

FRONTEND DEVELOPER

Практика



Зміст:

| | |
|--|----|
| Вступ | 3 |
| Теми для курсового проєкту. Курс HTML5 & CSS3 | 4 |
| Особисте резюме | 4 |
| Верстка сайту, використовуючи технологію flex-box: | 5 |
| Верстка сайту, використовуючи технологію Grid Layout CSS: | 5 |
| Теми для курсового проєкту. Курс JavaScript Essential | 8 |
| Теми для курсового проєкту. Курс JavaScript Advanced..... | 12 |
| Прогноз погоди | 12 |
| Теми для курсових проєктів. Курс HTML5 & CSS3 Advanced | 15 |
| Сайт піцерії | 15 |
| Теми для курсових проєктів. Курси Angular/TypeScript або React | 17 |
| Інтернет-магазин | 17 |
| Онлайн-кінотеатр | 19 |
| Інтернет-магазин | 20 |
| Web-застосунок «Піцерія» | 23 |
| Менеджер паролів | 24 |
| ToDo List | 25 |
| Книга контактів | 26 |

Вступ

У цьому документі зібрані рекомендації та варіанти курсових проєктів як для окремих курсів, так і для написання підсумкової роботи. Ви можете за бажанням обрати свою тему для курсового проєкту, погодивши її з тренером.

Вам потрібно вибрати та захистити як мінімум один підсумковий проєкт.

Ми рекомендуємо зробити максимально можливу кількість проєктів, але реальна кількість курсових проєктів залежатиме від Вашої успішності або кількості часу, яку Ви можете виділити на роботу над проєктами.

Для зберігання вихідного коду використовуйте репозиторій на <https://github.com>.

Пам'ятайте, що для розв'язку багатьох завдань, які описані в цьому документі та з якими ви зіткнетеся під час роботи, вам будуть потрібні додаткові навички та інструменти, вивчення застосування яких виходить за межі навчальної програми. Самостійне вивчення та застосування технік, які описані в додаткових матеріалах до завдань, будуть великим плюсом.

Теми для курсового проєкту. Курс HTML5 & CSS3

Особисте резюме

Мета: продемонструвати навички верстки, застосувати адаптивну верстку.

Обов'язкові мови/технології/бібліотеки: HTML, CSS.

Додаткові мови/технології/бібліотеки: SASS, Bootstrap.

Додаткові матеріали:

[Верстаємо сайт правильно](#)

[Робота з GIT](#)

[Верстка сайту за 30 хвилин на Flexbox](#)

Вимоги до готового рішення:

- Вихідний код має бути розміщений на github або іншому подібному сервісі.
- Стили мають бути винесені в окремий файл зображення з осмисленими іменами, які мають зберігатися в окремій теці поряд з HTML-розміткою.
- Ваша сторінка має коректно зображатися на роздільній здатності 1280 і більше. Як ускладнення завдання реалізуйте адаптивну верстку (якщо виконуєте це завдання після освоєння цієї теми або самостійно намагаєтеся освоїти медіазапити).
- Основні роздільні здатності:
 - 320 px – мобільний пристрій (портретна орієнтація);
 - 480 px – мобільний пристрій (альбомна орієнтація);
 - 600 px – невеликі планшети;
 - 768 px – планшети (портретна орієнтація);
 - 1024 px – планшети (альбомна орієнтація);
 - 1280 px та більше – РС.
- У резюме необхідно ввести ваші дані.
- Обов'язкові розділи резюме:
 - фото та ПІБ кандидата;
 - ціль;
 - блок із контактними даними;

- блок із досвідом роботи – у зворотному хронологічному порядку місця роботи, дати роботи, посада, опис обов'язків тощо;
 - блок з освітою;
 - блок із сертифікатами – із зображеннями сертифікатів та їхніх номерів, а також з посиланнями (за можливістю);
 - блок з хобі й особистими якостями;
 - інші блоки за бажанням.
- Можна вибрати свій стиль шаблону, але бажано виконати верстку за макетом за посиланням:
https://drive.google.com/drive/folders/1IdpTV4grpW_TPPlrq5NguY90nEDMpGH?usp=sharing

Верстка сайту, використовуючи технологію flex-box:

Лендінг бронювання квитків на подорож:

<https://www.figma.com/file/iodvXqdkR3FN1VF5UtL1Wg/Explore?t=xoM1D7ho4c4SkzcD-0>

Лендінг дизайну кімнат:
<https://www.figma.com/file/0SMIFh0ruD8sXtSpN8gP9T/RoomsDesign?node-id=0%3A1&t=STaAIqNnk87MKGIU-0>

Верстка сайту, використовуючи технологію Grid Layout CSS:

Лендінг SpaceX:

<https://www.figma.com/file/UF2WHFWJGs7p4NQBoVIfnF/SpaceX?node-id=0%3A1&t=Rde3tH0Bwyx9oERb-0>

Лендінг

ювелірних

виробів:

<https://www.figma.com/file/RhiOXtfujdpqgt5wkMYEkN/Jewelery?node-id=0%3A1&t=iQTqSzlaQMvwJixk-0>

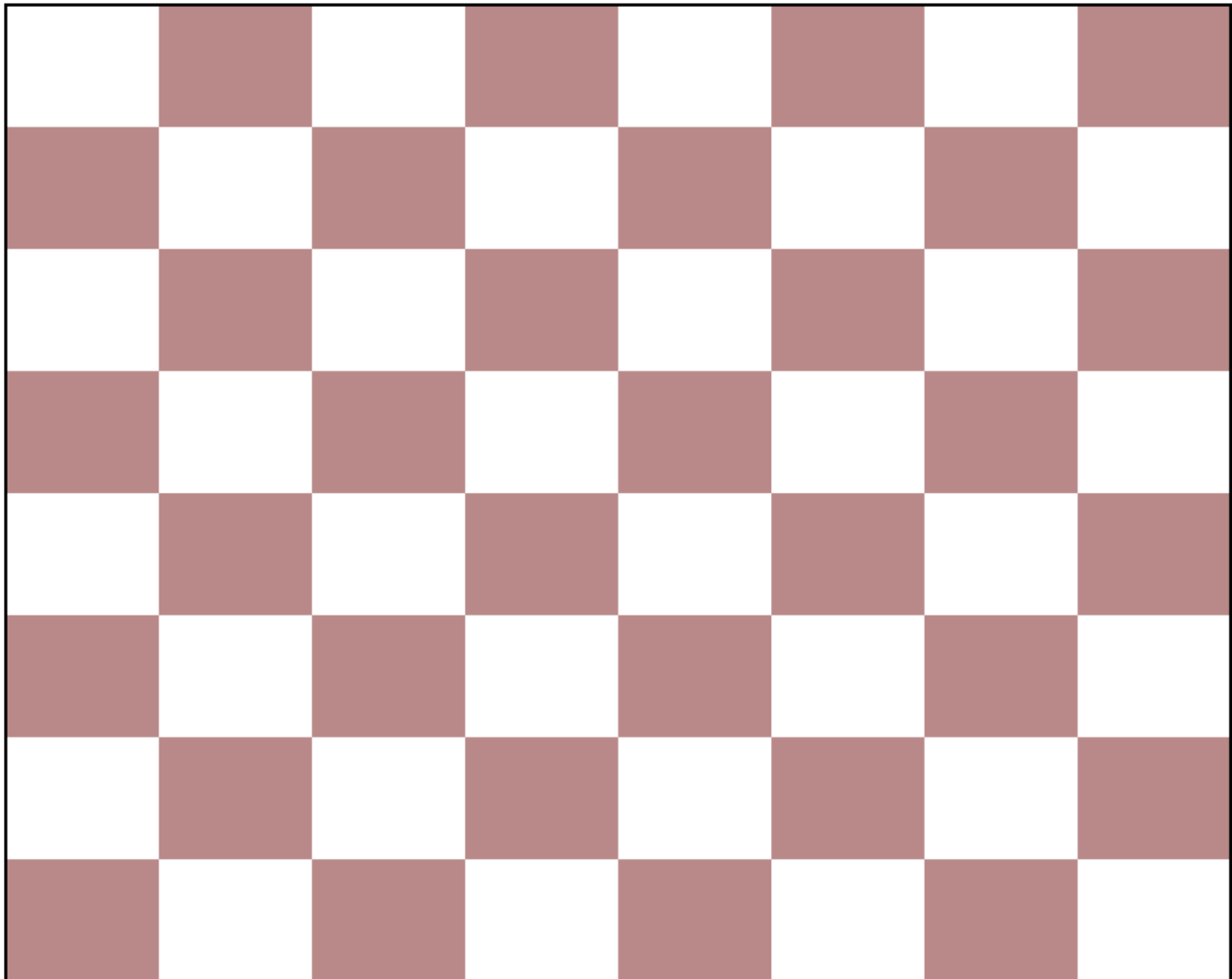
CSS:

1. Розробити аналог пошукової сторінки Google.

Дана сторінка повинна містити в собі:

- 1) Логотип Google;
- 2) Пошуковий рядок вгорі;
- 3) Гіперпосилання.

2. Розробити шахову дошку, розміром 8x8



background: rgb(185, 137, 137); // фоновий колір кольорової клітинки

HTML:

Зверстати таблиці використовуючи семантику таблиць:

| | | | | | | | |
|--|--|--|--|--|--|--|--|
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |

| | | | | | | | |
|--|--|--|--|--|--|--|--|
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |

Теми для курсового проєкту. Курс JavaScript Essential

Мета: продемонструвати навички використання базових конструкцій JavaScript.

Обов'язкові мови: JavaScript.

Оскільки завдання курсу JavaScript Essential не включають повноцінної взаємодії з API браузера, завдання у цьому розділі не є повноцінними курсовими проєктами, а додатковими завданнями для самостійної роботи. Перед виконанням цих завдань вирішіть усі домашні завдання, які входять до курсу JavaScript Essential.

Завдання 1

Створіть функцію, яка визначатиме, чи є введений користувачем рядок паліндромом.

Паліндром – число, буквосполучення, слово або текст, яке читається однаково зліва направо та справа наліво. Наприклад, 707, Пилип, око, мадам тощо.

Завдання 2

Напишіть функцію, яка порівнює два рядки та визначає, чи є вони анаграмою (слова складаються з одних і тих самих букв). Рядки отримуємо від користувача за допомогою 2-х методів prompt.

Приклад:

- перше слово – апельсин;
- друге слово – липень;

Результат: слова «апельсин» та «липень» є анаграмою.

Завдання 3

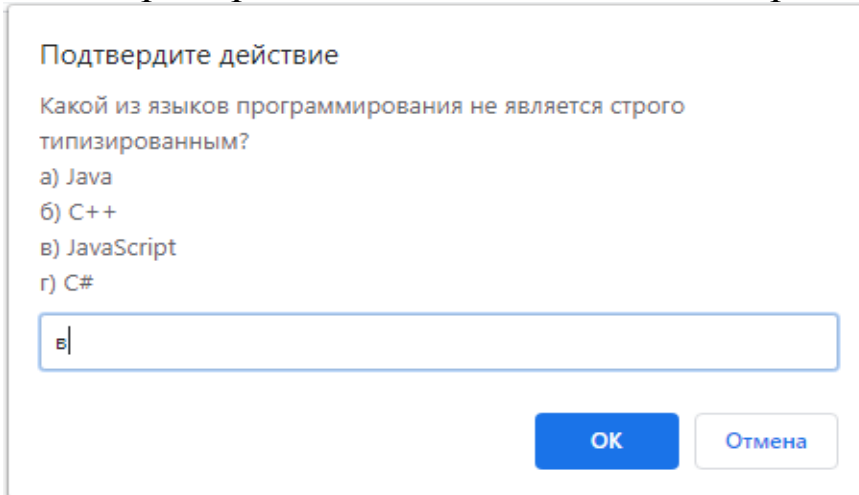
Гра «Як стати мільйонером».

Подумайте, як можна реалізувати подібний сценарій з використанням масивів та об'єктів. Пам'ятайте, що для кожного запитання потрібно прописати: запитання, 4 варіанти відповіді, правильну відповідь (яку порівнюватимемо з тим, що ввів користувач), сума, яку виграв користувач. Визначте щонайменше 5

питань. Запитання з варіантами відповідей ви можете вигадати самостійно, а можете скористатися пошуком.

Як має виглядати реалізація:

1. У вікні prompt виводяться запитання та варіанти відповідей.



Подтвердите действие

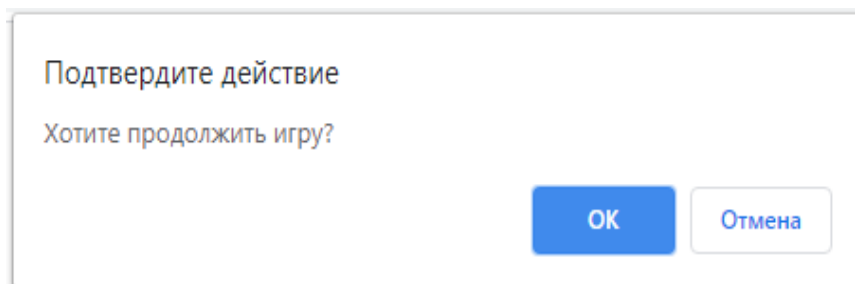
Какой из языков программирования не является строго типизированным?

а) Java
б) C++
в) JavaScript
г) C#

в|

ОК Отмена

2. Якщо користувач вводить правильну відповідь (у вигляді літери), то виводиться вікно confirm із запитанням: «Чи бажаєте продовжити гру?».



Подтвердите действие

Хотите продолжить игру?

ОК Отмена

3. Якщо «ОК», тоді продовжуємо. Користувачеві виводиться наступне запитання. Якщо ні, виводимо суму, яку виграв користувач, і виходимо з гри.
4. Якщо користувач вводить неправильну відповідь, виводимо користувачеві інформацію, що він вказав неправильну відповідь, і виходимо з гри.

Додаткове завдання

Збільште кількість запитань до 15 і додайте вогнетривкі суми на 5 і 10 запитаннях (\$5000 і \$25000 відповідно). Це означає, що якщо користувач відповів, наприклад, на 7 запитань і «заробив» \$10000, а на 8-е відповів неправильно, то гарантовано отримає \$5000.

5.

| Номер запитання | Приз |
|-----------------|------------------|
| 1 | 500 |
| 2 | 1 000 |
| 3 | 2 000 |
| 4 | 3 000 |
| 5 | 5 000 |
| 6 | 8 000 |
| 7 | 10 000 |
| 8 | 13 000 |
| 9 | 15 000 |
| 10 | 25 000 |
| 11 | 50 000 |
| 12 | 100 000 |
| 13 | 250 000 |
| 14 | 500 000 |
| 15 | 1 000 000 |

6. Написати функцію `upString(str)`, яка приймає рядок, та переводить усі символи, котрі були прописані у нижньому регістрі у верхній регістр.

Приклад:`upString('hello'); // HELLO`

7. Написати ф-ю `palindrom(str)`, яка буде перевіряти, чи є даний рядок паліндромом чи ні.

Примітка: *слово є паліндромом, якщо воно читається з ліворуч та праворуч однаково.*

Приклад: `palindrom('olo');` // true, `palindrom('hello');` // false, not palindrom

8. Реалізувати використовуючи HTML, CSS, JS додаток «Запис на прийом до лікаря»

У даному додатку передбачити форму запису на прийом, вона повинна містити:

- Поля для введення ПІБ;
- Поле для введення телефону;
- Вибір статі (реалізувати за допомогою `radiobutton`);
- Поля для обрання дати та часу;
- Кнопку «Оформити візит».

Дані з полів можна зберігати у **localStorage**.

9. Написати функцію `number(from, to)`, дана функція повинна виводити число кожну секунду, починаючи від `from` до `to`

! Зробити використовуючи `setInterval()`, а також використовуючи вкладений `setTimeout()`

- 10.

Теми для курсового проєкту. Курс JavaScript Advanced

Прогноз погоди

Мета: застосувати ООП та інші техніки на JavaScript.

Обов'язкові мови/технології/бібліотеки: HTML, CSS, JavaScript.

Додаткові мови/технології/бібліотеки: SASS, Bootstrap, jQuery, TypeScript.

Додаткові матеріали:

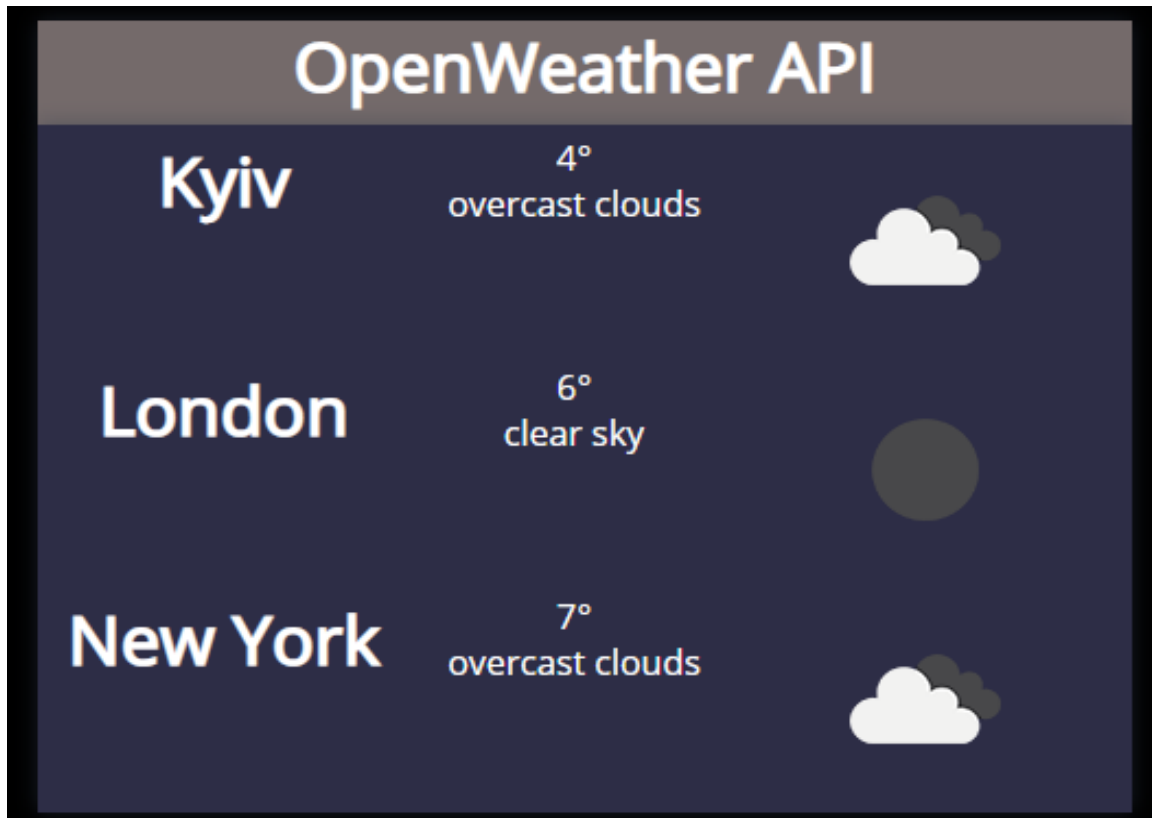
[JavaScript-шаблони](#)

[ECMAScript 6 – нові інструменти для JavaScript-розробника](#)

[Асинхронне програмування у JavaScript](#)

Вимоги до готового рішення:

- Застосунок має показувати поточну погоду щонайменше за трьома містами: Київ, Лондон, Нью-Йорк.
- Оформлення застосунку довільне, акцент потрібно зробити на структуру JavaScript-коду. Розділіть логіку роботи з API та інтерфейсом за окремими об'єктами. Не намагайтеся розмістити весь код в одному файлі, розбийте логіку на кілька файлів.
- Перевагою буде використання шаблону модуль. Докладніше про цей шаблон ви можете дізнатися у цьому відеоуроці: <https://itvdn.com/ru/video/javascript-patterns/create-patterns>.
- Схематично структура застосунку може виглядати так:



- Застосунок має зображати погоду, виконуючи запити до погодних сервісів. Докладна документація є на сайті <https://openweathermap.org/>.
- Посилання для отримання даних:

Київ

<http://api.openweathermap.org/data/2.5/weather?id=703448&appid=bf35cac91880cb98375230fb443a116f>.

Лондон

<http://api.openweathermap.org/data/2.5/weather?id=2643743&appid=bf35cac91880cb98375230fb443a116f>.

Нью-Йорк

<http://api.openweathermap.org/data/2.5/weather?id=5128638&appid=bf35cac91880cb98375230fb443a116f>.

- Зобразіть відповідну іконку для погоди, список ви можете знайти у документації за посиланням: <https://openweathermap.org/weather-conditions>.
- Для розміщення зображення використовуйте звернення за спеціальною адресою:
[http://openweathermap.org/img/wn/'"+data.weather\[0\]\['icon'\]+'@2x.png](http://openweathermap.org/img/wn/')
data – об'єкт, який було повернено із сервера з інформацією про погоду. Можете скористатися іншою логікою та підібрати свої зображення для погоди.
- Додайте можливість перемикати налаштування виведення температури в градусах за Цельсієм або за Фаренгейтом.
- Додаткові завдання:
 - Додайте висновок додаткових даних про погоду – напрям вітру, швидкість вітру, тиск, час сходу та заходу сонця.
 - Додайте можливість переглядати прогноз погоди на наступні кілька днів у таблиці для обраного міста. Ознайомтеся з документацією за посиланням: <https://openweathermap.org/>. Передбачте відповідні елементи інтерфейсу користувача, які дадуть можливість перемикати зображення між поточною погодою та прогнозом погоди.

Теми для курсових проєктів. Курс HTML5 & CSS3 Advanced

Сайт піцерії

Мета: продемонструвати навички верстки за макетом, застосувати адаптивну верстку, закріпити на практиці теми курсів з HTML і JavaScript.

Обов'язкові мови/технології/бібліотеки: HTML, CSS, SASS.

Додаткові мови/технології/бібліотеки: Bootstrap.

Додаткові матеріали:

[Верстаємо сайт правильно](#)

[Робота з GIT](#)

[Адаптивна верстка за допомогою FlexBox і Grid](#)

[Верстка сайту за 30 хвилин на Flexbox](#)

[Верстка з використанням Gulp, JavaScript, HTML/CSS. Частина 1](#)

[Верстка з використанням Gulp, JavaScript, HTML/CSS. Частина 2](#)

[Верстка з використанням Gulp, JavaScript, HTML/CSS. Частина 3](#)

Вимоги до готового рішення:

- Вихідний код має бути розміщений на github або іншому подібному сервісі.
- Перевірте сайт піцерії за макетом у файлі Pizza-Layout.zip. Усі сторінки мають бути максимально наближеними до макета.
- Продумайте структуру файлів і тек, використовуйте правильні та зрозумілі імена.
- Використовуйте адаптивну верстку.
- Використовуйте шрифти <https://fonts.google.com/>.
- Панель навігації (верхнє меню) має бути закріпленим і рухатися разом із сайтом.
- На сторінці «Про нас» розмістіть відео, використовуючи елемент video.
- Сайт має складатися з кількох сторінок: «Головна», «Меню», «Замовлення», «Про нас». На кожній сторінці має бути хедер з меню та футер, як на головній сторінці. Контент кожної сторінки свій і представлений у макеті. Меню сайту має виконувати переходи на відповідні сторінки.

- Форма на сторінці замовлення має перевіряти коректність введених даних і показувати користувачу помилки, якщо він їх припустився. Відправляти кудись дані з форми не потрібно, у разі коректного введення можна просто показати повідомлення про успішне відправлення форми.
- Не забувайте про семантичну верстку. На сторінках має бути один заголовок H1, HTML-елементи мають відповідати вмісту, блоки тексту – p, списки – ul або ol, статті – article тощо.
- Великою перевагою буде використання SASS. Великим плюсом буде використання Gulp. Додатково про Gulp та верстку ви можете дізнатися у серії вебінарів: <https://itvdn.com/ru/webinars/description/webinar-lending1>.

Теми для курсових проєктів. Курси Angular/TypeScript або React

Інтернет-магазин

Мета: продемонструвати навички створення SPA-застосунків, закріплення на практиці фреймворків і бібліотек, вивчених на курсі.

Обов'язкові мови/технології/бібліотеки: на вибір – Angular або React.

Додаткові мови/технології/бібліотеки: Gulp, Webpack, Angular CLI.

Додаткові матеріали:

[Angular 4 Jump Start](#)

[Створення першого проєкту на Angular](#)

Вимоги до завдання:

- Реалізуйте застосунок як SPA.
- Застосунок має складатися з таких розділів: «Каталог», «Кошик», «Панель адміністрування».
- Інтерфейс інтернет-магазину робіть на власний розсуд, можете використовувати стандартні стилі bootstrap або material design. Основний акцент зробіть на описі бізнес-логіки та структурі проєкту.
- Розділіть бізнес-логіку та логіку роботи з інтерфейсом користувача. Робота із зовнішнім API має бути винесена в окремі сервіси. Використовуйте впровадження залежностей для Angular або імпорту для React.
- Виділіть модель/компонент для роботи з даними. Як мінімум, у Вас мають бути класи product і cart, але переглядаючи документацію, можливо, ви додасте інші класи, наприклад, user.
- Як серверну логіку використовуйте Fake Store API. Посилання на документацію <https://fakestoreapi.com/docs>.

Вимоги до каталогу:

- Зображає список продуктів і ціни. Інформація на запит до сервера: <https://fakestoreapi.com/docs#p-all>.
- Дає змогу фільтрувати каталог за категоріями продукту. Виводити продукти за зростанням та за зменшенням ціни.
- Можна переглянути деталі кожного продукту – опис, ціну категорії тощо.
- З каталогу та зі сторінки перегляду деталей про продукт можна додати продукт до кошика.
- Реалізуйте пагінацію в каталозі.

Вимоги до кошика:

- Кошик має містити продукти, які користувач додав із каталогу.
- Кошик має містити суму всіх продуктів, які були додані.
- Інформація про кількість продуктів у кошику має зображатися у правому верхньому кутку сторінки.

Вимоги до панелі адміністрування

- Цей розділ має використовуватися адміністратором, але буде доступний для всіх користувачів. Налаштування авторизації користувачів не входить до завдання цього проєкту.
- У розділі мають зображатися всі кошики для всіх користувачів. Сума кожного кошика окремо та сума всіх кошиків. Документація: <https://fakestoreapi.com/docs#c-all>.
- Має бути можливість переглянути вміст одного конкретного кошика (побачити список усіх продуктів, які були куплені). Документація: <https://fakestoreapi.com/docs#c-single>.
- Має бути можливість фільтрувати виведення за датами. Адміністратор вказує дати з і до, інформація за кошиками зображається тільки для кошиків, які створених у ці дати. Документація: <https://fakestoreapi.com/docs#c-date>.

Онлайн-кінотеатр

Мета: продемонструвати навички створення SPA-застосунків, закріплення на практиці фреймворків і бібліотек, вивчених на курсі.

Обов'язкові мови/технології/бібліотеки: на вибір – Angular або React.

Додаткові мови/технології/бібліотеки: Gulp, Webpack, Angular CLI.

Додаткові матеріали:

[Angular 4 Jump Start](#)

[Створення першого проєкту на Angular](#)

Вимоги до завдання:

- Реалізуйте застосунок як SPA.
- Застосунок має складатися з таких розділів: «Каталог», «Кошик», «Панель адміністрування».
- Інтерфейс онлайн-кінотеатру робіть на власний розсуд, можете використовувати свої стилі або стандартні стилі bootstrap чи material design. Основний акцент зробіть на опис бізнес-логіки та структуру проєкту.
- Як серверну логіку використовуйте The Movie Database. Посилання на документацію: <https://developers.themoviedb.org/3/getting-started/introduction>.

Вимоги до каталогу:

- Зображає список фільмів та їхній короткий опис. Надає інформацію за запитом до сервера.
- Дає змогу відфільтрувати каталог за популярністю. Виводить фільми за зростанням дати релізу та за спаданням.
- Можна переглянути деталі фільму: опис, категорію тощо.
- З каталогу та зі сторінки перегляду деталей фільму можна додати його до списку бажаних для перегляду.
- Реалізуйте пагінацію в каталозі.

Вимоги до панелі адміністрування

- Цей розділ має використовуватися адміністратором, але буде доступний для всіх користувачів. Налаштування авторизації користувачів не входить до завдань цього проєкту.
- У розділі мають зображатися всі фільми, які є в списку бажаних для перегляду всіх користувачів. Документація: <https://developers.themoviedb.org/3/getting-started/introduction>.
- Має бути можливість переглянути вміст списку одного користувача.
- Має бути можливість фільтрувати виведення за датами. Адміністратор вказує дати з і до, інформація про списки зображається тільки для списків, які створені у ці дати.

Інтернет-магазин

Мета: Продемонструвати навички створення SPA застосунків, закріплення на практиці фреймворків та бібліотек, вивчених на курсі

Обов'язкові мови/технології/бібліотеки: React

Додаткові мови/технології/бібліотеки: Redux, React-Redux, gh-pages, axios, firebase

Вимоги до завдання:

- реалізуйте програму як SPA (single page application);
 - застосунок повинен складатися з розділів – каталог, кошик, панель адміністрування;
 - інтерфейс інтернет-магазину робіть на свій розсуд, можете використовувати стандартні стилі bootstrap або material design.
- Основний акцент зробіть на опис бізнес-логіки та структуру проєкту;

- розділіть бізнес-логіку та логіку роботи з інтерфейсом користувача;
- інтерфейс користувача реалізуйте компонентами React. Дотримуйтесь концепції презентаційних та контейнерних компонентів;
- робота із зовнішнім API має бути винесена в окремі сервіси. Використовуйте імпорт для React;
- реалізуйте роботу з даними відповідно до концепцій React. Використовуйте «моделі» props та state;
- виділіть модель/компонент для роботи з даними. Як мінімум у Вас повинні бути класи product та cart, але переглядаючи документацію, можливо, ви долучите інші класи, наприклад, user;
- в якості серверної логіки використовуйте Fake Store API або Firebase. Посилання на документацію <https://fakestoreapi.com/docs>, <https://firebase.google.com/docs>;
- проект повинен бути розгорнутий та запущений на ресурсі github-pages.

Вимоги до каталогу:

- відображає список продуктів та ціни. Інформація на запит до сервера <https://fakestoreapi.com/docs#p-all> або використовуйте базу даних Firebase, <https://firebase.google.com/products/realtime-database>;
- дозволяє фільтрувати каталог за категоріями продукту. Виводити продукти за зростанням та за зменшенням ціни;

- можна продивитися деталі кожного продукту – опис, категорію, ціну тощо.

- з каталогу та з сторінки перегляду деталей про продукт можна додати продукт до кошику;

- реалізуйте пагінацію у каталозі.

Вимоги до кошика:

- кошик повинен містити продукти, які користувач додав з каталогу;

- кошик повинен містити суму усіх продуктів, які були додані;

- інформація про кількість продуктів у кошику повинна відображатися у правому верхньому кутку сторінки.

Вимоги до панелі адміністрування:

- цей розділ повинен використовуватися адміністратором, але доступний для всіх користувачів. Налаштування авторизації користувачів не входить до завдання цього проекту;

- у розділі повинні відображатися всі кошики для всіх користувачів. Сума кожного кошика окремо та сума всіх кошиків.

Документація <https://fakestoreapi.com/docs#c-all>,
<https://firebase.google.com/docs/database> — має бути можливість переглянути вміст одного конкретного кошика (побачити список усіх продуктів, які були куплені). Документація <https://fakestoreapi.com/docs#c-single>;

- має бути можливість фільтрувати виведення за датами. Адміністратор вказує дати з/до та інформацію по кошикам відображається тільки для кошиків, створених у ці дати. Документація <https://fakestoreapi.com/docs#с-date>.

Рекомендації:

- для реалізації маршрутизації програми React використовуйте React Router DOM;
- для організації керування даними використовуйте бібліотеку Redux;
- для роботи з Firebase використовуйте npm пакет firebase;
- для роботи із запитами на сервер використовуйте npm-пакет axios.

Web-застосунок «Піцерія»

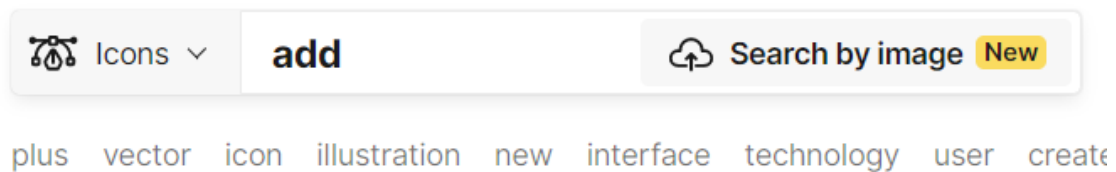
Створити web-додаток, піцерії у якому користувач зможе додати, видалити, оформити замовлення.

Всі дані можна зберігати використовуючи **localStorage**.

Для логотипів можете використовувати даний ресурс:

<https://icons8.com/icons/set/add>

Щоб здійснити пошук необхідного логотипу потрібно у вікні пошуку вказати назву логотипа англійською мовою.



Add New



Add New



Add New



Add New

Приклади піцерій:

- 1) Domino's Pizza: <https://dominos.ua/uk/kyiv/>
- 2) il Molino: <https://ilmolino.ua/pizza/>
- 3) Mr. Cat: <https://mistercat.com.ua/ua/magazin/>

! Використовуйте наведені вище посилання як приклади, поважайте авторські права.

Менеджер паролів

Мета: продемонструвати навички створення SPA-застосунків, закріплення на практиці фреймворків і бібліотек, вивчених на курсі.
Обов'язкові мови/технології/бібліотеки: React.

Додаткові матеріали:

Посилання на макет PasswordManager:

<https://www.figma.com/file/Ylx0ziSRmtcdOGI2pXLGls/PasswordManager?t=ZVamsWnbd22575og-6>

Застосунок «Менеджер паролів (Password Manager)» для управління всіма вашими паролями до різних облікових записів (пошти, пристроїв, акаунтів, серверів тощо).

Застосунок повинен мати мінімум 3 сторінки: **вхід, реєстрація, інформаційна панель (основне вікно для входу).**

Верстку даних сторінок здійснити за макетом, який доданий нижче.

Загалом у PasswordManager користувач повинен мати можливість **додавати/редагувати/видаляти/розкривати паролі на дашборді після входу до системи.**

У самому дашборді має бути **popup menu**, звертаю Вашу увагу на те, що при обранні певного **item** з **popup menu**, змінюється також й вигляд сторінки.

Всі дані про паролі повинні десь зберігатися, щоб при перезавантаженні сторінок користувач був переспрямований назад на дашборд та всі паролі були отримані.

Ви можете обрати будь-яку БД або локальне сховище (**localStorage**).

!Важливо: За замовчуванням паролі повинні бути приховані символом «*», коли користувач робить **double click** на полі з паролем, він бачить справжній пароль.

ToDo List

Мета: продемонструвати навички створення SPA-застосунків, закріплення на практиці фреймворків і бібліотек, вивчених на курсі.
Обов'язкові мови/технології/бібліотеки: React.

Додаткові матеріали:

Посилання на макет:

<https://www.figma.com/file/uSI4FwGvir3kk1Pc3hV0ri/ToDoList?t=ZVamsWnbd22575og-6>

ToDoList повинен містити контейнер, у якому є заголовок, input (у який потрібно вводити завдання), поряд з input, у нас має міститись кнопка, яка буде містити в собі знак add.

Як тільки користувач натисне на кнопку, завдання яке він ввів у поле input повинно додатись нижче, формуючи відповідний список завдань.

!Важливо: При додаванні нового завдання, старе завдання не повинно видалятись, допоки користувач сам не видалить його, натиснувши на відповідний знак.

Посилання на React Icons:

<https://react-icons.github.io/react-icons/search?q=add>

Книга контактів

Мета: продемонструвати навички створення SPA-застосунків, закріплення на практиці фреймворків і бібліотек, вивчених на курсі.

Обов'язкові мови/технології/бібліотеки: React.

Додаткові матеріали:

Посилання на макет:

<https://www.figma.com/file/f0OqEu7C4idlnXLfsftIyA/Contacts?node-id=0%3A1&t=nVP2sHG3JDuicPPl-0>

Ваша книга контактів повинна містити наступні поля:

FirstName:

LastName:

Email:

Phone:

А також кнопку «**Add to contact**» дана кнопка має додавати користувача до списку контактів.

*Для збереження контактних даних, можете використовувати будь-яку БД, або ж **localStorage***

Бонус завдання:

Передбачити можливість редагування контакту, а також видалення.