Лабораторна робота №3: Фронтенд-фреймворки

Мета: Отримати практичний досвід роботи з одним із сучасних фронтенд-фреймворків. Навчитись розробляти веб-сторінки за допомогою компонентів, керувати станами компонент та працювати з маршрутизацію для переходу між різними відображеннями (сторінками).

Теоретичні відомості

Основи фронтенд-фреймворків

Оскільки веб-додатки стають все більш динамічними та складними, розробникам потрібні кращі інструменти для управління користувацькими інтерфейсами (UI) та забезпечення їхньої підтримки і продуктивності. У той час як для невеликих проектів достатньо простого **JavaScript, HTML та CSS**, великі додатки потребують більш **структурованого** та **ефективного** підходу. Саме в таких ситуаціях потрібні **фронтенд-фреймворки**, такі як **Angular, React та Vue**.

Фреймворки дозволяють розробникам створювати **динамічні користувацькі інтерфейси**, які реагують на зміни в даних, без необхідності ручного оновлення DOM. Вони також пропонують **інструменти та шаблони**, які сприяють **модульній розробці**, полегшуючи створення, підтримку та масштабування веб-додатків. Основною перевагою використання фронтенд-фреймворку є **зменшення кількості повторюваних завдань** і **підвищення продуктивності розробників**, а також забезпечення високої продуктивності роботи користувацького досвіду.

Сучасні веб-фреймворки є невід'ємною частиною створення **інтерактивних** і **динамічних** веб-сайтів, які зараз широко використовуються в реальних додатках, таких як платформи електронної комерції, соціальні мережі та корпоративні інформаційні панелі.

Використання фронтенд-фреймворків

Розробка за допомогою інтерфейсних фреймворків стала стандартом для створення веб-додатків, оскільки вони спрощують реалізацію багатьох поширених завдань у структурований спосіб. Ось основні причини, чому розробники обирають фронтенд-фреймворки:

- **Архітектура на основі компонентів**: Такі фреймворки, як Angular, React та Vue, дозволяють розробникам створювати додатки з використанням незалежних компонент, які можна використовувати повторно. Такий модульний підхід зменшує дублювання коду і полегшує підтримку та масштабування великих додатків.
- **Оптимізація відображення**: Фронтенд-фреймворки призначені для ефективного відображення змін даних в інтерфейсі. Вони використовують такі технології, як Virtual DOM (React, Vue) та стратегії виявлення змін (Angular), щоб оновлювати лише ті частини сторінки, які потребують змін, замість того, щоб оновлювати всю сторінку.
- **Інструменти та бібліотеки**: Кожен фреймворк має набір інструментів, бібліотек та утиліт, які пришвидшують процес розробки. Сюди входить підтримка процесів

збірки, тестування, перевірки якості коду та інтеграція з іншими бібліотеками і внутрішніми сервісами.

- **Екосистема та спільнота**: Фреймворки підтримуються великими, активними спільнотами, що забезпечує регулярні оновлення, виправлення помилок та розробку безліч сторонніх плагінів і розширень.

Огляд Angular, React та Vue

Хоча Angular, React і Vue створені з однією метою - створення користувацьких інтерфейсів для сучасних веб-додатків - вони відрізняються за своєю архітектурою, філософією дизайну та екосистемою.

1. Angular (розроблений компанією Google)

Angular - це **повноцінний фронтенд-фреймворк**, який надає цілу екосистему для розробки веб-додатків. Він включає в себе не тільки базові можливості візуалізації інтерфейсу користувача, але й інструменти для роботи з формами, HTTP-запитами та складними сценаріями маршрутизації.

- **Найкраще підходить для**: Масштабні корпоративні додатки, особливо ті, що потребують повнофункціонального фреймворку з усіма інструментами.
- **Основні характеристики**:
    - **Використання мови TypeScript**: Angular побудовано з використанням TypeScript, тобто розширення мови JavaScript, яка додає статичну типізацію.
    - **Декларативні шаблони**: Angular використовує шаблони на основі HTML, які декларують структуру та поведінку інтерфейсу, що полегшує створення та розуміння макету.
    - **Директиви**: Директиви - це спеціальні атрибути в розмітці HTML, які додають додаткову функціональність елементам (наприклад, ngIf, ngFor).
    - **Інтеграція залежностей**: Angular має потужну систему інтеграції залежностей, яка допомагає ефективніше керувати функціоналом додатку та компонентами.
- **Використання в реальному житті**:
    - Google Ads, Google Analytics та інші великі сервіси Google.
    - Складні фінансові системи, такі як банківські платформи.
    - Масштабне корпоративне програмне забезпечення.

2. React (розроблений Meta/Facebook)

React - це **легка бібліотека JavaScript**, розроблена спеціально для створення користувацьких інтерфейсів. На відміну від Angular, вона фокусується в першу чергу на відображенні інтерфейсу і залишає інші аспекти розробки, такі як маршрутизація та управління станами, зовнішнім бібліотекам.

- **Найкраще для**: Створення інтерактивних, компонентно-керованих користувацьких інтерфейсів для веб- та мобільних додатків.
- **Основні характеристики**:
    - **Синтаксис JSX**: React використовує JSX, синтаксис, який дозволяє писати HTML-подібний код всередині JavaScript. Даний підхід забезпечує зручний спосіб створення компонентів.
    - **Односпрямований потік** даних: потоки даних рухаються в одному напрямку, від батьківських компонентів до дочірніх, забезпечуючи передбачуваний стан даних у всьому додатку.
    - **Повторне використання компонентів**: React наголошує на можливості повторно використовувати компоненти, що покращує підтримку коду та зменшує складність реалізації додатків.
    - **Віртуальний DOM**: React використовує віртуальний DOM для покращення продуктивності роботи додатків, мінімізуючи кількість оновлень реального DOM.
- **Використання в реальному житті**:
    - Веб-додатки Facebook, Instagram та WhatsApp.
    - Netflix, Airbnb та Uber.
    - Складні, інтерактивні платформи електронної комерції.

3. Vue.js (створений спільнотою)

Vue - це **прогресивний JavaScript-фреймворк**, розроблений для поступової адаптації. На відміну від Angular, Vue менш структурований і дозволяє розробникам використовувати його для створення різних типів веб-додатків - від простих віджетів до повноцінних SPA (односторінкові додатки).

- **Найкраще підходить для**: Швидка розробка малих і середніх додатків або інтеграція в існуючі проекти.
- **Основні характеристики**:
    - **Декларативний синтаксис**: Vue надає синтаксис, який дуже схожий на HTML та CSS, що полегшує роботу розробникам, знайомим з веб-стандартами та початок роботи над додатками.
    - **Реактивне відображення даних**: Vue автоматично оновлює інтерфейс користувача, коли змінюється модель даних, що покращує взаємодію з користувачем.
    - **Vue CLI**: Vue надає інструмент командного рядка, який спрощує налаштування проекту та виконання типових завдань розробки.
    - **Однофайлові компоненти**: Vue дозволяє розробникам розробляти компоненти в одному файлі з розділами <template>, <script> та <style>, що спрощує процес розробки.

- **Використання в реальному житті**:
    - Alibaba, Xiaomi та GitLab.
    - Інтерактивні новинні сайти, легкі додатки.
    - Внутрішні бізнес-додатки та інформаційні панелі.

Функції, спільні для Angular, React та Vue

Хоча Angular, React і Vue мають свої унікальні особливості, є кілька **спільних рис**, які роблять ці фреймворки зручними для створення динамічних, сучасних веб-додатків:

1. Компонентно-орієнтована архітектура

Всі три фреймворки підтримують **компонентну архітектуру**, яка розбиває додатки на менші, багаторазово використовувані одиниці. Компоненти керують власним станом і динамічно відображають інтерфейс користувача. Такий підхід дозволяє розробникам створювати додатки, які легше підтримувати та тестувати, інкапсулюючи логіку та представлення в автономні одиниці.

В усіх трьох фреймворків основна ідея однакова:

- **Angular** використовує вбудовані декоратори **NgModules** та **Components** для реалізації компонентного підходу.
- **React** використовує **функції** або **класи** для створення компонентів.
- **Vue** використовує **однофайлові компоненти** (файли .vue).

2. Обробка користувацького вводу та форм

Кожен фреймворк надає своє рішення для управління введенними даними користувачем, перевіркою форм та їх надсиланням на серверну частину. Наприклад:

- **Angular** надає елементи керування формами, такі як ngModel для двостороннього зв'язування даних та ReactiveForms для більш складного керування формами.
- **React** керує станом форми через керовані компоненти, використовуючи хуки на кшталт useState.
- **Vue** підтримує двостороннє зв'язування з v-model для форм, що робить роботу з формами простою та інтуїтивно зрозумілою.

3. Директиви

Кожен фреймворк надає способи додавання **користувацької поведінки** до HTML-елементів за допомогою спеціальних **директив**:

- **Angular** використовує вбудовані директиви, такі як ngIf, ngFor та ngClass.
- **React** використовує **JSX-вирази** для додавання логіки у функцію рендерингу, наприклад, умовний рендеринг з { умова ? <Компонент /> : null }.
- **Vue** використовує директиви v-if, v-for та інші для обробки логіки та рендерингу в шаблонах.

4. Прив'язка даних

Усі три фреймворки працюють з **прив'язкою даних**, хоча вони відрізняються за способом реалізації. Прив'язка даних гарантує, що зміни даних в одній частині програми автоматично оновлюють відповідні компоненти інтерфейсу.

- **Angular** підтримує **двостороннє зв'язування даних**, коли зміни в інтерфейсі поширюються на модель даних, і навпаки.
- **React** підтримує **одностороннє зв'язування даних**, коли дані рухаються в одному напрямку (від батьківського до дочірнього компонента), що робить їх більш передбачуваними.
- **Vue** також підтримує двостороннє зв'язування через v-model для синхронізації інтерфейсу з моделлю даних.

Вибір правильного фреймворку

Вибір між Angular, React та Vue залежить від потреб проекту та досвіду розробників. Angular найкраще підходить для великих додатків корпоративного рівня зі складними вимогами, тоді як React підходить для створення динамічних та інтерактивних інтерфейсів. Vue, будучи простішим і гнучкішим, добре підходить для малих і середніх проектів або проектів, які потребують інтеграції з іншими технологіями. Всі три фреймворки широко використовуються в індустрії, і їх розуміння допоможе вибрати правильний інструмент для проекту з веб-розробки.

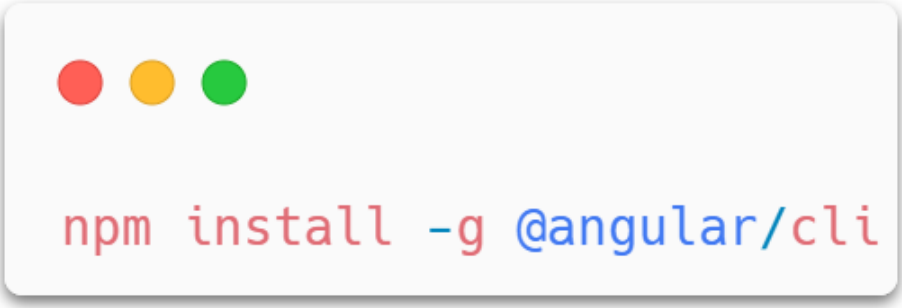
Порядок виконання роботи

Крок 1: Створення проекту

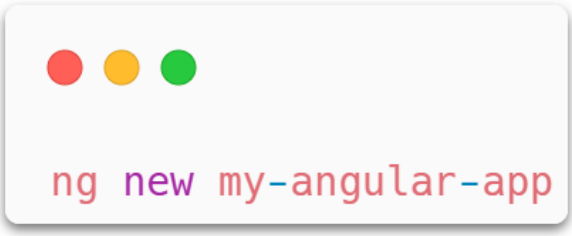
Опис:
Створити проект за допомогою одного з фреймворків. Всі фреймворки (Angular, React і Vue) мають власні інструменти CLI для швидкого створення нового проекту.

Для Angular:

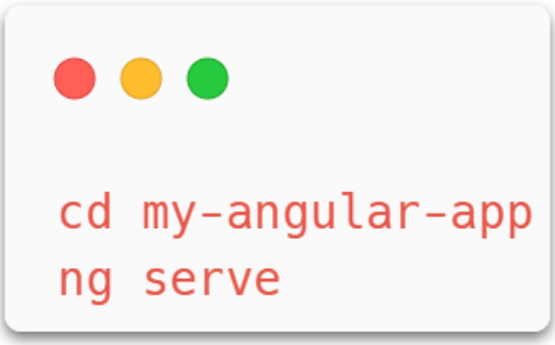
1. Встановити Angular CLI глобально за допомогою npm:



2. Створити новий Angular-проект:



3. Перейти до каталогу проекту і запустити додаток:

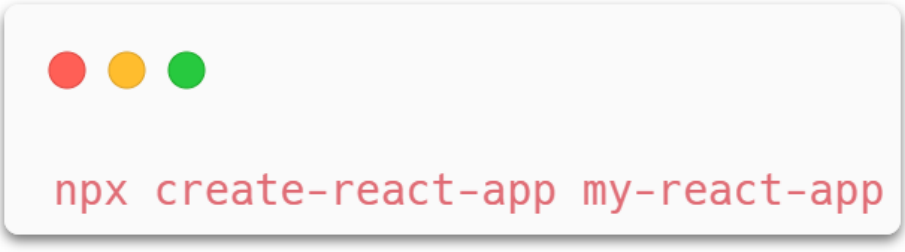


За замовчуванням Angular-додаток буде доступний за адресою http://localhost:4200.

Для React:

1. Встановити create-react-app глобально:

2. Створити новий React-додаток:

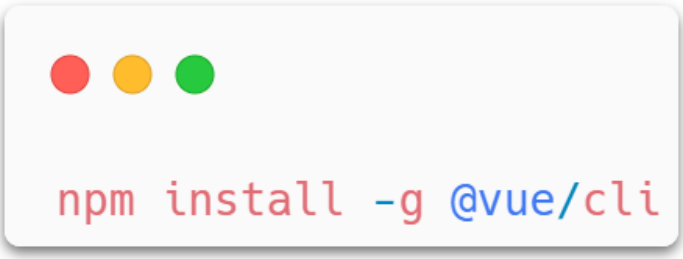


3. Перейти до каталогу проекту і запустити сервер розробки:

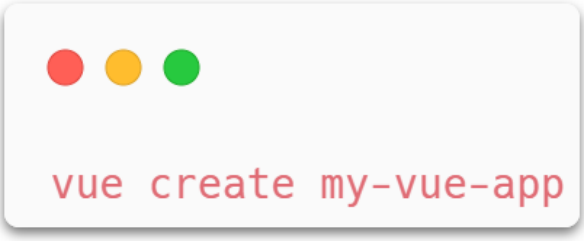
За замовчуванням ваш React-додаток буде доступний за адресою http://localhost:3000.

Для Vue:

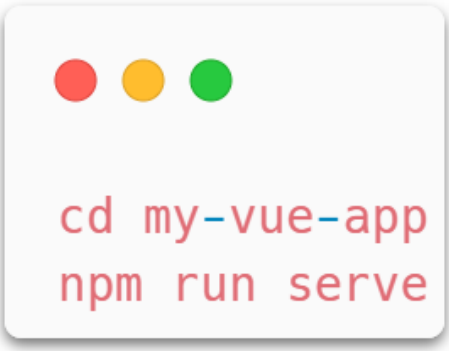
1. Встановити Vue CLI глобально за допомогою npm:



2. Створити новий проект Vue:



3. Перейтт до каталогу проекту і запустіть сервер розробки:



За замовчуванням додаток Vue буде доступний за адресою http://localhost:8080.

Завдання 2: Створення компонент

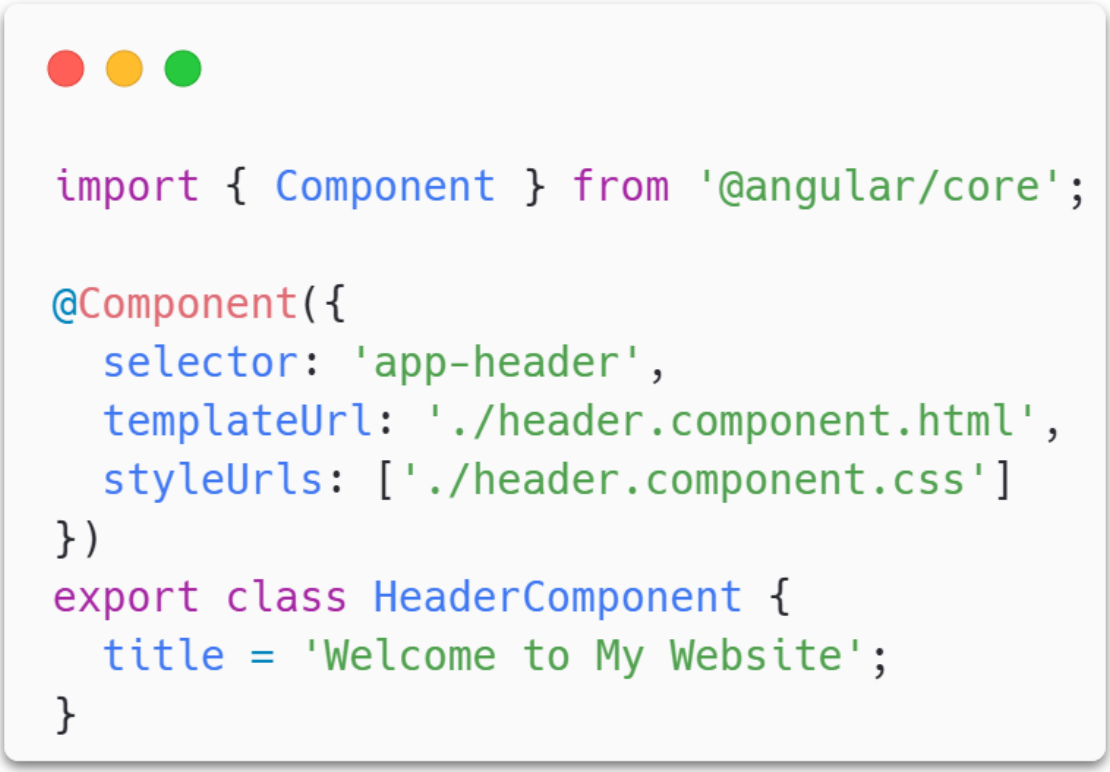
Опис:
Реалізація базової компонентної архітектури з використанням обраного фронтенд-фреймворку.

Кроки:

У Angular:

1. **Створення компонента:** Angular використовує команду ng generate component для створення нового компонента.
2. ng generate < заголовок компонента >
3. **Структура компонента:** Компонент Angular складається з наступних частин:
    - **HTML-шаблон**: Визначає структуру компонента.
    - **CSS**: Стилізує компонент.
    - **TypeScript**: Містить логіку компонента (наприклад, властивості та методи).
    - **Декоратор компонента**: Декоратор @Component визначає метадані для компонента.

Приклад компонента **Angular**:



```
import { Component } from '@angular/core';

@Component({
  selector: 'app-header',
  templateUrl: './header.component.html',
  styleUrls: ['./header.component.css']
})
export class HeaderComponent {
  title = 'Welcome to My Website';
}
```

У представленому прикладі заголовок компонента визначається у файлі TypeScript і може бути відображений у шаблоні за допомогою прив'язки даних Angular.

У React:

1. **Створення компонента:** React використовує JavaScript або JSX для визначення компонентів. Функціональний компонент виглядає так:

```
function Header() {
  return <h1>Welcome to My Website</h1>;
}
```

2. **Структура компонента React:**
    - **JSX**: HTML-подібний синтаксис, що використовується в JavaScript.
    - **Властивості (props)**: Використовується для передачі даних компонентам.
    - **Стан (state)**: Використовується для керування даними, специфічними для компонента.

Приклад передачі даних через властивості в React:

```
function Header(props) {
  return <h1>{props.title}</h1>;
}

// Usage
<Header title="Welcome to My Website" />
```

У Vue:

1. **Створення компонента:** Компоненти Vue зазвичай створюються в одному файлі (файл .vue), який містить шаблон, скрипт і стилі разом.

Приклад базового компонента Vue:

```
<template>
  <h1>{{ title }}</h1>
</template>

<script>
export default {
  data() {
    return {
      title: 'Welcome to My Website'
    };
  }
};
</script>

<style scoped>
h1 {
  color: blue;
}
</style>
```

2. **Структура компонента Vue:**
    - **Шаблон**: Визначає структуру HTML.
    - **Скрипти (script)**: Містить логіку JavaScript компонента.
    - **Стилі (style)**: Визначає стилі CSS (можуть бути застосовані до компонента).

Завдання 3: Керування станом у компонентах

Опис:
Керування станом всередині компонентів та передача даних між ними за допомогою системи керування станом фреймворку.

Кроки:

У Angular:

Стан компонента: В Angular керування станом здійснюється в межах класу компонента. Є можливість змінювати властивості компонента, після чого шаблон автоматично оновиться, щоб відобразити нові зміни у властивостях.

Приклад керування станом в Angular:

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  template: `
    <div>
      <h1>{{ title }}</h1>
      <button (click)="changeTitle()">Change Title</button>
    </div>
  `,
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title: string = 'Welcome to My Website';

  changeTitle() {
    this.title = 'Title has been changed!';
  }
}
```

У наведеному прикладі властивість title є частиною стану компонента і може бути оновлена за допомогою методу changeTitle().

У React:

Використання стану в React: React використовує функції useState для керування станом у компонентах.

```
import React, { useState } from 'react';

function Header() {
  const [title, setTitle] = useState('Welcome to My Website');

  const changeTitle = () => {
    setTitle('Title has been changed!');
  };

  return (
    <div>
      <h1>{title}</h1>
      <button onClick={changeTitle}>Change Title</button>
    </div>
  );
}
```

У представленому прикладі useState використовується для визначення стану заголовка, а функція setTitle оновлює його.

In Vue:

Використання стану у Vue: Vue використовує функції data для визначення стану компонента.

```
<template>
  <h1>{{ title }}</h1>
  <button @click="changeTitle">Change Title</button>
</template>

<script>
export default {
  data() {
    return {
      title: 'Welcome to My Website'
    };
  },
  methods: {
    changeTitle() {
      this.title = 'Title has been changed!';
    }
  }
};
</script>
```

У Vue управління станом відбувається всередині функції data(), а зміни автоматично відображаються в шаблоні.

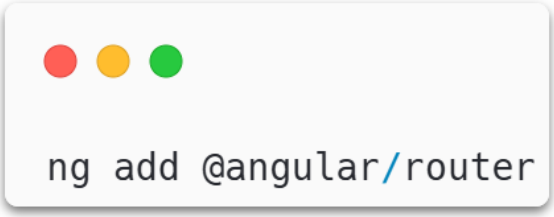
Завдання 4: Реалізація маршрутизації для переходу між сторінками

- **Опис:**
  Реалізація маршрутизації для навігації між різними представленнями (сторінками) веб-додатку.
- **Кроки:**

  1. Реалізувати **маршрутизатор** у вибраному фреймворку (наприклад, React Router для React, RouterModule для с або Vue Router для Vue).

  Для Angular:

  1. Переконатись, що Angular CLI встановлений і Angular-проект створений.
  2. Додати Angular Router (попередньо встановлений з Angular):

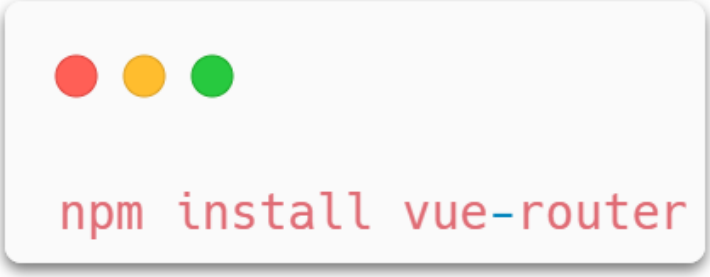


  Для React:

  1. Переконатись, що React-проект створений.
  2. Встановити React Router:

  Для Vue:

  1. Переконатись, що проект Vue створений.
  2. Встановити Vue Router:



2. Створити принаймні **два маршрути** для веб-сайту, наприклад, домашню сторінку та сторінку "Про компанію".

   У Angular:

   1. Відкрити app.routes.ts і визначити маршрути:

```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { HomeComponent } from './home/home.component';
import { AboutComponent } from './about/about.component';

const routes: Routes = [
  { path: '', component: HomeComponent },
  { path: 'about', component: AboutComponent }
];
```

   2. Імпортувати RouterModule та RouterOutlet у кореневий компонент, який використовує (обгортку домашньої сторінки та сторінки About) ці маршрути (наприклад, компонент app.component):

```
@Component({
  selector: 'app-root',
  imports: [RouterOutlet, RouterModule],
  templateUrl: './app.component.html',
  styleUrl: './app.component.css'
})
export class AppComponent {}
```

   3. Додати роутер в шаблон компонента .html, де будуть відображатись дочірні компоненти:

```
<router-outlet></router-outlet>
```

   4. Використати маршрутизатори в компонентах:

```
<a routerLink="/">Home</a>
<a routerLink="/about">About</a>
```

У React:

1. Визначити маршрути за допомогою react-router-dom:

```jsx
import { BrowserRouter as Router, Routes, Route, Link } from 'react-router-dom';
import Home from './Home';
import About from './About';

function App() {
  return (
    <Router>
      <nav>
        <Link to="/">Home</Link>
        <Link to="/about">About</Link>
      </nav>
      <Routes>
        <Route path="/" element={<Home />} />
        <Route path="/about" element={<About />} />
      </Routes>
    </Router>
  );
}

export default App;
```

In Vue:

1. Визначити маршрути в router.js:

```jsx
import { createRouter, createWebHistory } from 'vue-router';
import Home from './views/Home.vue';
import About from './views/About.vue';

const routes = [
  { path: '/', component: Home },
  { path: '/about', component: About }
];

const router = createRouter({
  history: createWebHistory(),
  routes
});

export default router;
```

2. Зареєструвати маршрутизатор як плагін, викликавши функцію use() у додатку в файлі main.js:

```
import { createApp } from 'vue'
import App from './App.vue'
import router from "@/router";

createApp(App).use(router).mount('#app');
```

3. Додати роутер в App.vue:

```
<template>
  <nav>
    <router-link to="/">Home</router-link>
    <router-link to="/about">About</router-link>
  </nav>
  <router-view></router-view>
</template>
```

3. Додати навігаційні посилання (наприклад, навігаційну панель), щоб користувачі могли перемикатися між сторінками.
4. Переконатись, що коли користувач натискає на посилання, вміст динамічно оновлюється на основі шляху адресної стрічки браузера.

Контрольні запитання

1. Що таке компонентно-орієнтована архітектура у фронтенд-фреймворках, і які її переваги?
2. Які ключові відмінності між Angular, React та Vue у способі керування станом компонента?
3. Що таке Virtual DOM і як він покращує продуктивність у React і Vue?
4. Як працює маршрутизація у фронтенд-фреймворках, і які бібліотеки використовуються для неї у Angular, React та Vue?
5. Що таке директиви у Angular та Vue, і як вони відрізняються від умовного рендерингу у React?

Вимоги до оформлення звіту

1. Увійти в папку проекту

2. Створити нову папку з назвою **lab3** і помістити всі файли створені під час виконання лабораторної роботи в дану папку.
3. Завантажити зміни у репозиторій

Варіанти завдань

Варіант 1

Веб-сайт ресторану

1. Створити компоненти для меню, окремих страв у меню, блоку інгредієнтів страви та відгуків про ресторан.
2. У компоненті Меню додати кнопку для сортування страв у меню за ціною (спадання/зростання) та реалізувати логіку сортування. Відображати компоненти пунктів меню у порядку, що відповідає вибраному типу сортування.
3. Реалізувати маршрутизацію переходів між інформацією про меню, список кухарів та інформацією про ресторан без перезавантаження сторінки в браузері.

Варіант 2

Веб-сайт організації онлайн-ігор

1. Створити компоненти для картки гри, картки турніру та блоку статистики гравця.
2. У компоненті "Ігри" додати кнопку для сортування ігор за рейтингом (від найпопулярніших до найменш популярних).
3. Реалізувати маршрутизацію між сторінками без перезавантаження сторінки браузера.

Варіант 3

Веб-сайт платформи для навчальних курсів

1. Створити компоненти для картки курсу, списку занять у розкладі та блоку прогресу студента.
2. У компоненті "Курси" додати кнопку для сортування курсів за тривалістю (від найкоротших до найдовших). Також, використати групу випадаючих списків (select) для відображення категорій курсів та їх фільтрації за вибраною категорією.
3. Реалізувати маршрутизацію між сторінками без перезавантаження сторінки браузера.

Варіант 4

Веб-сайт платформи для бронювання турів

1. Створити компоненти для картки туру, списку заброньованих турів та інтерактивної мапи.
2. У компоненті "Гарячі тури" додати кнопку для сортування турів за ціною (від дешевших до дорожчих).
3. Реалізувати маршрутизацію між сторінками без перезавантаження сторінки браузера.

Варіант 5

Веб-сайт онлайн-магазину книг

1. Створити компоненти для картки книги, елемента кошика та блоку інформації користувача.
2. У компоненті "Каталог" додати кнопку для фільтрації книг за жанром або автором.
3. Реалізувати маршрутизацію між сторінками без перезавантаження сторінки браузера.

Варіант 6

Веб-сайт платформи для оренди автомобілів

1. Створити компоненти для картки автомобіля, списку бронювань та блоку інформації про компанію.
2. У компоненті "Автомобілі" додати кнопку для фільтрації автомобілів за ціною, типом трансмісії та наявністю.
3. Реалізувати маршрутизацію між сторінками без перезавантаження сторінки браузера.

Варіант 7

Веб-сайт платформи для оренди житла

1. Створити компоненти для картки квартири, списку заброньованих квартир та інтерактивної мапи.
2. У компоненті "Доступні квартири" додати фільтри для сортування за ціною, кількістю кімнат і типом квартири.
3. Реалізувати маршрутизацію між сторінками без перезавантаження сторінки браузера.

Варіант 8

Веб-сайт платформи для онлайн-бронювання квитків на заходи

1. Створити компоненти для картки події, форми бронювання та списку бронювань.
2. У компоненті "Події" додати фільтри для сортування подій за датою та типом заходу (концерти, театри, виставки тощо).
3. Реалізувати маршрутизацію між сторінками подій, моїх бронювань та інформації про організаторів без перезавантаження сторінки.

Варіант 9

Веб-сайт платформи для організації волонтерських ініціатив

1. Створити компоненти для картки ініціативи, форми реєстрації на ініціативу та списку приєднаних ініціатив.
2. У компоненті "Доступні ініціативи" додати фільтри для сортування проєктів за датою, місцем та типом активності (екологія, допомога тваринам, соціальна підтримка тощо).
3. Реалізувати маршрутизацію між сторінками ініціатив, моїх ініціатив та інформації про організацію без перезавантаження сторінки.

Варіант 10

Веб-сайт онлайн-магазину спортивних товарів

1. Створити компоненти для картки товару, кошика та блоку акцій.
2. У компоненті "Продукти" додати фільтри для сортування товарів за ціною та рейтингом.
3. Реалізувати маршрутизацію між сторінками продуктів, кошика, акцій та профілю без перезавантаження сторінки.

Варіант 11

Веб-сайт платформи для пошуку робочих вакансій

1. Створити компоненти для картки вакансії, профілю користувача та блоку пошуку роботи.
2. У компоненті "Вакансії" додати кнопку для сортування вакансій за датою публікації (від найновіших до найстаріших).
3. Реалізувати маршрутизацію між сторінками без перезавантаження сторінки браузера.

Варіант 12

Веб-сайт платформи для замовлення їжі

1. Створити компоненти для картки страви, кошика та списку замовлень.
2. У компоненті "Меню" реалізувати фільтрацію страв за категоріями (піца, суші, напої тощо).
3. Реалізувати маршрутизацію між сторінками без перезавантаження сторінки браузера.

Варіант 13

Веб-сайт блогу для подорожей

1. Створити компоненти для картки статті, форми коментарів та списку коментарів.
2. У компоненті "Статті" додати сортування за датою публікації.
3. Реалізувати маршрутизацію між сторінками без перезавантаження сторінки браузера.

Варіант 14

Веб-сайт інтерактивної платформи для вивчення мов

1. Створити компоненти для картки уроку, графіку прогресу та тестів.
2. У компоненті "Уроки" реалізувати фільтрацію матеріалів за рівнем складності (A1, A2, B1, B2 тощо).
3. Реалізувати маршрутизацію між сторінками без перезавантаження сторінки браузера.

Варіант 15

Веб-сайт платформи для вивчення історії через інтерактивні події

1. Створити компоненти для хронології, картки події, тесту та графіку прогресу.
2. У компоненті "Хронологія" реалізувати фільтрацію за періодами (Стародавній світ, Середньовіччя, Новий час тощо).
3. Реалізувати маршрутизацію між сторінками без перезавантаження сторінки браузера.

Варіант 16

Веб-сайт симулятора управління власним стартапом

1. Створити компоненти для відображення звітів, конкурентів та інвесторів.
2. У компоненті "Ринок" реалізувати фільтрацію конкурентів за сферою діяльності.
3. Реалізувати маршрутизацію між сторінками без перезавантаження сторінки браузера.

Варіант 17

Веб-сайт платформи для створення та управління персональними цілями

1. Створити компоненти для картки цілі, прогресу та коментарів у спільноті.
2. У компоненті "Мої цілі" реалізувати фільтрацію за статусом (активні, завершені, відкладені).
3. Реалізувати маршрутизацію між сторінками без перезавантаження сторінки браузера.

Варіант 18

Веб-сайт платформи для віртуальних хакатонів та змагань із програмування

1. Створити компоненти для картки хакатону, списку учасників та результатів.
2. У компоненті "Змагання" реалізувати фільтрацію за категоріями (штучний інтелект, веб-розробка, кібербезпека тощо).
3. Реалізувати маршрутизацію між сторінками без перезавантаження сторінки браузера.

Варіант 19

Веб-сайт симулятора управління будівництвом містом

1. Створити компоненти для картки будівлі, бюджету та індикатора задоволеності жителів.
2. У компоненті "Моє місто" реалізувати фільтрацію доступних об'єктів за категоріями (житлові, комерційні, промислові).
3. Реалізувати маршрутизацію між сторінками без перезавантаження сторінки браузера.

Варіант 20

Веб-сайт симулятора космічної експедиції

1. Створити компоненти для корабля, місій та журналу подорожей.
2. У компоненті "Експедиції" реалізувати фільтрацію місій за складністю та типом (дослідницькі, рятувальні, колонізація).
3. Реалізувати маршрутизацію між сторінками без перезавантаження сторінки браузера.

Варіант 21

Веб-сайт платформи для моніторингу здоров'я та фізичної активності

1. Створити компоненти для картки тренування, графіку прогресу та плану харчування.
2. У компоненті "Тренування" додати фільтрацію за типами тренувань (кардіо, силові, йога тощо).
3. Реалізувати маршрутизацію між сторінками без перезавантаження сторінки браузера.

Варіант 22

Веб-сайт платформи для онлайн-курсу з фотографії

1. Створити компоненти для картки уроку, галереї та прогресу користувача.
2. У компоненті "Уроки" додати можливість відмічати пройдені уроки.
3. Реалізувати маршрутизацію між сторінками без перезавантаження сторінки браузера.

Варіант 23

Веб-сайт платформи для оренди спортивного обладнання

1. Створити компоненти для картки обладнання, списку оренд та форми оплати.
2. У компоненті "Обладнання" реалізувати фільтрацію за типами спорту (футбол, теніс, лижі тощо).
3. Реалізувати маршрутизацію між сторінками без перезавантаження сторінки браузера.

Варіант 24

Веб-сайт платформи для планування подорожей

1. Створити компоненти для картки місця для відвідування, списку подорожей та бюджету.
2. У компоненті "Місця для відвідування" реалізувати фільтрацію за типом (пляжі, гірські курорти, міста тощо).
3. Реалізувати маршрутизацію між сторінками без перезавантаження сторінки браузера.

Варіант 25

Веб-сайт платформи для створення та обміну рецептами

1. Створити компоненти для картки рецепту, форми для додавання нового рецепту та списку коментарів.
2. У компоненті "Категорії" реалізувати фільтрацію рецептів за типом страви.
3. Реалізувати маршрутизацію між сторінками без перезавантаження сторінки браузера.