Nama : I Komang Bintang Ari Medika

NPM : 50422686

Kelas : 3IA01

**GOLANG FOR INTERMEDIATE**

**PERTEMUAN 6**

**(Integrasi Lit Component dengan Golang)**

1. Jelaskan apa yang Anda ketahui mengenai LitElement! (5 point)

   LitElement merupakan sebuah library web components yang modern dan efisien.
   Library ini menggunakan bahasa pemrograman JavaScript versi terbaru yang memungkinkan pengembang dalam membuat custom elements dengan cara yang lebih mudah dan lebih cepat serta performa yang tinggi. LitElement merupakan penerus dari library polymer yang pertama kali dikembangkan untuk pembuatan web components. LitElement dibangun dengan menyempurnakan kekurangan – kekurangan yang dihadapi saat pengembangan polymer.

2. Apa keunggulan menggunakan Web Component (seperti Polymer, LitElement, dan sebagainya) dibandingkan menggunakan framework besar seperti ReactJS, Vue, dan sebagainya)? (5 point)

Web Components memiliki beberapa keunggulan dibandingkan dengan framework besar seperti ReactJS dan Vue, antara lain:

- Isolasi yang Kuat:Web Components memungkinkan encapsulasi gaya dan perilaku dengan menggunakan Shadow DOM. Ini mencegah konflik gaya dan memastikan komponen berfungsi secara independen dari konteks di sekitarnya.
- Kinerja:Karena Web Components adalah native pada browser, mereka cenderung lebih cepat dan efisien dalam hal kinerja dibandingkan dengan komponen yang dibangun menggunakan framework JavaScript yang besar.
- Penggunaan Lebih Sedikit Resources: Penggunaan Web Components dapat mengurangi ukuran bundle karena tidak perlu memuat seluruh library framework besar, yang berguna terutama untuk aplikasi dengan keterbatasan sumber daya.

ACT (Total 90 Point)

Sebuah perusahaan bernama "Menyala Abangku Corp." sedang mengembangkan sistem autentikasi berbasis web components menggunakan Lit. Anda ditugaskan untuk membuat sebuah aplikasi web sederhana yang menampilkan form login dan pesan rahasia menggunakan Web

Components dengan Lit library. Aplikasi ini harus memiliki sistem autentikasi dasar dan dapat menampilkan pesan rahasia hanya untuk pengguna yang sudah login.

Lakukan langkah – langkah pengerjaan berikut ini

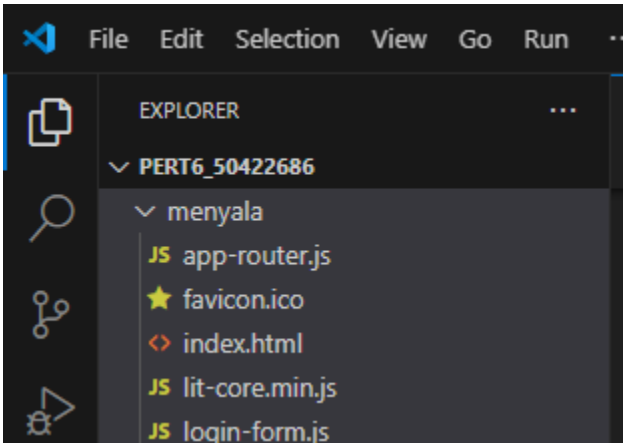
1. Import Library yang diperlukan! (5 point)
2. Deklarasikan variabel konstanta PORT dengan nilai sesuai lima digit terakhir NPM! (5 point)
3. Buatlah sebuah function untuk serve satu folder static! (10 point)
4. Buatlah login handler untuk login api! (10 point)
5. Buat function main untuk menjalankan server, buatlah route API endpoint /api/login untuk fungsi login handler (handleLogin) (10 point).

```go
package main

//No 1
import (
    "encoding/json"
    "fmt"
    "log"
    "net/http"
    "os"
    "path/filepath"
    "strings"
)

// No 2
const PORT = 22686 // Change the port to the last 5 digits of your student ID (NPM/NIM).

// No 3
func serveStaticFile(w http.ResponseWriter, r *http.Request, baseDir string, fileServer http.Handler) {
    path := strings.TrimPrefix(r.URL.Path, "/")
    fullPath := filepath.Join(baseDir, path)

    fileInfo, err := os.Stat(fullPath)
    if err != nil {
        if os.IsNotExist(err) {
            log.Printf("File does not exist: %s", fullPath)
        } else {
            log.Printf("Error checking file: %s. Error: %v", fullPath, err)
        }
    } else {
        log.Printf("retrieved %d bytes", fileInfo.Size())
    }
```

```
    fileServer.ServeHTTP(w, r)
}

// No 4
func handleLogin(w http.ResponseWriter, r *http.Request) {
    if r.Method != http.MethodPost {
        http.Error(w, "Method not allowed", http.StatusMethodNotAllowed)
        return
    }

    var loginRequest struct {
        Username string `json:"username"`
        Password string `json:"password"`
    }

    err := json.NewDecoder(r.Body).Decode(&loginRequest)
    if err != nil {
        http.Error(w, "Invalid request body", http.StatusBadRequest)
        return
    }

    if loginRequest.Username == "Bintang" && loginRequest.Password == "50422686"
{ //Change the username to your last name and the password to your NPM (student
ID number/MPH).
        w.WriteHeader(http.StatusOK)
        json.NewEncoder(w).Encode(map[string]string{"message": "Login
successful"})
    } else {
        w.WriteHeader(http.StatusUnauthorized)
        json.NewEncoder(w).Encode(map[string]string{"message": "Invalid username
or password"})
    }
}

//No 5

func main() {
    fileServer := http.FileServer(http.Dir("menyala"))
    http.HandleFunc("/", func(w http.ResponseWriter, r *http.Request) {
        serveStaticFile(w, r, "menyala", fileServer)
    })

    http.HandleFunc("/api/login", handleLogin)
```

```
    log.Printf("Server running on http://[::1]:%d/...", PORT)
    err := http.ListenAndServe(fmt.Sprintf(":%d", PORT), nil)
    if err != nil {
        log.Fatalf("Error starting the server on port %d: %v", PORT, err)
    }
}
```

6. (5 point) Buat folder bernama menyala , kemudian masukkan file lit-core.min.js ke dalamnya, lalu buat empat file bernama:
   a. index.html
   b. app-router.js
   c. login-form.js
   d. secret-message.js




7. Isikan Komponen login-form.js (15 point) dengan spesifikasi:
   a. Class Component (5 point)
   b. Form Template (5 point)
   c. Submit Handler (5 point)

```
import {LitElement, html, css} from '/lit-core.min.js';
//No 7
// Login Form Component

class LoginForm extends LitElement {
    static styles = css`
    :host {
        display: block;
        font-family: 'Roboto', sans-serif;
        --primary-color: #6200ea;
        --primary-color-hover: #3700b3;
        --background-color: #f5f5f5;
        --card-background: #ffffff;
        --input-border-color: #bdbdbd;
        --input-border-focus-color: #6200ea;
```

```
        --text-color: #000000;
    }
    .container {
        display: flex;
        justify-content: center;
        align-items: center;
        height: 100vh;
        background-color: var(--background-color);
    }
    .login-form {
        background-color: var(--card-background);
        padding: 24px;
        border-radius: 8px;
        box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
        width: 100%;
        max-width: 400px;
    }
    form {
        display: flex;
        flex-direction: column;
        gap: 16px;
    }
    input {
        padding: 16px;
        border: 1px solid var(--input-border-color);
        border-radius: 4px;
        font-size: 16px;
        color: var(--text-color);
        transition: border-color 0.3s ease;
    }
    input:focus {
        border-color: var(--input-border-focus-color);
        outline: none;
    }
    button {
        padding: 12px;
        background-color: var(--primary-color);
        color: white;
        border: none;
        border-radius: 4px;
        cursor: pointer;
        font-size: 16px;
        transition: background-color 0.3s ease;
    }
    button:hover {
```

```
        background-color: var(--primary-color-hover);
    }
    .error-message {
        color: red;
        font-size: 14px;
        margin-top: 8px;
    }
`;

    static properties = {
        username: { type: String },
        password: { type: String },
        errorMessage: { type: String },
    };

    constructor() {
        super();
        this.username = '';
        this.password = '';
        this.errorMessage = '';
    }

//Form Template
    render() {
        return html`
            <div class="container">
                <div class="login-form">
                    <form @submit=${this._handleSubmit}>
                        <input
                            type="text"
                            placeholder="Username"
                            .value=${this.username}
                            @input=${(e) => this.username = e.target.value}
                            required
                        >
                        <input
                            type="password"
                            placeholder="Password"
                            .value=${this.password}
                            @input=${(e) => this.password = e.target.value}
                            required
                        >
                        <button type="submit">Login</button>
                        ${this.errorMessage ? html`<p class="error-
message">${this.errorMessage}</p>` : ''}
```

```
                    </form>
                </div>
            </div>
        `;
    }

//Submit Handler
    async _handleSubmit(e) {
        e.preventDefault();
        this.errorMessage = '';

        try {
            const response = await fetch('/api/login', {
                method: 'POST',
                headers: {
                    'Content-Type': 'application/json',
                },
                body: JSON.stringify({
                    username: this.username,
                    password: this.password,
                }),
            });

            const data = await response.json();

            if (response.ok) {
                localStorage.setItem('isLoggedIn', 'true');
                this.dispatchEvent(new CustomEvent('login-success', {
                    bubbles: true,
                    composed: true
                }));
            } else {
                this.errorMessage = data.message || 'Login failed. Please try
again.';
            }
        } catch (error) {
            console.error('Login error:', error);
            this.errorMessage = 'An error occurred. Please try again later.';
        }
    }
}
customElements.define('login-form', LoginForm);
```

8. Buat file secret-message.js (15 point) yang menampilkan pesan rahasia untuk user yang sudah login.

a. Implementasi Basic Component (5 point).
b. Implementasi Template dan User Interface (10 point).

```
//?./menyala/secret-message.js
import { LitElement, html, css } from '/lit-core.min.js';

//implementasi basic component
class SecretMessage extends LitElement {
    static styles = css`
        :host {
            display: block;
            font-family: 'Roboto', sans-serif;
        }
        .container {
            display: flex;
            justify-content: center;
            align-items: center;
            height: 100vh;
            background-color: #f5f5f5;
        }
        .card {
            background-color: white;
            padding: 2rem;
            border-radius: 8px;
            box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
            text-align: center;
        }
        .secret-container {
            margin: 1rem 0;
            padding: 1rem;
            background-color: #e0e0e0;
            border-radius: 4px;
            cursor: pointer;
        }
        .secret-message {
            font-size: 1.2rem;
            font-weight: bold;
            opacity: 0;
            transition: opacity 0.5s ease;
        }
        .revealed .secret-message {
            opacity: 1;
        }
        button {
            margin-top: 1rem;
```

```
            padding: 0.5rem 1rem;
            background-color: #0056b3;
            color: white;
            border: none;
            border-radius: 4px;
            cursor: pointer;
        }
    `;

    static properties = {
        secretMessage: { type: String },
        isRevealed: { type: Boolean },
        isAuthenticated: { type: Boolean }
    };

    constructor() {
        super();
        this.secretMessage = "You're awesome! 🌟";
        this.isRevealed = false;
        this.isAuthenticated = false;
    }

    connectedCallback() {
        super.connectedCallback();
        this.checkAuthentication();
    }

    checkAuthentication() {
        const isLoggedIn = localStorage.getItem('isLoggedIn');
        this.isAuthenticated = isLoggedIn === 'true';
    }

    //Implementasi Template dan User Interface
    render() {
        if (!this.isAuthenticated) {
            return html`
                <div class="container">
                    <div class="card">
                        <h2>Access Denied</h2>
                        <p>Please log in to view the secret message.</p>
                    </div>
                </div>
            `;
        }
```

```
        return html`
            <div class="container">
                <div class="card">
                    <h2>Secret Message</h2>
                    <div class="secret-container ${this.isRevealed ? 'revealed' :
''}" @click="${this.toggleReveal}">
                        <p class="secret-message">${this.secretMessage}</p>
                    </div>
                    <button @click="${this.toggleReveal}">
                        ${this.isRevealed ? 'Hide' : 'Reveal'} Secret
                    </button>
                    <button @click="${this.logout}">Logout</button>
                </div>
            </div>
        `;
    }

    toggleReveal() {
        this.isRevealed = !this.isRevealed;
    }

    logout() {
        localStorage.removeItem('isLoggedIn');
        this.dispatchEvent(new CustomEvent('logout', {
            bubbles: true,
            composed: true
        }));
    }
}
customElements.define('secret-message', SecretMessage);
```

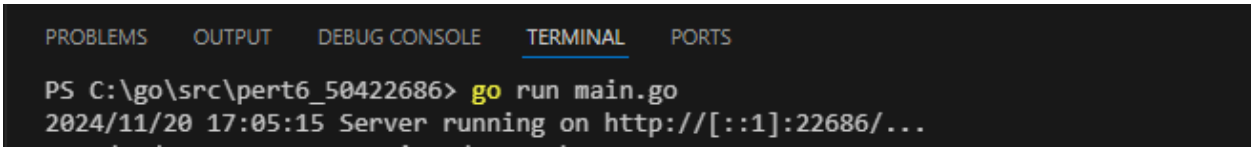
9. Isikan app-router.js (10 point) dengan spesifikasi:
    a. Implementasi Router Component:

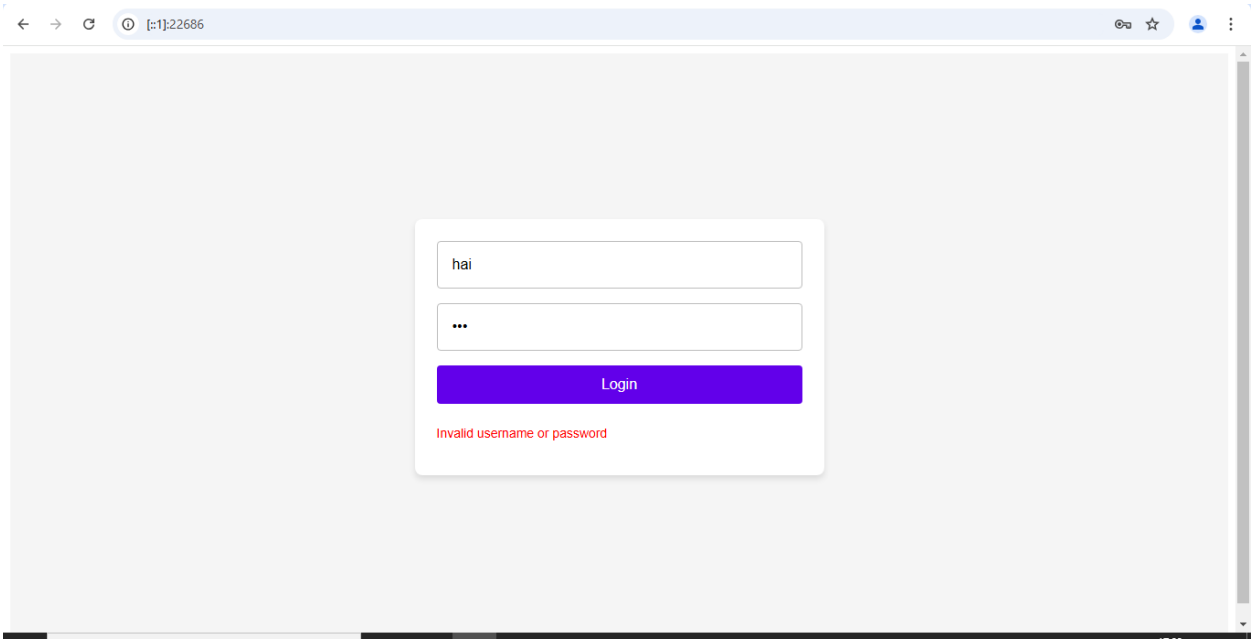
```
import { LitElement, html, css } from '/lit-core.min.js';
import './login-form.js';
import './secret-message.js';
class AppRouter extends LitElement {
    static properties = {
        route: { type: String }
    };

    constructor() {
        super();
```

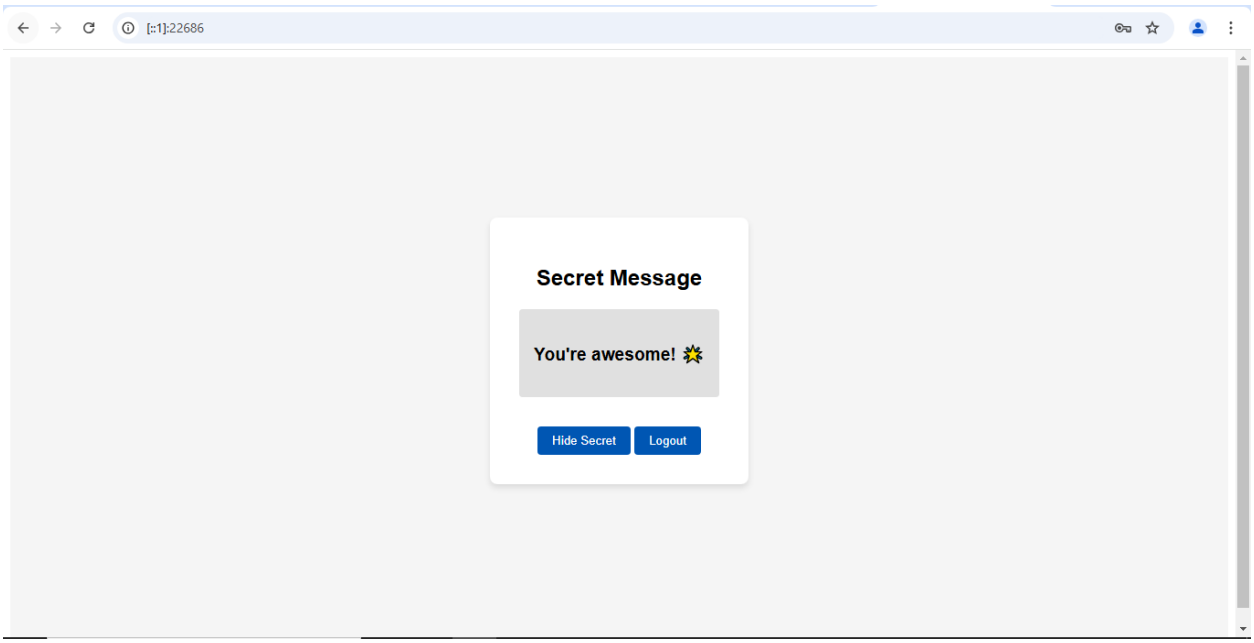
```
        this.route = localStorage.getItem('isLoggedIn') === 'true' ? 'secret' :
'login';
    }

    render() {
        return html`
            ${this.route === 'login'
                ? html`<login-form @login-
success="${this.handleLoginSuccess}"></login-form>`
                : html`<secret-message @logout="${this.handleLogout}"></secret-
message>`}
        `;
    }

    handleLoginSuccess() {
        this.route = 'secret';
    }

    handleLogout() {
        this.route = 'login';
    }
}
customElements.define('app-router', AppRouter);
```

10. Buat Index.html (5 point) yang berfungsi sebagai entry point aplikasi web yang menghubungkan semua komponen web.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Secret Message App</title>
    <script type="module" src="/lit-core.min.js"></script>
    <script type="module" src="/login-form.js"></script>
    <script type="module" src="/secret-message.js"></script>
    <script type="module" src="/app-router.js"></script>
</head>
<body>
    <app-router></app-router>
</body>
</html>
```


PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
PS C:\go\src\pert6_50422686> go run main.go
2024/11/20 17:05:15 Server running on http://[::1]:22686/...


[::1]:22686
hai
Login
Invalid username or password


[::1]:22686
Secret Message
You're awesome!
Hide Secret
Logout