МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХЕРСОНСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
Факультет комп'ютерних наук, фізики та математики
Кафедра інформатики, програмної інженерії та економічної кібернетики

СТВОРЕННЯ ПРОГРАМНОГО МОДУЛЯ РОЗВ'ЯЗАННЯ ФІЗИЧНИХ ЗАДАЧ (ФІЗИЧНИЙ КАЛЬКУЛЯТОР) ДЛЯ ПРОГРАМНИХ СИСТЕМ НАВЧАЛЬНОГО ПРИЗНАЧЕННЯ

Кваліфікаційна робота (проєкт)

на здобуття ступеня вищої освіти "магістр"

Виконав: здобувач 2 курсу 241М групи
Спеціальності 121 Інженерія програмного забезпечення
Освітньо-професійної програми
«Інженерія програмного забезпечення»
другого (магістрського) рівня освіти
Нерода Дмитро Олександрович
Керівники: доктор фізико-математичних наук, професор
Львов Михайло Сергійович
кандидат фізико-математичних наук, доцент
Єрмолаєв Вадим Анатолійович
Рецензент: кандидат педагогічних наук, доцент
Таточенко Володимир Іванович.

Херсон – 2021

ЗМІСТ

ВСТУП

Актуальність. Електронні гаджети все частіше і активніше застосовуються у всіх сферах нашого життя, у тому числі й у навчанні. Вони як розвивають та роблять зручнішими вже існуючі практики і засоби навчання, так і надають нові. Найголовніша можливість електронних гаджетів – це вирішення майже будь-якої поставленої задачі чи проблеми за допомогою знаходження потрібного програмного забезпечення або, якщо такого немає або існуюче не влаштовує – створення власного.

Однією з областей науки, використання гаджетів та відповідного програмного забезпечення у навчанні якої є найбільш доцільним та ефективним, є фізика. При вивченні цього предмета у школі або університеті використовуються такі види занять, як лекції, практичні роботи, лабораторні роботи та такі типи перевірок знань, як тестування, повне вирішення задачі, розгорнута відповідь на теоретичне питання. І якщо учнями та викладачами будуть використовуватись відповідні гаджети з відповідним програмним забезпеченням, то заняття можуть буди не тільки більш ефективними але й з’являться нові можливості, які раніше були недоступні. Одним з прикладів таких можливостей можуть бути спеціальні лабораторні роботи, проведення яких в аудиторії може бути небезпечним (наприклад, демонстрація вибуху) для життя або взагалі неможливим (наприклад, ядерний синтез). А на екрані комп’ютера можна буде все побачити на власні очі без ризику для життя та здоров’я і навіть самому керувати процесом.

Отже, ми з’ясували, що за допомогою гаджетів програмним забезпеченням навчального призначення можна значно підвищити якість навчання фізики. Тому має сенс створити відповідне програмне

забезпечення. Його можна реалізувати як у якості переліку окремих застосунків, так і в якості програмної системи, яка буде складатися з модулів.

Однією з найважливіших частин навчання фізиці є вирішення фізичних задач. І якщо пошук необхідних формул та ходу рішення – це повністю робота учня, то безпосередні обчислення та знаходження відповіді можна спростити завдяки використанню відповідного програмного забезпечення для обчислення. У фізиці у багатьох випадках результатами обчислень є дуже маленькі або великі числа, періодичні дроби тощо. Окрім цього, перед обчисленням необхідно перевести усі одиниці вимірювання до одного виду. Все це збільшує як час обчислення, так і вірогідність припуститись помилки та отримати невірну відповідь навіть за правильного ходу рішення. Через це у більшості випадків при вирішенні задач використовують звичайний калькулятор у смартфоні, проте це універсальний інструмент, який не вирішує всіх проблем. Тому існує необхідність створення ефективного спеціалізованого програмного інструменту, який буде полегшувати розв’язання задач з фізики шляхом обчислення фізичних виразів.

Отже, **метою** даної роботи є створення програмного модуля “Фізичний калькулятор”.

Тип програмного модуля – Web-застосунок. Це дозволить охопити майже усі види електронних гаджетів, такі як: персональні комп’ютери, ноутбуки, смартфони, планшети, смарт-годинники тощо. Окрім цього, апаратні вимоги до цих пристроїв будуть мінімальні. Єдиний недолік – буде потребуватись постійне підключення до мережі “Інтернет”.

Об’єкт дослідження – технології розроблення програмних систем навчального призначення.

Предмет дослідження – технології розроблення програмного модуля «Фізичний калькулятор».

Для досягнення зазначеної мети необхідно виконати наступні **завдання:**

1. Спроєктувати модель та можливості програмного модуля.
2. Спроєктувати структуру бази даних ПМ.
3. Вибрати інструменти та технології для розроблення; ознайомитись та прочитати літературу з них.
4. Розробити бекенд частину ПМ.
5. Створити дизайн ПМ.
6. Розробити фронтенд частину ПМ.
7. Протестувати ПМ.

Апробація результатів дослідження: результати з'ясування актуальності розроблення програмних систем для вивчання фізики вийшли в таку роботу автора, як "Електронні ресурси як інструмент для вивчення фізики", яка була подана до наукового видання "Магістерські студії" (Херсонський державний університет).

Структура й обсяг роботи. Кваліфікаційна робота складається з вступу, трьох розділів, висновків, списку використаних джерел та одного додатка. Повний обсяг кваліфікаційної роботи складає 49 сторінок.

РОЗДІЛ 1
ПРОЄКТУВАННЯ ПРОГРАМНОГО МОДУЛЯ

1.1 Педагогічні цілі програмного модуля

У вступі ми з'ясували, що існує потреба в створенні спеціального веб-застосунку, який буде використовуватися учнями або студентами при розв'язанні задач з фізики.

На основі цього буде розроблений і створений програмний модуль "Фізичний калькулятор", педагогічними цілями якого є:

1. Надати учням можливість навчатися у будь-якому місці та у будь-який час.
2. Підвищити ефективність навчання за рахунок спрощення рутинних дій та обчислень.
3. Полегшення процесу розв'язку нових задач за рахунок можливості перегляду раніше розв'язаних задач.

1.2 Аналіз існуючих аналогів

Тепер необхідно провести аналіз аналогічного або схожого програмного забезпечення. Такий аналіз допоможе з'ясувати, чи потрібно взагалі створювати новий застосунок, уникнути або мінімізувати знайдені недоліки та залишити і розвинути переваги.

1. "Omnicalculator" [1]

Це веб-сайт, на якому є калькулятори з різних наук та дисциплін, у тому числі і з фізики. На головній сторінці відображаються блоки з темами та рядок пошуку необхідного калькулятора. Якщо натиснути на блок "Physics", то відкривається сторінка із блоками конкретних формул, які згруповані за певними тематиками. Якщо після цього натиснути на блок із

формулою, наприклад, вільного падіння (“Free Fall Calculator”), то відкриється сторінка, скріншот якої на рисунку 1.1, на якій відображається блок із полями для вводу значень змінних, які використовуються при розрахунках, селектори для вибору одиниць вимірювання, теоретичний матеріал щодо формули і її тематики та калькулятори зі схожих тем. Після вводу значень відомих змінних, останнє невідоме розраховується та виводиться автоматично.

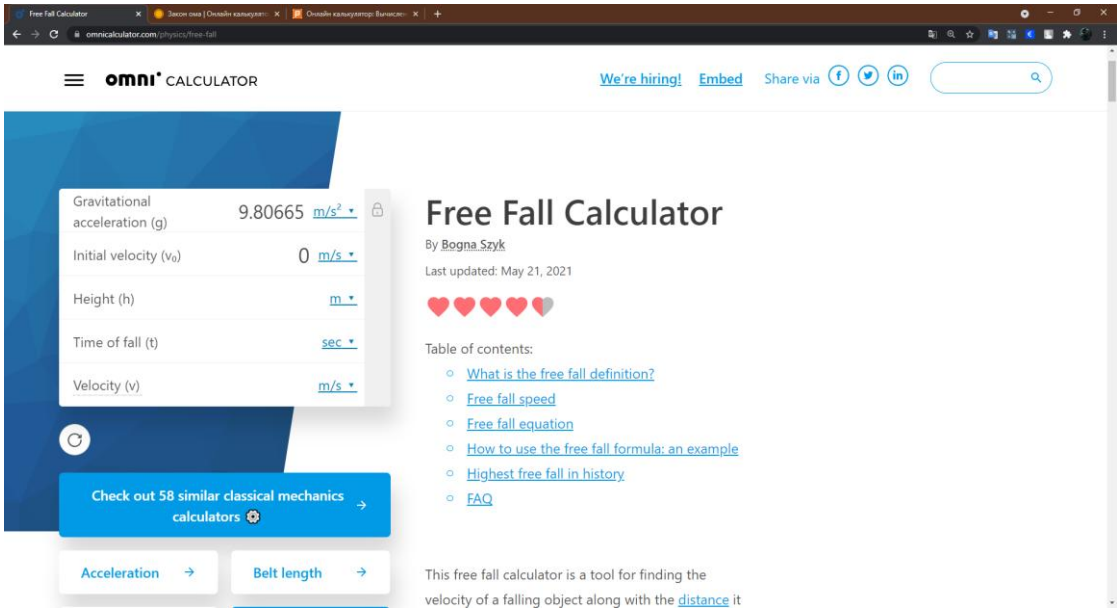

Рисунок 1.1 – Приклад фізичного калькулятора сервісу “Omnicalculator”

Переваги застосунку:

1) Наявність інтуїтивно зрозумілого інтерфейсу.
2) Адаптивний дизайн для різних пристроїв.
3) Наявність теоретичного матеріалу з обраної теми.

Недоліки:

1) Наявність обмеженої кількості одиниць вимірювання, відсутність багатьох префіксів.
2) Відсутність можливості обчислення власної формули.

3) Відсутність нативної реалізації розв’язку складної задачі, у рішенні якої більше однієї формули.
4) Відсутність української або російської локалізації.

2. “AllCalc” [2]

Це також веб-ресурс, на якому є калькулятори з різних дисциплін та тем. На головній сторінці треба обрати пункт “Физика” у стовпці “Учеба и наука”. Після цього відкривається список із темами та список із різними фізичними величинами для конвертування.

Якщо натиснути на одну з тем, наприклад “Закон Ома”, то відкриється сторінка, скріншот якої на рисунку 1.2, із блоками, кожен з яких позначає формулу з обраної теми. У цьому блоці знаходиться текстовий опис формули и поля для вводу значень змінних, кнопка “Расчитать” і поле для відповіді. Після того, як ввести всі необхідні значення змінних і натиснути на кнопку “Рассчитать”, відповідь з’явиться у відповідному полі.

Якщо ж зайти в на сторінку з конвертацією, наприклад, “Единицы измерения скорости”, то з’являться поля з деякими одиницями вимірювання з обраної теми, налаштування кількості знаків після коми та налаштування десяткового і розрядного роздільника. При введенні значення в одне з полів автоматично розраховуються та виводяться значення всіх інших.

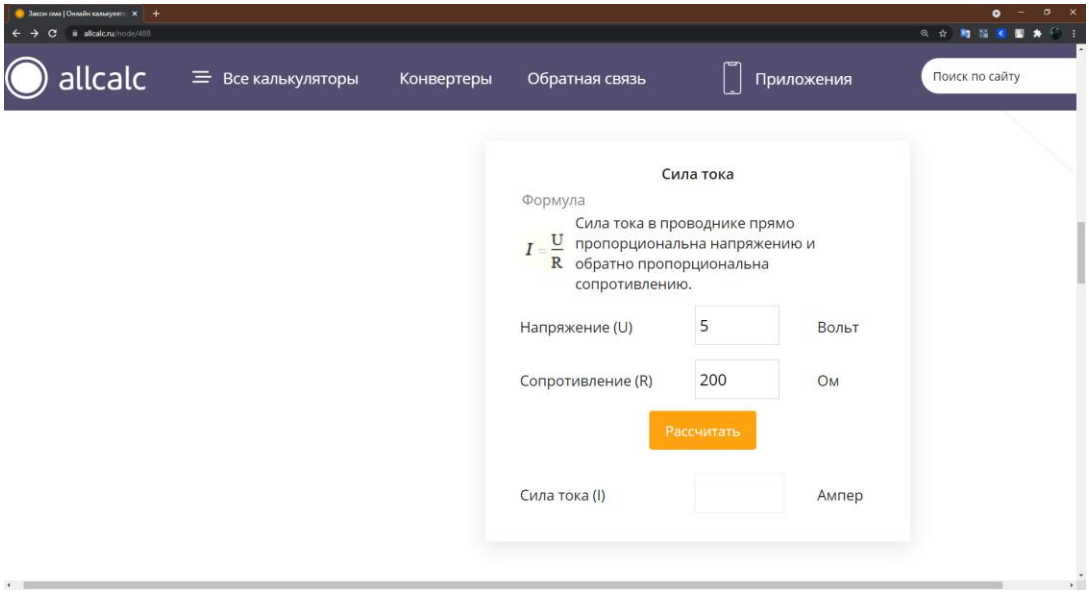

Рисунок 1.2 – Приклад фізичного калькулятора сервісу “AllCalc”

Переваги застосунку:

1) Наявність інтуїтивно зрозумілого інтерфейсу.
2) Адаптивний дизайн для різних пристроїв.
3) Наявність конвертера для одиниць вимірювання.
4) Наявність налаштувань точності округлення.

Недоліки:

1) При розрахунку формул треба вводити значення лише у зазначених одиницях вимірювання.
2) Відсутність можливості обчислення власної формули.
3) Відсутність нативної реалізації розв’язку складної задачі, у рішенні якої більше однієї формули.
4) Обмежена кількість одиниць вимірювання та префіксів при конвертуванні.
5) Мала кількість формул для розрахування.

3. “PLANETCALC” [3]

Цей сайт, як і обидва попередніх, має калькулятори з багатьох тем, серед яких є і фізика. На головній сторінці у списку тематик треба обрати

фізику, потім необхідний калькулятор, наприклад “Закон Ома”. Після цього відкриється сторінка, скріншот якої на рисунку 1.3, на якій буде відповідна формула, поля для вводу напруги і опору, налаштування кількості знаків після коми, поле для відображення відповіді, кнопка “Рассчитать” та калькулятори зі схожих тем. Розрахунок проходить автоматично після заповнення необхідних полів, або при натисканні на кнопку.

Окрім цього, на сайті присутні конвертери одиниць вимірювання із подібним інтерфейсом.

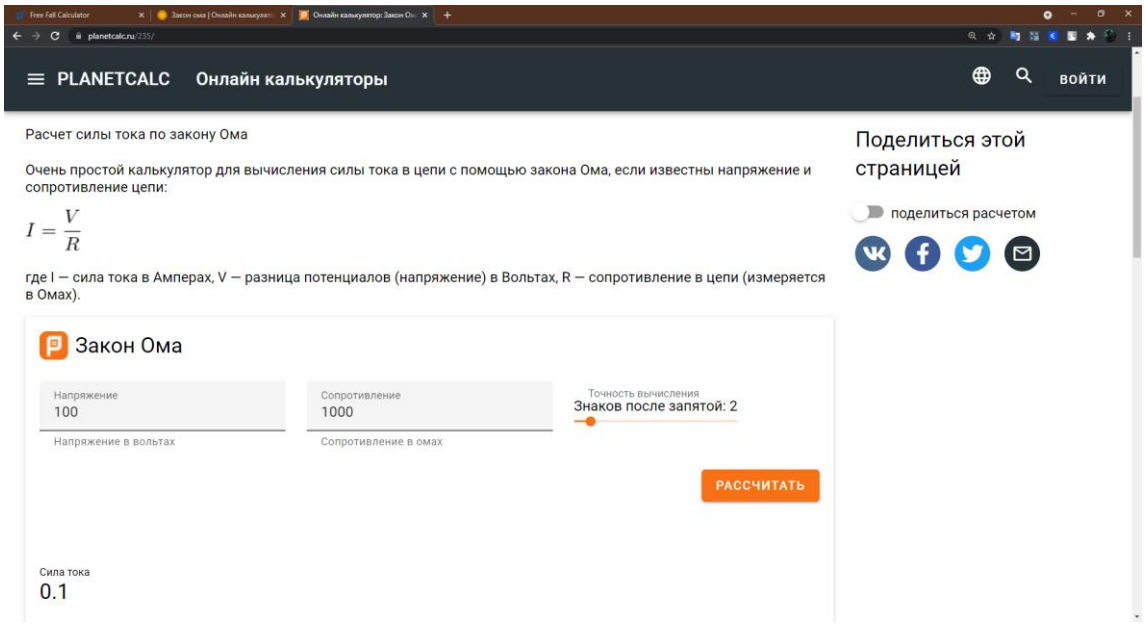

Рисунок 1.3 – Приклад фізичного калькулятора сервісу “PlanetCalc”

Переваги застосунку:

1) Наявність інтуїтивно зрозумілого інтерфейсу.
2) Адаптивний дизайн для різних пристроїв.
3) Наявність конвертера для одиниць вимірювання.
4) Наявність налаштувань точності округлення.

Недоліки:

1) При розрахунку формул треба вводити значення лише у зазначених одиницях вимірювання.

2) Відсутність можливості обчислення власної формули.
3) Відсутність нативної реалізації розв’язку складної задачі, у рішенні якої більше однієї формули.
4) Обмежена кількість одиниць вимірювання та префіксів при конвертуванні.
5) Мала кількість формул для розрахування.

Як ми бачимо, в усіх аналогічних веб-застосунках є три основних недоліка: відомі змінні треба вводити лише у строго зазначених одиницях вимірювання, можливість розрахунку тільки тих формул, які внесені до бази даних застосунку та неможливість розв’язання складної задачі, в якій потребується використання двох або більше формул. Одним із факторів, через які виникли ці недоліки є те, що всі три ресурси є універсальними, тобто на них присутні калькулятори із інших тем та галузей, на які витрачаються ресурси розробників та адміністрації.

Зі сказаного раніше випливає, що жоден з веб-застосунків не покриває всі потреби учнів та студентів, які розв’язують фізичні задачі. Тому доречніше буде використовувати спеціалізоване програмне забезпечення, яке повністю або частково усуне зазначені недоліки. Одним з варіантів такого ПЗ буде веб-застосунок “Фізичний калькулятор”, який є результатом цієї кваліфікаційної роботи. Далі ми визначимо функціональні вимоги, модель та компоненти програмного модуля, який ми розробляємо, завдяки яким він буде зручним та ефективним у використанні і зможе реалізувати зазначені у попередньому пункті педагогічні цілі.

1.3 Вимоги та модель програмного модуля

Перед початком розроблення необхідно зазначити вимоги програмного модуля “Фізичний калькулятор”. Зважаючи на все, що було написано у вступі, першому та другому пункті першого розділу можна

скласти наступний перелік функціональних вимог, які будуть у веб-застосунка, що розробляється:

1. Обчислення як значення, так і одиниці вимірювання розв’язку фізичних формул.
2. Конвертація усіх одиниць вимірювання до їх аналогів у міжнародній системі одиниць.
3. Можливість реєстрації та авторизації користувача за допомогою логіна та паролю.
4. Можливість зберігання розв’язаної задачі для авторизованого користувача.

Для більш детального опису функцій скористаємося діаграмою прецедентів, або use-case діаграмою (Рис. 1.4). Вона наочно продемонструє, які типи користувачів є у веб-застосунку, їх можливості та обмеження.

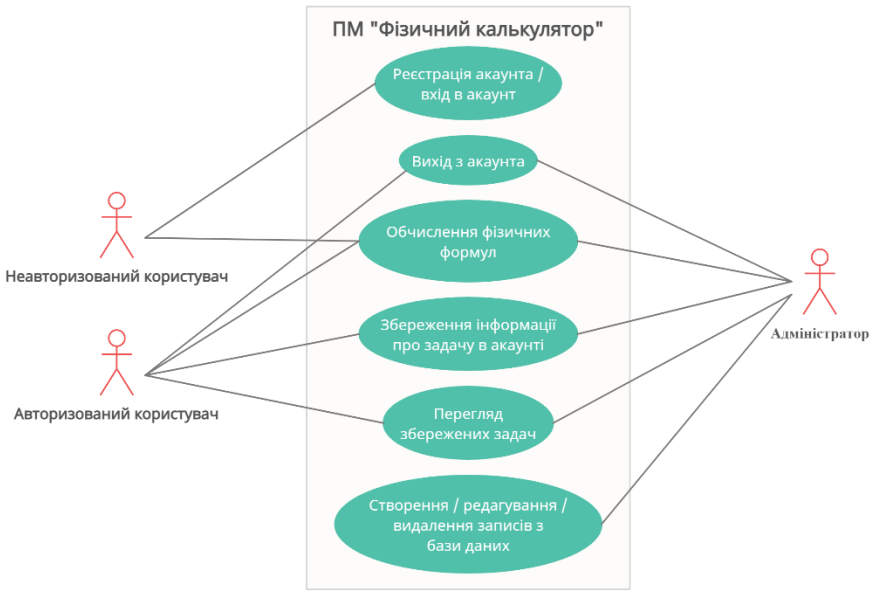

Рисунок 1.4 – Use-case діаграма ПМ “Фізичний калькулятор”

Тепер окреслимо модель та компоненти “Фізичного калькулятора”, розберемо кожну задачу більш детально та зробимо необхідні уточнення стосовно них.

Як вже було зазначено раніше, програмний модуль, який ми розробляємо, буде реалізований у вигляді веб-застосунка. Тому його складовими будуть:

1. Інтерфейс користувача
2. Сервер
3. База даних
4. Панель адміністрування

Схема взаємодії компонентів представлена на рисунку 1.5. Розглянемо її більш детально.

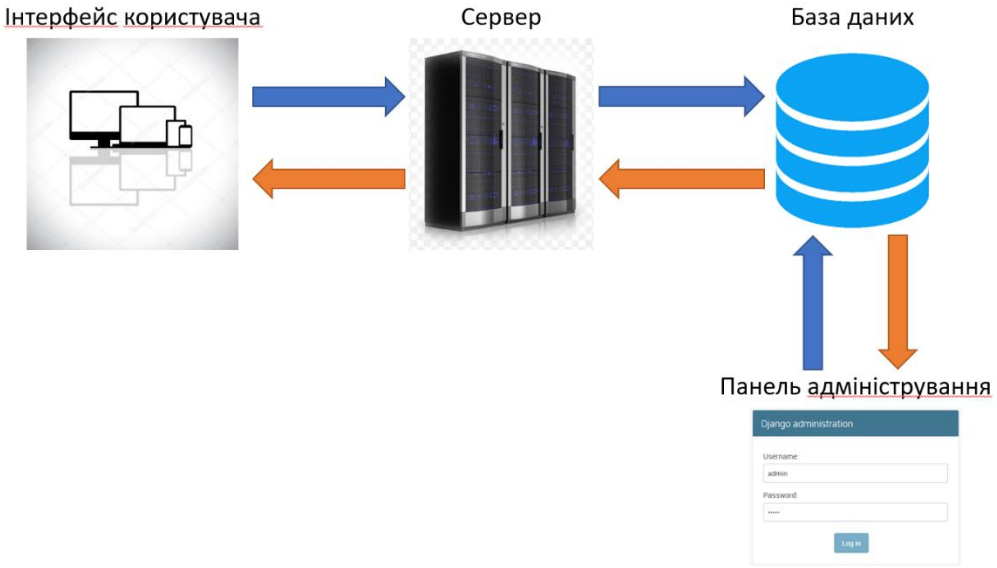

Рисунок 1.5 – Взаємодія компонентів ПМ “Фізичний калькулятор”

1) Інтерфейс користувача

Інтерфейс користувача представляє собою веб-сторінки, які відображаються у браузері і з якими безпосередньо взаємодіє сам користувач. У нашому застосунку таких сторінок буде три: головна сторінка із калькулятором та списком збережених задач, сторінка для реєстрації нового акаунта та сторінка для авторизації вже існуючого акаунта.

Головна сторінка буде складатися з наступних частин: поле для вводу відомих змінних та формул, поле для відображення відповіді, і, якщо користувач авторизований, список збережених задач, поле для вводу імені задачі та текстового опису задачі.

Сторінка для реєстрації включає в себе блок із полями для вводу імені користувача, пароля та підтвердження пароля.

Сторінка для авторизації складається з блоку, у якому поля для вводу імені користувача та паролю.

2) Сервер

Сервер – це комп’ютер, на якому працює програма, що обробляє отримані від користувача дані. У нашому випадку сервер приймає дані, що надходять від користувача, проводить їх валідацію, обчислює та повертає відповідь. Окрім цього, для валідації та обчислення одиниць вимірювання сервер отримує відповідну інформацію з бази даних.

3) База даних

База даних програмного модуля складається з 3 таблиць.

У першій таблиці зберігається інформація про одиниці вимірювання, а саме: фізичне явище, повна і скорочена назва одиниці вимірювання, належність до міжнародної системи одиниць, вигляд у міжнародній системі одиниць, множник при конвертуванні у основні одиниці міжнародної системи одиниць та перелік сумісних префіксів.

У другій таблиці зберігається інформація про префікси: повна та скорочена назва префіксу і його множник.

Остання таблиця містить у собі дані про збережені задачі: ім’я користувача, назва, текстовий опис, умову і відповідь задачі.

Більш наглядно поля і структура бази даних відображені на діаграмі класів (Рис 1.6).

Рисунок 1.6 – Діаграма класів бази даних

4) Панель адміністрування.

Панель адміністрування – це інтерфейс, за допомогою якого користувач із правами адміністратора має доступ до бази даних, може записувати, редагувати та видаляти записи з неї. Він складається з головної веб-сторінки, сторінки авторизації, сторінок зі списками об’єктів таблиць та сторінок створення/редагування об’єктів таблиць бази даних.

На головній сторінці буде список таблиць, які є у базі даних та список останній дій, які проходили в ній.

Сторінка самої таблиці буде містити у собі список об’єктів, які зберігаються в ній, у вигляді таблиці та кнопку для додавання об’єкта.

На сторінці для додавання об’єкта у таблицю бази даних будуть відповідні поля цього об’єкта та кнопка для збереження.

Сторінка редагування вже існуючого запису у базі даних буде схожою на попередню сторінку, з’явиться тільки кнопка для видалення об’єкта з бази.

1.4 Алгоритм розв’язання фізичних задач у програмному модулі

Перед початком створення програмного модуля необхідно спроєктувати принцип роботи його основного функціоналу, а саме –

розв’язання фізичних задач. Для цього ми окреслимо, які функції програми для цього знадобляться, які оператори, операнди та їх канонічні форми будуть реалізовані, і в наприкінці продемонструємо сам алгоритм розв’язання.

Перелік функцій програми:

1. Функція для обчислення числового значення виразу. Вона спрощує та обчислює вираз, у якому змінні були замінені на їх кількісні значення.
2. Функція для розрахунку одиниці вимірювання виразу. Вона спрощує та обчислює вираз, у якому змінні були замінені на їх одиниці вимірювання.
3. Функція для переведення одиниць вимірювання до їх аналогів у міжнародній системі одиниць. Вона замінює одиниці вимірювання, які є в умові задачі, на їх аналоги у міжнародній системі одиниць (якщо вони одразу не були у цій системі). Окрім цього, функція також змінює і числові значення змінних та виразів, домноживши їх на відповідний коефіцієнт (наприклад, за наявністю префікса (см=10^2 м) або зв’язних одиниць вимірювання (хв=60 с)).

Тепер перерахуємо валідні операнди та оператори, які можуть оброблюватися нашим програмними модулем:

1. Раціональні числа у вигляді десяткових дробів.
2. Змінні у вигляді словосполучень з латинських літер та необов’язковою цифрою у кінці.
3. Одиниці вимірювання у вигляді словосполучень з кирилістичних літер.
4. Знаки операцій: “+”, “-”, “*”, “/”, “^”, “**”, “(”, “)”, “=”.

Канонічні форми:

1. Знак степені – “^”.
2. Одиниця вимірювання – одиниця вимірювання у міжнародній системі одиниць.

Тепер перейдемо безпосередньо до алгоритму розв’язування задачі. Для того, щоб розв’язати задачу, користувач повинен написати повний хід рішення зверху-вниз. Тобто шукана змінна повинна бути першим рядком у ході рішення і змінні у будь-якому виразі повинні бути або відомі з умови задачі, або їх можна знайти з формули, яка стоїть нижче.

Отже, якщо користувач ввів усе правильно та натиснув кнопку “Вирішити”, то для розв’язання будуть потрібні 2 алгоритми: алгоритм спрощення відомої з умови змінної і алгоритм обчислення невідомої змінної.

Перший алгоритм:

1. З отриманих рядків із відомими з умови змінними обирається найнижчий неспрощений.
2. Він поділяється на назву змінної, її значення та одиницю вимірювання.
3. Проводиться валідація значення та одиниці вимірювання.
4. Одиниця вимірювання конвертується у її аналог, який складається з основних одиниць вимірювання міжнародної системи одиниць.
5. Значення змінної домножиться на множник, який виник у результаті конвертування одиниці вимірювання і спрощується.
6. Результат заноситься до словнику із вже відомими або знайденими змінними.

Другий алгоритм:

1. З отриманих рядків із невідомими змінними обирається найнижчий.

2. Він поділяється на змінну та формулу для знаходження (ліву і праву частину).
3. Проводиться валідація правої частини.
4. Права частина поділяється на 2 вирази: у першому змінні замінені на їх значення, у другому – на їх одиниці вимірювання.
5. Проводиться обчислення та спрощення кожної частини.
6. Після цього додатково числове значення домножиться на множник, який виник у частині із одиницями вимірювання.
7. Результат заноситься до словнику із вже відомими або знайденими змінними.

Після того, як усі рядки були оброблені, користувачу повертається значення та одиниця вимірювання останньої знайденої невідомої змінної.

У висновку до першого розділу можна зазначити, що наш програмний модуль буде побудований у вигляді веб-застосунку з традиційною клієнт-серверною архітектурою, базою даних та панеллю адміністрування. Вимоги, функції та модель веб-застосунку були спроєктовані на основі його цілей та аналізу переваг та недоліків аналогічного програмного забезпечення. А спроєктований алгоритм реалізації основної функції дозволить швидко реалізувати його у програмному коді.

РОЗДІЛ 2
ІНСТРУМЕНТИ ДЛЯ РОЗРОБЛЕННЯ ПРОГРАМНОГО МОДУЛЯ

Перед початком розроблення необхідно вибрати інструменти та технології, які ми будемо використовувати для створення програмного модуля. Тому у цьому розділі ми окреслимо, які засоби розроблення нам знадобляться, для чого вони необхідні та їх переваги конкретно для нашого веб-застосунка.

2.1 Фреймворк

Фреймворк – це програмна платформа, яка слугує каркасом для усіх складових програмного продукта. Він полегшує розробку програмного забезпечення шляхом автоматичного створення базових компонентів та зв’язків між ними. Окрім цього, фреймворк також надає інструментарій для зміни базових та інтегрування нових модулей у створений “каркас”. [5]

Так як наш програмний модуль розробляється у вигляді веб-застосунка, то нам необхідний саме веб-фреймворк. Таким фреймворком був обраний Django. Нижче буде наведено основну інформацію про нього та причини його вибору.

Django – це веб-фреймворк, побудований на архітектурі MVT, який використовує мову програмування Python. [4] Перша публічна версія вийшла 21 липня 2005 року. Розповсюджується за ліцензією “ 3-clause BSD”, за якої можливе необмежене безкоштовне користування та розповсюдження програмним забезпеченням за умови вказівки повідомлення про авторські права та відмови від гарантії за ліцензією. [6]

Перейдемо тепер до огляду архітектури Django, тобто MVT. MVT розшифровується як “Model-View-Template”, тобто “Модель-Вигляд-

Шаблон”. Схема роботи Django, яка забезпечується цією архітектурою, зображено на рисунку 2.1.

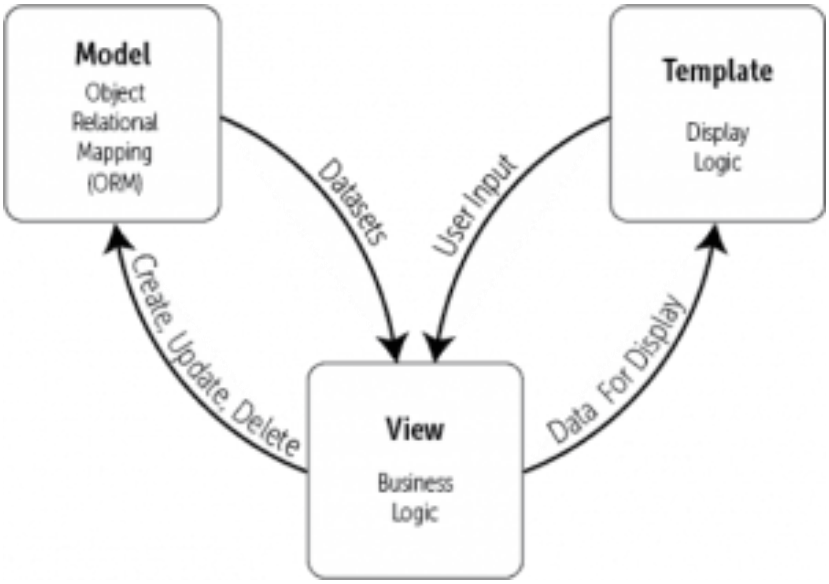

Рисунок 2.1 – Взаємодія компонентів застосунку на Django фреймворку

Як ми бачимо, “Template” забезпечує відображення отриманих від “View” даних та передачу інформації, отриману від користувача, до того ж самого “View” для її подальшої обробки. У свою чергу, “View” передає необхідну інформацію у вигляді об’єктів до “Model”, яка, за допомогою ORM (Object Relation Mapping, або об’єктно-реляційне відображення) перетворює отримані об’єкти на записи бази даних. Якщо ж навпаки, є необхідність отримати дані з бази даних, то за допомогою ORM проходить конвертація записів бази даних у об’єкти та передача їх до “View”.

З написаного випливає, що “Template” – це фронтенд частина, а “View” та “Model” – бекенд частина, при чому “View” відповідає за бізнес логіку, а “Model” – за роботу із базою даних.

Тепер окреслимо переваги веб-фреймворку Django.

Першою перевагою є наявність об’єктно-реляційного відображення. Воно значно спрощує роботу із базою даних, дозволяючи замість прямих SQL запитів використовувати класи та об’єкти, створених за допомогою

мови програмування Python (див. рис. 2.2). До того ж, значно підвищується безпека при роботі з базою даних та виключаються деякі види вразливостей (наприклад “SQL-ін’єкції”). Окрім цього, ORM у Django офіційно підтримує 5 реляційних систем управління базою даних (PostgreSQL, MariaDB, MySQL, Oracle, SQLite). [7] А інші системи, що не підтримуються офіційно, у тому числі й нереляційні, майже завжди мають свою бібліотеку для забезпечення сумісності із ORM Django.

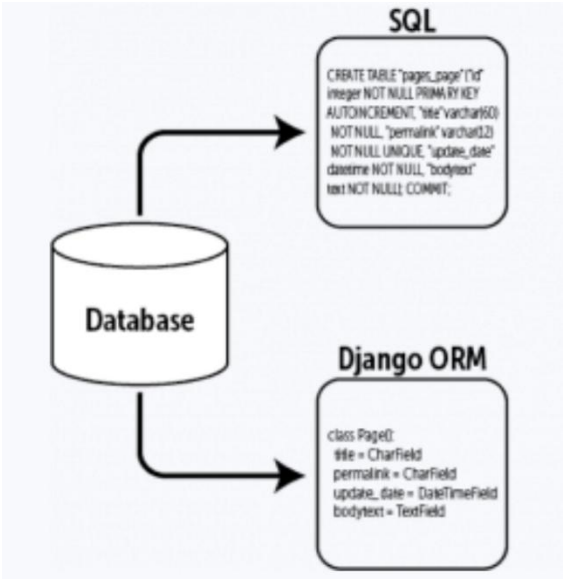

Рисунок 2.2 – Схема взаємодії бази даних та ORM Django

Друга перевага – наявність вбудованої панелі адміністратора. Вона є важливим інструментом для роботи із базою даних веб-застосунку і Django пропонує її зручну та функціональну реалізацію. До того ж, за необхідності, вбудовану панель адміністрування можна модифікувати під задачі конкретного проєкту.

Третя перевага – велике товариство та детально прописана документація. Обидва цих фактори дозволяють заощадити багато часу як при безпосередній розробці програмного забезпечення, так і при його тестуванні. У першому випадку документація та товариства допоможе швидко знайти способи правильної реалізації необхідних функцій, а у

другому – дозволить точно з’ясувати, яка помилка сталася, причини її виникнення та шляхи усунення.

Четверта перевага – масштабованість. Django побудована таким чином, що проєкти, створені на ньому, можуть буди дуже легко розширені за допомогою додаткових модулей та функцій, які не потребують зміни вже існуючого коду.

Четверта перевага – використання мови програмування Python.

П’ята перевага – наявність вбудованої системи управління базою даних SQLite.

Останні дві переваги будуть описані у наступних пунктах із інструментами для розроблення.

2.2 Мова програмування

Використання фреймворку для розроблення програмного забезпечення накладає обмеження на мову програмування, яку можна використовувати для бізнес логіки. У Django такою мовою є Python. Нижче буде наведено основну інформацію про нього та його переваги для нашого проєкту.

Python – це об’єктно-орієнтована мова програмування високого рівня, що інтерпретується та має динамічну типізацію. Перший публічний випуск пройшов 20 лютого 1991 року. Розповсюджується за ліцензією “Python Software Foundation”, яка дозволяє безкоштовно використовувати та розповсюджувати навіть модифіковані версії без вихідного коду. Основна філософія та концепція цієї мови – читаність коду. Це забезпечується завдяки обов’язковим відступам та мовним конструкціям. Окрім об’єктно-орієнтованої парадигми програмування підтримує також процедурну та функціональну.

Перша перевага – легкість у написанні коду. Як вже було написано раніше, програмний код написаний на мові Python є високу читаність завдяки своєму синтаксису, який є інтуїтивно зрозумілим та легким у вивченні. Приклад коду наведений на рисунку 2.3.

```
# Sum of natural numbers up to num

num = 16

if num < 0:
   print("Enter a positive number")
else:
   sum = 0
   # use while loop to iterate until zero
   while(num > 0):
       sum += num
       num -= 1
   print("The sum is", sum)
```

Рисунок 2.3 – Приклад коду на мові програмування Python

Друга перевага – наявність великої кількості вбудованих та сторонніх бібліотек та модулів. Це значно спрощує написання коду, бо замість написання власної реалізації якогось функціоналу можна взяти необхідну бібліотеку, у якій вже буде необхідні функції.

Третя перевага – товариство і документація. За даними професійної технічної організації IEEE, Python у 2021 році є найпопулярнішою мовою програмування. [8] І у попередні роки він також був у топ 3 найпопулярніших мов. Тому не дивно, що Python має велике товариство, яке дуже допомагає при розробці будь-якого продукту тим, що можна майже завжди знайти інформацію щодо помилки, або реалізації якогось функціоналу в мережі інтернет у людей, які вже стикалися з подібним. Окрім цього, Python має детально опрацьовану та структуровану

документацію. Все це робить розроблення на мові програмування Python дуже легким та зручним, і значно економить час.

2.3 СУБД

Ще один компонент нашого веб-застосунку – це база даних. Зважаючи на специфіку даних, які необхідно зберігати у ній, доцільно використовувати реляційну базу даних. Проте для того, щоб взаємодіяти із базою даних, необхідна СУБД (система управління базою даних). Django надає таку систему – SQLite.

Дата виходу першої публічної версії – 17 серпня 2000 року. SQLite є суспільним надбанням, тобто на неї не розповсюджується жодна ліцензія чи авторське право і можна вільно використовувати, модифікувати та розповсюджувати її код.

Ключова перевага цієї системи управління базою даних є те, що вона вбудована у кінцеву програму, коли більшість інших систем потребують розгортання і працюють за схемою “клієнт-сервер”. Це дозволяє створити та використовувати базу даних, маючи лише один файл з розширенням .db та бібліотеку для роботи з нею. Django надає обидва цих компонента. Приклад схеми файлу бази даних відображена на рис. 2.4.

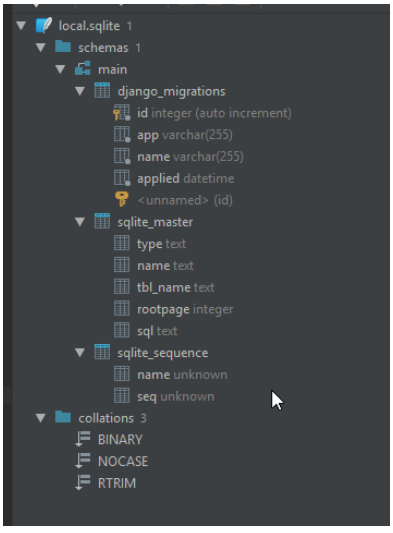

Рисунок 2.4 – Приклад структури SQLite бази даних

Друга перевага – продуктивність. SQLite завантажує лише необхідні дані, а не дані з усього файлу. До того ж, при редагуванні записів у базі даних, проходить перезапис тільки тих частин файлу, які змінюються. І на останок, читання і запис у базі даних SQLite на 35% швидше, аніж ті ж самі операції у файловій системі.

Третя перевага – надійність. Дані у базі даних оновлюються неперервно, що знижує ризик втрати інформації при збої (у роботі самої бази або живлення сервера). А створення бекапу є максимально простим – достатньо створити копію одного файлу. [9]

2.4 Фронтенд

Окрім раніше зазначених інструментів, будуть використані ще HTML, CSS та JavaScript. Це одні з основних технологій, які використовуються у мережі “Інтерент” і жоден сучасний сайт не може обійтись без них. Ці інструменти потрібні для створення фронтенд частини веб-застосунка.

HTML – це стандартизована мова розмітки для веб-сторінок у браузері. Браузер інтерпретує код з HTML-сторінки у інтерфейс, який бачить користувач на екрані монітора. Проте чистий HTML код майже ніколи не використовується. Для додаткового налаштування зовнішнього вигляду використовується CSS. [10]

CSS – це мова таблиць стилей. Вона використовується для опису кольору, шрифта, стилей елементів сторінки, яка написана за допомогою мови розмітки (у нашому випадку це HTML). Окрім цього, за допомогою CSS можна також задавати розташування елементів. [11]

```
<!DOCTYPE html>
<html>
<body>

<h1>My First Heading</h1>

<p>My first paragraph.</p>

</body>
</html>
```

Рисунок 2.5 – Приклад HTML коду

```
h1 {
  color: blue;
  background-color: yellow;
}

p {
  color: red;
}
```

Рисунок 2.6 – Приклад CSS коду

Останній інструмент, який буде необхідний для створення фронтенд частини веб-застосунка – JavaScript.

JavaScript – це об’єктно-орієнтована скриптова мова програмування. З’явилась 4 грудня 1995 року. Як правило, він використовується для налаштування поведінки веб-сторінок та асинхронного обміну даними із сервером. [12] Перше означає, що за допомогою JavaScript можна додавати, видаляти та змінювати HTML і CSS код сторінки, а друге – що JavaScript надає можливість за допомогою технології Ajax передавати необхідну інформацію на сервер або отримувати її звідти без перезавантаження сторінки.

Проте замість використання чистого JavaScript часто використовують одну з багатьох бібліотек, які засновані на ньому і спрощують написання коду. У нашому випадку ми будемо використовувати JQuery.

JQuery – це бібліотека JavaScript, створена для взаємодії HTML і JavaScript. Окрім цього, він надає функціонал для зручної роботи із Ajax. Перший публічний реліз склався 26 серпня 2006 року. Розповсюджується за ліцензією X11, тобто бібліотеку можна вільно використовувати і розповсюджувати за умовою надання тексту ліцензії разом із програмним забезпеченням, у якому вона використовується. [13] Приклад синтаксису наведений на рисунку 2.7.

```
$("#selectCulture").change(function () {
    var selectedCulture = this.value;

    if ($.cultures[selectedCulture]) {
        globalizePage(selectedCulture)
    } else {
        var globPath = "Scripts/globinfo/jQuery.glob."
            + selectedCulture + ".js";
        $.getScript(globPath, function () {
            globalizePage(selectedCulture)
        });
    }
});
```

Рисунок 2.7 – Приклад JavaScript коду

Перша перевага JQuery – це розповсюдженість. За даними сайту w3techs.com, JQuery використовується у 7,300,000 із 10,000,000 найпопулярніших веб-сайтів. Завдяки цьому у мережі “Інтернет” є багато інформації щодо можливостей бібліотеки, а також проблем, які виникають при користуванні єю. Все це дуже допомагає при розробленні програмного продукту з використанням JQuery.

Друга перевага – кросбраузерність. JQuery має низку оптимізацій та рішень, які роблять його код працездатним в усіх сучасних браузерах та вирішує проблеми сумісності із ними. Це також допомагає при розробленні веб-сайтів, бо відпадає необхідність робити спеціальні надбудови для кожного браузера і всі функції в кожному з них працюють однаково.

І третя перевага - зручність роботи. Причина цього – легкий синтаксис та зберігання коду окремо від коду веб-сторінки. Легкий синтаксис дозволяє значно зменшити кількість строк коду (у деяких випадках до п’яти раз у порівнянні із звичайним JavaScript), а відповідно і розмір файлів. А окреме зберігання коду дозволяє при внесенні змін працювати лише з JQuery кодом, тобто навігація по файлу стає набагато легшою.

РОЗДІЛ 3
СТВОРЕННЯ ПРОГРАМНОГО МОДУЛЯ

Після того, як ми обрали інструменти і технології для розроблення, можна перейти до створення програмного модуля. Для цього спочатку розглянемо інтерфейс веб-застосунка.

3.1 Інтерфейс та дизайн програмного модуля

Як ми вже зазначали у першому розділі, програмний модуль має 3 основні веб-сторінки та сторінки панелі адміністрування. Розглянемо спочатку основні сторінки:

1. *Сторінка реєстрації*

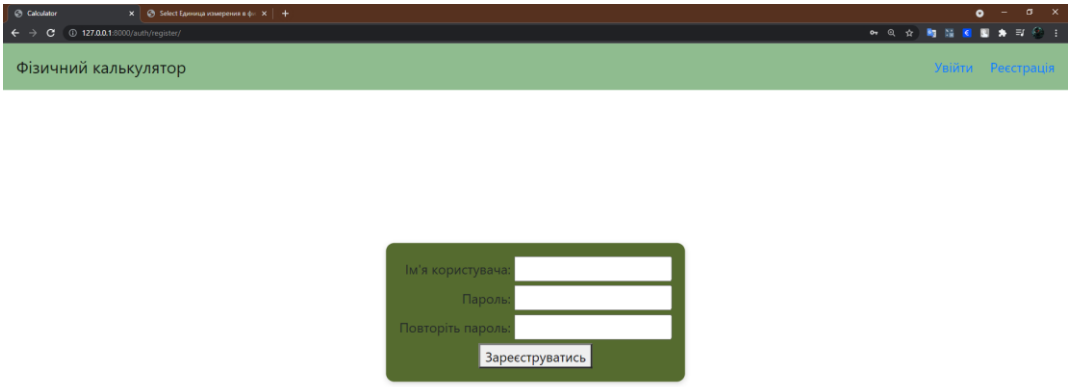

Рисунок 3.1 – Сторінка реєстрації веб-застосунку

Вона складається з навігаційної панелі та основного блока.

У навігаційниій панелі зліва відображено назву нашого веб-застосунка, а справа – посилання на сторінки авторизації та реєстрації.

У основному блоці присутня форма для реєстрації нового користувача. У цю форму входять наступні елементи:

1. Поле для введення імені користувача.
2. Поле для введення паролю.
3. Поле для повторного введення паролю.
4. Кнопки для підтвердження.

Окрім цього, у разі неправильного заповнення полів у формі буде з’являтись текстове повідомлення із роз’ясненням помилки. Приклад повідомлення у разі невідповідності паролів у відповідних полях наведено на рисунку 3.2.

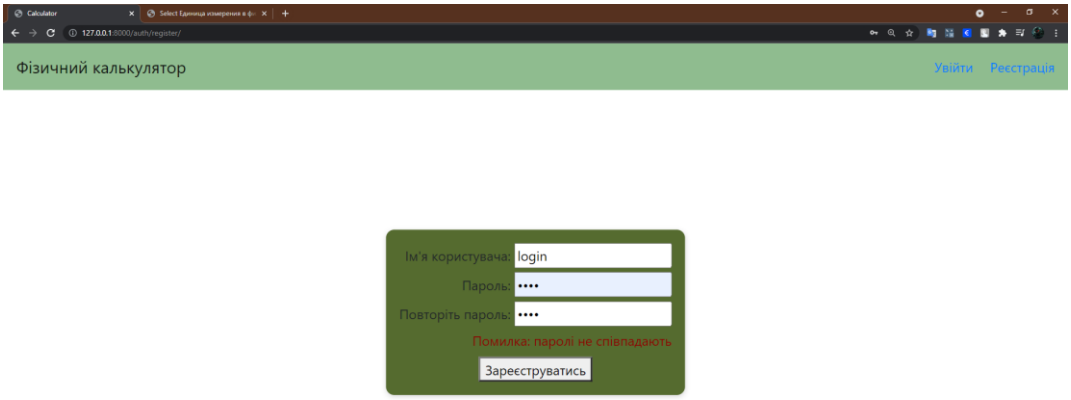

Рисунок 3.2 – Сторінка реєстрації веб-застосунку із повідомленням про помилку

Повний перелік можливих помилок:

1. Пусте поле з ім’ям користувача.
2. Пусте поле із паролем.
3. Паролі у відповідних полях не співпадають.
4. Користувач із таким ім’ям вже зареєстрований.

Якщо ж усі поля заповнені правильно і після натискання кнопки “Зареєструватись” реєстрація пройшла успішно, то користувач повернеться на головну сторінку. При цьому два посилання у навігаційній панелі замінюються на одне, у якому написане ім’я користувача у круглих дужках та слово “Вийти”

2. *Сторінка авторизації*

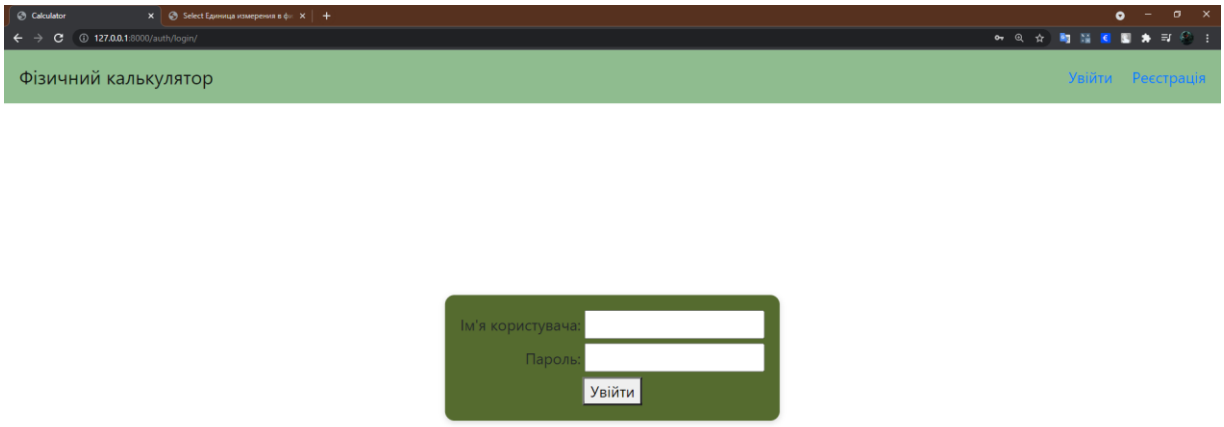

Рисунок 3.3 – Сторінка авторизації веб-застосунку

Вона також складається з навігаційної панелі та основного блоку.

Основний блок майже ідентичний основному блоку сторінки реєстрації, окрім двох відмінностей: відсутність поля для повторного введення пароля і кнопка має назву “Увійти” замість “Зареєструватись”.

Окрім цього, помилок може бути лише дві: неправильне ім’я користувача або невірний пароль. Текст повідомлення про обидві помилки однаковий – “Неправильне ім'я користувача або пароль” .

Якщо ім’я користувача та пароль було введено правильно, то після натискання кнопки “Увійти” користувача повертає на головну сторінку і

посилання у навігаційній панелі замінюються так само, як і при успішній реєстрації.

3. *Сторінка із калькулятором*

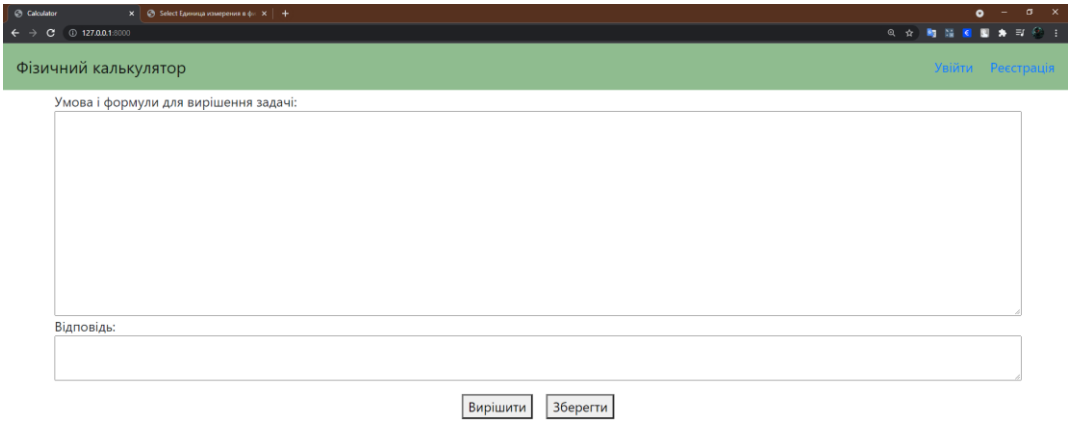

Рисунок 3.4 – Головна сторінка веб-застосунку за відсутності авторизації

На рисунку 3.4 зображено сторінку із калькулятором, яка відображається у незареєстрованого користувача. На ній присутня навігаційна панель та основний блок.

Навігаційна панель має ті ж самі елементи, що й у попередніх сторінках. При натисканні на посилання “Увійти” або “Реєстрація” відкриваються відповідні сторінки.

Основний блок складається з трьох частин:

1. Поле для введення умови та формул задачі і його назва.
2. Поле для відображення відповіді та його назва.
3. Кнопка для вирішення задачі.

У поле з умовою та формулами рішення задачі користувач повинен вводити відомі змінні та формули, розділяючи кожен новий запис новим рядком.

Після того, як користувач ввів необхідні дані у поле з умовою і натиснув кнопку “Вирішити”, спрацьовує один з двох сценаріїв:

1. Якщо умова і всі формули були введені правильно та у правильній послідовності, то у полі для відповіді з’явиться відповідне значення шуканої змінної разом із одиницею вимірювання у міжнародній системі одиниць.
2. Якщо в умові або формулах є помилка будь-якого типу (відсутність можливості знаходження значення змінної, неправильний порядок формул, помилка у записі формули тощо) виводиться повідомлення про неї.

Якщо користувач не ввів нічого до поля для формул, і натиснув на кнопку “Вирішити”, то з’явиться відповідне повідомлення – “Заповніть поле для формул”.

Приклад вигляду повідомлення при помилці або незаповненому полі для введення формул наведено на рис. 3.5.

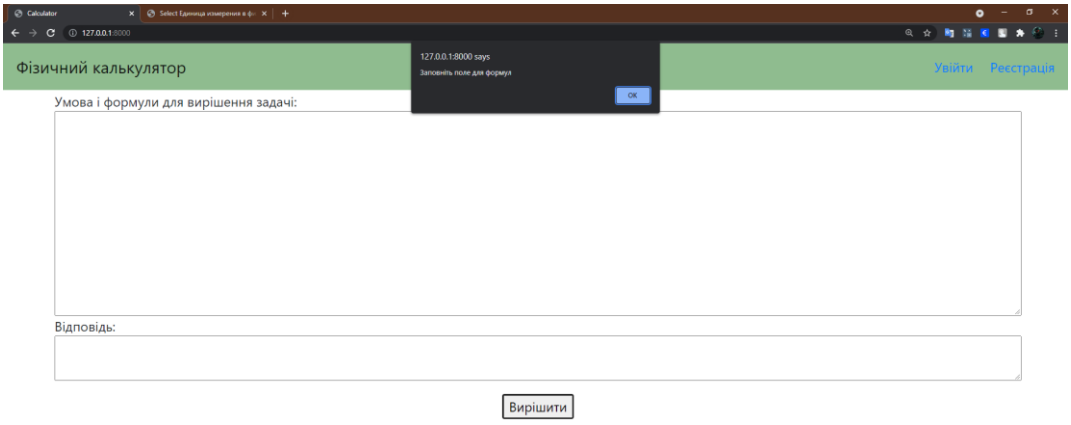

Рисунок 3.5 – Головна сторінка веб-застосунку із повідомленням про помилку

Тепер розглянемо інтерфейс сторінки калькулятора, коли користувач авторизований у системі. Для цього використаємо тестовий акаунт адміністратора.

Після авторизації зовнішній вигляд сторінки сильно змінюється (див. рис. 3.6).

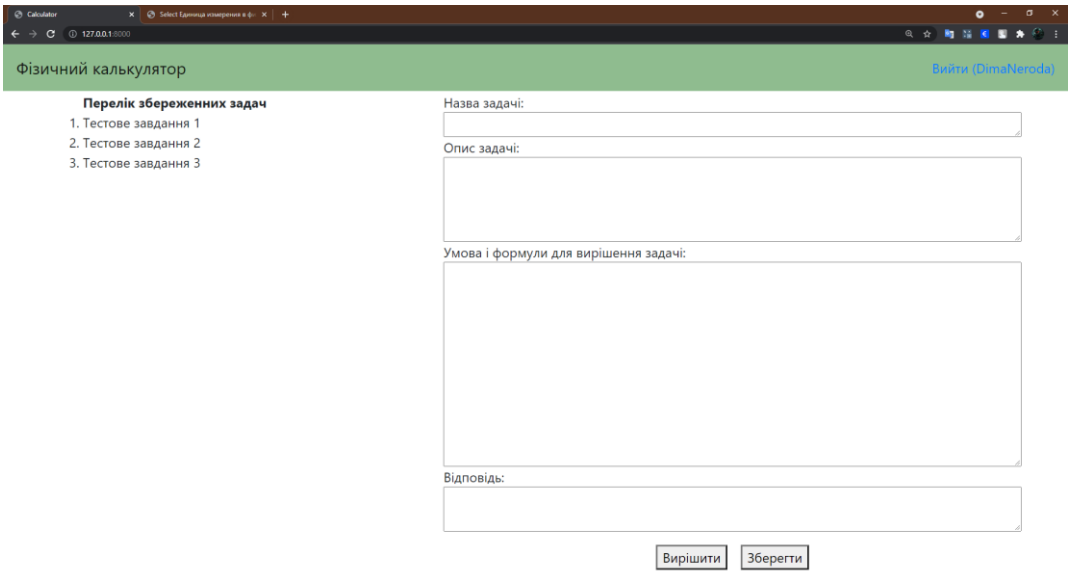

Рисунок 3.6 – Головна сторінка веб-застосунку для авторизованого користувача

У навігаційній панелі зникають посилання на сторінки реєстрації та авторизації і з’являється посилання із ім’ям користувача, при натисканні на яке відбувається вихід з акаунта.

У основному блоці з’являється 2 додаткових поля, одна кнопка та перелік збережених задач. Розглянемо кожний елемент більш детально.

Перше додаткове поле – це поле для введення назви задачі. Воно необхідно, якщо користувач хоче зберегти задачу.

У друге поле, за бажанням, можна внести текстовий опис задачі, яку треба зберігти. Це потрібно для кращого розуміння умови та формул, які записані у відповідному полі. Проте це поле не є обов’язковим для заповнення.

Кнопка “Зберігти” необхідна, коли є потреба переглянути задачу та її розв’язок у майбутньому. Для того, щоб операція збереження пройшла вдало, необхідно, щоб були заповнені усі поля на сторінці, окрім поля для опису задачі. Це означає, що поле для відповіді теж повинне бути заповненим, тобто задача повинна бути розв’язаною і хід рішення є правильним.

Якщо ж користувач буде намагатись зберігти задачу, не заповнивши хоча б одне обов’язкове поле, то з’явиться повідомлення, яке підкаже, які поля треба заповнити.

Після успішного збереження задачі, її назва з’являється у переліку в лівій частині основного блока. Якщо натиснути на неї, то поля справа будуть заповнені відповідними даними, які були збережені (див рис. 3.7).

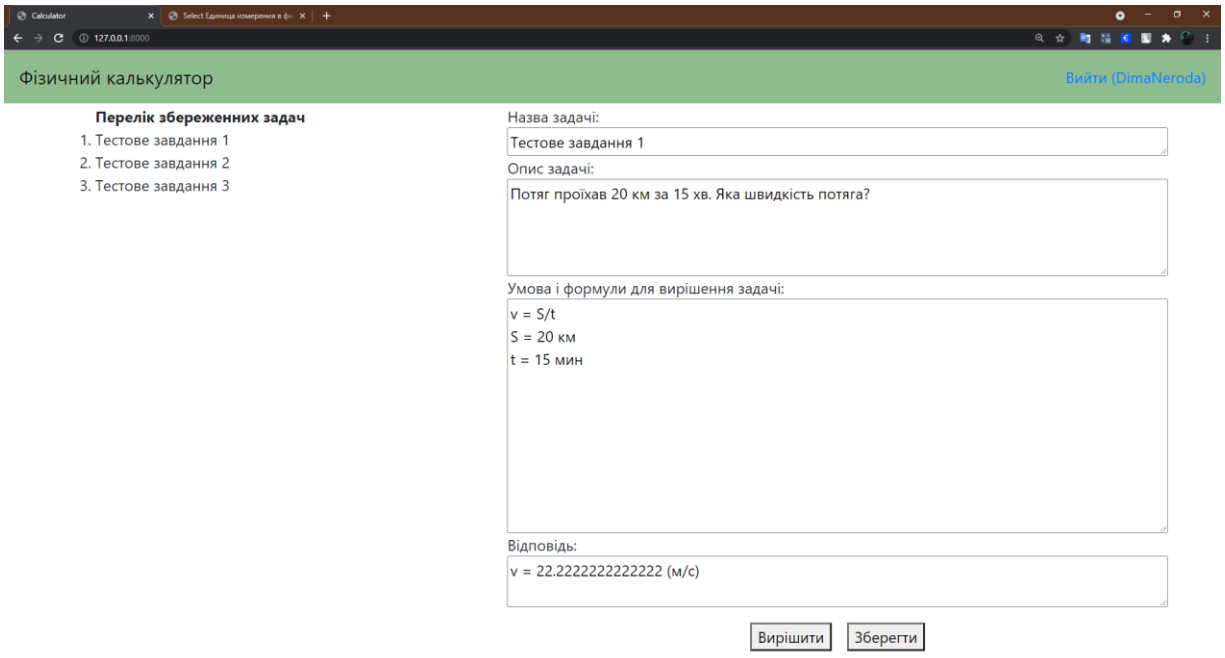

Рисунок 3.7 – Головна сторінка веб-застосункуза авторизованого користувача із збереженою задачею

Ми розглянули всі головні сторінки. Тепер перейдемо до огляду сторінок адміністрування.

Використовується базовий інтерфейс панелі адміністрування, який надається фреймворком Django за замовчуванням. Доступ здійснюється за допомогою створення у середовище розробки акаунта суперкористувача, логін і пароль якого треба потім на сторінці авторизації панелі адміністрування.

У додатках будуть наведений інтерфейс трьох основних сторінок – головна, сторінка із переліком записів у таблиці з одиницями вимірювання та сторінку редагування запису таблиці з одиницями вимірювання. Сторінки інших таблиць будуть схожими, будуть відрізнятися тільки поля відповідних записів.

3.2 Логіка програмного модуля

Після того, як ми окреслили зовнішній вигляд програмного модуля, перейдемо безпосередньо до бекенду веб-застосунка.

У цьому пункті будуть розглянуті лише основний алгоритм та функції, які відповідають за перевірку, валідацію та обчислення фізичних виразів.

1. *Первісна перевірка.*

Розпочнемо з моменту, коли користувач хоче вирішити задачу і заповнив поле для умови та формул і натиснув кнопку “Вирішити”. Після цього веб-застосунок, за допомогою коду JQuery перевіряє отриманий текст з відповідного поля, порожній він чи ні. Якщо порожній – виводиться відповідне повідомлення. Потім, за допомогою коду, який наведений нижче відбувається відокремлення кожного виразу шляхом розбиття первісної строки на елементи масиву за знаком переносу строки:

```
let condition_fld_list = value_condition_fld.split(/\n/).reverse();
```

Окрім цього, ми переставляємо місцями усі отримані строки да допомогою метода reverse().

Після цього ми у циклі перевіряємо кожну строку, чи є вона виразом відомої змінної або невідомої змінної. Окрім цього також перевіряється, чи однакове значення відомої змінної із її назвою. Це означає, що замість числового значення змінної ми будемо використовувати значення у вигляді літери.

Якщо на якомусь етапі проходження циклом по списку виникає помилка, то цикл зупиняється і користувачу виводиться відповідне повідомлення. Як приклад нижче наведений код, який виводить повідомлення та зупиняє виконання функції, якщо у одному з виразів відсутня права частина (тобто немає значення):

```
if (expression_val_unit === "=") {
   alert("В одному з виразів відсутня права частина");
   return false
}
```

2. Відправка даних на сервер

Якщо увесь цикл завершився вдало, то наприкінці ми отримаємо 2 списки та 2 словника. У списках зберігаються назви відомих і невідомих змінних, а у словниках – відповідні пари: назва змінної – словник із ключами “value” і “unit”, де зберігаються значення та одиниця вимірювання змінної відповідно.

Потім ми зберігаємо отримані структури даних в один словник “send_data”, який потрібен для передачі на сервер. Саму передачу ми будемо здійснювати за допомогою технології AJAX. Нижче наведено реалізація цього у вигляді коду:

```
$.ajax({
   url: '/physical_data/',
   type: 'GET',
   data: send_data,
   cache: true,
   success: function (data) {
```

```
        let parseData = JSON.parse(data);
        let answer_variable = "";
        for (let key in parseData) {
           let answer_value = parseData[key]["value"];
           answer_value = answer_value.replaceAll("**", "^");
           answer_value = delete_zero(answer_value);
           let answer_unit = parseData[key]["unit"];
           answer_unit = answer_unit.replaceAll("**", "^");
           if (answer_unit) {answer_unit = " (" + answer_unit + ")"}
           answer_variable += key + " = " + answer_value + answer_unit + "\n";}
        answer_fld.val(answer_variable);},
    error: function () {alert('Помилка в умові/формулах задачі');}});
```

Тепер більш детально розберемо, що позначає кожний з параметрів:

1. url – це адреса, за якою буде зберігатись інформація, яку ми передаємо.
2. type – тип запиту для отримання/відправки даних на сервер.
3. data – це дані, які ми відправляємо на сервер.
4. cache – налаштування кешування даних.
5. success – спрацьовує, коли сервер без помилок опрацював отримані дані та повернув правильну відповідь.
6. error – відповідно спрацьовує, коли сервер повертає помилку.

Код у блоці success приймає від сервера значення виразу, який треба було обчислити, у вигляді словника, за необхідністю видаляються зайві цифри “0” у кінці дробової частини чисел у значенні, записується відповідь у вигляді рядка і змінює значення поля для відповіді на отриманий рядок.

У випадку повернення помилки, функція блоку error виводить на екран користувача відповідне повідомлення.

Тепер перейдемо до оброблення відправлених даних на сервері.

3. Оброблення даних сервером

Після того, як сервер отримав дані з фронтенду за допомогою AJAX, необхідно перевести їх у потрібний формат. Для цього ми використаємо вбудовану бібліотеку json:

```
condition_data_var_dict = json.loads(request.GET.get('cond_data_var_dict'))
condition_data_var_list = json.loads(request.GET.get('cond_data_var_list'))
```

```
condition_data_for_dict = json.loads(request.GET.get('cond_data_for_dict'))
condition_data_for_list = json.loads(request.GET.get('cond_data_for_list'))
```

Таким чином ми отримали інформацію у вигляді двох словників та двох списків. Потім ми створюємо наступні словники:

1. Ключ – одиниця вимірювання, значення – список префіксів.
2. Ключ – одиниця вимірювання, значення – множник.
3. Ключ – префікс, значення – множник.
4. Ключ – одиниця вимірювання, значення – аналог у міжнародній системі одиниць.

Для цього ми спочатку витягуємо з бази даних усі записи таблиць “PhysicalUnits” та “PrefixPhysicalUnits”. Після чого у циклі проходим по об’єктам та заповнюємо словники необхідними даними.

Окрім цього, ми створюємо список, який складається з семи основних одиниць вимірювання міжнародної системи одиниць:

```
all_unit_SI_list = ['с', 'м', 'кг', 'А', 'К', 'моль', 'кд']
```

4. Конвертація одиниць вимірювання

Після цього ми проходимо у циклі по всьому списку з відомими змінними. Це потрібно для того, щоб перевести усі одиниці вимірювання до міжнародної системи одиниць. Окрім цього, може також змінитися значення відомої змінної, через наявність префіксу або специфічної одиниці вимірювання (наприклад, година).

Для того, щоб конвертувати одиницю вимірювання, використовується функція “full_calc_unit”:

```
def full_calc_unit(par_test_str, par_unit_prefix_dict, par_unit_value_dict,
par_prefix_value_dict, par_unit_si_dict, par_all_si_list)
```

Ця функція приймає строку з одиницею вимірювання, яку треба конвертувати, усі словники, які ми зробили у пункті 3, та список із основними одиницями вимірювання міжнародної системи одиниць. Вона

об’єднує у собі всі додаткові функції для валідації та конвертації. Тепер розберемо її більш детально.

Спочатку у ній ми переводимо строку із одиницями вимірювання у список, у якому елементами є окремі складові одиниці вимірювання та знаки “-”, “*”, “/”, “^”, за допомогою функції “parse_str_unit”:

```
def parse_str_unit(par_str_unit)
```

Параметрами вона приймає тільки одиницю вимірювання у вигляді рядка. Повертає список або булеве значення “False”, якщо у рядку є математична помилка.

Потім ми конвертуємо у міжнародну систему одиниць отриману одиницю вимірювання у вигляді списку за допомогою функції “check_unit_expr”:

```
def check_unit_expr(par_list_expr, par_prefix_unit_dict, par_multi_unit_dict,
par_multi_prefix_dict, par_unit_si_dict)
```

У вигляді параметрів приймає список, отриманий з попередньої функції та словники, які ми отримали у пункті 3.

Повертає неспрощений рядок із конвертованими одиницями вимірювання та множниками, якщо усі одиниці вимірювання та множники були у словнику. Якщо ж ні, повертається булеве значення “False”.

Після цього, за допомогою функції “create_dict_calc”, ми спрощуємо отриманий у попередній функції вираз та відокремоюємо загальний множник від одиниці вимірювання:

```
def create_dict_calc(par_unit_calc_str, par_unit_list)
```

Ця функція приймає у якості параметрів строку, яку ми отримали з попередньої функції “check_unit_expr” і список із позначеннями основних одиниць вимірювання міжнародної системи одиниць.

Для спрощення виразу ми використовуємо сторонню бібліотеку “sympy”:

```
for elem in par_unit_list:
    unit_calc_dict[elem] = Symbol(elem, positive=True, real=True)
test_sympy_unit_calc = parse_expr(par_unit_calc_str_copy,
                                  local_dict=unit_calc_dict,
                                  transformations=(auto_number, convert_xor),
                                  evaluate=True)
```

Функція “parse_expr” перетворює строку у спеціальний об’єкт, з яким може працювати “sympy”. У якості параметрів вона приймає вираз, який треба спростити, словник із позначеннями змінних, які можуть бути у виразі, при чому ми відмічаємо, що вони є додатними та раціональними, перелік допустимих перетворень (наприклад, convert_xor позначає, що замість знака “**” можна використовувати “^”) та позначаємо, що вираз можна спрощувати.

Наприкінці функція повертає словник із множником та одиницею вимірювання у міжнародній системі одиниць. Саме цей результат повертає функція “full_calc_unit”, яка об’єднує у собі всі попередні. Проте, якщо на якомусь кроці сталась помилка, функція поверне булеве значення “False”.

Якщо помилки немає, то отриманий результат зберігається у словнику “final_var_dict”.

Після того, як були оброблені усі відомі змінні, починається обчислення невідомих. Воно проходить у порядку розташування назв змінних у відповідному списку, і якщо якась змінна буде на своєму місці – з’явиться помилка.

Обчисленням займається функція “full_validation_and_calculation”:

```
def full_validation_and_calculation(par_expr_str, par_var_dict, par_unit_list,
par_multi_unit)
```

Параметрами вона приймає строку із виразом, словник із вже відомими змінними “final_var_dict”, списку із основними одиницями вимірювання міжнародної системи одиниць, і множник, на який треба

помножити вираз. Цей множник з’являється від конвертування одиниці вимірювання.

Функція “full_validation_and_calculation” також об’єднує кілька функцій для валідації, перетворення та обчислення.

Перша фунція – “validation_and_parser_expression”:

```
def validation_and_parser_expression(par_str_expression, par_input_var_dict)
```

Вона приймає у якості параметрів рядку із виразом, який треба обчислити, та словник із відомими на цей час змінними. Ця функція перевіряє рядок на правильність синтаксису фізичного виразу та на наявність змінних, яких немає у словнику. Окрім цього, вона перетворює строку на список, який складається з параметрів, числових значень, знаків “+”, “-”, “/”, “*”, “^”, “(”, “)”. Якщо вираз правильний, то функція повертає отриманий список, якщо ні – повертає логічне значення “False”.

Потім йде функція “prepare_for_calculate”:

```
def prepare_for_calculate(par_list_expr, par_input_variable_dict)
```

Вона замінює змінні у виразі на їх значення та на їх одиниці вимірювання.

Приймає у якості параметрів список із виразом, отриманим за допомогою функції “validation_and_parser_expression” та словник із відомими змінними.

Повертає два відповідних списку або логічне значення “False”, якщо сталася помилка.

Після цього використовуються 2 функції: “full_calculate_const” та “full_calculate_unit”:

```
def full_calculate_const(par_list_const, par_var_list, par_multi_unit)
def full_calculate_unit(par_list_unit, par_unit_list)
```

Перша обчислює значення виразу, а друга – одиницю вимірювання. Параметри функції “full_calculate_const”: список із значеннями виразу, список із назвами відомих змінних та множник від одиниці вимірювання. Параметри функції “full_calculate_unit”: список із одиницями вимірювання виразу та список із основними одиницями вимірювання міжнародної системи одиниць. В обчисленні та спрощенні використовується та ж сама бібліотека “sympy”.

Отримані відповіді повертаються головною функцією і зберігаються у словнику “final_var_dict”.

Після того, як були знайдені усі невідомі змінні, на фронтенд частину застосунка повертається значення та одиниця вимірювання останньої змінної, яка в блоку умови та формул була першою.

3.3 Тестування веб-застосунка

Програмний модуль тестувався протягом усього циклу розроблення. Для цього використовувався локальний сервер у вигляді комп’ютера, на якому проводилось розроблення і написання програмного коду. Це дозволяло відслідковувати, як зміні у коді впливають на програмний модуль і, якщо виникали помилки – швидко знаходити їх причини та усувати їх.

Але такого тестування недостатньо, якщо є наміри викладання програмного продукта у вільний доступ. Причина цього – різне апаратне та програмне забезпечення у користувачів. Тому, коли розроблення програмного модуля завершено, необхідно провести повноцінне тестування, яке буде включати в себе перевірку працездатності, швидкодії та розташування елементів інтерфейсу у різних браузерах на комп’ютері та смартфоні.

Перелік браузерів на операційній системі Windows:

1. Google Chrome
2. FireFox
3. Microsoft Edge
4. Opera
5. Internet Explorer

Перелік браузерів на операційній системі Android:

1. Google Chrome
2. Samsung Internet
3. Microsoft Edge
4. FireFox
5. UC
6. Via
7. Opera

Усі браузери мали останню стабільну публічну версію. Тепер перейдемо до переліку пристроїв, на яких проводилось тестування:

1. Комп’ютер із процесором AMD Ryzen 1600AF
2. Комп’ютер із процесором Intel Core i7 8700K
3. Смартфон Samsung Galaxy S10
4. Смартфон Google Pixel 4 XL
5. Смартфон Meizu 15 Plus

Для тестування розташування елементів інтерфейсу буде використано візуальний огляд сторінок сайту, а для тестування працездатності – сценарії використання з відповідної діаграми.

Тестування веб-застосунка було пройдено успішно. Усі елементи інтерфейсу були на своїх місцях, швидкодія була однакова на усіх пристроях, а всі функції працювали незалежно від браузера.

ВИСНОВКИ

У ході кваліфікаційної роботи було спроєктовано та розроблено програмний модуль, який був створений у вигляді веб-застосунка. Його основне призначення – розв’язання фізичних задач.

Для цього було:

1. Проаналізовано актуальність створення програмного модуля.
2. Спроєктовано модель та можливості програмного модуля.
3. Оглянуто та вивчено необхідні інструменти та технології, які потрібні для розроблення програмного модуля.
4. Розроблено дизайн інтерфейсу програмного модуля.
5. Написано програмний код фронтенд частини веб-застосунка.
6. Написано програмний код бекенд частини веб-застосунка.
7. Успішно протестовано програмний модуль у різних середовищах.

У результаті нам вдалося створити зручний та якісний веб-застосунок для розв’язку фізичних задач, яким зможуть скористатися усі учні та студенти із мобільним або десктопним гаджетом та виходом до мережі “Інтернет”.

Як показало тестування, створений програмний модуль “Фізичний калькулятор” вже можна використовувати як у середніх та вищих навчальних закладах. Окрім цього, немає жодних перешкод для того, щоб зробити веб-застосунок загальнодоступним, тобто щоб будь-який користувач мережі “Інтернет” міг зайти на нього та розв’язати необхідну фізичну задачу. Для цього потрібно тільки знайти віддалений сервер і розмістити веб-застосунок на ньому.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Omni Calculator [Електронний ресурс] – Режим доступу до ресурсу: https://www.omnicalculator.com/
2. AllCalc [Електронний ресурс] – Режим доступу до ресурсу: https://allcalc.ru/
3. PlanetCalc [Електронний ресурс] – Режим доступу до ресурсу: https://planetcalc.com/
4. What is Django? Advantages and Disadvantages [Електронний ресурс] – Режим доступу до ресурсу: https://hackr.io/blog/what-is-django-advantages-and-disadvantages-of-using-django
5. What is a Framework? [Definition] Types of Frameworks [Електронний ресурс] – Режим доступу до ресурсу: https://hackr.io/blog/what-is-frameworks
6. Open Source Licensing Guide [Електронний ресурс]. – 2015. – Режим доступу до ресурсу: https://www.newmediarights.org/open_source/new_media_rights_open_source_licensing_guide
7. Databases [Електронний ресурс] – Режим доступу до ресурсу: https://docs.djangoproject.com/en/3.2/ref/databases/
8. Top Programming Languages 2021 [Електронний ресурс] – Режим доступу до ресурсу: https://spectrum.ieee.org/top-programming-languages-2021
9. SQLite As An Application File Format [Електронний ресурс] – Режим доступу до ресурсу: https://www.sqlite.org/aff_short.html

10. Structuring the web with HTML [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Learn/HTML
11. CSS [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Glossary/CSS
12. JavaScript — Dynamic client-side scripting [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Learn/JavaScript
13. jQuery [Електронний ресурс] – Режим доступу до ресурсу: https://jquery.com/
14. Django [Електронний ресурс] – Режим доступу до ресурсу: https://www.djangoproject.com/
15. Quick install guide [Електронний ресурс] – Режим доступу до ресурсу: https://docs.djangoproject.com/en/3.2/intro/install/
16. Working with forms [Електронний ресурс] – Режим доступу до ресурсу: https://docs.djangoproject.com/en/3.2/topics/forms/
17. Models [Електронний ресурс] – Режим доступу до ресурсу: https://docs.djangoproject.com/en/3.2/topics/db/models/
18. Migrations [Електронний ресурс] – Режим доступу до ресурсу: https://docs.djangoproject.com/en/3.2/topics/migrations/
19. Making queries [Електронний ресурс] – Режим доступу до ресурсу: https://docs.djangoproject.com/en/3.2/topics/db/queries/
20. URL dispatcher [Електронний ресурс] – Режим доступу до ресурсу: https://docs.djangoproject.com/en/3.2/topics/http/urls/
21. The Django admin site [Електронний ресурс] – Режим доступу до ресурсу: https://docs.djangoproject.com/en/3.2/ref/contrib/admin/

22. What Is SQLite? [Електронний ресурс] – Режим доступу до ресурсу: https://www.sqlite.org/index.html
23. HTML [Електронний ресурс] – Режим доступу до ресурсу: https://html.spec.whatwg.org/multipage/
24. Learn to style HTML using CSS [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Learn/CSS
25. jQuery API [Електронний ресурс] – Режим доступу до ресурсу: https://api.jquery.com/
26. Browser Support [Електронний ресурс] – Режим доступу до ресурсу: https://jquery.com/browser-support/
27. Python For Beginners [Електронний ресурс] – Режим доступу до ресурсу: https://www.python.org/about/gettingstarted/
28. Beginner's Guide to Python [Електронний ресурс] – Режим доступу до ресурсу: https://wiki.python.org/moin/BeginnersGuide
29. General Python FAQ [Електронний ресурс] – Режим доступу до ресурсу: https://docs.python.org/3/faq/general.html
30. Library and Extension FAQ [Електронний ресурс] – Режим доступу до ресурсу: https://docs.python.org/3/faq/library.html
31. Summerfield M. Programming in Python 3: A Complete Introduction to the Python Language / Mark Summerfield., 2008. – (Addison-Wesley Professional)
32. Маттес Е. Пришвидшений курс Python / Ерік Маттес., 2021. – 600 с.
33. Лутц М. Изучаем Python / Марк Лутц., 2020.
34. Б. Дауни А. Основы Python. Научитесь думать как программист / Аллен Б. Дауни., 2021. – 304 с.

35. Пол Мюллер Д. Python и наука о данных для чайников / Д. Пол Мюллер, М. Лука., 2020. – 512 с.
36. Геддіс Т. Начинаем программировать на Python / Тоні Геддіс., 2019. – 768 с.
37. Дронов В. JavaScript. 20 уроков для начинающих / Володимир Дронов., 2021. – 352 с.
38. Дронов В. HTML и CSS. 25 уроков для начинающих / Володимир Дронов., 2021. – 400 с.
39. Тіттел Е. HTML5 и CSS3 для чайников / Е. Тіттел, К. Мінник., 2016. – 400 с.
40. Постолит А. Python, Django и PyCharm для начинающих / Анатолій Постолит., 2021. – 464 с.

ДОДАТКИ

Додаток А

Додаток 1

КОДЕКС АКАДЕМІЧНОЇ ДОБРОЧЕСНОСТІ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ ХЕРСОНСЬКОГО ДЕРЖАВНОГО УНІВЕРСИТЕТУ

Я, _____Нерода Дмитро Олександрович_____, учасник(ця) освітнього процесу Херсонського державного університету, **УСВІДОМЛЮЮ**, що академічна доброчесність – це фундаментальна етична цінність усієї академічної спільноти світу.

ЗАЯВЛЯЮ, що у своїй освітній і науковій діяльності **ЗОБОВ'ЯЗУЮСЯ**:

– дотримуватися:

- вимог законодавства України та внутрішніх нормативних документів університету, зокрема Статуту Університету;
- принципів та правил академічної доброчесності;
- нульової толерантності до академічного плагіату;
- моральних норм та правил етичної поведінки;
- толерантного ставлення до інших;
- дотримуватися високого рівня культури спілкування;

– надавати згоду на:

- безпосередню перевірку курсових, кваліфікаційних робіт тощо на ознаки наявності академічного плагіату за допомогою спеціалізованих програмних продуктів;
- оброблення, збереження й розміщення кваліфікаційних робіт у відкритому доступі в інституційному репозитарії;
- використання робіт для перевірки на ознаки наявності академічного плагіату в інших роботах виключно з метою виявлення можливих ознак академічного плагіату;

– самостійно виконувати навчальні завдання, завдання поточного й підсумкового контролю результатів навчання;

– надавати достовірну інформацію щодо результатів власної навчальної (наукової, творчої) діяльності, використаних методик досліджень та джерел інформації;

– не використовувати результати досліджень інших авторів без використання покликань на їхню роботу;

– своєю діяльністю сприяти збереженню та примноженню традицій університету, формуванню його позитивного іміджу;

– не чинити правопорушень і не сприяти їхньому скоєнню іншими особами;

– підтримувати атмосферу довіри, взаємної відповідальності та співпраці в освітньому середовищі;

– поважати честь, гідність та особисту недоторканність особи, незважаючи на її стать, вік, матеріальний стан, соціальне становище, расову належність, релігійні й політичні переконання;

– не дискримінувати людей на підставі академічного статусу, а також за національною, расовою, статевою чи іншою належністю;

– відповідально ставитися до своїх обов'язків, вчасно та сумлінно виконувати необхідні навчальні та науково-дослідницькі завдання;

– запобігати виникненню у своїй діяльності конфлікту інтересів, зокрема не використовувати службових і родинних зв'язків з метою отримання нечесної переваги в навчальній, науковій і трудовій діяльності;

– не брати участі в будь-якій діяльності, пов'язаній із обманом, нечесністю, списуванням, фабрикацією;

– не підроблювати документи;

– не поширювати неправдиву та компрометуючу інформацію про інших здобувачів вищої освіти, викладачів і співробітників;

– не отримувати і не пропонувати винагород за несправедливе отримання будь-яких переваг або здійснення впливу на зміну отриманої академічної оцінки;

– не залякувати й не проявляти агресії та насильства проти інших, сексуальні домагання;

– не завдавати шкоди матеріальним цінностям, матеріально-технічній базі університету та особистій власності інших студентів та/або працівників;

– не використовувати без дозволу ректорату (деканату) символіки університету в заходах, не пов'язаних з діяльністю університету;

– не здійснювати і не заохочувати будь-яких спроб, спрямованих на те, щоб за допомогою нечесних і негідних методів досягати власних корисних цілей;

– не завдавати загрози власному здоров'ю або безпеці іншим студентам та/або працівникам.

УСВІДОМЛЮЮ, що відповідно до чинного законодавства у разі недотримання Кодексу академічної доброчесності буду нести академічну та/або інші види відповідальності й до мене можуть бути застосовані заходи дисциплінарного характеру за порушення принципів академічної доброчесності.

07.09.2020		Дмитро Нерода
(дата)	(підпис)	(ім'я, прізвище)