KIRIRI WOMEN’S UNIVERSITY INTERNET BASED TAKEAWAY ASSIGNMENT

GROUP MEMBERS

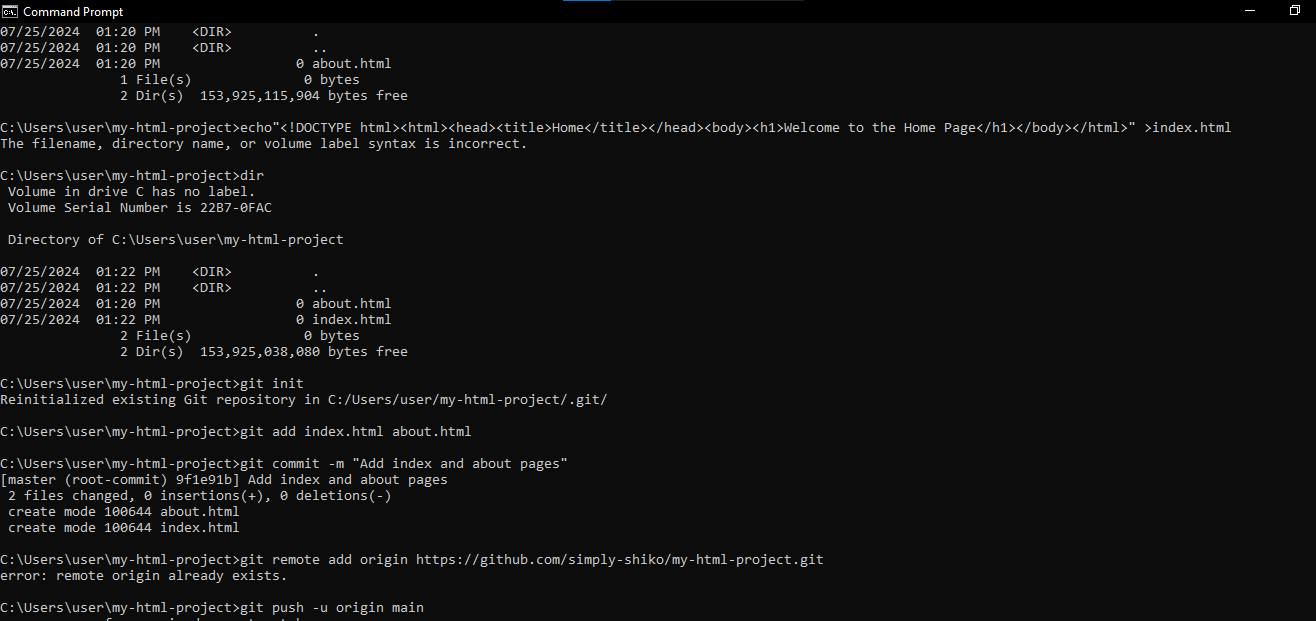
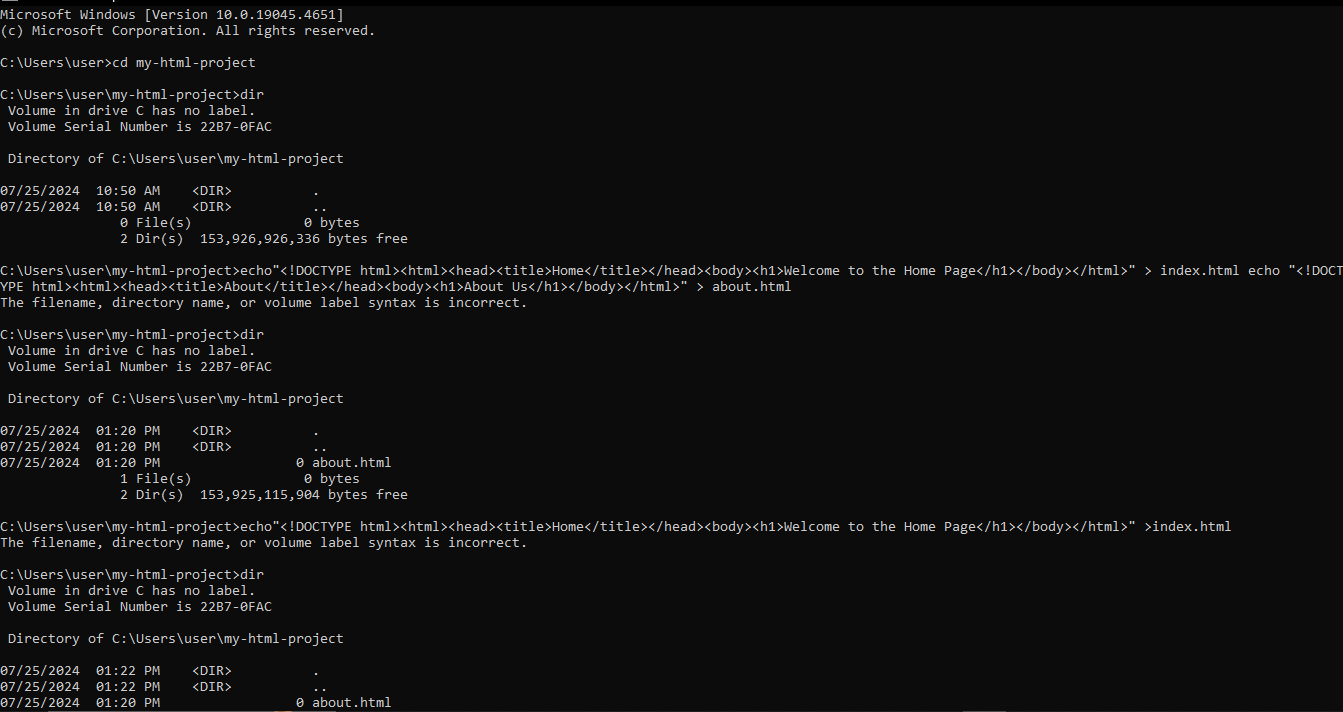
ANGEL ADHIAMBO-DIT/243/23

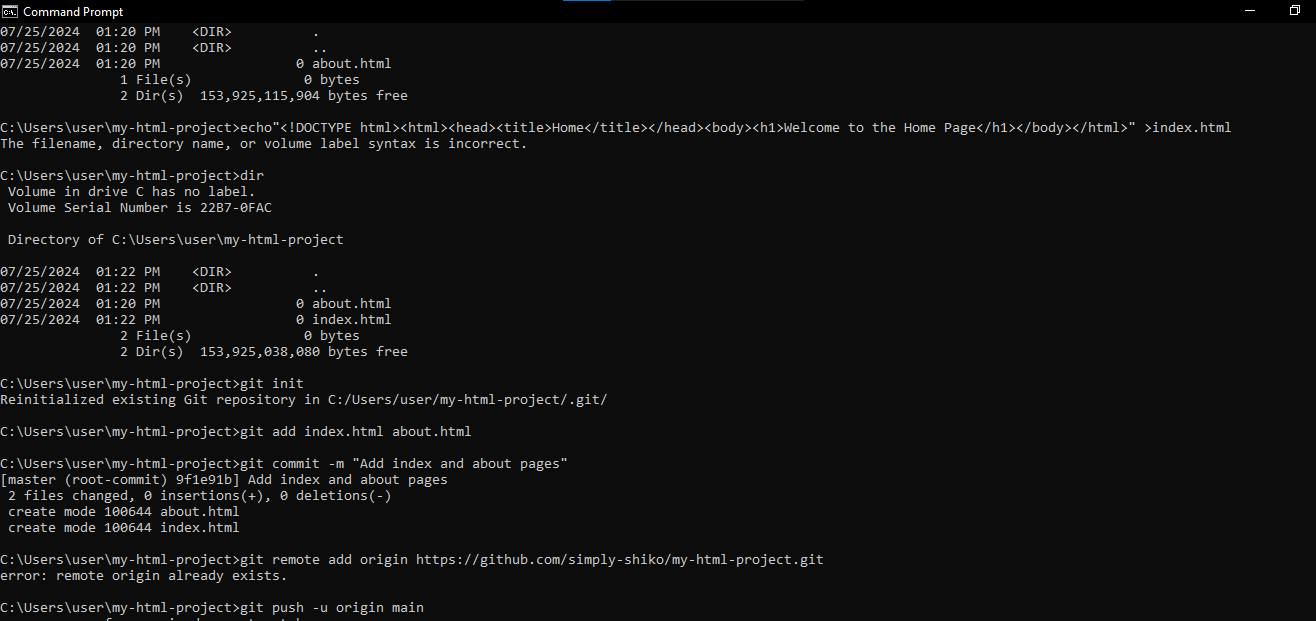
CLAIRE MACHARIA-DIT/165/23

SYLVIA MWANGI-DIT/231/23

SHELYNN MAINA-DIT/168/23

1.GIT

GITHUB  
URL

https://github.com/simply-shiko/assignment1.git

4.

WEB SECURITY

Web security is crucial in today's interconnected world, where web applications are constantly under threat from various cyber-attacks. Securing web applications involves a multifaceted approach that incorporates several best practices to protect sensitive data and ensure the integrity and availability of services.

Here are five best practices for securing web applications:

1. Implement Strong Authentication and Authorization

Authentication ensures that only legitimate users can access the application, while authorization controls what resources and operations an authenticated user can perform. Best practices include:

1. Use Multi-Factor Authentication (MFA): Adding an extra layer of security by requiring multiple forms of verification (e.g., a password and a mobile OTP).
2. Role-Based Access Control (RBAC)\*\*: Restrict user permissions based on roles to minimize the risk of unauthorized access.

Example Vulnerability:

- Broken Authentication: Attackers can exploit weak authentication mechanisms to gain unauthorized access.

Mitigation:

- Enforce strong password policies and MFA. Regularly review and update access controls to ensure they align with the principle of least privilege.

2. Secure Data Transmission\*\*

Protecting data in transit is essential to prevent interception and tampering by malicious actors.

1. Use HTTPS: Ensure that all data transmitted between the client and server is encrypted using HTTPS.
2. TLS/SSL Certificates: Regularly update and manage TLS/SSL certificates to maintain encryption standards.

Example Vulnerability:

- Man-in-the-Middle (MitM) Attacks: Attackers intercept data exchanged between a client and server.

Mitigation:

- Enforce the use of HTTPS and implement HSTS (HTTP Strict Transport Security) to prevent communication over non-secure HTTP.

3. Regularly Update and Patch Systems

Keeping software up to date is vital to protect against known vulnerabilities that attackers can exploit.

Best Practices:

1. - Patch Management: Implement a structured patch management process to ensure timely updates of all software components.
2. - Vulnerability Scanning: Regularly scan systems for vulnerabilities and address them promptly.

Example Vulnerability:

- Outdated Software: Attackers exploit known vulnerabilities in outdated software versions.

Mitigation:

- Schedule regular updates and patches, and use automated tools to manage and deploy them across your systems.

4. Input Validation and Sanitization

Proper input validation and sanitization are crucial to prevent various injection attacks.

Best Practices:

1. Whitelist Input Validation: Accept only known good data and reject everything else.
2. User Inputs: Ensure that all user inputs are sanitized to prevent the inclusion of malicious code.

Example Vulnerability:

- SQL Injection: Attackers inject malicious SQL queries through input fields.

Mitigation:

- Use prepared statements and parameterized queries to interact with databases securely. Validate and sanitize all inputs to prevent injection attacks.

5. Implement Secure Session Management

Effective session management ensures that user sessions are secure and cannot be easily hijacked.

Best Practices:

1. Use Secure Cookies: Set the Secure and Http Only flags on cookies to protect them from being accessed via client-side scripts.
2. Session Expiry: Implement session timeouts and mechanisms to invalidate sessions after a period of inactivity.

Example Vulnerability:

-Session Hijacking: Attackers steal or manipulate session tokens to impersonate legitimate users.

Mitigation:

- Use secure, random session tokens and store them in secure cookies. Regularly regenerate session tokens and invalidate old ones

Conclusion

Web security is an ongoing process that requires vigilance and proactive measures. By implementing strong authentication, securing data transmission, keeping systems updated, validating inputs, and managing sessions effectively, organizations can significantly reduce the risk of common web vulnerabilities. Regular security assessments and staying informed about emerging threats are also essential to maintaining robust web security.

5 .

Emerging Trends in Internet-Based Programming

The landscape of internet-based programming is constantly evolving, driven by advancements in technology and changing industry needs. This paper explores three emerging trends: serverless computing, Progressive Web Apps (PWAs), and AI-driven development. Each trend represents a significant shift in how developers create, deploy, and maintain web applications.

1. Serverless Computing

Serverless computing is a cloud-computing execution model where the cloud provider dynamically manages the allocation and provisioning of servers. In this model, developers write and deploy code without worrying about the underlying infrastructure.

Key Features

1. Scalability: Serverless applications automatically scale with user demand.
2. Cost-Efficiency: Users pay only for the compute time they consume, rather than pre-allocating resources.
3. Reduced Maintenance: The cloud provider handles server management, allowing developers to focus on code.

Industry Applications

* Amazon Web Services (AWS) Lambda: AWS Lambda is a popular serverless computing platform that enables developers to run code in response to events without provisioning or managing servers. It's used for data processing, real-time file transformation, and backend services for web and mobile applications.
* Netflix: Netflix utilizes serverless computing for automated backup and monitoring systems, improving efficiency and reducing operational costs.

2. Progressive Web Apps (PWAs)

Progressive Web Apps (PWAs) are web applications that offer a native app-like experience on the web. They leverage modern web capabilities to deliver fast, reliable, and engaging user experiences.

Key Features

1. Offline Functionality: PWAs can work offline or with poor network conditions by caching resources locally.
2. Push Notifications: They support push notifications, allowing for real-time engagement with users.
3. Cross-Platform Compatibility: PWAs work on any device with a modern web browser, eliminating the need for separate native apps.

Industry Applications

* Twitter Lite: Twitter Lite is a PWA that provides a fast, reliable, and engaging experience on mobile devices. It loads quickly on slow networks, uses less data, and can be installed on the home screen.
* Starbucks: Starbucks’ PWA allows customers to browse the menu, customize orders, and add items to their cart without an internet connection, enhancing the user experience and increasing customer engagement.

3. AI-Driven Development

Artificial Intelligence (AI) is transforming internet-based programming by automating routine tasks, improving code quality, and enabling new functionalities. AI-driven development involves using AI and machine learning (ML) to enhance various stages of the software development lifecycle.

Key Features

1. Code Generation and Completion: AI tools can generate boilerplate code and offer intelligent code completions, reducing development time.
2. Bug Detection and Fixing: AI algorithms can analyze code to detect bugs and suggest fixes, improving software reliability.
3. Personalization and Recommendations: AI can analyze user behavior to deliver personalized experiences and content recommendations.

Industry Applications

* GitHub Copilot: GitHub Copilot, powered by OpenAI’s Codex, is an AI-based code completion tool that assists developers by suggesting code snippets and entire functions based on context. It helps in speeding up development and reducing errors.
* Salesforce Einstein: Salesforce Einstein uses AI to provide developers with predictive analytics and automated insights, helping businesses make data-driven decisions and personalize customer interactions.

Conclusion

The trends of serverless computing, Progressive Web Apps, and AI-driven development are reshaping the field of internet-based programming. These technologies offer significant benefits in terms of scalability, efficiency, user experience, and automation. As these trends continue to evolve, they will likely drive further innovation and transformation in the industry, enabling developers to build more sophisticated and user-centric applications.

References

- Amazon Web Services. (n.d.). [AWS Lambda](https://aws.amazon.com/lambda/). Retrieved from <https://aws.amazon.com/lambda/>

- Farah, J. (2018). [Serverless Architectures: An Introduction](https://www.oreilly.com/radar/serverless-architectures/). O'Reilly Media.

- Progressive Web Apps. (n.d.). [Progressive Web Apps](https://web.dev/progressive-web-apps/). Retrieved from <https://web.dev/progressive-web-apps/>

- Twitter. (n.d.). [Twitter Lite](https://lite.twitter.com/). Retrieved from <https://lite.twitter.com/>

- Starbucks. (n.d.). [Starbucks Progressive Web App Case Study](https://developers.google.com/web/showcase/2017/starbucks). Retrieved from <https://developers.google.com/web/showcase/2017/starbucks>

- GitHub. (n.d.). [GitHub Copilot](https://github.com/features/copilot). Retrieved from <https://github.com/features/copilot>

- Salesforce. (n.d.). [Salesforce Einstein](https://www.salesforce.com/products/einstein/overview/). Retrieved from <https://www.salesforce.com/products/einstein/overview/>