Приватний заклад вищої освіти
Одеський технологічний університет «ШАГ»
Кафедра інформаційних технологій та фундаментальної підготовки

Випускна кваліфікаційна робота бакалавра

«СИСТЕМА ЕЛЕКТРОННОЇ КОМЕРЦІЇ ВЕБ-САЙТУ ТЕХНІКИ З СИСТЕМОЮ УПРАВЛІННЯ ДЛЯ АДМІНІСТРАТОРІВ»

на здобуття бакалаврського рівня вищої освіти
зі спеціальності «122 Комп'ютерні науки»

Виконавці проєкту

№	П.І.Б.	Група
1	Волков Ю. І.	КН-П-202
2	Чермалих О. О.	КН-П-202
3	Бондаренко Д.О	КН-П-202

Автор звіту

_____ Волков Юрій Ігорович

Одеса – 2024

АНОТАЦІЯ

УДК 004.9

В-67

Волков Ю. І. Платформа електронної комерції, система управління для адміністраторів: Автореферат випускної кваліфікаційної роботи на здобуття бакалаврського рівня вищої освіти зі спеціальності «122 Комп’ютерні науки». - Одеса: ОТУШ, 2024 - 26 с.

Дипломний проект присвячений розробці та впровадженню платформи електронної комерції для продажу електронної техніки, а також системи керування магазином для власника бізнесу. У проекті реалізовано серверну частину, що забезпечує зв'язок між інтернет-магазином, адміністративним додатком та базою даних для передачі даних.

Основні можливості сайту включають реєстрацію та авторизацію користувачів, відображення лише дозволених адміністратором даних про товар, зручний пошук за асортиментом товарів, відгуки та оформлення замовлень.

Для адміністратора платформи створено окрему кросплатформну програму, яка дозволяє легко додавати або видаляти товари, відстежувати активність, оформляти замовлення, переглядати та видаляти відгуки, а також виконувати багато інших функцій.

Серверна частина не тільки передає дані між сайтом та програмою, але й генерує токени авторизації (наприклад, при успішній верифікації користувача), щоб безпечно передавати особисті дані, такі як електронна пошта, номери телефонів, паролі та інше. Сервер також може надсилати повідомлення на електронну пошту (наприклад, для підтвердження адреси електронної пошти користувача) та генерувати чотиризначні коди для введення на сайті.

ЗМІСТ

ВСТУП

У світі інформаційних технологій електронна комерція стала невід'ємною частиною життя багатьох людей. Онлайн-торгівля проникає у різні сфери бізнесу, полегшуючи процес купівлі та продажу товарів та послуг. З розвитком інтернету та мобільних технологій стало можливим створення ефективних платформ для реалізації товарів та автоматизації процесів управління підприємством.

Метою даного проекту є створення комплексного рішення, яке спростить процеси управління рухом та обліку комерційних елементів (товарів, послуг тощо).

Проект було названо названо TechX, проект ділиться на 3 підпроекти:

1. TechX Store - Сучасний магазин електроніки дозволить споживачам легко знаходити товари, що їх цікавлять, забезпечуючи широкий вибір продукції, заводити профіль і дивитися історію замовлень, додавати товари в обрані, ділиться коментарями та багато іншого.
2. TechX Compass - кросплатформна програма яка дозволить власнику сайту зручно керувати магазином. Compass має сучасний та зручний дизайн для ефективної роботи з магазином. Завдяки інтуїтивно зрозумілому інтерфейсу та широкому набору функцій адміністратори зможуть легко керувати асортиментом товарів, контролювати запаси, обробляти замовлення, аналізувати статистику продажів і стежити за зворотним зв'язком від клієнтів. Додаток також підтримує сповіщення у реальному часі, що дозволяє оперативно реагувати на зміни та забезпечувати високий рівень обслуговування.
3. TechX Server - зберігає у собі запити для роботи з TechX Store, TechX Compass і базою даних. Це незамінна складова системи, яка забезпечує безперебійну роботу TechX Store та TechX Compass, а також забезпечує надійне зберігання та обробку даних. Цей серверний компонент відповідає за оптимізацію обміну даними між магазином електроніки, програмою для адміністраторів та базою даних, що забезпечує швидкодіючу та ефективну взаємодію.

РОЗДІЛ 1

TECHX COMPASS

TechX Compass – кросплатформна програма яка дозволить власнику сайту зручно керувати магазином. Програма була реалізована на фреймворку Electron.

Electron - це фреймворк для розробки десктопних програм з використанням HTML, CSS і JavaScript. У двійковий код Electron вже вбудовані Chromium і Node.js, і це дозволяє вам підтримувати лише JavaScript код і створювати кросплатформні програми, які працюватимуть як на Windows, так і на macOS і Linux без необхідності мати власний досвід розробки. Розглянемо основні принципи, за якими електрон відображає верстку у вікні програми:

1. Браузерне ядро: В основі електронних програм лежить браузерне ядро Chromium, яке відповідає за відображення HTML, CSS та виконання JavaScript. Це дозволяє електрону використовувати потужність та гнучкість веб-технологій для створення інтерфейсу програми.
2. Відображення HTML/CSS: Весь інтерфейс користувача в електронній програмі створюється з використанням HTML і стилізується за допомогою CSS, як у веб-розробці. Це дозволяє розробникам створювати інтерфейс за допомогою знайомих інструментів та підходів.
3. JavaScript для логіки: Вся логіка програми, включаючи взаємодію з користувачем та обробку подій, реалізується за допомогою JavaScript. Це робить можливим створення динамічних та інтерактивних інтерфейсів користувача, а також взаємодія з операційною системою та іншими ресурсами комп'ютера.
4. Потужні можливості: Електрон надає потужні можливості для взаємодії з операційною системою, такі як доступ до файлової системи, робота з мережею, використання системних повідомлень тощо. Це уможливлює створення повнофункціональних десктопних програм, незважаючи на те, що вони базуються на веб- технологіях.
5. Інтеграція з ОС: Електронні програми можуть інтегруватися з операційною системою, надаючи доступ до функцій, таких як панель завдань, меню програм, оповіщення, файлові операції та інші. Це дозволяє створювати додатки, які виглядають та поводяться як нативні для кожної операційної системи.

Переваги Electron

1. Кросплатформність

   Electron дозволяє створювати програми, які працюють на Windows, macOS та Linux із однієї кодової бази. WPF та WinForms: В основному орієнтовані на Windows, хоча є деякі можливості для перенесення на інші платформи з використанням Mono або .NET Core, але це потребує додаткових зусиль.

2. Багата екосистема та велика спільнота

   Electron підтримується спільнотою та має велику бібліотеку готових модулів та плагінів, WPF та WinForms мають підтримку від Microsoft та велику спільноту, але сфокусовані в основному на корпоративних та Windows-додатках.

   Інші фреймворки. Qt і JavaFX мають активні спільноти,

   менша кількість готових бібліотек та плагінів.

3. Продуктивність

   Electron в цілому, важче ресурсів через

   Chromium, але сучасне обладнання дозволяє запускати

   програми без значних проблем. WPF Більш продуктивний для Windows-програм, використовує DirectX для рендерингу.

   WinForms Менш продуктивний у порівнянні з WPF, але підходить для простих програм.

   Інші фреймворки: Qt та JavaFX пропонують хорошу продуктивність, але потребують вивчення специфічних технологій.

4. Гнучкість та розширюваність

   Висока гнучкість завдяки використанню Node.js для серверної логіки та npm для керування залежностями.

   WinForms: Найменша гнучкість у порівнянні з WPF, але простіше у використанні.

Таким чином, електронні програми використовують браузерне ядро Chromium для відображення верстки, а JavaScript для логіки програми, що дозволяє їм створювати крос-платформні десктопні програми за допомогою веб-технологій.

РОЗДІЛ 2

КОНТЕКСТНИЙ МІСТ

Контекстний міст (context bridge) - це механізм, який дозволяє безпечно обмінюватися даними між головним процесом (main process) та процесом відображення (renderer process) у Electron. У Electron головний процес управляє головним вікном програми, а процес відображення відповідає за відображення інтерфейсу користувача в цьому вікні. Через відмінності у безпеці між цими процесами використання контекстного мосту (context bridge) допомагає запобігти небезпечним операціям, таким як доступ до API Node.js з процесу відображення.

Контекстний міст дозволяє безпечно експортувати лише певні функції або об'єкти з головного процесу в контекст (глобальний простір імен) процесу відображення, що забезпечує вищий рівень безпеки програми.

У проекті контекстний міст експортує об'єкт "electron" до глобального простору імен процесу відображення. Цей об'єкт надає три методи обміну повідомленнями між процесами:

1. send(channel, data): Відправляє повідомлення з даними з процесу відображення головний процес через заданий канал (channel).
2. receive(channel, func): Очікує повідомлення за вказаним каналом у головному процесі та викликає функцію (func) для обробки отриманих даних у процесі відображення.
3. invoke(channel, data): Посилає синхронний запит (invoke) із процесу відображення в головний процес через вказаний канал і повертає результат назад у процес відображення.

```
contextBridge.exposeInMainWorld('electron',
{
        send: (channel, data) => { ipcRenderer.send(channel, data); },
        receive: (channel, func) => { ipcRenderer.on(channel, (event, ...args) => func(...args)); },
        invoke: (channel, data) => ipcRenderer.invoke(channel, data)
});
```

Ці методи допомагають встановлювати безпечну взаємодію між головним і процесом відображення в Electron, забезпечуючи захист від можливих уразливостей безпеки.

РОЗДІЛ 3

ВІКНО АВТОРИЗАЦІЇ

Першим етапом є перевірка пароля. Функція Confuse(str) використовується для хешування введеного пароля методом SHA-256. Потім функція Validator() отримує введений пароль з елемента з id "id-admin-password" і перевіряє, чи він не порожній. Якщо пароль порожній, з'являється повідомлення про помилку. Якщо пароль не порожній, викликається функція Confuse() для його хешування. Після отримання хешу відбувається порівняння із попередньо збереженим значенням. Якщо хеші збігаються, надсилається запит до електрона для відображення адміністративної панелі. Інакше з'являється повідомлення про помилку.

Друга частина відповідає за логіку приховування та відображення пароля. Функція Change() змінює тип елемента введення пароля з "password" на "text" і навпаки при кожному натисканні на елементі з id "toggle-password-visibility". При цьому змінюється іконка ока для відображення або приховування пароля.

```
function Confuse(str)  // <-- This function takes a string and returns the SHA-256
{                      // hash of that string for the password verification function.
            return crypto.subtle.digest('SHA-256', new TextEncoder().encode(str)).then(buffer =>
            {
                        return Array.prototype.map.call(new Uint8Array(buffer), x => ('00' +  x.toString(16)).slice(-2)).join('');
            });
}

function Validator()  // <-- Password check function.
{
            const _admin_password = document.getElementById("id-admin-password").value;
            if(_admin_password.trim() === "")
            {
                        ShowErrorMessage("Empty password string");
                        return;
            }
            Confuse(_admin_password).then(hashed_pass =>
            {
                        if(hashed_pass ===  'c1c224b03cd9bc7b6a86d77f5dace40191766c485cd55dc48      caf9ac873335d6f')
        electron.send('ShowAdminPanel');
      else
        ShowErrorMessage("Incorrect password");
  });}
```

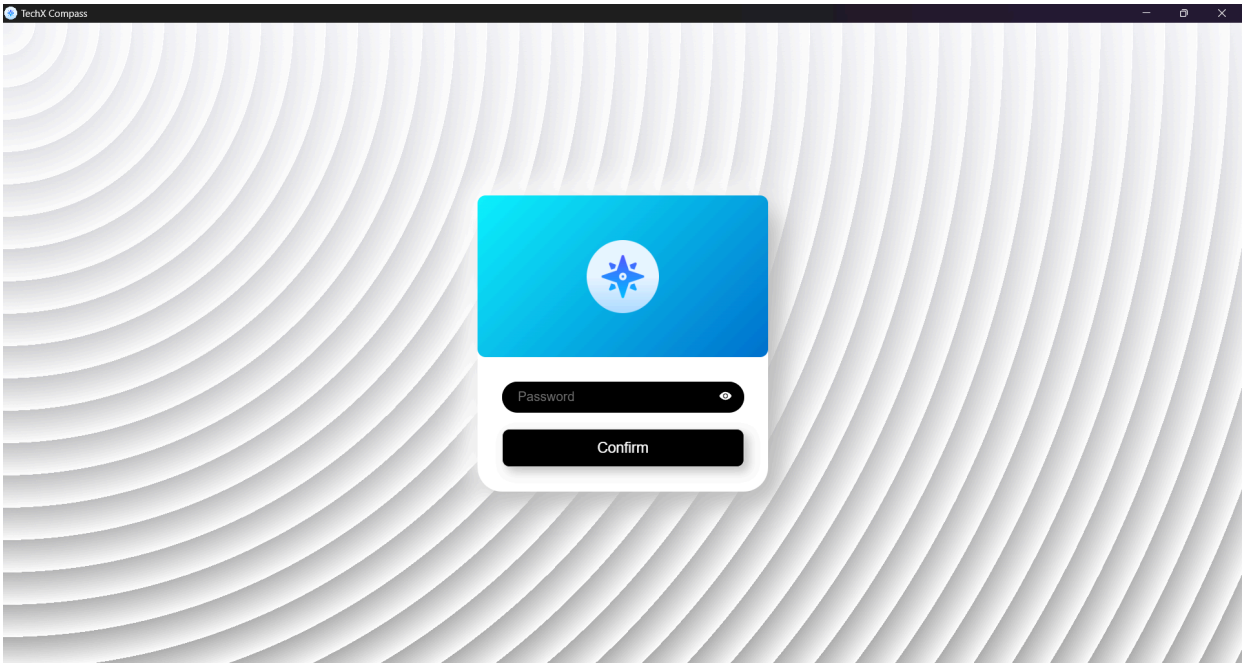

Рисунок 1 - Сторінка профілю

Функція ShowErrorMessage(mess) відображає повідомлення про помилку з надісланим текстом. При цьому відображається елемент з id "id-error-message", який містить повідомлення про помилку.

Після відображення повідомлення про помилку через 4 секунди вікно з повідомленням ховається. Функція BreakErrorMessage() використовується для примусового приховування вікна з повідомленням про помилку під час натискання елемента з id "id-close-error-message".

Рисунок 2 - Відображення помилки

РОЗДІЛ 4

ВІКНО МЕНЮ АДМІНІСТРАТОРА

Навігація по меню адміністратора

Логіка вертикального меню передбачає обробку кліків за пунктами меню та відображення відповідного контенту. Коли користувач натискає на певний пункт меню, відповідний контент стає видимим, а інші приховуються. Також у коді є функція повернення до попереднього елемента меню при виході з панелі адміністратора. За замовчуванням активується перший елемент меню, який відображає вміст, пов'язаний із продуктами. При натисканні на будь-який елемент списку меню всі елементи спочатку деактивуються, потім активується вибраний елемент, і відповідний контент відображається, приховуючи решту.

```
list_item.forEach((item) => item.classList.remove("active"));
list_item[0].classList.add("active");
products_content.style.display = "flex";
list_item.forEach((item) =>
{
  item.addEventListener("click", () =>
  {
    list_item.forEach((item) => item.classList.remove("active"));
    item.classList.add("active");
  });
});
list_item[0].addEventListener("click", () =>
{
  DisableContent();
  HideAllAddProductMenuItems();
  RecordWhereUserIsNow(0);
  products_content.style.display = "flex";
});
export function ReturnToPreviousTab()
{
 exit_content.style.display = "none";
 if (user_position_index_menu === 0)
 products_content.style.display = "flex";
 else if (user_position_index_menu === 1)
 orders_content.style.display = "flex";
 else if (user_position_index_menu === 2)
 activity_content.style.display = "flex";
 else if (user_position_index_menu === 3)
 reviews_content.style.display = "flex";
 else if (user_position_index_menu === 4)
 settings_content.style.display = "flex";
 list_item[5].classList.remove("active");
 list_item[user_position_index_menu].classList.add("active");
}
```

Запам'ятовується позиція користувача в меню за допомогою функції RecordWhereUserIsNow, що дозволяє при поверненні на попередню вкладку відображати правильний контент.

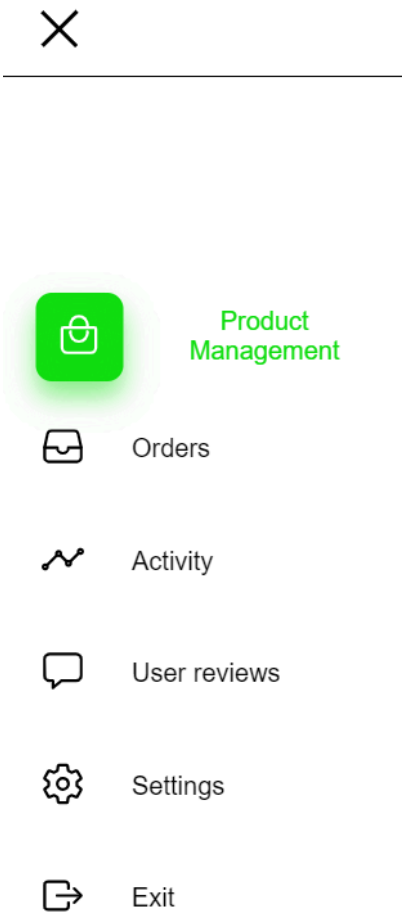

Рисунок 3 - Меню можливостей

У разі виходу з адміністративної панелі відображається відповідний контент, а при поверненні користувача на попередню вкладку відображається останній активний розділ, забезпечуючи зручність використання.

Створення нового товару

Створення нового продукту є невід'ємною частиною для керування товарами в Інтернет-магазині. У ньому визначено різні функції та обробники подій, що забезпечують зручний інтерфейс для додавання та керування товарами. Функція ShowContentProductCreation відповідає за відображення вмісту для створення нового товару. Після виклику цієї функції користувач бачить форму для додавання нового продукту. Функції ShowContentProduct(Product) активуються при виборі певної категорії товарів користувачем. Вони відображають відповідні поля для додавання товарів до обраної категорії.

Функція BackToMainMenu призначена для повернення користувача до основного меню після завершення роботи з додаванням товарів. Вона приховує всі поля та повертає користувача до основного інтерфейсу.

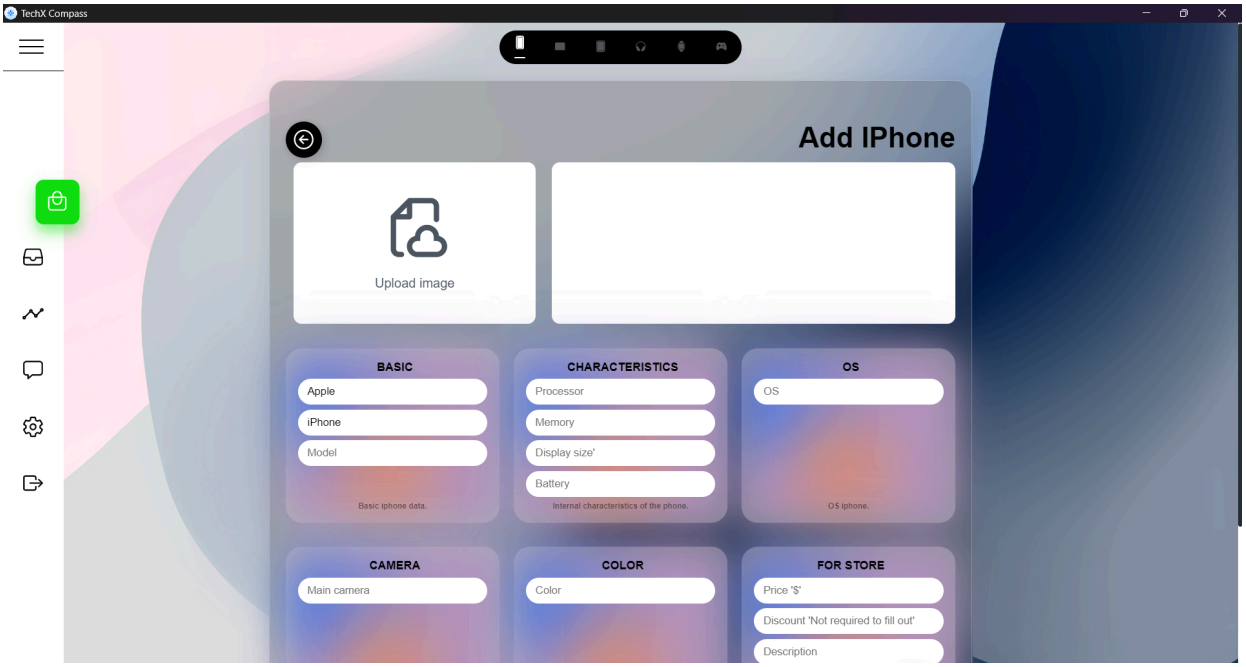

Рисунок 4 - Створення нового товару

Логіка для завантаження зображень з використанням функціоналу перетягування та скидання (drag-and-drop) для. Основні елементи включають область перетягування drop_area, елемент custum_file_upload, який змінює стиль кордону під час перетягування файлу, щоб показати користувачу, що область активна, і елемент drop_ul, що є список, де відображаються завантажені зображення.

Коли файл перетягується над областю drop_area, спрацьовує подію dragover, всередині якої зупиняється спливання події та запобігає його дефолтній поведінці. Змінюється стиль кордону елемента custum_file_upload на чорну пунктирну лінію, щоб показати користувачу, що область готова прийняти файл.
Якщо файл залишає область перетягування, спрацьовує подію dragleave, яка повертає стиль межі елемента custum_file_upload до вихідного стану. Коли файл скидається в область, спрацьовує подію drop, яка також зупиняє спливання події та запобігає її дефолтній поведінці. Потім межа елемента custum_file_upload повертається до початкового стану.

Файли, скинуті на область, перевіряються на тип: обробляються лише зображення. Для кожного файлу створюється новий елемент списку з іконкою зображення, видалення і текстом, що представляє шлях файлу. Шлях зображення додається до масиву.

```
drop_area.addEventListener('drop', (event) =>
{
 event.stopPropagation();
 event.preventDefault();
 custum_file_upload.style.border = '2px dashed #cacaca';
 const files = event.dataTransfer.files;
 for (let i = 0; i < files.length; i++)
 {
 if (!files[i].type.startsWith('image/'))
 continue;
 const child = document.createElement('li');
 const rm_icon = document.createElement('ion-icon');
 const div = document.createElement('div');
 const image_icon = document.createElement('ion-icon');
 image_icon.setAttribute('name', 'image');
 image_icon.id = 'id-drop-image';
 div.id = 'id-drop-div';
 rm_icon.setAttribute('name', 'trash');
 rm_icon.id = 'id-drop-trash'
 child.id = 'id-drop-li'
 child.textContent = files[i].path;
 _img_product_paths_iphone.push(files[i].path);
 rm_icon.onclick = () =>
 {
 div.removeChild(image_icon)
 div.removeChild(child)
 div.removeChild(rm_icon)
 drop_ul.removeChild(div)
 _img_product_paths_iphone.splice(i, 1);
 }
 div.appendChild(image_icon)
 div.appendChild(child)
 div.appendChild(rm_icon)
 drop_ul.appendChild(div)
 }
});
```

Іконка видалення прикріплена до обробника подій, який видаляє зображення зі списку та масиву шляхів при натисканні. Кожен елемент додається до списку drop_ul, забезпечуючи візуальне відображення завантажених зображень.

При натисканні на кнопку додавання товару дані додаються до локального сховища після перевірки введених даних на порожні поля та виконання необхідних операцій оновлення інтерфейсу користувача. Якщо функція `CheckForEmptyLines` повертає значення, що означає, що одне з полів порожнє, викликається функція `ShowErrorMessage` із зазначенням, яке поле має бути заповнене. Якщо всі поля заповнені, створюється новий об'єкт `ConsoleDataModel`, використовуючи значення з полів уведення та стан прапорця.

Потім викликається метод AddToLocalStorage цього об'єкта, який додає дані консолі в локальне сховище.

Якщо консоль успішно додана до локального сховища, всі поля введення очищаються. Після цього видаляються всі дочірні елементи з `drop_ul`, і масив `_img_product_paths_` очищається. Потім викликаються функції `FetchProducts` та `DisplayTable` для оновлення інтерфейсу та відображення оновленої таблиці даних продуктів.

Функція `CheckForEmptyLines` приймає всі елементи введення та перевіряє їх на наявність порожніх значень, ігноруючи поля з placeholder `Discount 'No required to fill out'`. Якщо знаходиться порожнє поле, повертається його місце, що дозволяє показати, яке поле необхідно заповнити.

```
const container = document.getElementById('id-content-phone-data');
 const inputs = container.querySelectorAll('.card-content-product-data-input');
 const in_carousel_checkbox = container.querySelector('#id-incarousel-ceckbox');
 let empty_fields_flag = CheckForEmptyLines(inputs);
 if (empty_fields_flag)
 ShowErrorMessage(`Field (${empty_fields_flag}) should not be empty`);
 else
 {
 const new_iphone = new IphoneDataModel(inputs, in_carousel_checkbox.checked,
_img_product_paths_iphone);
 const is_added = await new_iphone.AddToLocalStorage();
 if (is_added)
 {
 inputs.forEach(input => { input.value = ''; });
 inputs[0].value = 'Apple';
 inputs[1].value = 'iPhone';
 while (drop_ul.firstChild)
 drop_ul.removeChild(drop_ul.firstChild);
 _img_product_paths_iphone = [];
 await FetchProducts();
 DisplayTable();
 }
 }

function CheckForEmptyLines(inputs)
{
  for(let i = 0; i < inputs.length; i++)
  {
    if(inputs[i].getAttribute('placeholder') === "Discount 'Not required to fill out'")
      continue;
    if(!inputs[i].value)
      return inputs[i].getAttribute('placeholder');
  }
}
```

Управління товарами

Логіка управління товарами включає функції та обробники подій, які забезпечують зручний інтерфейс для перегляду, редагування та видалення товарів. Функція ShowContentProductManage() викликається при натисканні на кнопку керування товарами в основному меню. Вона вимикає інші розділи інтерфейсу, приховує елементи додавання товарів та відображає вміст для керування товарами. Після завантаження сторінки скрипт виконує запит до локального сховища, щоб отримати список товарів за допомогою функції FetchProducts(). Отримані дані оновлюються в масиві _products за допомогою функції UpdateProductsArray().

Функція DisplayTable() відображає таблицю з товарами на основі даних із масиву _products. Вона також обробляє пошук, фільтрацію та пагінацію товарів.

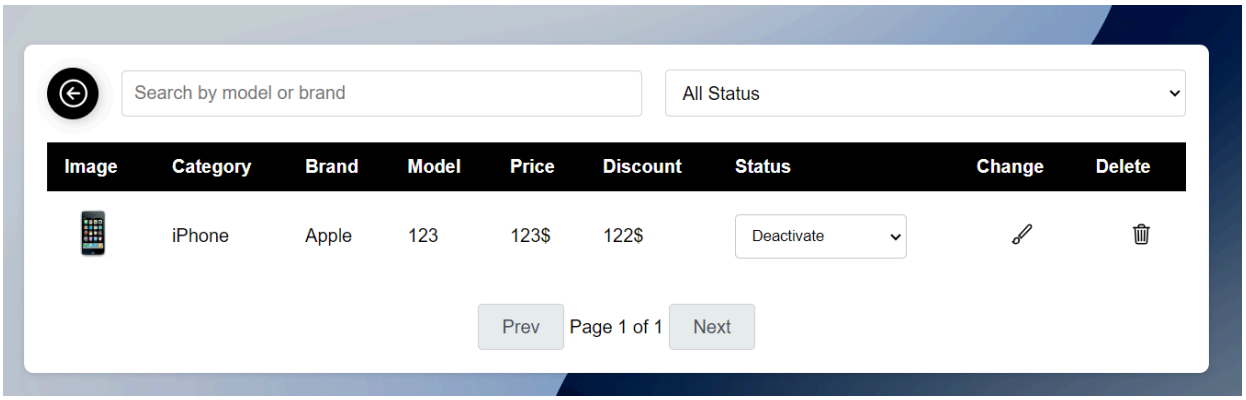

Рисунок 5 - Таблиця вироблених товарів

Клік по іконці кошика у таблиці товарів ініціює функцію OnTrashIconClick(), яка видаляє вибраний товар зі сховища даних та оновлює відображення таблиці.

При зміні статусу товару (активний/неактивний) виконується функція OnStatusChange(), яка оновлює статус товару у локальному сховищі. Функція ConfirmDeletingProductFromTable() викликається при видаленні товару та відображає модальне вікно для підтвердження видалення. Функція AddProduct() додає новий продукт на сервер за допомогою API. Вона також завантажує зображення товару на сервер та обробляє успішне чи неуспішне додавання. Функція RemoveProductFromDB() видаляє товар із сервера за допомогою API. Нарешті, кнопка "Back" в інтерфейсі дозволяє користувачеві повернутися до основного меню після завершення роботи з керуванням товарами.

```
async function AddProduct(product)
{
  let img_path_genirated_server = [];
  for (const image_path of product.images)
  {
    try
    {
      const response = await fetch(image_path);
      const file_blob = await response.blob();
      const file = new File([file_blob], image_path.split('/').pop());
      const form_data = new FormData();
      form_data.append('image', file);
      const upload_response = await fetch("https://techx-server.tech:443/AddNewProductImg",
      {
        method: 'POST',
        body: form_data,
      });
      if (!upload_response.ok)
        throw new Error('Failed to upload image');
      else
      {
        const data = await upload_response.json();
        img_path_genirated_server.push(...data.file_names);
      }
    }
    catch (error) { console.error('Error uploading image:', error); }
  }
  const new_produc_object = { product, server_img: img_path_genirated_server };
  const res_user_exists = await fetch("https://techx-server.tech:443/AddProduct",
  {
    method: "POST",
    headers: { "Content-Type": "application/json" },
    body: JSON.stringify(new_produc_object)
  });
  if (res_user_exists.ok)
    ShowSuccessMessage("Product added to the site");
  else
    ShowErrorMessage(res_user_exists.message);
}
```

Замовлення

Логіка відображення замовлень в інтерфейсі адміністратора онлайн-магазину та забезпечує функціонал пошуку, фільтрації, сортування та зміни статусу замовлень.

Змінні `number_orders` та `orders` використовуються для відстеження кількості замовлень та зберігання інформації про них відповідно.

Функція `LoadAllOrders()` виконує запит на сервер для отримання всіх замовлень. Отримані дані оновлюються у змінній `orders`. Якщо кількість замовлень змінилася, викликається функція `FilterAndSortOrders()` для фільтрації та сортування замовлень. Функція UpdateOrderTable(orders) оновлює таблицю замовлень в інтерфейсі на основі переданих даних про замовлення. Вона генерує HTML-код для кожного замовлення та додає його до таблиці. Також вона прив'язує обробники подій до списків, що випадають, статусу замовлень.

Секція "Search and Filter Functionality" містить функції для пошуку, сортування та фільтрації замовлень. Функція `FilterAndSortOrders()` оновлює таблицю замовлень згідно з введеним пошуковим запитом, вибраним порядком сортування та фільтром статусу. Функція `FilterOrders()` виконує фактичну фільтрацію та сортування замовлень.

```
function LoadAllOrders()
{
    fetch('https://techx-server.tech:443/GetOrder',
    {
        method: 'POST',
        headers: { 'Content-Type': 'application/json' }
    })
    .then(response => response.json())
    .then(data =>
    {
        const formatted_data = data.formatted_data;
        if (formatted_data.length !== number_orders)
        {
            number_orders = formatted_data.length;
            orders = formatted_data;
            FilterAndSortOrders();
        }
    })
    .catch(error => { console.error('Error:', error); });
}
```

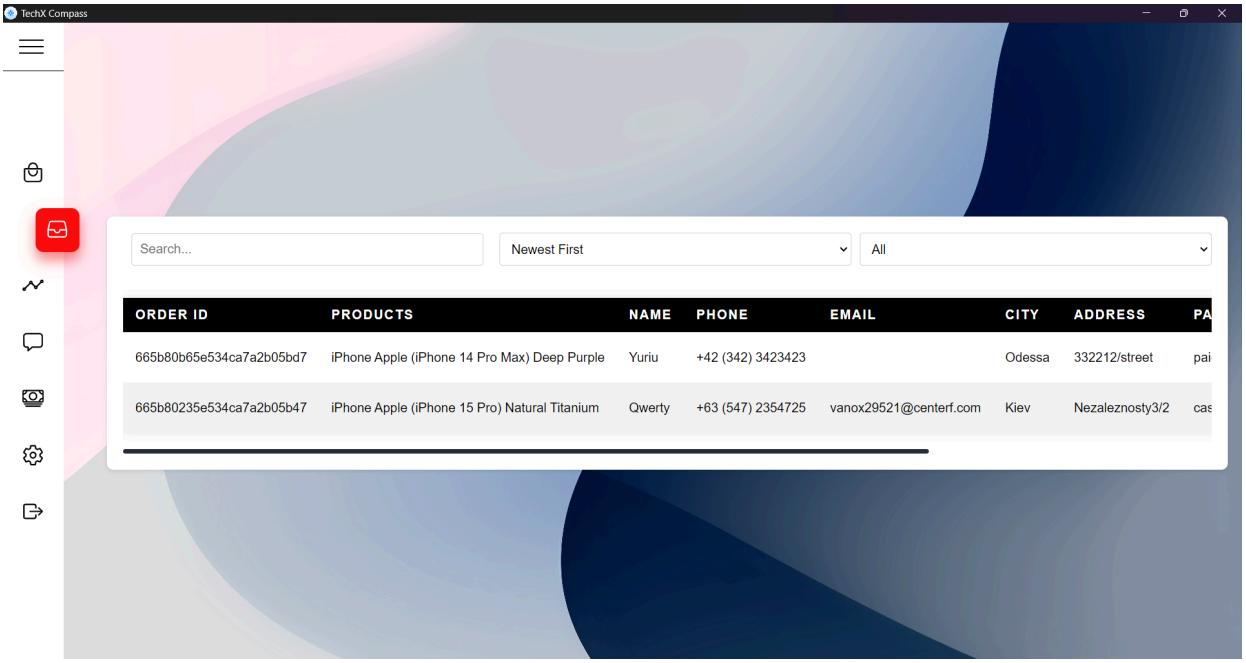

Рисунок 6 - Таблиця замовлень

Секція Not found message sector містить функції для відображення та приховування повідомлення про те, що замовлень не знайдено. Секція "Change status sector" містить функцію `ChangeStatusOrder()`, яка надсилає запит на сервер для зміни статусу замовлення.Остання частина скрипту містить обробник події `DOMContentLoaded`, який викликає функцію `LoadAllOrders()` під час завантаження сторінки та встановлює інтервал для періодичного оновлення замовлень.

Відгуки

Логіка завантаження та відображення відгуків про продукт є логікою управління відображенням відгуків про товар на веб-сторінці. При завантаженні сторінки спочатку викликається функція `LoadAllReviews()`, яка надсилає запит на сервер для отримання відгуків. Після успішної відповіді сервера відгуки відображаються на сторінці. Якщо немає відгуків, відображається повідомлення про відсутність відгуків. Функція `CollectReviewCard(review_data)` формує HTML-код для кожного відгуку на основі отриманих даних. Кожен відгук включає текст відгуку, зображення товару, назву, дату, ім'я користувача та рейтинг у вигляді зірок.

Користувач може здійснювати пошук відгуків за текстом відкликання, назвою товару, ім'ям користувача або датою.

Рисунок 7 - Таблиця замовлень

```
function LoadAllReviews()
{
 fetch('https://techx-server.tech:443/GetProductReview',
 {
 method: 'POST',
 headers: { 'Content-Type': 'application/json' }
 }).then(response =>
 {
 if (!response.ok)
 throw new Error('Network response was not ok');
 return response.json();
 }).then(review_data =>
 {
 if(review_data.length! = number_reviews)
 {
 number_reviews = review_data.length
 CollectReviewCard(review_data)
 HiddenNoReviews();
 let context_review = 0;
 review_data.forEach(new_review =>
 {
 if(!new_review.viewed_admin)
 context_review++;
 });
 window.electron.send('UpdateReviewCountForContextMenu', context_review);
 }
 else if(review_data.length == 0)
 ShowNoReviews();
 });
}
```

При введенні тексту в поле пошуку відгуків відображаються лише ті відгуки, які відповідають критеріям пошуку. Якщо не знайдено відгуків під час пошуку, відображається повідомлення про те, що результати пошуку відсутні. Якщо ж відгуків взагалі немає, відображається повідомлення про відсутність відгуків.

При наведенні адміністратором на відкликання, яке він ще не переглянув, викликається функція `AdminChecked(id)`, яка надсилає запит на сервер для позначки відкликання як переглянутого. Користувач також може видалити відгук. Під час натискання на іконку видалення відгуку відображається підтверджуюче вікно. Якщо користувач підтверджує видалення, відкликання видаляється з бази даних.

Активність

Елімент меню активності надає адміністратору сайту активність продукції а саме (Перегляди, Замовлені, У вибраних) у вигляді діаграми на веб-сторінці з використанням бібліотеки Chart.js.

Функція `LoadAllStatistics()` виконує запит на сервер за адресою `https://techx-server.tech:443/GetProductStatistics`, використовуючи метод POST і встановлюючи заголовок `Content-Type: application/json`. Вона отримує дані про статистику, такі як мітки моделей продуктів, кількість переглядів, кількість продажів і кількість додавань до обраного. Після отримання даних, вони зберігаються в глобальних змінних. Функція `UpdateChart()` відповідає за оновлення діаграми на веб-сторінці. Якщо існує діаграма ('product_chart'), вона знищується, щоб очистити попередні дані і запобігти дублювання. Потім створюється нова діаграма типу "стовпчаста", яка відображає три набори даних: перегляди, продаж та обране. Графік малюється на елементі `<canvas>` з id `id-product-chart` та налаштовується для адекватного відображення даних, включаючи налаштування кольорів та ширини ліній.

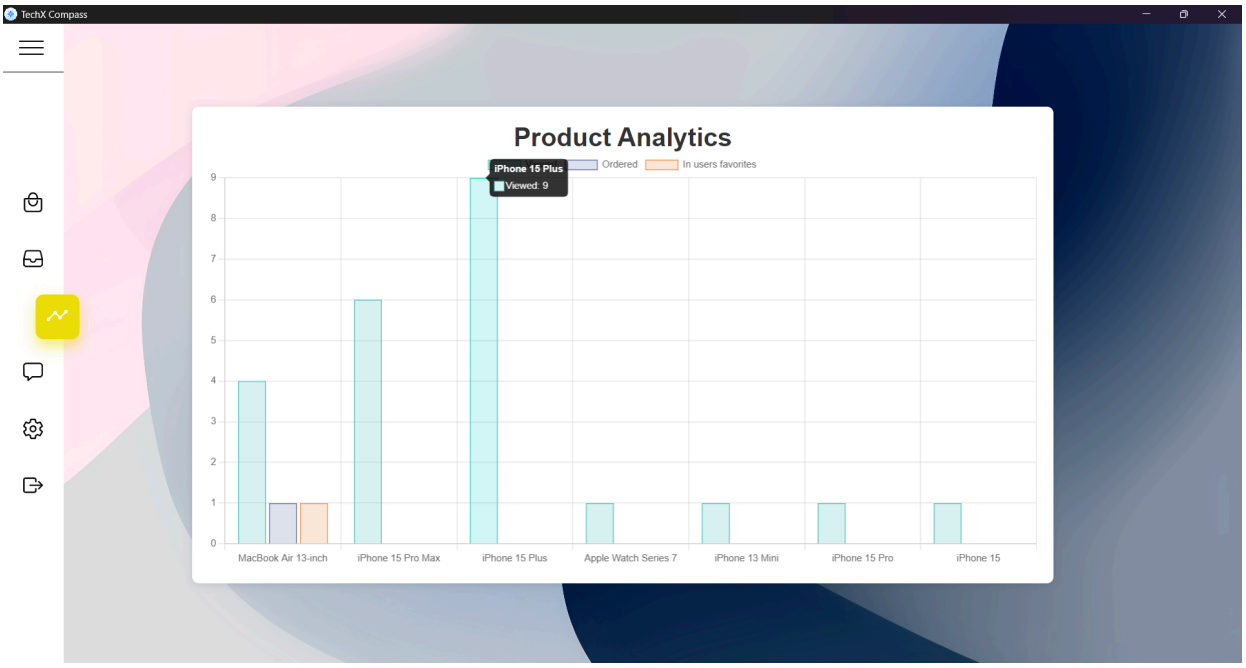

Рисунок 8 - Статистика продукції

Після завантаження сторінки, скрипт автоматично викликає `LoadAllStatistics()` для початкового завантаження даних. Для періодичного оновлення статистики встановлюється інтервал в 5 секунд, який викликає `LoadAllStatistics()`, оновлюючи дані на діаграмі без необхідності взаємодії користувача.

```
async function LoadAllStatistics()
{
  try
  {
    const response = await fetch('https://techx-server.tech:443/GetProductStatistics',
    {
        method: 'POST',
        headers: {'Content-Type': 'application/json'}
    });
    if (!response.ok)
        throw new Error('Network response was not ok ' + response.statusText);
    const data = await response.json();
    _product_statistics_labels = data.map(item => item.model);
    _product_statistics_data_view = data.map(item => item.number_views);
    _product_statistics_sales = data.map(item => item.number_sales);
    _product_statistics_favorites = data.map(item => item.number_favorites);
    UpdateChart();
  }
  catch (error) { console.error('There was a problem with the fetch operation:', error); }
}
```

Цей підхід забезпечує динамічне відображення актуальних даних статистики продуктів, що є корисним для моніторингу та аналізу продуктивності продуктів на веб-платформі.

Вікно виходу

Коли користувач натискає кнопку "Вихід", викликається функція ClickExitAccount(), яка надсилає повідомлення через Electron для відображення сторінки авторизації. Таким чином користувач виходить з адміністративної панелі і переходить на сторінку авторизації. Якщо користувач скасував вихід, викликається функція `ClickCancelAccount()`, яка повертає користувача на попередню вкладку адміністративної панелі.

Обидві функції прив'язані до відповідних кнопок на сторінці, щоб забезпечити функціональність виходу з адміністративної панелі при натисканні на відповідні елементи інтерфейсу.

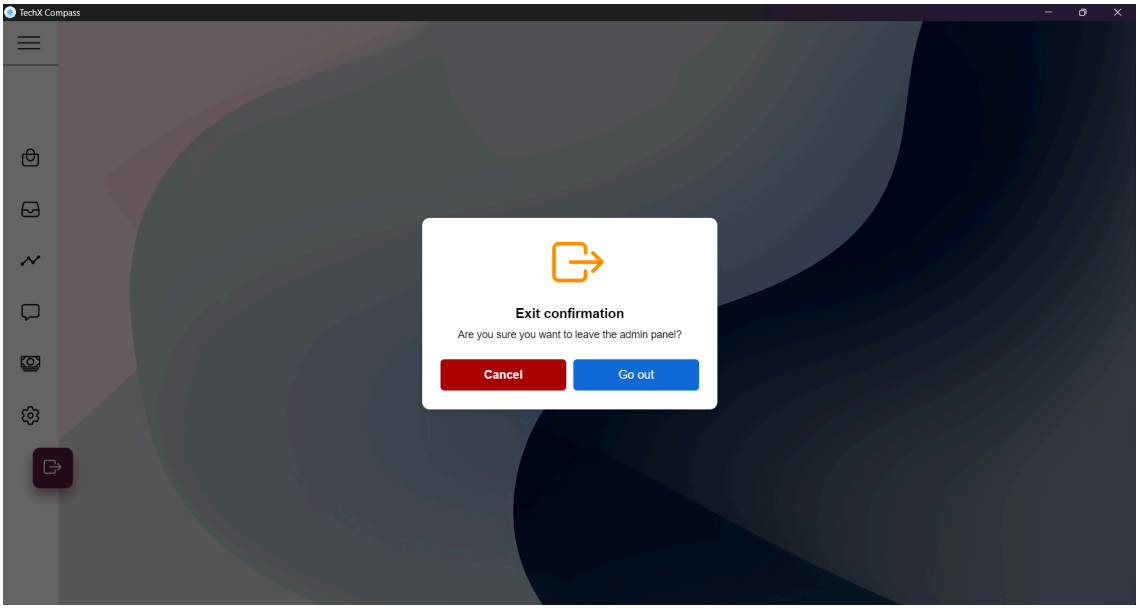

Рисунок 21 - Вихід

РОЗДІЛ 5

КОНТЕКСТНЕ МЕНЮ

Контекстне меню в Electron - це інтерактивне спливаюче меню, яке з'являється при натисканні правою кнопкою миші на іконці трею (або інших елементах інтерфейсу), надаючи користувачеві набір опцій для взаємодії з програмою. Контекстне меню створюється з використанням модуля Menu і може бути налаштоване за допомогою масиву об'єктів, кожен з яких є опцією меню з міткою (label) та функцією зворотного дзвінка (click), яка виконується при виборі даної опції.

```
function CreateTray()
{
   _tray = new Tray(path.join(__dirname, 'src', 'img', 'icon.png'));
   const context_menu = Menu.buildFromTemplate([
   {
     label: 'Open',
     click: () => { _app_window.show(); }
   },
   {
      label: order_count === 0 ? 'Order' : (${order_count}) Order,
      click: () =>
      {
         order_count = 0;
         UpdateContextMenu();
         _app_window.show();
         _app_window.webContents.send('ShowOrder');
      }
   },
   {
      label: review_count === 0 ? 'Reviews' : (${review_count}) Reviews,
      click: () =>
      {
         review_count = 0;
         UpdateContextMenu();
         _app_window.show();
         _app_window.webContents.send('ShowReview');
      }
   },
   {
      label: 'Block',
      click: () => {  _app_window.loadFile(path.join(__dirname, 'src', 'components', 'Authorization.html')); }
   },
   {
     label: 'Close',
     click: () =>
     {
       app.quit();
       _tray.destroy();
```

```
    }
  }]);
  _tray.setToolTip('TechX-compass');
  _tray.setContextMenu(context_menu);
  _tray.on('click', () => { _app_window.show(); });
  _app_window.on('close', (event) =>
  {
    if(!app.isQuitting)
    {
      event.preventDefault();
      _app_window.hide();
    }
  });
}
```

Функція CreateTray() створює та ініціалізує іконку трею (_tray) із зазначенням шляху до іконки. Потім вона створює контекстне меню (context_menu) із певними пунктами:

- Open: Якщо вибрати цей пункт, вікно програми _app_window відображається.
- Order: Відображає кількість замовлень або повідомляє, що їх немає. При виборі скидає order_count та відправляє сигнал на відображення замовлень у _app_window.
- Reviews: Аналогічно Order, але для відгуків.
- Block: Завантажує сторінку авторизації у _app_window.
- Close: Закриває програму та знищує іконку трею.

Крім того, встановлюється підказка для іконки трею (setToolTip) та контекстне меню (setContextMenu). Також визначено обробника події кліка по іконці трею, який відображає головне вікно програми.Функція UpdateContextMenu() аналогічна CreateTray(), але використовується для динамічного оновлення контекстного меню на основі поточних значень review_count та order_count. Це дозволяє оновлювати текст та функціональність пунктів меню у реальному часі.

Також в коді представлені обробники подій IPCMain (ipcMain.on), які слухають події від рендер-процесів та оновлюють review_count та order_count, після чого викликають функцію UpdateContextMenu() для оновлення контекстного меню з новими даними.

ВИСНОВКИ

У процесі роботи над дипломним проектом було розроблено платформу електронної комерції для продажу електронної техніки та систему управління магазином для власника бізнесу. До основних висновків можна віднести наступне:

1. Реалізація функціоналу платформи:
    - Було створено інтернет-магазин, що дозволяє користувачам реєструватися, авторизуватись, переглядати товари, здійснювати пошук, залишати відгуки та оформлювати замовлення.
    - Серверна частина забезпечує зв'язок між інтернет-магазином, адміністративним додатком та базою даних, а також генерує токени авторизації та надсилає повідомлення електронною поштою.
2. Система управління для адміністраторів:
    - Адміністраторам надано кросплатформова програма, створена на базі Electron, що дозволяє додавати або видаляти товари, відстежувати активність, оформляти замовлення, переглядати та видаляти відгуки.
    - Програма має сучасний та зручний дизайн, підтримує реальний годинник оповіщення та забезпечує легке управління асортиментом товарів, контроль запасів та аналіз статистики продажів.
3. Перспективи:

    Система створена за універсальною схемою та може бути адаптована до різних завдань електронної комерції.

    Перспективними напрямками досліджень є розширення функціональності платформи, зокрема:
    - Розробка інтелектуальної системи пошуку товарів, що враховує індивідуальні уподобання користувачів.
    - Впровадження можливості замовлення товарів із персоналізованими рекомендаціями з урахуванням аналізу поведінки клієнтів.
    - Додавання функціоналу бронювання товарів, що дозволить клієнтам резервувати необхідні їм товари на певний період.

ПЕРЕЛІК ВИКОРИСТАНИХ ТЕХНОЛОГІЙ

№ п.п.	Ресурс	Ступінь запозичення
1.	Next.js Розробник: Vercel; Документація: https://nextjs.org/	Використано у якості фреймворку React для платформи для розробки Web-додатку.
2.	React.js Розробник: Facebook; Документація: https://react.dev/	Використано у якості платформи для розробки Web-додатку.
3.	Electron Розробник: GitHub; Документація: https://www.electronjs.org/	Використана як створення системи для адміністраторів.
5.	Node.js Розробник: Ryan Dahl; Документація: https://nodejs.org/en	Використовується як розробка WEB-додатку, системи управління для адміністраторів та сервера
6.	JavaScript Розробник: Brendan Eich; Документація: https://developer.mozilla.org/en-US/docs/Web/JavaScript	Використовувався для написання системи для керування адміністраторів та сервера

7.	JSX Розробник: Jordan Walke; Документація: https://ru.legacy.reactjs.org/docs/introducing-jsx.html	Використано у якості язика програмування для розробки Web-додатку.
8.	MongoDB Розробник: 10gen; Документація: https://www.mongodb.com/	Використовувалося як глобальне сховище, для зберігання даних
9.	Visual Studio Code Розробник: Microsoft; Документація: https://code.visualstudio.com/	Використовувалося як редактор коду