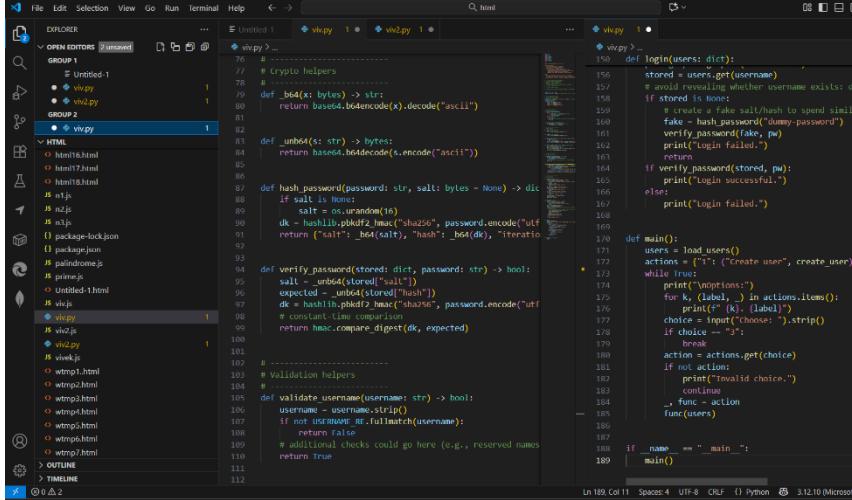
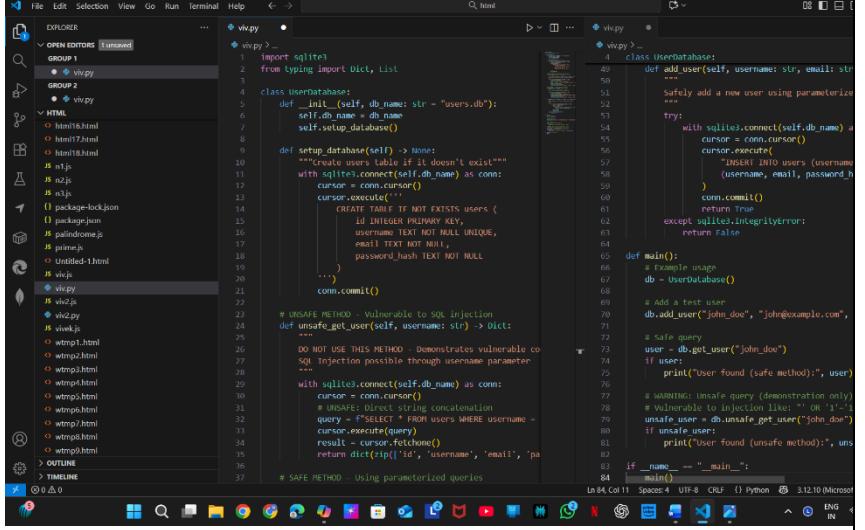
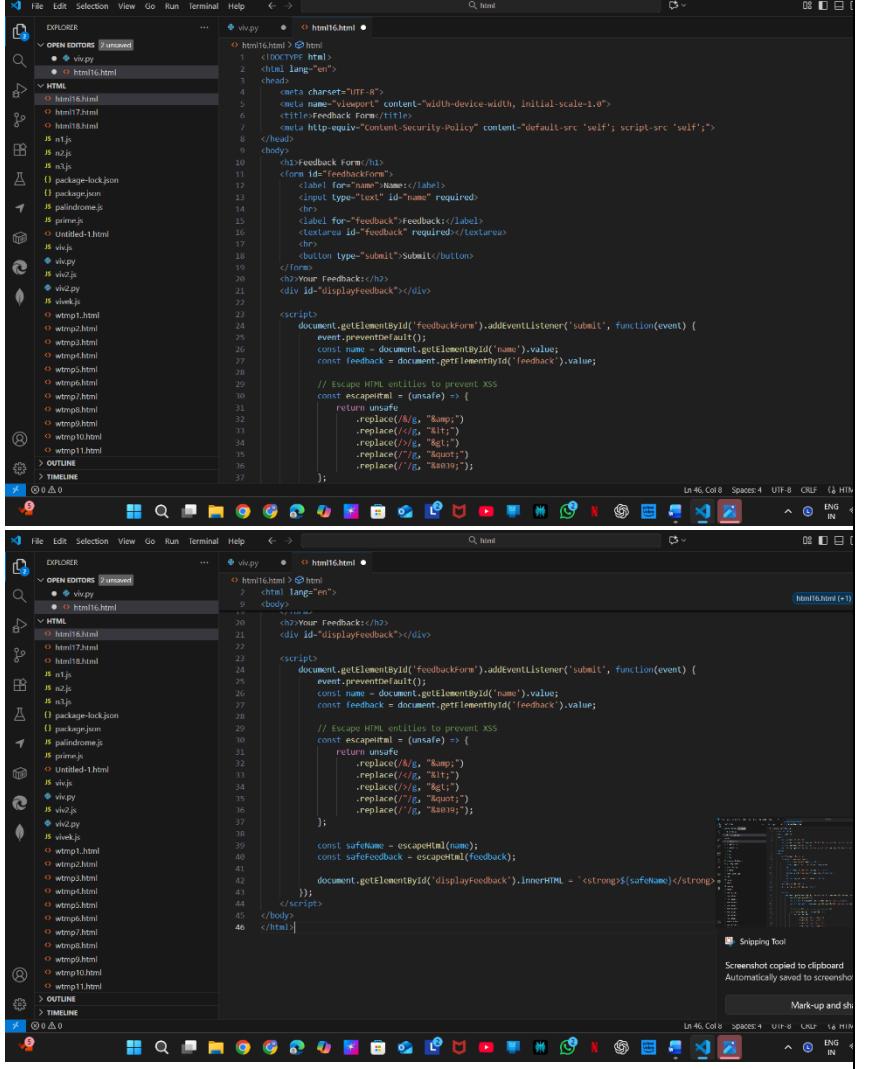


SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE		DEPARTMENT OF COMPUTER SCIENCE ENGINEERING	
<b>Program Name:</b> B. Tech		<b>Assignment Type:</b> Lab	
<b>Course Coordinator Name</b>		Venkataramana Veeramsetty	
<b>Instructor(s) Name</b>		Dr. V. Venkataramana (Co-ordinator) Dr. T. Sampath Kumar Dr. Pramoda Patro Dr. Brij Kishor Tiwari Dr.J.Ravichander Dr. Mohammand Ali Shaik Dr. Anirodh Kumar Mr. S.Naresh Kumar Dr. RAJESH VELPULA Mr. Kundhan Kumar Ms. Ch.Rajitha Mr. M Prakash Mr. B.Raju Intern 1 (Dharma teja) Intern 2 (Sai Prasad) Intern 3 (Sowmya) NS_2 (Mounika)	
<b>Course Code</b>	24CS002PC215	<b>Course Title</b>	AI Assisted Coding
<b>Year/Sem</b>	II/I	<b>Regulation</b>	R24
<b>Date and Day of Assignment</b>	Week10 - Monday	<b>Time(s)</b>	
<b>Duration</b>	2 Hours	<b>Applicable to Batches</b>	
<b>AssignmentNumber:</b> 20.1(Present assignment number)/ <b>24</b> (Total number of assignments)			
<b>Q.No.</b>	<b>Question</b>		<b>Expected Time to complete</b>
1	<b>Lab 20 – Security Testing: Identifying Vulnerabilities in AI-Generated Code</b> <b>Lab Objectives:</b> <ul style="list-style-type: none"> <li>• Understand how to test AI-generated code for common security vulnerabilities.</li> <li>• Learn to apply secure coding principles while analyzing AI</li> </ul>		Week10 - Monday



	 <ul style="list-style-type: none"> <li>• Explanation: CLI for local user management: create accounts and log in, storing data in users.json.</li> <li>• Username validated with a regex; passwords must meet length and complexity rules and have no control/leading-trailing whitespace.</li> <li>• Passwords are hashed with PBKDF2-HMAC-SHA256 using a random 16-byte salt and 200,000 iterations; salt and hash are base64-stored.</li> <li>• Verification uses hmac.compare_digest for constant-time comparison and performs a fake verify on missing users to reduce timing leaks.</li> <li>• Exposes create_user, login, and a menu loop; includes comments advising stronger production practices (bcrypt/argon2, TLS, account lockout).</li> <li>• </li> </ul>
	<p><b>Task 2 – SQL Injection Prevention</b></p> <p><b>Task:</b></p> <p>Test an AI-generated script that performs SQL queries on a database.</p> <p><b>Instructions:</b></p> <ul style="list-style-type: none"> <li>Ask AI to generate a Python script using SQLite/MySQL to fetch user details.</li> <li>Identify if the code is vulnerable to <b>SQL injection</b> (e.g., using string concatenation in queries).</li> <li>Refactor using <b>parameterized queries (prepared statements)</b>.</li> </ul> <p><b>Expected Output:</b></p> <ul style="list-style-type: none"> <li>A secure database query script resistant to SQL injection.</li> </ul>

	 <ul style="list-style-type: none"> <li>Explanation: Manages an SQLite users database and ensures the users table exists via setup_database().</li> <li>unsafe_get_user shows an insecure string-built query vulnerable to SQL injection (do not use).</li> <li>get_user uses parameterized queries to safely fetch a user by username.</li> <li>add_user inserts a new user with parameterized queries and returns False on IntegrityError (e.g., duplicate username).</li> <li>main demonstrates usage: create DB, add a test user, then fetch with the safe method and (for demonstration) the unsafe method.</li> <li></li> </ul>	
	<p><b>Task 3 – Cross-Site Scripting (XSS) Check</b></p> <p><b>Task:</b> Evaluate an AI-generated <b>HTML form with JavaScript</b> for XSS vulnerabilities.</p> <p><b>Instructions:</b></p> <ul style="list-style-type: none"> <li>Ask AI to generate a feedback form with JavaScript-based output.</li> <li>Test whether untrusted inputs are directly rendered without escaping.</li> <li>Implement secure measures (e.g., escaping HTML entities, using CSP).</li> </ul> <p><b>Expected Output:</b></p> <ul style="list-style-type: none"> <li>A secure form that prevents XSS attacks.</li> </ul>	

	 <pre> File Edit Selection View Go Run Terminal Help &lt;- &gt; O html EXPLORER OPEN EDITORS 2 unsaved ... html.html &gt; html &lt;!DOCTYPE html&gt; &lt;html lang="en"&gt;   &lt;head&gt;     &lt;meta charset="UTF-8"&gt;     &lt;meta name="viewport" content="width=device-width, initial-scale=1.0"&gt;     &lt;title&gt;Feedback Form&lt;/title&gt;     &lt;meta http-equiv="Content-Security-Policy" content="default-src 'self'; script-src 'self';"&gt;   &lt;/head&gt;   &lt;body&gt;     &lt;h2&gt;Feedback Form&lt;/h2&gt;     &lt;form id="feedbackForm"&gt;       &lt;label for="name"&gt;Name:&lt;/label&gt;       &lt;input type="text" id="name" required&gt;       &lt;br&gt;       &lt;label for="feedback"&gt;Feedback:&lt;/label&gt;       &lt;textarea id="feedback" required&gt;&lt;/textarea&gt;       &lt;br&gt;       &lt;button type="submit"&gt;Submit&lt;/button&gt;     &lt;/form&gt;     &lt;div id="displayFeedback"&gt;&lt;/div&gt;   &lt;/body&gt; &lt;/html&gt; </pre> <pre> File Edit Selection View Go Run Terminal Help &lt;- &gt; O html EXPLORER OPEN EDITORS 2 unsaved ... html.html &gt; html &lt;html lang="en"&gt;   &lt;head&gt;     &lt;meta charset="UTF-8"&gt;     &lt;meta name="viewport" content="width=device-width, initial-scale=1.0"&gt;     &lt;title&gt;Feedback Form&lt;/title&gt;     &lt;meta http-equiv="Content-Security-Policy" content="default-src 'self'; script-src 'self';"&gt;   &lt;/head&gt;   &lt;body&gt;     &lt;h2&gt;Your Feedback:&lt;/h2&gt;     &lt;div id="displayFeedback"&gt;&lt;/div&gt;   &lt;/body&gt; &lt;/html&gt; </pre> <p>• Explanation: Simple client-side feedback form with a Content-Security-Policy limiting scripts to 'self'.</p> <p>• JavaScript intercepts the form submit, prevents default behavior, and reads the name and feedback inputs.</p> <p>• escapeHtml replaces &amp;, &lt;, &gt;, ", and ' with HTML entities to mitigate XSS.</p> <p>• The code injects the escaped name and feedback into the page using innerHTML (name wrapped in &lt;strong&gt;).</p> <p>• No server persistence — data stays in the browser; escaping + CSP reduce XSS risk but server-side handling would be needed for storage.</p> <p>• </p>	<h3>Task 4 – Real-Time Application: Security Audit of AI-Generated Code</h3>
--	--	--

## Scenario:

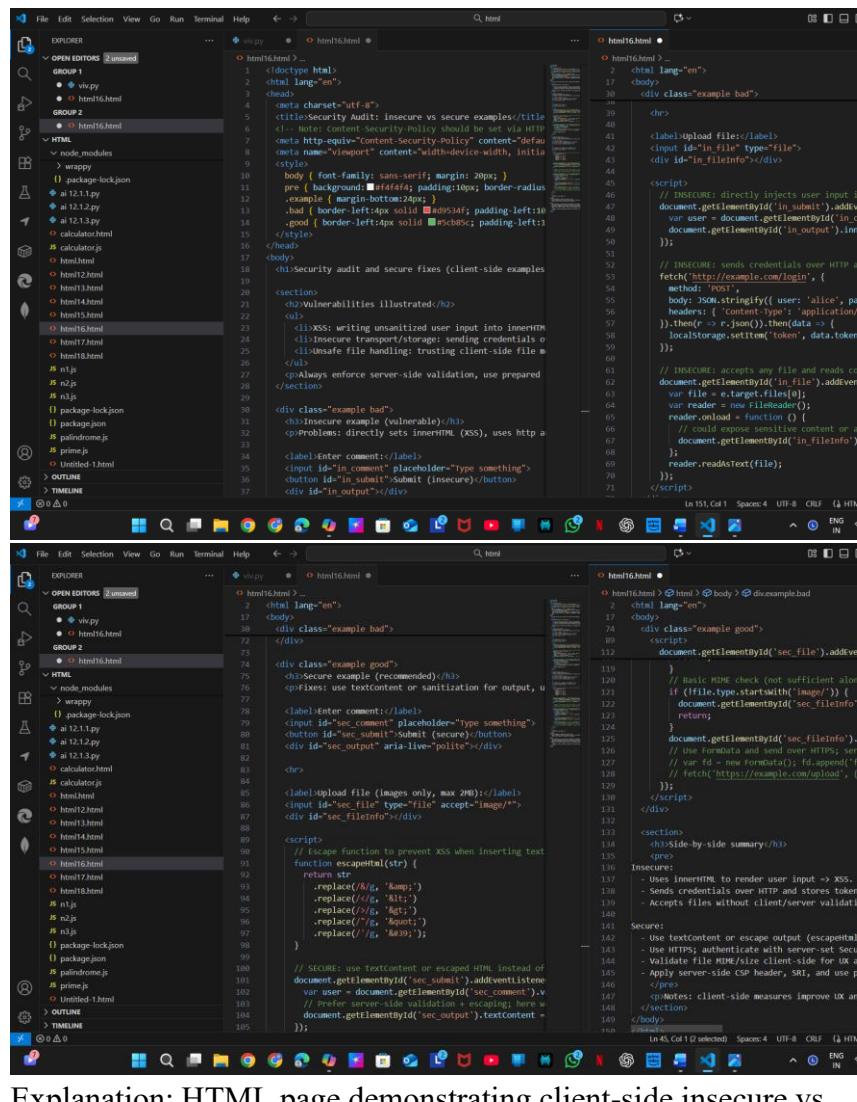
Students pick an **AI-generated project snippet** (e.g., login form, API integration, or file upload).

## Instructions:

- Perform a security audit to detect possible vulnerabilities.
- Prompt AI to suggest **secure coding practices** to fix issues.
- Compare insecure vs secure versions side by side.

## Expected Output:

- A security-audited code snippet with documented vulnerabilities and fixes.



The screenshot shows a code editor with two tabs: 'html16.html' and 'html16.html'. The left tab contains the original AI-generated code, which includes several security vulnerabilities. The right tab contains the audited code with fixes applied. Both tabs have explanatory notes at the bottom detailing the changes made.

**html16.html (Original AI-generated code):**

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>Security Audit: insecure vs secure examples</title>
    <!-- note: Content-Security-Policy should be set via HTTP -->
    <meta http-equiv="Content-Security-Policy" content="default-src 'self'; font-src 'self' ai123.js; script-src 'self' ai123.js; style-src 'self' ai123.css; img-src 'self' ai123.jpg; frame-src 'self' ai123.html; object-src 'self' ai123.html; media-src 'self' ai123.mp3; font-weight: bold; font-size: 16px; color: black; margin: 10px; border: 1px solid black; padding: 10px; background-color: white; border-radius: 5px; transition: all 0.3s ease-in-out; -->
<body>
    <h1>Security audit and secure fixes (client-side examples)</h1>
    <h2>Vulnerabilities illustrated</h2>
    <ul>
        <li>XSS: writing unsanitized user input into innerHTML</li>
        <li>Insecure transport/storage: sending credentials over unsafe file handling: trusting client-side file URLs</li>
        <li>Always enforce server-side validation, use prepared statements</li>
        <li>Div class="example bad":</li>
            <li>Insecure example (vulnerable):</li>
            <li>Problems directly sets innerHTML (XSS), uses http a href instead of https://</li>
            <li><label>Enter comment:</label><br/><input id="in_comment" placeholder="type something"><br/><button id="in_submit" type="submit">Submit (insecure)</button><br/><div id="in_output"></div></li>
        </ul>
    </body>
</html>
```

**html16.html (Audited code):**

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>Security Audit: insecure vs secure examples</title>
    <!-- note: Content-Security-Policy should be set via HTTP -->
    <meta http-equiv="Content-Security-Policy" content="default-src 'self'; font-src 'self' ai123.js; script-src 'self' ai123.js; style-src 'self' ai123.css; img-src 'self' ai123.jpg; frame-src 'self' ai123.html; object-src 'self' ai123.html; media-src 'self' ai123.mp3; font-weight: bold; font-size: 16px; color: black; margin: 10px; border: 1px solid black; padding: 10px; background-color: white; border-radius: 5px; transition: all 0.3s ease-in-out; -->
<body>
    <h1>Security audit and secure fixes (client-side examples)</h1>
    <h2>Vulnerabilities illustrated</h2>
    <ul>
        <li>XSS: writing unsanitized user input into innerHTML</li>
        <li>Insecure transport/storage: sending credentials over unsafe file handling: trusting client-side file URLs</li>
        <li>Always enforce server-side validation, use prepared statements</li>
        <li>Div class="example good":</li>
            <li>Secure example (recommended):</li>
            <li>Fixes: use textContent or sanitization for output, use https:// instead of http://</li>
            <li><label>Enter comment:</label><br/><input id="sec_comment" placeholder="type something"><br/><button id="sec_submit" type="submit" (secure)>Submit</button><br/><div id="sec_output" aria-live="polite"></div><br/><script>
                // escape function to prevent XSS when inserting text
                function escapeText(str) {
                    return str
                        .replace(/"/g, '&quot;')
                        .replace(/'/g, '&apos;')
                        .replace(/>/g, '&gt;')
                        .replace(/</g, '&lt;')
                        .replace(/\r/g, '&cr;')
                        .replace(/\n/g, '&lf;');
                }
            </script>
            <div id="sec_fileInfo"></div>
        </ul>
    </body>
</html>
```

**Notes (Bottom of both tabs):**

**Insecure:**

- Uses innerHTML to render user input → XSS.
- Sends credentials over HTTP and stores tokens in localStorage.
- Accepts files without client/server validation.

**Secure:**

- Use textContent or escape output (escapeText).
- Use HTTPS: authenticate with server-set Secu
- Validate file MIME/type client-side for UX a
- Apply server-side CSP header, SRI, and use p
- *(Note: client-side measures improve UX an*

- Explanations: HTML page demonstrating client-side insecure vs secure patterns and includes a meta Content-Security-Policy note.
- Insecure example: injects unsanitized input via innerHTML (XSS), posts credentials over HTTP and stores tokens in localStorage, and reads uploaded files with no validation.

	<ul style="list-style-type: none"><li>Secure example: escapes input (escapeHtml) or uses textContent to prevent XSS, recommends HTTPS and server-set Secure/HttpOnly cookies instead of localStorage, and performs basic client-side file type/size checks.</li><li>File handling: secure code restricts to images and max 2MB for UX, but explicitly notes the server must re-validate, scan, and enforce stricter checks.</li><li>Page includes a side-by-side summary and emphasizes that client-side measures help UX but all security-critical checks must be implemented on the server (CSP headers, prepared statements, HTTPS, etc.).</li><li></li></ul> <p><input checked="" type="checkbox"/> Deliverables (For All Tasks)</p> <ol style="list-style-type: none"><li>AI-generated prompts for code and test case generation.</li><li>At least 3 assert test cases for each task.</li><li>AI-generated initial code and execution screenshots.</li><li>Analysis of whether code passes all tests.</li><li>Improved final version with inline comments and explanation.</li><li>Compiled report (Word/PDF) with prompts, test cases, assertions, code, and output.</li></ol>	
--	--	--