**CMP 416 GROUP ASSIGNMENT**

**Adamani Mivanyi Enan – BHU/20/04/05/0044**

**Ikemka Romanus – BHU/20/04/05/0016**

**Peter Godswill Okatahi – BHU/20/04/05/0012**

**Obenta Christopher Owoicho – BHU/20/04/05/0090**

**Goji Sam – BHU/20/04/05/0142**

**QUESTIONS**

1. There are so many issues with web development, implementation, deployment and usage. Describe an issue (challenge) and discuss how the issue can be solved. (Use code snippets/web views to explain the challenge and mitigation method).
2. There are many process models that are adaptable in implementing software Engineering projects. Use your knowledge of web characteristics and of software process models to argue whether or not a model (selected by your group) is suitable for web features, and adjust the model in question to suite web application development.

NOTE: You are to describe with the aid of the diagram the process model in question, and the new (adjusted) model.

**ANSWERS**

1. There are many issues that arise during web development, implementation, deployment and usage. One of these issues includes:

**Handling Form Validation on the Client and Server**

When building web applications, ensuring data integrity and providing a good user experience often involves validating form inputs both on the client side (for instant feedback) and on the server side (for security and data integrity). A common issue arises when these validations are not synchronized, leading to inconsistent behavior and potential security vulnerabilities.

Scenario

Consider a user registration form where users must provide a username, email, and password. We need to validate the input fields to ensure:

1. The username is alphanumeric and at least 3 characters long.
2. The email is in a valid format.
3. The password is at least 8 characters long.

### Client-Side Validation

Client-side validation provides instant feedback to users, making the user experience smoother.’

HTML form:

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Registration Form</title>

    <style>

        .error { color: red; }

    </style>

</head>

<body>

    <h1>Register</h1>

    <form id="registrationForm">

        <label for="username">Username:</label>

        <input type="text" id="username" name="username" required>

        <span class="error" id="usernameError"></span><br>

        <label for="email">Email:</label>

        <input type="email" id="email" name="email" required>

        <span class="error" id="emailError"></span><br>

        <label for="password">Password:</label>

        <input type="password" id="password" name="password" required>

        <span class="error" id="passwordError"></span><br>

        <button type="submit">Register</button>

    </form>

    <script>

        document.getElementById('registrationForm').addEventListener('submit', function(event) {

            event.preventDefault();

            let isValid = true;

            // Username validation

            const username = document.getElementById('username').value;

            if (!/^[a-zA-Z0-9]{3,}$/.test(username)) {

                document.getElementById('usernameError').textContent = 'Username must be alphanumeric and at least 3 characters long.';

                isValid = false;

            } else {

                document.getElementById('usernameError').textContent = '';

            }

            // Email validation

            const email = document.getElementById('email').value;

            if (!/^[^\s@]+@[^\s@]+\.[^\s@]+$/.test(email)) {

                document.getElementById('emailError').textContent = 'Invalid email format.';

                isValid = false;

            } else {

                document.getElementById('emailError').textContent = '';

            }

            // Password validation

            const password = document.getElementById('password').value;

            if (password.length < 8) {

                document.getElementById('passwordError').textContent = 'Password must be at least 8 characters long.';

                isValid = false;

            } else {

                document.getElementById('passwordError').textContent = '';

            }

            if (isValid) {

                // Submit the form if all validations pass

                this.submit();

            }

        });

    </script>

</body>

</html>

### Server-Side Validation

Server-side validation ensures that no malicious data bypasses the client-side checks.

**Example in Node.js with Express:**

**Set up the server:**

npm install express body-parser

Server Code:

const express = require('express');

const bodyParser = require('body-parser');

const app = express();

app.use(bodyParser.urlencoded({ extended: true }));

app.use(bodyParser.json());

app.post('/register', (req, res) => {

    const { username, email, password } = req.body;

    let errors = {};

    // Username validation

    if (!/^[a-zA-Z0-9]{3,}$/.test(username)) {

        errors.username = 'Username must be alphanumeric and at least 3 characters long.';

    }

    // Email validation

    if (!/^[^\s@]+@[^\s@]+\.[^\s@]+$/.test(email)) {

        errors.email = 'Invalid email format.';

    }

    // Password validation

    if (password.length < 8) {

        errors.password = 'Password must be at least 8 characters long.';

    }

    if (Object.keys(errors).length > 0) {

        res.status(400).json({ errors });

    } else {

        res.status(200).json({ message: 'Registration successful' });

    }

});

app.listen(3000, () => {

    console.log('Server running on port 3000');

});

### Integration and Testing

To integrate client-side and server-side validation, you need to send the form data to the server via an AJAX request and handle the server response appropriately.

**Updated Client-Side Script:**

<script>

    document.getElementById('registrationForm').addEventListener('submit', function(event) {

        event.preventDefault();

        const formData = new FormData(this);

        fetch('/register', {

            method: 'POST',

            body: new URLSearchParams(formData)

        })

        .then(response => response.json())

        .then(data => {

            if (data.errors) {

                // Display server-side validation errors

                if (data.errors.username) {

                    document.getElementById('usernameError').textContent = data.errors.username;

                }

                if (data.errors.email) {

                    document.getElementById('emailError').textContent = data.errors.email;

                }

                if (data.errors.password) {

                    document.getElementById('passwordError').textContent = data.errors.password;

                }

            } else {

                // Handle successful registration (e.g., display a message, redirect, etc.)

                alert('Registration successful');

            }

        })

        .catch(error => console.error('Error:', error));

    });

</script>

### Conclusion

Handling form validation on both the client and server sides is crucial for maintaining a secure and user-friendly web application. By providing immediate feedback on the client side and ensuring data integrity on the server side, you can create a robust validation system. This example demonstrates how to set up both types of validation and integrate them to handle user input effectively.

1. Process Models:

Process models are essential tools used to represent and analyze the flow of activities in various systems. They serve to visualize, document, and improve processes within organizations. A process model is a graphical representation of a process that outlines the sequence of activities, decision points, and interactions between different components. These models are vital in identifying inefficiencies, optimizing workflows, and ensuring that processes are aligned with organizational goals.

#### Understanding Web Characteristics:

* **Dynamic and Rapid Development:** Web applications often need frequent updates.
* **User-Centric:** Focus on user experience, usability, and accessibility.
* **Scalability and Performance:** Must handle varying loads and ensure fast response times.
* **Security:** Protect against web-specific threats.
* **Cross-Platform Compatibility:** Ensure functionality across different browsers and devices.

Common software process models include: Waterfall Model, V-Model, Iterative Model, Spiral Model, Agile Model, DevOps. The model we chose is Agile Model.

The Agile Model is suitable for Web Development for reasons including:

* Flexibility: Agile’s iterative approach allows for adapting to changing requirements;
* Incremental Delivery: Frequent releases align well with the need for regular updates.
* Collaboration: Agile encourages communication among developers, designers and stakeholders;
* Customer Feedback: Continuous feedback from users helps refine the product quickly.
* Risk Management: Continuous testing and integration identify and mitigate risks early.

Adjusting the Agile Model for Web Development

We may apply DevOps techniques to further customise Agile for web development, with an emphasis on improved security and Continuous Integration/Continuous Deployment (CI/CD).

* Sprints and Iterations**:** Regular intervals (sprints) for development and release.
* CI/CD Pipelines: Automate testing and deployment processes to ensure frequent and reliable releases.
* Security Practices**:** Integrate security testing into the development pipeline.
* User Feedback Loop: Continuous collection and incorporation of user feedback.

Original Agile Model



Adjusted Agile + DevOps



To sum up, the integration of DevOps principles into the modified Agile paradigm yields notable improvements in web development projects' productivity, security, and user pleasure. The development process can be made more efficient and dependable by implementing Continuous Integration and Continuous Deployment (CI/CD) pipelines. The integration of enhanced security measures into the development pipeline guarantees the resilience and security of web applications against possible threats. In addition, Agile's iterative structure along with ongoing user feedback loops guarantee that the product develops in accordance with customer requirements and expectations, which in the end results in increased user satisfaction and a more user-centric method for product development.