Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Розробка вебсайту школи іноземний мов «Study isn’t hard»

Виконав: студент IV курсу, групи СНс-42
спеціальності 122 Комп’ютерні науки
(шифр і назва спеціальності)

_____ Луковський В.А.
(підпис) (прізвище та ініціали)

Керівник _____ Дмитроца Л. П.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Марценко С. В.
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Боднарчук І.О.
(підпис) (прізвище та ініціали)

Рецензент _____ Яцишин В. В.
(підпис) (прізвище та ініціали)

Тернопіль
2024

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
«__» _червня_ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)

за спеціальністю 122 Комп'ютерні науки
(шифр і назва спеціальності)

Студенту Луковському Володимиру Андрійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка вебсайту школи іноземних мов «Study isn't hard»

Керівник роботи Дмитроца Леся Павлівна, кандидат технічних наук,
Доцент кафедри комп'ютерних наук
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «*29*» *квітня* 2024 року № *4/7-472*

2. Термін подання студентом завершеної роботи 24 червня 2024р.

3. Вихідні дані до роботи технічне завдання на розробку програмного забезпечення, мови програмування: Java, JavaScript, MySQL, HTML, CSS стандарти IEEE 830-1998

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. 1. Аналіз предметної області та постановка завдання 1.2 Аналіз предметної області 1.2 Аналітичний огляд існуючих рішень. 1.3 Технічне завдання. 1.3.1 Загальні характеристики. 1.3.2 Архітектура розроблюваного ПЗ. 1.3.3 Вимоги до серверу. 1.3.4 Вимоги до збереження даних. 1.4 Висновок до першого розділу. 2. Проектна частина. 2.1 Розробка структури сайту і вебсторінок. 2.2 Розробка дизайну. 2.3 Проєктування та розробка бази даних. 2.4 Висновок до другого розділу. 3. Програмна реалізація. 3.1 Верстка вебсторінок. 3.2 Розробка серверу. 3.3 Розробка клієнтської частини. 3.4 Висновок до третього розділу. 4. Безпека життєдіяльності, основи охорони праці. 4.1 Психологічні чинники небезпеки. 4.2 Правила техніки безпеки при експлуатації обладнання. 4.3 Висновок до четвертого розділу. Висновки. Перелік джерел. Додатки.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
Слайди презентації 1-15.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці		12.06.2024	15.06.2024

7. Дата видачі завдання 29 січня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	30.01.2024	*Виконано*
2.	Підбір джерел про школи іноземних мов	31.01.2024-03.02.2024	*Виконано*
3.	Опрацювання джерел по темі кваліфікаційної роботи	04.02.2024-06.02.2024	*Виконано*
4.	Виконання дослідження щодо використання	07.02.2024-11.02.2024	*Виконано*
	англійської мови та методів її вивчення. Розроблення		
	технічного завдання		
5.	Оформлення розділу «Аналіз предметної області	03.06.2024-05.06.2024	*Виконано*
	та постановка завдання»		
6.	Оформлення розділу «Проєктна частина»	06.06.2024-08.06.2024	*Виконано*
7.	Оформлення розділу «Програмна реалізація»	09.06.2024-11.06.2024	*Виконано*
	»		
8.	Виконання завдання до підрозділу «Безпека життєдіяльності»	12.06.2024-13.06.2024	*Виконано*
9.	Виконання завдання до підрозділу «Основи охорони	14.06.2024-15.06.2024	*Виконано*
	праці»		
10.	Оформлення кваліфікаційної роботи	16.06.2024-17.06.2024	*Виконано*
11.	Нормоконтроль	18.06.2024-19.06.2024	*Виконано*
12.	Перевірка на плагіат	20.06.2024	*Виконано*
13.	Попередній захист кваліфікаційної роботи	21.06.2024	*Виконано*
14.	Захист кваліфікаційної роботи	29.06.2024	

Студент _____ Луковський В. А.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Дмитроца Л. П.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Розробка вебсайту школи іноземних мов «Study isn’t hard» // Кваліфікаційна робота освітнього рівня «Бакалавр» // Луковський Володимир Андрійович// Тернопільський національний технічний університет імені Івана Пулюя, факультет комп’ютерно-інформаційних систем і програмної інженерії, кафедра комп’ютерних наук, група СНс-42 // Тернопіль, 2024 // 78 С. , рис. – 39, додат. – 5, бібліогр. – 44.

Ключові слова: вебсайт, дизайн, сервер, база даних, англійська мова, структура, верстка.

Кваліфікаційна робота присвячена розробці вебсайту для навчання англійської мови. В першому розділі кваліфікаційної роботи проаналізовано предметну область. Розглянуто існуючі рішення. Визначено технічне завдання.

В другому розділі кваліфікаційної роботи розроблено структуру вебсайту. Проведено роботу над дизайном вебсторінок. Подано структуру бази даних.

В третьому розділі кваліфікаційної роботи описано програмну реалізацію. Реалізовано сервер та клієнтську частину вебсайту. Зверстано вебсторінки.

В четвертому розділі кваліфікаційної роботи досліджено чинники психологічної небезпеки. Подано правила техніки безпеки при експлуатації обладнання.

ANNOTATION

Development of the website for the foreign language school «Study isn’t hard»// Qualification work of the educational level "Bachelor" // Volodymyr Lukovskyi // Ternopil Ivan Pulyu National Technical University, Computer and Information Systems and Software Engineering Faculty, Computer Sciences Department, group SNs-42 // Ternopil, 2024 //78 P., fig. – 39, annexes. – 5, references – 44.

Keywords: website, design, server, database, English language, structure, layout.

The qualification work is dedicated to the study of the use and process of learning English through web resources. The goal of the work is to develop a website aimed at implementing educational activities, specifically improving the level of English proficiency in Ukrainian society. The first section of the qualification paper analyzes the subject area, reviews existing solutions, and defines the technical task.

In the second section of the qualification work, the website structure is developed, and work on the web pages' design is conducted. The database structure is presented.

The third section of the qualification work describes the programming implementation, including the development of the server and client parts of the website, and the layout of the web pages.

The fourth section of the qualification work investigates psychological safety factors and provides safety rules for operating equipment.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

БД – база даних.

Клієнт-сервісна архітектура (або клієнт-серверна архітектура) –це модель мережевої архітектури, в якій функціональні обов'язки розподіляються між постачальником послуг (сервером) і споживачем послуг (клієнтом).

ПЗ – програмне забезпечення.

Сервер - це комп'ютерна система або програмне забезпечення, яке надає послуги або ресурси іншим комп'ютерам, відомим як клієнти, через мережу.

СУБД – система управління базами даних.

Хешування (або хеш-функція) - це процес перетворення вхідних даних будь-якої довжини в фіксований вихідний хеш-значення (хеш).

ЗМІСТ

ВСТУП

За даними W3Techs, понад п’ятдесят відсотків вмісту всіх інтернет-ресурсів представлені англійською мовою [1]. Цього вже достатньо для обґрунтування теми кваліфікаційної роботи. Проте, розглянемо тему важливості вивчення англійської мови детальніше.

Сьогодні усім відомі такі корпорації, як Apple, Microsoft, Meta, Google, а про такі, як Oracle, NVIDIA, TSMC чули якщо не всі, то більшість. Продуктами вище перечислених компаній ми користуємось кожного дня. Більше того, вони стали невід’ємною частиною нашого життя. Що цікаво, з даного списку лише одна - не американська, і це – TSMC. Якщо дослідити ринкову капіталізацію найбільший IT компаній, то виявиться що в рейтингу чотирнадцять із двадцяти компаній – це компанії із США [2].

Щоб дослідження було конструктивним і аргументованим, варто розглянути конкурентів «великої п’ятірки». Посилаючись на список найдорожчих технологічних компаній за ринковою капіталізацією, варто виділити Tencent, Alibaba, Pinduoduo. Для західного користувача ці корпорації відомі лише завдяки TikTok і AliExpress. В основному вони орієнтовані на внутрішній китайський ринок, тому більшість продуктів компанії залишаються невідомими широким масам населення. У розробці технологій активно використовуються західні винаходи, чого не відбувається в зворотньому напрямку. Загалом, китайський технічний сектор наздоганяє західний, і лише в рідкісний випадках обходить, хоч і дана тенденція змінюється.

Також можна виділити європейський, японський, корейський та корейські IT-сектори. Проте спільною мовою спілкування для них є англійська. Тай загальний вплив компаній з вище перерахованих держав не перевищить США. Мовою оригіналу для усіх сучасних розробок та технологій у першу чергу є саме англійська. Переклад іншими мовами, зазвичай, є результатом старань ентузіастів, які безоплатно їх перекладають. Проте очевидною є перевага

дослідження, до прикладу, документації мовою оригіналу у порівнянні із перекладом. Це вкотре підкреслює важливість знання англійської.

За даними інтернет-видання «Суспільне мовлення» лише один відсоток населення володіє «вільною» англійською, а майже сорок чотири відсотки не знає даної мови [3]. Загалом, оцінити такий показник досить складно, адже мова це абстрактна річ і абсолютних показників її знання визначити неможливо. Але існують різні індекси, які намагаються оцінити даний параметр. До прикладу, EF EPI – індекс, що оцінює рівень володіння англійською у різних державах. І за останніми даними Україна посідає сорок п’яте місце у даному рейтингу. Загалом, це краще ніж більшість, але з іншої сторони ми суттєво відстаємо від практично усіх держав Європи [4].

Тому метою даної кваліфікаційної роботи є розробка вебсайту школи іноземних мов. Щоб досягнути поставленої мети потрібно:

– Дослідити існуючі реалізації, в тому числі функціонал, дизайн, визначити сучасні тенденції.

– Розробити структуру бази даних. Описати залежності між таблицями та їхні поля.

– Створити дизайн вебсайту, що включає в себе реалізацію адаптивних макетів вебсторінок для покращення користувацького досвіду роботи.

– Зверстати вебсторінки по дизайнерських шаблонах.

– Розробити сервер, за описаною документацією, із дотриманням сучасних вимог та тенденцій, щоб забезпечити можливість підтримки і вдосконалення у майбутньому.

Практичне значення одержаних результатів полягає у створенні інструменту, який сприятиме покращенню рівня володіння англійською мовою серед користувачів. Вебсайт школи іноземних мов забезпечить зручний доступ до різноманітних ресурсів для вивчення англійської, включаючи навчальні матеріали, тести та інші інструменти.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1 Аналіз предметної області

Предметна область кваліфікаційної роботи включає розробку вебсайту для школи іноземних мов «Study isn't hard», яка надає послуги з вивчення англійської мови за допомогою різних методів та технологій [5]. Основна мета вебсайту полягає в забезпеченні ефективного та зручного навчального середовища для учнів різного віку та рівня знань.

У вступі до даної кваліфікаційної роботи вже було згадано індекс EF EPI. Проте не було зроблено висновків на основі даного показника. Згідно із дослідженнями EF English Proficiency Index можна виявити кореляцію між рівнем володіння англійською громадян держави і її економічними показниками. Оцінка проводиться на дорослому населенні у більш ніж ста країнах світу. Отже, високий рівень володіння англійської корелює із фінансовою забезпеченістю її носіїв, і навпаки.

Проте вплив англійської не обмежується на фінансових та економічних питаннях. Ця мова незамінна у дипломатичній та наукових сферах. За даними PLOS – некомерційної науково-видавничої організації, більше дев’яносто п’яти відсотків робіт та досліджень публікуються англійською [6]. Точних даних щодо використання мов дипломатами знайти не вдалось. Проте Diplo – міжнародна некомерційна організація, яка спеціалізується на дослідженнях у політичній сфері, підкреслює важливість англійської. Ця теза аргументується статусом та вживанням цієї мови у різних світових організаціях та її знанням дипломатами. Схожа ситуація й в освітній сфері. Найвпливовіші та найвідоміші університети, такі як Гарвард, Оксофорд, Кебридж, що очолюють рейтинги найкращих у різних категоріях, використовують англійську у абсолютній більшості своїх навчальних програм. Статистика також підтверджує дану тенденцію у сфері медіа та розваг. За даними дослідження Nielsen, сімдесят

відсотків найбільш відомих серіалів та кіно у світі творені англійською мовою [7].

Загалом, світові тенденції очевидні. Навряд чи у найближчій перспективі знайдуться конкуренти англійській мові. За аналітикою WORK.ua можна зробити висновок, що дані тенденції також не оминули Україну [8]. Вакансії із вимогою знання англійської з’являються все дедалі частіше.

1.2 Аналітичний огляд існуючих рішень

Сьогодні існує безліч підходів до вивчення англійської. Зазвичай все залежить від вподобань людини. Тому усі можливі варіанти неможливо проаналізувати, тому розглянемо кілька наступних:

- традиційні курси та школи;
- соціальні мережі та мовні обміни;
- онлайн платформи та програми.

Напевне, перший варіант знайомий, якщо не усім, то більшості українцям. Банально у державних освітніх закладах викладається англійська мова за програмою. Проте такий підхід у більшості випадках менш ефективний у порівнянні із іншими варіантами, наведеними вище [9]. Індивідуальні заняття, до прикладу із репетитором, дозволяють врахувати усі особливості учня, що позитивно впливає на увесь процес навчання. Вчителям на групових заняттях складніше враховувати конкретний рівень знань, швидкість навчання кожного учасника, без втрати ефективності навчання. Хоча такий підхід має й очевидні переваги. Групові заняття дешевші за індивідуальні і можуть забезпечити кращу комунікативну практику.

Соціальні мережі та мовні обміни. Існують як і спеціалізовані сервіси для обміну досвіду між різними носіями мов, так і, до прикладу, усім знайомий Facebook. Як і раніше, все залежить від вподобань користувача.

Розглянемо вебсайт «Studystream». Загалом він не є конкретно спеціалізований на вивченні англійської, а швидше для організації навчання та

обміну досвідом загалом. Платформа надає можливість приєднатись до онлайн-кабінетів, де учасники можуть працювати над своїми завданнями. Також «Studystream» надає інструменти для планування навчання та обміну ресурсами. Загалом, цей ресурс найбільш корисний для встановлення нових соціальних контактів та обміну досвідом, що є важливим аспектом у вивченні будь-якої мови. Головну сторінку «Studystream» зображено на рисунку 1.1.

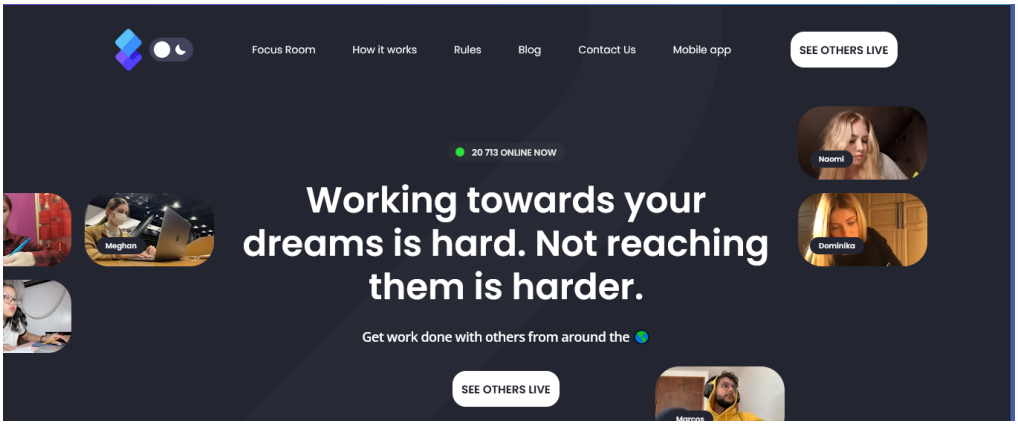

Рисунок 1.1 – Головна сторінка «Studystream»

Розглянемо сервіс, що називається «Study together». Платформа також надає можливість долучатись до конференцій для спільного навчання, організації навчання тощо. Проте даний ресурс дещо більш спеціалізований. «Studystream» надає просто загальні кімнати, із безліччю учасників зі всього світу. У «Study together» є можливість обрати уже існуючу спеціалізовану кімнату, або ж створити власну. Також успішність навчання відслідковується. Власну статистику можна переглянути у власному кабінеті, що зображений на рисунку 1.2.

Загалом, ці сервіси корисні для здобуття нових корисних соціальних зав’язків та організації навчання. Проте найбільш поширеною мовою спілкування є саме англійська, тому «Study together» і «Studystream» надають можливість вдосконалити свої навички писання та говоріння на практиці безкоштовно.

Іншим інструментом для вивчення англійської може бути практично будь-яка соціальна мережа. Вони також виконують функції об’єднання однодумців, зі спільними інтересами або мотивами. Таким чином і виникають групи, де учасники можуть ділитися корисними статтями, ресурсами, думками тощо. До прикладу розглянемо «BBC Learning English», що є офіційною групою у Facebook однойменної організації. У ній кожного для викладають пости, відео, аудіо матеріали для вивчення англійської, що зображено на рисунку 1.3.

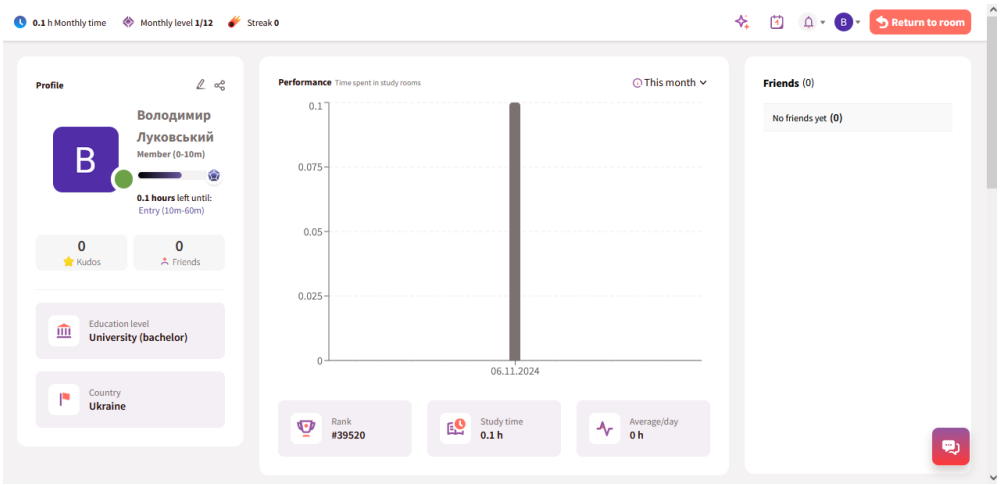

Рисунок 1.2 – Відслідковування прогресу навчання у власному кабінеті «Study together»

Проте даний підхід є неефективним: навчання часто обмежується поверхневими знаннями, адже зазвичай матеріали у таких групах це не конспекти, які глибоко розкривають ту, чи іншу тему, а короткі відео з яких можна вивчити кілька корисних фраз для побутового мовлення. Також негативну роль відіграє саме використання соціальних мереж у контексті навчання. Це підтверджується дослідженнями Гарвардської медичної школи [10]. Доведено, що використання соціальних мереж під час роботи, або ж навчання негативно впливає на когнітивні функції, такі як концентрація та пам'ять. У цьому і полягає основа відмінність між раніше описаними підходами. «Study together» і «Studystream» надають середовище для концентрації уваги, оскільки обмежують зовнішні подразники, щоб підвищити

ефективність навчання. У свою чергу соціальні мережі, до прикладу Facebook, намагаються якнайдовше затримати користувача, але з іншою метою.

Останнім підходом у списку зазначено онлайн платформи на програми. Раніше описані сервіси можна також сюди віднести, проте вони не специфікуються саме не вивченні мови, а надають можливості для підвищення ефективності навчання.

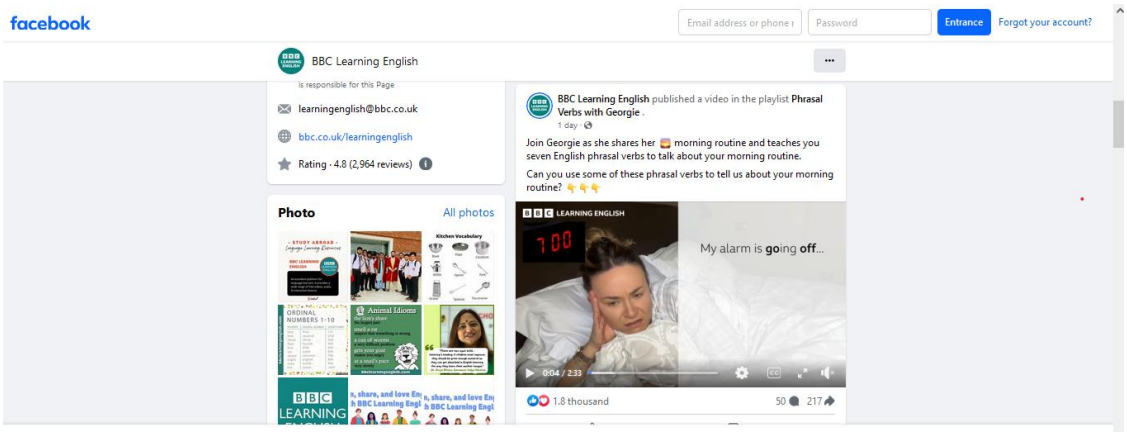

Рисунок 1.3 – Матеріали у групі «BBC Learning English» у Facebook

Спершу коротко розглянемо онлайн платформу «Italki». Це сервіс, який являє собою велику базу репетиторів, менторів тощо, які будуть навчати студента. Для користувача надаються безліч фільтрів, за якими він має знайти собі наставника, а на рисунку 1.4 це продемонстровано.

Рисунок 1.4 – Пошук наставника на платформі «Italki»

Далі розглянемо найпопулярніший сервіс для вивчення англійської – «Duoligo». Дана платформа надає готовий план, по якому користувач має поступово рухатись, тим самим покращуючи свої навички. «Duoligo» характерна гейміфікація. До прикладу: виконуючи завдання, учень отримує бали, за які може потім придбати доступ до додаткових матеріалів, або змінити декорації у власному профілі. Також сервіс намагається систематизувати навчання, відслідковуючи прогрес користувача. «Duoligo» стверджує: *«Дослідження показали, що, пройшовши 5 частин курсу Duolingo, ви опануєте стільки само матеріалу, скільки й за 5 семестрів в університеті»* [11]. Для підтвердження даної тези надається посилання на наукове дослідження [12]. На рисунку 1.5 зображено приклад використання даного сервісу.

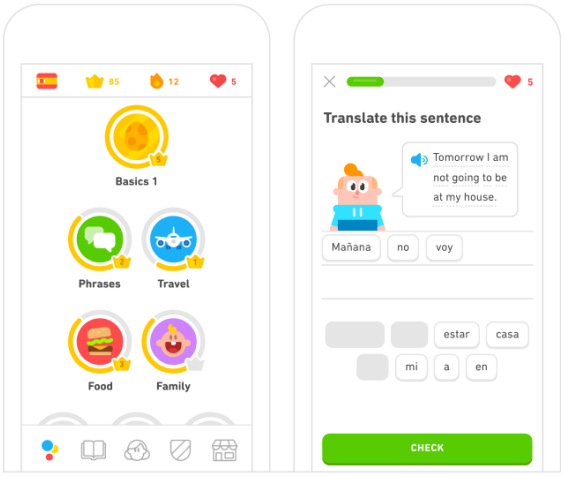

Рисунок 1.5 – Використання Duolingo

Загалом, Duolingo – це одина з найкращих платформ для вивчення англійської, проте має свої недоліки. Щоб навчальний план повістю адаптувався до користувача потрібно заплатити. У безкоштовній версії частина матеріалів є недоступними. Також є обмеження у тривалості навчання, оскільки після кількох невдалих спроб пройти тест, доступ до нього може закритись до наступного дня. У Duolingo не можна сконцетрувати навчання саме на певній навичці. До прикладу лексика вивчається разом із граматикою, немає якогось словника для повернення до вивчених слів.

Іншим схожим сервісом є «Babbel». Платформа також надає навчальний план, відповідно до рівня знань користувача. Вагомим недоліком сервісу є неможливість отримання доступу до матеріалів, що належать до вищого рівня. При ресєтарації користувачу пропонується план на рівень A1. Але щоб отримати доступ до матерілаів B1 необхідно закінчити усі попередні. Duolingo для початку вимгає проходження тесту для визначенян рівня знань користувача і відно до результатів будує навчальний план, хоч повністю цей функціонал розкривається лише у платній версії продукту.

Отже, загалом існує безліч готових рішень, вибір яких залишається за користувачем. Як не дивно, найкращі з них є платними. Варто виділити наступні тенденції, які зустрічаються у більшості розглянутих сервісах та платформах:

- заохочення до систематизації навчання;
- озвучка англійських слів;
- гейміфікація або мінімалізація процесу.

Також проаналізовані платформи можна розділити на дві групи: для самостійного навчання та для навчання з ментором або вчителем. У даному випадку все також залежить від вподобань користувача, оскільки за результатами наукових досліджень явної перевали будь-якого з методів не виявлено [13].

1.3 Технічне завдання

1.3.1 Загальні характеристики

Вебсайт школи іноземних мов «Study isn’t hard» створюється для надання освітніх послух, а саме вдосконалення навичок володіння англійської. Цільовою аудиторією проєкту є українці без конкретного вікового цензу. Вебсайт повинен надавати можливості для збільшення словникового запасу, вдосконалення слухових навичок, вивчення граматики для користувачів.

1.3.2 Архітектура розроблюваного ПЗ

Вебсайт необхідно розробити на основі клієнт-сервісної архітектури [14]. За цією моделлю сервер повинен відповідати за обробку усієї необхідної для користувача інформації. За зберігання та первинну обробку даних необхідно використовувати СУБД. Взаємодія користувача із розробленим ПЗ повинна відбуватись за допомогою веб браузера.

1.3.3 Вимоги до серверу

Сервер вебсайту повинен бути розробленим за допомогою мови програмування Java, що забезпечить можливість його розгортання на більшості операційних систем. Для ефективної розробки та підтримки серверної частини необхідно використовувати наступні бібліотеки, фреймворки та інші інструменти, що не входять у стандартну JDK:

- Spring Framework;
- Lombok;
- Liquibase;
- PostgreSQL driver;
- Hibernate;
- ModelMapper;
- Powermock;
- io.jsonwebtoken;
- Thymeleaf;
- Testcontainers.

Цей перелік включає основні інструменти, необхідні на початкових етапах розробки. Spring Framework забезпечує комплексну підтримку для створення сучасних веб-додатків, включаючи управління конфігурацією, безпекою, і підключенням до баз даних. Lombok полегшує процес написання коду, зменшуючи кількість шаблонного коду. Liquibase використовується для

управління змінами в базі даних, дозволяючи автоматизувати процеси міграції. PostgreSQL driver необхідний для інтеграції з базою даних PostgreSQL, а Hibernate забезпечує ORM (Object-Relational Mapping), що спрощує роботу з базами даних. ModelMapper використовується для полегшення мапінгу між об'єктами, Powermock — для написання модульних тестів, а io.jsonwebtoken — для роботи з JWT токенами.

Thymeleaf є шаблонним механізмом, який дозволяє створювати динамічні веб-сторінки. Testcontainers забезпечує інтеграційне тестування, дозволяючи запускати ізольовані контейнеризовані середовища для тестів.

Важливо зазначити, що за необхідності дозволено використання інших додаткових засобів для розширення функціоналу та оптимізації процесу розробки. Це може включати інші бібліотеки та інструменти, які можуть стати актуальними на наступних етапах проекту.

1.3.4 Вимоги до збереження даних

Дані, необхідні для роботи вебсайту, повинні зберігатися в базі даних. Для забезпечення ефективної розробки та обробки даних рекомендується використовувати СУБД PostgreSQL. Основні вимоги до збереження та обробки даних включають:

– Використання СУБД PostgreSQL: Всі дані вебсайту мають зберігатися в базі даних PostgreSQL. Ця СУБД обрана завдяки її надійності, продуктивності та сумісності з різними операційними системами.

– Налаштування та керування даними: Налаштування, керування та обробка даних повинні виконуватись на сервері. Це забезпечить централізований контроль над усіма операціями з даними та підвищить загальну безпеку системи.

– Відстеження змін у структурі даних: Зміни у структурі таблиць та їх вмісті необхідно відстежувати за допомогою інструменту Liquibase. Це

дозволить автоматизувати процес управління версіями бази даних і забезпечити консистентність даних під час розробки та розгортання.

– Захист конфіденційних даних: Конфіденційні дані користувачів, такі як паролі, повинні бути надійно захищеними. Для цього необхідно використовувати сучасні алгоритми хешування, такі як bcrypt або Argon2. Це забезпечить високий рівень безпеки та захистить дані від несанкціонованого доступу.

Таким чином, всі аспекти збереження та обробки даних мають бути ретельно продумані та реалізовані з урахуванням сучасних стандартів безпеки та ефективності роботи системи.

1.3.5 Вимоги до дизайну

Необхідно розробити чітку структуру вебсайту, яка забезпечить зручність користування навчальними матеріалами школи іноземних мов «Study isn’t hard». Основними вимогами до дизайну є:

– Єдиний стиль: Для всіх вебсторінок необхідно застосовувати єдиний загальний стиль, що включає використання однакових шрифтів, кольорових палітр, розміру тексту та інтервалів. Важливо забезпечити візуальну цілісність та привабливість сайту.

– Кольорова палітра: Визначити основну та додаткові кольорові палітри, які відображатимуть бренд школи та створюватимуть приємне враження у користувачів. Палітра повинна бути узгодженою та не викликати візуальної перевантаженості.

– Шрифти: Вибрати основні та додаткові шрифти для використання на всіх сторінках вебсайту. Шрифти повинні бути читабельними та відповідати загальному стилю дизайну.

– Розміри тексту та інтервали: Визначити стандарти для розмірів заголовків, основного тексту, підзаголовків та інших елементів тексту. Зазначити інтервали між рядками та абзацами для забезпечення читабельності.

– Мобільна адаптивність: Дизайн повинен бути адаптивним, забезпечуючи зручність користування як на стаціонарних комп'ютерах, так і на мобільних пристроях. Важливо передбачити зручну навігацію та доступ до всіх функцій сайту з різних пристроїв.

– Навігація: Створити інтуїтивно зрозумілу навігаційну структуру, яка дозволить користувачам легко знаходити необхідні матеріали та інформацію. Навігаційні елементи повинні бути чітко видимими та логічно структурованими.

– Візуальні елементи: Використовувати іконки, зображення та інші візуальні елементи для покращення сприйняття інформації та підвищення інтересу користувачів до навчальних матеріалів.

Результатом роботи має бути загальний шаблон дизайну вебсайту, створений у Figma. Цей шаблон повинен включати всі необхідні елементи дизайну, зокрема сторінки курсу, головну сторінку, сторінку реєстрації та входу, адміністративний інтерфейс та інші важливі розділи вебсайту.

1.3.6 Вимоги до клієнтської частини

Окрім HTML, CSS, JavaScript, для розробки клієнтської частини вебсайту необхідно використати Vue.js для побудови інтерактивних користувацьких інтерфейсів. Реалізація вебсторінок повинна відбуватись на основі дизайнерських макетів.

Додатково, необхідно забезпечити адаптивну верстку, яка дозволить вебсайту коректно відображатися на різних пристроях та роздільних здатностях екрану. Це включає використання медіа-запитів (media queries) у CSS для налаштування стилів в залежності від розміру екрану та орієнтації пристрою. Важливо також передбачити гнучку структуру елементів на сторінці, яка дозволить їм адаптуватися до різних умов перегляду, зберігаючи при цьому зручність користування та естетичну привабливість.

1.3.7 Вимоги до функціоналу вебсайту

Список функціональних вимог до вебсайту школи іноземних мов «Study isn’t hard»:

– Словник для підвищення словникового запасу користувачів.

– Окрема вкладка для вивчення граматики.

– Можливість створення додаткових наукових матеріалів (статей, тестів).

– Інструменти для покращення навичок аудіювання.

Забезпечення всіх цих функцій дозволить створити повноцінне середовище для ефективного вивчення англійської мови онлайн.

1.4 Висновок до першого розділу

У першому розділі кваліфікаційної роботи було проведено детальний аналіз предметної області. Загалом, було доведено, що високий рівень володіння англійською мовою значною мірою впливає на економічні та соціальні показники держави. Володіння англійською є важливим у дипломатії, науці, освіті та медіа, що підкреслює актуальність і необхідність створення якісного вебсайту для вивчення цієї мови.

Також було проведено аналіз існуючих рішень. Було розглянуто різноманітні підходи та методи для вивчення англійської. Програмі реалізації було розглянуто детальніше. Виявлено, що найбільш ефективними, популярними та комерційно успішними є ті платформи, які заохочують до систематизації навчання, використовують озвучку англійських слів та гейміфікацію.

На основі проведених аналізів було сформовано технічне завдання, у якому чітко описано архітектуру вебсайту, його структуру та дизайн. Визначено засоби та інструменти, необхідні для розробки.

РОЗДІЛ 2. ПРОЕКТНА ЧАСТИНА.

2.1 Розробка структури сайту і вебсторінок

Вебсайт – це єдина частина розроблюваного програмного забезпечення, з якою користувач взаємодіятиме. Тому дуже важливо створити чітку структурну модель, яка б надала швидкий доступ до усіх навчальних інструментів та матеріалів.

У технічному завданні було визначено основні функціональні вимоги до вебсайту. Почнемо із механізму зберігання та опрацювання досягнень користувачів. Для реалізації цієї вимоги необхідно реалізувати систему авторизації. Для початку вебсайт повинен містити сторінки для входу та виходу із системи. Зберігання особистих даних користувачів відбуватиметься у власному кабінеті.

Розглянемо наступну функціональну вимогу, а саме наявність словника. Для цього варто виділити окрему сторінку. Проте усю повязану із словником інформацію не вдасться розмісити в одній вкладці. Тому головну сторінку словника варто зробити збірником інформації та посилань, із доступом до:

- колекції вивчених слів;
- колекції ще не вивчених слів;
- колекції усіх слів;
- сторінки редагування даних пов’язаних із словником;
- сторінки тестування (для вивчення слів).

Узагальнена структура сторінки словника зображена на рисунку 2.1.

Рисунок 2.1 – Струтура сторінки словника

Сторінка граматики – це ще одна функціональна вимога до вебсайту. Загалом, навчальні матеріали на дану тематику можна досить чітко структурувати да типізувати. Тому на сторінці граматики потрібно розробити окремі секції для доступу до статей за темами, до прикладу: часи, частини мови, модальні дієслова тощо. Для доступу до додаткових навчальних матеріалів необхідно реалізувати окрему спеціальну секцію.

Останньою функціональною вимогою є надання можливості створення додаткових навчальних матеріалів, що представляють собою тести і статті. Надання такої можливості усім користувачам може бути небезпечним. Без контролю якості, цензури та інших факторів, створення навчальних освітніх матеріалів будь-яким користувачем, навіть авторизованим, є неможливим. Тому для реалізації даної вимоги відповідний функціонал потрібно перенести до особистого кабінету адміністратора. Таким чином дана сторінки для різних користувачів буде мати різних вигляд і структуру, що зображено на рисунку 2.2.

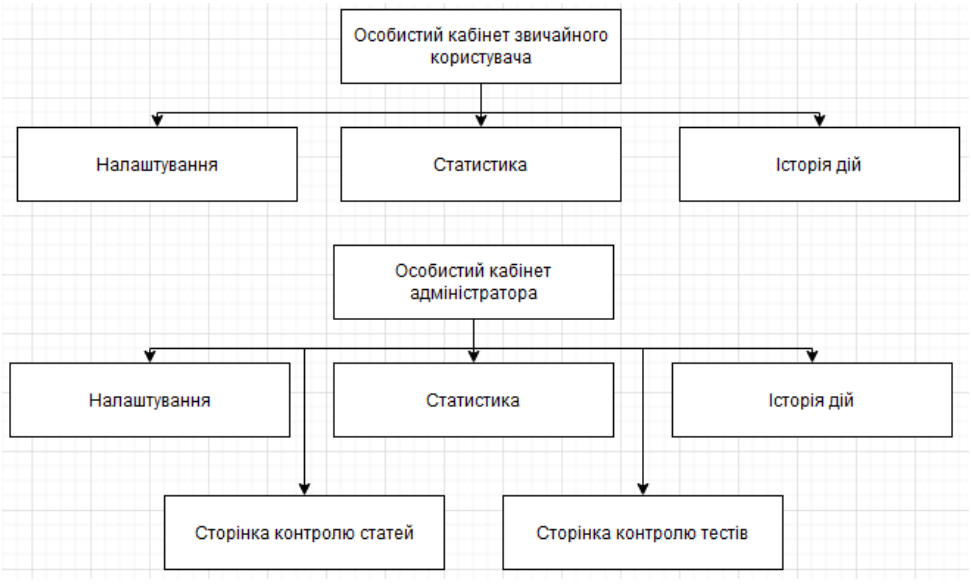

Рисунок 2.2 – Структура особистого кабінету користувачів в залежності від їхньої ролі

Описаних раніше структур буде достатньо для організації вмісту, який необхідно презентувати користувачам. Проте щоб реалізувати зручну навігацію

по вебсайту цього буде недостатньо. Розробимо меню, за допомогою якого, користувач зможе легко переміщатись між різними вебсторінками. Воно зображено на рисунку 2.3.

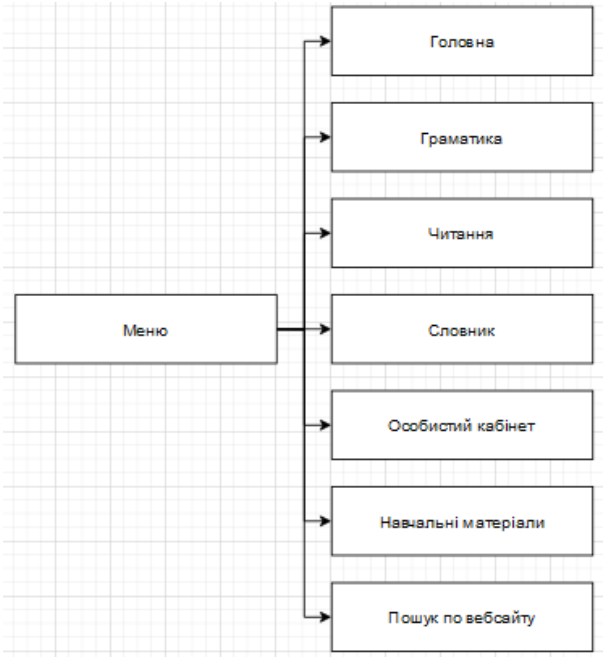

Рисунок 2.3 – Структура навігаційного меню вебсайту

Враховуючи розроблене меню, реалізованих раніше структурних елементів має бути достатньо для організації вебсайту. Із головної сторінки можна буде отримати доступ до усього доступного вмісту. Узагальнена структура вебсайту зображено на рисунку 2.4.

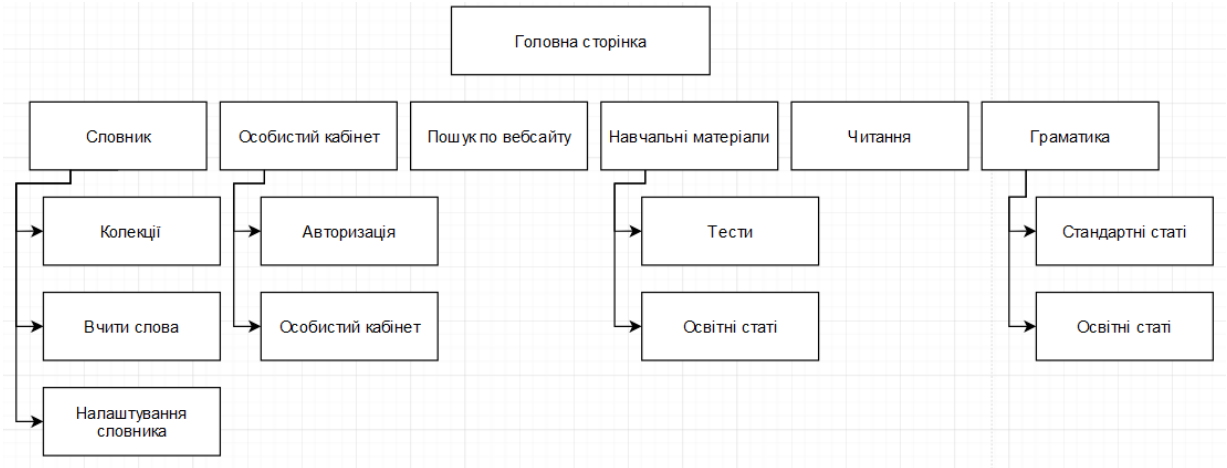

Рисунок 2.4 – Структура вебсайту

Структура вебсайту 2.4 є узагальненою. У даному розділі не буде детального опису композиції кожної вебсторінки, оскільки ця робота проводитиметься на етапі розробки дизайну. Для реалізації функціональної вимоги щодо додання нових статей та тестів будуть використовуватись динамічні сторінки, вміст і будова яких мінятиметься в залежності від вмісту. Тому їх не було зображено у загальній структурі.

2.2 Розробка дизайну

Розробка дизайну вебсайту школи іноземних мов «Study isn’t hard» починалась із формування загального стилю, що включає в себе наступні елементи:

– визначення кольорової гами;
– вибір шрифтів, їх розміру та стилізації.

Шрифти відіграють ключову роль не лише у загальному вигляді вебсайту, його читабельності та інформативності, але й у формуванні образу бренду. Типографіка створює візуальну ієрархію, що дозволяє виділяти важливі повідомлення та зменшувати помітність другорядної інформації. Це сприяє покращенню читабельності та розуміння контенту. Дослідження свідчать, що правильний вибір шрифтів може підвищити утримання користувачів більш ніж на 20% [15].

Проте у технічному завданні не було визначення вимог щодо використання певних типографій. Тому вибір впав на «Comfortaa». По-перше, це круглий геометричний шрифт, що надає тексту сучасного вигляду, що добре відповідає загальній атмосфері школи іноземних мов, де важливо створити комфортне та приємне середовище для студентів. По-друге, «Comfortaa» добре читається на різних екранах та у різних розмірах, що є критичним для забезпечення доступності контенту для всіх користувачів. Крім того, цей шрифт підтримує багато мов, у тому числі українську і англійську. Використання «Comfortaa» також сприяє загальній естетиці вебсайту,

підкреслюючи його професіоналізм та увагу до деталей. На рисунку 2.5 зображено текст, написаний за використанням даного шрифту.

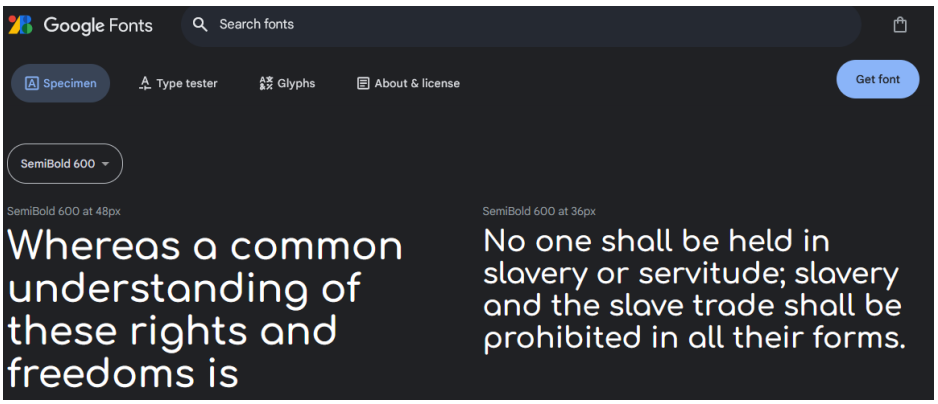

Рисунок 2.5 – Текст написаний шрифтом «Comfortaa»

У якості додаткового шрифту використовується «Inter». Цей шрифт здебільшого застосовується для офіційних даних, таких як напис «All rights reserved», а також у логотипі та інших елементах брендингу. На рисунку 2.6 зображено приклад тексту, який написаного за допомогою «Inter».

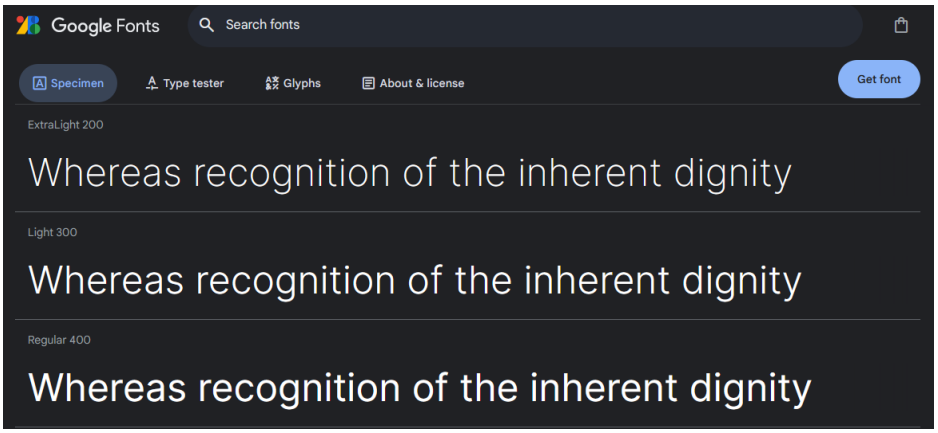

Рисунок 2.6 – Текст написаний шрифтом «Inter»

За допомогою рисунків 2.5 та 2.6 можна побачити різницю між шрифтами: строгі форми «Inter» створюють враження професійності та офіційності, що підходить для юридичних та технічних аспектів вебсайту, таких як авторські права та логотип. З іншого боку, заокруглені форми

«Comfortaa» надають дружнього та відкритого вигляду, що сприяє привабливості та комфортності для користувачів.

Тематика вебсайту школи іноземних мов «Study isn’t hard» передбачає створення атмосфери, де навчання є доступним, приємним та ефективним. Використання «Comfortaa» допомагає передати цю ідею через свої м'які та округлені форми, тоді як «Inter» додає відчуття професійності та надійності, підкреслюючи важливі офіційні повідомлення.

Розробка кольорової палітри для вебсайту "Study isn’t hard" розпочалася з детального аналізу тематики та цілей проекту, що зазначені у технічному завданні. Вибір кольорів був обґрунтований на основі досліджень, включених у статтю [16], яка досліджує психологічний вплив кольорів на людину. Наприклад, зелений асоціюється з природою та сприяє підтримці концентрації, а жовтий стимулює активність та відчуття впевненості. Загалом, існує багато наукових досліджень, що детально розглядають вплив кольорів на різні аспекти людської діяльності. Отже, на основі цих даних була сформована загальна кольорова палітра вебсайту, яка представлена на рисунку 2.7.

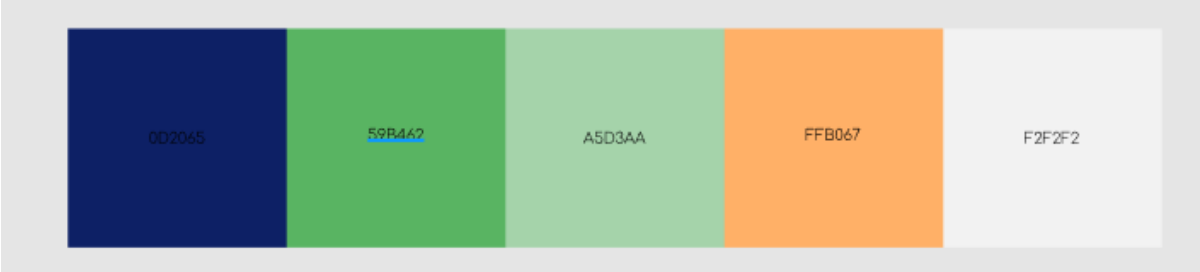

Рисунок 2.7 – Кольорова палітра вебсайту

На рисунку 2.7 кожен колір також додатково позначено відповідним шістнадцятковим кодом моделі RGB.

Далі було розроблено дизайн логотипів школи іноземних мов «Study isn’t hard». За основу було взято абревіатуру «SIH», як було стилізовано відповідно до обраної кольорової палітри. Загалом, ідей реалізації було досить багато. Відповідні макети зображено рисунку 2.8.

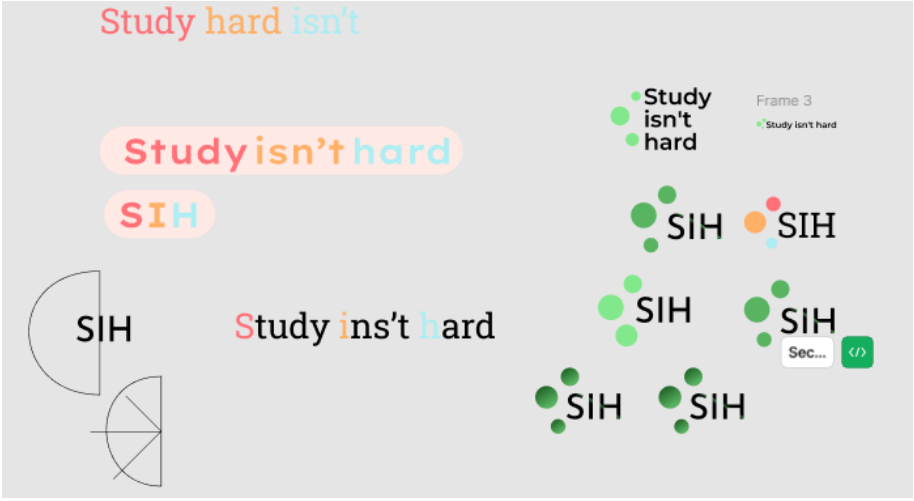

Рисунок 2.8 – Макети дизайну логотипу вебсайту

Було прийнято рішення відмовитись від повного найменування школи у логотипі, оскільки це виглядало досить громіздко. Таке лого складно масштабувати, не втрачаючи читабельності та лаконічності. Ідею логотипу із жирними, кольоровими буквами на фоні оранжевого тла також було відкинуто, зважаючи на вікову аудиторію користувачів вебсайту, що була зазначена у технічному завданні. Отже, у підсумку було обрано найбільш лаконічний, стриманий варіант. Він знаходиться у нижньому правому куті рисунка 2.8.

Далі було розроблено дизайн для найбільш вживаних компонентів вебсторінок: меню і футера. Структура навігаційної панелі була детально описана раніше. Тому на її основі було створено макет, з урахуванням визначених шрифтів на кольорової гами. Він зображений на рисунку 2.9.

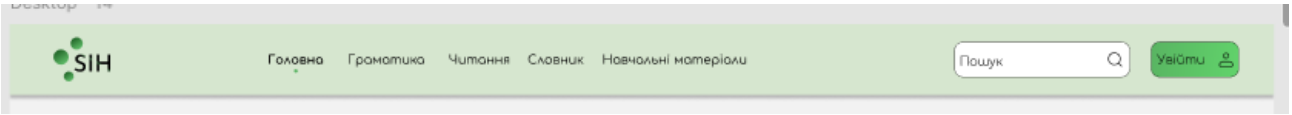

Рисунок 2.9 – Макет навігаційної панелі вебсайту

Конкретних вимог до розроби футера не було. Тому його вміст і дизайн в основному заснованих на сучасних тенденціях розробки вебсайтів. У дизайні використано контрастний синій колір, щоб розділити основний зміст вебсторінок від додаткової інформації що міститься у футері. Готовий макет

зображено на рисунку 2.10. Варто зауважити, що у футері розміщено ще не реалізовані компоненти, на кшталт посилань на соціальні мережі, або довідки.

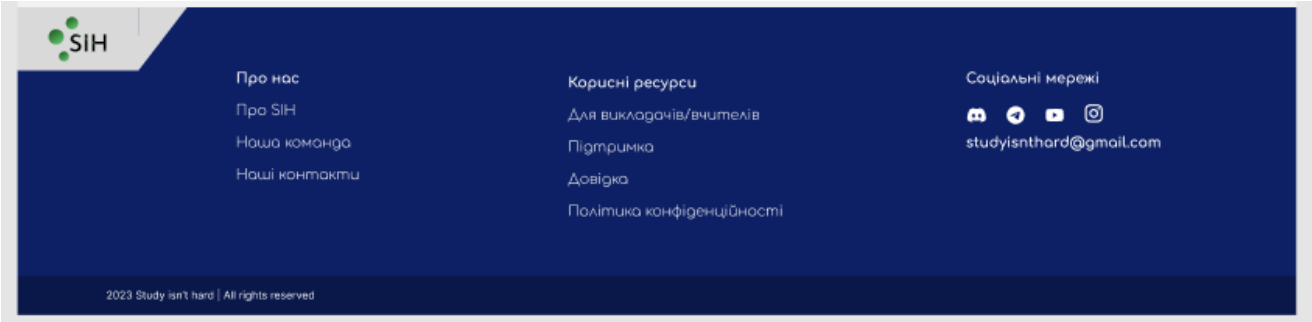

Рисунок 2.10 – Дизайн макету футеру для вебсторінок.

Щоб почати розробляти дизайн для основної частини вебсторінок, необхідно визначитись із:

– Основними найбільш вживаними компонентами (кнопки, форми тощо);

– Правилами оформлення тексту (товщина, величина шрифтів).

Правило оформлення тексту було сформовано на онові сучасних тенденцій у веброзробці [17]. Вживання усіх шрифтів наведено на рисунку 2.11.

Рисунок 2.11 – Використання визначено сітки стилізації шрифтів

Шрифти, використані на рисунку 2.11 було чітко визначено. Їх параметри та контекст використання наведено у списку:

– Основний вміст – 14 px, regular.
– Підзаголовки – 28 px, regular.
– Заголовки – 48 px, bold.
– Додаткова інформація (підтекст) – 14 px, light.

У даному випадку «regular», «light», «bold» – це товщина шрифту: звичайний, тонкий та жирний відповідно.

Далі було розроблено дизайн для кнопок та елементів форм. Строгих правил, як у випадку зі шрифтами, визначено не було. Це обґрунтовано тим, що дизайн цих елементів має бути гнучким і адаптивним, залежно від конкретних потреб користувачів та контексту використання. До прикладу наведено макети, що зображені на рисунку 2.12.

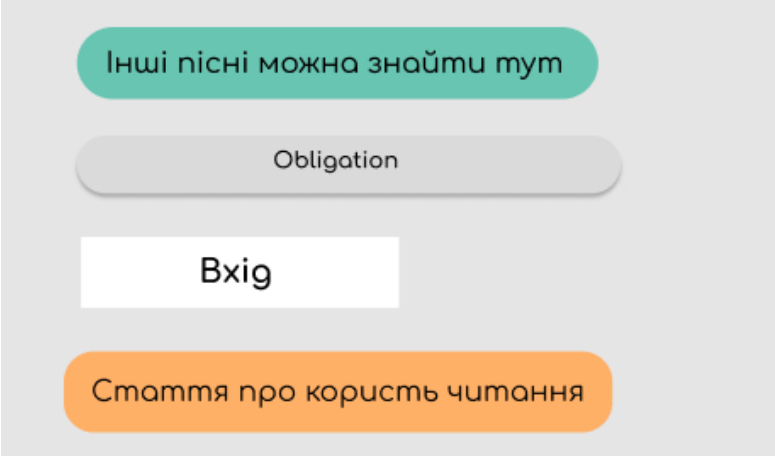

Рисунок 2.12 – Мекети кнопок вебсайту

Після розробки таких компонентів та визначення правил оформлення таких компонентів, як меню, футер, шрифти, почалась реалізація дизайну основних вебсторінок. Зазвичай при першому вході користувача, його зустрічає «Landing page». Ця сторінка призначена для першого знайомства відвідувачів з вебсайтом і містить ключову інформацію про компанію, її функціонал,

досягнення та інші важливі елементи, що захоплюють увагу користувача. Частина її макету наведено на рисунку 2.13.

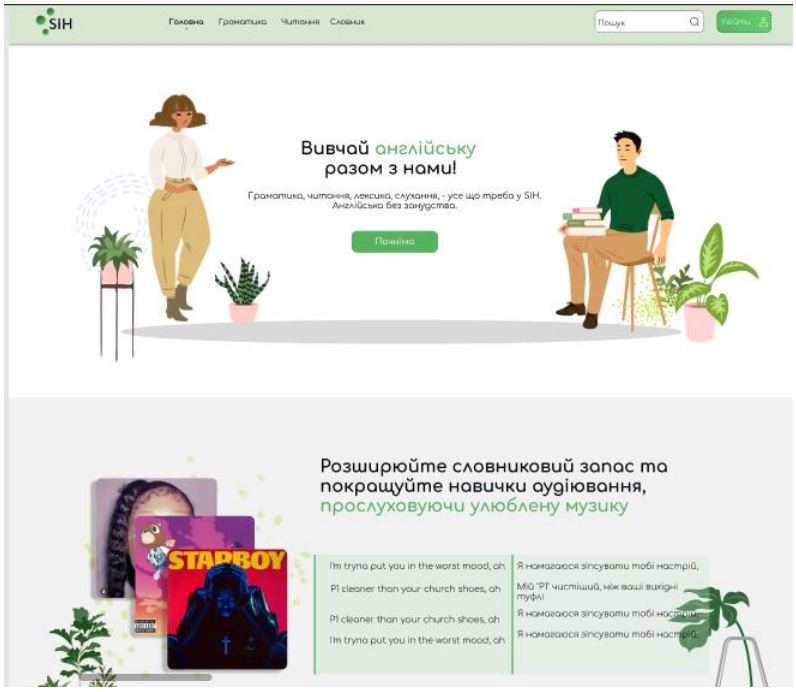

Рисунок 2.13 – Частина мекету головної сторінки неавторизованого користувача.

У дизайні сторінки 2.13 використовується зелений колір шрифту, що з першого погляду є відхиленням раніше від раніше визначених стилів. Проте, таке рішення було прийняте з метою привернення уваги користувачів, оскільки сторінка 2.13 є цільовою (landing page), основна функція якої полягає в залученні та утриманні відвідувачів. Використання яскравого зеленого кольору допомагає виділити ключову інформацію, що підвищує її помітність та сприяє кращому сприйняттю. Це особливо важливо на цільових сторінках, де кожен елемент має працювати на досягнення максимального ефекту залучення, тому такі відхилення від загального стилю цілком виправдані.

На наступному етапі було розробелно дизайн для сторінки словника, з урахуванням функціональних вимог, зазначених у технічному завданні. До

обовзякових елемнтів, такі як посилання на колекції користувача, було додано діагарму, для візуалізації успішності клієнта. Це зображено на рисунку 2.14.

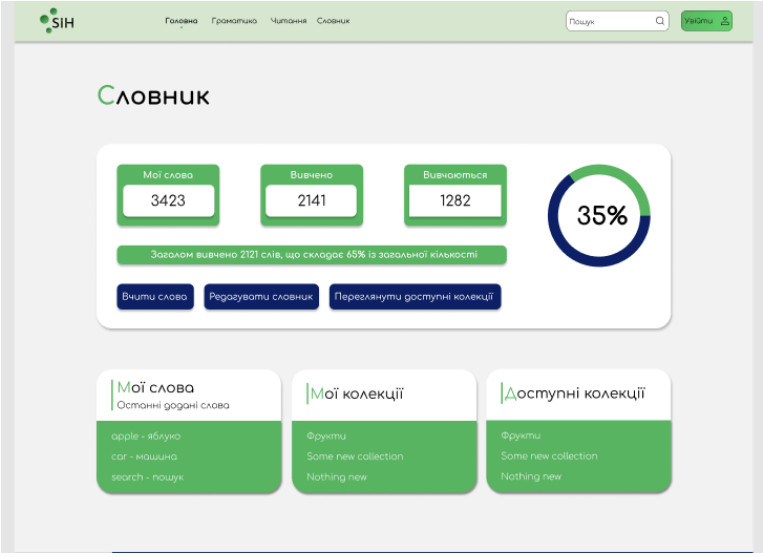

Рисунок 2.14 – Макет вебсторінки словника

У стилі, продемонстрованому на рисунках 2.14, 2.13 було розроблено дизайн для сторінок-шаблонів. Вони використовуються в якості макетів для динамічних вебсторінок, до прикладу статей або тестів. Готовий варіант зображено на рисунку 2.15. На ньому розміщено два різних макети, що накладені один на одного для кращої візуалізації.

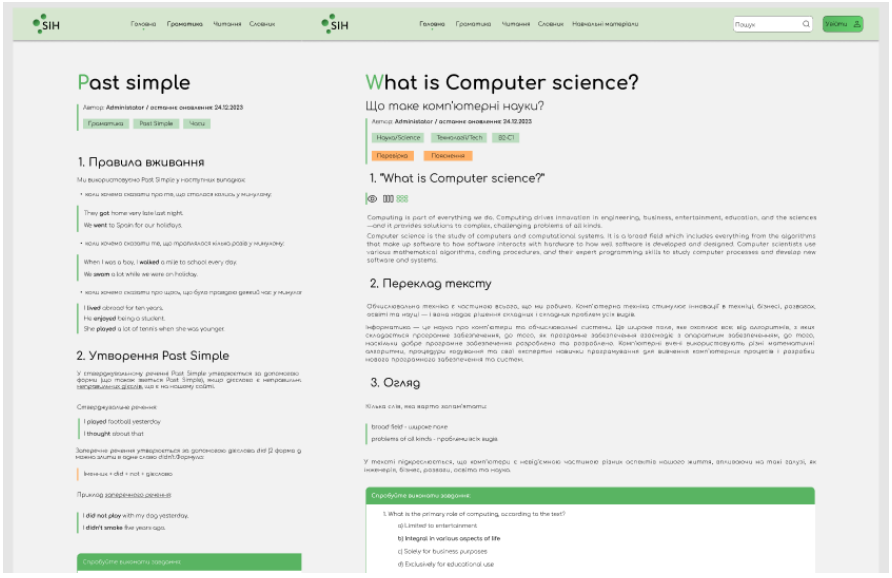

Рисунок 2.15 – Мекети динамічних вебсторінок

На рисунку 2.15 зображено два різних навчальних матеріали: публіцистичну статтю та правила вживання і формулювання речень у Present Simple. Обидва варіанти мають схожий стиль, але наповнені різними компонентами. Наприклад, у статті «What is Computer Science» використовується блок для тестування з метою оцінки розуміння та засвоєння матеріалу. У матеріалах про Present Simple додано блок тегів.

Загалом, було розроблено більше десяти макетів вебсторінок, кожен з яких враховує специфіку контенту та потреби користувачів. Готовий дизайн наведено на рисунку 2.16 у середовищі розробки Figma.

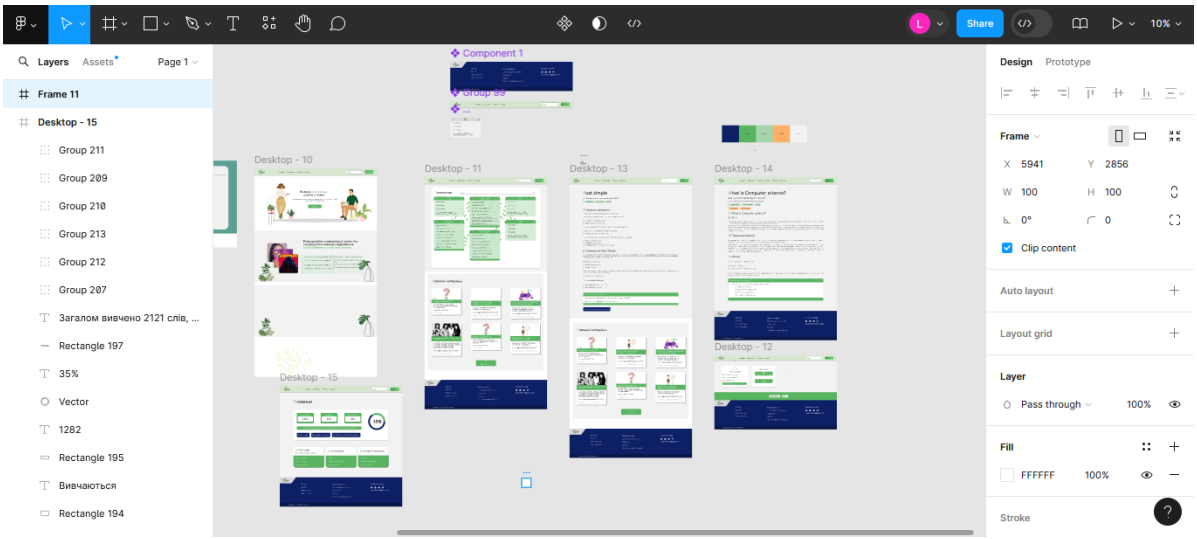

Рисунок 2.16 – Готовий дизайн вебсайту у Figma

Різноманітність розроблених шаблонів дозволяє ефективно презентувати вміст, забезпечуючи зручність навігації та інформаційність. Кожен макет ретельно опрацьований, щоб відповідати загальному стилю вебсайту..

2.3 Проєктування та розробка бази даних

У цьому розділі описується процес проектування та розробки бази даних для вебсайту школи іноземних мов «Study isn’t hard». База даних є ключовим

компонентом системи, забезпечуючи зберігання та управління даними про користувачів, навчальні матеріали та інші важливі аспекти діяльності школи.

У технічному завдання вже було визначено використання PostgreSQL – це відкра реляційна система управління базами даних. PostgreSQL була обрана через її надійність, масштабованість та відповідність вимогам проекту. Основні переваги використання PostgreSQL у порівнянні з іншими СУБД, такими як MySQL або об'єктно-орієнтовані бази даних, наведено у списку [18]:

– Масштабованість. PostgreSQL дозволяє користувачам додавати нові типи даних, функції, оператори та індекси, що забезпечує високу гнучкість у проектуванні та розробці бази даних

– Підтримка стандартів SQL. PostgreSQL дотримується стандартів SQL точніше, ніж, до прикладу, MySQL, що забезпечує сумісність з іншими реляційними СУБД і спрощує перенесення даних між системами.

– Підтримка ACID-транзакцій. PostgreSQL забезпечує повну підтримку ACID-транзакцій, що гарантує надійність і цілісність даних. Хоча MySQL також підтримує ACID-транзакції в режимі InnoDB, PostgreSQL вважається більш стабільною та надійною в цьому аспекті.

Загалом, можна виділити й інші переваги, як і недоліки. Проте зваживши усі фактори, PostgreSQL – це оптимальний вибір.

Де втілення усіх функціональних вимог, що були визначені у технічному завданні, достатньо буде використати одну базу даних. У разі необхідності, до прикладу масштабування бізнесу, або розділення вебсайту на частини, процес додання кількох БД буде спрощено за рахунок використання Java та PostgreSQL.

Для збереження особистих даних користувачів було створено таблицю «user». Вона відіграє ключову роль у системі розмежування прав доступу, автентифікації та авторизації клієнтів. У ній реалізовано наступні поля:

– «id» – унікальний ідентифікатор, що слугує зовнішнім ключем для інших таблиці.

– «username» - унікальне ім’я користувача.

– «password» - поле для зберігання захешованого пароль користувача.
– «email» - електронна пошта користувача.
– «user_status» - статус облікового запису.
– «date_of_registration» - дата реєстрації.
– «last_activity» - дата останньої активності.
– «refresh_token_key» - ключ оновлення токену.
– «role» - роль користувача.

Наведені поля досить тісно пов’язані із програмною реалізацією вебсайту. До приладку, використання «refresh_token_key», або ж хешування паролю неможливо пояснити, або ж аргументувати без контексту коду серверу. Про це буде детально описано у відповідному розділі даної кваліфікаційної роботи. Більшість полів таблиці є типом «VARCHAR», відрізняючись розміром. До прикладу, «username» обмежено довжиною у двадцять символів, в той час як поле «refresh_token_key» може містити двісті двадцять п’ять символьних елементів. Вирізняються поля «id», що може містити лише ціле цифрове значення. Поля «date_of_registration» та «last_activity» зберігають данні у типі «TIMESTAMP», тобто форматі дати й часу [19].

На початкових етапах розробки також було реалізовано таблиці що містила доступні ролі користувачів та статуси облікових засобів. Проте під програмної реалізації серверної частини, цю ідею було відкинуто, з метою оптимізації швидкості опрацювання. Ці таблиці містини данні у вигляді ключ – значення, де поле «id» відповідало за ідентифікацію та слугувало зовнішнім ключем, а поле «value» зберігало роль або статус. Відповідна структурна схема зображена на рисунку 2.17.

Проте даний підхід є досить неефективним. Кожного разу, коли серверу знадобиться отримати дані користувача, потрібно буде роботи дві додаткові транзакції, або ускладнювати SQL-ін’єкцію об’єднанням таблиць [20]. Це збільшить час оброки кожного запиту та ускладнить бізнес-логіку програмної реалізації. Такий підхід може бути ефективним у випадку великої кількості різноманітної за змістом інформації. Проте система розмежування прав доступу

вебсайту пердбачає використання лише трьох ролей та такої ж кількості статусів облікового запису.

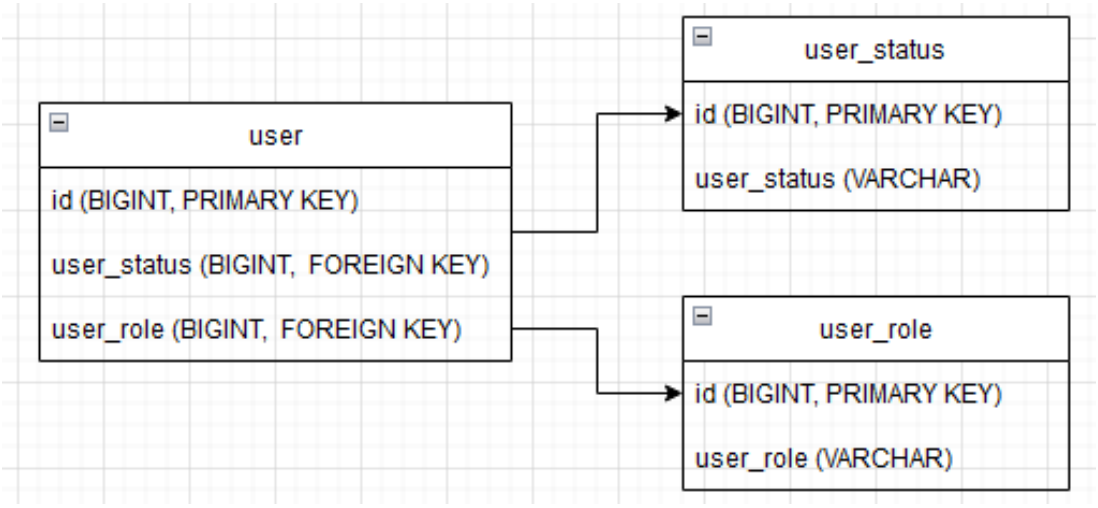

Рисунок 2.17 – Структурна схема звязку між таблицями «user», «user_status», «user_role»

У структурній схемі таблиці «user», що зображено 2.17 опущено більшість реалізованих полів для кращої репрезентації.

Таблиця «test» розроблена для зберігання даних пов’язаних із тестами. І ній містяться наступні поля:

– «id» - унікальний ідентифікатор та зовнішній ключ;
– «test_name» - назва тесту;
– «user_id» - посилання на автора;
– «test_description» - опис тесту.

Ця таблиця описує загальну інформацію про тест. Для зберігання вмісту такого, як правильні і неправильні відповіді створена окрема сутність. Це таблиця під назвою «question», у якій реалізовано наступні поля (далі опису для поля «id» не буде, оскільки його значення завжди ідентичне для кожної таблиці):

– «id»;
– «test_id» - посилання на тест, до якого належить запитання;
– «question_text» - повний текст запитання;

– «correct_answer» - поле, що вказує на правильність відповіді.

На рисунку 2.18 зображено спрощену структурну схему звязку між таблицями «question», «user» та «test». Як і у випадку з рисунком 2.17, це зроблено з метою покращення візуалізації.

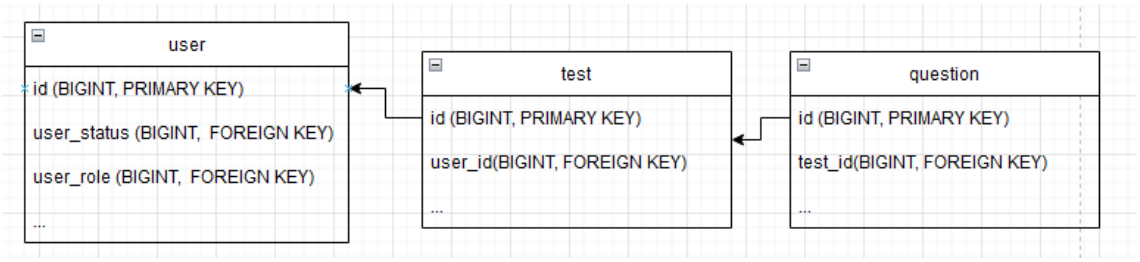

Рисунок 2.18 – Структурна схема звязку таблиць «question», «user» та «test»

Між таблицям «question», «user» та «test» використовується зв'язок один до багатьох. Тобто один користувач може бути автором багатьох тестів, які в свою чергу можуть мати необмежену кількість запитань.

Для організації та структуризації навчальних матеріалів було реалізовано систему тегів, які зберігаються в окремій таблиці під назвою «tag», яка містить наступні поля:

– «id»;

– «tag_name» - значення тегу, до прикладу: «Граматика», «Музика»;

– «tag_type» - тип навчального вмісту, до якого тег може бути застосовано.

Поле «tag_type» таблиці «tag» є посиланням на іншу таблицю. Структура останньої є схожою до «user_status», тому детально описуватись вона не буде.

Для зберігання інформації, повязнаю із навчальними статтями користувачів реалізовано окрему групу таблиць:

– «article»;

– «article_additional»;

– «article_tag».

Перша у списку таблиця є ключовою. У ній зберігається детальна інформація про статтю. Реалізовані у таблиці поля:

– «id»;
– «article_name» - назва, або заголовок статті;
– «article_desc» - опис статті;
– «article_data» - вміст статті;
– «user_id» - посилання та автора статті.

Вміст статті зберігається у вигляді HTML коду, який пізніше вставляється у шаблон динамічної сторінки. Більшість полів зберігають дані у форматі «VARCHAR». Винятком є поля «id» та «user_id», у яких інформація представлена у цілих числах.

У допоміжних таблицях зберігаються додаткові дані про статтю. До прикладу, в «article_additional» міститься інформація про дату створення, редагування. А у таблиці «article_tag» зберігаються теги, закріплені до статті. Спрощена структурна схема звязку між «article_additional», «user», «article», «article_tag», «tag», «tag_type» наведено на рисунку 2.19.

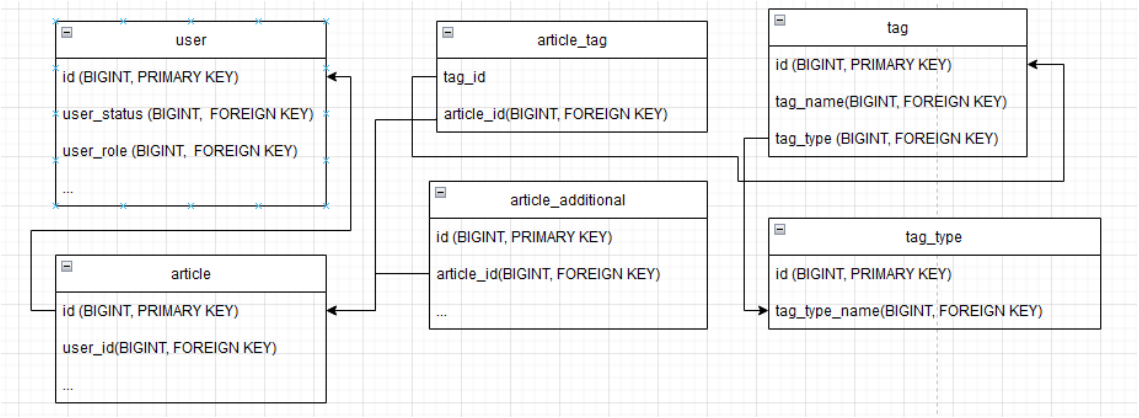

Рисунок 2.19 – Структурна схема зв’язку між таблицями «article_additional», «user», «article», «article_tag», «tag», «tag_type»

Загалом, для збереження вмісту вебсайту було створено більше десяти таблиць. Найважливіші з них було детально описано та проаналізовано. Структурі схеми, зображені на рисунках 2.19, 2.18 наведено поля таблиць, які безпосередньо впливають на зв'язок між ними. Також було візуалізовано тільки

групи взаємопов’язаних таблиць за спільною тематикою. Тому на рисунку зображено спрощену структурну схему усієї бази даних.

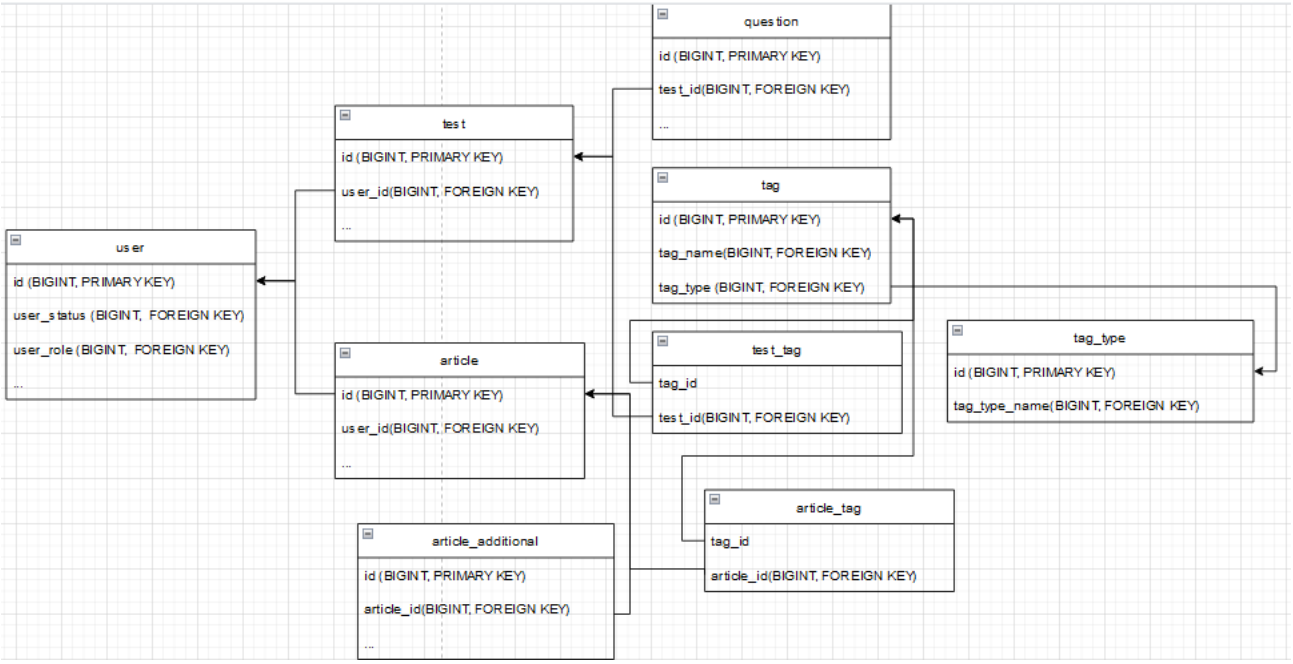

Рисунок 2.20 – Структура схема бази даних вебсайту школи іноземних мов «Study isn’t hard»

У таблицях, зображених на рисунку 2.20 також упущено більшість полів.

2.4 Висновок до другого розділу

Розробки структури сайту та вебсторінок школи іноземних мов "Study isn't hard" вимагає детального підходу до організації контенту та забезпечення зручної навігації для користувачів. Враховуючи вимоги технічного завдання, було визначено основні функціональні блоки сайту, що включають систему авторизації, словник, граматичні матеріали та інструменти для створення додаткових навчальних матеріалів. Кожен з цих блоків має свої особливості та вимагає різного підходу до розробки [21].

Перший етап розробки дизайну вебсайту включав визначення загального стилю, вибір кольорової гами та шрифтів. Використання шрифтів "Comfortaa" та "Inter" сприяє створенню сучасного та професійного вигляду сайту,

забезпечуючи читабельність та зручність для користувачів [22]. Кольорова палітра була обрана з урахуванням психологічного впливу кольорів, що сприяє освітньому процесу, в тому числі підтримувати концентрацію та зосередженість учнів. Логотипи та основні компоненти вебсторінок, такі як меню і футер, були розроблені з урахуванням сучасних тенденцій вебдизайну та специфіки цільової аудиторії. На наступному етапі було створено макети основних вебсторінок, що наведені у списку:

– головна;
– словник;
– граматика;
– динамічні шаблони статей.

Дизайн даних вебсторінок було представлено на рисунках 2.13, 2.14, 2.15 відповідно у порядку, наведеному у списку вище. Детально описано використання різноманітних компонентів (рисунок 2.12), чітко визначені шрифти (рисунок 2.7) та кольорова схема (рисунок 2.6 та 2.5). Сторінки, які не були наведені у списку, детально не описувались, але були наведені на рисунку 2.16, на якому зображено більшу частину макетів.

Використання PostgreSQL для реалізації бази даних було краще обґрунтовано у даному розділі кваліфікаційної роботи. Структура БД включає таблиці для зберігання інформації про користувачів, навчальні матеріали та інші важливі дані. Основною таблицею є «user», що містить поля для зберігання особистих даних користувачів, забезпечуючи автентифікацію та авторизацію клієнтів. Використання хешованих паролів та ключів оновлення токенів підвищує безпеку зберігання даних. Завдяки масштабованості PostgreSQL, система зможе легко адаптуватись до зростання обсягу даних та потреб користувачів. Структурна схема, зображена ну рисунку 2.20, показує основні зваємо зв’язки між таблицями. Версія схеми зі всіма полями винесено у додаток А.

РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Верстка вебсторінок

Перший етап програмної реалізації полягає у верстці вебсторінок. Вона є ключовим процесом, який забезпечує візуальну привабливість та зручність користування сайтом. Вона включає в себе використання HTML для структурування контенту та CSS для його стилізації [23].

Для початку було реалізовано дизайнерські вимоги щодо шрифтів, кольорів та компонентів. Код, пов'язаний із стилізацією тексту, розміщено в окремому файлі «fonts.css». Усі шрифти, визначені у пункті 2.2 даної кваліфікаційної роботи, були встановлені, як у прикладі, наведеному у лістингу 3.1.

Лістинг 3.1 – CSS реалізація шрифтів для тексту

```
 .font-header {
  font-size: 32px;
  font-family: 'Comfortaa';
  font-style: normal;
  font-weight: 500;
}
.font-title {
  font-size: 28px;
  font-family: 'Comfortaa';
  font-style: normal;
  font-weight: 500;
}
```

Назви для класів стилю були підібраними відповідно зазначених шрифтів у дизайнерському макеті. Також у файлі «fonts.css» додатково реалізовано поведінку класів у відповідності до розміру екрану. Це продемонстровано у лістингу 3.2.

Лістинг 3.2 – Адаптивність шрифтів до розміру екрану

```
@media (max-width: 768px) {
  body {
```

```
    font-size: 12px;
  }
  .font-header {
    font-size: 24px;
    font-family: 'Comfortaa';
    font-style: normal;
    font-weight: 500;
  }
  .font-title {
    font-size: 20px;
    font-family: 'Comfortaa';
    font-style: normal;
    font-weight: 500;
  }
```

Між лістингами 3.1 та 3.2 можна побачити чітку різницю: розмір шрифтів зменшується на вісім пікселів для екранів, із шириною меншою за 768 пікселів [24]. І у схожому стилі було реалізовано усі інші стилі для тексту. На рисунку 3.1 продемонстровано використання розробленого «fonts.css» файлу.

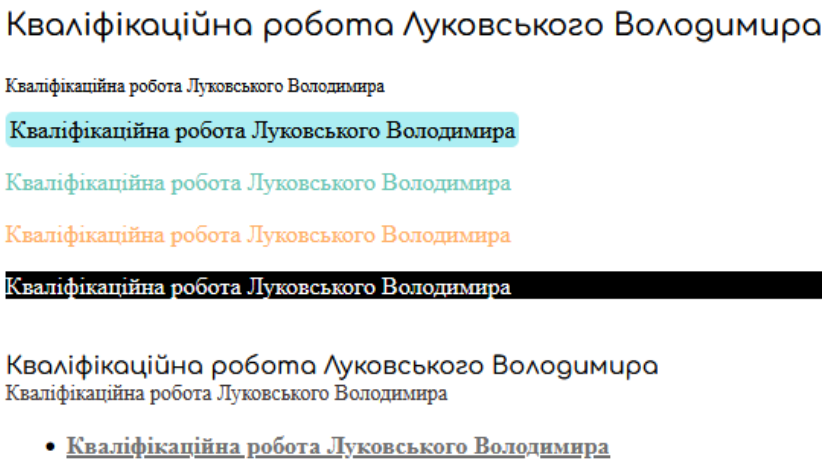

Рисунок 3.1 – Використання розроблених стилів для шрифтів

Далі реалізація верстки відбувалась у порядку розробки дизайну: футер і меню. У лістингу 3.3 наведено код навігаційної панелі.

Лістинг 3.3 – HTML структура меню

```
<div class="nav-wrapper">
        <div class="nav-logo">
            <img :src="/assets/SIH_nav_logo.svg" alt="">
        </div>
```

```
<div class="nav-links" id="nav-links" v-show="isListVisible">
            <ul class="nav-ul">
            </ul>
</div>
        <div class="nav-inputs-wrapper">
<div class="nav-searching-input-wrapper">
                <input    type="text"    class="nav-searching-input"
placeholder="Пошук">
            </div>
            <div class="nav-user-profile">
                <button   class="nav-login-btn"   href="google.com"
type="button">
                </button> …
```

У лістингу 3.3 наведено лише структуру меню. Контейнер з класом «nav-wrapper» є обгорткою для всього вмісту навігаційної панелі. У ньому містяться наступні блоки-обгортки:

– «nav-logo» - відповідає за відображення логотипу школи;

– «nav-links» - контейнер, у якому знаходяться посилання для навігації по вебсайту;

– «nav-inputs-wrapper» - компонент із кнопкою для авторизації, доступу до особистого кабінету.

Далі було розроблено файл «navbar.css» для стилізації меню. На рисунку 3.2 зображено навігаційну панель, до якої не додано CSS.

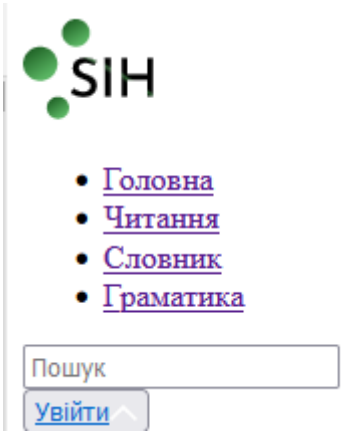

Рисунок 3.2 – Розроблена навігаційна панель без стилізації

У лістингу 3.4 наведено код класу «nav-wrapper» із файлу «navbar.css», що є обгорткою для всього меню.

Лістинг 3.4 – Код класу «nav-wrapper» із файлу «navbar.css»

```
.nav-wrapper {
    width: 100%;
    height: 50px;
    background: rgba(211, 228, 203, 0.99);
    box-shadow: 0px 4px 4px 0px rgba(0, 0, 0, 0.25);
    display: flex;
    flex-direction: row;
    flex-wrap: nowrap;
    position: sticky;
    top: 0;
    z-index: 100;
    justify-content: space-around;
}
```

Більшість структурних елементів розроблених вебсторінок використовують flex-контейнери. Використання Flexbox забезпечує адаптивне вирівнювання та розподіл простору між елементами в межах контейнера. Це дозволяє створювати більш гнучкі та динамічні макети, які легко адаптуються до різних розмірів екрану і пристроїв .

На рисунку 3.3 показано навігаційну панель після застосування стилів з файлу «navbar.css», що надає меню сучасний і привабливий вигляд.

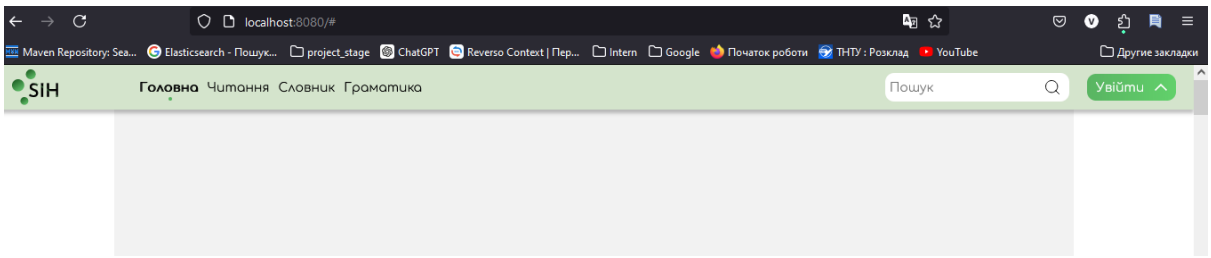

Рисунок 3.3 – Стилізована навігаційна панель

Завдяки використанню flex-контейнерів, навігаційна панель автоматично підлаштовується під різні розміри екрану, зберігаючи свою структуру і функціональність. Це значно покращує користувацький досвід, оскільки елементи меню залишаються доступними і зручними для використання на

будь-якому пристрої. Проте для малих за розміром екранів, посилання різні вебсторінки буде виплавати за межі браузеру, щоб зробить їх недоступними. Для таких випадків було розроблено спеціальну версію меню, що зображена на рисунку 3.4.

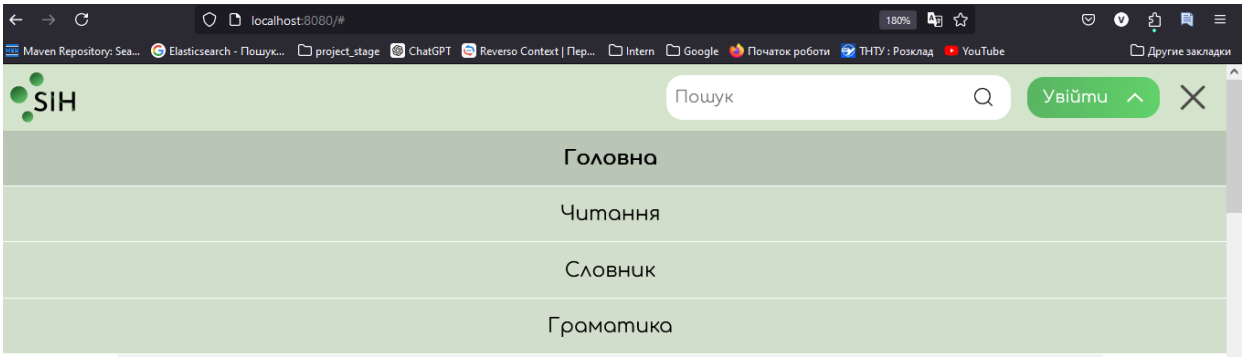

Рисунок 3.4 – Навігаційна панель на екранах, шириною менше 768 px

Детальна реалізація випадаючого списку, зображеного на рисунку 3.4, буде наведена у розділі 3.2 даної кваліфікаційної роботи, оскільки для цього було використано JavaScript, Vue.js Framework, що не відносяться до верстки вебсторінок [25].

Футер – невід'ємна частина вебсайту, яка містить інформацію про авторські права, контактні дані, посилання на політику конфіденційності та інші важливі посилання (рисунок 2.10). У лістингу 3.5 наведено HTML структуру футера [26].

Лістинг 3.5 – HTML структура футеру

```
<div class="footer-wrapper">
        <div class="footer-upper-wrapper">
        </div>
        <div class="footer-main-section-wrapper">
               <div class="footer-main-content-wrapper">
                      …
                </div>
      <div class="footer-main-content-wrapper">
                      …
                </div>
      <div class="footer-main-content-wrapper">
                      …
```

```
                </div>
            </div>
            <div class="footer-bottom-wrapper">
        <p class="footer-lince">…</p>
            </div>
</div>
```

Загальний стиль написання коду, зокрема використання BEM-методології (Block, Element, Modifier), забезпечує зрозуміле і логічне іменування класів. Структура HTML забезпечує зрозумілий і чіткий розподіл елементів. Це підтверджується раніше веденими лістингами 3.1-3.5.

Загалом, далі було розроблено HTML макети інших вебсторінок із вмістом, що був визначений дизайнерськими шаблонами. На рисунку 3.6 зображено загальну HTML структуру головної сторінки.

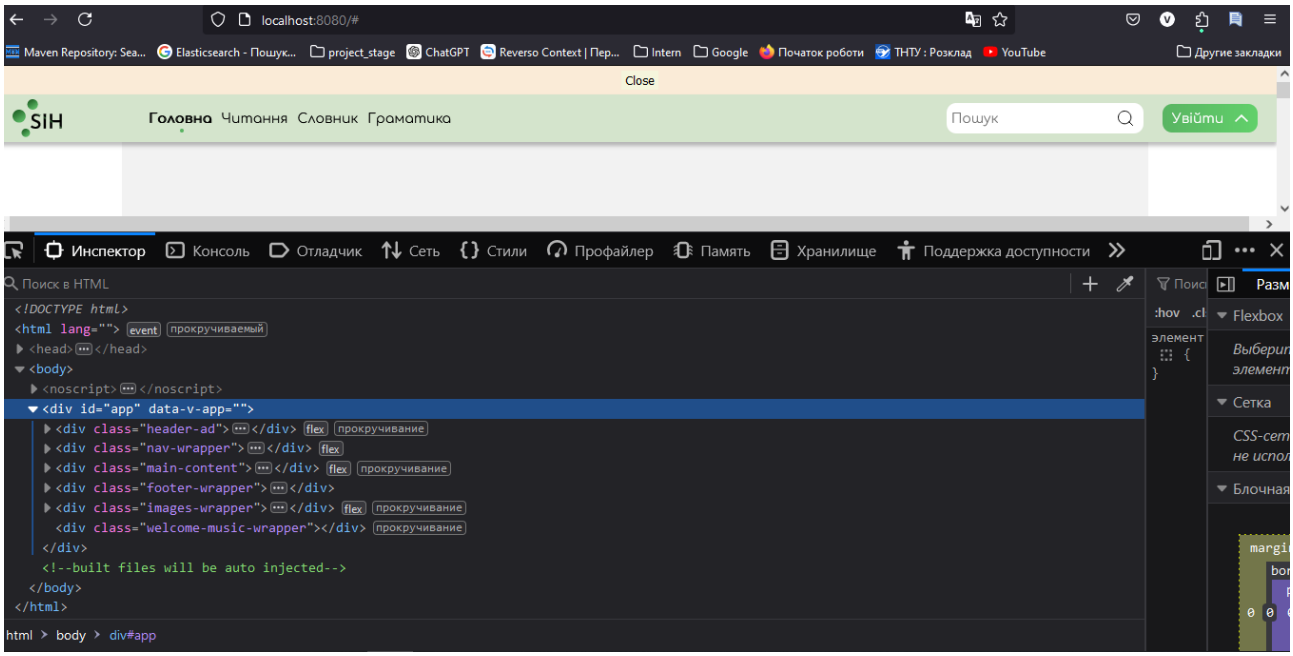

Рисунок 3.6 – Структура головної сторінки

Структура, наведена на рисунку 3.6, було задукоментовано за допомогою інструментів розробника у браузері FireFox [27]. Усі інші сторінки побудовані аналогічно, за виключенням блоків, з класами «images-wrapper» та «welcome-music-wrapper». Це тестові контейнери, що використовувались лише при розробці головної сторінки.

У підсумку було створено більше п’ятнадцяти різних за вмістом та призначенням шаблонів. Файлова структура проєкту наведена на рисунку 3.7. Назви шаблонів вказують на їх функціональне призначення, що полегшує орієнтацію в проєкті.

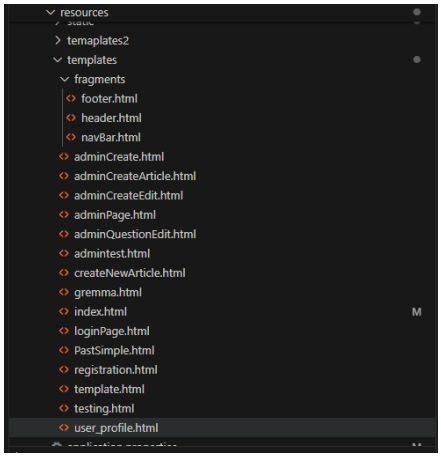

Рисунок 3.7 – Файлова структура розроблених HTML файлів

Для покращення структури та повторного використання коду, фрагменти, які застосовуються на всіх вебсторінках, були винесені в окремі файли, що знаходяться в директорії «fragments».

Матеріали, що використовуються у HTML сторінках було розміщено у директорії static. В основному це картинки, CSS файли. Їх структуру зображено на рисунку 3.8.

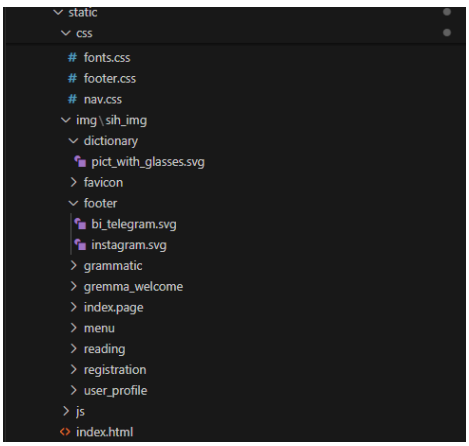

Рисунок 3.8 – Файлова структура елентів, що використовуються у HTML шаблонах вебсторінок

До структурних частин вебсторінок, які були фрагментовані, належать: меню, футер, посилання на CSS стилі. Розділення коду на фрагменти дозволяє чітко розділити відповідальності між різними частинами коду, що робить його більш організованим.

На даному етапі було зверстано макети усіх вебсторінок, наведених у розділі 2.2 даної кваліфікаційної роботи.

3.2 Розробка серверу

Сервер є ключовим компонентом вебсайту, оскільки саме на ньому формується і обробляється вся інформація. Клієнтська частина сайту відповідає лише за відображення інформації, яку надсилає сервер. База даних також не є самостійною, оскільки вона є місцем зберігання даних і використовується сервером для зберігання, оновлення і вибору інформації для подальшої обробки [28].

Для формування шаблону серверу було використано вебресурс Spring Initializr, що надав можливість швидко створити початкову конфігурацію проєкту на основі Spring Framework [29]. Цей інструмент дозволив легко налаштувати проєкт з необхідними залежностями та структурою каталогів, що наведена на рисунку 3.9.

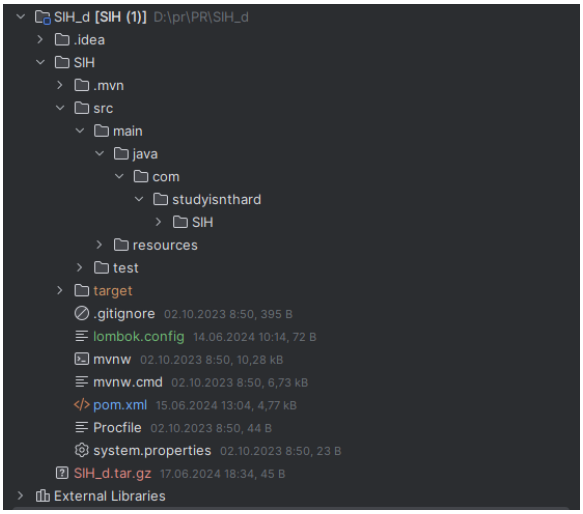

Рисунок 3.9 – Файлова структура серверу, згенерована завдяки Spring Initializr

Для керування та автоматизації проєкту обрано Maven. Його основний конкурент, Gradle, використовує JSON для конфігураційних файлів. Отже, з огляду на особисті вподобання було обрано Maven, оскільки він застосовує XML. Обидва інструменти надають аналогічний функціонал.

Maven дозволяє здійснювати керування залежностями проекту, автоматизувати процес збирання, тестування, пакування та розгортання програмного забезпечення. Він також забезпечує структуру проєкту, дозволяє визначати та виконувати цільові задачі (goals), які описані у файлі pom.xml [30].

Для пошуку "artifactid", що є ідентифікатором ресурсів для Java, використовувався Maven Repository. Це централізоване сховище забезпечує доступ до тисяч бібліотек, плагінів та інших компонентів програмного забезпечення, які можна інтегрувати у проекти Maven. У лістингу 3.6 наведено код, що відповідає за імпортування бібліотек, що не вдалось додати за допомогою Spring Initializr.

Лістинг 3.6 – Імпорт необхідних для розробки бібліотек у файлі pom.xml

```
<dependency>
     <groupId>io.jsonwebtoken</groupId>
     <artifactId>jjwt</artifactId>
     <version>0.12.3</version>
</dependency>
<dependency>
     <groupId>org.modelmapper</groupId>
     <artifactId>modelmapper</artifactId>
     <version>3.1.1</version>
</dependency>
<dependency>
     <groupId>jakarta.xml.bind</groupId>
     <artifactId>jakarta.xml.bind-api</artifactId>
     <version>4.0.0</version>
</dependency>
```

У якості основного середовища розробки було використано IntelliJ IDEA Ultimate Edition. Дана IDE надає безліч корисних функцій, що значно пришвидшують розробку. До прикладу, Maven не надає графічного інтерфейсу.

Таким чином додання будь-якої нової бібліотеки буде вимагати записку відповідної команди через консоль. Проте завдяки IntelliJ IDEA це можна зробити за допомогою двох кнопок, що продемонстровано на рисунку 3.10.

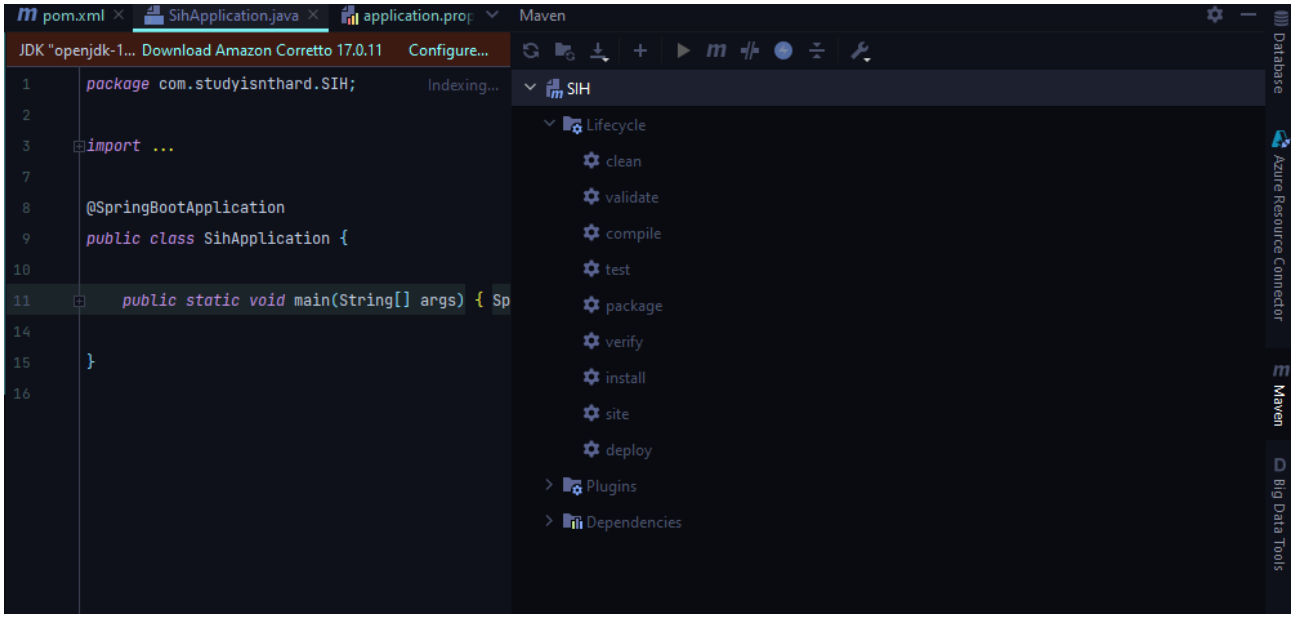

Рисунок 3.10 – Вікно для роботи із Maven у IntelliJ IDEA Ultimate Edition

Додані раніше бібліотеки вимагають налаштування конфігурації для конкретної роботи. Тому спершу було створено файл «lombok.config» у кореневій директорії проєкту. Це стандарте розташування, що визначено у документації Project Lombok. На офіційній сторінці бібліотеки описано причини для її застосування, цитата перекладено українською: *«Project Lombok — це бібліотека Java, яка автоматично підключається до інструментів редактора та збірки, покращуючи вашу Java. Ніколи більше не пишіть інший метод отримання чи дорівнює, з однією анотацією ваш клас має повнофункціональний конструктор, автоматизує ваші змінні журналювання та багато іншого»* [31]. Далі наведено лістинг файлу конфігурації «lombok.config»

```
lombok.addLombokGeneratedAnnotation = true
lombok.accessors.chain = true
```

Перший параметр вказує Lombok додати анотацію @Generated до автоматично генерованого коду, яка використовується для позначення класу або методу, що були автоматично згенерований і не повинні редагуватися вручну користувачем. Другий параметр вмикає функціонал ланцюжкових методів для Lombok, що дозволяють викликати послідовно кілька методів на одному об'єкті, що полегшує читання та написання коду.

LiquiBase є ще однією важливою бібліотекою у проєкті, яка використовується для керування версіями бази даних. Вона дозволяє автоматизувати процеси змін в структурі бази даних через код, що робить розгортання та управління схемою даних більш простими та надійними. LiquiBase не вимагає створення окремого файлу для конфігурації. Для цього використовується «application.properties». Це стандартний конфігураційних файл, що генерується по замовчуванню у проєктах Spring Framework. У лістингу 3.7 наведено параметри, встановлені для PostgreSQL та LiquiBase.

Лістинг 3.7 – Конфігурація для PostgreSQL та LiquiBase у «application.properties»

```
spring.datasource.url=jdbc:postgresql://localhost:5432/sih
server.port=${PORT:8080}
spring.datasource.username=postgres
spring.datasource.password=root
spring.jpa.show-sql=true
spring.jpa.generate-ddl=false
spring.jpa.hibernate.ddl-auto=validate
spring.datasource.driver-class-name= org.postgresql.Driver
spring.jpa.properties.hibernate.jdbc.lob.non_contextual_creation=t
rue
spring.main.allow-circular-references=true
#spring.jpa.show-sql: true
#LiquiBase configuration
spring.liquibase.change-log=classpath:/db/db.changelog-master.xml
spring.liquibase.enabled=true
```

ChangeLog у LiquiBase є основним концептом, що описує всі зміни, які потрібно застосувати до бази даних. Це XML-файл (у лістингу 3.7 це файл із назвою «db.changelog-master.xml»), який містить опис кожного кроку, які

потрібно виконати для зміни структури бази даних. При запуску додатку Spring Framework, LiquiBase автоматично зчитує реалізовану конфігурацію і перевіряє її відповідність до стану бази даних. На рисунку 3.11 наведено файлову структуру, реалізовану для управління таблиць БД.

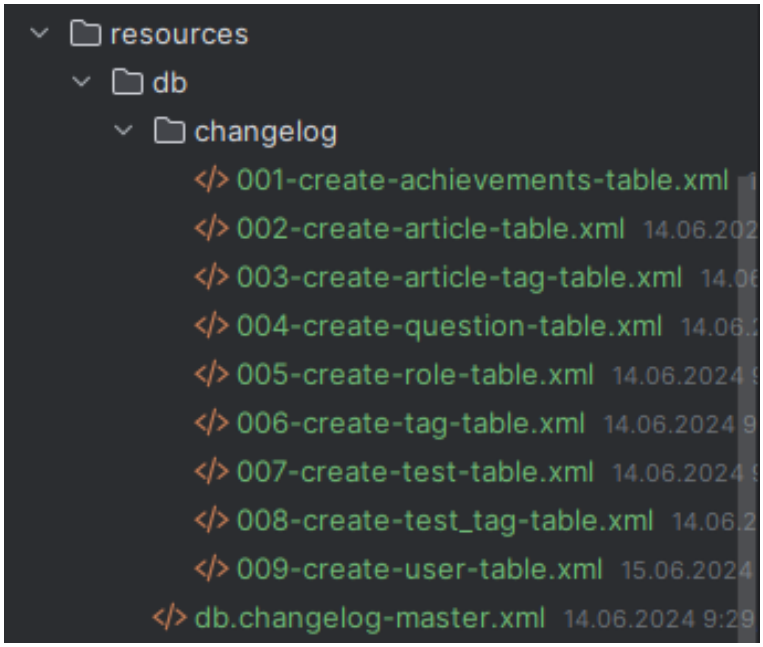

Рисунок 3.11 – Структура файлів, за допомого яких LiquiBase контролює базу даних

Hibernate є ORM-інструментом, який використовується у сервері школи іноземних мов для мапування об'єктів Java на таблиці бази даних [32]. Він автоматично генерує SQL-запити для взаємодії з базою даних, спрощуючи роботу з об'єктною моделлю. Кожна таблиця, описана у розділі 2.3 кваліфікаційної роботи, має відповідне представлення у вигляді Java-класу, що називається «Entity». До прикладу наведено лістинг 3.8, у якому представлено код із файлу «User.java», що розміщений у пакеті «Entity». Кожне поле класу «User» відповідає полю у таблиці бази даних.

Лістинг 3.8 – Поля класу «User.java»

```
@Id
@GeneratedValue(strategy = GenerationType.IDENTITY)
private Long id;
```

```
@Column(nullable = false, length = 20, unique = true, name =
"username")
private String userName;
@Column(nullable = false, name = "password")
private String passWord;
@Column(nullable = false, unique = true)
private String email;
@Enumerated(value = EnumType.STRING)
@Column(name = "user_status")
private UserStatus userStatus;
@Column(nullable = false, name = "date_of_registration")
private LocalDateTime dateOfRegistration;
@Column(name = "last_activity")
private LocalDateTime lastActivity;
@Column(nullable = false, name = "refresh_token_key")
private String refreshTokenKey;
@Enumerated(value = EnumType.STRING)
@Column(nullable = false)
private Roles role;
```

У коді класу «User.java», призначеного для мапування на таблицю бази даних за допомогою Hibernate, використовуються анотації для визначення взаємозв'язку між Java-об'єктами і записами бази даних [33]. Наприклад, анотація @Id вказує на те, що поле id є унікальним ідентифікатором сутності, яке автоматично генерується. Анотація @Column використовується для визначення відповідності поля бази даних стовпцю і задає його властивості, такі як довжина, унікальність і назва. Також використовується анотація @Enumerated, щоб мапувати перерахування Java на відповідний тип стовпця бази даних ENUM. У розділі 2.3 кваліфікаційної роботи, на етапі проєктування БД, було прийнято рішення відмовитись від таблиць «user_role» та «user_status». Замість цього, поля "role" і "user_status" було включено безпосередньо до сутності «User» як ENUM типи, що відображають статус користувача та його роль. Це підходить для спрощення структури даних та зменшення кількості таблиць у базі даних, зберігаючи при цьому необхідність унікального ідентифікатора для кожного користувача і основних параметрів, таких як ім'я користувача, пароль і електронна пошта. У лістингу 3.9 наведено клас «UserStatus», який заміняє таблицю «user_status».

Лістингу 3.9 – Код класу «UserStatus»

```
public enum UserStatus {
    ACTIVE,
    INACTIVE,
    BLOCKED
}
```

Після представлення таблиці у вигляді Java-класу за допомогою Hibernate, наступним кроком є створення цієї таблиці у базі даних. Для цього й використовується LiquiBase. Спершу, необхідно створити файл у форматі XML, відповідно до документації [34]. Його вміст наведено у лістингу 3.10.

Лістинг 3.10 – Опис змін до бази даних за допомогою LiquiBase

```
<?xml version="1.0" encoding="UTF-8"?>
<databaseChangeLog
        xmlns="http://www.liquibase.org/xml/ns/dbchangelog"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://www.liquibase.org/xml/ns/dbchangelog
        http://www.liquibase.org/xml/ns/dbchangelog/dbchangelog-
4.5.xsd">

    <changeSet id="3" author="lukovskj">
        <createTable tableName="sih_user">
            <column name="id" type="BIGINT" autoIncrement="true">
                <constraints primaryKey="true"/>
            </column>
            <column name="username" type="VARCHAR(20)">
                <constraints unique="true" nullable="false" />
            </column>
            <column name="password" type="VARCHAR">
                <constraints nullable="false" />
            </column>
            <column name="user_status" type="VARCHAR">
                <constraints nullable="false" />
            </column>
               …
        </createTable>
    </changeSet>
</databaseChangeLog>
```

У лістингу 3.10 описано таку ж саму таблицю користувачів «user», яка була визначена у розділі 2.3 даної кваліфікаційної роботи, а також її структура ідентична полям класу «User.java», що наведено у лістингу 3.8. Таким чином,

при запуску серверу, LiquiBase створить таблицю, описану схемою з лістингу 3.10, якщо вона ще не існує, і прописана у відповідному файлі конфігурації.

Раніше було налаштовано «application.properties» для LiquiBase. Параметр «spring.liquibase.change-log=classpath:db/db.changelog-master.xml», що наведений у лістингу 3.7 вказує на розташування файлу конфігурації бази даних. Тому, для того, щоб створити таблицю «User», було додано посилання на відповідний файл зі змінами, що був описаний раніше. У лістингу 3.11 наведено файл конфігурації LiquiBase.

Лістинг 3.11 – Конфігураційний файл LiquiBase

```
<?xml version="1.0" encoding="UTF-8"?>
<databaseChangeLog …>
    <include            file="db/changelog/001-create-achievements-
table.xml"/>
    <include file="db/changelog/009-create-user-table.xml"/>
    <include file="db/changelog/006-create-tag-table.xml"/>
    <include file="db/changelog/007-create-test-table.xml"/>
    <include file="db/changelog/004-create-question-table.xml"/>
    <include file="db/changelog/002-create-article-table.xml"/>
    <include             file="db/changelog/003-create-article-tag-
table.xml"/>
    <include file="db/changelog/005-create-role-table.xml"/>
    <include file="db/changelog/008-create-test_tag-table.xml"/>
</databaseChangeLog>
```

У лістингу 3.11 наведено розроблений конфігураційний файл LiquiBase. Загалом, у даних XML містяться усі зміни до бази даних, які були здійснені під час розробки. Насамперед, це створення таблиць, їх заповнення тестовими даними.

Загалом, на даному етапі розробки було реалізовано базу даних та всі таблиці, що було визначенні у розділі 2.3 даної кваліфікаційної роботи, інструментами LiquiBase, а також їх ORM представлення за допомогою Hibernate. Проте опрацювання даних сервером поки не можливе. Для створення SQL-запитів до бази даних було розроблено спеціальні інтерфейси. Код одного із них наведено у лістингу 3.12.

Лістинг 3.12 – Інтерфейс-репозиторій для взаємодії із таблицею «question»

```
public interface QuestionRepository extends JpaRepository<Question, Long> {
    List<Question> findByTest(Test test);

    @Transactional
    @Modifying
    @Query("update Question t set t.correct_answer = :correct_answer, t.question_text = :question_text where t.id = :question_id")
    void updateQuestionData(@Param("question_id") Long question_id, @Param("correct_answer") String correct_answer,
                            @Param("question_text") String question_text);

    @Transactional
    @Modifying
    @Query("update Question t set t.question_text = 'test', t.correct_answer='test' where t.id = :question_id")
    void Update(@Param("question_id") Long question_id);

    @Transactional
    void deleteQuestionById(Long id);
}
```

У лістингу 3.12 наведено інтерфейс-репозиторій, який використовується для взаємодії з таблицею «question» у базі даних. Цей інтерфейс використовує Spring Data JPA для забезпечення доступу до даних через методи репозиторію [35].

«QuestionRepository» наслідує інтерфейс JpaRepository, який надає готові методи для роботи з базовими операціями CRUD (створення, читання, оновлення, видалення). Також було реалізовано додаткові методи:

– findByTest(Test test): Повертає список питань, що належать до вказаного тесту.

– updateQuestionData(Long question_id, String correct_answer, String question_text): Виконує оновлення даних питання, вказаного за ідентифікатором, з новими значеннями correct_answer та question_text.

– Update(Long question_id): Просте оновлення тексту питання і правильної відповіді на тестові значення для вказаного ідентифікатора питання.

– deleteQuestionById(Long id): Видаляє питання за вказаним ідентифікатором.

Ці методи використовують аннотації @Query, які дозволяють вказати власні SQL-запити для виконання спеціалізованих операцій з базою даних, що не включені в стандартні методи Spring Data JPA.

Усі Java-класи було розміщено у відповідні пакети «entity», у якому містяться представлення таблиць, та «repos», у якому розміщені репозиторії інтерфейси для виконання CRUD-операцій. Ця файлова структура зображена на рисунку 3.12.

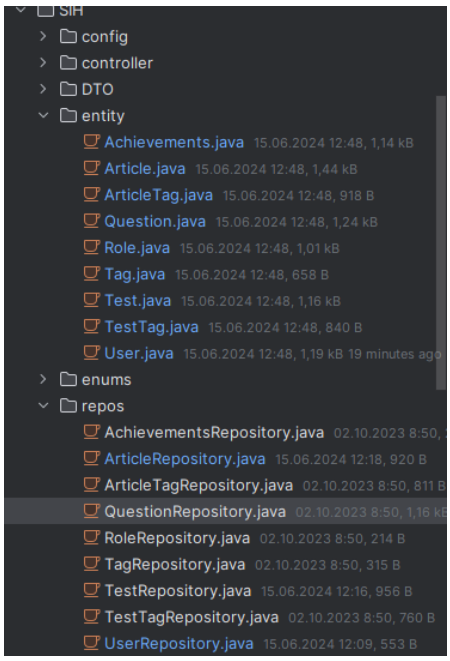

Рисунок 3.12 – Файлова структура Java-класів, що працюють із базою даних

Для налаштувань безпеки, в тому числі авторизації, автентифікації та розмежування доступу, використовується Spring Security [36]. Це один із компонентів Spring Framework, який забезпечує надійний інфраструктурний шар для захисту додатків. Spring Security дозволяє ефективно впроваджувати заходи безпеки без низькорівневої реалізації [37].

Налаштування відбувається за допомогою спеціального файлу «SecurityConfiguration.java». Загалом, файл досить великий, із багатьма

методами. Тому розглянемо основний функціонал – налаштування фільтрів. За це відповідає метод «securityFilterChain», код якого наведено у лістингу 3.13

Лістинг 3.13 – Код методу «securityFilterChain»

```
@Bean
public     SecurityFilterChain     securityFilterChain(HttpSecurity
httpSecurity) throws Exception{
        httpSecurity.cors(corsCustomizer                          ->
corsCustomizer.configurationSource(request -> {
      CorsConfiguration config = new CorsConfiguration();
                    config.addAllowedOrigin("*");
                    config.setAllowedMethods(
                            Arrays.asList("GET",           "POST",
"OPTIONS", "DELETE", "PUT", "PATCH"));
                   …
                .addFilterBefore(
                        new
AccessTokenAuthenticationFilter(jwtTool,   authenticationManager(),
userService),UsernamePasswordAuthenticationFilter.class)
                .exceptionHandling(exception -> exception
                        .authenticationEntryPoint((req, resp, exc)
-> resp.sendError(SC_UNAUTHORIZED, "Authorize first."))
                        .accessDeniedHandler((req,  resp,  exc)  ->
resp.sendError(SC_FORBIDDEN, "You don't have authorities.")))
                .authorizeHttpRequests(req -> req
                        .requestMatchers("/static/css/**",
       …
));
        return httpSecurity.build();
}
```

Метод «securityFilterChain», із лістингу 2.13, одночасно виконує досить багато функцій. Тому Розглянемо кожен аспект окремо:

1. CORS налаштування: Використовується для налаштування Cross-Origin Resource Sharing (CORS), що дозволяє браузерам зробити запити на різні домени/порти незважаючи на політику схоронності Same-Origin. У методі встановлюється дозвіл на запити з будь-якого джерела (*) і підтримуються методи GET, POST, OPTIONS, DELETE, PUT, PATCH.

2. Використання фільтрації: Додається фільтр «AccessTokenAuthenticationFilter» перед стандартним фільтром «UsernamePasswordAuthenticationFilter». Цей кастомний фільтр

використовується для перевірки і обробки токенів доступу (Access Token), що надає автентифікацію користувачів за допомогою JWT (JSON Web Token), інтегруючи механізми автентифікації з сервісами користувачів через «userService» та «authenticationManager».

3. Обробка винятків: Налаштовується обробка винятків для неавторизованих (SC_UNAUTHORIZED) і недостатньо авторизованих (SC_FORBIDDEN) запитів. Це дозволяє відправляти коректні HTTP відповіді з відповідними статус-кодами і повідомленнями у випадку відмови у доступі.

4. Авторизація запитів: Використовується метод «authorizeHttpRequests», що дозволяє визначити, які HTTP запити потребують певних прав доступу. У цьому прикладі показано декілька правил авторизації, де деякі шляхи запитів (наприклад, /static/css/**) доступні всім користувачам, тоді як інші мають обмеження щодо ролей.

Загалом, клас «SecurityConfiguration.java» є ключовим у налаштуванні безпеки серверу. Проте він використовує кілька допоміжних ресурсів для авторизації та фільтрування прав доступу. До прикладу, серед них клас «User.java» для представлення даних користувачів та відповідний репозиторій. Для шифрування даних, створення токенів доступу використовується «JwtTool.java», у якому реалізовано відповідні методи.

Розроблений функціонал дозволяє ефективно обробляти усі запити користувачів і залишати дані конфіденційними. Пароль користувача у базі даних зберігається лише у захешованому вигляді. Тобто, до прикладу під час реєстрації на вебсайті користувач вводить пароль «Qwerty12345». Проте у базі даних буде збережено лише послідовність символів, яка буде результатом роботи криптографічного алгоритму HS256. Приклад його роботи продемонстровано на рисунку 3.12 за допомогою вебсервісу 100L5.

Автентифікація користувача відбувається за допомогою токену доступу JWT. Він генерується, коли клієнт передає свій логін на пароль. Токен генерується під час аутентифікації, коли клієнт передає свій логін та пароль [38]. У токен записуються дані користувача, такі як його роль, час дії токена

тощо. При кожному подальшому запиті користувач повинен передавати цей токен у заголовку запиту.

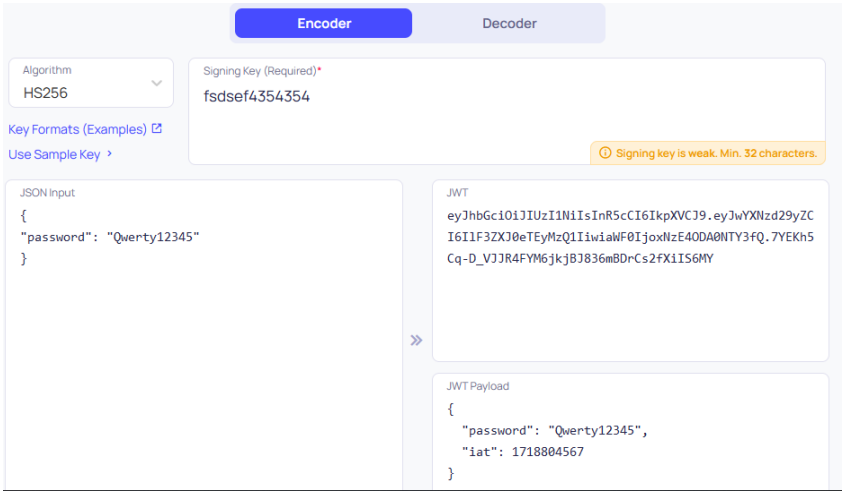

Рисунок 3.12 – Шифрування паролю користувача

Spring Security забезпечує перевірку токена. Після отримання запиту з токеном, фреймворк використовує фільтр, що представлений Java-класом «AccessTokenAuthenticationFilter», для вилучення і перевірки токена. Якщо він валідний і не протермінований, з нього витягуються дані користувача, і доступ до ресурсів надається відповідно до ролі користувача та правил доступу, налаштованих у файлі конфігурації, частина якого наведена у лістингу 3.13. Якщо токен недійсний або відсутній, користувач отримує відмову в доступі.

Взаємодія серверу з клієнтом відбувається за допомогою контролерів. У них визначаються точки доступу, тобто URL-шляхи, із відповідним HTTP методом, вони називаються – ендпоінт. До прикладу розглянемо процес реєстрації користувача. Клас «AuthorizationController.java» – це контролер, у якому зібрані усі ендпоінти, пов’язані із авторизацією користувачів. У лістингу 3.13 представлено ендпоінт для реєстрації користувача.

Лістинг 3.13 – Частина коду із класу «AuthorizationController.java»

```
@RestController
@RequestMapping("/auth")
@AllArgsConstructor
```

```
@Slf4j
@Validated
public class AuthorizationController {
    private final AuthorizationService authorizationService;

    @PostMapping("/signup")
    public ResponseEntity<UserAuthorizationDTO> signUp(
            @Valid @RequestBody UserSignUpDTO userRegistrationDTO)
throws BadRequestException {
        return
ResponseEntity.status(HttpStatus.OK).body(authorizationService.sig
nUp(userRegistrationDTO));
    }
```

Отже, для реєстрації користувачу потрібно надіслати POST запит за URL-посилання «DOMAIN_NAME/auth/ signup», із прикріпленим JSON файлом, у якому містяться пароль, логін та електронна пошта користувача. За допомогою сервісного шару серверу, який відповідає за бізнес-обчислення, метод «signUp» перевіряє валідність переданих даних. Якщо жодних проблем під час перевірки не виявлено, дані користувача записуються до бази даних.

Для візуалізації роботи серверу було використано ілюстрацію із вебресурсу Medium, що зображено на рисунку 3.13. У ньому показано як різні компоненти системи взаємодіють між собою.

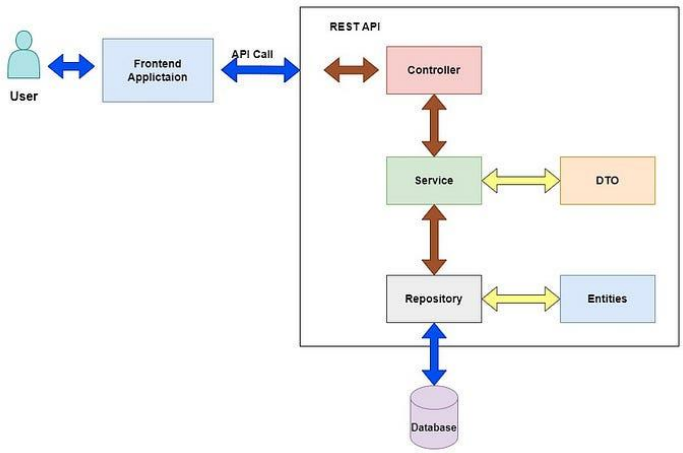

Рисунок 3.13 – Візуалізації архітектури серверу та взаємодії між його компонентами [39]

У підсумку, розроблений на сервері функціонал забезпечує виконання усіх вимог, зазначених у технічному завдані.

3.3 Розробка клієнтської частини

Зверстані вебсторінки, що описані у пункті 3.1 кваліфікаційної роботи, є статичними. Тобто вони не можуть оновлюватись та надавати користувачам актуальної інформації.

Для вирішення цієї проблеми було використано JavaScript, Vue.js та Thymeleaf. За допомогою першого було реалізовано асихронні запити до серверу, анімації, різноманітні обчислення. До прикладу, на головній сторінці використовується слайдер для демонстрації музикальних альбомів. Його зображено на рисунку 2.13. Стандартними засобами HTML, CSS реалізувати таке завдання технічно неможливо. Тому для цього було розроблено спеціальні JavaScript функції. Відповідний код наведено у лістингу 3.14.

Лістинг 3.14 – Функція, яка анімує альбоми на головній сторінці.

```
getAnimation(id) {
      const clickedImg = document.getElementById(id);
      if (clickedImg.dataset.imgOrder !== "3") {
        const moveBackImg = document.querySelector('[data-img-
order="3"]');
        clickedImg.classList.add("img-swap-up");
        const tempTernar = `img-swap-back-${300 - 100 *
clickedImg.dataset.imgOrder}`;
        moveBackImg.classList.add(tempTernar);
        this.setImagesZIndex(clickedImg, moveBackImg)
        moveBackImg.dataset.imgOrder =
clickedImg.dataset.imgOrder;
        clickedImg.dataset.imgOrder = 3;
        return Promise.all([
          new Promise((resolve) => {
            clickedImg.addEventListener('transitionend', (event)
=> {
              if (event.propertyName === 'transform') {
                clickedImg.classList.remove("img-swap-up");
                moveBackImg.classList.remove(tempTernar);
                resolve();
              }
            });
```

```
    }),
    new Promise((resolve) => {
      moveBackImg.style.left = clickedImg.style.left;
      clickedImg.style.left = '200px';
      resolve();
    }
…
```

Загалом, функція, код якої наведено у лістингу 3.14 відстежує кліки користувача. А далі переміщує компоненти HTML, змінюючи їхню стилізацію. Тестовий приклад її використання наведено на рисунку 3.14. Слайдер ще не був вмонтований у головну сторінку, оскільки його розробка велась окремо і вимагала додаткових змін у вже зверстані макети вебсторінок.

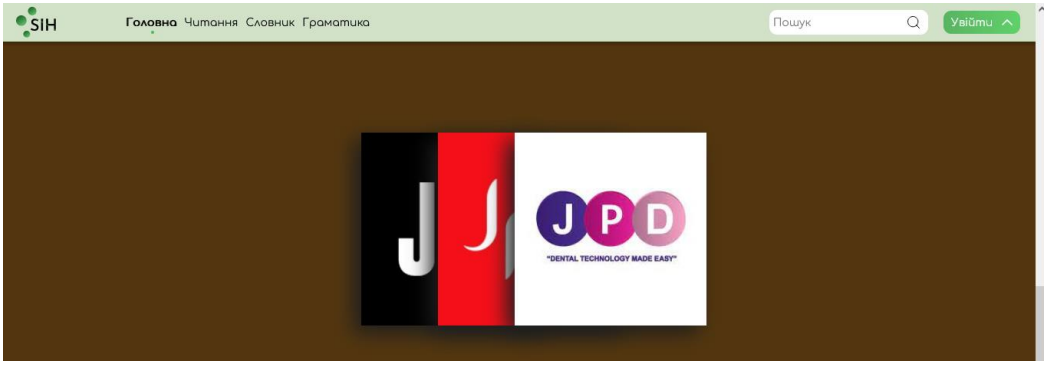

Рисунок 3.14 – Розробелний слайдер

Для реалізації динамічних вебсторінок було застосовано два підходи: генерування вмісту за допомогою Thymeleaf та Vue.js. Перший інструмент використовується для серверної генерації HTML-сторінок. Це дозволяє інтегрувати Java-код із шаблонами HTML, що спрощує створення складних динамічних сторінок. Thymeleaf особливо корисний для рендерингу сторінок, які вимагають попередньої обробки даних на сервері. Наприклад, на сторінці профілю користувача, де необхідно показати персональну інформацію та історію активності, Thymeleaf генерує HTML на основі даних із бази даних, отриманих за допомогою Spring Data JPA. У лістингу 3.15 наведено приклад використання шаблонізатора.

Лістинг 3.15 – Приклад використання Thymeleaf

```
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <th:block            th:replace="fragments/header            ::
header_links"></th:block>
    <title>Past Simple | SIH</title>
    <!--sec:authorize="hasAuthority('ROLE_ADMIN')"-->
</head>
```

Кожна вебсторінка містить одинакові посилання на підключення CSS файлів, що відповідають за стилізацію вебсторінок. Для того, щоб уникнути дублювання використовуються засоби Thymeleaf. У лістингу 3.15 це продемонстровано рядком «<th:block th:replace="».

Іншим прикладом використання є генерація даних за допомогою Thymeleaf. У звязці із Spring Framework, можна передавати параметри, на основі яких будуть генеруватись дані на HTML сторінці. Це продемонстровано у лістингу 3.16.

Лістинг 3.16 – Динамічна вебсторінка, створена за допомогою Thymeleaf

```
<th:block if="${article_list}">…
    <th:block th:each="article_list, state : ${article_list}">…
                                <p    th:text="'Стаття    №'    +
(${state.index}+1) + ' : ' + ${article_list.getArticleName}"></p>
                                <p   th:text="'Опис   тесту:   '   +
${article_list.getArticleDesc}"></p>
    </div>
</th:block>
```

Для кращої візуалізації у лістингу 3.16 було наведено лише ті фрагменти HTML, у формуванні яких Thymeleaf приймає безпосередню участь. За допомогою «${article_list.getArticleName}» можна парсити дані із серверу. До прикладу, користувач звертається за шляхом «/user/profile». Spring Security автентифікує користувача і далі шукає доступні для нього статті («article_list» – це список доступних статей).

Аналогічна робота була проведена й з іншими вебсторінками. Проте у них для їх реалізація була значно простішою та частково описана у розділі про

розробку дизайну на верстки вебсторінок. У більшості випадків було додано невеликі функції, які б дозволяли робити асинхронні запити до серверу, або функції анімації, як із слайдером, що наведений у лістингу 3.14.

3.4 Висновок до третього розділу

Підсумовуючи виконану роботу, можна сказати, що в процесі розробки клієнтської частини було зверстано статичні вебсторінки, що за допомогою JavaScript, Vue.js та Thymeleaf було перетворено на динамічні. Впровадження асинхронних запитів, анімацій та обчислень дозволило забезпечити користувачам актуальну інформацію, зокрема реалізацію слайдеру для демонстрації музичних альбомів. Для генерації вмісту використано Thymeleaf та Vue.js, де Thymeleaf застосовувався для серверної генерації HTML-сторінок на основі даних з бази даних, а Vue.js для клієнтської частини.

Розробка серверу включала використання Spring Initializr для створення початкової конфігурації проєкту на основі Spring Framework, зокрема для керування залежностями, автоматизації процесів збирання, тестування та розгортання програмного забезпечення. Maven забезпечував структуру проєкту, а IntelliJ IDEA Ultimate Edition використовувалася як основне середовище розробки. Також було застосовано бібліотеки Lombok та LiquiBase для автоматизації керування версіями бази даних та генерації SQL-запитів для взаємодії з нею.

Кожна таблиця бази даних мала відповідне представлення у вигляді Java-класу, а Hibernate, як ORM-інструмент, забезпечував мапування об'єктів Java на таблиці бази даних. Для кожної сутності були створені відповідні класи, що описували структуру таблиць, зокрема, для таблиці «User» визначено поля та анотації для коректного мапування.

РОЗДІЛ 4. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Психологічні чинники небезпеки

Психологічні чинники небезпеки є важливим аспектом у питаннях безпеки життєдіяльності, особливо у сфері науки та освіти, де психоемоційний стан людини відіграє ключову роль. У контексті розробки вебсайту школи іноземних мов «Study isn't hard», важливо враховувати ці чинники, щоб створити безпечне і сприятливе середовище як для учнів, так і для працівників [40].

У процесі своєї діяльності людина використовує не тільки свої фізичні можливості, а й витрачає значні психологічні зусилля, такі як особливості характеру, волю, розумові здібності тощо. Психофізіологічними називаються небезпечні фактори, зумовлені особливостями фізіології та психології людини. Вони можуть бути як постійними, так і тимчасовими.

До постійних психофізіологічних небезпек належать недоліки органів відчуття (дефекти зору, слуху тощо), порушення зв’язків між сенсорними та моторними центрами, дефекти координації рухів, підвищена емоційність, та відсутність мотивації до трудової діяльності. Такі чинники можуть призводити до низької продуктивності та підвищеної втоми, що негативно впливає на загальну безпеку життєдіяльності.

Тимчасові психофізіологічні небезпеки включають недостатність досвіду, необережність, втому, та емоційні явища (особливо конфліктні ситуації, душевні стреси, пов'язані з побутом, сім'єю, друзями, керівництвом). Втома після важкої, але потрібної праці, пов'язана з позитивним емоційним станом, може бути зменшена через активний відпочинок та зміну виду діяльності.

На успіх діяльності особливо впливає стан людини. Будь-який вид діяльності викликає втому, що призводить до зниження продуктивності через витрату енергетичних ресурсів організму людини. Цей стан виникає через певне ставлення людини до праці та звички до фізичного і розумового

напруження. Монотонна, нецікава робота призводить до швидшої втоми, ніж цікава та захоплююча діяльність.

При перевтомі період оптимальної працездатності скорочується, а період нестійкої компенсації збільшується. Це призводить до порушення відновних процесів в організмі, збільшення кількості помилок та браку у роботі, а також погіршення загального стану здоров'я, що може впливати на безпеку життєдіяльность.

Для забезпечення психологічної безпеки на робочих місцях необхідно проводити регулярні тренінги та заходи з покращення психоемоційного стану працівників. Це включає організацію робочого часу з урахуванням перерв для відпочинку та створення умов для емоційної підтримки з боку керівництва та колег [41].

Ефективне управління психологічними чинниками небезпеки вимагає комплексного підходу, що включає як організаційні заходи, так і індивідуальну роботу з працівниками. Дотримання стандартів ДСТУ та рекомендацій міжнародних організацій допоможе створити безпечне та здорове робоче середовище, сприятиме підвищенню ефективності та зниженню ризиків.

Вплив психологічних факторів на безпеку праці та життєдіяльність охоплює широкий спектр, включаючи емоційний стан, мотивацію, увагу та індивідуальні психофізіологічні особливості. Психологічна напруга, втома та стрес можуть значно знизити працездатність і підвищити ризик аварій та травм.

Серед основних психологічних чинників, що можуть впливати на безпеку життєдіяльності, виділяють наступні:

– стресові ситуації: робота в умовах високої відповідальності або підвищеного навантаження може викликати стрес, що впливає на здатність приймати правильні рішення;

– мотиваційні чинники: відсутність мотивації або демотивація може знижувати якість виконання завдань та підвищувати ймовірність помилок;

– психоемоційна стійкість: важливо враховувати рівень психоемоційної стійкості працівників, оскільки нестабільність може призводити до непередбачуваних реакцій та поведінки в стресових ситуаціях;

– соціально-психологічний клімат у колективі: доброзичлива атмосфера в колективі сприяє зниженню рівня стресу та підвищенню загального рівня безпеки праці.

Врахування психологічних чинників небезпеки є критичним аспектом у забезпеченні безпеки життєдіяльності та ефективності праці. У контексті розробки вебсайту школи іноземних мов "Study isn't hard", необхідно створити сприятливі умови для навчання та роботи, що враховують можливі психофізіологічні небезпеки. Дотримання рекомендацій міжнародних організацій та стандартів ДСТУ допоможе забезпечити здорове робоче середовище, підвищити мотивацію і продуктивність працівників, а також знизити ризики аварій та травм. Таким чином, комплексний підхід до управління психологічними чинниками є запорукою успішної та безпечної діяльності в освітній сфері [42].

4.2 Правила техніки безпеки при експлуатації обладнання

Для забезпечення ефективної і безпечної роботи важливо дотримуватися правил техніки безпеки при експлуатації комп'ютерного обладнання та іншої техніки, яка використовується в процесі розробки.

Правильне розташування робочого місця. Для забезпечення безпеки та комфорту працівників необхідно правильно організувати робоче місце:

– Відстань до монітора: монітор повинен розташовуватися на відстані 50-70 см від очей користувача.

– Висота столу та стільця: висота столу повинна бути на рівні ліктів, коли руки розташовані горизонтально, а стілець повинен бути регульованим по висоті.

– Освітлення: робоче місце має бути добре освітленим, щоб уникнути напруги очей. Варто уникати прямого світла на монітор.

Для зниження ризику виникнення професійних захворювань важливо використовувати ергономічне обладнання:

– Клавіатура і мишка: повинні бути ергономічними, щоб зменшити навантаження на зап'ястя.

– Крісло: повинно бути з підтримкою попереку та регульованою спинкою, щоб підтримувати правильну поставу.

– Монітор: повинен мати регульовану висоту та нахил, щоб зменшити навантаження на шию.

Комп'ютерне обладнання та периферія потребують належного електробезпеки для запобігання аваріям та пошкодженням:

Заземлення: всі електричні пристрої повинні бути належним чином заземлені [43]:

– Кабелі: не повинні бути пошкодженими або перевантаженими. Вони повинні бути належним чином закріплені, щоб уникнути падіння або заплутування.

– Розетки: не слід використовувати занадто багато пристроїв в одній розетці. Розетки повинні бути розташовані в зручних місцях і бути легкодоступними.

Для запобігання професійним захворюванням і збереження здоров'я працівників необхідно дотримуватися таких заходів:

– Перерви: працівники повинні робити короткі перерви кожні 1-2 години для розминки та відпочинку очей.

– Вправи: рекомендується виконувати прості вправи для зняття напруги з очей, рук та спини.

– Медичні огляди: регулярні медичні огляди допоможуть вчасно виявити та запобігти професійним захворюванням.

Для забезпечення безпеки даних та уникнення кіберзагроз необхідно дотримуватися правил безпеки при роботі з програмним забезпеченням:

– Паролі: використовувати складні паролі та регулярно їх змінювати.

– Антивірусне програмне забезпечення: встановити та регулярно оновлювати антивірусні програми.

– Оновлення: своєчасно оновлювати операційну систему та програмне забезпечення, щоб уникнути вразливостей.

– Працівники повинні бути належним чином проінструктовані щодо правил техніки безпеки та регулярного навчання:

– Інструктаж: нові співробітники повинні проходити інструктаж з техніки безпеки.

4.3 Висновок до четвертого розділу

У розділі було досліджено психологічні чинники небезпеки, які мають важливе значення для безпеки життєдіяльності в контексті роботи над створенням вебсайту школи іноземних мов "Study isn't hard". Було підкреслено важливість врахування психологічного клімату у колективах та індивідуальних особливостей працівників і учнів, щоб запобігти стресовим ситуаціям і забезпечити ефективну діяльність. Детально розглянуто вплив постійних і тимчасових психофізіологічних небезпек на продуктивність праці та загальний стан здоров'я працівників. Також досліджено правила техніки безпеки при експлуатації комп'ютерного обладнання, підкреслюючи важливість правильного розташування робочого місця, використання ергономічного обладнання та дотримання електробезпеки [44]. Зроблено акцент на заходах для зниження ризику професійних захворювань, таких як регулярні перерви, вправи та медичні огляди. Висновки підкреслюють важливість комплексного підходу до управління безпекою праці, що включає як організаційні, так і індивідуальні заходи, що сприяють створенню безпечного та здорового робочого середовища.

ВИСНОВКИ

Під час написання цієї кваліфікаційної роботи були досліджені різноманітні аспекти розробки вебсайту школи іноземних мов "Study isn't hard". Робота складається з чотирьох розділів, кожен з яких містить важливі етапи аналізу, проектування, реалізації та оцінки безпеки життєдіяльності та охорони праці.

У першому розділі було здійснено ґрунтовний аналіз предметної області. Було розглянуто актуальні тенденції та виклики, з якими стикаються сучасні навчальні заклади у сфері дистанційного навчання. Аналіз існуючих рішень дозволив визначити сильні та слабкі сторони поточних платформ для навчання іноземних мов.

На основі цього аналізу було сформульовано технічне завдання, яке включає загальні характеристики, архітектуру розроблюваного програмного забезпечення, вимоги до серверу, збереження даних, дизайну, клієнтської частини та функціоналу вебсайту. Таким чином, у першому розділі були визначені ключові аспекти, необхідні для успішної реалізації проекту.

Другий розділ присвячений проектній частині роботи. На цьому етапі було розроблено структуру сайту та вебсторінок, що забезпечує логічну та зручну навігацію для користувачів. Окрему увагу приділено дизайну, що повинен бути сучасним, естетично привабливим та інтуїтивно зрозумілим.

Важливим етапом проектування стала розробка бази даних, яка забезпечує ефективне зберігання та обробку інформації. Всі ці заходи спрямовані на створення надійного та функціонального вебсайту, який відповідає сучасним вимогам користувачів та забезпечує високий рівень зручності і безпеки.

У третьому розділі була здійснена програмна реалізація проекту. Було виконано верстку вебсторінок з використанням сучасних технологій HTML, CSS та JavaScript, що забезпечило кросбраузерну сумісність та адаптивність дизайну.

Розробка серверної частини включала налаштування серверного середовища та реалізацію бекенд-логіки на мові програмування Java, що забезпечило стабільність та високу продуктивність роботи вебсайту. Клієнтська частина була реалізована з використанням JavaScript, що дозволило створити інтерактивний і динамічний користувацький інтерфейс. Під час реалізації були протестовані всі компоненти системи, що забезпечило їх безперебійну роботу та відповідність вимогам технічного завдання.

Четвертий розділ роботи присвячений питанням безпеки життєдіяльності та основам охорони праці. Враховуючи психологічні чинники небезпеки, було розроблено рекомендації щодо організації безпечних умов праці для працівників та учнів. Описано правила техніки безпеки при експлуатації обладнання, включаючи належне розташування робочих місць, використання ергономічного обладнання та дотримання електробезпеки.

Важливим аспектом стало дослідження заходів для зниження ризику професійних захворювань, включаючи регулярні перерви, вправи та медичні огляди. Таким чином, були розроблені комплексні заходи, що сприяють створенню безпечного та здорового робочого середовища.

Підбиваючи підсумки, можна стверджувати, що в ході виконання кваліфікаційної роботи було досягнуто поставленої мети – створення функціонального, зручного та безпечного вебсайту для школи іноземних мов «Study isn't hard». Виконані завдання дозволили реалізувати всі заплановані етапи проекту, забезпечивши високу якість кінцевого продукту. Результати роботи можуть бути корисними для подальших досліджень та вдосконалення освітніх платформ в умовах сучасного інформаційного суспільства.

.

ПЕРЕЛІК ДЖЕРЕЛ

1. W3Techs. 'Usage statistics of content languages for websites'. Available at: https://w3techs.com/technologies/overview/content_language (Accessed: 10 June 2024).

2. Companies Market Cap. 'Largest tech companies by market cap'. Available at: https://companiesmarketcap.com/tech/largest-tech-companies-by-market-cap/ (Accessed: 10 June 2024).

3. Суспільне. 'Володіння англійською в Україні: що показало опитування і як правильно вчити іноземну мову'. Available at: https://suspilne.media/rivne/580749-volodinna-anglijskou-v-ukraini-so-pokazalo-opituvanna-i-ak-pravilno-vciti-inozemnu-movu/ (Accessed: 10 June 2024).

4. EF. 'The world’s largest ranking of countries and regions by English skills'. Available at: https://www.ef.co.uk/epi/ (Accessed: 10 June 2024).

5. Strutynska, I., Kozbur, H., Dmytrotsa, L., Hlado, O., Kozbur, I., Gashchyn, N. (2023) 'Analysis of the SMEs’ Digitalization State Using HIT Index and Machine Learning Technique', 13th International Conference on Advanced Computer Information Technologies (ACIT). IEEE. Wroclaw, Poland, pp. 332-337.

6. PLOS One. 'Disadvantages in preparing and publishing scientific papers caused by the dominance of the English language in science: The case of Colombian researchers in biological sciences'. Available at: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0238372 (Accessed: 10 June 2024).

7. Nielsen. 'Navigate today’s media with the latest audience data'. Available at: https://www.nielsen.com/data-center/ (Accessed: 10 June 2024).

8. Work.ua. 'Роботодавці очікують знання англійської та розміщують вакансії для ветеранів: як змінювався ринок праці у лютому'. Available at: https://www.work.ua/articles/analytics/3252/ (Accessed: 10 June 2024).

9. Teacher Plus. 'Individual vs collective learning'. Available at: https://www.teacherplus.org/individual-vs-collective-learning/ (Accessed: 10 June 2024).

10. Harvard. 'Dopamine, Smartphones & You: A battle for your time'. Available at: https://sitn.hms.harvard.edu/flash/2018/dopamine-smartphones-battle-time/ (Accessed: 10 June 2024).

11. Duolingo. 'Duolingo працює'. Available at: https://uk.duolingo.com/efficacy (Accessed: 10 June 2024).

12. Duolingo. 'Finishing half of B1 on Duolingo comparable to five university semesters in reading and listening'. Available at: https://duolingo-papers.s3.amazonaws.com/reports/Duolingo_whitepaper_language_read_listen_2021.pdf (Accessed: 10 June 2024).

13. BMC Medical Education. 'A story half told: a qualitative study of medical students' self-directed learning in the clinical setting'. Available at: https://bmcmededuc.biomedcentral.com/articles/10.1186/s12909-021-02913-3 (Accessed: 11 June 2024).

14. GeeksforGeeks. 'Client-Server Model'. Available at: https://www.geeksforgeeks.org/client-server-model/ (Accessed: 11 June 2024).

15. Inkbot Design. 'How to Choose the Best Fonts for a Website'. Available at: https://inkbotdesign.com/best-fonts-for-a-website/ (Accessed: 11 June 2024).

16. Mental Health America. 'How do colors in my home change my mood? Color psychology explained'. Available at: https://mhanational.org/surroundings/color-psychology-explained (Accessed: 11 June 2024).

17. Toptal. 'Designing for Readability: A Guide to Web Typography (With Infographic)'. Available at: https://www.toptal.com/designers/typography/web-typography-infographic (Accessed: 12 June 2024).

18. AltexSoft. 'Comparing Database Management Systems: MySQL, PostgreSQL, MSSQL Server, MongoDB, Elasticsearch, and others'. Available at: https://www.altexsoft.com/blog/comparing-database-management-systems-mysql-

postgresql-mssql-server-mongodb-elasticsearch-and-others/ (Accessed: 12 June 2024).

19. PostgreSQL. 'PostgreSQL Official documentation'. Available at: https://www.postgresql.org/docs/current/datatype-datetime.html (Accessed: 12 June 2024).

20. Imperva. 'SQL (Structured query language) Injection'. Available at: https://www.imperva.com/learn/application-security/sql-injection-sqli/ (Accessed: 12 June 2024).

21. Wix. 'What makes a good website structure? Everything you need to know'. Available at: https://www.wix.com/blog/website-structures (Accessed: 13 June 2024).

22. Google Fonts. 'Comfortaa'. Available at: https://fonts.google.com/specimen/Comfortaa (Accessed: 13 June 2024).

23. Mozilla Developer Network. 'What do common web layouts contain?'. Available at: https://developer.mozilla.org/en-US/docs/Learn/Common_questions/Design_and_accessibility/Common_web_layouts (Accessed: 13 June 2024).

24. BrowserStack. 'Ideal screen sizes for responsive design'. Available at: https://www.browserstack.com/guide/ideal-screen-sizes-for-responsive-design (Accessed: 13 June 2024).

25. WeAreDevelopers. '25 Essential Tools for Front End Developers'. Available at: https://www.wearedevelopers.com/magazine/best-tools-for-front-end-development (Accessed: 13 June 2024).

26. Jetpack. 'What is a Footer on a Website? What Should It Contain?'. Available at: https://jetpack.com/blog/website-footer-tips/ (Accessed: 13 June 2024).

27. Mozilla. 'Firefox DevTools User Docs'. Available at: https://firefox-source-docs.mozilla.org/devtools-user/ (Accessed: 14 June 2024).

28. KayVee Media. 'The 3 Components of A Website'. Available at: https://kayveemedia.com/blog/2020/02/03/the-3-components-of-a-website/ (Accessed: 14 June 2024).

29. Spring. 'Spring Initializr'. Available at: https://start.spring.io/ (Accessed: 14 June 2024).

30. Apache Maven. 'Maven documentation'. Available at: https://maven.apache.org/guides/ (Accessed: 14 June 2024).

31. Project Lombok. 'Project Lombok'. Available at: https://projectlombok.org/ (Accessed: 14 June 2024).

32. Hibernate. 'Hibernate ORM'. Available at: https://hibernate.org/orm/ (Accessed: 15 June 2024).

33. Jakarta EE. 'Annotation Type Id'. Available at: https://jakarta.ee/specifications/persistence/2.2/apidocs/javax/persistence/id (Accessed: 15 June 2024).

34. Liquibase. 'Liquibase Documentation'. Available at: https://docs.liquibase.com/home.html (Accessed: 15 June 2024).

35. Spring. 'Spring Data JPA'. Available at: https://spring.io/projects/spring-data-jpa (Accessed: 15 June 2024).

36. Marco Behler. 'Spring Security: Authentication and Authorization In-Depth'. Available at: https://www.marcobehler.com/guides/spring-security (Accessed: 15 June 2024).

37. Холод Д. М. Проблеми захисту комп’ютерних систем / Д. М. Холод, Г. В. Шимчук // Збірник тез доповідей VI Міжнародної науково-технічної конференції молодих учених та студентів „Актуальні задачі сучасних технологій“, 16-17 листопада 2017 року. — Т. : ТНТУ, 2017. — Том 2. — С. 179–180. — (Комп’ютерно-інформаційні технології та системи зв’язку).

38. Романець А. Проблеми аутентифікації акаунтів у соцмережах / А. Романець, Галина Володимирівна Козбур // Матеріали X науково-технічної конференції „Інформаційні моделі, системи та технології“, 7–8 грудня 2022 року. — Т. : ТНТУ, 2022. — С. 44. — (Інформаційні системи та технології, кібербезпека).

39. Sudusinghe, P. 'The architecture of the Spring Boot REST applications'. Available at: https://medium.com/@piyumisudusinghe/the-architecture-of-the-spring-boot-rest-applications-3643e905c5f8 (Accessed: 15 June 2024).

40. Надія Куравська Прояви захищеності як чинника психологічної безпеки студентів в умовах ВНЗ // 28 січня 2016 року. – Збірник наукових праць: психологія №21, 20 серпня 2021. – С. 79-86.

41. Цюман, Т., & Нагула , О. (2021). ПСИХОЛОГІЧНА ФОРМУЛА БЕЗПЕКИ ЯК КОНЦЕПТУАЛЬНА ОСНОВА ФОРМУВАННЯ НАВИЧОК БЕЗПЕЧНОЇ ПОВЕДІНКИ ОСОБИСТОСТІ. Педагогічна освіта: Теорія і практика. Психологія. Педагогіка., (35 (1), 94–100. https://doi.org/10.28925/2311-2409.2021.3513

42. О. Бондаренко, О. Євдін, К. Жебровська, М. Кучма, В. Скакун, А. Ющенко Джерела фізичного походження природних надзвичайних ситуацій ДСТУ 4934:2008 // 20 липня 2008 року. – ДЕРЖСПОЖИВСТАНДАРТ України. - С. 1-12.

43. В. Божко, А. Гінайло, О. Гончаров, М. Зеленкевич, В. Мозирський, В. Плакидюк, Д. Розинський Електробезпека в будівлях і спорудах. Вимоги До захисних заходів від ураження електричним струмом // ДСТУ Б В.2.5-82:2016 // 1 липня 2016. – С. 1-110. ДЕРЖСПОЖИВСТАНДАРТ України. -

44. В. Гажаман; В. Каньшин; М. Лисюк, канд. техн. наук; О. Михайленко Загальні вимоги та номенклатура видів захисту // ДСТУ 7237:2011 // 2 лютого 2011 року // ДЕРЖСПОЖИВСТАНДАРТ України.

ДОДАТКИ

Додаток А

Лістинг файлу «AccessTokenAuthenticationFilter.java»

```
package studyisnthard.SIH.security;

import io.jsonwebtoken.ExpiredJwtException;
import jakarta.servlet.FilterChain;
import jakarta.servlet.ServletException;
import jakarta.servlet.http.Cookie;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import lombok.AllArgsConstructor;
import lombok.extern.slf4j.Slf4j;
import
org.springframework.security.authentication.AuthenticationManager;
import
org.springframework.security.authentication.UsernamePasswordAuthen
ticationToken;
import org.springframework.security.core.Authentication;
import
org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.web.filter.OncePerRequestFilter;
import studyisnthard.SIH.DTO.UserDTO;
import studyisnthard.SIH.service_api.UserService;

import java.io.IOException;
import java.util.Arrays;
import java.util.Optional;

@Slf4j
@AllArgsConstructor
public      class      AccessTokenAuthenticationFilter      extends
OncePerRequestFilter {
    private final JwtTool jwt;
    private final AuthenticationManager authenticationManager;
    private final UserService userService;

    private String getTokenFromCookies(Cookie[] cookies) {
        return Arrays.stream(cookies)
                .filter(c -> c.getName().equals("accessToken"))
                .findFirst()
                .map(Cookie::getValue).orElse(null);
    }

    private String extractToken(HttpServletRequest request) {
        Cookie[] cookies = request.getCookies();
        String uri = request.getRequestURI();
        if (cookies != null && uri.startsWith("/management")) {
            return getTokenFromCookies(cookies);
        }
        return jwt.getTokenFromHttpServletRequest(request);
```

```
    }

    @Override
    protected void doFilterInternal(HttpServletRequest request,
HttpServletResponse response, FilterChain filterChain) throws
ServletException, IOException {
        String token = extractToken(request);

        if (token != null) {
            try {
                Authentication authentication =
authenticationManager
                        .authenticate(new
UsernamePasswordAuthenticationToken(token, null));
                Optional<UserDTO> user =
userService.getActiveUserByEmail((String)
authentication.getPrincipal());
                if (user.isPresent()) {
                    log.info("User successfully authenticate -
{}", authentication.getPrincipal());

SecurityContextHolder.getContext().setAuthentication(authenticatio
n);
                }
            } catch (ExpiredJwtException e) {
                log.info("Token has expired: " + token);
            } catch (Exception e) {
                log.info("Access denied with token: " +
e.getMessage());
            }
        }
        filterChain.doFilter(request, response);
    }
}
```

Додаток Б

Лістинг файлу «JwtAuthenticationProvider.java»

```
package studyisnthard.SIH.security;

import io.jsonwebtoken.Jwts;
import io.jsonwebtoken.UnsupportedJwtException;
import io.jsonwebtoken.security.Keys;
import
org.springframework.security.authentication.AuthenticationProvider
;
import
org.springframework.security.authentication.UsernamePasswordAuthen
ticationToken;
import org.springframework.security.core.Authentication;
import
org.springframework.security.core.authority.SimpleGrantedAuthority
;

import javax.crypto.SecretKey;
import java.util.List;
import java.util.stream.Collectors;

public class JwtAuthenticationProvider implements
AuthenticationProvider {
    private final JwtTool jwtTool;

    /**
     * Constructor.
     *
     * @param jwtTool {@link JwtTool}
     */
    public JwtAuthenticationProvider(JwtTool jwtTool) {
        this.jwtTool = jwtTool;
    }

    /**
     * Method that provide authentication.
     *
     *    @param    authentication    {@link    Authentication}    -
authentication that has jwt
     *                          access token.
     *   @return   {@link   Authentication}   if   user   successfully
authenticated.
     *  @throws  io.jsonwebtoken.ExpiredJwtException     -  if  the
token expired.
     *  @throws  UnsupportedJwtException                    if  the
argument does not
     *                                                 represent an
Claims JWS
```

```
     * @throws io.jsonwebtoken.MalformedJwtException if the string
is not a valid
     *                                                        JWS
     * @throws io.jsonwebtoken.SignatureException      if the JWS
signature validation
     *                                                        fails
     */
    @Override
    public        Authentication        authenticate(Authentication
authentication) {
        SecretKey                       key                      =
Keys.hmacShaKeyFor(jwtTool.getAccessTokenKey().getBytes());

        String email = Jwts.parser()
                .verifyWith(key).build()
                .parseSignedClaims(authentication.getName())
                .getPayload()
                .getSubject();
        String ROLE = "role";
        @SuppressWarnings({"unchecked, rawtype"})
        List<String> authorities = (List<String>) Jwts.parser()
                .verifyWith(key).build()
                .parseSignedClaims(authentication.getName())
                .getPayload()
                .get(ROLE);

        return new UsernamePasswordAuthenticationToken(
                email,
                "",

authorities.stream().map(SimpleGrantedAuthority::new).collect(Coll
ectors.toList()));
    }

    /**
     * {@inheritDoc}
     */
    @Override
    public boolean supports(Class<?> authentication) {
        return
authentication.equals(UsernamePasswordAuthenticationToken.class);
    }
}
```

Додаток В

Лістинг файлу «JwtTool.java»

```
package studyisnthard.SIH.security;

import com.fasterxml.jackson.core.JsonProcessingException;
import com.fasterxml.jackson.databind.JsonNode;
import com.fasterxml.jackson.databind.ObjectMapper;
import io.jsonwebtoken.ClaimsBuilder;
import io.jsonwebtoken.Jwts;
import io.jsonwebtoken.security.Keys;
import jakarta.servlet.http.HttpServletRequest;
import lombok.Getter;
import lombok.extern.slf4j.Slf4j;
import
org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Component;
import studyisnthard.SIH.entity.User;
import studyisnthard.SIH.enums.Role;

import javax.crypto.SecretKey;
import java.nio.charset.StandardCharsets;
import java.util.*;

@Component
@Slf4j
public class JwtTool {
    private static final String ROLE = "role";
    private final Integer accessTokenValidTimeINMinutes;
    private final Integer refreshTokenValidTimeInMinutes;
    @Getter
    private final String accessTokenKey;

    @Autowired
    public JwtTool(@Value("120") Integer
accessTokenValidTimeINMinutes,
                   @Value("600") Integer
refreshTokenValidTimeInMinutes,

@Value("Gsd293225395354dfkdfgdfglnsds32SD=23se") String
accessTokenKey) {
        this.accessTokenValidTimeINMinutes =
accessTokenValidTimeINMinutes;
        this.refreshTokenValidTimeInMinutes =
refreshTokenValidTimeInMinutes;
        this.accessTokenKey = accessTokenKey;
    }

    public String createAccessJWT(String subject, Role role)
{
```

```
            ClaimsBuilder claimsBuilder =
Jwts.claims().subject(subject);
            claimsBuilder.add(ROLE,
Collections.singleton(role.name()));
            Date now = new Date();
            Calendar calendar = Calendar.getInstance();
            calendar.setTime(now);
            calendar.add(Calendar.MINUTE,
accessTokenValidTimeINMinutes);
            return Jwts.builder()
                    .claims(claimsBuilder.build())
                    .issuedAt(now)
                    .expiration(calendar.getTime())
                    .signWith(Keys.hmacShaKeyFor(

accessTokenKey.getBytes(StandardCharsets.UTF_8)),
                            Jwts.SIG.HS256)
                    .compact();
        }

        public boolean isJWTValid(String token, String tokenKey)
{
            boolean isValid = false;
            SecretKey key =
Keys.hmacShaKeyFor(tokenKey.getBytes());
            try {

Jwts.parser().verifyWith(key).build().parseSignedClaims(token);
                isValid = true;
            } catch (Exception e) {
                log.info("Given token is not valid: {}",
e.getMessage());
            }
            return isValid;
        }

        public String createRefreshToken(User user){
            ClaimsBuilder claimsBuilder =
Jwts.claims().subject(user.getUsername());
            claimsBuilder.add(ROLE,
Collections.singleton(user.getRole().name()));
            Date now = new Date();
            Calendar calendar = Calendar.getInstance();
            calendar.setTime(now);
            calendar.add(Calendar.MINUTE,
refreshTokenValidTimeInMinutes);
            return Jwts.builder()
                    .claims(claimsBuilder.build())
                    .issuedAt(now)
                    .expiration(calendar.getTime())
                    .signWith(
```

```
Keys.hmacShaKeyFor(user.getRefreshTokenKey().getBytes(StandardChar
sets.UTF_8)),
                                Jwts.SIG.HS256)
                        .compact();
        }

        public String getEmailOutOfAccessToken(String token) {
            String[] splitToken = token.split("\\.");
            Base64.Decoder decoder = Base64.getUrlDecoder();
            String payload = new
String(decoder.decode(splitToken[1]));
            ObjectMapper objectMapper = new ObjectMapper();
            JsonNode jsonNode;
            try {
                jsonNode = objectMapper.readTree(payload);
            } catch (JsonProcessingException e) {
                throw new RuntimeException("Error parsing JSON",
e);
            }
            return jsonNode.path("sub").asText();
        }

        public String
getTokenFromHttpServletRequest(HttpServletRequest servletRequest)
{
            return Optional

.ofNullable(servletRequest.getHeader("Authorization"))
                        .filter(authHeader ->
authHeader.startsWith("Bearer "))
                        .map(token -> token.substring(7))
                        .orElse(null);
        }

        public String generateTokenKey() {
            return UUID.randomUUID().toString();
        }
                                                                }
```

Додаток Г

Лістинг файлу «UserSignInDTO.java»

```
package studyisnthard.SIH.DTO;

import jakarta.validation.constraints.Email;
import jakarta.validation.constraints.NotBlank;
import jakarta.validation.constraints.Pattern;
import jakarta.validation.constraints.Size;
import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@AllArgsConstructor
@NoArgsConstructor
@Builder
public class UserSignInDTO {
    @NotBlank
    @Email(message = "Email should be valid")
    private String email;

    @NotBlank
    @Size(min = 6, message = "Password must be at least 6
characters long")
    @Pattern(
            regexp = "^(?=.*[a-z])(?=.*[A-
Z])(?=.*\\d)(?=.*[@$!%*?&])[A-Za-z\\d@$!%*?&]{6,}$",
            message = "Password must be at least 6 characters
long, contain at least one uppercase letter, one lowercase letter,
one number, and one special character"
    )
    private String password;
}
```