НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп'ютерних систем

До захисту допущено

Завідувач кафедри

_____ Віталій Романкевич
(підпис) (ініціали, прізвище)

"___" _____ 202___ р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Системне програмування»

спеціальності **123 «Комп'ютерна інженерія»**

на тему: Онлайн-сервіс для навчання касирів роботі з касовими апаратами

Виконав: Калитка Євгеній Олександрович

студент IV курсу, групи __КВ-62__
(шифр групи)

_____Калитка Євгеній Олександрович_____ _____
(прізвище, ім'я, по батькові) (підпис)

Керівник __Радченко Костянтин Олександрович_____ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант з нормоконтролю, доц.каф.СПСКС, к.т.н. Клятченко Я.М. _____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали) (підпис)

Рецензент_____
_____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2020 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп'ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Спеціальність 123 «Комп'ютерна інженерія»

Освітньо-професійна програма «Системне програмування»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Віталій Романкевич
(підпис) (ініціали, прізвище)
«___» _____20___ р.

ЗАВДАННЯ
на дипломний проєкт студента
Калитки Євгенія Олександровича
(прізвище, ім'я, по батькові)

1. Тема проєкту

Онлайн-сервіс для навчання касирів роботі з касовими апаратами_____,
керівник проєкту Радченко Костянтин Олександрович _____
затверджені наказом по університету від «__» _____ 20__ р. №_____

2. Термін подання студентом проєкту 22.05.2020

3. Вихідні дані до проєкту Назва. Онлайн-сервіс для навчання касирів роботі з касовими апаратами.

4. Зміст пояснювальної записки

1. Аналіз існуючих рішень та обґрунтування теми_____
2. Обґрунтування засобів реалізації_____
3. Розробка сервісу для навчання касирів роботі з касовими апаратами

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо) презентація; структурна схема веб-сервісу; схема алгоритму роботи тесту веб-сервісу; схема алгоритму роботи серверу; Схема структури взаємодії React-компонентів.

6. Консультанти розділів проєкту*

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
нормо-контроль	Ярослав КЛЯТЧЕНКО		

7. Дата видачі завдання 17.09.2019

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	Вивчення літератури за тематикою проекту	15.02.2020	
2.	Розроблення та узгодження технічного завдання	20.03.2020	
3.	Аналіз існуючих рішень	29.03.2020	
4.	Підготовка матеріалів першого розділу дипломного проекту	15.04.2020	
5.	Підготовка матеріалів другого розділу дипломного проекту	24.04.2020	
6.	Підготовка графічної частини дипломного проекту	5.05.2020	
7.	Оформлення документації дипломного проекту	14.05.2020	
8.	Попередній огляд матеріалів диплому на кафедрі	20.05.2020	

Студент _____ Калитка Євгеній
(підпис) (Ім'я та ПРІЗВИЩЕ)

Керівник проєкту _____ Радченко Костянтин
(підпис) (Ім'я та ПРІЗВИЩЕ)

АНОТАЦІЯ

Метою дипломного проєкта було створення веб-сервісу для навчання касирів роботі з касовим апаратом, а саме – відпрацювання виконання найбільш вживаних касових операцій які будуть необхідні під час зміни касира.

Для виконання проєкту було проаналізовано існуючі рішення для ознайомлення працівників з касовими апаратами та їх можливостями. На основі даного аналізу було розроблено новий веб-сервіс, що дозволяє:

- Ознайомитись з касовими апаратами
- Вивчити процедуру повсякденної роботи з касовими апаратами
- Вивчити основні команди необхідні для ефективної роботи з представленими касовими апаратами
- Відточити свої навички роботи в інтерактивному тесті

Основними мовами розробки стали JavaScript для Front-End частини та Node.js для Back-End частини сервісу. Дані мови надають широкий спектр інструментів для ефективної, швидкої і зручної розробки різно-направлених веб-програм.

Ключові слова: веб-сервіс, касовий апарат, тест, JavaScript, Node.js

ANNOTATION

The purpose of the diploma project was to create a web service for training cashiers to work with the cash register, namely - to practice the most common cash transactions that will be needed when changing the cashier.

To implement the project, the existing solutions of acquainting employees with cash registers and their capabilities were analyzed. Based on this analysis, a new web service was developed that allows:

- Get acquainted with cash registers
- To study the procedure of daily work with cash registers
- To study the basic commands necessary for effective work with the presented cash registers
- Hone your skills in an interactive test

The main development languages were JavaScript for the Front-End part and Node.js for the Back-End part of the service. These languages provide a wide range of tools for efficient, fast and convenient development of multidirectional web programs.

Keywords: web service, cash register, test, JavaScript, Node.js

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ДП.045440.002 ТЗ	Онлайн-сервіс	3		
			для навчання касирів			
			роботі з касовими			
			апаратами			
			Технічне завдання			
	А4	ДП.045440.003 ТП	Онлайн-сервіс	2		
			для навчання касирів			
			роботі з касовими			
			апаратами			
			Відомість технічного			
			проєкту			
	А4	ДП.045440.00.004	Онлайн-сервіс	54		
			для навчання касирів			
			роботі з касовими			
			апаратами			
			Пояснювальна записка			
	А4	ДП.045440.01.005	Структура веб-сервісу.	1		
			Структурна схема.			

Змін.	Арк.	№ докум.	Підпис	Дата	ДП.045440.001 ОА		
Розробив		Калитка Є.О.			Онлайн-сервіс для навчання касирів роботі з касовими апаратами **Опис альбому**	Літ.	Арк 1 / Арк 2
Перевірив		Радченко К.О.					
Консульт.						КПІ ім. Ігоря Сікорського, ФПМ КВ-62	
Н. контроль		Клятченко Я.М.					
Зав. каф.		Романкевич В.О.					

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ДП.045440.02.006	Алгоритм роботи тесту	1		
			веб-сервісу.			
			Блок-схема алгоритму.			
	А4	ДП.045440.03.007	Алгоритм роботи серверу.	1		
			Блок-схема алгоритму.			
	А4	ДП.045440.04.008	Структура взаємодії	1		
			React-компонентів.			
			Структурна схема.			

Змін.	Арк.	№ докум.	Підпис	Дата	ДП.045440.001 ОА	Арк. 2

ЗМІСТ

					ДП.045440.002 ТЗ		
Зм	Арк.	№ докум.	Підп.	Дата			
Розроб.		Калитка Є.О			Онлайн-сервіс для навчання касирів роботів з касовими апаратами **Технічне завдання**	Літ.	Аркуш 1 / Аркушів 3
Перев.		Радченко К.О.					
Н.контр.		Клятченко Я.М.				КПІ ім. Ігоря Сікорського, ФПМ КВ-62	
Затв.		Романкевич					

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ РОЗРОБКИ

Назва розробки: «Веб-сервіс для навчання касирів роботі з касовими апаратами».

Галузь застосування: веб-індустрія, бізнес.

2. ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на дипломне проектування на здобуття першого (бакалаврського) рівня вищої освіти, затверджене кафедрою системного програмування і спеціалізованих комп'ютерних систем Національного технічного університету України «Київський Політехнічний Інститут імені Ігоря Сікорського».

3. МЕТА І ПРИЗНАЧЕННЯ РОБОТИ

Метою проекту є створення веб-сервісу, який би виконував поставлені перед ним задачі.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелом інформації є технічна та науково-технічна література, технічна документація, публікації у періодичних виданнях та електронні статті у мережі Інтернет.

5. ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до програмного продукту, що розробляється

- сумісність із всіма популярними браузерами
- добре виконаний UI/UX інтерфейс;
- адаптивна HTML/CSS розмітка;
- якісно виконана Front-End частина веб-сервісу;
- оптимальна серверна частина веб-сервісу;

5.2. Вимоги до апаратного забезпечення

- Процесор: Intel Pentium G5600F;
- Оперативна пам'ять: 2 Гб;
- Наявність доступу до мережі Internet (GPRS, EDGE, 3G, 4G);

5.3. Вимоги до програмного та апаратного забезпечення користувача

— Можливість запуску будь-якого із сучасних браузерів;

6.ЕТАПИ РОЗРОБКИ

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	Вивчення літератури за тематикою проекту	15.02.2020	
2.	Розроблення та узгодження технічного завдання	20.03.2020	
3.	Аналіз існуючих рішень	29.03.2020	
4.	Підготовка матеріалів першого розділу дипломного проекту	15.04.2020	
5.	Підготовка матеріалів другого розділу дипломного проекту	24.04.2020	
6.	Підготовка графічної частини дипломного проекту	5.05.2020	
7.	Оформлення документації дипломного проекту	14.05.2020	
8.	Попередній огляд матеріалів диплому на кафедрі	20.05.2020	

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ДП.045440.00.004 ПЗ	Онлайн-сервіс	54		
			для навчання касирів			
			роботі з касовими			
			Пояснювальна записка			
	А4	ДП.045440.01.005 Д1	Структура веб-сервісу.	1		
			Структурна схема.			
	А4	ДП.045440.02.006 Д2	Алгоритм роботи тесту	1		
			веб-сервісу.			
			Блок-схема алгоритму.			
	А4	ДП.045440.03.007 Д3	Алгоритм роботи серверу.	1		
			Блок-схема алгоритму.			
	А4	ДП.045440.04.008 Д4	Структура взаємодії	1		
			React-компонентів.			
			Структурна схема.			

Змін.	Арк.	№ докум.	Підпис	Дата	ДП.045440.003 ТП		
Розробив		Калитка Є.О.			Онлайн-сервіс для навчання касирів роботі з касовими апаратами **Відомість технічного проекту**	Літ.	Аркуш 1 / Аркушів 1
Перевірив		Радченко К.О.					
Консульт.						КПІ ім. Ігоря Сікорського,ФПМ КВ-62	
Н. контроль		Клятченко Я.М.					
Зав. каф.		Романкевич В.О.					

Пояснювальна записка
до дипломного проєкту

на тему: Онлайн-сервіс для навчання касирів роботі з касовими апаратами

Київ – 2020 року

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

DARPA - Defense Advanced Research Projects Agency
ARPAN - Advanced Research Projects Agency Network
NCP – Network Control Protocol
TCP - Transmission Control Protocol
IP - Internet Protocol
DNS - Domain Name System
NFSnet - National Science Foundation Network
HTTP - HyperText Transfer Protocol
HTML - HyperText Markup Language
URL - Uniform Resource Locator
UX - User Experience
UI - User Interface
CSS - Cascading Style Sheets
NFC - Near field communication
DON - Document Object Model

ВСТУП

З самого початку свого існування перед людством стояла задача опрацювання фінансових операцій при торгівлі. В давні часи це був звичайний бартер, який з винайденням ваг і впровадженням валют пережив цілу революцію в результаті якої фінансові операції переросли в цілу науку.

Розвиток комп'ютерних технологій по своїй значущості для фінансової сфери можна прирівняти до появи ваг у античні часи. З появою касових апаратів появилась можливість швидко і зручно опрацьовувати фінансові операції між покупцем і продавцем, а саме головне значно спростити катологізацію і підрахунки прибутків для великих торгових компаній. В наш час важко уявити велику мережу магазинів яка б не використовувала касові апаратів.

Проте з розвитком комп'ютерів і електроніки як науки не стояли на місці і способи оплати. Банковські безготівкові розрахунки або безконтактна оплата NFC виділяється своє простотою і зручністю порівняно з традиційно готівковим способом оплати. Можна впевнено сказати, що з плином часу методи електронної оплати повністю виштовхнуть готівку з ринку. В таких умовах наявність касового апарату для обробки даних способів оплати стає необхідністю не тільки для великих фірм, а й для будь-якого підприємця.

Збільшення попиту на касові апарати породило значну конкуренцію між їх виробниками і значно збільшило модельний ряд даних пристроїв, кожний з яких відрізняється функціоналом, способами обробки фінансових операцій і багатьма іншими нюансами. При такій кількості представлених на ринку рішень з'являється потреба в навчанні персоналу користуванню конкретною моделлю касового апарату яку використовує фірма або підприємець.

Для рішення даної проблеми було вирішено створити веб-додаток в якому централізовано можна було б зберігати інформацію по використанню більшості типів касових апаратів представлених на ринку. Все, що потрібно зробити новому робітнику це зареєструватися на веб-сервісі та обрати модель

касового апарата з яким йому доведеться працювати на новій роботі. В результаті працівник матиме доступ до короткої довідки по основним касовим операціям обраного касового апарату, денного розпорядку при експлуатації касового апарату, а також матиме змогу закріпити свої навички в інтерактивному тесті. Дане рішення призване спростити роботу як і для роботодавців яким більше не доведеться створювати свої власні методи навчання персоналу, так і для нових робітників які тепер матимуть змогу знайти всю необхідну їм інформацію на одному сервісі.

1. АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ТА ОБҐРУНТУВАННЯ ТЕМИ ДИПЛОМНОГО ПРОЄКТУ

1.1 Що таке веб-сервіс та історія їх появи

Поява веб-сервісів, як і будь-яких інших веб-додатків нерозривно пов’язана з без сумніву одним з найважливіших винаходів XX століття – мережі Інтернет.

Історія створення всесвітньої павутини почалася в 1962 році з висловлення Джозефом Ліклайдером на той час керівником Агенства передових оборонних дослідницьких проєктів США ідеї Всесвітньою комп’ютерної мережі. В 1969 році почалася розробка того, що в майбутньому стане називатися інтернетом. Міністерство оборони США почало фінансування системи надійної передачі інформації, яка була реалізована у вигляді комп’ютерної мережі запропонованої агенством DARPA. В розробкі даної системи приймали участь: Каліфорнійський університет Лос-Анджелеса, Стенфорський дослідницький центр, Університети штатів Юта і Каліфорнія. Розроблена мережа яка в майбутньому переросте в всесвітню павутину дали назву ARPANE.

29.10.1969 року в 21.00 між двома першими вузлами мережі ARPANE які розташовувались на відстані 640 кілометрів один від одного відбувся перший сеанс зв’язку. Саме цю дату можна вважати днем народження Інтернету. Вже в 1970 році мережа активно використовувалась для пересилки електронної пошти, в цей час з’являються перші групи новин та дошки оголошень.

Проте, незважаючи на стрімкий ріст популярності, залишалася проблема стандартизації передачі даних. Мережа просто не могла взаємодіяти з іншими системами побудованими на інших технічних стандартах. Рішення цієї проблеми відбулося у 1983 році при переході з протоколу NCP на всім добре відомий протокол TCP/IP. Саме після цього за мережею ARPANE закріпилося всім відома назва “інтернет”.

У 1984 році було розроблено систему доменних назв DNS. В той же час ARPANE почала виштовхувати новостворена Національним науковим фондом США NSFnet. В 1989-1990 роках ARPNE припинив своє існування програвши NFSnet основною перевагою якої була значно вище пропускна здатність. Також в цей час британський вчений Тім Бернер-Лі розробив протокол HTTP, мову гіпертекстової розмітки HTML та ідентифікатори URL.

В 1992 році компанія Sprint Corporation вперше почали подавати послугу пілключення до мережі інтернет. Саме після цього можна констатувати становлення інтернету таким яким ми його знаємо.

Після початку надання послуг підключення до інтернету масовому користувачу почали активно розвиватися всі типи веб програм, а також способи їх розробки. Даний розвиток не обійшов стороною і веб-сервіси які в той час почали оформлятися в свою остаточну форму.

Веб-сервіси – це як правило односторінкові веб-програми за допомогою яких користувач в режимі реального часу отримує послугу або виконує певні операції, наприклад: обробляє фотографії, створює презентації, перераховує одну валюту в іншу тощо. Головною задача веб-сервісу – бути корисним, користувач повинен вирішувати свою задачу швидко і з мінімальними зусиллями. По цій же причині інтерфейс має бути максимально лаконічним.

Яскравим прикладом веб-сервісів є:

- Каталоги
- Пошук товарі і порівняння цін
- Онлайн тести
- Онлайн консультації
- Системи бронювання квитків

Загалом веб-сервіси за своїм функціоналом нагадують програми або мобільні додатки проте розташовані у веб-просторі, що значно спрощує користувачеві доступ до їх функціоналу.

Веб-сервіси як правило призвані вирішувати якусь вузько спеціалізовану задачу для людей із вузькоспеціалізованими вимогами. По цій причині хороший веб-сервіс повинен відповідати таким критеріям:

- Детально пропрацьований UI/UX дизайн – оскільки веб-сервіс як правило робиться на замовлення під певну унікальну задачу необхідне якісне виконання всього інтерфейсу сервісу для швидкого і простого виконання поставлених перед ним завдань.
- Лаконічність інтерфейсу – користувач використовує веб-сервіс для вирішення поставленого перед ним завдання. Ніщо не повинно відволікати або заплутувати користувача при роботі із веб-сервісом
- Добре пророблена Front-End частина сервісу – даний критерій необхідний не тільки для правильної, безпомилкової роботи сервісу в цілому, а й для спрощення його підтримки і модифікації в майбутньому.
- Детально пророблена Back-End архітектура – являється критично важливою при створенні веб-сервісів які взаємодіють з великим потоком інформації. Втрата важливих даних може спричинити великі проблеми як для користувача сервісу та і для його оператора.
- Адаптивна HTML/CSS верстка сервісу – в наш час важко перелічити весь спектр пристроїв на яких може відображатися веб-сервіс. Це може бути смартфон, планшет, вузько форматний чи широко форматний монітор. Через це коректність відображення інформації та інтерфейсу сервісу на різних засобах виводу інформації відіграє важливу роль у формуванні позитивного користувацького досвіду

1.2 Актуальність теми

З приходом комп’ютерів в банківську і фінансову сферу почали активно розвиватись системи безготівкової оплати. За статистикою за 2019 рік в Україні кількість безготівкових операцій складала 5057.3 млн. штук, а їх

грошовий обсяг – 3576.7 млрд грн. Порівняно з минулим роком зріст становить 29.2% і 24.3% відповідно. Загальний обсяг безготівкових операцій від усіх фінансових операцій становить 50.5%, а ще 5 років тому показник складав 25%. На цьому фоні можна зробити висновок в планомірному витисканні готівкових методів оплати з ринку.

До того ж зростає загально доля платіжних безконтактних карток, смартфонів та інших NFC-пристроїв. Це зумовлено загальною простотою, надійністю та швидкістю оплати при використанні системи NFC. На даний момент 30% всіх безконтактних операцій у торгових мережах припадають саме на NFC пристрої і цей показник зростає з кожним роком.

На цьому фоні відбувається значне розширення платіжною системи по всій країні. В першу чергу безкотакної, що обумовлено різким збільшенням попиту на цей вид оплати. Тільки за 2019 рік кількість терміналів безконтактної оплати зросла на 19.7%, а загалом з 2015 року їх кількість збільшилася вдвічі.

В результаті цього можна констатувати, що в найближчий час наявність касового апарату для проведення електронних фінансових операцій, в особливості безконтактної оплати, стане не приємним бонусом для покупця, а абсолютною необхідністю для будь-якого бізнесу.

Проте сучасні касові апарати являють собою досить складну систему яка здатна до обробки десятків різних фінансових операцій. А збільшення складності певної системи закономірно породжує більш високі вимоги до персоналу який їх обслуговує. Таким чином само собою виникає питання: “А яким чином робітник, а інколи і сам власник бізнесу може отримати інформацію про всі можливості, які має використовуваний ним касовий апарат?”.

Великі мережі магазинів створюють свої спеціалізовані курси для підвищення кваліфікації працівників і часто мають підтримку від виробника використовуваної ними марки касового апарату. Але такі курси як правило поширюються лише на робітників певної мережі і навчають роботі лише з тою

маркою апаратів яку масово експлуатує даний магазин. Малий бізнес в свою чергу може опиратися лише на інформацію доступну у відкритих джерелах.

Проаналізувавши доступні варіанти нажаль можна констатувати, що якогось універсального рішення даної проблеми просто не існує. Залежно від марки і виробника косового апарата можна наткнутися, як на короткі довідки касира з переліком доступних команд та дій так і на інструкції по експлуатації тим чи іншим пристроєм. Що перший ,що другий варіант досить слабо підходять для швидкого і ефективного навчання персоналу взаємодії з касовими апаратами оскільки:

- Короткі довідки касира часто не несуть в собі необхідної кількості інформації необхідної для продуктивної роботи
- Мануали навпаки мають забагато непотрібної оператору даного пристрою інформації, в результаті чого на їх вивчення може бути необхідно дуже багато часу
- Інформація не централізована із-за чого при покупці певного касового апарату в його оператора можуть з’явитися проблеми з пошуком актуальної інформації по використанню відповідного пристрою, що може призвести до пошуку інформації в не офіційних джерелах, які в свою чергу можуть надати хибну інформацію
- Відсутність будь-яких способів перевірити на практиці освоєний матеріал, що в свою чергу може спричинити помилки при реальній роботі з касовим апаратом

1.3 Обгрунтування теми дипломного проєкта

Проаналізувавши усі доступні варіанти навчання персоналу роботи з касовими системами можна констатувати відсутність спеціалізованого сервісу який би в повній мірі задовольняв цю потребу. Але саме подібного сервісу потребує велика кількість підприємців у сфері послуг. І ця потреба з ростом

долі безготівкових фінансових операцій і як результат ростом загальної кількості касовим систем у використанні продовжуватиме зростати.

Тому в якості мети дипломного проекту встановлено розробити доступний, зручний та простий у використанні веб-сервіс який дозволе перекрити потребу малого бізнеса в подібного виду послугах.

Для отримання якісного продукту при розробці програмного засобу встановимо основні вимоги яких повинен дотримуватись створений веб-сервіс, а саме:

- Простота і лаконічність інтерфейсу
- Детально пророблена інформаційна частина яка матиме всю необхідно користувачеві інформації і в той же час не буде переповнена непотрібними деталями
- Можливість пройти тест на основі представленого в інформаційній частині матеріалу, що дозволить на практиці закріпити отримані навички перед початком реальної роботи
- Адаптивна HTML/CSS верстка, що забезпечить можливість працювати з веб-сервісом на різного форм-фактору пристроях.

Висновок до розділу 1

У даному розділі було пояснену причини різкого збільшення попиту на касові апарати на ринку, а також обґрунтована необхідність створення веб-сервісу для навчання персоналу роботі з даними апаратами.

Було проаналізовано представленні на ринку аналоги, а також вивченні основні нюанси створення веб-сервісів. На основі чого були сформовані завдання які повинна виконувати розроблена програма, а також критерії яких вона повинна дотримуватись.

2.ОБҐРУНТУВАННЯ ЗАСОБІВ РЕАЛІЗАЦІЇ

2.1 Основні засоби розробки

В якості основних засобів розробки для веб-сервісу було використано добре відому будь-якому веб-розробнику і прекрасно зарекомендовану з плином часу зв’язку HTML/CSS/JavaScript для Front-End частини та Node.js для Back-End частини сервісу.

Почнемо з Front-end частини проєкта і розберемо більш детально причини вибору саме такої взаємодії мов програмування і загальну тенденцію використання цієї зв'язки при розробці практично будь якого веб-ресурсу. Для більш кращого розуміння ролей які виконує кожна із представлених мов представимо веб-сервіс у виді людини.

HTML – це мова гіпертекстової розмітки яка була розроблена вченим Тімом Бернерсом-Лів в 1991 році і являється наслідником метамови SGML. На початку свого існування HTML задумувався як платформа для простого структурування і форматуванні документів пов’язаних з науковою та інженерною діяльністю, людьми які не є спеціалістами в області верстки. HTML успішно вирішував проблему складності SGML шляхом введення набора структурних і семантичних елементів – дескрипторів. З плином часу за дескрипторами закріпилась назва “теги”. Метою введення HTML було відображення тексту без стилістичних помилок на пристроях різного форм-фактору.

З цього ми можемо зробити висновок, що вже з перших днів існування HTML отримав плюси які були, є і будуть незамінними в області веб-розробки, а саме:

- Простота структурування документа, а в майбутньому і веб-сайтів шляхом використання невеликої кількості структурних елементів - дескрипторів(тегів)
- Можливість форматування документа без прив’язки до засобів відображення, що знову ж таки враховуючи незчисленну кількість пристроїв які можуть мати доступ для веб-ресурсу є важливим

З плином часу HTML відійшов від своєї початково задуманої ролі в користь потреб мультимедійного і графічного оформлення в веб-просторі. А завдяки W3C яка випустила HTML з єдиним набором тегів HTML набув статусу універсальної мови розмітки сайтів.

Завдяки плюсам переліченим вище сучасний HTML являється відносно простим способом розмітки веб-програм які завдяки його універсальності будуть коректно відображатися у всіх сучасних браузерах і на пристроях різного форм-фактор будь то смартфон, ноутбук тощо. HTML розмітка виконує роль такого собі “скелету” до якого кріпляться всі інші частини веб-сервісу.

CSS (Cascading Style Sheet – каскадні табиці стилів) – формальна мова яка описує зовнішній вид документа написаного з використання мови гіпертекстової розмітки HTML. Наразі є невід’ємною частиною будь якої веб-програми.

CSS представляє собою широкий спектр технологій які було одобрено консорціумом W3C і стандартизовано до єдиного набору правил на основі яких веб-дизайнери і програмісти проектують сайти. Сама ідея каскадних таблиць стилів була запропонована норвезьким вченим і спеціалістом в області інформаційних технологій Хоконом Віум Лі який в той час і працював на W3C.

Перша версія CSS дозволяла встановлювати розмір шрифта і його стилі, визначати рамки, колір тексту і фону, відстань між словами, внутрішні і зовнішні відступи. Тобто перша версія була направлена сугубо на стилізацію розмітки HTML.

Проте з розвитком веб-ресурсів з’явилась потреба в більш “кастомному” позиціонуванні елементів HTML розмітки. Саме на це і була направлена друга версія CSS і цей напрямок (позиціонування елементів) активно розвивається і тепер flexbox і bootstrap є яскравим прикладом цього розвитку.

Третя версія, крім покращення всіх аспектів мови, перенесла CSS на модульну структуру і в подальшому кожний окремий модуль розвивався незалежно один від одного.

Сучасний CSS є незамінним інструментом веб-дизайнера який дозволяє реалізувати будь-які побажання клієнта в області візуального оформлення веб-продукту. Загалом CSS виконує роль “шкіри” веб-сервісу яка відповідає за всю його візуальну частину.

JavaScript – мова програмування, що підтримує об’єктно орієнтований, імперативний і функціональний стилі програмування. Являється реалізацією стандарту ECMAScript. Особливо широкого застосування набув в браузерах в якості мови сценаріїв для надання веб-сайтам інтерактивності. На даний момент являється основною і універсальною мовою веб-програмування.

Спочатку в структурі HTML-документа застосовувались лише методи оформлення тексту, пізніше з’явилася можливість вставки і проігрування звукових файлів і відео кліпів. З появою і введенням у використанні CSS з’явилась можливість широкої стилізації HTML-документів. Проте залишалася проблема динамічної обробки і реакції на дії користувача. HTML не міг задовольнити подібні потреби, адже створювався для зовсім інших цілей. Таким чином з’явилася необхідність в застосуванні ще однієї мови програмування.

В 1995 році фірма Netscape випустила браузер Netscape Navigator з підтримкою мови LiveScript. Функціонал даної мови був досить куций, але для свого часу він виявився досить практичним і прогресивним. Трохи пізніше назву LiveScript було замінено на JavaScript. Нинішньої популярності і статусу основної мови веб-програмування JavaScript набув завдяки неймовірно популярному браузеру тих часів Internet Explorer, який використовував свою редакцію JavaScript під назвою Jscript. Шалена популярність цього браузера спонукала розробників інших браузерів інтегрувати підтримку мови JavaScript. В результаті чого весь майбутній розвиток цієї мови був направлений саме на веб-розробку.

Хоча JavaScript являє собою об’єктно орієнтовану мову програмування, використання прототипування обумовлює деякі відмінності в роботі з об’єктами.

На відміну від клас-орієнтованих мов в JavaScript не потрібно спочатку створювати клас, а потім об’єкт класу. Замість цього ми можемо відразу створити об’єкт. Проте під час створення великих веб-проектів обов’язково з’явиться необхідність створення великою кількості однотипних об’єктів. На допомогу прийде конструктор, який в JS є звичайною функцією. За допомогою нього можна створити велику кількість однотипних незалежних об’єктів, але кожний створений нами об’єкт займає місце в пам’яті включаючи його властивості і методи, хоча багато з них можуть повторюватись. Щоб цього уникнути використовуються прототипи. У кожному новоствореному об’єкті зберігається ссилка на його прототип. В результаті цього ми можемо централізовано зберігати методи і спільні властивості об’єктів в їх прототипі.

Слід зазначити, що в нових версіях JS були добавлені класи. Проте вони не виконують стандартної для них ролі. А являються лише більш зручною обгорткою написаного вище.

Говорячи про об’єкти в JavaScript не можна обійти стороною питання економії пам’яті. Враховуючи можливість запуску десятків, а то і сотень вкладок коректне використання веб-ресурсами доступної пам’яті є критичною для швидкої роботи браузера. На фоні цієї проблеми JavaScript має значну перевагу над іншими мовами, а саме автоматичне керування пам’яттю і складальник сміття.

Розглянемо більш детально його роботу, нехай в змінній student знаходиться ссилка на об’єкт (Рис. 2.1)

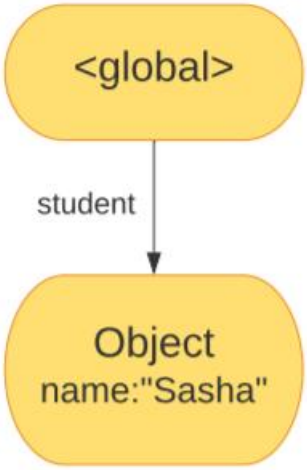

Рисунок 2.1 – Взаємозв’язок об’єкту і змінної яка на нього вказує

На рисунку стрілкою позначено ссилку на об’єкт. Змінна student зсилається на обєкт {name:”Sasha”}. Тепер перезапишемо значення змінною student в результаті чого втратимо ссилку на об'єкт (Рис 2.2)

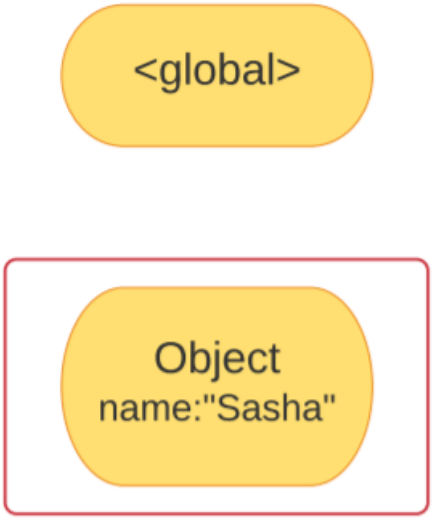

Рисунок 2.2 – Об’єкт втратив будь-які ссилки які б вказували на нього

Тепер об’єкт стає повністю недоступним. До нього немає доступу оскільки змінна через яку ми могли б отримати до нього доступ була перезаписана. В результаті складальник сміття видалить такий об’єкт і звільнить пам’ять. Тобто JavaScript з певною періодичністю перевіряє кожен створений об’єкт на його доступність, в випадку якщо будь які зв’язки з об’єктом відсутні він буде автоматично видалений.

Крім цього в даній мові програмування присутні елементи функціональних мов, наприклад: функції як об’єкти першого класу,

замикання, об’єкти як списки, анонімні функції, що надає мові додаткової гнучкості і функціональності.

Загалом до особливостей архітектури JavaScript можна віднести такі пункти:

- Динамічна типізація – метод, що використовується в мовах програмування при якому змінна зв’язується з типом в момент присвоєння значення, а не в момент її об’яви
- Слабка типізація – характеристика яка характеризує можливість мови автоматично проводити неявні перетворення автоматична, навіть якщо це може привезти до втрати точності
- Прототипне програмування – стиль об’єктно орієнтованого програмування який характеризується відсутністю класів, а наслідування відбувається шляхом вже існуючого екземпляра об’єкта – прототипа (В нових версіях JavaScript були добавлені класи, але вони виконують роль “синтаксичного цукру”)
- Автоматичне керування пам’яттю – JavaScript має автоматичний складальник сміття, який час від часу звільняє пам’ять і видаляє об’єкти доступ до яких було втрачено.

В розробленому веб-сервісі JavaScript виконує роль “м’язів”, дана мова програмування відповідає за всі динамічні дії веб-ресурсу, обробку дій користувача та зв’язок з Back-End частиною ресурсу. Враховуючи більш ніж 20 років веб-направленої розробки даної мови, JavaScript нині являється найкращим вибором при розробці будь яких веб-програм оскільки:

- Має повну інтеграцію з HTML/CSS
- Прості речі робляться просто
- Підтримує всі можливі браузери і встановлена по замовчуванню
- Має обширний набір бібліотек і фреймворків, що здатні задовольнити будь які потреби веб-розробника

З усього вище сказаного можемо зробити висновок, що вибір в якості основних засобів створення веб-сервісу зв’язки HTML/CSS/JavaScript є

повністю виправданою, адже ці мови практично з самого свого зародження були взаємопов’язані, в результаті чого вони розвивалися і доповнялися враховуючи особливості і нюанси архітектури один одного. Історія використання зв’язки цих 3 мов нараховує більше 20 років і на даний момент дозволяє створювати веб-ресурси будь-якої складності і формату.

Перейдемо до Back-End сторони веб-сервісу. В якості мови програмування для обробки інформація яка надійшла від Front-End частини було обрано Node.js.

Node.js – програмна платформа основана на двигуні V8 яка перетворює мову JavaScript із спеціалізованої в мову загального призначення.

Раніше JavaScript міг запускатися лише в браузерах. Проте з появою Node.js який перетворює JS в машинний код, який комп’ютер може запускати без необхідності в його інтерпретації дозволило використовувати JavaScript в нових для нього сферах. Враховуючи сферу використання JavaScript і професійну діяльність людей які ним користуються, немає нічого дивного, що Node.js знайшов своє місце і в веб-розробці, в якості мови Back-end частини веб-програм. Практично однаковий синтаксис і особливості архітектури як JS так і Node.js дозволило веб-розробникам вивчаючи фактичну єдину мову розробляти як Front-End так і Back-End частини сайтів.

То які ж переваги має Node.js порівняно з іншими серверними мовами?

- Як правило Front-End частина веб-програми написана на JS. В такому випадку розробник може розраховувати на універсальність коду в використовуваному стекі технологій
- Інструменти на кшталт webpack-у допомагають використовувати одні і ті ж частини коду як на сервері, так і на клієнті, що спрощує розробку і підтримку такої програми
- При використанні JS на клієнті і на сервері можна створити веб-програму яка буде рендеритись і в браузері, і на сервері. При цьому розробник може розраховувати на чітку і зрозумілу роботу системи.

- Відносно недавня поява в Node.js конструкції async/await, яка повністю змінила підхід до написання асинхронного коду. Завдяки цій конструкції асинхронний коди нагадує синхронний і по своїй конструкції і по поведінці
- Можливість із Node.js звертатись до сторонніх бібліотек написаних іншими мовами програмування, тобто якщо є якась провірена часом бібліотека, наприклад на C++, яка виконує якісь складні обрахунки, то розробнику не потрібно шукати її аналоги. Він може за допомогою Node наладити взаємодію з даною бібліотекою використовуючи мікросервіс і звертатися до потрібних функцій цієї бібліотеки використовуючи REST API

З усього вище переліченого можна виділити кілька типів веб-проєктів використання в яких Node.js буде більш ніж виправдане. Розглянемо їх:

- Веб-програми в реальному часі – це можуть бути онлайн чати, системи онлайн коференцій, онлайн ігри, тощо. Тобто ті види програм, швидкість реакції яких на дії користувача можна охарактризувати як “негайно”. Для нормальної роботи таких програм, система на якій вона засновано має характеризуватись вискою швидкістю обробки інформації і малими затримками. Node.js відповідає цим вимогам адже його архітектура забезпечує швидку обробку великої кількості викликів від клієнтів, просту роботу із сторонніми бібліотеками та чудову синхронізацію даних між клієнтом і сервером
- Невеликі веб-програми – перевагою Node є те, що серверну частину проєкта можна масштабувати в залежності від поставлених перед розробником задач. Node дозволяє створювати маленькі програми, не підтримуючи при цьому величезної інфраструктури, більша частина якої просто не буде задіяна, а в випадку розростання програми

дозволяє швидко і просто розширити серверну архітектуру відповідно до поставлених задач.

- Односторінкові веб-програми – це програми які представляють собою одну єдину веб-сторінку. Вміст таких програм динамічно обновлюється на основі дій користувача. Односторінкові веб-програми стали великим кроком вперед в області веб-розробки і хоча більша частина навантаження в таких проєктах лежить на клієнтській частині актуальною залишається проблема рендерінгу нових даних. Ця проблема може бути вирішена методом серверного рендерингу основаного на можливостях Node.js.

Тож враховуючи той факт, що веб-сервісами являються як правило односторінкові програми і при їх розробці досить гостро стоїть проблема клієнт-серверної синхронізації та обробки великої кількості викликів від клієнтів, можемо зробити висновок, що вибір Node.js в якості мови Back-End частини веб-сервісу є повністю виправданим.

2.2 Вибір фреймворків, бібліотек та технологій для веб-сервісу

При розробці даного веб-сервісу для Front-End частини було використано такі бібліотеки: React, Redux, JQuery, для Back-End частини застосовано фреймворк Express, а для надання веб-сервісу адаптивної HTML/CSS верстки було використано технологію Flexbox.

Розглянемо більш детально причини їх використання та переваги які вони надають при розробці веб-ресурсів.

React – це декларативна ефективна JavaScript бібліотека для створення користувацьких інтерфейсів. Вона дозволяє складати складні UI з простих елементів, що називаються “компонентами”.

React був створений Джорданом Валкє, розробником програмного забезпечення, що працює на Facebook. Цією ж компанією було вперше приміненно дану бібліотеку в стрічці новин Facebook в 2011 році, а трохи

пізніше і в Instagram. Із-за використання даної бібліотеки такою великою компанією на розвиток React розвиток багато фінансів і зусиль, що в свою чергу гарантує розробникам добре описану документацію та постійні оновлення бібліотеки.

Одною із основних особливостей React є можливість використання JSX, мови програмування з близьким до HTML синтаксисом, який компілюється в JavaScript. Це дозволяю не тільки описувати HTML розмітку прямо в js файлі, а й розбивати програму на логічно незалежні між собою модулі.

Веб-сторінку написану на чистому Javascript являє собою монолітну конструкцію (Рис 2.3) в якій HTML розмітка образно “обгорнута” в CSS стилі і до якої було прикріплено обробники дій користувача, написані на JavaScript.

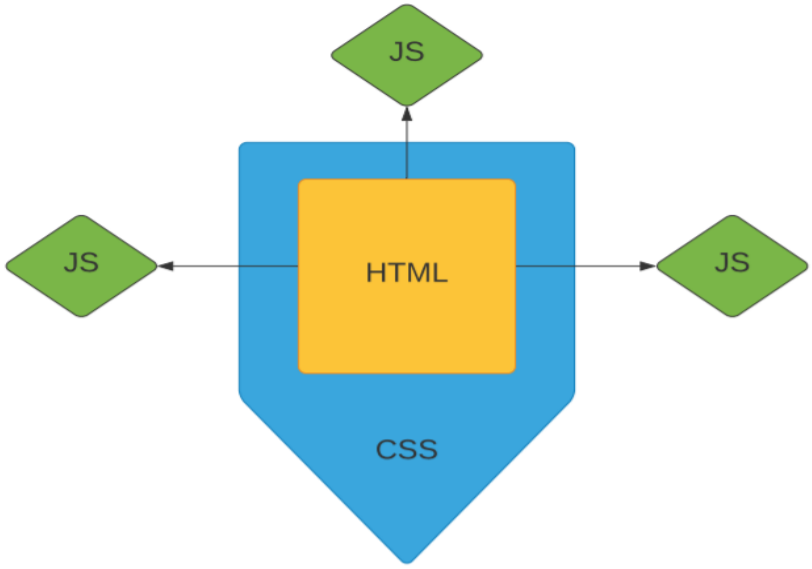

Рисунок 2.3 - Умовне зображення веб-сторінки з використання чистого JavaScript

Така структура веб-сторінки добру зарекомендувала себе як проста і надійна, проте у той же час із-за своєї монолітності вона погано піддається змінам і масштабуванню. Інколи ввести модифікації в сторінку відповідно до побажань клієнта може бути досить складно.

В свою чергу веб-сторінка написана з використанням React має модульну структуру (Рис. 2.4).

Особливість такої структури в тому що html розмітка у вигляді JSX коду

і самі обробники дій прив’язані до цієї розмітки знаходяться в одному компоненті і не зв’язані логічно з іншими компонентами.

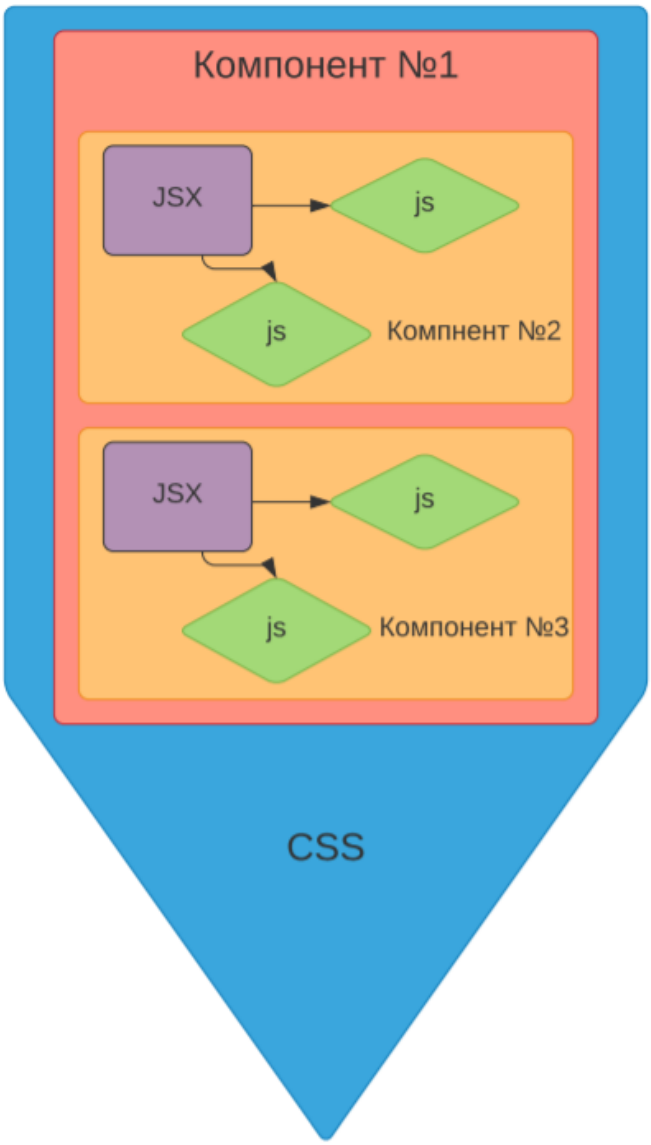

Рисунок 2.4 – Структура веб-сторінки з використанням React

Враховуючи можливість блоків визивати один одного, при правильному розбитті веб-сторінки на компоненти можна досягти не тільки модульності проекту а й повторного використання певних компонентів, що зменшить загальний обсяг роботи про розробці програмного продукту.

Загалом такий метод розробки веб-сторінок дозволяє легко їх модифікувати і масштабувати шляхом створення нових і редагування уже створених компонентів. При цьому розробник може бути впевнений, що при

зміні певних модулів він не зламає інші оскільки вони логічно не взаємопов’язані.

Ще однією не мало важливою особливістю React є використання React DOM. Дана функція є особливо корисною при розробці односторінкових веб-програм. Для розуміння всієї важливості React DOM слід пояснити, що таке звичайний DOM.

DOM – це спосіб представлення структурного документа за допомогою об’єктів. Це кросплатформенна і мовно-незалежна угода для обробки, відображення і взаємодії з даними в HTML, XML і т. д.. Тобто при отриманні HTML документа браузер преображає його в DOM дерево з яким ми можемо взаємодіяти за допомогою JavaScript і CSS.

Основна проблема DOM в тому, що він ніколи не був розрахований для створення динамічного користувацького інтерфейсу. Ми можемо змінювати представлення лише шляхом оновлення всього DOM дерева(Рис 2.5).

Як бачимо з рисунку при для відображення в нової інформації в полі нам необхідно повністю перерисувати і поле, і кнопку, і взагалі весь блок в якому вони знаходяться. Даний метод не тільки не зручний у використанні а й при наявності десятків і сотень DOM вузлів дуже ресурсу затратний і може потребувати значної кількості часу, що для односторінкової веб-програми, а в особливості веб-сервісу не допустимо.

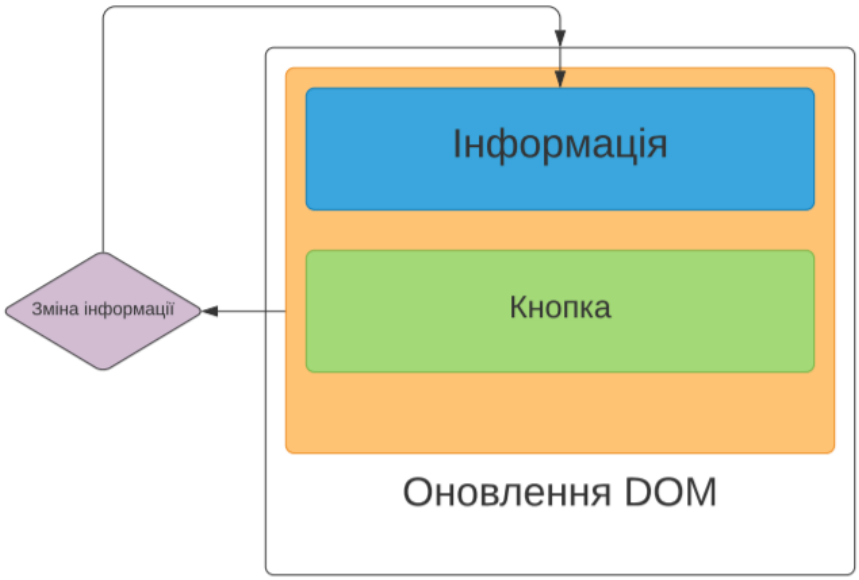

Рисунок 2.5 – Оновлення поля інформації з використанням чистого JavaScript

На відміну від DOM-елементів, елементи React це прості об’єкти які не віднімають багато ресурсів. В свою чергу React DOM автоматично обновляє DOM дерево, щоб воно відповідало переданим React-елементам. Можна сказати, що React зберігає віртуальне DOM дерево, після обновлення інформації React DOM порівнює нове віртуальне DOM дерево з його минулою версією і вносе в DOM тільки мінімально необхідні зміни(Рис 2.6).

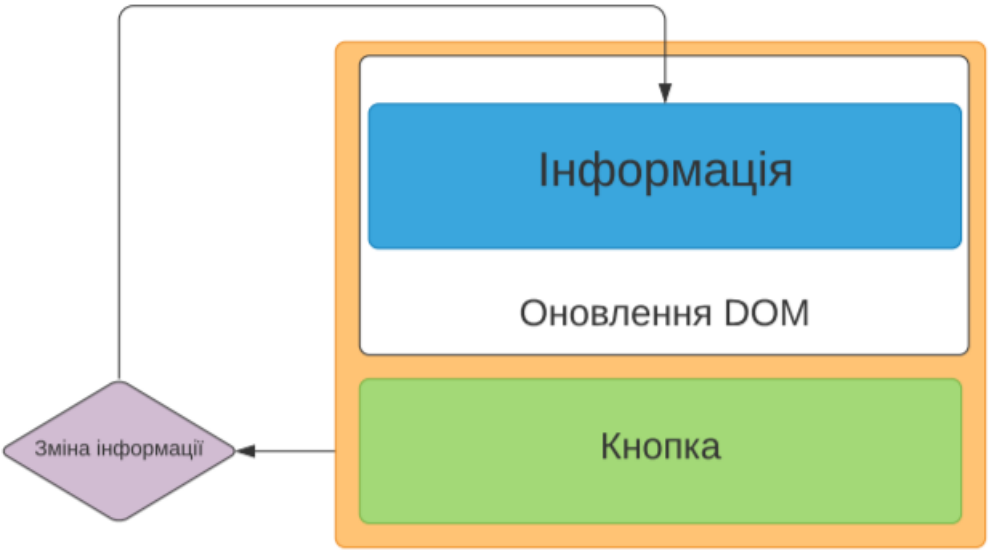

Рисунок 2.6 – Оновлення поля інформації з використанням React

З цього рисунку можна зрозуміти, що при зміні інформації React DOM перемалював лише той елемент який був змінений. Така тактика з внесенням лише необхідних змін в DOM дозволяє ефективно і швидко обновляти представлення односторінкової веб-програми і є ідеальною для реалізації на її основі веб-сервісу.

Таким чином можна зробити висновок, що бібліотека React є незамінної при створенні односторінкового веб-сервісу, адже дозволяє реалізує модульну конструкцію завдяки якій можна легко і швидко вносити зміни або добавляти нові елементи до веб-сервісу, а також дозволяє ефективно маніпулювати і обновляти DOM.

Redux – це бібліотека керування станами для програми, написана на JavaScript. Вона дозволяє писати програми, котрі ведуть себе однаково в різних умовах і легко піддаються тестуванню.

Важливо відмітити, що хоча Redux частіше за все використовується в зв’язку з React, дану бібліотеку можна використовувати з будь якими іншими view-бібліотеками. Таке часте використання Redux саме в React-орієнтвоаних проєктах пов’язане з тим, що дана бібліотеки дозволяє вирішити давню проблему контролю станів.

Для розуміння причин необхідності використання Redux, розберемо схему контролю станів в веб-програмах створених на основі React. Сама структура веб-програм розроблених з використанням цієї бібліотеки має вид дерева(Рис. 2.7).

Програма має “корневий” компонент який може викликати інші компоненти і т.д. Крім того, як вже описувалось в частині посвяченій React кожен компонент є логічно незалежним від інших, а це означає, що кожен із них має свої змінні, обробники подій тощо. Змінні які передаються ззовні компоненту зберігаються в об’єкті props, компонент може лише використовувати їх і не може змінювати. В свою чергу змінні які створюються всередині компоненту зберігаються в об’єкті state і можуть оновлюватись.

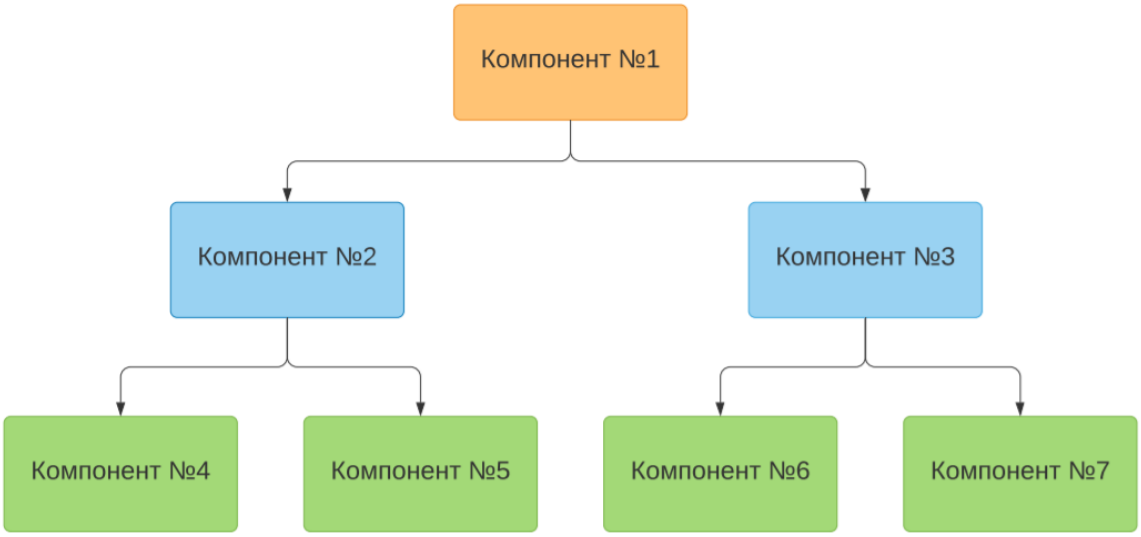

Рисунок 2.7 – Структура програми розробленої з використанням React

При розробці великої веб-програми розробник неодмінно зіткнеться з необхідністю передачі інформації від компоненту до компоненту. В такому випадку це можна реалізувати шляхом передачі необхідних параметрів до об’єкту props “дочірнього” компонета при його визові “батьківським”(Рис. 2.8)

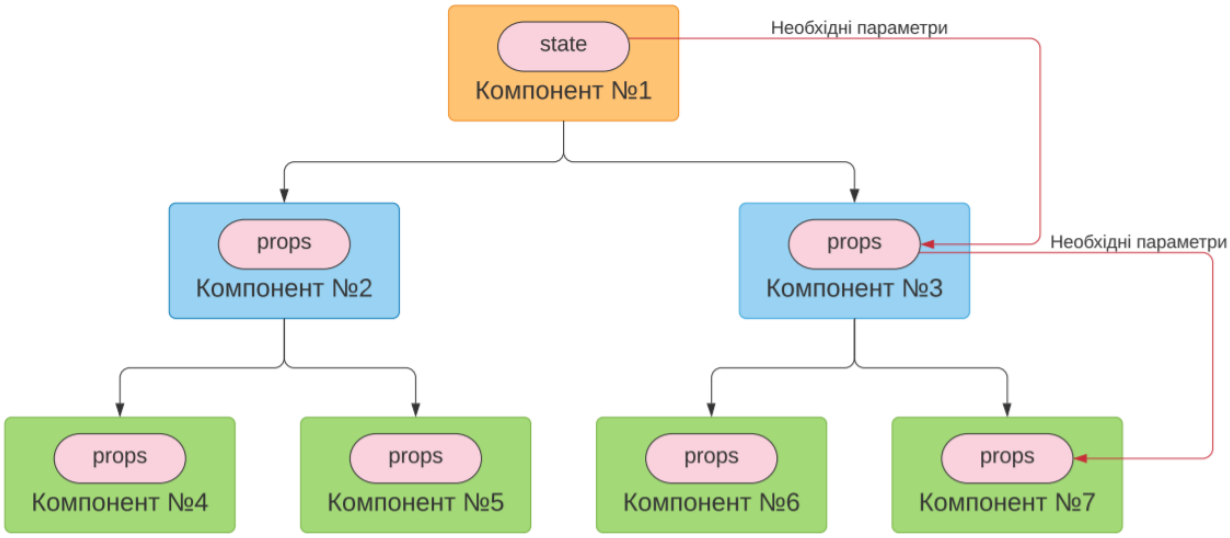

Рисунок 2.8 – Передача параметрів від “батьківського” компонента “дочірнім”

На рисунку можемо бачити, що передача параметрів на перший погляд виглядає просто і ефективно. Проте припустимо, Компоненту №7 необхідний параметр Компоненту №1. В такому випадку пряма передача неможлива. Для

задоволення такої потреби розробнику доведеться спочатку передати необхідний параметр до Компоненту №3 після чого вже з нього надати необхідну інформації Компоненту №7. Результатом такої дії стане присутність в Компоненті №3 параметру який в ньому не використовується. З розвитком програми відбувається поступове захаращення компонентів параметрами які вони просто перенаправляють вниз по дереву.

Ситуації тільки погіршується якщо в результаті маніпуляцій користувача з якимсь компонентом потрібно буде обновити вміст параметра компоненту, що знаходяться вище по ієрархії. В такому випадку доведеться застосовувати колбеки (Рис. 2.9)

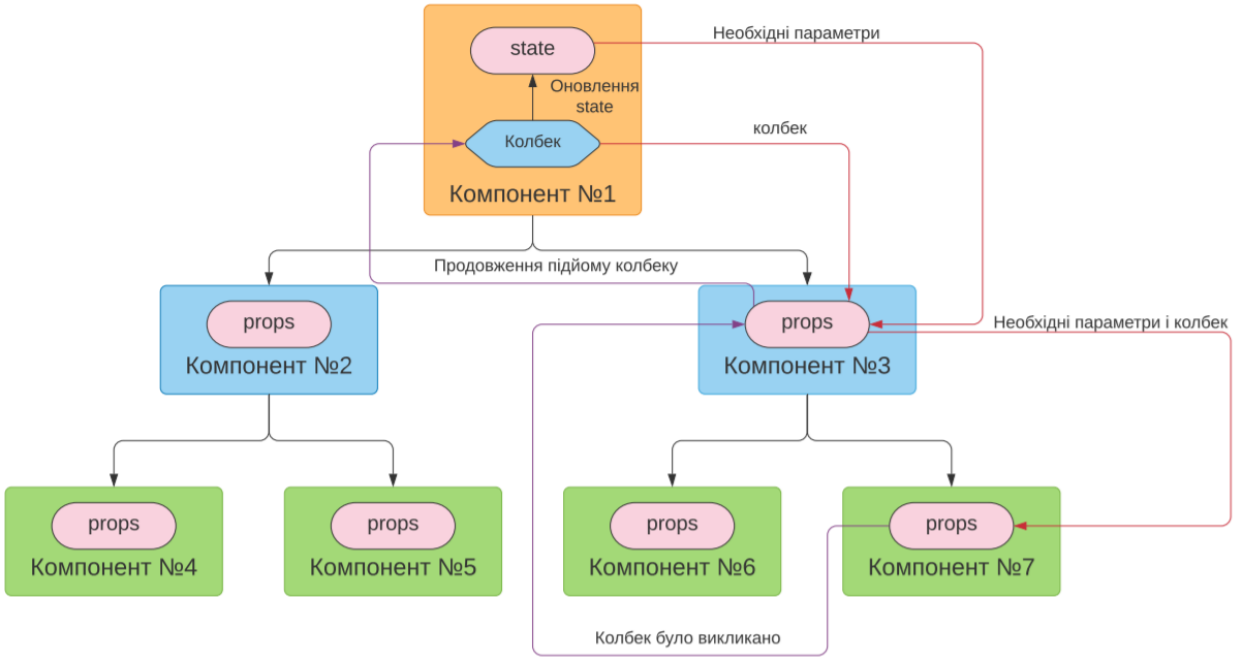

Рисунок 2.9 - Передача параметрів і їх колбеків від “батьківського” компонента “дочірнім”

Навіть при погляді на дану схему стає проблематично зрозуміти, що відбувається. В великих програмах де наявні десятки і сотні компонентів, нагромадження передаваємих параметрів та колбеків стає більшим, ніж загальна кількість параметрів які використовує сам компонент. В результаті, починає втрачатись модульність програми, її стає важко підтримувати, а сам код стає погано читаємим. Дана проблема отримала назву - “Проблема сверла”.

Частково з цією проблемою можна боротись, зберігаючи змінні параметри у кожному компоненті окремо (Рис. 2.10). Це частково покращить ситуацію, але не вирішить проблему повнісю, адже передача параметрів і колбеків продвжиться, хоч і з меншою кількістю “проміжних” компонентів. Тобіш при такій структурі розробник доб’ється лише зменшення загального захаращення програми, але не усуне його повністю.

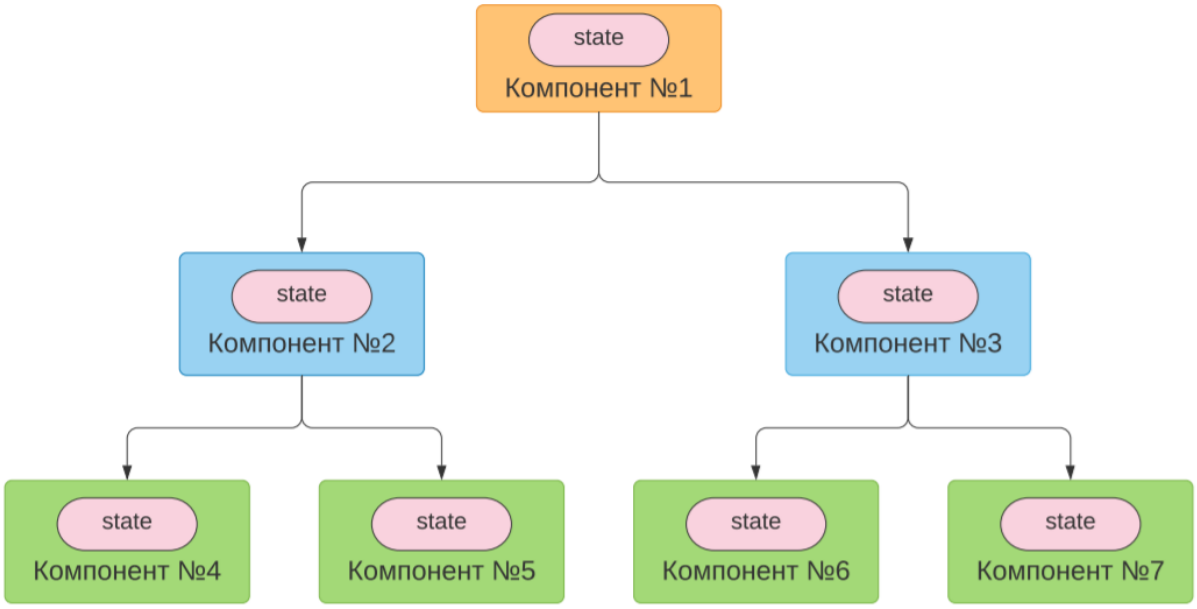

Рисунок 2.10 – Зберігання змінюваних параметрів окремо в кожному компоненті

Тепер після того, як була описана одна із головних проблем наявних у React повернемося до Redux.

Redux надає в користування розробника спеціальне сховище в якому можна зберігати параметри. Обновляти ці параметри можна лише через спеціальні структури які мають назву reducer. Які в свою чергу, запускаються спеціальними функціями які мають назву action. На перший погляд звучить складно, проте головна перевага Redux в тому, що ми можемо винести всі параметри в спеціальне зовнішнє сховище з якого вже можемо передавати необхідні компнентам дані, а також action які відповідають за зміни параметрів напряму, уникаючи ланцюга послідовних передач від компонента до компонента. Як результат використання Redux дозволяє повністю позбавити

компоненти не потрібних для їх роботи параметрів і колбеків, що в свою чергу робить програму більш простою в підтримкі, легкою в модифікації, а її код легким в розумінні. Наглядно процес використання Redux в зв'яку з React можна побачити на Рис. 2.11.

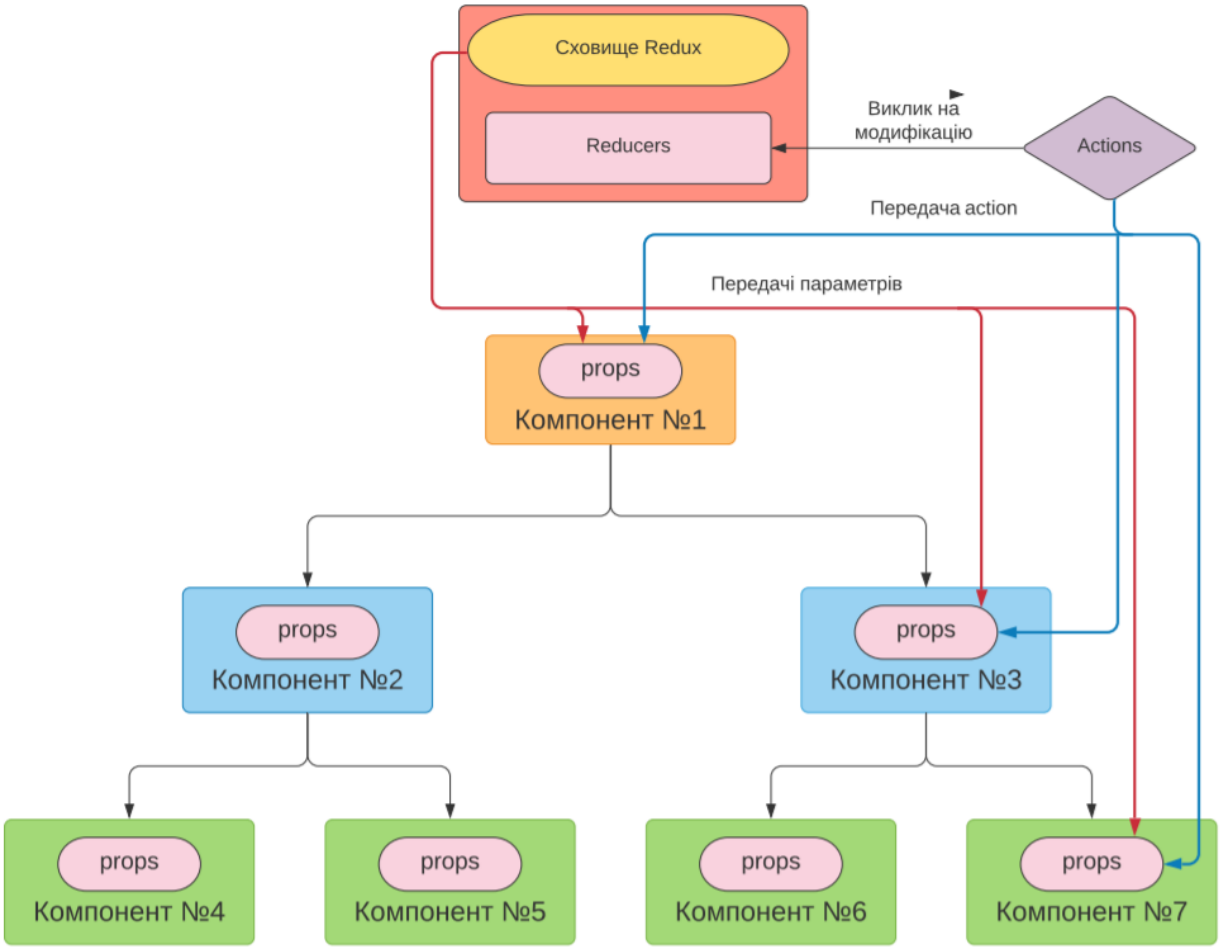

Рисунок 2.11 - Передача параметрів компонентам з використанням сховища Redux

Як можна побачити із схеми з використання Redux параметри передаються напряму компонентам які їх потребують. Також слід відмітити передачу action до компонентів. Їх використання позбавляє необхідності використання колбеків. У випадку коли, в результаті взаємодії з компонентом користувача, розробнику необхідно змінити якийсь параметр, що зберігається в сховищі Redux, він просто передає до компоненту пов'язаний з цим параметром action і викликає його передаючи в якості параметра нове значення. Після виклику action запускає reducer, який в свою чергу обновляє значення параметра в сховищі на значення яке було передано в action. Також

слід зазначити, що для можливості зміни параметру, потрібен лише action, передавати сам параметр до компоненту не є необхідним.

Як результат можна зробити висновок, що використання Redux має серйозні причини, функціонал цієї бібліотеки дозволяє уникнути і обійти одну із найбільш серйозних проблем з якою може зустрітися розробник при використанні React.

JQuery – це набір функцій в JavaScript, що фокусуються на покращенні взаємодії JS і HTML. JQuery була створена Джоном Резігом в 2006 році. Основної ціллю розробника було спрощення використання багаторазових частин коду. Дана бібліотека дозволяє легко отримувати доступ до будь-яких DOM-елементів, звертатись до атрибутів і вмісту цих елементів, а також легко ними маніпулювати. Роботу з JQuery можна розділити на 2 типи, а саме:

- Виклик глобальних методів JQuery об’єкта
- Отримання JQuery об’єкта шляхом пошуку в DOM-дереві елементів по CSS-селектору і подальша робота зі знайденими DOM-елементами

На даний момент JQuery залишається однією з найбільш популярних бібліотек при веб-розробці. Причиною цьому є певна складність доступу і зміни DOM-елементів, а також погана читаність коду при використанні чистого JavaScript.

Наприклад перед розробником стоїть задача змінити стиль кнопки при клікові на неї. В такому випадку, щоб реалізувати дану функцію без використання JQuery потрібно: знайти необхідну кнопку в DOM-дереві і занести її до змінної, змінити необхідні стилі, застосувати внесені зміни. Тобто на виконання однієї дії з DOM-елементом потрібно виконати три команди. В результаті при необхідності в великій кількості подібних змін, розробнику прийдеться мати справу з великою кількістю повторюваних частин коду.

JQuery в свою чергу дозволяє спростити цей процес і дозволяє відразу виконати всі вище перечисленні маніпуляції. В результаті чого покращується загальна читаність коду і пришвидшується швидкість розробки.

При створенні веб-сервісу бібліотека JQuery в основному використовувалася для пришвидшення доступу до DOM-елементів заради зміни їх стилів, а також для добавлення або видалення певних CSS класів.

Express – фреймворк веб-програм для Node.js, спроектований для створення веб-програм і API. Являється стандартним каркасом для Node.js.

Причиною створення Express стала відсутність підтримки багатьох задач веб-розробки в чистому Node.js. У випадку якщо розробник хоче добавити певну обробку для окремих HTTP запитів, наприклад: GET, POST, DELETE, або необхідно опрацьовувати запити з різних URL, потрібно було самому писати код виконання цих дій. Із-за цієї причини використання Express в Node.js є обов'язковим для створення хорошого веб-серверу.

Даний фреймворк є найбільш популярною веб-інфраструктурою Node та є базисом для декількох інших веб-фреймворків Node. Він забезпечує виконання таких дій:

- Можливість написання обробників запитів з різними HTTP-методами в різних URL-адресах
- Інтеграція з механізмом рендеринга "view" для генерації відповідей шляхом вставки даних у вже готові шаблони
- Задання загальних параметрів веб-сервера таких як порт, що використовується для підключення або розміщення шаблонів, що використовуються для створення відповідей
- Добавляє додаткову обробку запитів спеціальним "проміжним ПО" в любій точці конвеєра обробки запитів

Хоча сам Express досить мінімалістичний, розробники створили додаткові пакети проміжного програмного забезпечення для задоволення будь-який потреб при веб-розробці. Існують бібліотеки для роботи з файлами cookie, логінами користувачів, параметрами URL і т.д.

Таким чином використання Express є обов'язковим при створення веб-серверу на основі Node.js, оскільки він надає уже готові механізми взаємодії з

URL-запитами, без необхідності створювати і відлагоджувати ці механізми вручну.

Flexbox – це новий макетний метод який був добавлений в CSS 3. Попередня версія CSS містила 4 методи макетування, а саме:

- Блок схему для макетування документів
- Лінійну схему для моделювання текстів
- Табличну схему
- Схему заданого позиціонування

Проте ці методи мали кілька недоліків, наприклад з їх допомогою важко, а то й неможливо досягти таких простих дій над макетом:

- Вертикальне вирівнювання блока всередині “батьківського” блока
- Оформлення блоків таким чином, щоб вони розділили між собою всю доступну ширину і висоту або створення однакових відступів між блоками основуючись на загальному вільному місці на екрані
- Зробити всі колонки в макеті однакової висоти незалежну від їх вмісту

При розробці веб-сервісу найбільш корисними функціями Flexbox-у були вертикальне позиціонування блоків, адже до появи даної технології просто не існували чітких методів робити з блоками в вертикальній площині, та можливість автоматичного встановлення відступів між блоками на основі вільного простору, що дозволяє швидко і легко створювати адаптивні HTML/CSS макети. В залежності від висоти і ширини монітору за допомогою якого користувач переглядає веб-сервіс, Flexbox буде автоматично вираховувати відступи між блоками макету.

Таким чином використання даної технології значно спрощує створення адаптивний веб-сторінок, що є дуже важливим при розробці веб-сервісу.

2.3 Вибір системи збірки програми

Для збірки програми було застосовано пакет модулів Webpack у взаємодії з системою перетворення коду Babel. Розглянемо особливості та причини їх використання більш детально.

Webpack – це пакет модулів написаний на JavaScript з відкрити кодом. Основна задача Webpack це обробка JS файлів, проте дана система може працювати і із іншими ресурсами таким як HTML,CSSу випадку підключення відповідних модулів.

Webpack дозволяє скомпілювати велику кількість JavaScript файлів в єдиний JS файл. Крім цього при використані даної системи можливе виконання таких операцій:

- Збірка в воєдино ресурсів розробленого веб-продукту
- Відстеження змін і повторне виконання задач
- Виконання перетворення нових версій JavaScript до більш стандартної версії ES5 за допомогою Babel, що дозволяє використовувати новий функціонал JavaScript не хвилюючись про підтримку даних функцій в браузерах користувачів
- Може виконувати транспіляцію CoffeeScript на JavaScript
- Може конвертувати зображення, що використовуються в програмі
- Дозволяє використовувати require() для CSS файлів
- Може запускати web-dev-server, з можливістю автоматичної перезавантаження браузера при внесенні змін в програму
- Може працювати з Hot Module Replacement
- Може розділяти вихідний файл на кілька файлів, щоб уникнути довгої загрузки сторінки із-за завантаження всього коду відразу

Загалом для використання Webpack достатньо вказати лише точку входу в програму і Webpack автоматично проаналізує присутні файли і об’єднає їх в набір вихідних файлів(Рис. 2.12), крім того використання даної системи не обмежується Front-end частиною веб-сайту, Webpack можна успішно використовувати і при серверній розробці на Node.js

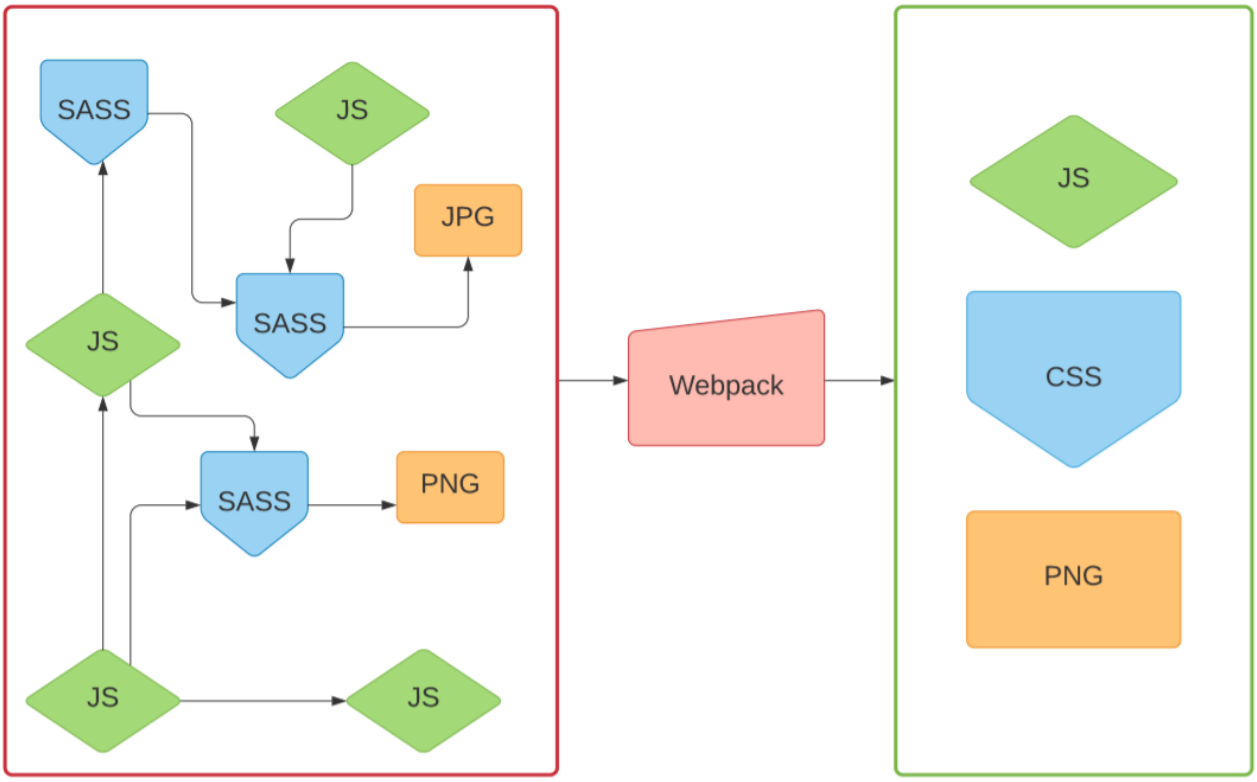

Рисунок 2.12 – Схематичне зображення роботи Webpack

Babel – це транскомпілятор JavaScript коду, який в основному використовується для перетворення ECMAScript 2015+ (ES6+) у більш старі версії JavaScript(в основному ES5). Це гарантує запуск веб-програми навіть на браузерах в які ще не було добавлено підтримки можливостей новий версій JavaScript. Наприклад стрілкові функції які були добавлені в ES6 Babel перетворе в звичайні. Також використовуючі Babel можна сміливо використовувати нестандартний синтакис на кшталт JSX і бути впевненим в коректності сприймання такого коду браузерами.

Загалом використання Webpack і Babel є чудовим рішенням для створення оптимізованих, швидкодіючих веб-програм в який розробник може задіяти всі нововведння які надають новітні стандарти JavaScript.

Висновок до розділу 2

В даному розділі було розглянуто стек технологій, що використовувася при розробці веб-сервісу.

Основними засобами розробки стали:

- HTML
- Java Script
- CSS
- Node.js

Було обгрунтовано причини вибору саме цих мов програмування для написання програми.

Також було описано причини застосування та пояснено проблеми які вирішують використані бібліотеки, фреймоврки та збірники програм, а саме:

- React
- Redux
- JQuery
- Express
- Webpack
- Babel

3.РОЗРОБА ВЕБ-СЕРВІСУ ДЛЯ НАВЧАННЯ КАСИРІВ РОБОТІ З КАСОВИМИ АПАРАТАМИ

3.1 Інтерфейс і алгоритм роботи авторизації та реєстрації на веб-сервісі

При першому заході на веб-сервіс користувача очікує вікно входу (Рис. 3.1). При авторизації, введені користувачем дані відправляються на сервер, де вони звіряються з базою зареєстрований користувачів. У випадку збігу формується об’єкт з інформацією про користувача і відправляється на клієнтську частину веб-сервісу разом з дозволом на вхід. У випадку відсутності даного користувача в базі, сервер відправляє повідомлення про помилку авторизації. Реакцією клієнтської частини веб-сервісу на це повідомлення, буде вивід попередження про неправильно введений пароль або логін.

Рисунок 3.1 – Вікна авторизації

У випадку якшо користувач не має активного акаунту на веб-сервісі, він може нажати кнопку реєстрація, після чого відкриється реєстраційна форма (Рис. 3.2). Для реєстрації користувач має заповнити поля і натиснути кнопку Реєстрація. Після цього клієнська частина програми проводе ряд перевірок. В разі знайдення помилок проблемні поля буде підсвічено червоним кольором і програма не буде проводити відправку форми на сервер. Поля вводу паролів звіряються відразу як вони були заповненні і втратили фокус. Це робиться для

того, щоб користувач відразу був оповіщений про допущену опечатку. В разі необхідності він може нажати на значок ока, що зніме шифровку з символів паролю.

Загалом на клієнській частині веб-сервісу проводяться такі перевірки:

- Перевірка на заповненість всіх поля вводу
- Перевірка на валідність адреси електронної пошти
- Перевірка на ідентичність полів Пароль і Підтвердіть Пароль

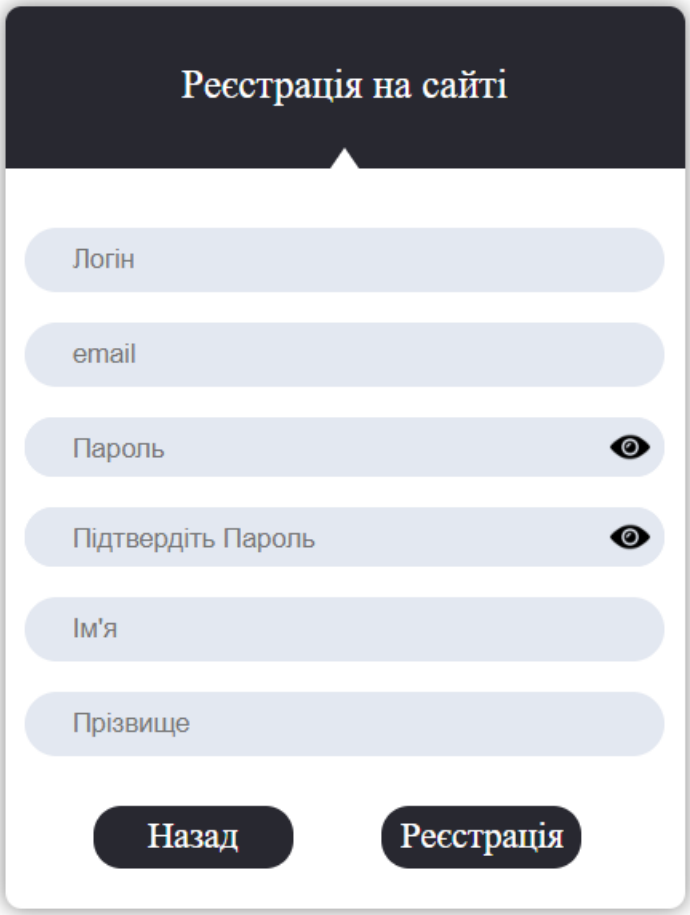

Рисунок 3.2 - Реєстрація на сайті

Після проходження клієнської провірки дані з форми відправляються на сервер, де буде проведено другий етап перевірки. Логін і email користувача має бути унікальними, саме тому перед внесенням нового користувача до бази веб-сервісу введені користувачем Логін і email перевіряються на співпадіння сереж уже існуючих користувачів.

У випдку знаходження співпадіння процес реєстрації користувача зупиняється і сервер відправляє повідомлення про помилку реєстрації а також причину цієї помилки. Дією Front-End частини веб-сервісу на це повідомлення являється підсвічення проблемного поля червноним кольором.

У випадку успішного проходження перевірок, інформація про нового користувача буде внесена до бази веб-сервісу, а як відповідь буде вілправлено півідомлення про успішну реєстрацію. У відповідь на це клієнтьсяка частина сервісу виведе повідомлення в якому повідоме користувача про успішну реєстрацію.

Після проходження реєстрації користувач може пройти авторизацію як було описано вище. Слід також відзначити, що при упішній авторизації веб-сервіс зберігає інформацію про користувача в локальному сховищі браузера.

Як результат під час відкриття сторінки веб-сервіса, до відображення його вмісту проходе звертання до локального сховища. У випадку, якщо воно не є порожнім інформація з нього копіюється в об’єкт користувача і поле авторизації не буде з’являтися. Тобто користувач може лише раз авторизуватися і при всіх подальших відвідинах веб-сервісу він буде автоматично попадати до його вмісту, уникаючи необхідності повторної реавторизації при кожному перезапускові програми.

3.2 Інтерфейс і алгоритм роботи особистого кабінету веб-сервісу

Після успішної авторизації користувач попадає до основної частини веб-сервісу(Рис 3.3). В лівій частині знаходиться вертикальне основне меню веб-сервісу. У його верхній частині знаходяться пункти “Особисти кабінет”

(в якому зараз і знаходиться користувач), “Література” і “Тест”. В низу меню знаходиться Логін користувача у спеціальному віконці, а також кнопка “Вийти”, при клікі на яку відбувається розлогінювання користувача та видалення його даних з локального сховища браузера. В цьому випадку при

наступному зоході на сторінку веб-сервісу буде необхідно заново авторизуватися.

Весь інший простір є робочим і в ньому відображається контент веб-сервісу. Початково при заході до сервісу користувач попадає в особистий кабінет, де в спеціальній формі відображуютьяс його особиста інформація, а також є можливість її зміни, якщо при реєстрації була допущена помилка яку користвач не замітив.

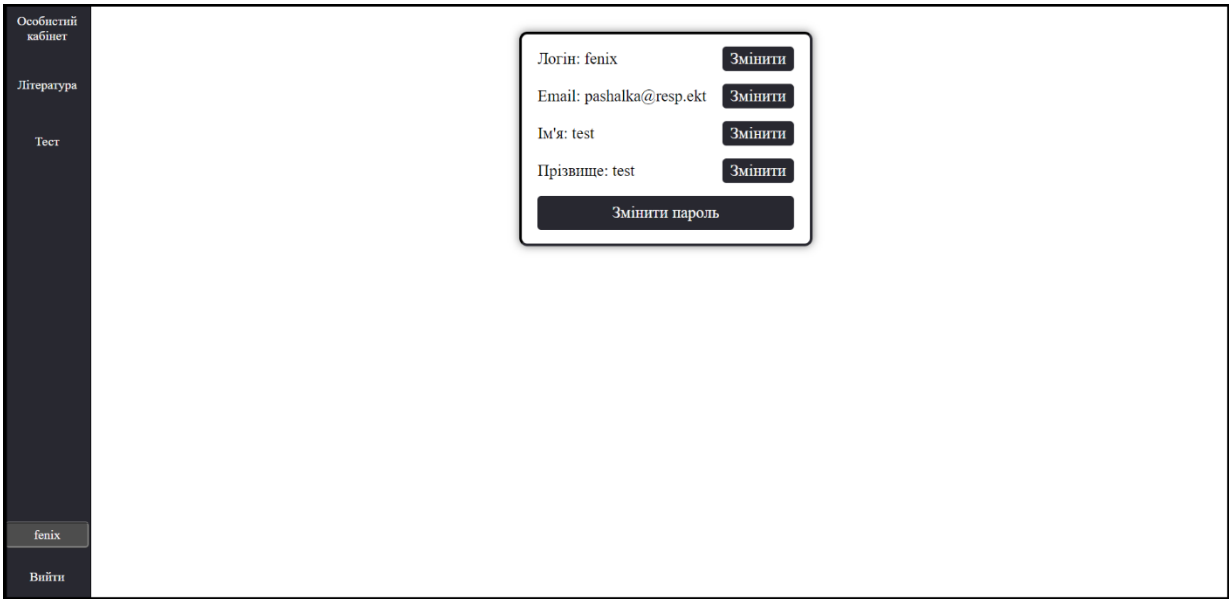

Рисунок 3.3 – Особистий кабінет до якого користувач попадає після авторизації

При клікові на кнопку змінити відкриється спеціальна форма(Рис. 3.4)

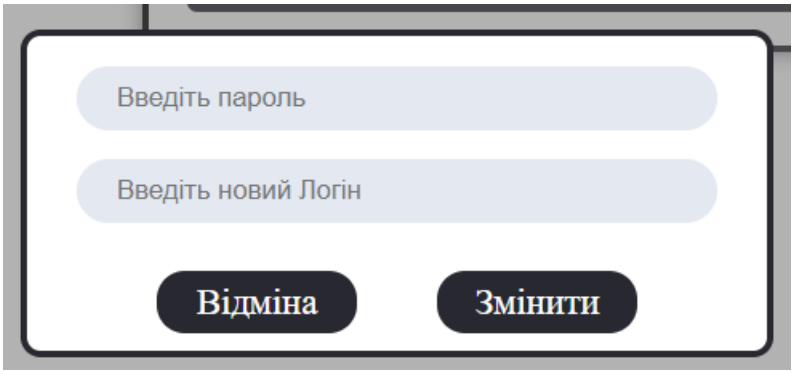

Рисунок 3.4 – Форма редагування профілю

В залежності від вибору поля для зміни, буде змінюватися і напис в нижньому полі. Для демонстрації на рисунку було натиснуто на кнопку зміни логіна. При появі форми користувач має заповнити форму, а саме ввести

пароль від свого профілю у верхнє поле і новий Логін, email, пароль і т. д. у нижнє поле.

Після цього він має натиснути кнопку Змінити, що запусте процес перевірки введеної інформації. В першу чергу програма перевірить поля на не порожність. Якщо ця перевірка буде пройдена, то відбувається перевірка введеного пароля з інформацією яка знаходиться в об’єкті користувача. У випадку помилки в проблемні поля будуть підсвічені червоним.

Якщо клієнтські перевірки були пройдені то дані відправляються на сервер, де вони проходять другий етап перевірок. Такі поля як Логін і email повинні бути унікальними, а тому при зміні саме цих пунктів профілі сервер перевіряє їх на унікальність.

У випадку проходження перевірки, сервер редагує профіль користувача і відсилає Front-End частині веб-сервісу нову інформацію про користувача разом з повідомленням про успішну зміну даних. Реакцією на це буде перезапис об’єкту користувача на клієнті, а також оновлення інформації в локальному сховищі браузера.

У випадку знайдення помилки при серверній перевірці клієнту буде відправлено повідомлення про цю помилку. В результаті чого на екрані з’явиться повідомлення про те, що такий Логін чи email вже використовується іншим користувачем

В випадку якщо користувач хоче відмінити змінити зміну даних він повинен натиснути на кнопку “Відміна”. Потрібно зазначити, що при відкритті форми редагування профілю, блокуються всі дії із сторінкою (Рис. 3.5).

Рисунок 3.5 – Блокування взаємодії із сторінкою при відкритті форми зміни профілю

Для візуального відображення даного обмеження весь простір крім форми зміни профілю покривається напівпрозорим сірим блоком. Керування над сторінкою буде відновлено або після успішної зміни інформації в профілі, або після відміни даної дії.

3.3 Інтерфейс навчальної частини веб-сервісу

Якщо користувач хоче ознайомитись з інформацією по використання касового апарати йому слід натиснути кнопку “Література”(Рис. 3.6)

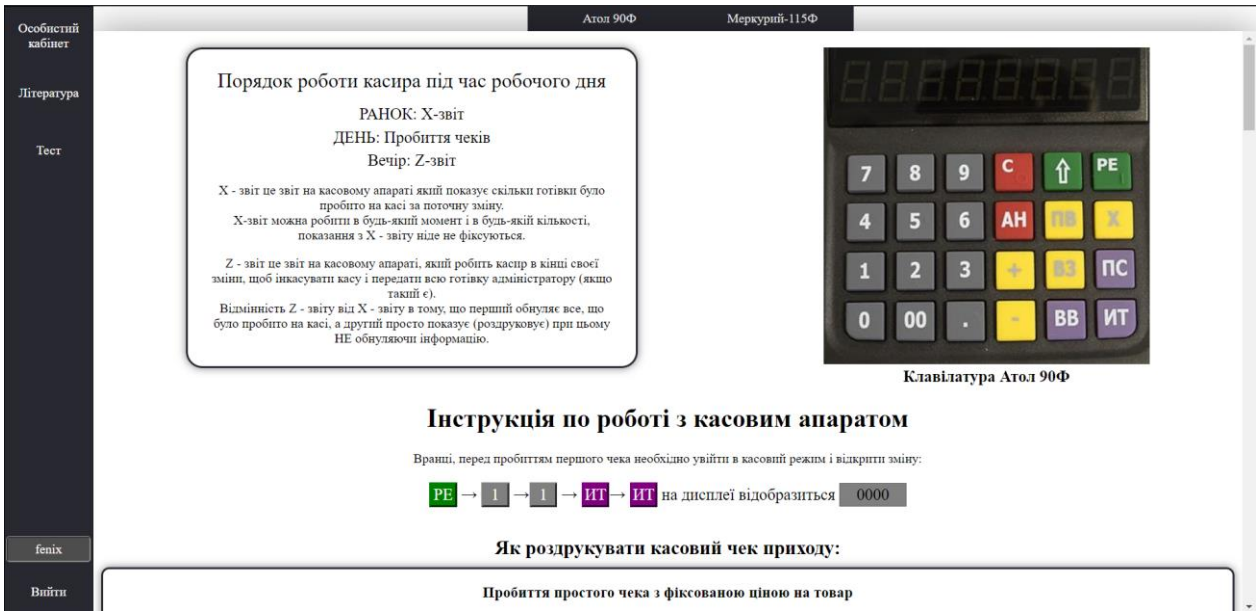

Рисунок 3.6 – Вкладка “Література” розробленого веб-сервісу

У вкладкі “Література”, як і у вкладкі “Тест”, з'вляється допоміжнє меню яке знаходиться зверху робочої зони. За його допомогою корстувач може вибирати тип касового апарати, роботі з яким йому необхідно попрактикуватися.

Основне робоче поле ділиться на три зони:

- Верхній лівий блок, у ньому описано повсякденний порядок роботи з касовим апаратом
- Верхній правий блок, у ньому знаходиться фото клавілатури реального касового апарату вибраної моделі, це зроблено для того, щоб користувач краще орієнтувався в розміщенні кнопок
- Нижній блок, у ньому описано всі основні команди які необхідно знати оператору цього касового апарату. Всі команди розбиті на категорії і офромлені в цвітовій палітрі реальної клавілатури касового апарату(Рис. 3.7).

Рисунок 3.7 – Приклад однієї з категорій у вкладкі “Література”

Таком слід зазначити, що весь інформаційний зміст даної вкладки розміщенно в спеціально блоці з прокруткою. Тобто при скролі інформації користувач не гортає саму сторінку, в результаті чого він завжди має швидки доступ до основного та допоміжнього меню. Крім того на сторінці реалізовна адаптивна HTML/CSS верстка(Рис. 3.8) яка підтримує екрани від 1024 пікселів до 1980.

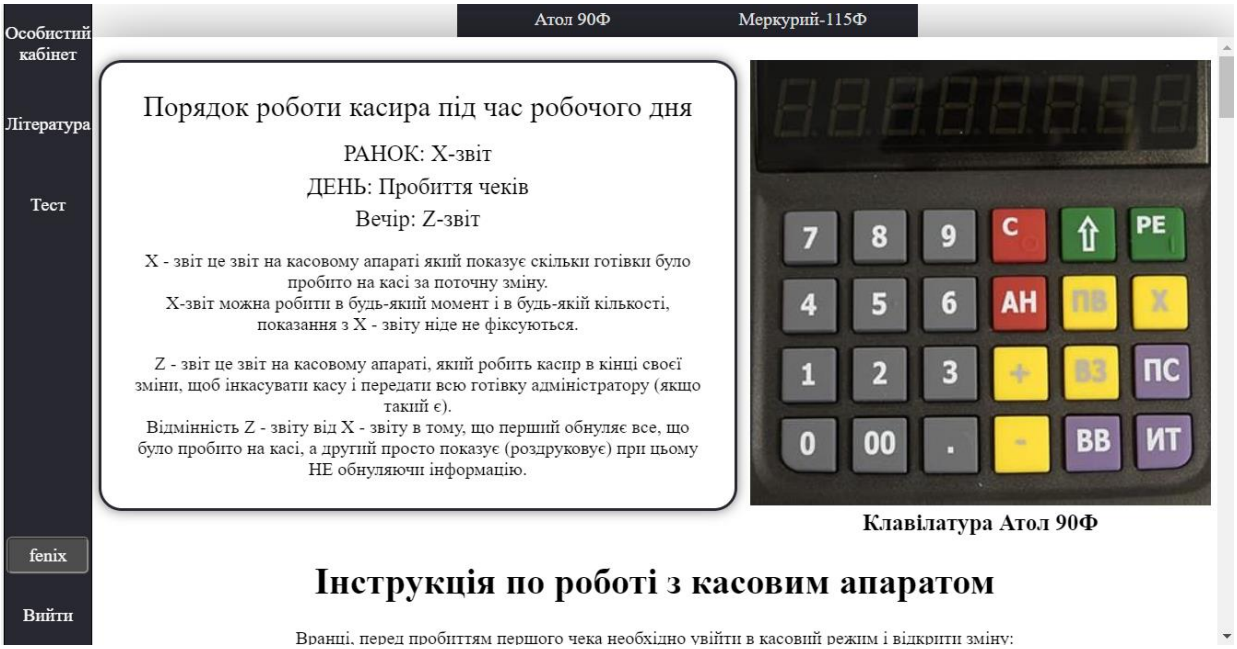

Рисунок 3.8 – Вигляд веб-сторінки при ширині екрана в 1024 пікселі

3.4 Інтерфейс і алгоритм роботи тестів розміщених на веб-сервісі

Перейдемо до головної частини Веб-сервісу – вкладки “Тест”(Рис. 3.9)

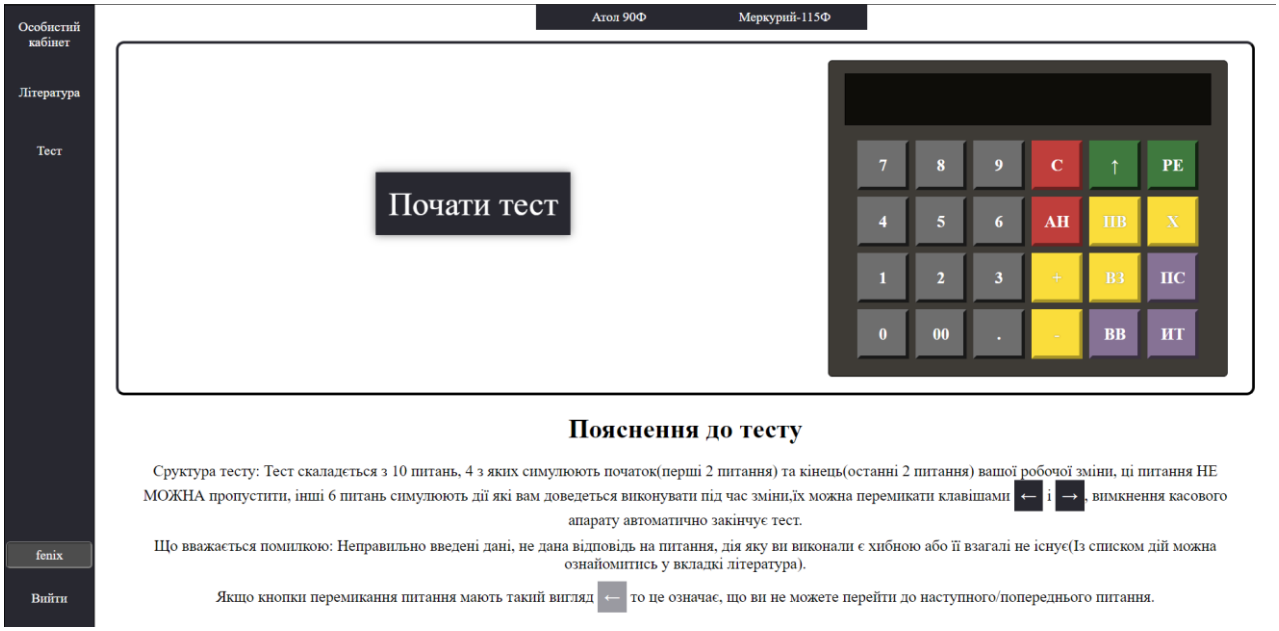

Рисунок 3.9 – Зовнішній вигляд вкладки “Тест”

Робоча зона вкладки “Тест” складається з допоміжнього меню, яке слугує для вибору касового апарату з яким буде проводитися тестування, поля самого тесту і поянювальної записки.

Розглянемо більш детально поле тесту (Рис. 3.10)

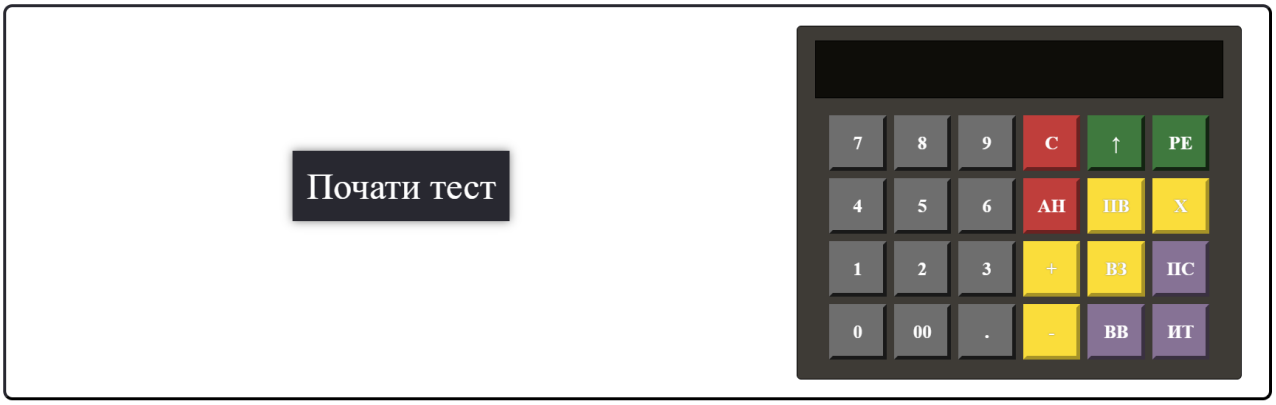

Рисунок 3.10 – Поле тесту

Воно містить точну копію клавілатури касового апарату, яка відтворює всі функції які прописані у вкладці література, а також зона на якій буде відображуватись питання тесту та успішність їх проходження.

До початку тесту ніяких дій з клавілатурою касового апарату проводити не можна. Щоб почати тест слід натиснути кнопку “Почати тест”, результатом на це стане створення тестових завдань і відображення їх користувачеві (Рис. 3.11)

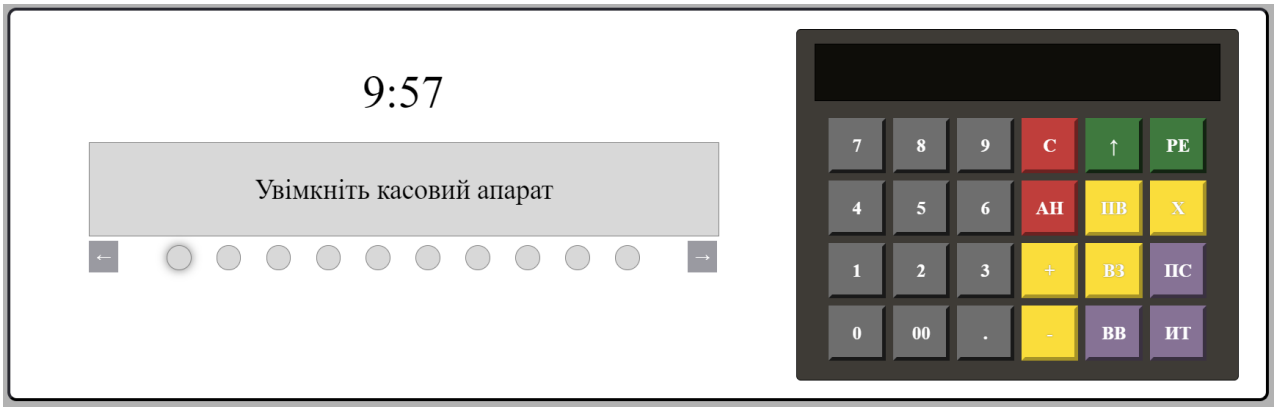

Рисунок 3.11 – Тест було запущено

Розберемо більш детально інтерфес тесту (Рис. 3.12). В центрі знаходиться основне поле тесут в якому вказане питання на яке потрібно дати відповідь. Вище нього розімщенно таймер, який дає користувачеві оцінити скільки ще часу залишилося на прходження тесту. Під осовним блоком розміщені кнопки зміни питання, які можуть блокуватись, процес даної функції буде описано нище, і графічне зобрраження процесу виконнання тесту в виді кружків.

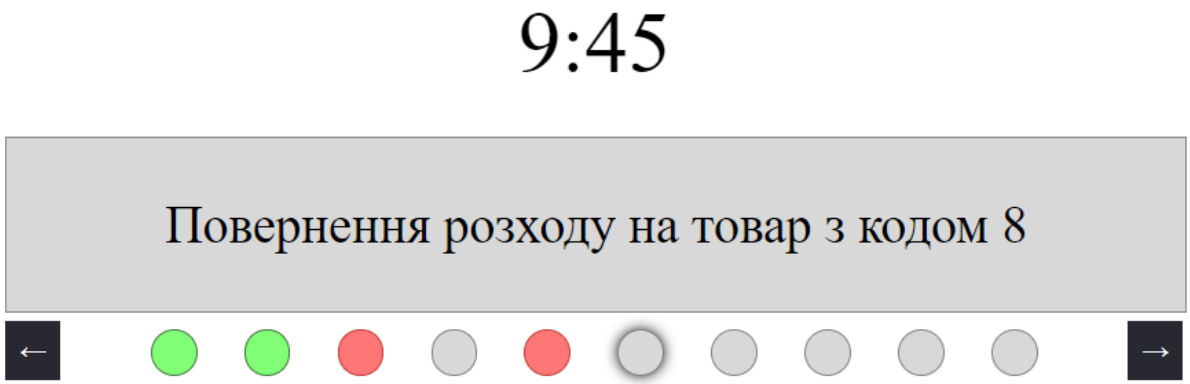

Рисунок 3.12 – Інтерфейс тесту в процесі проходження

В разі якщо на питання було дано правильну відповід відповідний йому кружок буде зафарбовано в зелений колір, при хибній відповіді в червоний, а тести на які ще не було дано відповіді будуть відображатися білим кольором. Також на цій панелі користувач може побачити на якому саме тесті він знаходиться

Загалом після запуска тесту, відбудуться такі дії:

- Відбудеться розблокування взаємодії користувача з макетом касового апарату
- Буде запущено таймер із зворотнім відліком, при закінченні якого тест буде автоматично завершено, при цьому всі питання на які не було дано відповіді вважатимуться неправильними
- Відбудеться запуск генератору питань, який на основі шаблонів згенерує набір питань для тесту

Слід також зазначити, що система тестувань має два типи питань.

Сталі питання повторюються при кожному новому тестуванні, це як правило два питання на початку тесту і два в кінці. Користувач не може їх пропустити, адже вони симулюють початок і кінець його зміни під час роботи з касовим апаратом. При реальній роботі користувач просто не зможе почати свою зміну, якщо він не знатиме як увімкнути пристрій, або увійти у режим касира. Тому при відображенні цих питань перемикання (←, →) блокуються, а саме питання не може отримати статус неправильно вирішеного. Програма буде очікувати або правильного виконання такої дії, або кінця часу тестування.

Другий тип питань це динамічні питання. Вони генеруються під час запуску тесту за допомогою спеціальних шаблонів. В результаті чого в тест попадають не тільки різні дії які повинен виконати користувач, а й числа які він повинен вводити (суму яку дав покупець, код товару, тощо) будуть відрізнятися. Дані тести можуть отримувати статус як правильно так і неправильно виконаних, а самі питання користувач може перемикати розблокованими кнопками (←, →).

При взаємодії користувача з макетом касового апарату, програма перевіряє правильність виконання команди яка вказана в питанні. В разі вводу неіснуючої або хибної команди, чи некоректний числових даних, питання отримає статус проваленого. Програма автоматично перейде до наступного тесту, це зроблено для того, щоб користувачеві не довелося самому, після відповіді на питання, перемикати його, оскільки при великій кількості проходження тестів, це може дратувати, що в свою чергу призведе до поганого користувацького досвіду. При повернені до питання на яке було дано хибну відповідь кнопками(←, →) сам блок з питанням матиме червоний колір (Рис. 3.13).

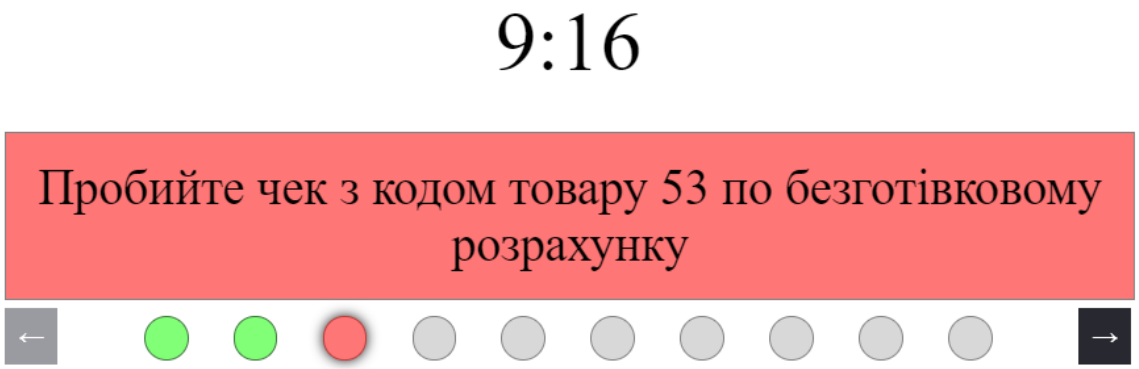

Рисунок 3.13 – Питання на яке було дано хибну відповідь

В випадку правильної відповіді на питання тест буде окращено в зелений колір (Рис. 3.14). Слід також згадати про те, що при перемиканні тесту, налаштування касового апарату будть скидуватись на початкові. Це зробелно для того, щоб якщо користувач забува відповідь на питання, він міг просто перемкнути тест і не очищати поле вводу касового апарата вручну.

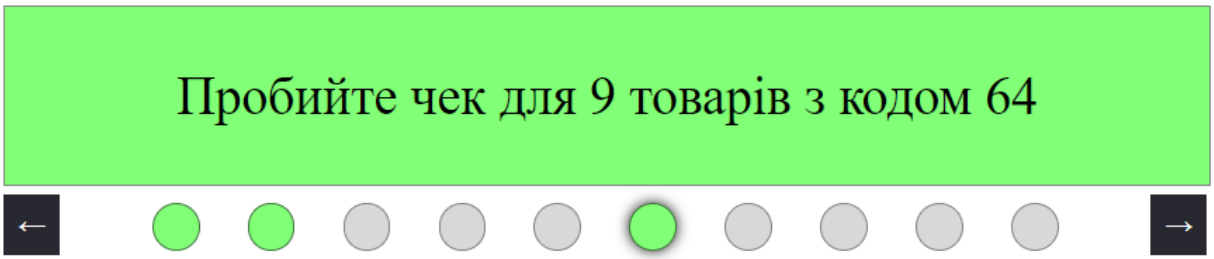

Рисунок 3.14 – Питання на яке було дано правильну відповідь відповідь

Також слід вказати, що при перемикання на питання з уже даними відповідями, як правильними так і хибним, взаємодія з клавіатурою касового апарата блокується.

Макет касового апарату також копіює всі надписи і виводи які відображує справжній касовий апарат (Рис. 3.15), а також має змогу вмикатись і вимикатись тими ж способам, як це роблять реальні аналоги.

Рисунок 3.15 – Відображення переходу в режим повернення витрат аналогічно реальному аналогу

Після дачі відповідей на всі питання або закінчення часу, відведеного на виконання тесту, на екран буде виведена загальний результат і кнопка повтореного проходження тесту(Рис. 3.16). У випадку якщо кількість правильних відповідей станове більше або рівняється 80% тест вважається пройденим і результат буде відображено зеленим кольором. Інакше,

користувачу рекомендується ще раз перечитати літературну частину веб-серісу і спробувати пройти тест ще раз. В разі не проходження тесту результат буде відображено червоним кольором.

Рисунок 3.16 – Закінчений тест з поганим результатом

Після закінчення тесту та виводу результатів на екран, відбувається скидання всіх змінних і показників, які використовуються для перевірки правильності відповідей на питання, до початкових параметрів.

Якщо користувач хоче спробувати свої сили ще раз, він може натиснути кнопку “Нова спроба”. При цьому буде перезапущено формувач питань і буде складена нова добірка тестів для проходження. Це гарантує, що при кожному новому тестуванні список питань буде різним.

Не можна також не згадати про необхідність блокування всієї взаємодії з іншими частинами веб-серівсу, при проходженні тесту. Це необхідно, щоб користувач не зміг під час проходження опитування зайти у вкладку “Література” і піддивитись відповідь. Для цього при старті тесту весь простір крім поля блоку покривається сірим, напівпрозорим блоком (Рис. 3.17). Ще одна роли даного блока це візуального виділення і фокусування уваги користувача саме на полі з тестом.

Крім того веб-сервіс коректно відображає тест і інформації в ньому на моніторах шириною від 1024 пікселів(Рис 3.18). При цьом слід звернути увагу на блок пояснювальної інформації під полем тесту. При зменшенні висоти екрану, у разі якшо вся інформація не буде вміщуватись в один екран, до даного блоку автоматично буде додана функцію прокрутки вмісту, щоб користувачеві не доводилось прогортати поле з тестом.

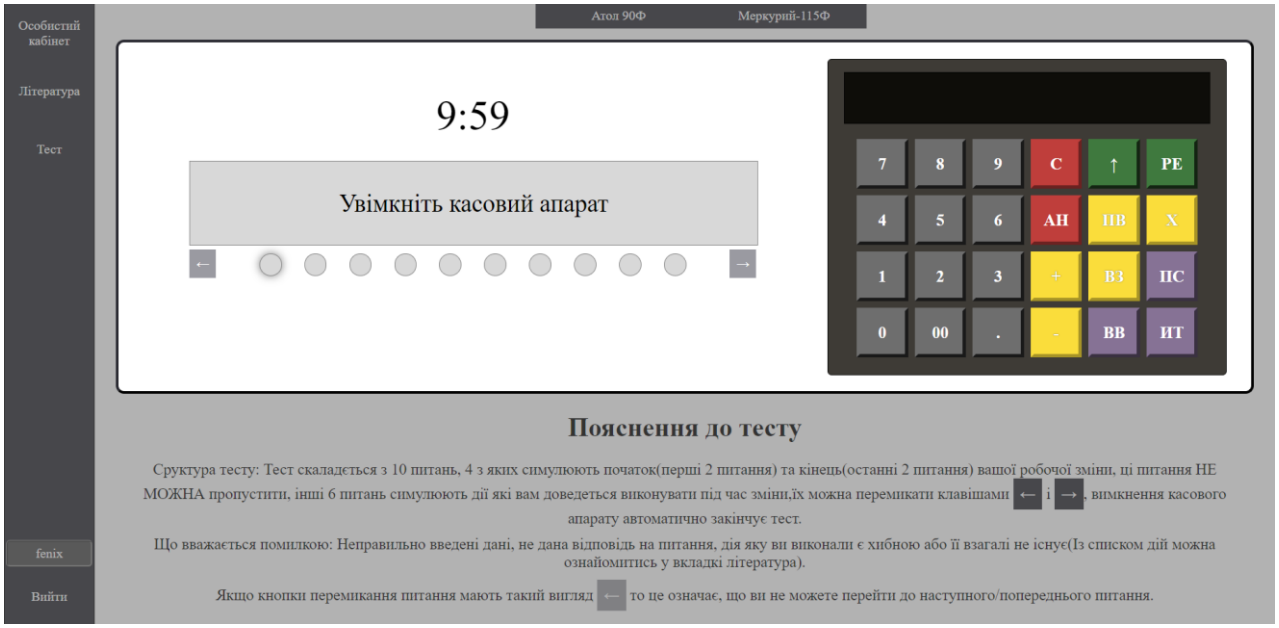

Рисунок 3.17 – Блокування взаємодії з веб-сервісом при проходженні тесту

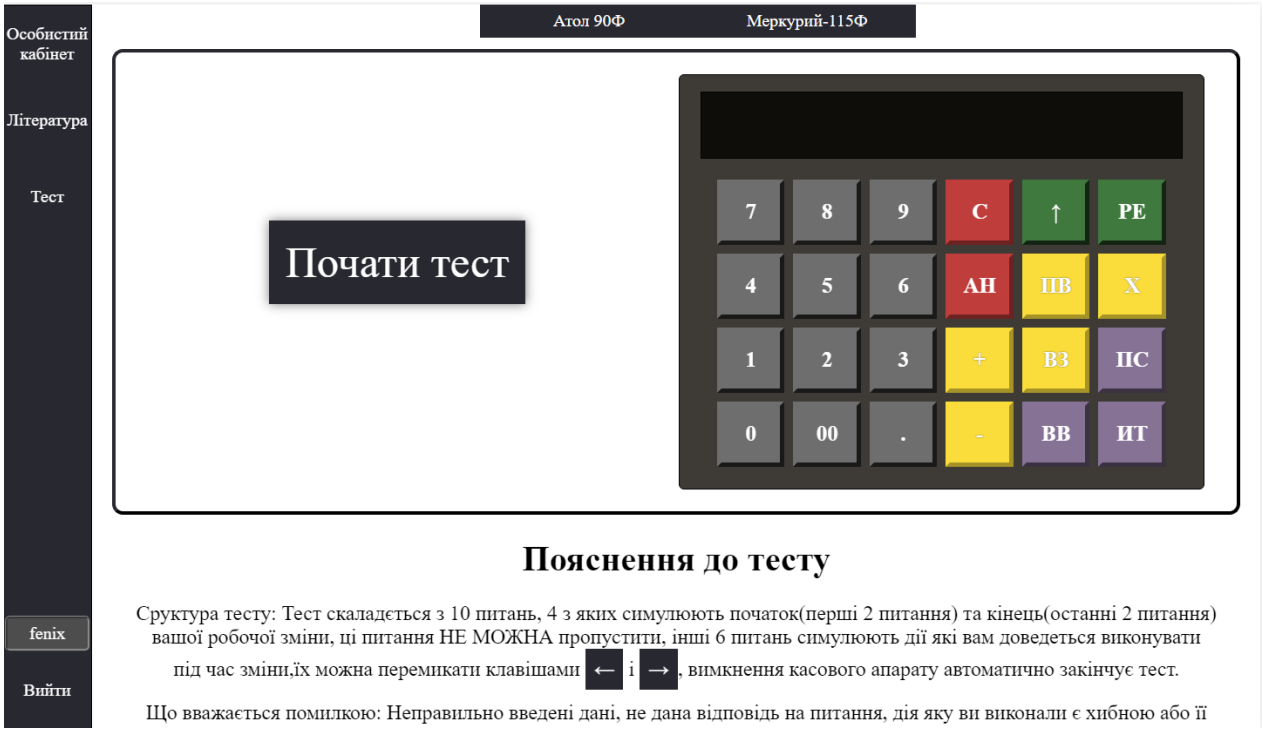

Рисунок 3.18 – Відображення тесту при ширині екрану в 1024 пікселя

Висновок до розділу 3

В даному розділі було показано і описано роботу розробленого веб-сервісу.

Було продемонстровано функціонування всіх частин веб-програми і доведено, що дана програма виконує всі поставлені перед нею задачі і повністю відповідає критеріям, які було сформовано на початку розробки.

ВИСНОВОК

Темою даного дипломного проєкту було обрано створення веб-сервісу для навчання касирів роботі з касовим апаратом.

Було доведено, що питання створення такого веб-сервісу буде стояти досить гостро в найближчі роки, із-за збільшенні долі електронних та безготівкових платежів у світі. На основі аналізу існуючих рішень, було сформулбовано основні вимоги для створюваної програми.

Також було описано і обгрунтовано причини вибору тієї чи іншої технології, для розробки представленого веб-сервісу. Було пояснено їх переваги та недоліки, а також проблеми які може вирішити розробник, використовуючи такий стек бібліотек, фреймворків та технологій.

На основі набутого досвіду, було розроблено простий та зручний у використанні веб-сервіс, який повністю задоволюняє поставленим перед початком розробки вимогам.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Офіційна документація React: URL: https://ru.reactjs.org/docs/getting-started.html(дата звертання 13.05.2020)
2. Офіційна документація Redux: URL: https://redux.js.org/ (дата звертання 14.05.2020)
3. Офіційна докуменація Webpack: URL:https://webpack.js.org/concepts/ (дата звертання 16.05.2020)
4. Офіційна документація JQuery: URL: https://api.jquery.com/ (дата звертання 13.05.2020)
5. Стаття “Всё, что нужно знать, чтобы войти в React в 2018 году”: URL: https://medium.com/@stasonmars/%D0%B2%D1%81%D0%B5-%D1%87%D1%82%D0%BE-%D0%BD%D1%83%D0%B6%D0%BD%D0%BE-%D0%B7%D0%BD%D0%B0%D1%82%D1%8C-%D1%87%D1%82%D0%BE%D0%B1%D1%8B-%D0%B2%D0%BE%D0%B9%D1%82%D0%B8-%D0%B2-react-%D0%B2-2018-%D0%B3%D0%BE%D0%B4%D1%83-bdbf3a776d21 (дата звертання 13.05.2020)
6. Стаття “Виравнивание элемементов во Flex контейнере”:URL: https://developer.mozilla.org/ru/docs/Web/CSS/CSS_Flexible_Box_Layout/%D0%92%D1%8B%D1%80%D0%B0%D0%B2%D0%BD%D0%B8%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D1%8D%D0%BB%D0%B5%D0%BC%D0%B5%D0%BD%D1%82%D0%BE%D0%B2_%D0%B2_Flex_%D0%BA%D0%BE%D0%BD%D1%82%D0%B5%D0%B9%D0%BD%D0%B5%D1%80%D0%B5 (дата звертання 16.05.2020)
7. Офіційна документація Babel: URL: https://babeljs.io/docs/en/ (дата звертання 15.05.2020)
8. Стаття “Пособие по webpack”: URL: https://habr.com/ru/post/309306/ (дата звертання 16.05.2020)

9. Стаття “Веб-фремворк Express”: URL: https://developer.mozilla.org/ru/docs/Learn/Server-side/Express_Nodejs (дата звертання 15.05.2020)
10. Стаття “React.js, Angular5 и Vue.js – какой фреймворк выбрать в 2018 году?”: URL: https://medium.com/nuances-of-programming/reactjs-angular5-%D0%B8-vue-js-%D0%BA%D0%B0%D0%BA%D0%BE%D0%B9-%D1%84%D1%80%D0%B5%D0%B9%D0%BC%D0%B2%D0%BE%D1%80%D0%BA-%D0%B2%D1%8B%D0%B1%D1%80%D0%B0%D1%82%D1%8C-%D0%B2-2018-%D0%B3%D0%BE%D0%B4%D1%83-101702e51043 (дата звертання 17.05.2020)
11. Стаття “Node.js и JavaScript для серверной разработки”: URL: https://habr.com/ru/company/ruvds/blog/345164/ (дата звертання 16.05.2020)

Додаток 1

Копії графічних матеріалів

Київ – 2020 року

Додаток 2

Лістинг програми

Київ – 2020 року

Додаток 3

Слайди презентації

Київ – 2020 року