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

Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено:

Завідувач кафедри

_____ Олександр Коваль

«___»_____2021 р.

Дипломна робота

на здобуття ступеня бакалавра

спеціальності 121 «Інженерія програмного забезпечення»

освітня програма «Інженерія програмного забезпечення розподілених систем»

на тему: «Інформаційна система автоматизації планування навчального навантаження кафедри»

Виконав:
студент IV курсу, групи ТВ-71
Осипов Владислав Ігорович _____

Керівник:
Асистент
Діденко Олексій Олександрович _____

Рецензент:

_____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2021 року

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший рівень

спеціальності 121 «Інженерія програмного забезпечення»

освітня програма «Інженерія програмного забезпечення розподілених систем»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Олександр Коваль
(підпис)
”___”_____2021р.

ЗАВДАННЯ
на дипломну роботу студенту

_____Осипову Владиславу Ігоровичу_____
(прізвище, ім’я, по батькові)

1. Тема роботи ___інформаційна система автоматизації планування навчального навантаження кафедри_____
_____

керівник роботи __асистент Діденко О.О._____
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від ”_24_” травня 2021р. № 1267-с_____

2. Строк подання студентом роботи _____

3. Вихідні дані до роботи _робочий навчальний план; навчальне навантаження кафедри_____
_____

4.Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)__Аналіз задачі автоматизації планування навчального навантаження кафедри; огляд існуючих алгоритмів та інформаційних технологій; вибір засобів розробки програмного продукту; проектування алгоритму обчислення величин; розробка програмного забезпечення та його тестування; підготовка опису програмного продукту для користувача._____

5. Перелік ілюстративного матеріалу

___Актуальність, Існуючі системи, Постановка задачі, Засоби розробки, Програмна реалізація, Тестування, Взаємодія з системою, Висновки._____
_____
_____

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання ”___”_____202___ р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Затвердження теми роботи		
2.	Вивчення та аналіз задачі		
3.	Розробка архітектури та загальної структури системи		
4.	Розробка структур окремих підсистем		
5.	Програмна реалізація системи		
6.	Оформлення пояснювальної записки		
7.	Захист програмного продукту	13.05.2021	
8.	Передзахист	27.05.2021	
9.	Захист	15.06.2021	

Студент _____ _____Осипов В.І._____
(підпис) (прізвище та ініціали,)

Керівник роботи _____ _____Діденко О.О._____
(підпис) (прізвище та ініціали,)

АНОТАЦІЯ

За обсягом робота займає 48 сторінок, містить 32 рисунки, 3 додатки та посилається на використані джерела з 8 найменувань.

Метою даною дипломної роботи є розробка системи для автоматизації процесу планування навчального навантаження кафедри.

У роботі розглянуто використання бібліотек, програмних рішень, що дозволяють розробити програмне забезпечення, що дозволить мінімізувати похибку та прискорити планування навчального навантаження кафедри. Задля зручності використання та збереження незалежності від технологій користувача реалізовано сервіс, що розгортається на стороні користувача та надає прикладний веб-інтерфейс для взаємодії користувача з системою.
Ключові слова: робочий навчальний план, навчальне навантаження кафедри, JavaScript, веб-сторінка.

ABSTRACT

The thesis consists of 48 pages, contains 32 pictures, 3 attachments, refers to 8 references.

The aim of the thesis is to implement system for automating the process of planning the workload of the cathedra. For the sake of usability while keeping technology-independency the self-hosted service, which provides web interface to connect with user, was developed.
Keywords: working curriculum, workload of cathedra, JavaScript, web page.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ..... 6
ВСТУП..... 7
1. ПОСТАНОВКА ЗАДАЧІ..... 9
2. АНАЛІЗ ПРОБЛЕМИ АВТОМАТИЗОВАНОГО ОБЧИСЛЕННЯ ПЕДАГОГІЧНОГО НАВАНТАЖЕННЯ КАФЕДРИ..... 10
2.1. АНАЛІЗ НАВЧАЛЬНОГО ПЛАНУ КАФЕДРИ..... 10
2.2. ОГЛЯД ІСНУЮЧИХ СИСТЕМ ДЛЯ УНІВЕРСИТЕТІВ..... 12
3. ЗАСОБИ РОЗРОБКИ ПРОГРАМНОГО ПРОДУКТУ..... 14
3.1. СЕРЕДОВИЩЕ РОЗРОБКИ VISUAL STUDIO CODE..... 14
3.2. МОВА РОЗМІТКИ HTML 5..... 15
3.3. CSS..... 16
3.4. МОВА ПРОГРАМУВАННЯ JAVASCRIPT..... 17
3.5. BOOTSTRAP..... 19
4. ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ..... 22
4.1. ЗЧИТУВАННЯ ДАНИХ..... 22
4.2. ПРОВЕДЕННЯ ОБЧИСЛЕНЬ.....25
4.3. СТВОРЕННЯ ПРЕДСТАВЛЕННЯ..... 36
5. РОБОТА КОРИСТУВАЧА З СИСТЕМОЮ..... 40
5.1. СИСТЕМНІ ВИМОГИ..... 41
5.2. СЦЕНАРІЙ РОБОТИ КОРИСТУВАЧА З СИСТЕМОЮ..... 41
ВИСНОВКИ..... 47
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ..... 48
ДОДАТОК А..... 49
ДОДАТОК Б..... 51
ДОДАТОК В..... 60

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

JSON – JavaScript Object Notation.

JS – JavaScript.

DOM – Document Object Model.

BOM – Browser Object Model.

HTTP – HyperText Transfer Protocol.

HTML – HyperText Markup Language

CSS – Cascading Style Sheets

ВСТУП

З розвитком технологій та так званою «діджиталізацією» більшості процесів у сучасному світі комп’ютер стає невід’ємною частиною суспільного життя. Вища освіта також не стоїть на місці — з кожним роком студенти мають отримувати все більше інформації, щоб бути конкурентоспроможними спеціалістами. Цей процес майже повністю залежить від програми навчання, основною складовою якого є навчальний план.

Робота викладача пов’язана не тільки з навчанням інших, а й з заповненням великої кількості документації з предметів, які вони викладають. Та на деяких лягає ще серйозніша роль — формування робочого навантаження для кафедри, що базується на навчальному плані. Основною проблемою, з якою стикається працівник — це формат отримуваних даних, роботу з яким досить важко автоматизувати, що змушує перераховувати усі дані самостійно, що призводить до збільшення витраченого часу та ймовірності помилки у розрахунках через людський фактор.

Щоб уникнути таких проблем, рекомендується створити систему, яка може обробляти документи. Система має наступні завдання:

1. зчитати всі необхідні дані з плану
2. виконати всі необхідні обчислення
3. надати можливості додавання даних про виправлення
4. створити новий документ із даними результатів у необхідному форматі.

Актуальність даної теми безумовна, оскільки на цьому етапі успіх розрахунку навчального навантаження повністю залежить від уваги та точності працівників, які беруть участь у розрахунку. І оскільки план тренувань щороку змінюється з метою адаптації до сучасної реальності, навантаження постійно змінюється, що призводить до необхідності автоматизації цього процесу.

Одним з найкращих варіантів впровадження системи є веб-сторінка, яка потім завантажується на певний мережевий ресурс у відділі, щоб викладачі могли виконувати завдання без прив’язки до місця чи певного ПЗ, яке може бути недоступне на робочому комп’ютері.

1. ПОСТАНОВКА ЗАДАЧІ

З розвитком технологій всю документацію почали зберігати не тільки у паперовому, а й у електронному вигляді, що полегшує роботу з нею у декілька разів. Та проблемою залишається формат даних, який є застарілим та досить важким у роботі. Так як замінити форму збереження даних для усього університету є досить важкою та довгою задачею, то потрібно знайти спосіб полегшити обробку даних хоча б на рівні кафедри.

Навчальний план є регулюючим документом університету, який характеризує зміст навчання та керує організацією навчального процесу зі спеціальності (спеціалізації). Розробка навчальних планів є одним з найбільш важливих та відповідальних видів організаційної роботи для наукових та педагогічних працівників. Навчальні плани встановлюються робочою групою випускової кафедри, а за потреби також беруть участь представники інших кафедр.

Так як наразі змінити спосіб формування навчальних планів неможливо за відсутності підходящого забезпечення та довготривалого процесу його розробки, потрібно створити систему, що зможе обробляти файл таблиць Excel.

Задача включає в себе такі етапи:

– аналіз предметної області;

– обрання формату системи;

– можливість завантажити у систему файл для обробки;

– можливість вводити дані, що можуть змінюватись за рішенням кафедри;

– виводити результуючі дані у зручному вигляді

Додаток написаний мовою JavaScript.

2. АНАЛІЗ ПРОБЛЕМИ АВТОМАТИЗОВАНОГО ОБЧИСЛЕННЯ ПЕДАГОГІЧНОГО НАВАНТАЖЕННЯ КАФЕДРИ

Навчальний план є направляючим документом, що формулює зміст освіти та регулює організацію професійного (спеціалізованого) навчального процесу. Відповідно до цього документу розраховується навчальне навантаження кафедри.

Розрахунок педагогічного навантаження є одним з ключових методичних занять, і з цього формується план освітнього процесу на наступний навчальний рік. Аналіз існуючих систем показує, що в даний час не існує програмного продукту, який би дозволяв виконувати це завдання автоматично.

2.1. Аналіз навчального плану кафедри

Навчальний план - основний нормативний документ навчального закладу, за допомогою якого організований навчальний процес. Програма бакалавра (із зазначенням певної спеціальності) складаються у відповідному форматі. План включає графік навчального процесу, його план за семестрами та розподіл залікових кредитів між дисциплінами. Він характеризує перелік та обсяг навчальних дисциплін, обсяг та формат лекцій, лабораторних робіт та інший виів занять, форми проведення поточного та підсумкового контролю, а також державної атестації.

Педагогічне навантаження кафедри визначає процес навчання наступного року. Від цього залежить план навчання, кількість предметів, які буде викладати кожен вчитель тощо.

Плани базуються на відповідній освітньо-професійній програмі і визначають:

1. Кількість часу (години)
2. Каталог, обсяг та порядок вивчення нормативних та факультативних предметів.
3. Тип та обсяг навчальних курсів.
4. Час для самостійної роботи студентів.
5. Форма семестрового контролю.
6. Вид, обсяг та терміни практики.
7. Форму проведення державної атестації.

Робочі плани розробляються індивідуально для кожного освітньо-кваліфікаційного рівня та формату освіти. Педагогічне навантаження затверджується кафедрою і є основою для формування навчального процесу у наступному навчальному році. Графік навчального процесу для різних семестрів та форм навчання щороку встановлюється відділом університетської освіти та затверджується першим проректором. План повинен містити таку інформацію:

1. Нормативну частину навчальної програми, яка відповідає нормативній частині змісту освітньо-професійної програми.
2. Вибіркову частину програми (самойстійний вибір ВНЗ і вільний вибір студентів).
3. Назва нормативного та варіативних навчальних дисциплін.
4. Шифри та назви практик.
5. Дипломна робота.
6. Кредити та години ECTS з кожного предмета.
7. Розподіл загальної кількість годин на заняття (лекції, практики або семінари, лабораторні роботи або комп'ютерні практикуми) та самостійну роботу студентів.
8. Семестровий контроль (іспити, заліки, диференційовані заліки).
9. Курсові проекти (роботи).

10. Кількість аудиторних годин на курс та семестр.

11. Загальна кількість годин аудиторного роботи на тиждень.

12. Кількість екзаменів.

13. Кількість заліків.

14. Кількість курсових проектів (робіт).

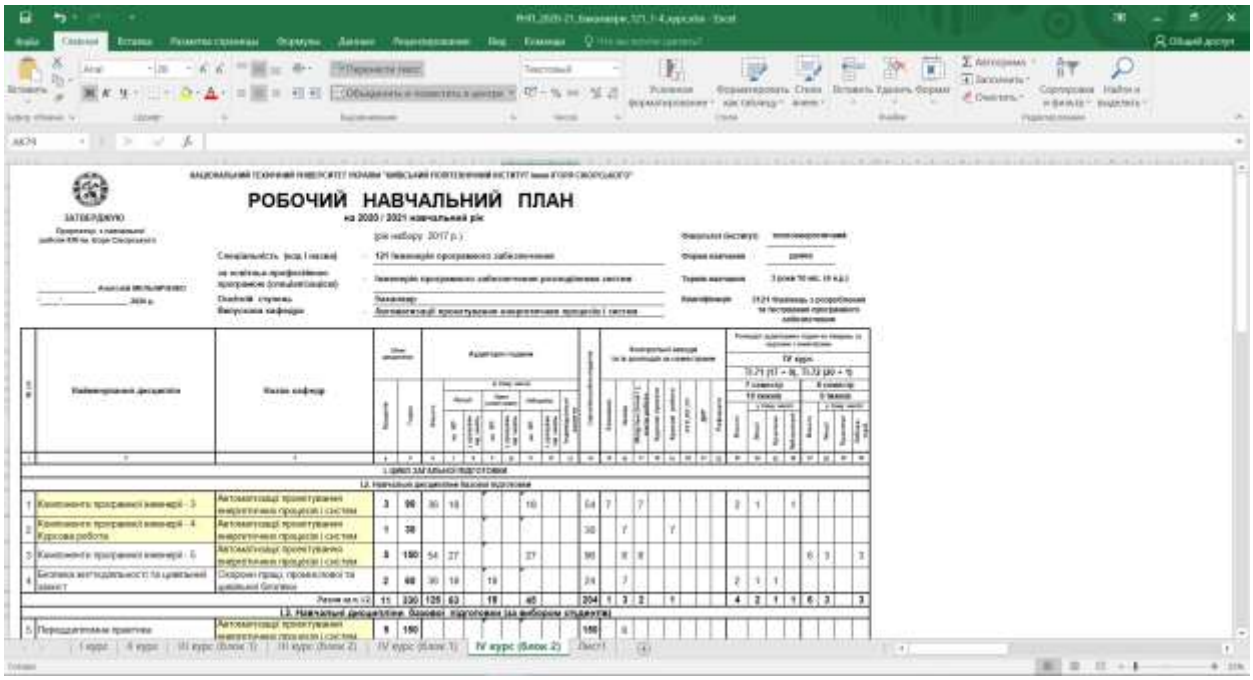

Рисунок 2.1 — зразок робочого навчального плану

2.2. Огляд існуючих систем для університетів

Попередні дослідження демонструють, що створення системи для обчислення педагогічного навантаження кафедри є досить проблематичним. Головною перешкодою для розробки є незручний формат збереження даних в університеті. Проблемою формату є нездатність до його автоматизації. Якщо у таблиці додається хоча б один стовпчик або рядок, то програма почне деякі дані обчислювати абсолютно неправильно. Також не виключається можливість розбіжностей між планами різних навчальних закладів.

Ці фактори спричинюють неможливість створення загальної системи обчислення педагогічного навантаження для вищих навчальних закладів. Якби така система і існувала, то обсяг коду був би величезним, так як програма проходила б велику кількість перевірок на різні формати даних та знаходження текстових даних. А знайти чисельні дані при зміні формату взагалі неможливо. Все вищезгадане породжує необхідність створення локальної системи для обчислення педагогічного навантаження кафедри.

3. ЗАСОБИ РОЗРОБКИ ПРОГРАМНОГО ПРОДУКТУ

Для розробки програмного продукту було використано наступні засоби:

– середовище Visual Studio Code;
– мова програмування JavaScript 6;
– мова розмітки HTML;
– таблиця стилів CSS;
– фреймворк Bootstrap

3.1. Середовище розробки Visual Studio Code

Visual Studio Code [1] - це редактор вихідного коду, розроблений Microsoft. Позиціонується як "полегшений" редактор коду для крос-платформної розробки веб-і хмарних додатків. Включаючи налагоджувач, інструмент пошуку Git, підсвічування синтаксису, інструмент рефакторингу. Він також має широкий спектр параметрів налаштування: власні теми, комбінації клавіш, велику кількість плагінів для покращення роботи певних інструментів тощо. Поширюється безкоштовно та розроблений як програмне забезпечення з відкритим кодом. Код Visual Studio заснований на Electron і реалізований через веб-редактор Monaco, розроблений для Visual Studio Online.

На рисунку 3.1 зображено інтерфейс Visual Studio Code.

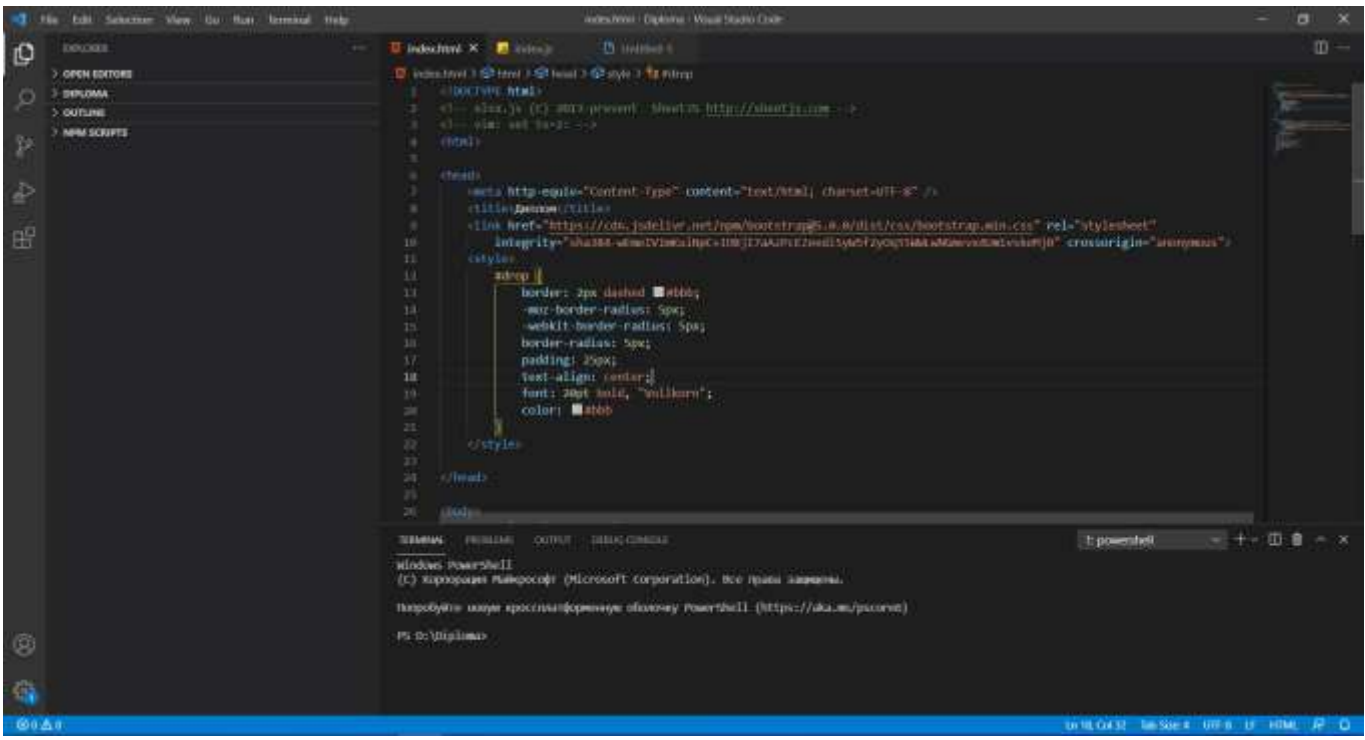

Рисунок 3.1 – інтерфейс Visual Studio Code

3.2. Мова розмітки HTML 5

Мова розмітки HTML [2] - стандартизована мова розмітки у Всесвітній павутині. Більшість веб-сторінок містять описи розмітки у форматі HTML (або XHTML). HTML інтерпретується браузером; сформований відформатований текст відображається на моніторі комп'ютера або екрані мобільного пристрою. У Всесвітній павутині сторінки HTML зазвичай передаються із сервера в браузер за допомогою HTTP або HTTPS, простого тексту або за допомогою шифрування.

У HTML ви можете вбудовувати код мови програмування JavaScript для контролю поведінки та вмісту веб-сторінок. Крім того, CSS, що міститься в HTML, описує зовнішній вигляд і макет сторінки. Усі HTML документи є наборами елементів, а початок і кінець елемента представлені спеціальними тегами. Елемент може бути порожнім, тобто він не містить жодного тексту чи

інших даних. У цьому випадку кінцевий тег зазвичай не вказується (наприклад, тег розриву рядка
 є одиничним і не потребує закриття). Крім того елемент може мати атрибути, які визначають будь-яку з його властивостей (наприклад, атрибут href = "" у посиланні). Атрибути вказані у початковій роботі.

Окрім елементів, в документах HTML є англійські сутності - "спеціальні символи". Сутність починається з символу & і має форму &name; або &#NNNN;, де NNNN - код символу Unicode у десяткових записах.

Кожен документ HTML, який відповідає будь-якій версії специфікації HTML, повинен бути у версії HTML <! DOCTYPE …>

Потім початок і кінець документа позначається тегами <html> та </html> відповідно. Усередині цих тегів повинні бути теги заголовка (<head> </head>) та тіла (<body> </body>) документа.

3.3. CSS

CSS [3] - це формальна мова, що використовується для опису зовнішнього вигляду документів (веб-сторінок), написаних мовою розмітки (як правило, HTML або XHTML). Він також може бути застосований до будь-якого XML-документа, такого як SVG або XUL.

Творці веб-сторінок використовують CSS для визначення кольорів, шрифтів, стилів, макету окремих блоків та інших аспектів зовнішнього вигляду цих сторінок. Основною метою розробки CSS є відокремлення опису логічної структури веб-сторінки (заповненої з використанням HTML або інших мов розмітки) від опису зовнішнього вигляду веб-сторінки (тепер заповнена за допомогою формальної мови CSS). Цей розділ може збільшити доступність документа, забезпечити більшу гнучкість та можливість керувати

його поданням, а також зменшити складність та повторюваність структурованого вмісту.

Крім того, CSS дозволяє представити один і той же документ у різних стилях або методах виводу, таких як візуалізація екрана, друк, читання голосу (спеціальний голосовий браузер чи програма зчитування з екрана) або під час виведення на пристрої, що використовують шрифт Брайля.

3.4. Мова програмування JavaScript

JavaScript або JS [4] - мова програмування з багатьма парадигмами. Підтримує об’єктно-орієнтований, імперативний та функціональний стилі. Це реалізація специфікації ECMAScript.

JavaScript часто використовується як вбудована мова для програмного доступу до об'єктів програми. Найбільш широко використовується як мова сценаріїв у браузерах, щоб зробити веб-сторінки інтерактивними[5].

Основні архітектурні особливості: динамічні тип, слабкі тип, автоматичне управління пам’яттю, програмування прототипу, функція — об’єкт першого класу.

На JavaScript вплинуло багато мов, і мета створення полягала в тому, щоб зробити мову схожою на Java. JavaScript не належить жодній компанії чи організації, яка відрізняє його від багатьох мов програмування, що використовуються у веб-розробці.

Основними перевагами JavaScript є:

- Швидкість для кінцевого користувача, так як веб-сторінки частково обробляються на стороні користувача без запитів до сервера
- Підтримуваність скриптів усіма популярними браузерами
- Потужна інфраструктура з великою кількістю фреймворків та готових рішень

- Простота у використанні — прості задачі вирішуються невеликою кількістю рядків коду, для важких завдань існує багато готових рішень, які можна адаптувати під конкретну задачу
- Зручність для створення інтерфейсів користувача, зумовлена великою кількістю обробників подій

Недоліками мови є:

- Доступність для зловмисників — у вільну скриптову мову легко додати фрагмент, що містить шкідливий код
- Нестрога типізація, що може викликати несподівані результати при приведенні різних типів даних
- Відсутність можливості знаходження помилок на ранніх етапах розробки [7]

В JavaScript:

- всі ідентифікатори чутливі до регістру
- в назвах змінних можна використовувати підкреслення, літери, арабські цифри, символ долара
- назви змінних не можуть починатися з цифри
- для оформлення однорядкових коментарів використовуються //, багаторядкові і внутрішні коментарі починаються з / * і закінчуються * /

Структурно JavaScript можна представити у вигляді об'єднання трьох відмінних одна від одної частин:

- ядро (ECMAScript),
- об'єктна модель браузера (Browser Object Model або BOM (англ.)),
- об'єктна модель документа (Document Object Model або DOM).

Якщо розглядати JavaScript не у браузерному середовищі, то об'єктна модель браузера і об'єктна модель документа можуть не підтримуватися.

ECMAScript не є браузерною мовою, а також не визначає методи вводу та виводу. Швидше, це основа для побудови мови сценаріїв. Специфікація

ECMAScript описує типи даних, інструкції, ключові слова та зарезервовані слова, оператори, об’єкти та регулярні вирази, не обмежуючи авторів похідних мов розширювати їх на нові компоненти.

Об'єктна модель браузера - браузер-специфічна частина мови, що є прошарком між ядром і об'єктною моделлю документа. Основне призначення об'єктної моделі браузера - управління вікнами браузера і забезпечення їх взаємодії. Кожне з вікон браузера представляється об'єктом window, центральним об'єктом DOM.

Об'єктна модель документа - інтерфейс програмування додатків для HTML і XML-документів. Згідно DOM, документ (наприклад, веб-сторінка) може бути представлений у вигляді дерева об'єктів, з багатьма властивостями, що дозволяють виконувати над ним різні операції:

- генерація і додавання вузлів,
- отримання вузлів,
- зміна вузлів,
- зміна зв'язків між вузлами,
- видалення вузлів[7].

3.5. Bootstrap

Bootstrap[8] — вільний набір інструментів для створення сайтів і веб-додатків. Включає в себе HTML- і CSS-шаблони оформлення для типографіки, веб-форм, кнопок, міток, блоків навігації та інших компонентів веб-інтерфейсу, включаючи JavaScript-розширення.

Bootstrap використовує сучасні напрацювання в області CSS і HTML, тому необхідно бути уважним при підтримці старих браузерів.

Перевагами використання Bootstrap для розробки веб-додатків:

- Висока швидкість розробки якісної адаптивної верстки навіть розробниками-початківцями
- Кроссбраузерність та кроссплатформеність в усіх браузерах, що його підтримують
- Низький вхідний поріг — для роботи з фреймворком немає необхідності мати глибокі знання в HTML, CSS, JavaScript, достатньо знати хоча б основи
- Однорідність дизайну для усіх компонентів
- Велика кількість навчальних матеріалів
- Можливість налаштувань під свій проект

Так як і більшість подібних фреймворків, Bootstrap має і недоліки:

- Кінцеві файли проекту мають більший розмір, так як файли містять універсальний код, з якого потрібною для проекту є лише частина
- Складність використання Bootstrap для верстки сайтів з унікальним дизайном

Основні інструменти Bootstrap:

- Сітки - заздалегідь задані розміри колонок, які можна відразу ж використовувати, наприклад, ширина колонки 140px відноситься до класу .span2 (.col-md-2 в третій версії фреймворка), який можна використовувати в CSS-описі документа.
- Шаблони - фіксований або гумовий шаблон документа.
- Типографіка - опис шрифтів, визначення деяких класів для шрифтів, таких як код, цитати і т. П.
- Медіа - надає деяке упорядкування зображень та відео.
- Таблиці - кошти оформлення таблиць, аж до додавання функціональності сортування.
- Форми - класи для оформлення форм і деяких подій, що відбуваються з ними.
- Навігація - класи оформлення для панелей, вкладок, переходу по сторінках, меню і панелі інструментів.

- Алерт - оформлення діалогових вікон, підказок і спливаючих вікон.

4. ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

Для створення системи було використано середовище Visual Studio Code, так як у нього безліч переваг, а основною з них є можливість широкої кастомізації середовища під власні потреби.

Мовою програмування для реалізації даного проекту обрано JavaScript. Серед переваг можна виділити, в першу чергу, динамічну типізацію даних, що значно полегшує зчитування даних з файлу та зміну їх типів. Також велику роль відіграє широкий набір бібліотек для цієї мови. Додатковою причиною було те, що додаток планується розмістити на певному веб-ресурсі, тому він має бути реалізований у форматі веб-сторінки.

4.1 Зчитування даних

Першою частиною роботи програми є зчитування даних з .xlsx файлу. Для цього була використана бібліотека SheetJS, що надає широкий вибір можливостей парсингу файлу у різні формати. З допомогою влаштованих функцій бібліотеки були реалізовані дві функції, що відповідають за форматування вмісту .xlsx файлу у JSON формат, з яким можна зручно працювати. Функція do_file() відповідає за створення нової сутності FileReader та виклику методу readAsArrayBuffer() для зчитування файлу, після чого викликається наступна функція. Функція process_wb() відповідає за обробку книги. Всередині реалізовано функцію форматування змісту у JSON. У масив result записується результат парсингу книги, здійснений за допомогою влаштованого методу бібліотеки sheet_to_json(), після чого повертається рядок JSON. В кінці функція викликає саму себе та виконує функцію to_json().

```
var process_wb = (function () {
    var to_json = function to_json(workbook) {
        var result = {};
        workbook.SheetNames.forEach(function (sheetName) {
            var roa = X.utils.sheet_to_json(workbook.Sheets[sheetName], { header: 1 });
            if (roa.length) result[sheetName] = roa;
        });
        return JSON.stringify(result, 2, 2);
    };

    return function process_wb(wb) {
        global_wb = wb;
        output = to_json(wb)

    };
})();
```

Рисунок 4.1 – функція process_wb()

```
var do_file = (function () {
    return function do_file(files) {
        var f = files[0];
        var reader = new FileReader();
        reader.onload = function (e) {
            var data = e.target.result;
            data = new Uint8Array(data);
           process_wb(X.read(data, { type: 'array' }));
        };
        reader.readAsArrayBuffer(f);
        calculate_button.disabled = false
    };
})();
```

Рисунок 4.2 – функція do_file()

Обидві функції є типу IIFE (Immediately Invoked Function Expression), тобто функціями, які виконуються одразу після того, як були визначені.

Завантаження файлу для обробки виконано у вигляді drag-and-drop, обробляє який відповідна анонімна функція. Вона містить дві функції: для обробки перетягування файлу, у якій відбувається виклик функції обробки файлу, та скидання його у поле прийому. Також у цій функції реалізовано додавання обробників трьох подій: dragenter, dragover, drop.

```
(function () {
    var drop = document.getElementById('drop');
    if (!drop.addEventListener) return;

    function handleDrop(e) {
        e.stopPropagation();
        e.preventDefault();
        do_file(e.dataTransfer.files);
    }

    function handleDragover(e) {
        e.stopPropagation();
        e.preventDefault();
        e.dataTransfer.dropEffect = 'copy';
    }

    drop.addEventListener('dragenter', handleDragover, false);
    drop.addEventListener('dragover', handleDragover, false);
    drop.addEventListener('drop', handleDrop, false);
})();
```

Рисунок 4.3 – функція для drag-and-drop

В результаті виконання цих функцій ми отримуємо рядок JSON, з зчитаною інформацією з файла.

4.2 Проведення обчислень

Основною функцією усієї програми є функція calculate(), яка приймає в себе JSON рядок та за допомогою влаштованої функції об’єкту JSON parse() конвертує його у об’єкт, зображений на рисунку 4.4.


```
"{
  "I курс": [
    [
      null,
      "НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ \"КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО\""
    ],
    [],
    [
      null,
      "РОБОЧИЙ   НАВЧАЛЬНИЙ   ПЛАН"
    ],
    [
      null,
      null,
      null,
      null,
      null,
      null,
      null,
      null,
      null,
      null,
      null,
      null,
      null,
      null,
      null,
      null,
      null,
      null,
      null,
      null,
      null,
      null,
      "          ЗАТВЕРДЖУЮ",
      null,
      null,
      null,
      "на 2020 / 2021 навчальний рік   "
    ],
    [
      null,
      null,
      null,
      null,
      null,
      null,
      null,
      null,
      null,
      null,
      null,
      null,
      null,
      null,
      null,
      null,
      null,
      null,
      null,
      null,
      null,
      null,
      "     Проректор  з навчальної роботи КПІ ім. Ігоря Сікорського",
      null,
      null,
      null,
      "прийом 2020 року",
      null,
      null,
      null,
```


Рисунок 4.4 — результат виконання JSON.parse()

Після цього створюється новий об’єкт courses, з такими самими ключами. Також здійснюється перевірка, чи значення не дорівнює «Лист1», так як там немає потрібної для обробки інформації. Після цього значення за ключами заповнюються у новий об’єкт після перевірки, чи значення містить рядок з назвою кафедри.

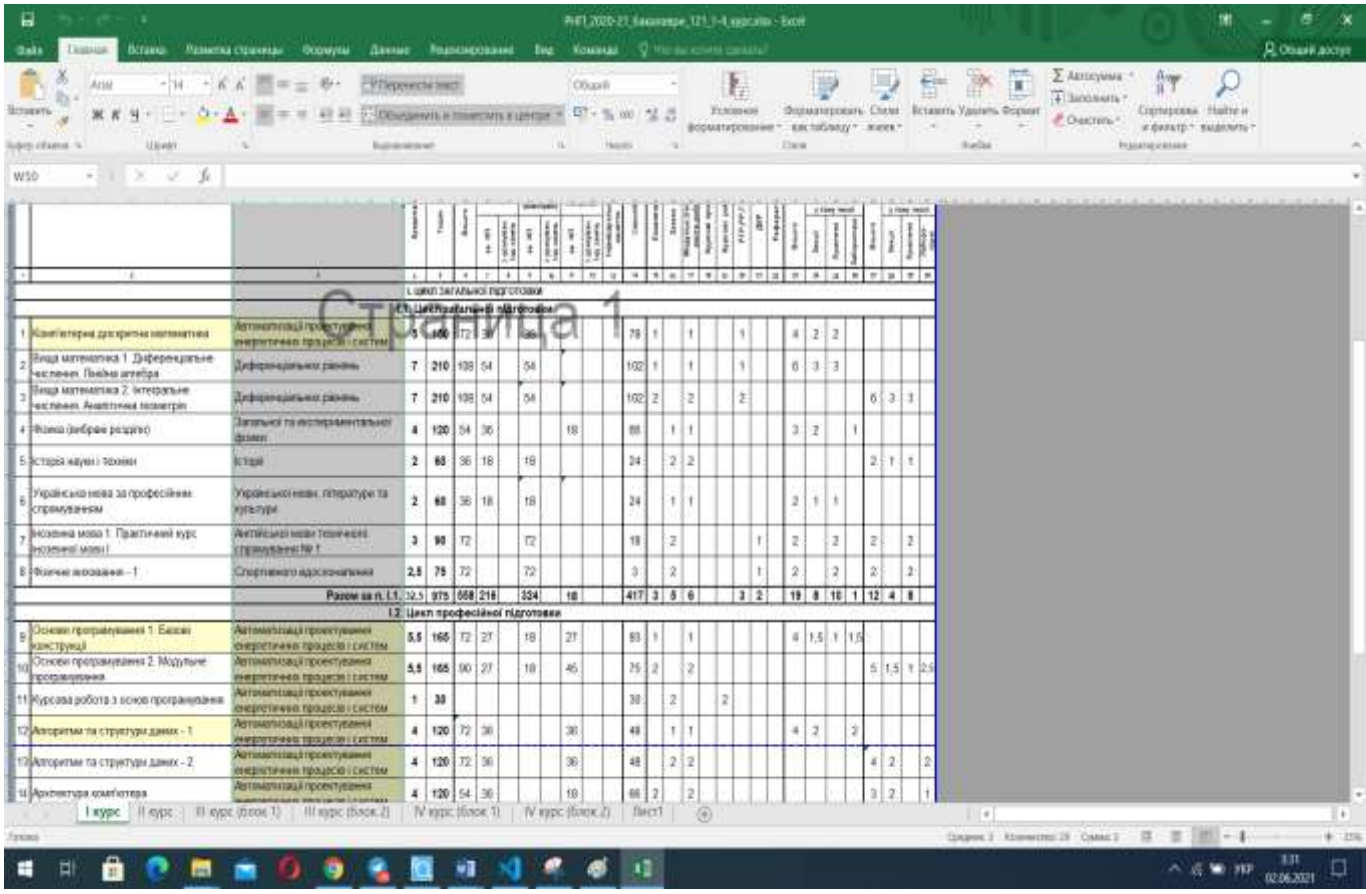

Рисунок 4.5 — стовпчик, за яким здійснюється фільтрація за кафедрою

Також відфільтровується масив, що містить рядок «Випускова кафедра», так як це рядок з «шапки» документу, і не містить жодної потрібної інформації. Наступним етапом є додавання до певного елементу кожного масиву об’єкту, що містить назву дисципліни, курсу, на якому вона читається. Це відбувається після перевірки значення ключа, так як третій та четвертий курси поділені на два листи з додатковими написами «(блок 1)» та «(блок 2)». Результат виконання зображений на рисунку 4.6. Після цього здійснюється виклик інших функцій програми.

```
index.js:102
▼{I курс: Array(8), II курс: Array(14), III курс (блок 1): Array(16), III курс (блок 2): Array(16), IV курс (блок 1): Array(11), …}
  ▼I курс: Array(8)
    ▶0: (55) [null, 1, null, null, null, null, null, null, null, null, null, null, null, null, null, null, null, null, null, null, "Комп'ютерна дискретна математика - I курс", null, null, …
    ▶1: (55) [null, 9, null, null, null, null, null, null, null, null, null, null, null, null, null, null, null, null, null, null, "Основи програмування 1. Базові конструкції - I курс", nu…
    ▶2: (58) [null, 10, null, null, null, null, null, null, null, null, null, null, null, null, null, null, null, null, null, null, "Основи програмування 2. Модульне програмування - I курс…
    ▶3: (55) [null, 11, null, null, null, null, null, null, null, null, null, null, null, null, null, null, null, null, null, null, "Курсова робота з основ програмування - I курс", null, n…
    ▶4: (55) [null, 12, null, null, null, null, null, null, null, null, null, null, null, null, null, null, null, null, null, null, "Алгоритми та структури даних - 1 - I курс", null, null,…
    ▶5: (58) [null, 13, null, null, null, null, null, null, null, null, null, null, null, null, null, null, null, null, null, null, "Алгоритми та структури даних - 2 - I курс", null, null,…
    ▶6: (58) [null, 14, null, null, null, null, null, null, null, null, null, null, null, null, null, null, null, null, null, null, "Архітектура комп'ютера - I курс", null, null, "Автомати…
    ▶7: (58) [null, 15, null, null, null, null, null, null, null, null, null, null, null, null, null, null, null, null, null, null, "Компоненти програмної інженерії 1. Вступ до програмної …
     length: 8
    ▶__proto__: Array(0)
  ▶II курс: (14) [Array(55), Array(53), Array(55), Array(51), Array(55), Array(55), Array(58), Array(58), Array(58), Array(55), Array(58), Array(58), Array(58), Array(58)]
  ▶IV курс (блок 1): (11) [Array(55), Array(55), Array(58), Array(55), Array(55), Array(58), Array(55), Array(55), Array(54), Array(55), Array(55)]
  ▶IV курс (блок 2): (15) [Array(55), Array(55), Array(58), Array(55), Array(55), Array(55), Array(54), Array(54), Array(55), Array(58), Array(58), Array(55), Array(55), Array(58), A…
  ▶III курс (блок 1): (16) [Array(54), Array(55), Array(47), Array(55), Array(58), Array(57), Array(58), Array(55), Array(55), Array(55), Array(55), Array(55), Array(57), Array(57), …
  ▶III курс (блок 2): (16) [Array(54), Array(55), Array(47), Array(55), Array(58), Array(57), Array(58), Array(55), Array(55), Array(55), Array(55), Array(55), Array(57), Array(57), …
```

Рисунок 4.6 — результат виконання функції calculate()

```
const calculate = (output) => {
    var output_parsed = JSON.parse(output)
    let download_button = document.getElementById("download_button")
    let refresh_button = document.getElementById("refresh_button")
    Object.keys(output_parsed).forEach((course) => {
        if (course !== 'Лист1') {
            courses[course] = [];
        }
    });

    Object.entries(output_parsed).forEach(([course, sheet]) => {
        sheet.forEach(lessonsArr => {
            if (lessonsArr.includes(cathedra)) {
                courses[course].push(lessonsArr);
            }
        })

    });
    for (const course in courses) {
        courses[course] = courses[course].filter(lesson => !lesson.includes(graduation_cathedra))
    }
```

Рисунок 4.7 – функція calculate()

```
    Object.entries(courses).forEach(([course, data]) => {
        data.forEach(d => {
            if (course === "III курс (блок 1)" || course === "III курс (блок 2)") {
                d[19] = d[19].concat(" - III курс")
            } else if (course === "IV курс (блок 1)" || course === "IV курс (блок 2)") {
                d[19] = d[19].concat(" - IV курс")
            } else
                d[19] = d[19].concat(" - ", course);
        })
    });
    fillFlows(courses)
    fillGroups(output_parsed)
    fillResultArr(courses);
    createXLSX(fillFinalResult(sortResult(result_arr)))
    calculate_button.disabled = true
    download_button.disabled = false
    refresh_button.disabled = false
}
```

Рисунок 4.8 – функція calculate()

Наступним завданням було реалізувати можливість додати кількість потоків для кожного курсу. Для цього була створена функція fillFlows(), що приймає у себе об’єкт courses. На початку функції оголошено чотири змінні для кількості потоків і присвоєно їм значення інструменту взаємодії з користувачем prompt. Після чого введене користувачем значення, конвертоване у числове унарним оператором «+», за ключем додається до відповідного масиву у об’єкті.

```
const fillFlows = courses => {
    let flow_1 = prompt("Введіть кількість потоків для 1 курсу")
    let flow_2 = prompt("Введіть кількість потоків для 2 курсу")
    let flow_3 = prompt("Введіть кількість потоків для 3 курсу")
    let flow_4 = prompt("Введіть кількість потоків для 4 курсу")
    Object.entries(courses).forEach(([course, value]) => {
        value.forEach(v => {
            if (course === "I курс") {
                v.push(+flow_1)
            } else if (course === "II курс") {
                v.push(+flow_2)
            } else if (course === "III курс (блок 1)" || course === "III курс (блок 2)") {
                v.push(+flow_3)
            } else if (course === "IV курс (блок 1)" || course === "IV курс (блок 2)") {
                v.push(+flow_4)
            }
        })
    })
}
```

Рисунок 4.9 – функція fillFlows()

Наступною реалізованою функцією є функція для заповнення змінної groupsArr групами — fillGroups(). Спочатку відбувається перевірка ключа на «Лист1», після чого у двох блоках умовного оператора масиву присвоюється значення конкретної комірки, що містить рядок з групами, розділений комою. Далі у кожен масив загального об’єкту додаються елементи з кількістю груп, що обчислюється за кількістю елементів у рядку з групами розділених комами, кількість підгруп для практичних занять та лабораторних робіт, кількість бюджетних та контрактних студентів, обчислена у наступній функції та загальний рядок з групою для подальшого використання.

```
const fillGroups = output => {
    Object.entries(output).forEach(([course, group]) => {
        let groupsArr
        if (course === 'Лист1') {
            return
        }
        if (course.includes("IV курс")) {
            groupsArr = group[13][49].split(', ');
            let resultSubGroups = fillSubGroups(groupsArr)
            courses[course].forEach(value => {
                value.push(group[13][49].split(', ').length, resultSubGroups[0], resultSubGroups[0], resultSubGroups[1], resultSubGroups[2], group[13][49])
            })

        } else {
            groupsArr = group[12][49].split(', ');
            let resultSubGroups = fillSubGroups(groupsArr)
            courses[course].forEach(value => {
                value.push(group[12][49].split(', ').length, resultSubGroups[0], resultSubGroups[0], resultSubGroups[1], resultSubGroups[2],group[12][49])
            })
        }
    });

}
```

Рисунок 4.10 – функція fillGroups().

Функція fillSubGroups() приймає параметр groupsArr. На початку функції оголошено три змінні: count, amountOfBudgetStudents, amountOfContractStudents, які відповідають за назву групи з кількістю студентів, кількість бюджетників та кількість контрактників відповідно. Наступним завданням є знайти кількість студентів у кожній групі. Під час перебору groupsArr створюються дві локальні змінні, що відповідають за загальну кількість студентів у групі та рядок з кількостями студентів. Для отримання значень кількостей змінній з рядком спочатку присвоюється значення, відділене дужками, після чого переприсвоюється за індексом, меншим на 1 від довжини рядка. Далі відбувається перевірка, чи довжина після знаку «+» більша за одиницю, після чого відділені значення кількості бюджетників та контрактників присвоюються відповідним змінним. Якщо сумарна кількість студентів у групі більша за 15, то надається можливість ввести кількість підгруп у даній групі, після чого значення присвоюється змінній count. В іншому випадку count збільшується на одиницю. Функція повертає масив, що складається з кількості підгруп, бюджетників та контрактників.

```
const fillSubGroups = groupsArr => {
    let count = 0
    let amountOfBudgetStudents = 0
    let amountOfContractStudents = 0
    groupsArr.forEach(group => {
        let totalAmountOfStudentsInGroup = 0
        let stringAmountOfStudentsInGroup
        stringAmountOfStudentsInGroup = group.substring(group.indexOf('(') + 1)
        debugger
        stringAmountOfStudentsInGroup = stringAmountOfStudentsInGroup.substring(0, stringAmountOfStudentsInGroup.length - 1)
        debugger
        if (stringAmountOfStudentsInGroup.split(' + ').length > 1) {
            amountOfBudgetStudents += +stringAmountOfStudentsInGroup.split(' + ')[0]
            amountOfContractStudents += +stringAmountOfStudentsInGroup.split(' + ')[1]
            totalAmountOfStudentsInGroup = +stringAmountOfStudentsInGroup.split(' + ')[0] + +stringAmountOfStudentsInGroup.split(' + ')[1]
        } else {
            amountOfBudgetStudents += +stringAmountOfStudentsInGroup
            totalAmountOfStudentsInGroup = +stringAmountOfStudentsInGroup
        }
        if (totalAmountOfStudentsInGroup > 15) {
            count += +prompt(`Оберіть кількість підгруп для ${group}`)
        } else {
            count += 1
        }
    })
    return [count, amountOfBudgetStudents, amountOfContractStudents]
}
```

Рисунок 4.11 – функція fillSubGroups().

Наступним етапом є об’єднання однакових предметів, що читаються у різних груп на одному курсі. Функція sortResult() приймає масив, значення якого копіюються spread-оператором у нову змінну та ініціалізується додатковий масив. Реалізацію було здійснено наступним чином:

- Знаходження однакових предметів
- Додавання потрібних величин до першого предмета
- Заміна назви одного з однакових предметів на нуль
- Додавання ненульових предметів у масив

Першим етапом є цикл, який проходиться масивом, перевіряє, чи перший елемент не є нулем, після чого починається цикл while(), де перевіряються перші елементи двох масивів на рівність, і якщо перевірка успішні, то значення групи другого масива приєднується до відповідного елемента першого, а також додаються значення кількості підгруп. Після чого значенню першого елемента другого масива присвоюється нуль. Наступний цикл пропускає нульові значення, а інші додає у результуючий масив, який

функція повертає. Також на етапі виконання цієї функції до першого елемента масиву, що містить назву дисципліни, додаються:

- Кількість груп, для яких вона читається
- Кількість годин, виділених на предмет

В результаті функція повертає оновлений масив з відсортованими даними.

```
const sortResult = (result_arr) => {
    let result_updated = []
    let result_tmp = [...result_arr]

    for (let i = 0; i < result_tmp.length; i++) {
        if (result_tmp[i][0] === 0) {
            continue
        }
        let j = i + 1
        while (j < result_tmp.length) {
            if (result_tmp[i][0] === result_tmp[j][0]) {
                result_tmp[i][20] = result_tmp[i][20].concat(", ", result_tmp[j][20])
                for (let k = 14; k < 19; k++) {
                    result_tmp[i][k] += result_tmp[j][k]
                }

                result_tmp[j][0] = 0
                break
            }
            j++
        }

    }
    for (let i = 0; i < result_tmp.length; i++) {
        if (result_tmp[i][0] === 0) {
            continue
        }
        result_tmp[i][0] = result_tmp[i][0].concat(", ", result_tmp[i][14], "гр, ", result_tmp[i][20], ", ", result_tmp[i][1], "год")

        result_tmp[i].splice(20, 1)
        result_updated.push(result_tmp[i])

    }
    debugger
    return result_updated
}
```

Рисунок 4.9 – функція sortResult().

Далі функція fillFinalResult() приймає вихідне значення попередньої функції, і у цей масив додає підраховані значення, такі як кількість лекційних годин, практичних тощо. Повертає функція оновлений масив.

```
const fillFinalResult = result_updated => {
    result_updated.forEach(res => {
        res.push(res[2] * res[19], res[3] * res[15], res[4] * res[16], res[5] * res[14] * 0.25, res[6] * res[17] * 0.33, res[7] * res[14] * 2,
        res[8] * res[17] * 0.25, res[9] * res[17], res[10] * res[17], res[11] * res[17] * 0.5, res[12] * res[17] * 0.33, res[13] * res[17] * 0.25,
        res[6] * res[14] * 2 * 0.06 * res[1] * res[17] / 25)
    })
    result_updated.forEach(res => {
        res.push(res[20] + res[21] + res[22] + res[23] + res[24] + res[25] + res[26] + res[27] + res[28] + res[29] + res[30] + res[31])
    })
    return result_updated
}
```

Рисунок 4.10 – функція fillFinalResult().

Завершальною функцією процесу розрахунку є функція формування .xlsx файлу createXLSX(). Вона приймає вихідне значення попередньої функції, і спочатку вирізає допоміжні значення. Після чого засобами бібліотеки створюється нова книга, новому масиву присвоюються значення результуючого, і туди ж на початок додаються елементи, що будуть «шапкою» документу. Після чого масив масивів форматується у лист, викликається функція запису та її результат повертається.

```
const createXLSX = result_updated => {
    console.log(result_updated)
    for (let i = 0; i < result_updated.length; i++) {
        result_updated[i].splice(17, 0, null)
        result_updated[i].splice(20, 0, null)
        result_updated[i].splice(21, 0, null)
    }
    let wb = XLSX.utils.book_new();
    wb.SheetNames.push("Test Sheet");
    let ws_data = result_updated;
    ws_data.unshift([
        "Факультет(ін-т), який забезпечується (абрев.)\n Назва  дисципліни, її загальний обсяг в годинах. Курс навчання, шифр груп, к-сть студентів в кожній групі (Б+К)",
        "Обсяг дисципліни за семестр, N годин",
        "Лекції, Л годин",
        "Практ. (комп.практикум, семінари), П годин",
        "Лабор.,  L годин",
        "Індивідуальні заняття  I",
        "Екзамени, E",
        "Заліки, Z  *",
        "Контр.роб (мод., темат.), M",
        "Курсові проекти, Q годин",
        "Курсові роботи, G годин",
        "РГР, РР, ГР,    R",
        "ДКР, D",
        "Реферати,   F",
        "Академічні бюдж., Г",
        "Підгрупи для ПЗ, ГП",
        "Підгр. для лаб.роб., ГL",
        "Академ. контрактні, ГК",
        "Б",
        "К",
        "БК",
        "КК",
```

Рисунок 4.11 – функція createXLSX().

```
        "Кількість бюджетних потоків, P",
        "Лекції, Л x P",
        "Практ.заняття (комп. практикум, семінари), П x ГП",
        "Лабораторні заняття, L x ГL",
        "Індивідуальні заняття IxГx0,25",
        "Екзамени, 0.33xEx(Б+БК)",
        "Заліки,  2xГx2",
        "Контр.роб.(мод.,темат.), 0.25xMx(Б+БК)",
        "Курсові проекти, Qx(Б+БК)",
        "Курсові роботи, Gx(Б+БК)",
        "РГР, РР, ГР,  0.5xRx(Б+БК)",
        "ДКР, 0.33xDx(Б+БК)",
        "Реферати, 0.25xFx(Б+БК)",
        "Консультації, 2 x E x Г + 0,06 x N x (Б+БК)/25",
        "Всього годин"
    ])
    let ws = XLSX.utils.aoa_to_sheet(ws_data);
    wb.Sheets["Test Sheet"] = ws;
    wbout = XLSX.write(wb, { bookType: 'xlsx', type: 'binary' });
    return wbout
}
```

Рисунок 4.11 – функція createXLSX().

Так як тип контенту для Excel має бути octet stream, то для цього нам необхідно повернене значення попередньої функції конвертувати. Функція s2ab() приймає бінарні значення, та конвертує їх спочатку у ArrayBuffer, після чого з цього значення формується масив Uint8Array. Далі усі символи конвертуються у octet і повертаються.

```
function s2ab(s) {
    var buf = new ArrayBuffer(s.length); //convert s to arrayBuffer
    var view = new Uint8Array(buf);  //create uint8array as viewer
    for (var i = 0; i < s.length; i++) view[i] = s.charCodeAt(i) & 0xFF; //convert to octet
    return buf;
}
```

Рисунок 4.12 – функція s2ab().

Завершальним етапом є завантаження кінцевого файлу. Функція downloadFile() приймає повернене значення, і використовуючи функціонал бібліотеки FileSaverJs зберігає файл на пристрої.

```
const downloadFile = (wbout) => {
    saveAs(new Blob([s2ab(wbout)], { type: "application/octet-stream" }), 'test.xlsx');
};
```

Рисунок 4.13 – функція downloadFile().

На цьому етап проведення обчислень було завершено, наступним етапом стало оформлення зовнішнього вигляду елементів додатку.

4.3 Створення представлення

Візуальна частина програмного продукту виконана засобами HTML, CSS та Bootstrap. У верхній частині сторінки розташований блок, що містить drag-and-drop поле. Під ним розташовані три кнопки: «Рахувати», «Завантажити», «Оновити». На початку усі кнопки є неактивними.

Рисунок 4.14 – Інтерфейс продукту

Після перетягування файлу у поле, активується кнопка «Рахувати». По закінченню обчислень знову деактивується, і стають доступними кнопки «Завантажити» та «Оновити», що завантажують файл та оновлюють сторінку відповідно. Результатом виконання програми є файл формату .xlsx.

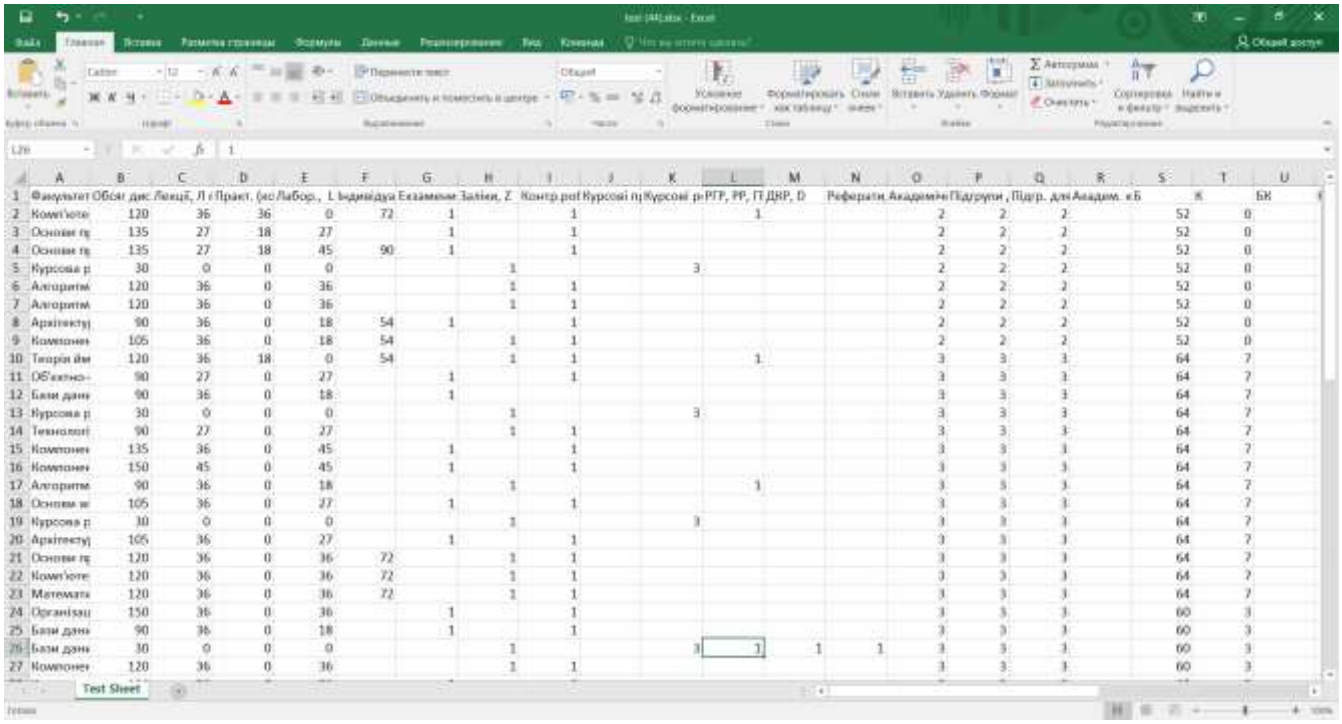

Рисунок 4.15 – Результуючий файл

Розмітка сторінки написана засобами HTML і знаходиться у основному файлі index.html. Основним елементом є з классом «container», у якому знаходяться усі інші елементи. У ньому реалізований блок, що містить поле для перетягування файлів, а також три елемента <button>, що відповідають за кнопки.

```html
<!DOCTYPE html>
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <title>Диплом</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0/dist/css/bootstrap.min.css" rel="stylesheet"
        integrity="sha384-wEmeIV1mKuiNpC+IOBjI7aAzPcEZeedi5yW5f2yOq55WWLwNGmvvx4Um1vskeMj0" crossorigin="anonymous">
        <link href="index.css" rel="stylesheet">
</head>

<body>
    <div class="container">
        <div id="drop" class="row m-3 justify-content-center">Перетягніть файл для обробки</div>
        <div class="row m-3 justify-content-center">
            <button type="button" id='calculate_button' class="btn btn-outline-primary col-4" disabled>Рахувати</button>
            <button type="button" id='download_button' class="btn btn-outline-secondary col-4"
                disabled>Завантажити</button>
            <button type="button" id='refresh_button' class="btn btn-outline-success col-4" disabled>Оновити</button>
        </div>
        <div id="button"></div>
        <div id="button_1"></div>
        <div id="button_2"></div>
    </div>
    <pre id="out"></pre>
    <br />
    <script src="dist/cpexcel.js"></script>
    <script src="shim.js"></script>
    <script src="jszip.js"></script>
    <script src="xlsx.js"></script>
    <script src="FileSaver.min.js"></script>
    <script src="index.js">
    </script>
</body>

</html>
```

Рисунок 4.16 – index.html

Стиль для поля міститься у файлі index.css, підключеному до головного. Для задання правила був використаний селектор за

ідентифікатором «drop». Для коректного відображення елементу у різних браузерах були використані властивості з префіксами –moz та –webkit.

Перший відповідає за відображення у браузері Mozilla Firefox, другий — за браузери Opera, Safari, Chrome.

```
#drop {
    border: 2px dashed #bbb;
    -moz-border-radius: 5px;
    -webkit-border-radius: 5px;
    border-radius: 5px;
    padding: 25px;
    text-align: center;
    font: 20pt bold, "Vollkorn";
    color: #bbb
}
```

Рисунок 4.17 – index.css

Стилі для кнопок були обрані з тих, що надає фреймворк Bootstrap. Для того, щоб використовувати ці стилі необхідно або за допомогою спеціального тегу <link> підключити інструмент до проекту, або встановити його безпосередньо у сам проект. Стилі підключаються завдяки наданню елементу відповідного класу, які можна взяти на офіційному сайті фреймворку.

```
<div id="drop" class="row m-3 justify-content-center">Перетягніть файл для обробки</div>
<div class="row m-3 justify-content-center">
    <button type="button" id='calculate_button' class="btn btn-outline-primary col-4" disabled>Обчислити</button>
    <button type="button" id='download_button' class="btn btn-outline-secondary col-4"
        disabled>Завантажити</button>
    <button type="button" id='refresh_button' class="btn btn-outline-success col-4" disabled>Оновити</button>
```

Рисунок 4.18 — елементи з Bootstrap класами

5. РОБОТА КОРИСТУВАЧА З ПРОГРАМНОЮ СИСТЕМОЮ

Основним призначенням розробленого програмного продукту є автоматизація планування навчального навантаження кафедри. Загальну use-case діаграму роботи користувача з системою можна побачити на рисунку 5.1.

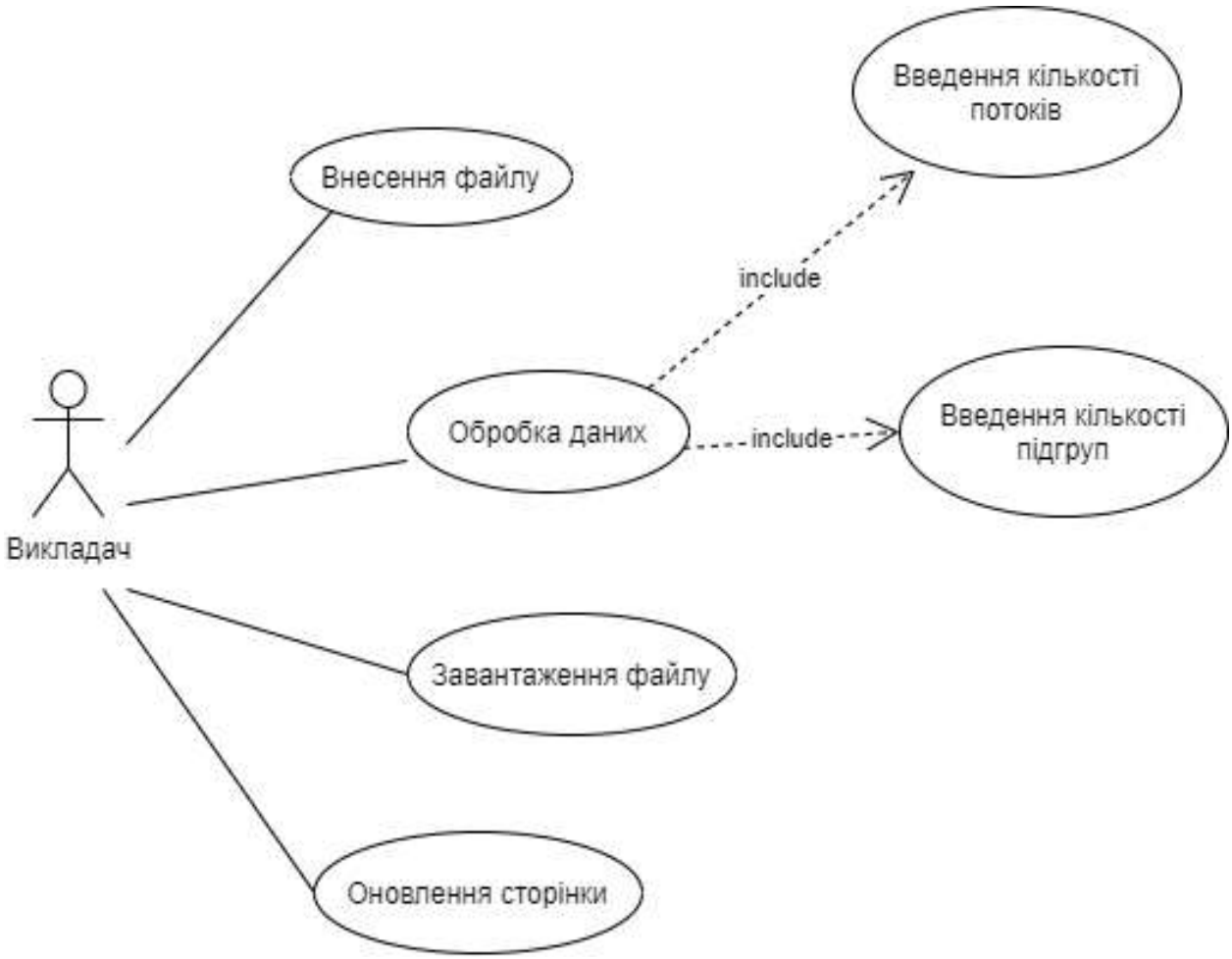

Рисунок 5.1 — use-case діаграма процесу

На діаграмі зображено загальний сценарій роботи користувача з системою, у якій він може внести файл, запустити обробку даних, ввести додаткові дані, завантажити файл та оновити сторінку.

Для забезпечення безвідмовної роботи системи треба дотримуватися основних вимог та рекомендацій щодо її використання.

5.1. Системні вимоги

Робота системи потребує підтримки JavaScript вашим браузером та наявність проекту з допоміжними файлами на комп’ютері, так як продукт поки являється локальним.

5.2. Сценарій роботи користувача з системою

Так як на даному етапі застосунок не є завантаженим на сервер кафедри, то запустити його можна просто відкривши файл index.html. Якщо за замовчуванням у користувача не обрано, у якому браузері відкривати файли такого типу, то буде запропоновано обрати браузер.

На веб-сторінці користувач має змогу побачити поле для перетягування файлу та 3 неактивні кнопки. Головний процес починається після перетягування користувачем файлу з планом, зразок якого можна побачити на рисунку 2.1.

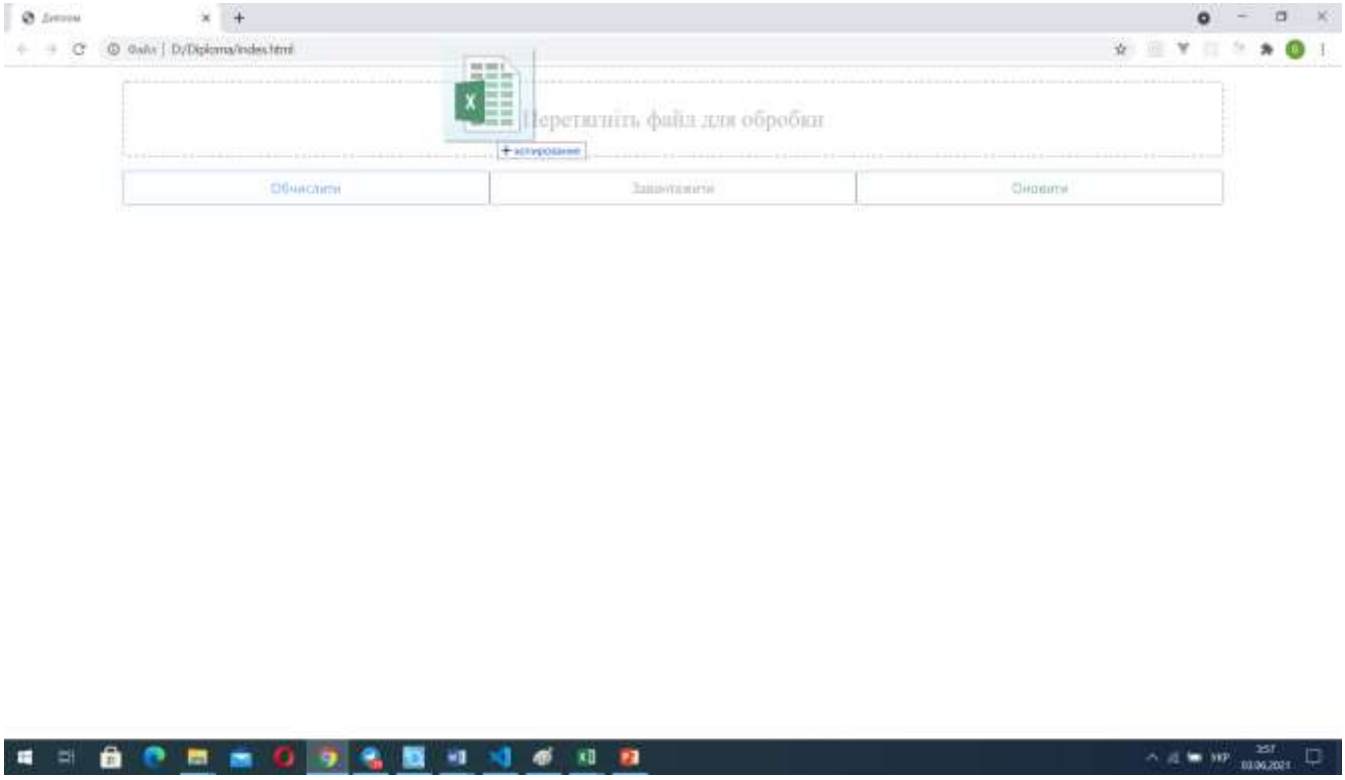

Рисунок 5.1 — приклад перетягування файлу

Після того, як користувач відпускає ЛКМ, починається зчитування файлу. По закінченню цього процесу, який зазвичай займає менше однієї секунди, активується кнопка «Обчислити».

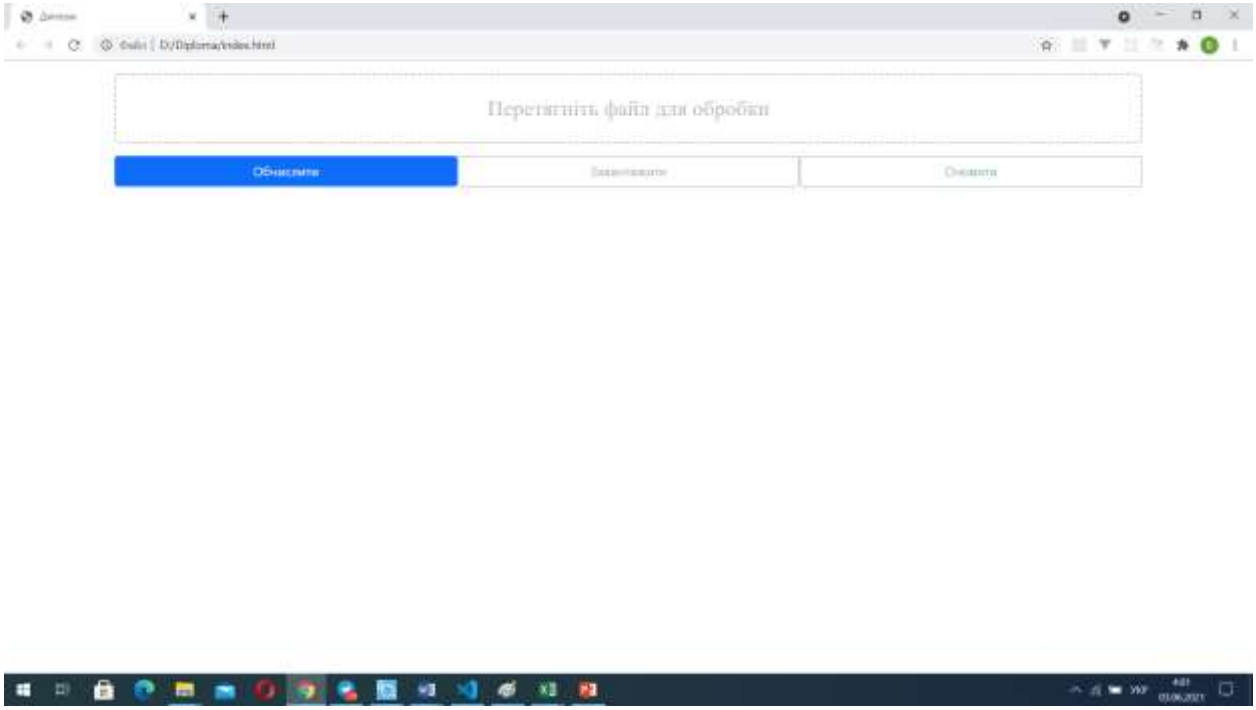

Рисунок 5.2 — кнопка «Обчислити» активована

Якщо користувач завантажує файл, що не відповідає загальному вигляду робочого навчального плану, то система видає попередження, що файл неправильний, та перезавантажує сторінку, після чого користувач може знову завантажити новий файл.

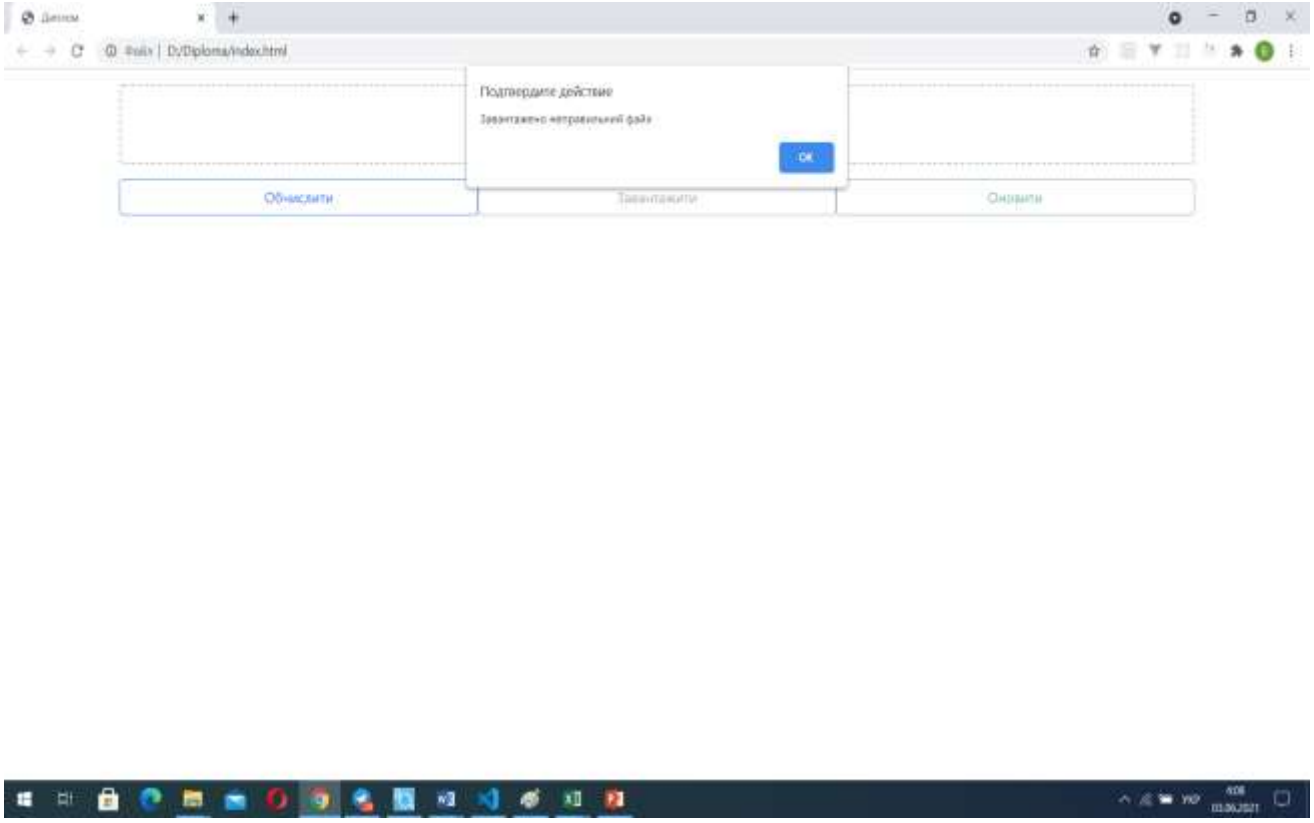

Рисунок 5.3 — попередження про неправильний завантажений файл

Якщо файл пройшов перевірку, то після натискання кнопки «Обчислити» починається процес обробки даних. Користувачеві надається змога ввести кількість потоків для кожного з курсів у відповідних полях, що з’являться у верхній частині екрану.

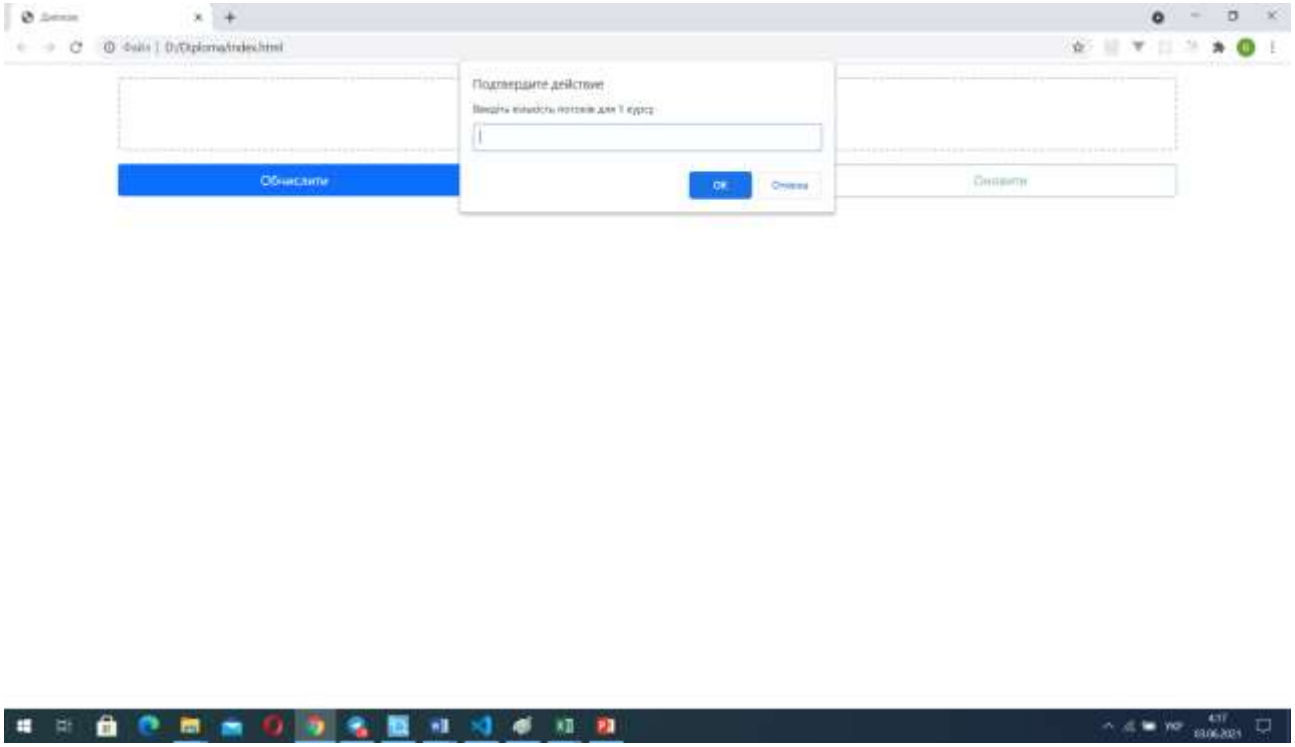

Рисунок 5.4 — приклад поля для вводу кількості потоків

Після введення даних про кількість потоків, програма продовжує обробку файлу, і якщо виявляє групи, у яких кількість студентів перевищує 15, то пропонує користувачеві для таких груп обрати кількість підгруп, на які вони будуть поділені під час практичних або лабораторних занять.

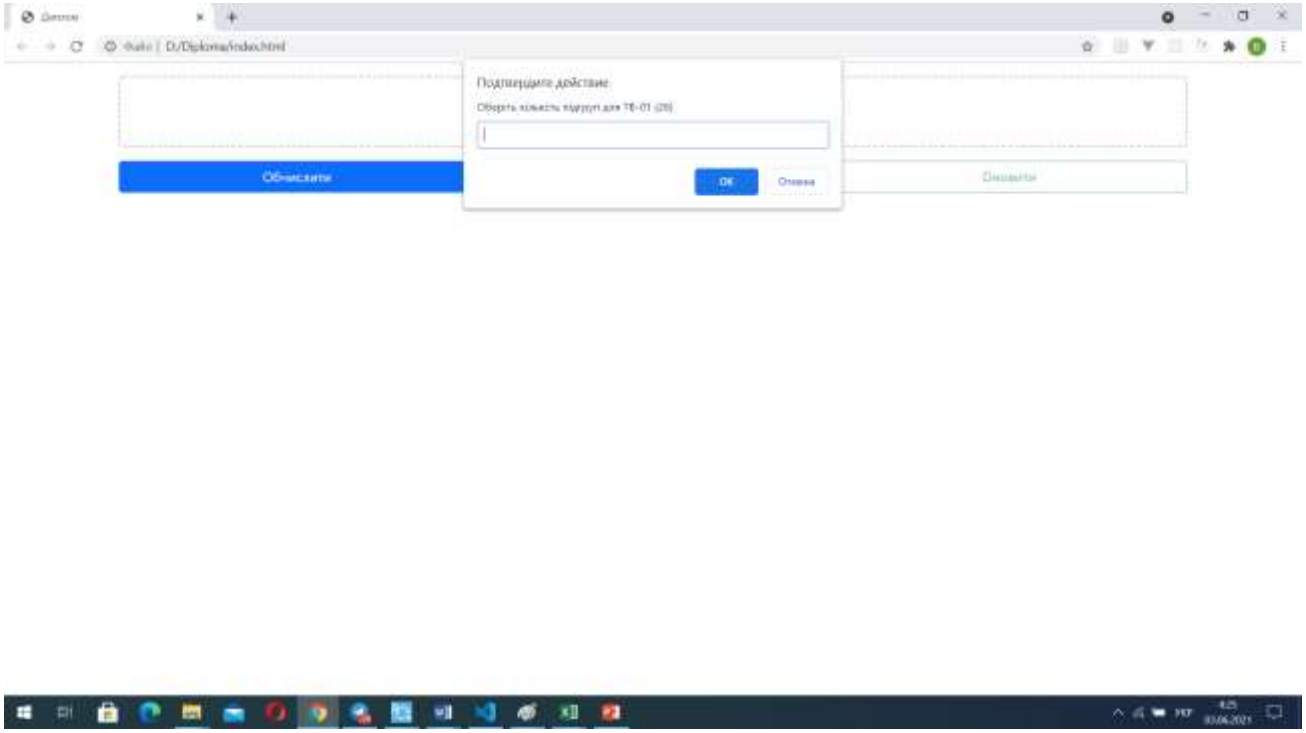

Рисунок 5.5 — приклад поля для вводу кількості підгруп для групи

По закінченню введення даних про кількість підгруп для кожної з груп, програма продовжує обробку даних. Після закінчення усіх процесів обробки даних та формування нового Excel файлу з обчисленим навантаженням кафдери кнопка «Обчислити» деактивується, натомість користувач отримує змогу завантажити файл, натиснувши вже активну кнопку «Завантажити».

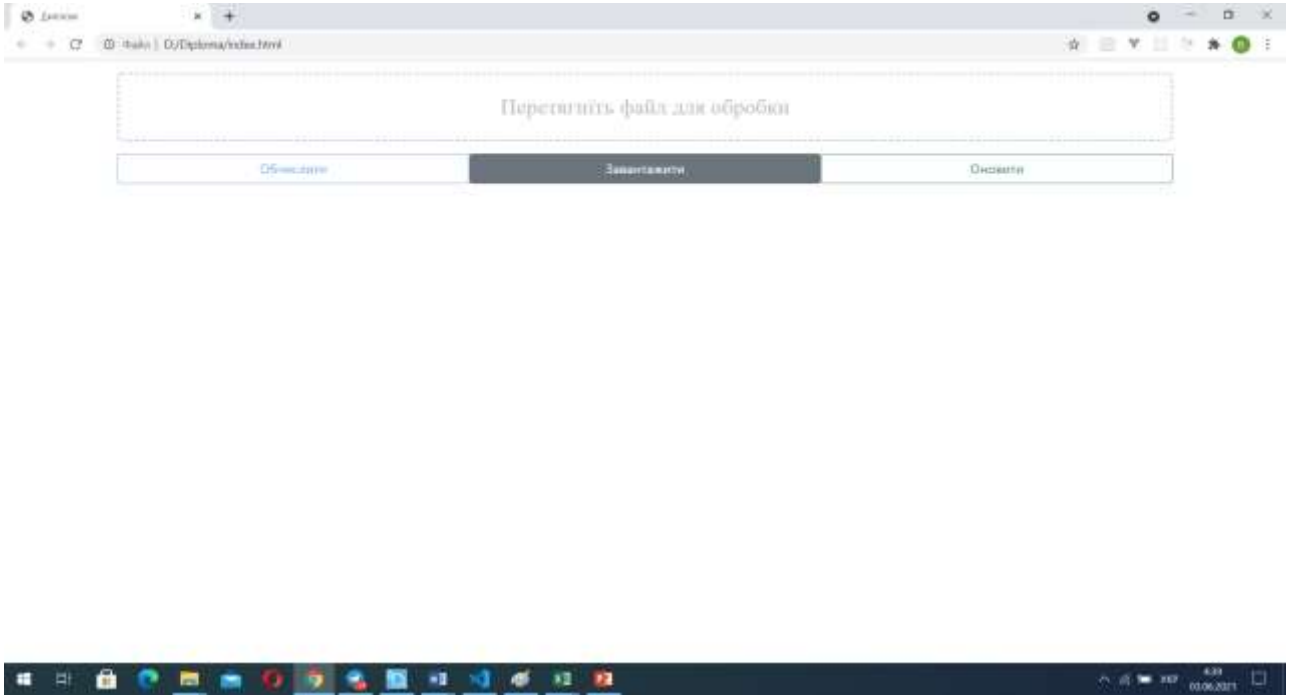

Рисунок 5.6 — кнопка «Завантажити» активована

Також користувачеві надається змога оновити сторінку для продвоження роботи з додатком. Для цього користувач тисне кнопку «Оновити», що була активована разом з кнопкою для завантаження, після чого програма оновлює сторінку браузера.

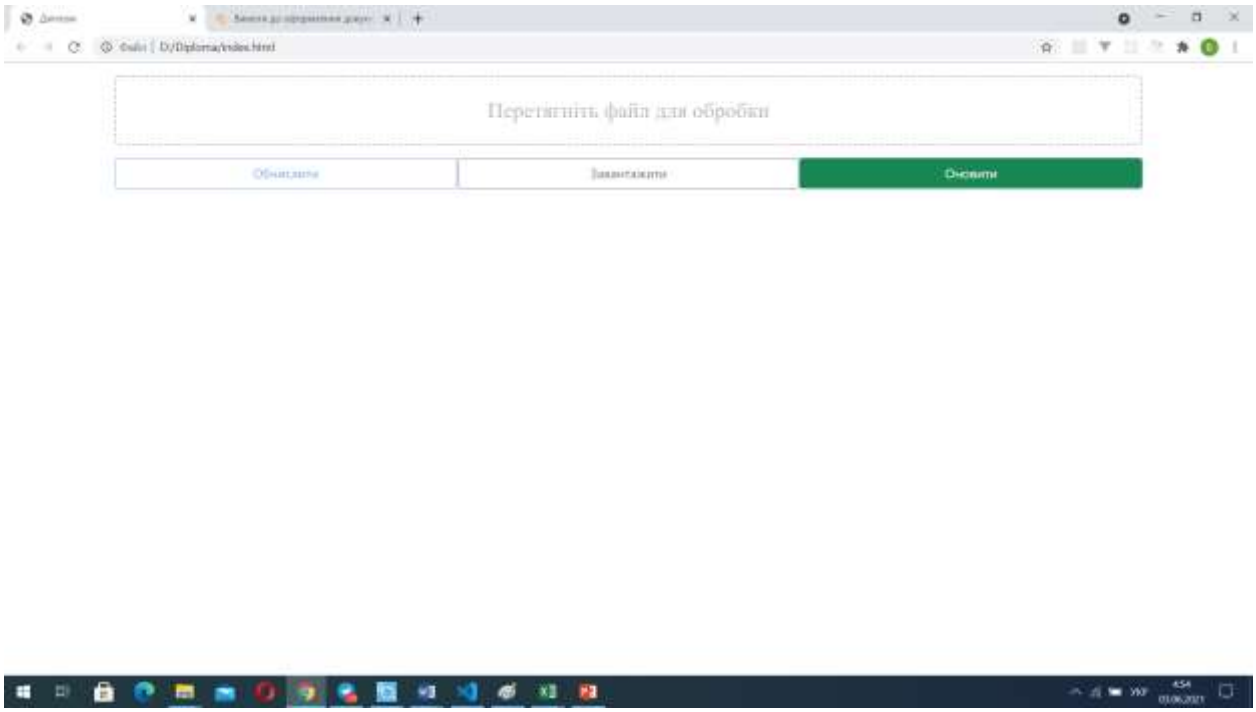

Рисунок 5.7 — кнопка «Оновити» активована

ВИСНОВКИ

В ході виконання даної роботи було досліджено процес обробки навчального плану для обчислення педагогічного навантаження кафедри. Було проведено дослідження на предмет існування програмних рішень для автоматизації цього процесу. Обгрунтовано актуальність створення програмного продукту для автоматизованого обчислення педагогічного навантаження кафедри. Розроблено програмний продукт, що дозволяє обчислити всі необхідні дані та внести додаткові. Створена система є веб-додатком, тому не потребуватиме встановлення зі сторони користувача при розміщенні її на сервері кафедри.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Visual Studio Code [Електронний ресурс] — Режим доступу: https://code.visualstudio.com/docs
2. HTML [Електронний ресурс] — Режим доступу: http://htmlbook.ru/html
3. Самоучитель CSS [Електронний ресурс] — Режим доступу: http://htmlbook.ru/samcss
4. Современный учебник JavaScript [Електронний ресурс] — Режим доступу: https://learn.javascript.ru/
5. JavaScript [Електронний ресурс] — Режим доступу: https://developer.mozilla.org/en-US/docs/Web/JavaScript
6. Флэнаган Д. JavaScript. Подробное руководство, 6-е издание. – Пер. с англ. – СПб: Символ-Плюс, 2012. – 1080 с. – ISBN 978-5-93286-215-5
7. Крокфорд Д. Как устроен JavaScript – Питер – 2019. – 304 с. – ISBN 978-5-4461-1260-9
8. Bootstrap Docs [Електронний ресурс] — Режим доступу: https://getbootstrap.com/docs/5.0/getting-started/introduction/

Додаток А

Інформаційна система автоматизації планування навчального навантаження кафедри

Специфікація

УКР.НТУУ”КПІ”ІМ.ІГОРЯ_СІКОРСЬКОГО_ТЕФ_АПЕПС_ТВ7186_21А 1

Аркушів 2

Київ – 2021

Позначення	Найменування	Примітки
Документація		
УКР.НТУУ”КПІ”ІМ.ІГОРЯ_СІКОРСЬКОГО_ТЕФ_АПЕПС_ТВ7186_21А	Пояснювальна записка.docx	Пояснювальна записка
Компоненти		
УКР.НТУУ”КПІ”ІМ.ІГОРЯ_СІКОРСЬКОГО_ТЕФ_АПЕПС_ТВ7186_21А-1	index.js, , index.html, index.css	Основний функціонал та стилізація
УКР.НТУУ”КПІ”ІМ.ІГОРЯ_СІКОРСЬКОГО_ТЕФ_АПЕПС_ТВ7186_21А-2	jszip.js, shim.js, xlsx.js, cpexcel.js, FileSaver.min.js	Допоміжні файли бібліотек

Додаток Б

Інформаційна система автоматизації планування навчального навантаження кафедри

Лістинг програми

УКР.НТУУ”КПІ”ІМ.ІГОРЯ_СІКОРСЬКОГО_ТЕФ_АПЕПС_ТВ7186_21Б 2

Аркушів 13

Київ – 2021

index.js

```
var X = XLSX;
var global_wb;
var output = "";
const cathedra = "Автоматизації проектування енергетичних процесів і систем"
const graduation_cathedra = "Випускова кафедра"
const courses = {};
const groups = {}
const result_arr = []
let wbout = null
let calculate_button = document.getElementById("calculate_button")


var process_wb = (function () {
    var to_json = function to_json(workbook) {
        var result = {};
        workbook.SheetNames.forEach(function (sheetName) {
            var roa = X.utils.sheet_to_json(workbook.Sheets[sheetName], { header: 1 });
            if (roa.length) result[sheetName] = roa;
        });
        if (Object.values(result)[0][2][1] !== "РОБОЧИЙ   НАВЧАЛЬНИЙ   ПЛАН") {
            alert("Завантажено неправильний файл")
            document.location.reload()
        }
        return JSON.stringify(result, 2, 2);
    };

    return function process_wb(wb) {
        global_wb = wb;
        output = to_json(wb)
    };
})();

var do_file = (function () {
    return function do_file(files) {
        var f = files[0];
        var reader = new FileReader();
        reader.onload = function (e) {
            var data = e.target.result;
            data = new Uint8Array(data);
            process_wb(X.read(data, { type: 'array' }));
        };
        reader.readAsArrayBuffer(f);
        calculate_button.disabled = false
    };
})();

(function () {
    var drop = document.getElementById('drop');
    if (!drop.addEventListener) return;
```

```
    function handleDrop(e) {
        e.stopPropagation();
        e.preventDefault();
        do_file(e.dataTransfer.files);
    }

    function handleDragover(e) {
        e.stopPropagation();
        e.preventDefault();
        e.dataTransfer.dropEffect = 'copy';
    }

    drop.addEventListener('dragenter', handleDragover, false);
    drop.addEventListener('dragover', handleDragover, false);
    drop.addEventListener('drop', handleDrop, false);
})();



const calculate = (output) => {
    var output_parsed = JSON.parse(output)
    let download_button = document.getElementById("download_button")
    let refresh_button = document.getElementById("refresh_button")

    Object.keys(output_parsed).forEach((course) => {
        if (course !== 'Лист1') {
            courses[course] = [];
        }
    });

    Object.entries(output_parsed).forEach(([course, sheet]) => {
        sheet.forEach(lessonsArr => {
            if (lessonsArr.includes(cathedra)) {
                courses[course].push(lessonsArr);
            }
        })

    });
    for (const course in courses) {
        courses[course] = courses[course].filter(lesson => !lesson.includes(graduation_cathedra))
    }
    Object.entries(courses).forEach(([course, data]) => {
        data.forEach(d => {
            if (course === "III курс (блок 1)" || course === "III курс (блок 2)") {
                d[19] = d[19].concat(" - III курс")
            } else if (course === "IV курс (блок 1)" || course === "IV курс (блок 2)") {
                d[19] = d[19].concat(" - IV курс")
            } else
                d[19] = d[19].concat(" - ", course);
        })
    });
```

```
    fillFlows(courses)
    fillGroups(output_parsed)
    fillResultArr(courses);
    createXLSX(fillFinalResult(sortResult(result_arr)))
    calculate_button.disabled = true
    download_button.disabled = false
    refresh_button.disabled = false
}

const fillResultArr = courses => {
    Object.values(courses).forEach(value => {
        value.forEach(v => {
            if (v[41] !== null) {
                result_arr.push(Array.of(v[19], v[31] - 30, v[33] + v[34], v[35] + v[36], v[37] + v[38],
v[39], v[41], v[42], v[43], v[44], v[45], v[46], v[47], v[48],
                    v[v.length - 6], v[v.length - 5], v[v.length - 4], v[v.length - 3], v[v.length - 2],
v[v.length - 7], v[v.length - 1]))
            } else {
                result_arr.push(Array.of(v[19], v[31], v[33] + v[34], v[35] + v[36], v[37] + v[38],
v[39], v[41], v[42], v[43], v[44], v[45], v[46], v[47], v[48],
                    v[v.length - 6], v[v.length - 5], v[v.length - 4], v[v.length - 3], v[v.length - 2],
v[v.length - 7], v[v.length - 1]))
            }

        })
    })
}
// 14-18
const sortResult = (result_arr) => {
    let result_updated = []
    let result_tmp = [...result_arr]

    for (let i = 0; i < result_tmp.length; i++) {
        if (result_tmp[i][0] === 0) {
            continue
        }
        let j = i + 1
        while (j < result_tmp.length) {
            if (result_tmp[i][0] === result_tmp[j][0]) {
                result_tmp[i][20] = result_tmp[i][20].concat(", ", result_tmp[j][20])
                for (let k = 14; k < 19; k++) {
                    result_tmp[i][k] += result_tmp[j][k]
                }

                result_tmp[j][0] = 0
                break
            }
            j++
        }

    }
    for (let i = 0; i < result_tmp.length; i++) {
```

```
        if (result_tmp[i][0] === 0) {
            continue
        }
        result_tmp[i][0] = result_tmp[i][0].concat(", ", result_tmp[i][14], "гр, ", result_tmp[i][20],
", ", result_tmp[i][1], "год")
        result_tmp[i].splice(20, 1)
        result_updated.push(result_tmp[i])
    }
    debugger
    console.log(result_updated)
    return result_updated
}

const fillFinalResult = result_updated => {
    result_updated.forEach(res => {
        res.push(res[2] * res[19], res[3] * res[15], res[4] * res[16], res[5] * res[14] * 0.25, res[6] *
res[17] * 0.33, res[7] * res[14] * 2,
            res[8] * res[17] * 0.25, res[9] * res[17], res[10] * res[17], res[11] * res[17] * 0.5, res[12]
* res[17] * 0.33, res[13] * res[17] * 0.25,
            res[6] * res[14] * 2 * 0.06 * res[1] * res[17] / 25)
    })
    result_updated.forEach(res => {
        res.push(res[20] + res[21] + res[22] + res[23] + res[24] + res[25] + res[26] + res[27] +
res[28] + res[29] + res[30] + res[31])
    })
    return result_updated
}

const fillFlows = courses => {
    let flow_1 = prompt("Введіть кількість потоків для 1 курсу")
    let flow_2 = prompt("Введіть кількість потоків для 2 курсу")
    let flow_3 = prompt("Введіть кількість потоків для 3 курсу")
    let flow_4 = prompt("Введіть кількість потоків для 4 курсу")
    Object.entries(courses).forEach(([course, value]) => {
        value.forEach(v => {
            if (course === "I курс") {
                v.push(+flow_1)
            } else if (course === "II курс") {
                v.push(+flow_2)
            } else if (course === "III курс (блок 1)" || course === "III курс (блок 2)") {
                v.push(+flow_3)
            } else if (course === "IV курс (блок 1)" || course === "IV курс (блок 2)") {
                v.push(+flow_4)
            }
        })
    })
}

const fillGroups = output => {
    Object.entries(output).forEach(([course, group]) => {
        let groupsArr
        if (course === 'Лист1') {
```

```
            return
        }

        if (course.includes("IV курс")) {
            groupsArr = group[13][49].split(', ');
            let resultSubGroups = fillSubGroups(groupsArr)
            courses[course].forEach(value => {
                value.push(group[13][49].split(', ').length, resultSubGroups[0], resultSubGroups[0],
resultSubGroups[1], resultSubGroups[2], group[13][49])
            })
            debugger
        } else {
            groupsArr = group[12][49].split(', ');
            let resultSubGroups = fillSubGroups(groupsArr)
            courses[course].forEach(value => {
                value.push(group[12][49].split(', ').length, resultSubGroups[0], resultSubGroups[0],
resultSubGroups[1], resultSubGroups[2], group[12][49])
            })
        }
    });
}

const fillSubGroups = groupsArr => {
    let count = 0
    let amountOfBudgetStudents = 0
    let amountOfContractStudents = 0
    groupsArr.forEach(group => {
        let totalAmountOfStudentsInGroup = 0
        let stringAmountOfStudentsInGroup
        stringAmountOfStudentsInGroup = group.substring(group.indexOf('(') + 1)
        stringAmountOfStudentsInGroup = stringAmountOfStudentsInGroup.substring(0,
stringAmountOfStudentsInGroup.length - 1)
        if (stringAmountOfStudentsInGroup.split(' + ').length > 1) {
            amountOfBudgetStudents += +stringAmountOfStudentsInGroup.split(' + ')[0]
            amountOfContractStudents += +stringAmountOfStudentsInGroup.split(' + ')[1]
            totalAmountOfStudentsInGroup = +stringAmountOfStudentsInGroup.split(' + ')[0] +
+stringAmountOfStudentsInGroup.split(' + ')[1]
        } else {
            amountOfBudgetStudents += +stringAmountOfStudentsInGroup
            totalAmountOfStudentsInGroup = +stringAmountOfStudentsInGroup
        }
        if (totalAmountOfStudentsInGroup > 15) {
            count += +prompt(`Оберіть кількість підгруп для ${group}`)
        } else {
            count += 1
        }
    })
    return [count, amountOfBudgetStudents, amountOfContractStudents]
}

const createXLSX = result_updated => {
    for (let i = 0; i < result_updated.length; i++) {
```

```
        result_updated[i].splice(17, 0, null)
        result_updated[i].splice(20, 0, null)
        result_updated[i].splice(21, 0, null)
    }
    let wb = XLSX.utils.book_new();
    wb.SheetNames.push("Test Sheet");
    let ws_data = result_updated;
    ws_data.unshift([
        "Факультет(ін-т), який забезпечується (абрев.)\n Назва  дисципліни, іі загальний обсяг
в годинах. Курс навчання, шифр груп, к-сть студентів в кожній групі (Б+К)",
        "Обсяг дисципліни за семестр, N годин",
        "Лекції, Л годин",
        "Практ. (комп.практикум, семінари), П годин",
        "Лабор.,  L годин",
        "Індивідуальні заняття  I",
        "Екзамени, E",
        "Заліки, Z  *",
        "Контр.роб (мод., темат.), M",
        "Курсові проекти, Q годин",
        "Курсові роботи, G годин",
        "РГР, РР, ГР,    R",
        "ДКР, D",
        "Реферати,   F",
        "Академічні бюдж., Г",
        "Підгрупи для ПЗ, ГП",
        "Підгр. для лаб.роб., ГL",
        "Академ. контрактні, ГК",
        "Б",
        "К",
        "БК",
        "КК",
        "Кількість бюджетних потоків, P",
        "Лекції, Л x P",
        "Практ.заняття (комп. практикум, семінари), П x ГП",
        "Лабораторні заняття, L x ГL",
        "Індивідуальні заняття IxГx0,25",
        "Екзамени, 0.33xEx(Б+БК)",
        "Заліки,  2xГxZ",
        "Контр.роб.(мод.,темат.), 0.25xMx(Б+БК)",
        "Курсові проекти, Qx(Б+БК)",
        "Курсові роботи, Gx(Б+БК)",
        "РГР, РР, ГР,  0.5xRx(Б+БК)",
        "ДКР, 0.33xDx(Б+БК)",
        "Реферати, 0.25xFx(Б+БК)",
        "Консультації, 2 x E x Г + 0,06 x N x (Б+БК)/25",
        "Всього годин"
    ])
    let ws = XLSX.utils.aoa_to_sheet(ws_data);
    wb.Sheets["Test Sheet"] = ws;
    wbout = XLSX.write(wb, { bookType: 'xlsx', type: 'binary' });
    return wbout
}
```

```
function s2ab(s) {
   var buf = new ArrayBuffer(s.length); //convert s to arrayBuffer
   var view = new Uint8Array(buf);  //create uint8array as viewer
   for (var i = 0; i < s.length; i++) view[i] = s.charCodeAt(i) & 0xFF; //convert to octet
   return buf;
}

const downloadFile = (wbout) => {
   saveAs(new Blob([s2ab(wbout)], { type: "application/octet-stream" }), 'test.xlsx');
};

document.addEventListener("DOMContentLoaded", () => {
   document.getElementById("calculate_button").addEventListener("click", () => {
calculate(output) }),
      document.getElementById("download_button").addEventListener("click", () => {
downloadFile(wbout) }),
      document.getElementById("refresh_button").addEventListener("click", () => {
location.reload() })
});
```

index.html

```
<!DOCTYPE html>
<html>
<head>
	<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
	<title>Диплом</title>
	<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0/dist/css/bootstrap.min.css"
rel="stylesheet"
		integrity="sha384-
wEmeIV1mKuiNpC+IOBjI7aAzPcEZeedi5yW5f2yOq55WWLwNGmvvx4Um1vskeMj0"
crossorigin="anonymous">
		<link href="index.css" rel="stylesheet">
</head>

<body>
	<div class="container">
		<div id="drop" class="row m-3 justify-content-center">Перетягніть файл для
обробки</div>
		<div class="row m-3 justify-content-center">
			<button type="button" id='calculate_button' class="btn btn-outline-
primary col-4" disabled>Обчислити</button>
			<button type="button" id='download_button' class="btn btn-outline-
secondary col-4"
				disabled>Завантажити</button>
			<button type="button" id='refresh_button' class="btn btn-outline-success
col-4" disabled>Оновити</button>
		</div>
		<div id="button"></div>
		<div id="button_1"></div>
		<div id="button_2"></div>
	</div>
	<pre id="out"></pre>
	<br />
	<script src="cpexcel.js"></script>
	<script src="shim.js"></script>
	<script src="jszip.js"></script>
	<script src="xlsx.js"></script>
	<script src="FileSaver.min.js"></script>
	<script src="index.js">
	</script>
</body>

</html>
```

Додаток В

Інформаційна система автоматизації планування навчального навантаження кафедри

Опис програмного коду

УКР.НТУУ”КПІ”ІМ.ІГОРЯ_СІКОРСЬКОГО_ТЕФ_АПЕПС_ТВ7186_21В 3

Аркушів 7

Київ – 2021

АНОТАЦІЯ

Задля зручності використання та збереження незалежності від технологій користувача реалізовано сервіс, що розгортається на стороні користувача та надає прикладний веб-інтерфейс для взаємодії користувача з системою.

Сервіс реалізовано мовою програмування JavaScript, з використанням допоміжних бібліотек SheetJS та FileSaverJS для обробки даних з Excel та збереження файлу відповідно. Додаток розроблено у середовищі Visual Studio Code, з використанням фреймворку Bootstrap для UI-елементів. Було використано дані з робочого навчального плану ВНЗ.

ЗМІСТ

1. ЗАГАЛЬНІ ВІДОМОСТІ..... 63
2. ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ..... 64
3. ОПИС ЛОГІЧНОЇ СТРУКТУРИ..... 65
4. ВИКОРИСТАНІ ТЕХНІЧНІ ЗАСОБИ..... 66
5. ВХІДНІ ТА ВИХІДНІ ДАНІ..... 67

ЗАГАЛЬНІ ВІДОМОСТІ

Додаток А містить специфікацію додатку.

Додаток Б містить програмний код основних програмних модулів розробленої системи.

Додаток В містить опис основного файлу, в якому реалізовано алгоритм роботи програмного продукту.

Для коректної роботи додатку потрібен браузер з підтримкою JavaScript ES6.

Систему розроблено у середовищі розробки Visual Studio Code мовою програмування JavaScript.

ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

Інформаційна система автоматизації планування навчального навантаження кафедри має такі функціональні можливості:

1. Завантаження файлу з робочим навчальним планом у додаток
2. Можливість ввести кількість потоків для кожного курсу
3. Можливість ввести кількість підгруп для групи, у якій кількість студентів більша за 15
4. Можливість завантажити файл з навантаженням

ОПИС ЛОГІЧНОЇ СТРУКТУРИ

Основним типом даних, що використовується у програмі, є масив. Через особливості обраної бібліотеки вся інформація, зчитана з файлу з робочим навчальним планом, введена та обчислена додається до масиву відповідної дисципліни.

Масив з дисциплінами в свою чергу формує масив масивів, з якого формується Excel файл з навантаженням.

ВИКОРИСТАНІ ТЕХНІЧНІ ЗАСОБИ

Для користування системою користувач потребує браузер з підтримкою JavaScript ES6 та папку з проектом, розміщену локально або на сервері кафедри.

Якщо користувач відкриває додаток локально, а не з серверу кафедри, то він не потребує підключення до мережі Інтернет.

ВХІДНІ ТА ВИХІДНІ ДАНІ

Вхідними даними для роботи системи є:

1. Робочий навчальний план
2. Кількість потоків для кожного курс
3. Кількість підгруп для груп, де кількість студентів перевищує 15

Результатом роботи системи є Excel файл з навчальним навантаженням кафедри.