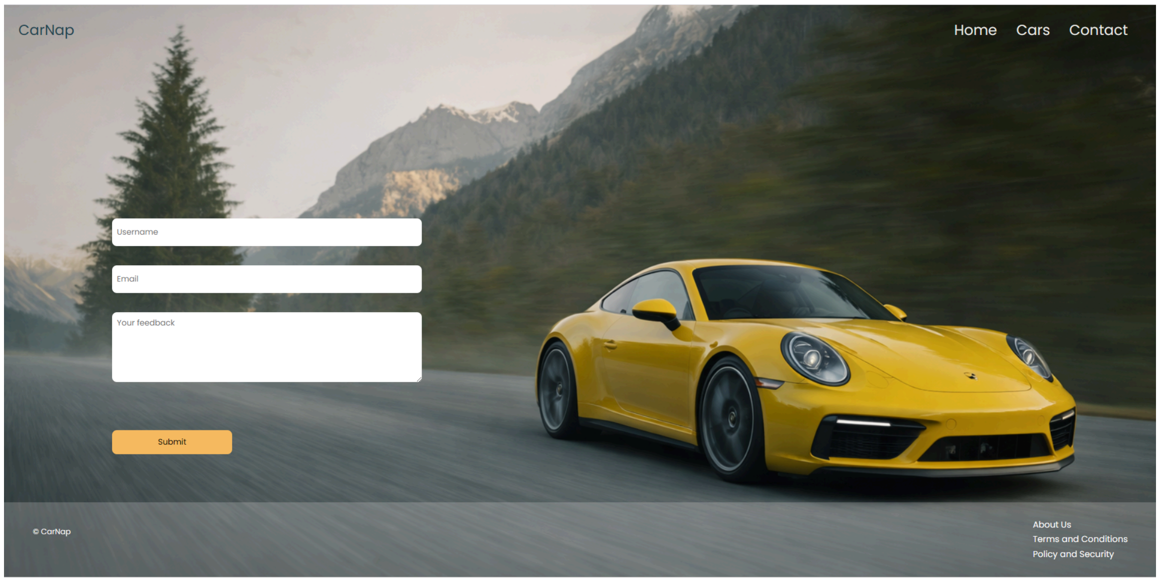
Andrei Jay Amoroto
BSIT 3E
Laboratory #3

Short Report

Overview

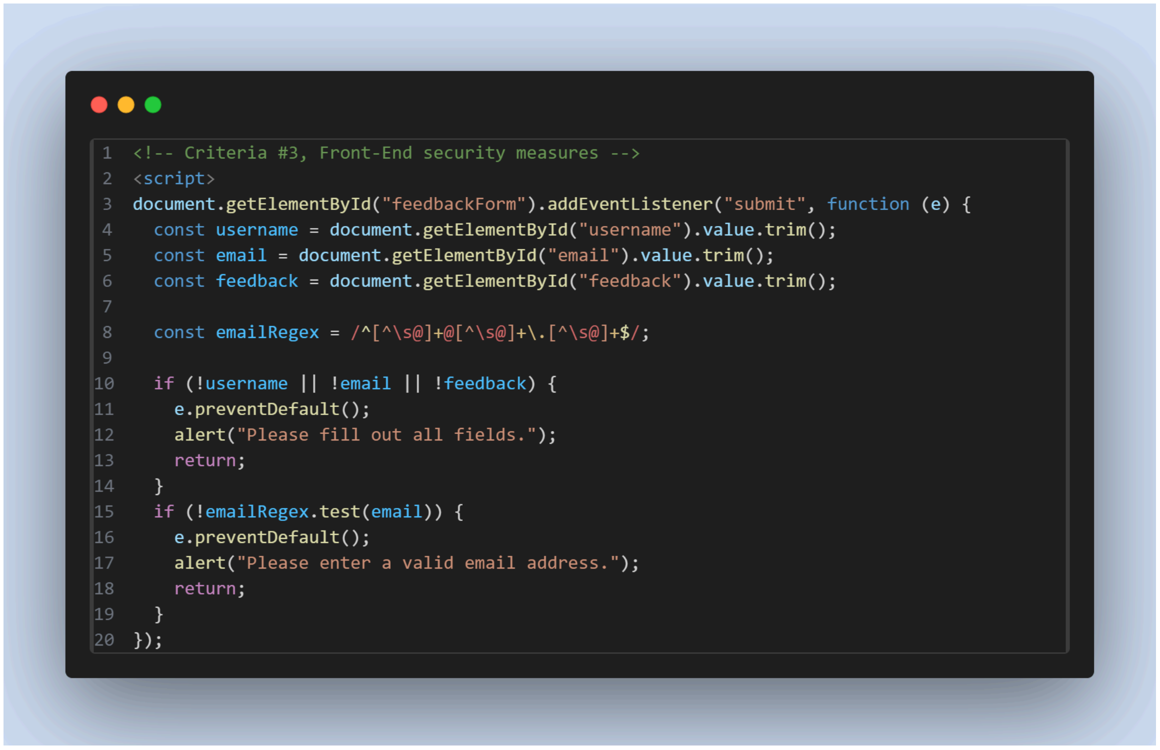
This activity involved designing a simple and user-friendly homepage with basic information security features. The website includes a header with navigation links, a main content section containing a feedback form, and a footer. The goal of the activity is to demonstrate how security principles can be integrated into a basic web application using front-end and back-end techniques, as required in Information Assurance and Security.



Front-End Implementation and Security Measures

On the front end, HTML and CSS were used to create a clean and responsive homepage layout. JavaScript was implemented to perform client-side input validation. The system checks whether all form fields are filled out and validates the email format using a regular expression before allowing submission.

Client-side validation improves user experience by preventing incorrect or incomplete inputs. However, it is not considered secure on its own because it can be bypassed by attackers. Therefore, additional security measures were implemented on the server side.



```
<!-- Criteria #3, Front-End security measures -->
<script>
document.getElementById("feedbackForm").addEventListener("submit", function (e) {
  const username = document.getElementById("username").value.trim();
  const email = document.getElementById("email").value.trim();
  const feedback = document.getElementById("feedback").value.trim();

  const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;

  if (!username || !email || !feedback) {
    e.preventDefault();
    alert("Please fill out all fields.");
    return;
  }
  if (!emailRegex.test(email)) {
    e.preventDefault();
    alert("Please enter a valid email address.");
    return;
  }
});
```

Back-End Security Measures

Node.js with Express was used as the server-side technology to handle form submissions securely. The server re-validates all user inputs to ensure that required fields are not empty and that the email format is correct. This prevents attackers from bypassing client-side validation.

To protect against Cross-Site Scripting (XSS) attacks, user input is sanitized before being processed or displayed. Error handling is implemented using predefined HTML pages for errors and success responses. This prevents sensitive server information, such as file paths or stack traces, from being exposed to users.

Testing and Security Verification

The system was tested using valid and invalid inputs. Submissions with empty fields, invalid email formats, and special characters were correctly rejected by the server. Proper error messages were displayed without revealing internal system details. Successful submissions redirected users to a confirmation page, demonstrating secure and user-friendly feedback handling.