ДЕРЖАВНИЙ ТОРГОВЕЛЬНО-ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ

Кафедра комп'ютерних наук та інформаційних систем

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
на тему:

«Розробка UI бібліотеки з використанням фреймворку Vue.js»

Студента 4 курсу, 8 групи, Харлаєнко Вадим Андрійович

спеціальності
122 «Комп'ютерні науки» _____
підпис студента

Науковий керівник
доктор фізико-математичних наук, професор _____ Пурський Олег Іванович
підпис керівника

Гарант освітньої програми
кандидат технічних наук, доцент _____ Демідов Павло Георгійович
підпис керівника

Київ 2023

Державний торговельно-економічний університет

Факультет інформаційних технологій
Кафедра комп'ютерних наук та інформаційних систем
Спеціальність 122 «Комп'ютерні науки»

Затверджую
Зав. кафедри _____Пурський О.І.
«12» грудня 2022р.

Завдання
на випускну кваліфікаційну роботу студентці

Харлаєнко Вадим Андрійович
(прізвище, ім'я, по батькові)

1. Тема випускної кваліфікаційної роботи (проекту)
Розробка UI бібліотеки з використанням фреймворку Vue.js
Затверджена наказом ректора від *«09» грудня 2022 р. № 3332*
2. Строк здачі студентом закінченої роботи *30 травня2023 року*
3. Цільова установка та вихідні дані до роботи
Мета роботи: розробка UI бібліотеки з використанням фреймворку Vue.js
Об'єкт дослідження: процес розробки UI бібліотеки, та процес її взаємодії з фреймворком Vue.js .
Предмет дослідження: методи та засоби створення UI бібліотеки
4. Перелік графічного матеріалу_____
_____
_____

5. Консультанти по роботі із зазначенням розділів, за якими здійснюється консультування:

Розділ	Консультант (прізвище, ініціали)	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Пурський О. І.	15.12.2022 р.	15.12.2022 р.
2	Пурський О. І.	15.12.2022 р.	15.12.2022 р.
3	Пурський О. І.	15.12.2022 р.	15.12.2022 р.

6. Зміст випускної кваліфікаційної роботи (проекту) (перелік питань за кожним розділом)

Вступ

РОЗДІЛ 1. ОГЛЯД СУЧАСНИХ ТЕНДЕНЦІЙ РОЗВИТКУ ПРИЗНАЧЕНОГО ДЛЯ КОРИСТУВАЧА ІНТЕРФЕЙСУ

1.1. Роль UI бібліотек у сучасній веб-розробці

1.2. Аналіз існуючих UI бібліотек з використанням Vue.js

1.3. Огляд тенденцій у дизайні інтерфейсу

РОЗДІЛ 2 ОПИС ІНСТРУМЕНТІВ РОЗРОБКИ

2.1. Мова програмування JavaScript

2.2. Фреймворк Vue.js

РОЗДІЛ 3. ДИЗАЙН АРХІТЕКТУРИ UI БІБЛІОТЕКИ ТА ЇЇ РОЗРОБКА

3.1. Визначення потреб та цілей UI бібліотеки

3.2. Розробка структури компонентів

3.3 Створення дизайну в програмі Figma

3.4 Розробка основних компонентів

3.5 Відображення результату

ВИСНОВКИ

ВИКОРИСТАНА ЛІТЕРАТУРА

7. Календарний план виконання роботи

№ пор.	Назва етапів випускного кваліфікаційного проекту	Строк виконання етапів роботи	
		За планом	фактично
1	2	3	4
1	*Вибір теми випускного кваліфікаційного проекту*	04.10.2022	04.10.2022
2	*Розробка та затвердження завдання на випускний кваліфікаційний проект*	15.12.2022	15.12.2022
3	*Вступ*	03.02.2023	03.02.2023
4	*РОЗДІЛ 1. Огляд сучасних тенденцій розвитку призначеного для користувача інтерфейсу*	28.02.2023	28.02.2023
5	*РОЗДІЛ 2. Опис інструментів розробки*	06.05.2023	06.04.2023
6	*РОЗДІЛ 3. Дизайн архітектури UI бібліотеки та її розробка*	12.05.2023	12.05.2023
7	*Висновки*	15.05.2023	15.05.2023
8	*Здача випускного кваліфікаційного проекту на кафедру науковому керівнику*	30.05.2023	30.05.2023
9	*Попередній захист випускного кваліфікаційного проекту*	31.05.2023 -01.06.2023	31.05.2023 -01.06.2023
10	*Виправлення зауважень, зовнішнє рецензування випускного кваліфікаційного проекту*	02.06.2023	02.06.2023
11	*Представлення готового зшитого випускного кваліфікаційного проекту на кафедру*	05.06.2023	05.06.2023
12	*Представлення готової зшитої випускної кваліфікаційної роботи на кафедру*	05.06.2023	05.06.2023
13	*Публічний захист випускної кваліфікаційної роботи*	*За розкладом роботи ЕК*	

8. Дата видачі завдання «15 » грудня 2022 р.

Керівник випускної кваліфікаційної роботи (проекту)
Пурський О. І.
(прізвище, ініціали, підпис)

Гарант освітньої програми Демідов П.Г.
(прізвище, ініціали, підпис)

Завдання прийняв студент-дипломник Харлаєнко В. А.
(прізвище, ініціали, підпис)

9. Відгук керівника випускної кваліфікаційної роботи (проекту)

_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____

Керівник випускної кваліфікаційної роботи (проекту)

_____ 30.05.2023 р.

(підпис, дата)

10. Висновок про випускну кваліфікаційну роботу (проект)

Випускна кваліфікаційна робота Харлаєнко В. А.

(прізвище, ініціали)

може бути допущена до захисту в екзаменаційній комісії.

Гарант освітньої програми_____ Демідов П.Г.

(підпис, прізвище, ініціали)

Завідувач кафедри _____ Пурський О.І.

(підпис, прізвище, ініціали)

«_____»_____2023р.

Анотація

Ця Кваліфікаційна робота присвячена розробці користувальницького інтерфейсу бібліотеки (User Interface) за допомогою популярного фреймворку Vue.js. У поєднанні зі швидким розвитком веб-розробки створення власних компонентів і бібліотек для побудови інтерфейсів стає все більш важливим завданням. У цій роботі будуть розглянуті фундаментальні принципи розробки бібліотеки інтерфейсу користувача, а також розглянуті ключові особливості фреймворку Vue.js.

Далі у роботі надається огляд фреймворку Vue.js, описуються його основні функції та архітектура компонентів. Цей огляд має на меті створити основу для розробки інтерфейсу користувача бібліотеки.

У третій частині роботи розглядається дизайн архітектури UI бібліотеки. Визначення потреб та цілі бібліотеки, розробка структури компонентів та їх взаємодію. Також розглядається розробка компонентів і шаблонів за допомогою Vue.js.

У наступному розділі описано процес впровадження інтерфейсу користувача бібліотеки за допомогою Vue.js. Розробка основних компонентів, стилізація та дизайн компонентів.

КЛЮЧОВІ СЛОВА: *UI БІЛБЛІОТЕКА, VUE.JS, UI*

Anotation

This qualification work is devoted to the development of the user interface of the library (User Interface) using the popular Vue.js framework. Coupled with the rapid development of web development, creating your own components and libraries for building interfaces is becoming an increasingly important task. This paper will cover the fundamental principles of user interface library development, as well as the key features of the Vue.js framework.

Next, the paper provides an overview of the Vue.js framework, describes its main functions and component architecture. This overview is intended to provide a framework for library user interface development.

In the third part of the work, the design of the UI architecture of the library is considered. Determining the needs and goals of the library, developing the structure of components and their interaction. Development of components and templates using Vue.js is also covered.

The next section describes the process of implementing the library's user interface using Vue.js. Core component development, styling and component design.

Keywords*: Ui Library, Vue.Js, Ui*

ЗМІСТ

ВСТУП

У сучасному світі веб-розробка стала важливою частиною ведення успішного бізнесу в онлайн-середовищі. Від інтерфейсу користувача залежить те, наскільки легко користувачі взаємодіють із веб-програмою, їхнє задоволення та дотримання очікувань. Візуальна привабливість, ефективність і швидкість розробки інтерфейсу стають важливими факторами отримання конкурентної переваги на ринку. Щоб полегшити процес розробки інтерфейсів користувача та забезпечити високу якість їх реалізації, використання бібліотек інтерфейсу користувача набирає популярності серед веб-розробників. Ці бібліотеки надають готові компоненти та елементи інтерфейсу, які можна швидко та легко інтегрувати у веб-додатки. Однак існуючі бібліотеки інтерфейсу користувача не завжди повністю відповідають потребам розробників або забезпечують найкращу інтеграцію з популярними фреймворками.

Ця дипломна робота присвячена розробці інтерфейсу користувача бібліотеки за допомогою фреймворку Vue.js, одного з найпопулярніших фреймворків розробки веб-додатків. Основною метою цієї роботи є створення набору готових компонентів та елементів інтерфейсу, які будуть ефективно працювати з фреймворком Vue.js та забезпечуватимуть легку та швидку розробку для розробки інтерфейсу користувача веб-додатків.

Розділ реалізації охоплює весь процес розробки інтерфейсу бібліотеки за допомогою фреймворку Vue.js, включаючи розробку основних компонентів та дизайн стилю.

Робота завершується прикладом використання розробленої бібліотеки інтерфейсу користувача, включаючи розробку простої форми та демонстрацію можливостей і функціональності бібліотеки.

Мета роботи: розробка UI бібліотеки з використанням фреймворку Vue.js

Об'єкт дослідження: UI бібліотека, яка буде розроблена, та її взаємодія з фреймворком Vue.js

Предмет дослідження: методи та засоби створення UI бібліотеки

Методи дослідження: Огляд поточних тенденцій у розробці інтерфейсу користувача, ролі бібліотек інтерфейсу користувача у веб-розробці та аналіз існуючих бібліотек інтерфейсу користувача за допомогою Vue.js. Такий підхід дозволить зібрати та систематизувати наявну інформацію про розвиток UI бібліотеки та тенденції її використання.

Основні завдання дослідження:

1. Визначення потреб та цілі розробки UI бібліотеки, розробити архітектуру компонентів та їх взаємодії.
2. Розробка інтерфейсу користувача бібліотеки за допомогою фреймворку Vue.js, включаючи реалізацію основних компонентів, стиль і дизайн, а також тестування та налагодження
3. Прикладне дослідження, розробивши просту форму та продемонструвавши можливості та функціональність розробленої бібліотеки інтерфейсу користувача.

Структура та обсяг випускної кваліфікаційної роботи. Випускна кваліфікаційна робота складається із вступу, трьох розділів, висновків, списку використаних джерел із 10 найменувань, додатків і містить 28 сторінки основного тексту, 19 рисунків і 6 таблиці.

РОЗДІЛ 1. ОГЛЯД СУЧАСНИХ ТЕНДЕНЦІЙ РОЗВИТКУ ПРИЗНАЧЕНОГО ДЛЯ КОРИСТУВАЧА ІНТЕРФЕЙСУ

1.1 Роль UI бібліотек у сучасній веб-розробці

У сучасній веб-розробці використання UI-бібліотеки може прискорити та спростити процес розробки інтерфейсу.

Однією з переваг використання бібліотеки інтерфейсу є можливість значно скоротити час розробки. Замість того, щоб створювати компоненти з нуля, розробники можуть використовувати готові рішення, які пройшли перевірку та оптимізацію. Це дозволяє пришвидшити процес розробки та зосередитися на бізнес-логіці вашої програми.

Крім того, бібліотеки інтерфейсу користувача допомагають забезпечити узгодженість інтерфейсу. Вони забезпечують узгоджений стиль і дизайн для всіх компонентів, що дає змогу створити єдину та цілісну взаємодію з користувачем у вашій веб-програмі.

1.2 Розгляд існуючих UI бібліотек для Vue.js

Перед тим як почати розробку власної UI бібліотеки, потрібно розглянути вже існуючі та ознайомитись з їх перевагами та недоліками, оглянути їх популярність, щоб зрозуміти основні критерії на які слід звернути увагу. Було розглянуто декілька популярних бібліотек (табл 1.1).

Таблиця 1.1 Особливості існуючих бібліотек [2]

Назва бібліотеки	**Особливості**	**GitHub starts**
1. **Element UI**	Бібліотека є душе поширеної серед розробників. Ця бібліотека має різні версії для Angular і React і особливо надійна для десктопних програм.	Більше 45 тис.
1. Vuetify	Бібліотека, яка має опцію рендерінгу на стороні сервера, що дає змогу використовувати її в своєму проекті, без зайвих налаштувань та помилок. Також допомагає створювати семантичні компоненти інтерфейсу користувача, які можна повторно використовувати багато разів	Більше 23 тис.
2. Bootstrap Vue	Великим плюсом цієї бібліотеки є те, що більшість спільноти розробників знають про Bootstrap, тому поріг входу стає мінімальним. Окрім цього є готові не тільки компоненти, а ще й готові форми та багато інших готових рішень	Більше 10 тис.

Обираючи UI бібліотеку для розробки, слід враховувати кілька особливостей, які можуть вплинути на ваш вибір. Ось кілька ключових аспектів, які слід враховувати:

- Функціональність та компоненти: спочатку оцініть компоненти та функціональність, які надає бібліотека. Переконайтеся, що є всі необхідні вам компоненти, такі як кнопки, форми, таблиці, моди тощо. Подивіться, чи можна легко налаштувати ці компоненти відповідно до ваших потреб.
- Дизайн і стиль: враховуйте стиль дизайну вашої бібліотеки. Виберіть той, який найкраще підходить до вашого вигляду та естетики. Переконайтеся, що ви можете легко налаштувати стиль ваших компонентів відповідно до вашого проекту.
- Розмір і продуктивність: оцініть розмір бібліотеки та її вплив на продуктивність програми. Якщо у вас є обмеження щодо розміру або вам потрібна найвища продуктивність, виберіть бібліотеку з компактним розміром і оптимізованим кодом.
- Документація та спільнота: переконайтеся, що бібліотека має добре написану документацію та активну спільноту користувачів. Це допоможе вам швидко знайти відповіді на свої запитання, отримати підтримку та скористатися прикладами коду
- Сумісність і розширюваність: переконайтеся, що бібліотека сумісна з версією Vue.js, яку ви використовуєте.

1.3 Огляд тенденцій у дизайні інтерфейсу

Дизайн є важливою частиною розробки інтерфейсу, розуміння принципів і методів дизайну може значно покращити якість продуктів, які ви створюєте. Дизайн інтерфейсу веб-додатків постійно розвивається, і тенденції в уподобаннях і звичках користувачів мають вирішальне

значення для створення привабливих і зручних продуктів [3]. Ось кілька ключових тенденцій, на які варто звернути увагу:

1. Мінімалізм в дизайні: нині мінімалізм в дизайні є дуже поширений. Він характеризується своєю простотою, обмеженим використанням кольорів та чіткими лініями. Обравши мінімалістичний підхід при для дизайну інтерфейс для користувача стає легшим до сприйняття та концентрує увагу на важливих елементах.

2. Адаптивність дизайну: важливо щоб дизайн був адаптивним, так як користувач може використовувати девайс різних розмірі від телефону до телевізорів. Таким чином, адаптивний дизайн, який адаптується до різних розмірів екрана та пристроїв, стає необхідністю. Це забезпечує комфорт при використанні та приємний досвід користування.

3. User Experience (користувацький досвід): треба розуміти сформований досвід користувача який він накопичив за всі свої роки користування додатками або веб-сайтами. Наприклад: при кліку на синій текст, майже у всіх випадках перенесе вас на сайт. Тому дуже важливо враховувати цей момент при будуванні дизайну.

РОЗДІЛ 2. ОПИС ІНСТРУМЕНТІВ РОЗРОБКИ

2.1 Мова програмування JavaScript

Мова програмування JavaScript — це динамічна, об’єктно-орієнтована, прототипна мова програмування. Найчастіше використовується для створення сценаріїв веб-сторінок, які надають можливість взаємодіяти з користувачем на стороні клієнта (пристрій кінцевого користувача), керувати браузером, асинхронно обмінюватися даними з сервером і змінювати структуру та зовнішній вигляд веб-сторінки. сторінки. Хоча способів використання в яких виконується JS, постійно зростає (від браузерів до серверів(Node.js), роботів та ін.), єдина його середа, в якій йому немає конкурентів це веб-розробка.[4]

Мову JavaScript класифікують як прототипну (підмножина об'єктноорієнтованої), скриптову мову програмування з динамічною типізацією. Окрім прототипної, JavaScript також частково підтримує інші парадигми програмування (імперативну та частково функціональну) і деякі відповідні архітектурні властивості, зокрема: динамічна та слабка типізація, автоматичне керування пам'яттю, прототипне наслідування, функції як об'єкти першого класу.

Мова JavaScript використовується для:

— Для надання інтерактивності для веб-сторінок;

— створення SPA (React, AngularJS, Vue.js);

— для програмування серверів (Node.js);

— для стаціонарних девайсів(Electron);

— для мобільних застосунків (React Native);

— в PDF документах тощо.

Незважаючи на подібні назви, Java і JavaScript — це дві різні мови з дуже різною семантикою, хоча вони мають схожі характеристики з точки

зору стандартних бібліотек і угод про іменування. Синтаксис обох мов «успадкований» від C, але семантика і дизайн JavaScript є результатом впливу обох мов Self та Scheme [5].

Використання інструментів налагодження має важливе значення при розробці великих і важливих веб-додатків за допомогою JavaScript. Оскільки браузери від різних виробників дещо відрізняються своєю поведінкою JavaScript і реалізацією об’єктної моделі документа, необхідно мати налагоджувач для кожного браузера, якщо веб-програма націлена на нього.

Оскільки JavaScript є інтерпретатором без суворої типізації, він може працювати в різних середовищах, кожне зі своїми особливостями Сумісність, програміст повинен бути дуже обережним і повинен перевірити свій код належним чином у широкому діапазоні можливих конфігурацій.

Програмний код аналізується інтерпретатором окремо. в Інтернеті Вам потрібно поєднати блоки JavaScript і HTML, синтаксичні помилки легше помітити, якщо ви зберігаєте функції в окремих блоках коду, або використовуйте багато невеликих пов’язаних файлів .js. щоб він був синтаксичним щоб не було збою усієї сторінки.

2.2 Фреймворк Vue.js

Фреймворк Vue.js - JavaScript-фреймворк що використовує паттерн MVVM для створення інтерфейсів користувача на основі моделей даних, через реактивне зв'язування даних. Фреймворк Vue має свій синтаксис шаблонів на основі HTML, що дозволяє декларативно зв'язувати рендеринг DOM з основними екземплярами даних в Vue [6]. Одна із сильніших особливостей Vue — це реактивна система. Моделі це прості JavaScript об’єкти. Завдяки цьому керування станами стає дуже простим та

інтуїтивно зрозумілим, особливо для тих хто тільки починає розбиратися в Vue (рис. 1.1).

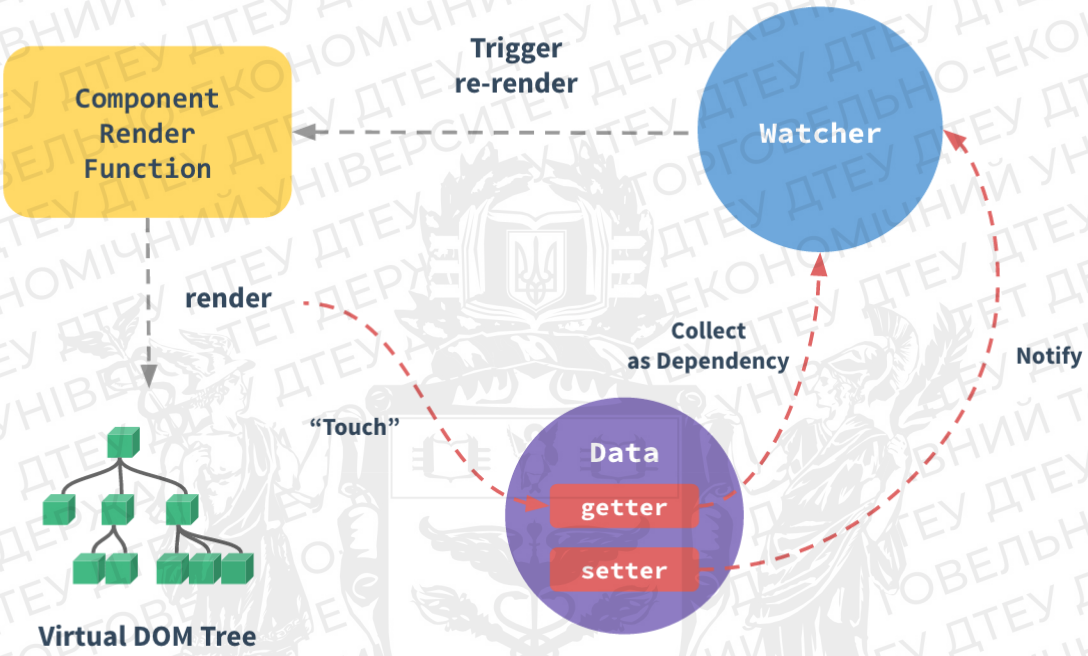


Рисунок 1.1. Схема роботи реактивності Vue.js

РОЗДІЛ 3. ДИЗАЙН АРХІТЕКТУРИ UI БІБЛІОТЕКИ ТА ЇЇ РОЗРОБКА

3.1 Визначення потреб та цілей UI бібліотеки

Головною метою розробки UI бібліотеки є створення декількох універсальних UI компонентів, які буде зручно використовувати розробнику у своїх проектах, дизайн проектів може будуватися на основі компонентів UI бібліотеки. Компонент – це функціонально готова частинка коду, яка має свій HTML шаблон, набір функцій та пропси. Наприклад: кнопка, у якої може бути декілька станів, на цей стан може впливати пропс, назвемо його “color”, кнопка приймаючи цей проп буде змінювати свій колір, а при натисканні на неї відбувається який функціонал який встановив розробник при використанні готового компоненту. Відповідно ціль розробника UI бібліотеки передбачити всі можливі сценарії використання, та пропрацювати їх. Для виконання цієї цілі треба врахувати всі потреби розробника як користувача бібліотеки. Для зручного використання розробники завжди покладаються на документацію та можливість модернізувати стилі або поведінку компонентів, в залежності від поставлених для нього задач. Тому документація є чи не самою головною складовою бібліотеки, на яку треба приділити максимум зусиль та уваги. Також при виборі бібліотеки часто звертають увагу на її вагу, чим менше тим краще. Були виділені основні напрямки для виконання заданих цілей:

- Зробити бібліотеку з мінімальним розміром
- Документація яка описує кожний компонент
- Можливість змінювати стилі
- Гарний дизайн

3.2 Розробка структури компонентів

Розробка структури компонентної є важливим етапом розробки. Цей етап включає в себе визначення кількість створених компонентів. Основна мета цього етапу включає в себе створення логічної та зручної структури для використання компонентів.

В даній бібліотеці буде створено 5 самих частих елементів, які використовують при розробці веб-сайту. Кожен з цих компонентів має стилізацію в одному стилі (кольори, шрифт, анімації та ін.). Перші чотири компоненти будуть використовуватися замість стандартних HTML тегів. а 5 є перемикачем (можна використати як перемикають зі світлої на темну тему). Список компонентів з їх майбутніми назвами:

- KH-Input
- KH-Checkbox
- KH-Button
- KH-Select
- KH-Switcher

3.3 Створення дизайну в програмі Figma

Figma – це зручний веб-додаток для створення дизайну інтерфейсів. По-перше, треба обрати кольори які буду використовуватися для компонентів. Коли настає час обирати колір, важливо обрати кольори які підходять один до одного, такі правила гармонічного підбору кольорів. Колірна гармонія використовує математику для розрахунку відмінностей та підбору кращих кольорів які підходять до основного. Є кілька різних правил, аналогічних, одноколірних, тріадних і додаткових, щоб назвати декілька [7]. При виборі кольорів дуже важливо думати про доступність. Дальтонізм дуже поширений і вражає 1 з 12 чоловіків [8].

Основним кольором був обраний #1073EB. Тепер потрібно обрати сумісні для нього кольори для тексту та різних станів анімації (наведення, та натискання) (рис. 2.1).

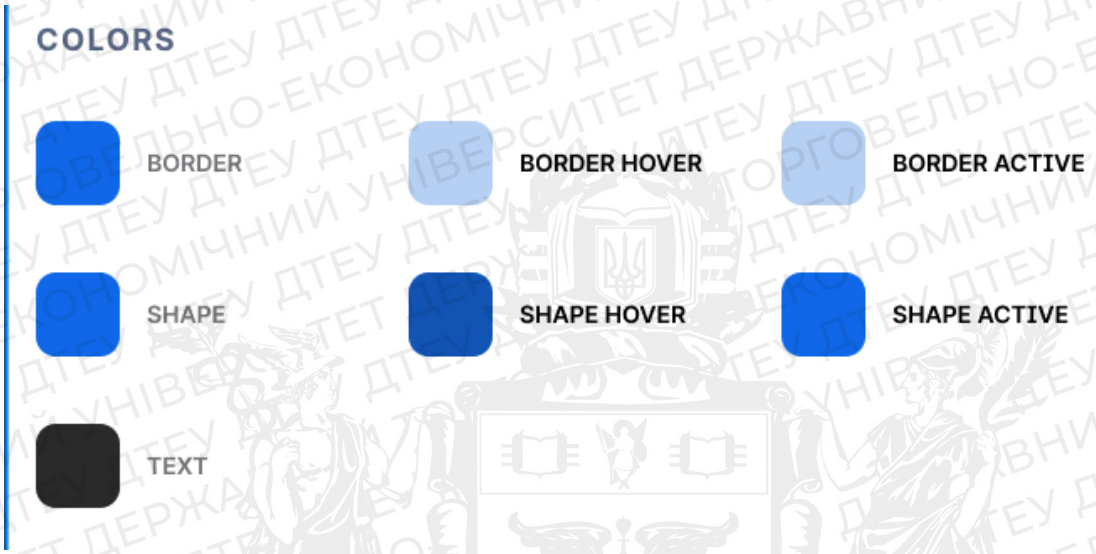


Рисунок 2.1. Кольорова палітра для бібліотеки

3.3.1 Дизайн Компонентів

Розробляючи дизайн компонентів треба врахувати можливі стани для анімацій. При наведенні колір буде трохи темнішим за стандартний, а при натисканні трохи світлішим. Також для краси треба додати тіні та бордери. Для зручності розробки бібліотеки, на дизайні потрібно покрити всі можливі стани, щоб під час розробки звертати увагу тільки на функціональні аспекти. Тому на дизайні будуть всі компоненти та їх зовнішній вигляд під впливом дій користувача. Обов'язково враховуємо UX, тобто, розробити зовнішній вигляд таким як зазвичай виглядають елементи інтерфейсів в інших програмах, те до чого звикли користувачі ПО.

Далі будуть розглянутий дизайн компонентів. Кнопка, вона має округлий вигляд, відступи всередині себе для кращого вигляду тексту та

невелику тінь для покращення вигляду. У кнопки буде декілька виглядів, маленька та велика за розмірами, також одна з кольорових фоном, а інша з прозорим, зроблено це для різноманіття її вигляду. (рис. 3.1).

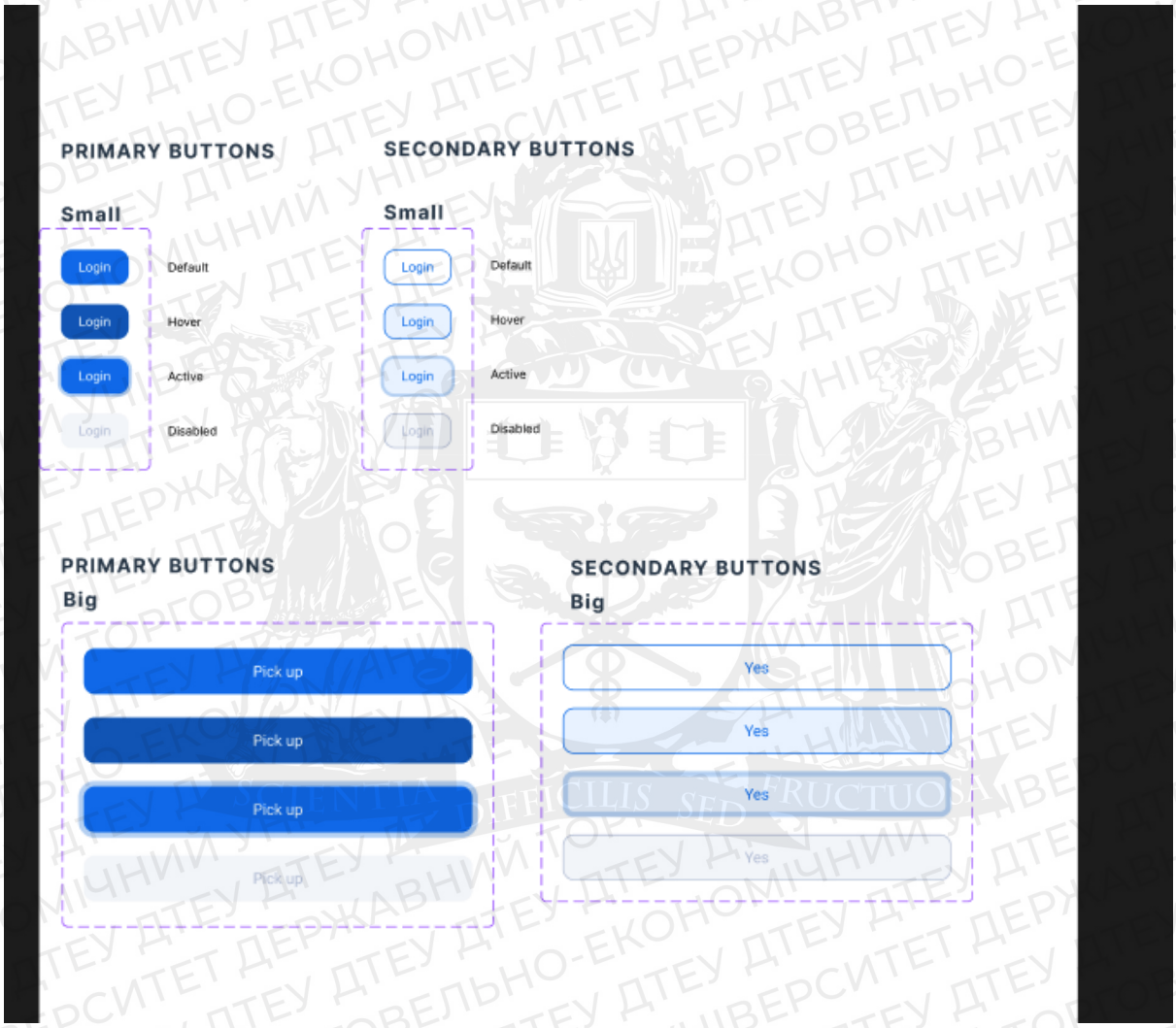


Рисунок 3.1. Дизайн компоненту KH-Button

Інпут, треба врахувати що кастомізувати стандартний HTML input є непростою задачею, тому потрібно зробити мінімальну зміну стилів, щоб покращити його зовнішній вигляд, не додаючи багато зайвих елементів, так як це може ускладнювати його розробку, в різних браузерах може бути різний результат. У інпута існує багато атрибутів, але звернемо увагу на placeholder – підказка для користувача, яка допомагає зрозуміти що від

нього очікують[9]. Це означає що використовують його майже кожний раз, тому дамо йому більш виразний колір (рис. 4.1).

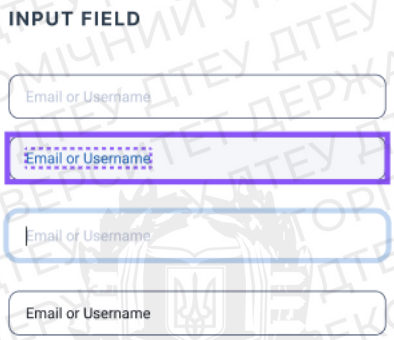


Рисунок 4.1. Дизайн компоненту KH-Input

Перемикач, може мати два стани (вимкнений та увімкнений) для них зроблений окремий зовнішній вигляд (рис. 5.1).

Рисунок 5.1. Дизайн компоненту KH-Switcher

Чекбокс, стандарний HTML тег, input з аттрибутом «type» і значенням «checkbox». У нього буде три стани для зовнішнього вигляну: наведення, коли його ввімкнули та вимкнули (Рис. 6.1).

Рисунок 6.1. Дизайн компоненту KH-Checkbox

Останній компонент це «select», взагалі стандартний «select», на даний момент, не має можливості дати йому стилів з допомогою css, тому розробники частіше за все роблять свій селект. Для селекту потрібно вивести обране значення та сам список елементів які користувач може обрати (рис. 7.1).

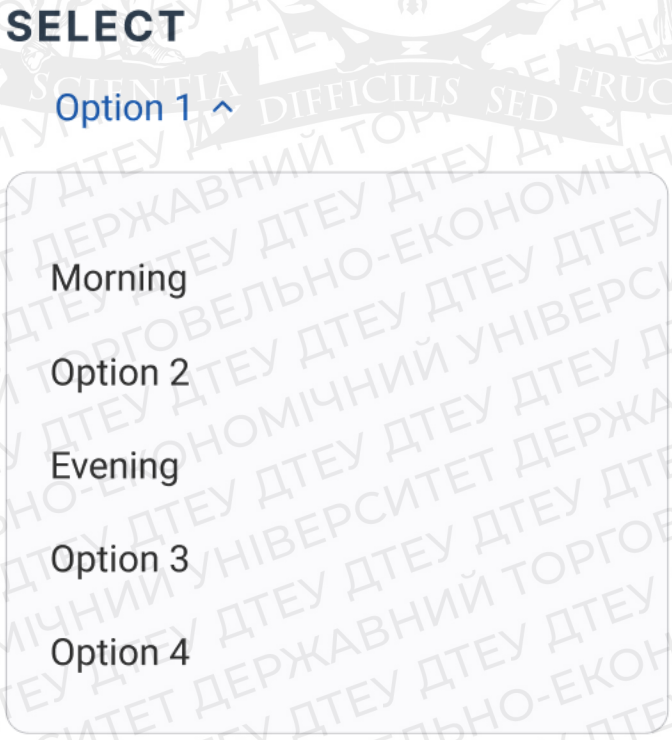


Рисунок 7.1. Дизайн компоненту SW-Select

3.4 Розробка компонентів

Для створення проекту з використанням потрібно зробити декілька етапів. Створимо папку з назвою проекту “KH-UI-KIT”, після створення папки потрібно відкрити її в терміналі, встановити vue.js.

Для спрощення подальшої розробки, заздалегідь, треба створити «variables.css» файл (рис. 8.1), в якому створені змінні для запису в них кольорів. Така підготовка прискорює розробку, так як не потрібно відволікатися на кольори в майбутньому і надає змогу перезаписати ці змінні, якщо при використанні бібліотеки буде потреба змінити кольори.

Рисунок 8.1. Файл «variables.css»

3.4.1 Створення компоненту KH-Button

Створимо файл «KHButton.vue», для кнопки нам потрібно описати «props» (табл. 2.1) для виконання задуманої логіки компоненту. Згідно дизайну робимо верстку, головне не забувати про семантику тегів, вони покращать SEO та покращують досвід користування для людей з певними вадами здоров'я [10].

Таблиця 2.1 Опис props для компоненту KH-Button

Prop	Type	Default
Variant	String	Primary
Size	String	Small

Написання HTML у VUE.js відбувається в тезі «template» (рис. 9.1). Опишемо кнопку, для цього створимо тег «button» на який відразу потрібно проставити декілька атрибутів для зручного використання компоненту. Перший атрибут це «v-bind="$attrs"», завдяки цьому атрибуту, розробник користуючись «KH-Button» може проставити стандартні атрибути тегу «button». Наступний атрибут це «v-on="{ ...$listeners }"» він працює майже як і минулий атрибут, але це вже не стандартний, а атрибут який дає змогу прокинути функції які викликаються коли відбувається якась подія, наприклад: на KH-button можливо поставити функцію яка буде реагувати на клік «@click=”func”». Такі дії будуть виконані для кожного компоненту.

KHButton.vue

src > lib-components > KHButton.vue > { } "KHB

```
1   <template>
2     <button
3       v-bind="$attrs"
4       v-on="{ ...$listeners }"
5       class="button"
6       :class="className"
7     >
8       <slot></slot>
9     </button>
10  </template>
11
```

Рисунок 9.1. Шаблон компоненту «KH-Button»

Далі описання функціоналу, треба реагувати на props, коли є prop «Size» буде підставлятися клас «big» або «small», а на prop «Variant» будуть клас «primary» або «secondary», цей фунціонал буде опрацьовувати computed «className» (рис. 10.1), який в залежності від наданих props буде видавати клас, який буде ставитись на кнопку. Стилів для кожного випадку взято з дизайну, для кожного випадку є свій клас в якому описані стилі.

```
<script>
const VARIANT_SECONDARY = "secondary";
const VARIANT_SIZE = "big";

export default {
    name: "KHButton",

    props: {
        variant: {
            type: String,
            default: "primary",
        },
        size: {
            type: String,
            default: "small",
        },
    },

    computed: {
        className() {
            return {
                secondary: this.variant === VARIANT_SECONDARY,
                big: this.size === VARIANT_SIZE,
                primary: this.variant !== VARIANT_SECONDARY,
                small: this.size !== VARIANT_SIZE,
            };
        },
    },
};
</script>
```

Рисунок 10.1. Функціональна частина компоненту KH-Button

3.4.2. Створення компоненту KH-Checkbox

Першочергово зробити дії як і для першого компоненту, але вже на тег input з атрибутом «type=checkbox». Стилізувати стандартний checkbox під дизайн неможливо, тому доведеться за допомогою стилів сховати його, та зробити його кастомним. Для іконки галочки буде використано тег SVG, його особливість в тому що при зміні розмірів екрану, його якість не погіршиться, також за допомогою css його можна фарбувати. В випадку

використання картинок, довелося робити дві копії, що збільшує розміри бібліотеки.

Також для зручності використання директиви «v-model» буде зроблено додатковий функціонал, який включає в себе реагування на подію зміни (changeEvent), після якої компонент буде викликати «emit» зі зміною значення. Таким чином налаштування для директиви «v-model» є завершиним.

В таблиці описано props (табл. 2.2).

Таблиця 2.2 Опис props для компоненту KH-Checkbox

Prop	**Type**	**Default**
Variant	String	Primary
Size	String	Small

На рисунках нижче представлено реалізацію кастомного checkbox (рис. 11.1) та функціональність компоненту (рис. 12.1).

```
<template>
   <div class="container-checkbox-message">
      <div class="input-checkbox">
         <label class="check option">
            <input
               class="check__input"
               v-bind="$attrs"
               :value="value"
               type="checkbox"
               name="checkbox"
               @change="changeValue"
            />

            <span class="check__box" :class="className">
               <svg
                  xmlns="http://www.w3.org/2000/svg"
                  width="8"
                  height="6"
                  viewBox="0 0 8 6"
                  fill="none"
               >
                  <path
                     fill-rule="evenodd"
                     clip-rule="evenodd"
                     d="M7.7 0.3C7.3 -0.1 6.7 -0.1 6.3 0.3L3 3.6L1.7 2.3C1.3 1.9 0.7 1.9 0.3 2.3C-0.1 2.7 -0.1 3.3 0.3 3.7L2.3
                     fill="white"
                  />
               </svg>
            </span>
            <slot> </slot>
         </label>
      </div>
   </div>
</template>
```

Рисунок 11.1. Шаблон компоненту KH-Checkbox

```
<script>
const VARIANTRDIUSSTYLE = "secondary";
const VARIANT__SIZE = "big";
export default {
  name: "KHCheckbox",
  props: {
    variant: {
      type: String,
      default: "primary",
      note: "Only primary or secondary",
    },
    size: {
      type: String,
      note: "Only big or small",
      default: "small",
    },
  },
  data() {
    return {
      value: "",
    };
  },
  methods: {
    changeValue(elem) {
      this.$emit("input", elem.target.checked);
    },
  },
  computed: {
    className() {
      return {
        secondary: this.variant === VARIANTRDIUSSTYLE,
        big: this.size === VARIANT__SIZE,
      };
    },
  },
};
</script>
```

Рисунок 12.1. Реалізація функціоналу компоненту KH-Checkbox

3.4.3. Створення компоненту KH-Input

По аналогії до компоненту KH-Checkbox, потрібно налаштувати компонент для роботи з директивою «v-model». Так як компонент реалізує тег input, потрібно продумати шляхи його використання. Частіше за для input прописують атрибут «type», тому логіка буде відштовхуватися від значення переданого в цей атрибут, його знання буде в props (табл. 2.3).

Компонент буде обробляти «type» тільки для двох значень: text,

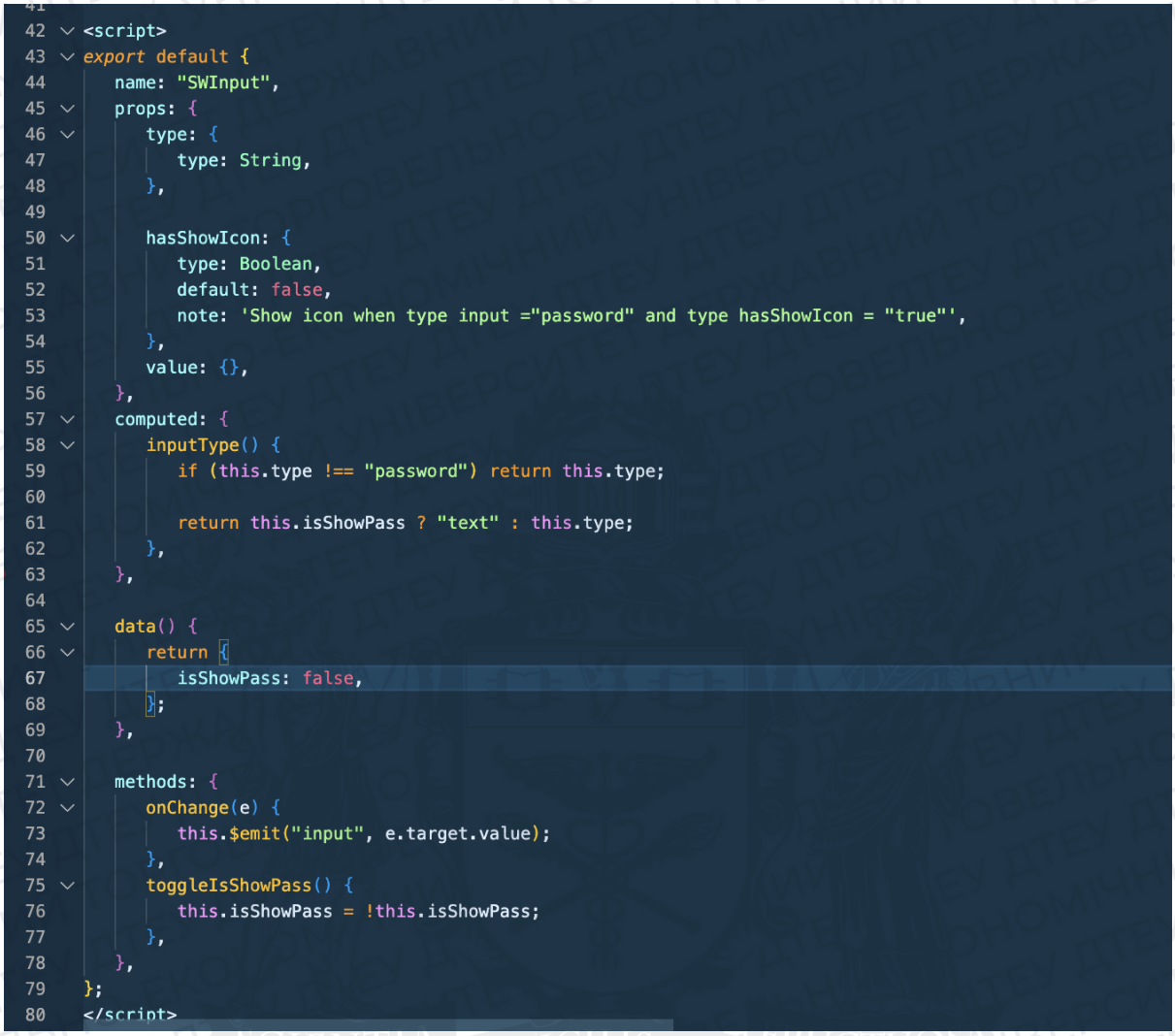
password. Для password буде додатково додана ікона з оком, використовуватися це може в випадку коли на якомусь веб-сайті користувач захоче побачити те що він ввів в поле з типом password. В інших випадка буде проставлятися тип text. Реалізацію іконки для показу паролю наведена в шаблоні (рис. 13.1) та в місці реалізації логіки компоненту (рис. 14.1).

Таблиця 2.3 Опис props для компоненту KH-Input

Prop	**Type**	**Default**
Type	String	Empty
hasShowIcon	Boolean	False

```vue
<template>
  <div class="block">
    <input
      v-bind="$attrs"
      v-on="{ ...$listeners, input: onChange }"
      :type="inputType"
      :value="value"
      class="input"
      :class="className"
    />
    <!--       <input
      v-bind="$attrs"
      :type="inputType"
      :value="value"
      class="input"
      :class="className"
      @input="onChange"
      ref="input"
    /> -->
    <button
      @click.stop="toggleIsShowPass"
      class="buttonImg"
      type="button"
      v-if="hasShowIcon && type === 'password'"
    >
      <svg
        xmlns="http://www.w3.org/2000/svg"
        width="16"
        height="10"
        viewBox="0 0 16 10"
        fill="none"
      >
        <path
          d="M7.99935 0C4.66602 0 1.81935 2.07333 0.666016 5C1.81935 7.92667 4.66602 10 7.99935 1
          fill="#445275"
        />
      </svg>
    </button>
  </div>
</template>
```

Рисунок 13.1. Шаблон компоненту KH-Input



```
<script>
export default {
    name: "SWInput",
    props: {
        type: {
            type: String,
        },

        hasShowIcon: {
            type: Boolean,
            default: false,
            note: 'Show icon when type input ="password" and type hasShowIcon = "true"',
        },
        value: {},
    },
    computed: {
        inputType() {
            if (this.type !== "password") return this.type;

            return this.isShowPass ? "text" : this.type;
        },
    },

    data() {
        return {
            isShowPass: false,
        };
    },

    methods: {
        onChange(e) {
            this.$emit("input", e.target.value);
        },
        toggleIsShowPass() {
            this.isShowPass = !this.isShowPass;
        },
    },
};
</script>
```

Рисунок 14.1. Реалізація функціоналу компоненту KH-Input

3.4.4 Створення компоненту KH-Select

Задача компоненту вивести опції які передав в props (табл. 2.4) розробник. Треба врахувати те що опцій може бути велика кількість, для зручності користувача треба зробити так, щоб список не виходив за межі екрану. Для цього встановити максимальну висоту блоку з опціями, але ще треба врахувати те, що при відкриванні списку опції, вони все одно можуть вийти за межі екрану. При показу списку опції прораховується висота екрану користувача, відстань від самого select до кінця екрану та прорахувати яку висоту можна задати для блоку списків. Сам список

виводиться за допомогою директиви «v-for». Для кожного елементу списків задано стан при наведенні, для того щоб користувачу було зручніше орієнтуватися на якому він зараз елементі. При виборі опції, вона прибирається зі списку всіх опцій, та стає обраною.

Prop «selected» (табл. 2.4), створений для випадку коли потрібно щоб зі списку була вже обрана опція. Якщо значення цього prop присутнє воно буде обраним та не буде відображене в списку опцій.

Таблиця 2.4 Опис props для компоненту KH-Select

Prop	**Type**	**Default**
options	array	[]
selected	String	empty

Нижче розглянуто реалізацію функціоналу прорахування висоти блоку з опціями (рис. 15.1). Коли відкривається список, викликається функція «updateList», за допомогою директиви “ref” ми в коді можемо отримати DOM елемент списку. Далі іде прорахунок та виставлення висоти.

```
updateList() {
    if (!this.areOptionVisibel) return;
    const rect = this.$refs.options.getBoundingClientRect();

    const listHeight = rect.height;

    if (listHeight > this.$options.MAX_HEIGHT_OPTIONS) {
        this.$refs.options.style.height =
            this.$options.MAX_HEIGHT_OPTIONS + "px";
    }

    if (window.innerHeight < this.$options.MAX_HEIGHT_OPTIONS) {
        this.$refs.options.style.height =
            window.innerHeight > 400
                ? window.innerHeight - 100 + "px"
                : "150px";
    }

    if (window.outerWidth < rect.right) {
        this.$refs.options.style.width =
            rect.width - (rect.right - window.outerWidth + 20) + "px";
    }
},
```

Рисунок 15.1. Реалізація функціоналу зміни висоти для KH-Select

В шаблоні описано HTML для компоненту, згідно дизайну виводиться обраний елемент, під ним список (рис. 16.1).

```
<template>
  <div class="plashkaMobailAdvice">
    <div class="awakeningDivMobail">
      <div class="select">
        <div @click="toggleOptions" class="svgSelect">
          <p ref="selectedItem" class="awakening awakeningModal">
            {{ selected }}
          </p>
          <div
            v-if="!areOptionVisibel"
            :class="{ activeImg__arrow1: !areOptionVisibel }"
            class="img__arrow1"
          >
            <svg…
            </svg>
          </div>
          <div
            v-else
            class="img__arrow2"
            :class="{ activeImg__arrow1: areOptionVisibel }"
          >
            <svg…
            </svg>
          </div>
        </div>
        <div
          v-if="areOptionVisibel"
          tabindex="0"
          ref="options"
          class="options"
          :class="{ above: isOptionsAbove }"
        >
          <div class="scroll_categoruBody">
            <div
              @click="selectOption(option)"
              v-for="option in options"
              :class="{ nonClickable: !option.name }"
              class="options__arr"
              tabindex="0"
              :key="option.value"
            >
              <template v-if="option.name !== selected">
                <div class="containerCategory">
                  <p class="awakeningModal">{{ option.name }}</p>
                  <div class="btnSpan" v-if="option.actions">
                    <button
                      v-for="(action, index) in option.actions"
                      :key="index"
                      @click.stop="action.callback(option.value)"
                    >
                      <span
                        class="span_btn"
                        :is="action.icon"
                      ></span>
                    </button>
                  </div>
                </div>
              </template>
            </div>
```

Рисунок 16.1. Шаблон компоненту KH-Select

3.4.5 Створення компоненту KH-Switcher

Останній компонент, перемикач. Для реалізації логіки компоненту стандартний input прибрано стилями. Щоб коректно працювали всі події input потрібно в тег label помістити input та span який буде відображати зовнішній вигляд, завдяки тому що label зв'язаний з input, а тег span в середині input поведінка компоненту буде схожа як у стандартного (рис. 17.1). Очікувати компонент буде тільки один prop (табл. 2.5), «label», який відображає текст переданий в значені.

Таблиця 2.5 Опис props для компоненту KH-Switcher

Prop	**Type**	**Default**
Label	String	Empty

Рисунок 17.1. Шаблон компоненту KH-Swither

В тезі script прописано функцію (рис. 18.1), яка реагує на зміну значення компоненту. Функція відправляє оновлене значення.

```
methods: {
    toggleSwitch() {
        this.$emit("input", !this.value);
    },
},
```

Рисунок 18.1. Реалізація функціоналу KH-Swither

3.5 Відображення результату

Для відображення всіх компонентів, створено сторінку, на якій зображено список компонентів. Створення списку відбувається в тегу ul, кожний з елементів списку поміщено в тег li в якому вставлений компонент та його назва (рис. 19.1).

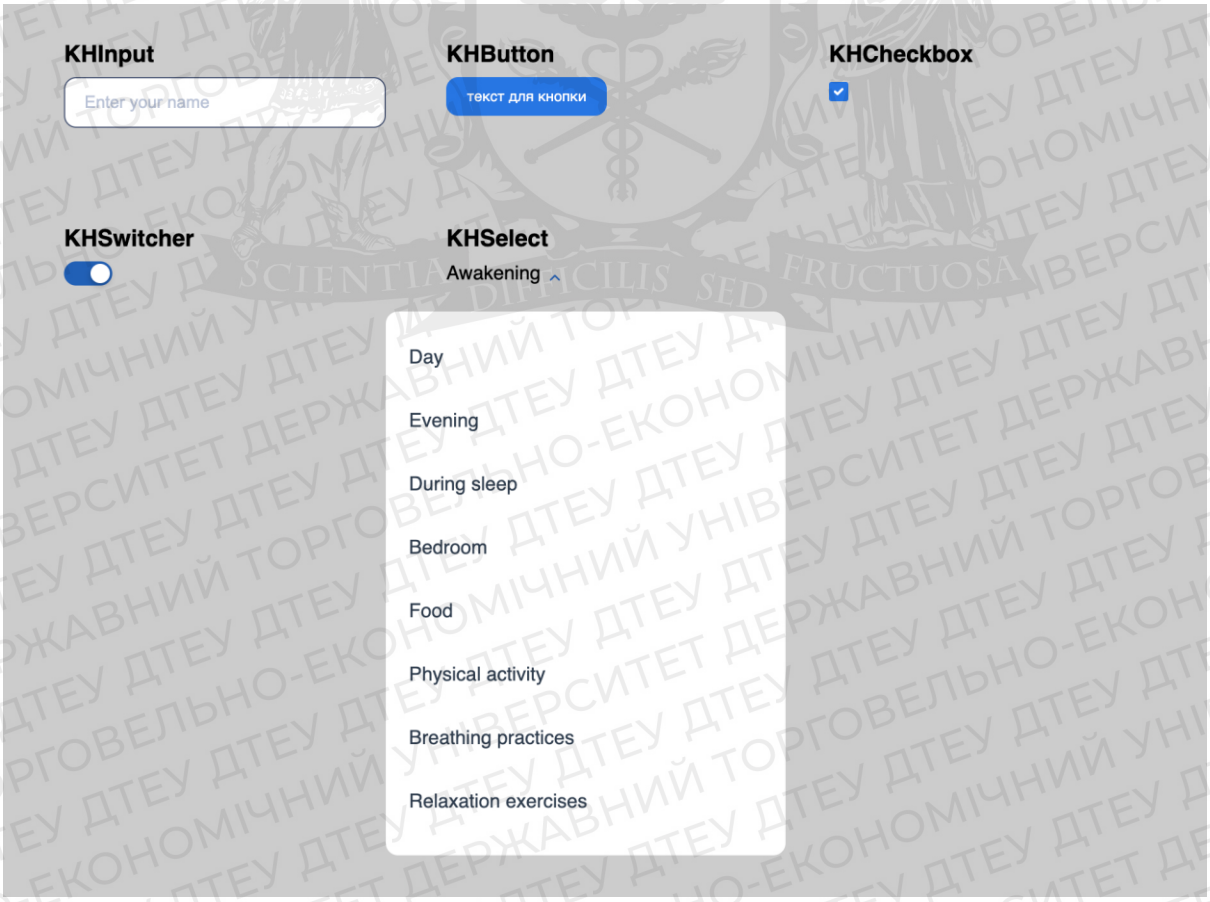


Рисунок 19.1. UI-бібліотека

Висновок

У результаті виконання дипломної роботи була розроблена UI бібліотека з використанням фреймворку Vue.js. Метою роботи було створити універсальні компоненти користувацького інтерфейсу, який спростить розробку веб-сайтів для розробників.

Під час дослідження було проведено проаналізовано сучасні UI бібліотеки та їх особливості.

Розробка дизайну за допомогою інструменту Figma. Дизайн елементів створений за сучасними тенденціями дизайну. Це дало змогу створити сучасний та знайомий для користувача інтерфейси.

В результаті реалізації UI бібліотеки були розроблені основні компоненти, проведена стилізація та оформлення, а також написання логіки. Бібліотека була успішно створена та продемонстровано їх на загальній сторінці.

Отже, розроблена UI бібліотека є важливим внеском у сучасну веб-розробку. Вона дозволяє розробникам швидко та просто створювати стильні та інтерактивні блоки на сайті, прискорюючи процес розробки. Робота відкриває перспективи для подальшого розвитку UI бібліотек та сприяє покращенню веб-дизайну та користувацького досвіду.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Introduction [Електронний ресурс]. – Режим доступу: https://v2.vuejs.org/v2/guide/
2. 10 Most Significant Vue.js UI Component Libraries in 2022 [Електронний ресурс]. – Режим доступу: https://www.atatus.com/blog/vuejs-ui-frameworks-libraries/
3. Jenifer Tidwell, Charles Brewer, Aynne Valencia, Designing Interfaces: Patterns for Effective Interaction Design [3rd Edition] 2020. – с. 67
4. Kyle Simpson, You Don't Know JS Yet: Get Started 2020. – с. 32
5. Кейт Джонс. DOM Scripting: Web Design with JavaScript and the Document Object Model. / К. Джонс 2005. — 368 с.
6. Anthony Gore — Full-Stack Vue.js 2 and Laravel 5 [Електронний ресурс]. — 2015. — Режим доступу: https://bit.ly/2OEODzR
7. Zena O'Connor. Colour harmony revisited. Color Research & Application - 35(4):267–273, 2010.
8. David Benyon. Designing Interactive Systems. Pearson Education, 3rd edition, 2013.
9. Jennifer Niederst Robbins, HTML5 Pocket Reference [5th Edition] 2013. - 74 с.
10. Matthew MacDonald, HTML5: The Missing Manual 2011. - с. 58