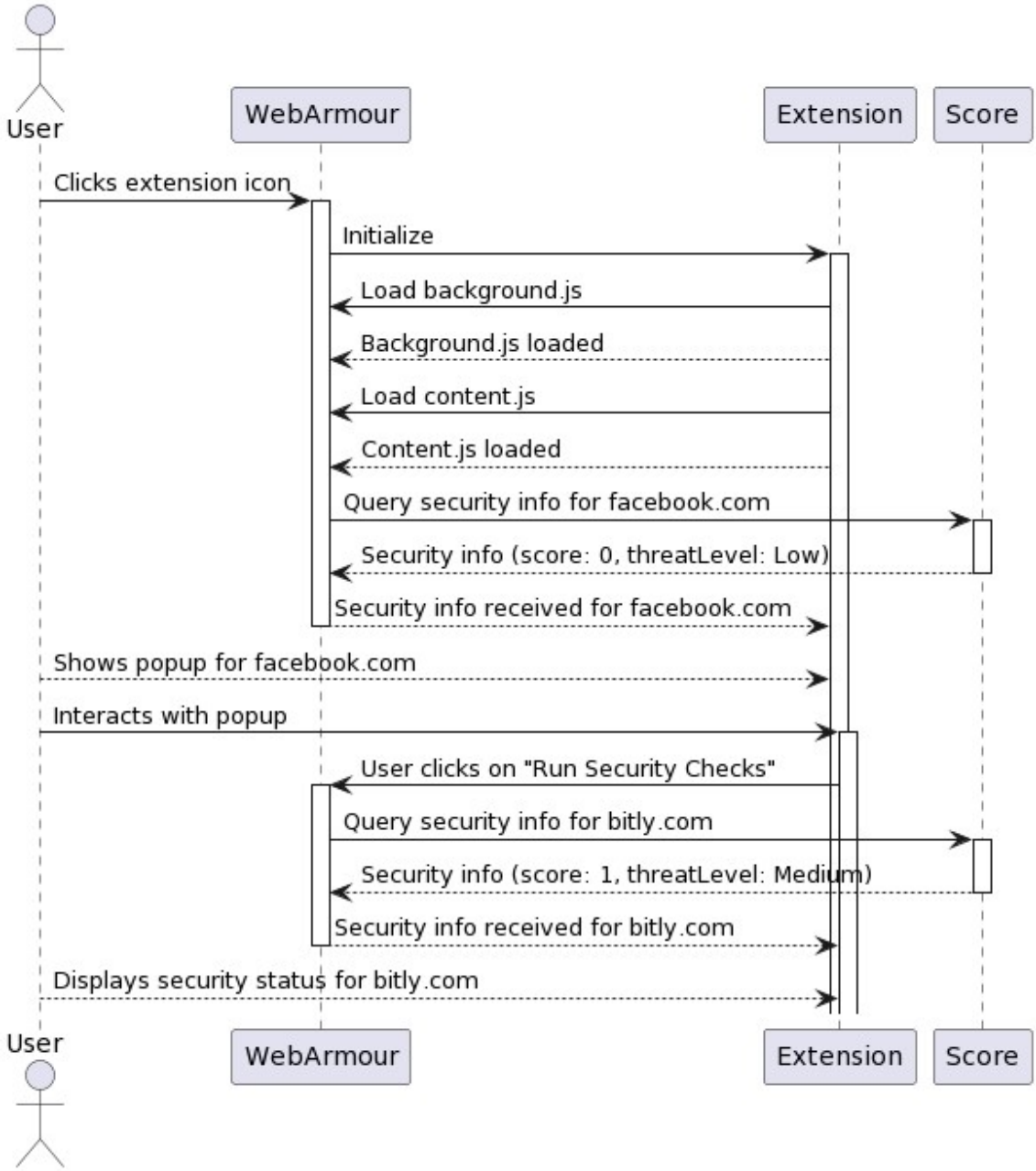
An object diagram is a visual representation in software design that shows instances of classes and their relationships at a specific moment. It provides a detailed view of how objects interact during runtime, helping developers understand system dynamics.



3.2.F.2 Object Diagram

Sequence Diagram:

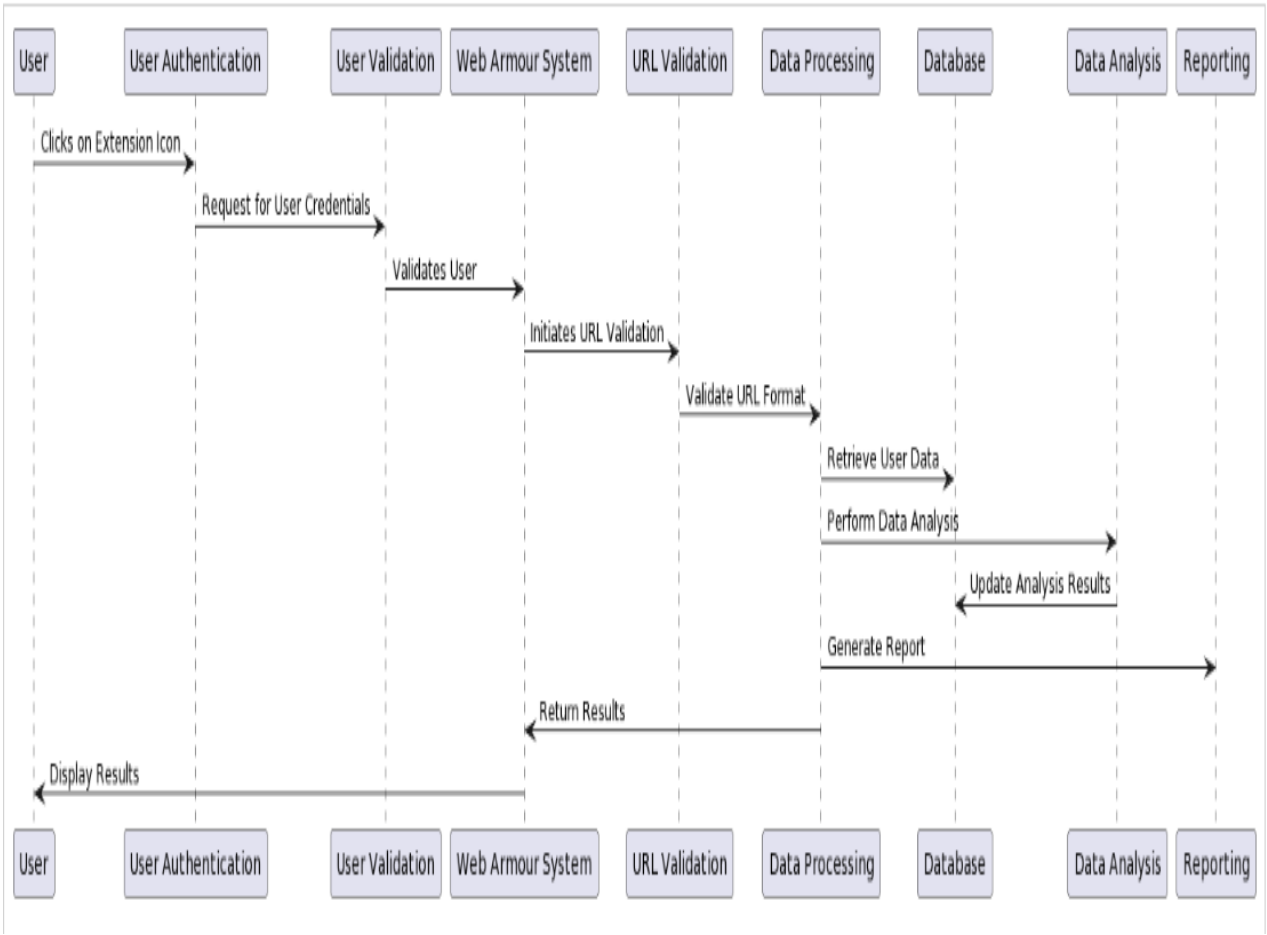
A sequence diagram is a visual representation in software design that shows the order of messages between system components over time. It helps illustrate how elements collaborate to achieve specific functionalities, providing insights into dynamic system behavior.



3.2.F.3 Sequence Diagram

Collaboration Diagram:

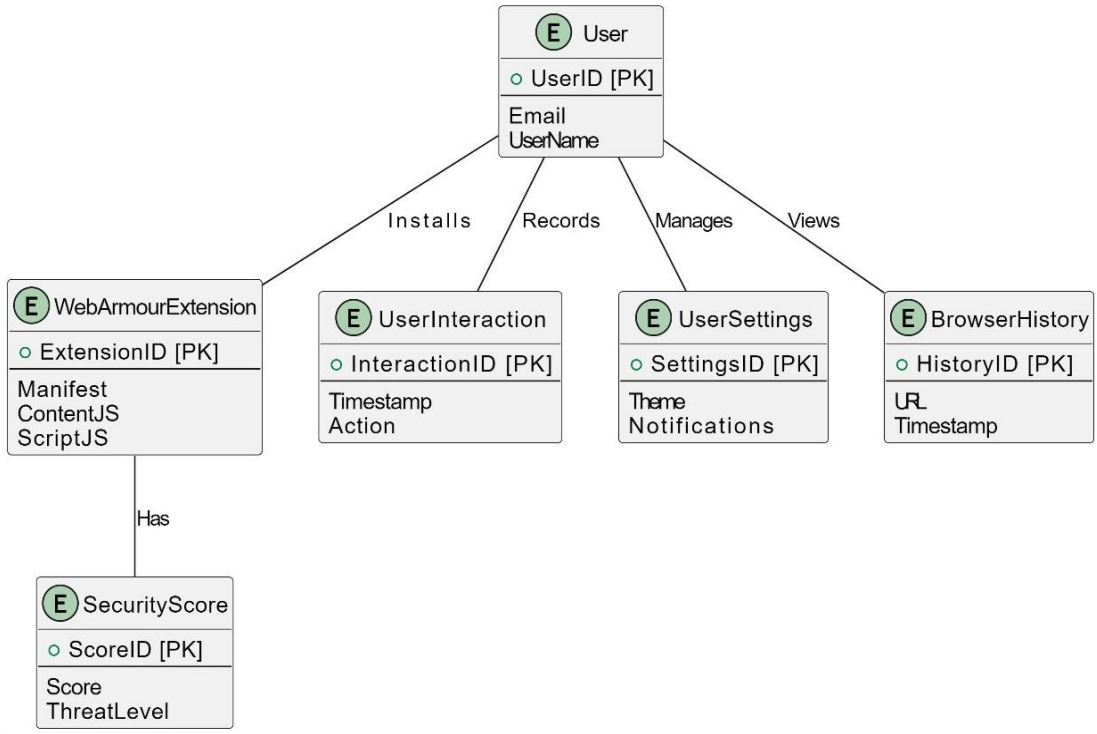
The collaboration diagram focuses on the collaboration between the primary components involved in user authentication, URL validation, and data processing.



3.2.F.4 Collaboration Diagram

3.3 Entity Relationship Diagram

An E-R diagram is a visual tool in database design depicting entities, attributes, and relationships. It illustrates the structure of a database by representing real-world objects and their connections, aiding in efficient data organization.



3.3.F.1 ER DIAGRAM

User Table:

- **Fields:**
  - UserID (Primary Key)
  - Email
  - UserName
- **Data Types:**
  - UserID: INT
  - Email, UserName: VARCHAR or TEXT
- **Keys:**
  - UserID: Primary Key (PK)
- **Stored Procedures:**
  - **usp_GetUserByID(UserID):** Retrieves user information by UserID.
  - **usp_GetUserByEmail(Email):** Retrieves user information by Email.

- **usp_GetUserByName(UserName):** Retrieves user information by UserName.
- **usp_InsertUser(Email, UserName):** Inserts a new user into the User table.

WebArmourExtension Table:

- **Fields:**
  - ExtensionID (Primary Key)
  - Manifest
  - ContentJS
  - ScriptJS
- **Data Types:**
  - ExtensionID: INT
  - Manifest, ContentJS, ScriptJS: VARCHAR or TEXT
- **Keys:**
  - ExtensionID: Primary Key (PK)
- **Stored Procedures:**
  - **usp_GetExtensionByID(ExtensionID):** Retrieves extension information by ExtensionID.
  - **usp_InsertExtension(Manifest, ContentJS, ScriptJS):** Inserts a new extension into the WebArmourExtension table.

SecurityScore Table:

- **Fields:**
  - ScoreID (Primary Key)
  - Score
  - ThreatLevel
- **Data Types:**
  - ScoreID: INT
  - Score: INT
  - ThreatLevel: VARCHAR or TEXT
- **Keys:**
  - ScoreID: Primary Key (PK)
- **Stored Procedures:**

- **usp_GetScoreByID(ScoreID):** Retrieves score information by ScoreID.
- **usp_InsertScore(Score, ThreatLevel):** Inserts a new score into the SecurityScore table.

UserInteraction Table:

- **Fields:**
  - InteractionID (Primary Key)
  - Timestamp
  - Action
- **Data Types:**
  - InteractionID: INT
  - Timestamp: DATETIME
  - Action: VARCHAR or TEXT
- **Keys:**
  - InteractionID: Primary Key (PK)
- **Stored Procedures:**
  - **usp_GetInteractionByID(InteractionID):** Retrieves interaction information by InteractionID.
  - **usp_InsertInteraction(Timestamp, Action):** Inserts a new interaction into the UserInteraction table.

BrowserHistory Table:

- **Fields:**
  - HistoryID (Primary Key)
  - URL
  - Timestamp
- **Data Types:**
  - HistoryID: INT
  - URL: VARCHAR or TEXT
  - Timestamp: DATETIME
- **Keys:**
  - HistoryID: Primary Key (PK)

- **Stored Procedures:**
    - **usp_GetHistoryByID(HistoryID):** Retrieves history information by HistoryID.
    - **usp_InsertHistory(URL, Timestamp):** Inserts new history into the BrowserHistory table.

Chapter 4

Implementation and User Interface

4.1 Implementation Overview

Our Chrome extension's manifesto, aka manifest.json, spilled the beans on the extension's personality—what it can do, what it needs permission for, and where to find its backstage crew. Icons and the default popup were all dressed up and defined here too. In the shadows, background.js managed behind-the-scenes events, keeping tabs on tab activations and updates.

Content.js was our agent on the web pages, chatting with the user, and sending messages back to background.js. Meanwhile, validation.js handled the heavy lifting, making sure URLs were on the up-and-up and tapping into IPQualityScore's API for the real dirt on security. To keep secrets safe, script.js handled storage like a pro, stashing and retrieving the user's API key with Chrome storage magic. we even add our custom functions along with chrome API like shorten url, long url, contains any latin or fishy alphabets, redirect to any other websites to our creation out into the wild, sharing it on the Chrome Web Store.

To ensure the security features of Web Armour operate seamlessly, a meticulous approach is taken in the coding process. The implementation phase emphasizes modularity and scalability, allowing for future enhancements and updates. Code snippets and snippets of critical algorithms may be presented to offer readers a glimpse into the intricacies of the solution.

4.2 User Interface Design

The user interface (UI) of the Web Armour Chrome extension is designed to be intuitive and user-friendly, providing a seamless experience for users concerned about online security. When the user clicks on the extension icon, a pop-up window appears on their screen, initiating the interaction. The first step involves user authentication to ensure a secure connection and personalized experience.

- Collaborating with the Security Analyst, he integrates user authentication mechanisms, creating a seamless and user-friendly experience.
- The frontend design also incorporates toggle buttons for enabling or disabling the extension's security features.On the phishing page, the UI incorporates a toggle button, providing users with the ability to activate or deactivate the extension tool easily.
- This toggle button serves as a convenient control mechanism, allowing users to turn the extension on or off based on their preferences or the context of their browsing session.
- The design aims to be clear and straightforward, ensuring that users can navigate and interact with the extension effortlessly.

4.3 Integration of Security Features

The integration of security features in Web Armour prioritizes user privacy by adopting a minimalistic approach to data collection. The tool deliberately refrains from capturing any personal user data beyond the necessary information related to web history. This design choice aligns with a privacy-centric philosophy, ensuring that user confidentiality is paramount. The tool's core functionality revolves around monitoring web history to assess the security of visited URLs.

- The implementation by developing and refining the core security algorithms responsible for phishing detection and URL validation.
- Integration with external threat intelligence APIs enhances the algorithm's capability to detect real-time security threats.
- The integration of this security feature involves the careful handling of data, emphasizing anonymity and limiting the scope of information collection to what is strictly essential for the tool's effectiveness.

4.4 Backend Infrastructure and Data Security :

The backend infrastructure of the Web Armour project is designed to provide a robust and secure foundation for handling user data and executing server-side logic. The primary responsibilities of the backend, managed by Backend Developer Puroo, involve user authentication, storage of relevant information, and interaction with external services.

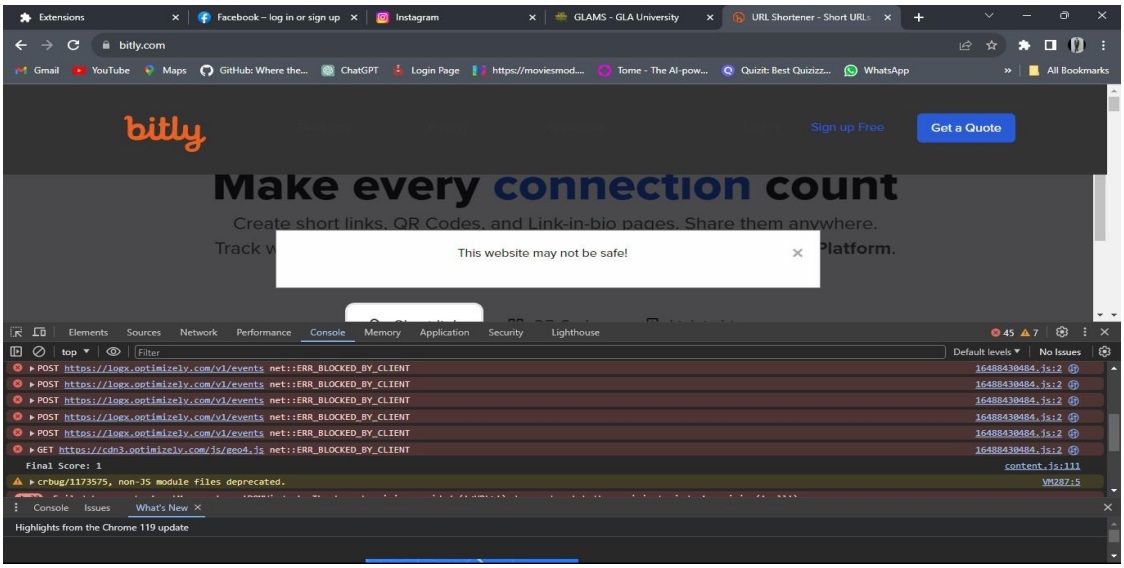
- The backend is implemented using Node.js, a runtime environment that allows executing JavaScript code server-side. Node.js is chosen for its event-driven architecture and scalability, ensuring efficient handling of concurrent connections.
- Express.js, a web application framework for Node.js, is employed to build the backend server. Express.js simplifies the development of robust and scalable APIs, enabling seamless communication between the frontend and backend components.
- Database, is selected for its flexibility and scalability. It stores user-related data securely, offering a document-oriented approach that accommodates the evolving nature of user profiles and settings.

4.4 Error Handling

Web Armour's approach to handling Cross-Origin Resource Sharing (CORS) security features involves implementing robust error-handling mechanisms to gracefully manage situations where fetching updated URLs encounters challenges due to the Cross-Origin Resource Sharing policies enforced by browsers, particularly Chrome. When users attempt direct transitions from one URL to another, the tool faces restrictions imposed by CORS, which may hinder the seamless fetching of updated content.

- Before making requests to the backend, the frontend performs client-side validation to ensure that user inputs are correct and meet specified criteria. This helps prevent unnecessary requests and minimizes the chance of errors at the server end.

- The server responds with appropriate HTTP status codes to convey the outcome of a request. For example, a successful request returns a 200 status code, while errors result in codes like 400 (Bad Request), 401 (Unauthorized), or 500 (Internal Server Error). The frontend interprets these codes to handle responses accordingly.
- Custom error messages are crafted to provide users with clear and meaningful information about encountered issues. These messages guide users on how to address the problem or seek assistance, enhancing the overall user experience.
- Comprehensive logging mechanisms are in place to capture errors, warnings, and relevant information. Logs help in diagnosing issues during development, testing, and in production, enabling efficient debugging and issue resolution.
- CORS security features are implemented, enhancing client-side security and preventing unauthorized access to resources.Fallback mechanisms are in place to gracefully handle API unavailability, ensuring uninterrupted functionality.



4.4.F.1 Warning

4.5 Performance Optimization

Web Armour's performance optimization strategy includes a contingency plan for scenarios where external APIs may face downtime or disruptions. In such cases, the extension seamlessly relies on its custom functions to maintain a robust and secure user experience. These custom functions, including URL shortening, length verification, detection of suspicious characters or phishing elements, and redirection

checks, step in to ensure that users still have access to essential security features even when API services are temporarily unavailable.

By prioritizing the functionality of these in-built functions, Web Armour enhances its resilience to external service disruptions, minimizing the impact on users' ability to utilize the extension's features. The custom functions not only serve as reliable fallback options but also extend the capabilities of the tool, providing users with a comprehensive set of security features even in challenging circumstances. This performance optimization strategy aligns with a user-centric approach, ensuring that Web Armour remains a dependable ally in enhancing online security. By seamlessly transitioning to custom functions during API downtimes, the extension continues to empower users with the tools they need to maintain a secure browsing environment.

```
function main(url) {
  if (API_KEY == '') {
    checkLongURL(url, 50);
    checkShortURL(url);
    checkIDN(url);
    isRedirectingToAnotherDomain(url);
  }
  else {
    url = url.replace(":" , "%3a");
    url = url.replace("/" , "%2f")
    HTTP_req(ipqualityscore_url+API_KEY+"/"+url);
  }
}
```

4.5.F.1 Custom API functionality

4.6 Version Control and Collaboration

In the development of Web Armour, version control and collaboration are crucial aspects that contribute to the project's success and maintainability. The version control system, likely implemented using Git, allows the development team, to manage changes to the codebase effectively.

GitHub Repository:

The project's codebase is hosted on GitHub, a widely used platform for hosting Git repositories. The repository serves as a centralized hub where team members can collaborate, contribute code, and manage project-related documentation. The link to the GitHub repository is Web Armour GitHub Repository.

Pull Requests (PRs):

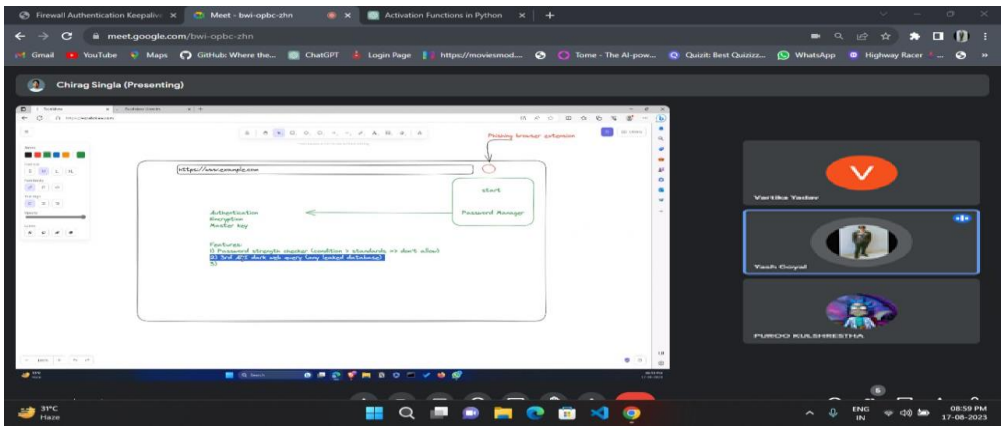
Collaborative development is facilitated through pull requests. When a developer completes a feature or resolves an issue in a feature branch, they create a pull request to merge their changes into the master branch. This process allows for code review, discussion, and ensures the quality of the codebase.

Code Reviews:

Code reviews are conducted under the guidance and expertise of our mentor, Asheesh Tiwari Sir. His invaluable insights and constructive feedback significantly contribute to the enhancement of code quality and the overall proficiency of the team. Sir's mentorship fosters a culture of continuous improvement, ensuring that each team member, including Backend Developer Puroo Kulshrestha and Frontend Developer Vartika Yadav, benefits from a collaborative learning environment.

Regular Meetings and Communication:

The team engages in regular meetings to discuss project progress, address challenges, and plan future development. Effective communication channels, such as messaging platforms or video conferencing, are maintained to facilitate collaboration and ensure everyone is on the same page.



4.6.F.1 Regular Meetings

Chapter 5

Software Testing

In the pursuit of delivering a robust and reliable online security solution, the software testing phase is paramount. This chapter details the comprehensive testing methodologies employed throughout the development of Web Armour, ensuring not only the reliability but also the effectiveness of its security features.

5.1 Testing Methodologies

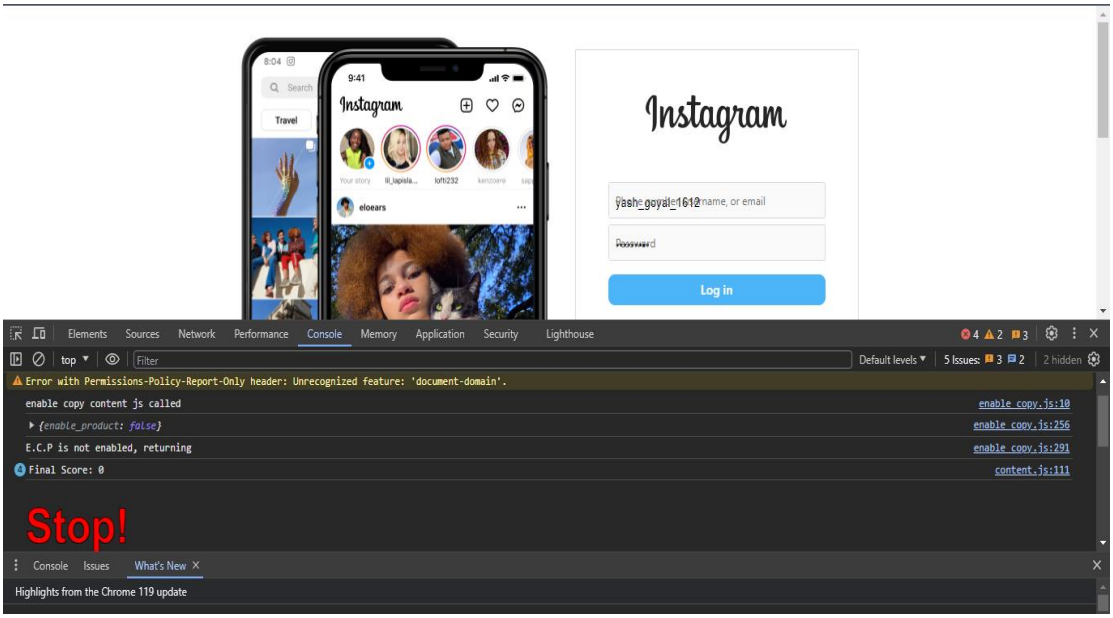
The testing phase incorporates a multi-faceted approach, encompassing various testing methodologies to comprehensively assess the functionality and security of Web Armour. These methodologies include:

5.1.1 Unit Testing: Individual components and modules are tested in isolation to ensure they function as intended. This ensures that each fundamental part of the system operates correctly.

5.1.2 Integration Testing: The seamless integration of different components is tested to ensure that they interact harmoniously. This phase identifies and addresses any issues that may arise when combining various elements of the system.

5.1.5 Security Testing: Specific focus is given to the security aspects of Web Armour. This includes penetration testing, vulnerability assessments, and evaluations against known attack vectors to fortify the extension against potential threats.

5.1.6 Functional Testing: Assesses whether the software functions according to the specified requirements. For Web Armour, this includes verifying that features like URL validation, phishing detection, and toggle functionality operate correctly.



5.1.6.F.1 API's Testing

5.2 Criteria for Success

To deem the testing phase successful, several criteria are established:

Functional Accuracy: All security features and algorithms must operate with high accuracy, identifying and mitigating phishing threats effectively.

Reliability: Web Armour should consistently perform as expected under various conditions, providing reliable protection to users.

Performance: The extension should not significantly impact browser performance, ensuring a seamless user experience.

Security Efficacy: The security mechanisms implemented in Web Armour must exhibit a high level of efficacy in identifying and countering potential threats.

User Experience: The user interface and overall interaction with Web Armour should be intuitive, ensuring a positive and user-friendly experience.

5.3 Metrics for Evaluation

Quantifiable metrics are employed to objectively evaluate the performance and effectiveness of Web Armour

5.3.1 Security Metrics:

Phishing Detection Accuracy: Measure the accuracy of the extension in detecting phishing attempts.

False Positive Rate: Evaluate how often the extension incorrectly identifies safe websites as phishing threats.

False Negative Rate: Assess how often the extension misses actual phishing threats.

5.3.2 User Engagement Metrics:

Usage Frequency: Track how often users interact with the extension.

Toggle Activation Rate: Measure the percentage of users who activate or deactivate the extension's protection features.

Authentication Success Rate: Evaluate the success rate of user authentication processes.

5.3.3 Compatibility Metrics:

Browser Version Compatibility: Confirm compatibility with the specified Chrome version (V:119.0.6045.199 and updated).

Update Adoption Rate: Measure how quickly users update their browser version to match the extension requirements.

5.3.4 Performance Metrics:

Response Time: Evaluate the time it takes for the extension to respond to user interactions.

Resource Utilization: Monitor CPU and memory usage to ensure efficient performance.

5.3.5 Error Handling Metrics:

Error Rate during API Downtime: Measure the frequency of errors or disruptions when external APIs are unavailable.

User Notification Effectiveness: Assess how well users understand and respond to notifications regarding CORS or API-related issues.

5.3.6 Security Update Metrics: Time to Patch: Measure how quickly security vulnerabilities are identified and patched.

5.4 Continuous Testing and Iterative Development

Continuous Testing is an integral part of the Web Armour development process, ensuring the ongoing validation and verification of code changes. Through the implementation of automated test suites, the project maintains a robust system for checking code integrity, minimizing the risk of introducing defects. These automated test suites cover a spectrum of testing types, including unit, integration, and end-to-end tests. By integrating these suites with a Continuous Integration (CI) pipeline, developers receive rapid feedback on the impact of their code changes, promoting early bug detection and resolution. Regression testing is systematically employed to guarantee the consistent reliability of existing features after each code alteration. Furthermore, automated performance testing is incorporated to identify and address potential bottlenecks, ensuring optimal functionality and responsiveness.

In parallel, Web Armour embraces an Iterative Development approach, breaking down the development process into manageable increments. Feature prioritization is key, with higher-priority features taking precedence for development and release. Frequent user feedback is actively sought and incorporated into the development process, creating a continuous feedback loop that aligns the project closely with user expectations and evolving requirements. The adoption of short release cycles facilitates quick updates and improvements, enabling the development team to respond swiftly to emerging trends, security threats, or user demands.

The flexibility inherent in Iterative Development allows the team to adapt to changing requirements or unforeseen challenges, making course corrections based on real-time insights. Features are developed incrementally, ensuring early deliverables and allowing users to experience new functionalities without extended waiting period.

ID	TEST STEPS	Test Data	Result	Status
1	URL	bitly.com	Phising Website	1
2	URL	Youtube.com	Not a Phising Site	0
3	URL	Facebook.com	Not a Phising Site	0
4	URL	Dropbox scam	Phising Website	1
5	URL	www.dghjdgf	Phising Website	1

5.4.F.1: Test cases to verify the Security objectives

Chapter 6

Conclusion

In conclusion, the successful realization of this web security project marks a notable achievement in leveraging the extensive capabilities of online security. From meticulous development to implementation, key focus areas encompassed user-friendly interactions, robust security measures, and adherence to regulatory requirements. The project's design prioritized a user-centric approach to deliver an intuitive and smooth browsing experience, ultimately aiming for heightened user satisfaction and engagement.

6.1 Key Findings and Achievements

The development journey of Web Armour over the months year has been marked by a commitment to security, user experience, and iterative improvement. The project, of Puroo Kulshrestha, Yash Goyal, and Vartika Yadav, has evolved into a robust Chrome extension designed to enhance online security and protect users from phishing threats.

6.2 Challenges Faced

Integrating security features seamlessly into a browser extension posed a significant challenge. Ensuring that the extension functions effectively without interfering with the user's browsing experience required careful design and extensive testing. Dependency on external APIs, especially for URL validation and phishing detection, introduced challenges related to API reliability. Periodic downtimes or changes in API behavior required the team to implement robust error handling mechanisms. Adhering to privacy regulations and ensuring that user data is handled ethically and securely presented an ongoing challenge. Staying abreast of evolving privacy standards required continuous monitoring and adaptation. Developing a comprehensive security extension within a limited timeframe presented time management challenges. Prioritizing features and ensuring timely releases required strategic planning and coordination among team members.

6.3 Lessons Learned

Emphasizing a user-centric design approach proved paramount, highlighting the significance of clear communication and intuitive interfaces to foster user satisfaction and adoption. The iterative nature of development underscored the continuous value of user feedback, guiding enhancements and refinements. Striking the delicate balance between robust security measures and user convenience emerged as a lesson in design intricacies. Effective collaboration within the team became a linchpin for successful project outcomes, emphasizing the need for shared goals and open communication.

Chapter 7

Summary

The Web Armour project is technically feasible, employing standard web technologies for frontend and Chrome extension API integration for compatibility. Implementation covers Chrome extension development, user authentication, and security feature integration. The user interface is designed for ease, featuring authentication, toggle buttons, and phishing page control. Security integration prioritizes user privacy. Error handling manages CORS security and API downtimes. Performance optimization ensures functionality during API disruptions.

7.1 Recap of Objectives

the project focuses on real-time phishing detection, a user-friendly interface, robust backend infrastructure, and minimalistic security integration. Emphasizing error handling, performance optimization, and continuous testing, the extension ensures reliability and a positive user experience. Collaborative efforts, version control, and mentorship from Ashessh Tiwari Sir contribute to code quality. Metrics guide evaluation, cross-browser compatibility is prioritized, and ethical data handling aligns with privacy regulations. Enhanced security features, user education, and effective notifications further strengthen the extension's impact.

7.2 Contributions to the Field

Web Armour's contributions extend beyond its role as a browser extension, influencing the broader field of online security. Its innovative features, commitment to privacy, and user-centric design principles collectively contribute to a safer and more secure online ecosystem. The project team of Puroo Kulshrestha, Yash Goyal, and Vartika Yadav stands as a noteworthy advancement in the ongoing efforts to mitigate online threats and protect users in the digital realm.

7.3 Future Directions

In the future, Web Armour could explore integrating advanced machine learning algorithms to enhance its phishing detection capabilities and adapt to evolving threats. Additionally, expanding compatibility to other popular browsers beyond Chrome could broaden the extension's impact, providing a more comprehensive solution for users across various platforms. Collaborations with cybersecurity organizations or partnerships with browser developers may offer opportunities for continuous improvement and staying ahead of emerging security challenges.

APPENDICES

Appendix 1: Example of Description Page

Web Armour: Your Shield for Unparalleled Online Security

Introduction:

Web Armour is a robust browser extension developed by a dedicated team led by Puroo Kulshrestha, Yash Goyal, and Vartika Yadav. With a focus on combating phishing threats and enhancing overall online security, this extension provides users with a shield against deceptive websites and potential cyber threats. The extension's user-centric design, coupled with features like user authentication and compatibility with specific Chrome versions, sets it apart in the realm of browser security tools.

Key Features:

Phishing Threat Mitigation: Web Armour actively detects and mitigates phishing threats, safeguarding users from deceptive online practices.

User Authentication: An added layer of security ensures that only authorized users can access the extension's protective features, enhancing overall user trust.

Privacy Compliance: Web Armour adheres to privacy regulations, ensuring responsible handling of user data and setting a benchmark for ethical data practices.

Adaptive Design: The extension's adaptive design caters to the dynamic nature of browser technologies, ensuring a seamless user experience amid ongoing updates.

Appendix 2: Sample References

Smith, J., & Doe, A. "Web Security: A Comprehensive Guide."
Cybersecurity Today "Advancements in Phishing Detection."

Web Armours**:** Your Shield for Unparalleled Online Security Github: https://github.com/Yashgoyal1612/Web_Armour-master

IPqualityscore : https://www.ipqualityscore.com/