Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра інформатики факультету інформатики

Розробка вебзастосунку для систематизації навчального процесу у автошколах

Текстова частина до курсової роботи

за спеціальністю 121 «Інженерія програмного забезпечення»

Керівник курсової роботи

доцент, к. ф-м.н. Нагірна А. М.

_____ *(підпис)*

“____” _____ 2024 р.

Виконала студентка Марченко Світлана Вікторівна

“____” _____ 2024 р.

Київ 2024

ЗМІСТ

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра інформатики

ЗАТВЕРДЖУЮ

Зав. кафедри інформатики

доцент, к.ф-м.н.

Гороховський С.С.

_____ *(підпис)*

“____” _____ 2023 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

на курсову роботу

студенту Марченко Світлані Вікторівні

факультету інформатики 3 курсу бакалаврської програми

Тема: **Розробка вебзастосунку для систематизації навчального процесу у автошколах**

Зміст ТЧ курсової роботи:

Індивідуальне завдання

Календарний план

Анотація

Вступ

Огляд рішень, що існують на ринку

Проєктування системи

Практична реалізація застосунку

Висновки

Джерела

Дата видачі “____” _____ 2023 р. Керівник Нагірна А. М.

_____ *(підпис)*

Завдання отримала _____ *(підпис)*

Тема: Розробка вебзастосунку для систематизації навчального процесу в автошколах.

Календарний план виконання роботи:

№ п/п	Назва етапу	Термін виконання	Примітка
1.	Отримання завдання на курсову роботу	14.09.2023	
2.	Аналіз матеріалів за темою	15.10.2023	
3.	Розробка практичної частини	20.12.2023	
4.	Написання текстової частини	15.04.2024	
5.	Коригування виконання роботи	02.05.2024	
6.	Створення презентації для доповіді	13.05.2024	
7.	Захист курсової роботи	24.05.2024	

Студент_____

Керівник _____

“____” _____

АНОТАЦІЯ

У ході виконання даної роботи було розроблено вебзастосунок для систематизації навчального процесу в автошколах. Для розробки було використано мову програмування Java та фреймворк Spring Boot для backend частини та мову програмування JavaScript та бібліотеку React для frontend частини.

Ключові слова: автошкола, водіння, Spring Boot, React, вебзастосунок

ВСТУП

Вміння керувати транспортним засобом – надзвичайно важлива навичка як у мирному житті, так і під час війни. З кожним днем все більше людей починають вчитися як для себе, так і для роботи або про всяк випадок (наприклад, при потребі евакуйовуватися через війну).

Однак у багатьох автошколах процес навчання застарілий, неавтоматизований та незручний: запис на практичні заняття здійснюється по телефону або через месенджер з інструктором, тому і студентам, і інструкторам, і адміністраторам необхідно самим вести їх облік, посилання конференції з теоретичними заняття необхідно шукати та розміщувати на інших ресурсах, їх записи, додаткові матеріали та презентації також необхідно шукати на сторонніх платформах. Таким чином студент може більше фокусуватися на організаційних моментах, ніж на процесі навчання та здобуття нових знань та навичок, мати складнощі у пошуках матеріалів з попередніх занять або курсів, а працівники школи витрачати забагато часу на механічну роботу з розміщення цих даних та обліку, яку можна автоматизувати та систематизувати.

Саме тому було вирішено створити застосунок для систематизації навчального процесу в автошколах, а саме, за допомогою якого можна буде бачити інформацію про актуальні набори на курси, залишати заявки на них, бачити розклад інструктора та записуватися/виписуватися з практичних занять, проходити теоретичні заняття, бачити статистику проходження курсів або занять, загальну статистику по курсах/працівниках/студентах (для адміністрації), читати та писати відгуки на курси, тощо. Особливістю застосунку стало те, що його бізнес логіка орієнтована на процеси та особливості отримання водійського посвідчення в Україні.

При розробці вебзастосунку використовувалася мова програмування Java та фреймворк Spring Boot для серверної частини, мова програмування JavaScript та бібліотека React для клієнтської частини та інші додаткові бібліотеки.

Метою роботи є створення вебзастосунку для автошкіл, який допоможе систематизувати навчальний процес, тим самим покращить його якість.

Завданням роботи є розробка вебзастосунку для систематизації навчального процесу у автошколі, який буде мати потрібні функції як для навчальної платформи для курсів двох типів (теоретичного та практичного), так і для обліку даних.

Об’єктом дослідження є огляд існуючих застосунків, які можна використати для систематизації навчального процесу в автошколах.

Предметом дослідження є розробка вебзастосунку, який буде мати функції, що зазначені у розділах роботи.

Дана курсова робота складається з анотації, вступу, трьох розділів, висновків та джерел. У першому розділі проводиться огляд готових рішень, що існують на ринку та відбувається постановка задачі. Другий розділ описує проєктування системи: структура бази даних, архітектура застосунку, засоби та середовища розробки тощо. Третій розділ описує реалізований застосунок та його особливості.

РОЗДІЛ 1. Огляд рішень, що існують на ринку

1.1 Moodle

Moodle – модульне об’єктно-орієнтоване динамічне навчальне середовище (або LMS (learning management system), CMS (content management system), VLE (virtual learning environment)), що має багато інструментів для електронного навчання. Близько двох третин ЗВО Європи використовують його [1].

Перевагами цієї системи є те, що вона є безкоштовною та open-source, її можна налаштувати під потреби організації, має велику кількість функцій (від створення курсів та завдань до листування та календаря). Однак систему досить складно налаштувати, також велика частина функцій системи створювалася саме під специфіку ЗВО, тому вони можуть бути не потрібними або вимагати суттєвої кастомізації під потреби сфери автошколи, а також необхідно мати окрему платформу для пошуку студентів на курси.

1.2 KUTS

KUTS – український маркетплейс онлайн-освіти, який пропонує можливість використовувати їх LMS систему [2].

Перевагами використання цієї системи є зручний конструктор курсів, можливість додати 10 різних варіантів тестувань, перегляд детальної аналітики та звітів, перегляд календарю подій, сповіщень, додаткових матеріалів, а також генерація цифрових дипломів. Недоліками використання є вартість (20% від вартості курсу), необхідність використання окремого ресурсу для пошуку учнів на курси та складність підбору зручної форми навчання для практичного курсу.

1.3 Microsoft Teams

Microsoft Teams – платформа для командної роботи від Microsoft [3].

Перевагами використання є те, що при використанні цієї системи не потрібно використовувати додаткові рішення для нарад (наприклад, Google Meet, Skype) та зберігання записів (наприклад, Google Drive, YouTube), можливість використання чатів, команд та створення завдань. Недоліками є складність введення аналітики, перевантаження непотрібними для автошколи функціями, використання окремого ресурсу для реклами курсів та необхідність підбору формату (або іншої платформи) для введення практичних курсів.

1.4 Постановка задачі

Проаналізовані системи на ринку хоч і пропонують варіанти зручної онлайн освіти, однак їх важко адаптувати під формат автошколи, а саме: поєднання двох різних типів курсів (теоретичні та практичні), функціоналу для записування/виписування для практичних занять, залучення нових учнів, специфічна інформація (категорія водійського посвідчення), яка не вимагається у інших нішах тощо.

Основним завданням курсової роботи є розробка застосунку для систематизації навчального процесу у автошколах з врахуванням особливості галузі, тобто на створеній платформі необхідно мати можливість створювати теоретичні та практичні курси, залишати заявку на проходження та проходити їх, переглядати календар подій, записуватися та виписуватися з практичних занять, залишати відгуки на курси, проходити заняття як у синхронному так і у асинхронному режимі, бачити статистику проходження та мати доступ до загальних модулів зі статистичними даними щодо працівників, курсів та студентів. Використання

застосунку має полегшити організацію навчального процесу як для студентів, так і для викладачів, адміністраторів та інструкторів.

РОЗДІЛ 2. Проєктування системи

2.1 Структура бази даних

2.1.1 ER – модель

Для проєктування структури бази даних застосунку було створено ER – модель (див. Рисунок 1).

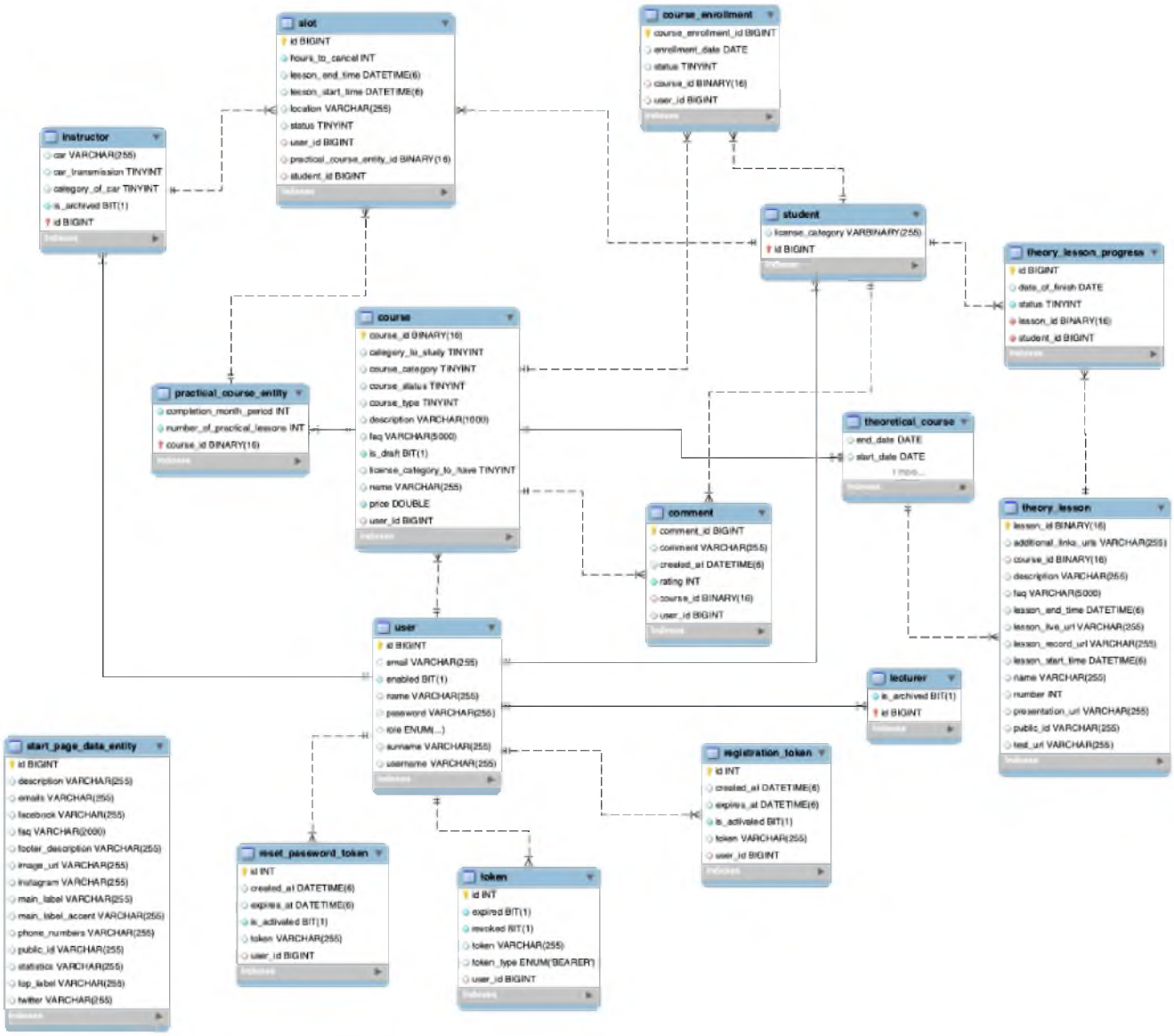

Рисунок 1. ЕР-модель застосунку

ЕР-модель була створена за допомогою програми MySQL Workbench.

Далі ЕР-модель було перетворено у реляційну.

2.1.2 Реляційна модель

USER, відповідає сутності “user”

Ключ	Ім’я атрибуту	Тип атрибуту	NULL/NOT NULL	FK: ON DELETE, ON UPDATE
PK	id	BIGINT	NOT NULL	
	name	Varchar (45)	NOT NULL	
	surname	Varchar (45)	NOT NULL	
	username	Varchar (45)	NOT NULL	
	email	Varchar (45)	NOT NULL	
	password	Varchar (255)	NOT NULL	
	role	Varchar (255)	NOT NULL	
	enabled	Boolean	NOT NULL	

Таблиця “USER” відображає загальну інформацію про користувача, яка є у всіх користувачів системи, а саме: ід, ім’я, прізвище, нікнейм, електронна пошта, пароль, роль та чи активований акаунт.

STUDENT, відповідає сутності “student”

Ключ	Ім’я атрибуту	Тип атрибуту	NULL/NOT NULL	FK: ON DELETE, ON UPDATE
PK, FK	user_id	BIGINT	NOT NULL	On delete: cascade, on update: cascade
	license_category	Varbinary (255)	NOT NULL	

Таблиця “STUDENT” є похідною від таблиця “USER” та містить інформацію, яка властива лише користувачам з роллю студент, а саме: наявні у нього категорії

водійського посвідчення. Вона пов’язана з таблицею “USER” за допомогою FK user_id.

LECTURER, відповідає сутності “lecturer”

Ключ	**Ім’я атрибуту**	**Тип атрибуту**	**NULL/NOT NULL**	**FK: ON DELETE, ON UPDATE**
PK, FK	user_id	BIGINT	NOT NULL	On delete: cascade, on update: cascade
	is_archived	Boolean	NOT NULL	

Таблиця “LECTURER” є похідною від таблиця “USER” та містить інформацію, яка властива лише користувачам з роллю лектор, а саме: чи він архівований у системі. Вона пов’язана з таблицею “USER” за допомогою FK user_id.

INSTRUCTOR, відповідає сутності “instructor”

Ключ	**Ім’я атрибуту**	**Тип атрибуту**	**NULL/NOT NULL**	**FK: ON DELETE, ON UPDATE**
PK, FK	user_id	BIGINT	NOT NULL	On delete: cascade, on update: cascade
	is_archived	Boolean	NOT NULL	
	car	Varchar (255)	NOT NULL	
	categoryOfCar	Tinyint	NOT NULL	
	carTransmission	Tinyint	NOT NULL	

Таблиця “INSTRUCTOR” є похідною від таблиця “USER” та містить інформацію, яка властива лише користувачам з роллю інструктор, а саме: чи він архівований, машина, категорія машини та тип коробки передач у машині. Вона пов’язана з таблицею “USER” за допомогою FK user_id.

COURSE, відповідає сутності “course”

Ключ	**Ім’я атрибуту**	**Тип атрибуту**	**NULL/NOT NULL**	**FK: ON DELETE, ON UPDATE**
PK	course_id	Binary (16)	NOT NULL	
FK	user_id	BIGINT	NOT NULL	On delete: no action, on update: cascade
	name	Varchar (255)	NOT NULL	
	description	Varchar (1000)	NOT NULL	
	license_category_to_have	Tinyint	NOT NULL	
	category_to_study	Tinyint	NOT NULL	
	course_type	Tinyint	NOT NULL	
	course_category	Tinyint	NOT NULL	
	price	Double	NOT NULL	
	course_status	Tinyint	NOT NULL	
	is_draft	Boolean	NOT NULL	
	faq	Varchar (5000)	NULL	

Таблиця “COURSE” забезпечує зберігання загальної інформації про курс (яка є як у теоретичного, так і у практичного курсу), таку як: ід, user_id (унікальний ідентифікатор автора курсу), назву, опис, категорію посвідчення, яку необхідно мати для навчання на курсі, категорія навчання, тип курсу, категорія курсу, вартість, статус курсу, чи чернетка він та питання-відповіді до курсу. Таблиця пов’язана з таблицею “USER” за допомогою FK user_id.

THEORETICAL_COURSE, відповідає сутності “theoretical_course”

Ключ	**Ім’я атрибуту**	**Тип атрибуту**	**NULL/NOT NULL**	**FK: ON DELETE, ON UPDATE**
PK, FK	course_id	Binary (16)	NOT NULL	On delete: cascade on update: cascade
	start_date	Date	NULL	
	end_date	Date	NULL	

Таблиця “THEORETICAL_COURSE” є похідною від таблиця “COURSE” та складається з атрибутів, які специфічні лише теоретичним курсам, а саме: дата початку та кінця курсу. Вона пов’язана з таблицею “COURSE” за допомогою FK course_id.

PRACTICAL_COURSE, відповідає сутності “practical_course”

Ключ	**Ім’я атрибуту**	**Тип атрибуту**	**NULL/NOT NULL**	**FK: ON DELETE, ON UPDATE**
PK, FK	course_id	Binary (16)	NOT NULL	On delete: cascade on update: cascade
	completion_month_period	INT	NOT NULL	
	number_of_practical_lessons	INT	NOT NULL	

Таблиця “PRACTICAL _COURSE” є похідною від таблиця “COURSE” та складається з атрибутів, які специфічні лише практичним курсам, а саме: кількість практичних занять та кількість місяців доступу до навчання. Вона пов’язана з таблицею “COURSE” за допомогою FK course_id.

COMMENT, відповідає сутності “comment”

Ключ	**Ім’я атрибуту**	**Тип атрибуту**	**NULL/NOT NULL**	**FK: ON DELETE, ON UPDATE**
PK	id	BIGINT	NOT NULL	
FK	course_id	Binary (16)	NOT NULL	On delete: cascade, on update: cascade
FK	user_id	BIGINT	NOT NULL	On delete: set null, on update: cascade
	comment	Varchar (1000)	NOT NULL	
	created_at	Datetime	NOT NULL	
	rating	INT	NOT NULL	

Таблиця “COMMENT” забезпечує зберігання інформації про коментарі до курсів, а саме: ід, ід курсу, до якого написаний коментар, ід користувача, який залишив відгук, текст коментаря, дата та час створення та оцінка від користувача. Таблиця пов’язана з таблицею “USER” за допомогою FK user_id, а з таблице “COURSE” за допомогою FK course_id.

SLOT, відповідає сутності “slot”

Ключ	**Ім’я атрибуту**	**Тип атрибуту**	**NULL/NOT NULL**	**FK: ON DELETE, ON UPDATE**
PK	id	BIGINT	NOT NULL	
FK	practical_course_id	BIGINT	NULL	On delete: cascade, on update: cascade
FK	student_id	BIGINT	NULL	On delete: cascade, on update: cascade

FK	instructor_id	BIGINT	NOT NULL	On delete: cascade, on update: cascade
	lesson_start_time	Datetime	NOT NULL	
	lesson_end_time	Datetime	NOT NULL	
	location	Varchar (255)	NOT NULL	
	status	Tinyint	NOT NULL	
	hours_to_cancel	INT	NOT NULL	

Таблиця “SLOT” забезпечує зберігання інформації про часові слоти для занять практичного курсу, а саме: ід, ід практичного курсу, ід студента, який забронював слот, ід інструктора, який створив слот, дата та час початку заняття, дата та час кінця заняття, місце заняття, статус та за скільки годин до заняття його можна скасувати. Таблиця пов’язана з таблицею “STUDENT” за допомогою FK student_id, з таблице “PRACTICAL_COURSE” за допомогою FK course_id, з таблицею “INSTRUCTOR” за допомогою FK instructor_id.

COURSE_ENROLLMENT, відповідає сутності “course_enrollment”

Ключ	**Ім’я атрибуту**	**Тип атрибуту**	**NULL/NOT NULL**	**FK: ON DELETE, ON UPDATE**
PK	id	BIGINT	NOT NULL	
FK	student_id	BIGINT	NOT NULL	On delete: cascade, on update: cascade
FK	course_id	BIGINT	NOT NULL	On delete: cascade, on update: cascade
	enrollment_date	Date	NOT NULL	
	status	Tinyint	NOT NULL	

Таблиця “COURSE_ENROLLMENT” забезпечує зберігання інформації про заявки на курси, а саме: ід, ід студента, який залишив заявку, ід курсу, на який була

залишена заявка, дата створення заявки та її статус. Таблиця пов'язана з таблицею "STUDENT" за допомогою FK student_id, з таблице " COURSE" за допомогою FK course_id.

THEORY_LESSON, відповідає сутності "theory_lesson"

Ключ	**Ім'я атрибуту**	**Тип атрибуту**	**NULL/NOT NULL**	**FK: ON DELETE, ON UPDATE**
PK	id	Binary (16)	NOT NULL	
FK	course_id	BIGINT	NOT NULL	On delete: cascade, on update: cascade
	number	INT	NOT NULL	
	name	Varchar (255)	NOT NULL	
	description	Varchar (1000)	NOT NULL	
	lesson_start_time	Datetime	NOT NULL	
	lesson_end_time	Datetime	NOT NULL	
	lesson_live_url	Varchar (255)	NULL	
	lesson_record_url	Varchar (255)	NULL	
	test_url	Varchar (255)	NULL	
	presentation_url	Varchar (255)	NULL	
	public_id	Varchar (255)	NULL	
	additional_links_urls	Varchar (1000)	NULL	
	faq	Varchar (5000)	NULL	

Таблиця "THEORY_LESSON" забезпечує зберігання інформації про заняття теоретичного курсу, а саме: ід, ід курсу, номер заняття, назва, опис, дата та час початку, дата та час закінчення заняття, посилання на зустріч, посилання на запис, посилання на тест, посилання на презентацію, яка збережена за допомогою хмарного сервісу, ід збереження презентації у сервісі, посилання з додатковими матеріалами та запитання-відповіді після заняття. Таблиця пов'язана з таблицею з таблице " COURSE" за допомогою FK course_id.

THEORY_LESSON_PROGRESS, відповідає сутності “theory_lesson_progress”

Ключ	**Ім’я атрибуту**	**Тип атрибуту**	**NULL/NOT NULL**	**FK: ON DELETE, ON UPDATE**
PK	id	BIGINT	NOT NULL	
FK	lesson_id	Binary (16)	NOT NULL	On delete: no action, on update: cascade
FK	student_id	BIGINT	NOT NULL	On delete: cascade, on update: cascade
	status	Tinyint	NOT NULL	
	date_of_finish	Date	NULL	

Таблиця “THEORY_LESSON_PROGRESS” забезпечує зберігання інформації про проходження заняття курсу студентом, а саме: ід, ід заняття, ід студента, статус та дата проходження заняття. Таблиця пов’язана з таблицею з таблицею “ THEORY_LESSON” за допомогою FK lesson_id, а з таблицею “STUDENT” за допомогою FK student_id.

RESET_PASSWORD_TOKEN, відповідає сутності “reset_password_token”

Ключ	**Ім’я атрибуту**	**Тип атрибуту**	**NULL/NOT NULL**	**FK: ON DELETE, ON UPDATE**
PK	id	INT	NOT NULL	
FK	user_id	BIGINT	NOT NULL	On delete: cascade, on update: cascade
	token	Varchar (255)	NOT NULL	
	expires_at	Datetime	NOT NULL	
	is_activated	Boolean	NOT NULL	

Таблиця “RESET_PASSWORD_TOKEN” забезпечує зберігання інформації про токени для скидання паролю, а саме: ід, ід користувача, токен, дата та час завершення дії та чи був токен активований у системі. Таблиця пов’язана з таблицею з таблицею “USER” за допомогою FK user_id.

TOKEN, відповідає сутності “token”

Ключ	**Ім’я атрибуту**	**Тип атрибуту**	**NULL/NOT NULL**	**FK: ON DELETE, ON UPDATE**
PK	id	INT	NOT NULL	
FK	user_id	BIGINT	NOT NULL	On delete: cascade, on update: cascade
	token	Varchar (255)	NOT NULL	
	revoked	Boolean	NOT NULL	
	expired	Boolean	NOT NULL	
	token_type	Tinyint	NOT NULL	

Таблиця “TOKEN” забезпечує зберігання інформації про токени оновлення (refresh token) користувачів.

REGISTRATION_TOKEN, відповідає сутності “registration_token”

Ключ	**Ім’я атрибуту**	**Тип атрибуту**	**NULL/NOT NULL**	**FK: ON DELETE, ON UPDATE**
PK	id	INT	NOT NULL	
FK	user_id	BIGINT	NOT NULL	On delete: cascade, on update: cascade
	token	Varchar (255)	NOT NULL	
	expires_at	Datetime	NOT NULL	
	is_activated	Boolean	NOT NULL	

Таблиця “REGISTRATION _TOKEN” забезпечує зберігання інформації про токени для реєстрації (підтвердження електронної пошти), а саме: ід, ід користувача, токен, дата та час завершення дії та чи був токен активований у системі. Таблиця пов’язана з таблицею з таблицею “USER” за допомогою FK user_id.

START_PAGE_DATA, відповідає сутності “start_page_data”

Ключ	**Ім’я атрибуту**	**Тип атрибуту**	**NULL/NOT NULL**	**FK: ON DELETE, ON UPDATE**
PK	id	BIGINT	NOT NULL	
	top_label	Varchar (255)	NULL	
	imageUrl	Varchar (255)	NULL	
	public_id	Varchar (255)	NULL	
	main_label	Varchar (255)	NULL	
	main_label_accent	Varchar (255)	NULL	
	description	Varchar (255)	NULL	
	footer_description	Varchar (255)	NULL	
	instagram	Varchar (255)	NULL	
	twitter	Varchar (255)	NULL	
	facebook	Varchar (255)	NULL	
	emails	Varchar (255)	NULL	
	phone_numbers	Varchar (255)	NULL	
	statistics	Varchar (255)	NULL	
	faq	Varchar (2000)	NULL	

Таблиця “START_PAGE_DATA” забезпечує зберігання інформації з текстом та посиланням на зображення головної сторінки застосунку та нижнього меню.

2.2 Архітектура застосунку

Для застосунку був обраний клієнт-серверний шаблон архітектури, у якому клієнтська частина взаємодіє з користувачем та відправляє запити на серверну частину, яка впроваджує бізнес логіку та опрацьовує запити від клієнта. Серверна

частина взаємодіє з базою даних. Перевагами цього шаблону є централізоване зберігання даних (це полегшує керуваннями ними, забезпечує їх узгодженість), масштабованість, можливість повторного використання серверу для різних клієнтів та безпека. Недоліками є надійність (тобто існує залежність від доступності серверу, адже без його роботи станеться збій, який вплине на всю мережу) та збільшена затримка через мережевий зв'язок [4]. Для застосунку цей шаблон був обраний через легшу інтеграцію наступних продуктів (наприклад, мобільних застосунків) у систему, адже клієнт та сервер розробляються незалежно.

Для серверної частини застосунку був обраний багаторівневий підхід для архітектури, який складається з 4 рівнів: Presentation Layer (рівень контролерів), Business Layer (рівень сервісів), Persistence Layer (рівень репозиторіїв) та Database Layer (база даних) [5]. Перевагою цього підходу є розділення відповідальності між рівнями та їх відокремленість (тобто чітка межа між «входом у сервер», бізнес логікою та доступом до даних) та полегшення тестування, адже кожен рівень можливо тестувати окремо. Рівень контролеру є «вхідною точкою» у сервер, тобто відповідає за отримання запитів на API від клієнтів, проводить аутентифікацію. Класи цього рівня позначаються анотацією @RestController. Рівень репозиторію забезпечує взаємодію з базою даних. Класи/інтерфейси цього рівня позначаються анотацією @Repository. Рівень сервісу відповідає за бізнес логіку застосунку, він є посередником між репозиторієм (для зберігання та отримування інформації з бази даних) та контролером (для отримання запитів). Класи цього рівня позначаються анотаціє @Service [6]. Також кожен рівень оперує своїми моделями даних: контролер – DTO (data transfer object) класами, сервіс – звичайними об'єктами (POJO), репозиторій – сутностями (класи з анотацією @Entity). Перетворення об'єктів цих моделей відбувається за допомогою методів інтерфейсу з анотацією @Mapper з бібліотеки MapStruct, код для яких генерується автоматично під час компіляції застосунку. Цей код можна знайти у /target/generated-sources/annotations/ (див. Рисунок 2). Використання згенерованих класів для перетворення об'єктів

значно пришвидшує розробку, адже позбавляє розробника від написання однотипного коду.

```
@Generated(
    value = "org.mapstruct.ap.MappingProcessor",
    date = "2024-05-02T18:05:08+0300",
    comments = "version: 1.5.5.Final, compiler: javac, environment: Java 17.0.8.1 (Amazon.com Inc.)"
)
@Component
public class CommentMapperImpl implements CommentMapper {

    2 usages
    @Override
    public CommentDto commentToCommentDto(Comment comment) {
        if ( comment == null ) {
            return null;
        }

        CommentDto commentDto = new CommentDto();

        commentDto.setId( comment.getId() );
        commentDto.setStudent( studentToStudentDto( comment.getStudent() ) );
        commentDto.setCourse( courseToCourseDto( comment.getCourse() ) );
        commentDto.setComment( comment.getComment() );
        commentDto.setCreatedAt( comment.getCreatedAt() );
        commentDto.setRating( comment.getRating() );

        return commentDto;
    }
```

Рисунок 2. Приклад згенерованого коду

Взаємодія між клієнтом та сервером відбувається за допомогою методів HTTP (Hypertext Transfer Protocol) протоколу. Метод GET використовується для отримання даних з серверу. Метод POST використовується для створення нового запису на сервері. Метод PUT використовується для повного оновлення запису на сервері. Метод DELETE використовується для видалення запису. Метод PATCH використовується для часткового оновлення запису, без переписування тих полів, які не були передані [7].

2.3 Засоби розробки

2.3.1 Клієнтська частина застосунку

Клієнтська частина застосунку написана за допомогою мови програмування JavaScript та JavaScript-бібліотеки React [8]. Перевагами використання цієї бібліотеки є те, що однієї з її фундаментальних концепцій є перевикористовувані компоненти, з яких і формується весь додаток. Їх можна комбінувати, об'єднувати декілька невеликих компонентів у один більший, вони модульні, тобто їх можна розробляти та тестувати незалежно. Наступною перевагою є велика спільнота та наявна детальна документація, тобто знайти рішення потенційній проблемі буде легше, адже існує багато форумів, тематичних сайтів та туторіалів. Також у даній бібліотеці використовується Virtual DOM (Document Object Model), що покращує ефективність додатку, адже мінімізує кількість оновлень реального DOM. На додачу React «дружній» до SEO, тобто пошуковим системам (Google, Yahoo тощо) легше індексувати такі застосунки. З недоліків можна виділити складність опанування JSX (JavaScript XML), тобто написання HTML коду у JS файлах, станів і пропсів за допомогою яких передаються дані між компонентами, розуміння односпрямованого потоку даних, тобто передача відбувається тільки від батьківських до дочірніх компонентів [9].

Для стилів був використаний CSS- фреймворк Tailwind CSS [10]. Перевагами використання саме цього фреймворку є пришвидшення написання CSS стилів, адже стилі пишуться одразу у компонентах, а не у окремому файлі, наявність готових класів, які можна використовувати у проєкті, краща підтримуваність. З мінусів можна помітити гіршу читабельність коду, особливо при додаванні великої кількості стилів та збільшений розмір CSS файлу проєкту [11].

Для завантаження додаткових бібліотек було використано Node Package Manager (скорочено npm).

2.3.2 Серверна частина застосунку

Серверна частина застосунку за допомогою мови програмування Java та фреймворку Spring Boot [12].

Управління та збирання проєкту відбувається за допомогою Maven. Уся інформація про проєкт та його конфігурацію міститься у pom (Project Object Model) файлі, який має XML структуру. Необхідні залежності знаходяться у частині <dependencies> (див. Рисунок 3).

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
```

Рисунок 3. Приклад залежності у pom.xml

Потрібні для проєкту залежності завантажуються у локальний репозиторій на машині розробника. Залежності, які були не знайдені у локальному репозиторії, будуть завантажені з центрального, який наповнює спільнота Maven, або з віддаленого, якщо такий вказаний у конфігураційному файлі [13].

Для доступу до бази даних було використано Spring Data JPA [14], такий собі шар абстракції над Java Persistence API. Його використання значно пришвидшує процес розробки (порівняно з використанням Spring JDBC) через використання ORM (у застосунку використовується Hibernate) та JPA репозиторіїв. Більш низькорівневий аналог Spring JDBC дає прямий контроль над SQL запитами та збільшує продуктивність, однак вимагає написання великої кількості шаблонного коду навіть для базових Create, Read, Update, Delete (надалі CRUD) операцій та ускладнює роботу зі складними моделями даних через те, що перетворення Java об’єктів на таблиці бази даних пишуться розробником вручну, а не створюються автоматично. Також при використанні JDBC необхідно використовувати ручне

керування транзакціями та з'єднанням з базою даних [15]. На противагу йому Spring Data JPA дозволяє абстрагуватися від явного SQL, що значно підвищує рівень абстракції. JPA репозиторіїв надають базові CRUD операції та дозволяють легко писати прості запити (за умови використання спеціального іменування методів інтерфейсу) [16]. Для складних запитів до бази даних є можливість використовувати Java Persistence Query Language (скорочено JPQL) разом з анотацією @Query. Його синтаксис схожий на SQL. На рисунку показано (див. Рисунок 4), що відносно простий запит (наприклад, «Знайти усі курси, у яких ІД автора = переданому») написаний за допомогою спеціального іменування назви метода, а складніший запит вже через JPQL та @Query. Для відображення об'єктів у таблицях реляційної бази даних використовується анотація @Entity.

```
13  @Query("SELECT c FROM TheoreticalCourseEntity c WHERE c.isDraft = false AND c.startDate >= :currentDate")
14  List<TheoreticalCourseEntity> findAllAvailableCourses(LocalDate currentDate);
    1 usage   Svitlana-Marchenko
15  List<TheoreticalCourseEntity> findAllByAuthorId(Long id);
```

Рисунок 4. Метод репозиторію з анотацією @Query та без неї

2.3.3 База даних

У якості бази даних було обрано реляційну базу даних MySQL [17]. Використання реляційної бази даних було обрано через те, що дані застосунку структуровані та мають багато зв'язків одне з одним.

Для збереження файлів з презентаціями для теоретичних занять та зображення головної сторінки було використано хмарну систему Cloudinary [18].

2.3.4 Swagger

Для полегшення ведення документації API було підключено open-source фреймворк Swagger [19]. Згенерована документація містить інформацію про

ендпоінти серверу (див. Рисунок 5). Зручним є перегляд деталей ендпоінту: які передані параметри очікує сервер та формат відповіді (див. Рисунок 6).

Рисунок 5. Список ендпоінтів серверу

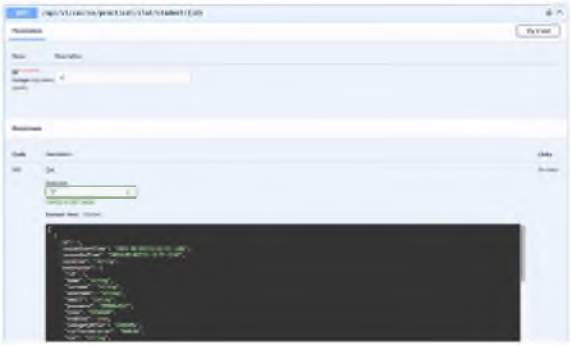

Рисунок 6. Деталі ендпоінту серверу

Також у документації є структури Data Transfer Object (надалі DTO) об’єктів. JSON з структурою DTO об’єкту приймає і віддає серверна частина (див. Рисунок 7).

Рисунок 7. Структура DAO об’єктів, які приймає сервер

2.3.5 Postman

Для тестування серверної частини була створена Postman колекція (група збережених запитів) до API (див. Рисунок 8). Це полегшує процес розробки, адже не потрібний кожен раз вводити інформацію для запиту (тип авторизації, параметри, заголовки, тіло запиту тощо) [20]. Використання колекцій зручно як при розробці серверу для ручного тестування ендпоінтів, так і при розробці клієнтської частини

та її взаємодії з серверною, адже при виникненні помилок можна легко зрозуміти чи це проблема на стороні клієнта, чи на стороні сервера.

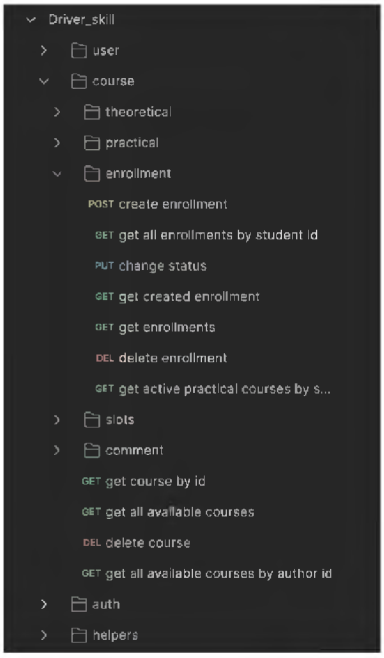

Рисунок 8. Postman колекція створеного застосунку

2.3.6 Профіль розробки

Для зручності було створено два профілі розробки: prod (для використання користувачами) та dev (розробницький). Вони підключені до різних баз даних: prod – MySQL, dev – H2. Їх конфігурації визначені у окремих properties файлах (див. Рисунок 9).

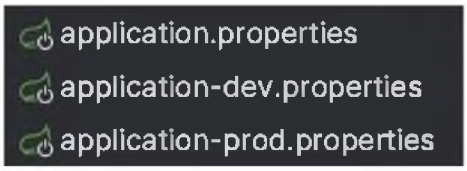

Рисунок 9. Properties файли серверу проєкту

Метою цього розділення стало те, що при початковій розробці часто змінюється схема бази даних, необхідно додавати/видаляти/редагувати дані для перевірки ендпоінтів, і саме in-memory (тобто яка зберігається у оперативній пам'яті) база даних добре для цього підходить. При кожній зупинці сервреру вся інформації у ній видаляється. Для наповнення тестовими даними при запуску серверу був доданий компонент CommandLineRunner, у якому прописано створення всіх необхідних тестових даних.

2.4 Середовище розробки

Для написання серверної частини застосунку було використано середовище розробки IntelliJ IDEA від компанії JetBrains, для клієнтської – WebStorm від компанії JetBrains. Для розгортання тестового SMTP (Simple Mail Transfer Protocol) серверу був використаний Docker.

2.5 Опис користувачів системи та їх можливостей

Система має 5 категорій користувачів: незареєстрований користувач, учень, лектор, інструктор та адміністратор.

Незареєстрований користувач може:

- переглядати доступні до запису теоретичні та практичні курси
- фільтрувати (за категорією курсу, типом курсу, категорією навчання, необхідною категорією для навчання), шукати за назвою/автором та сортувати за ціною/назвою
- переглядати детальну інформацію про теоретичний та практичний курс (назва, опис, загальна інформація, заняття (кількість для практичного, назви, опис, дати для теоретичного), поширені питання, відгуки та рейтинг курсу заснований на відгуках)
- зареєструватися або авторизуватися

- переглядати інформацію про автошколу, а саме: загальну інформацію, контакти (телефони, електронні пошти, соцмережі), поширені запитання
- переглядати профіль лектора/інструктора та бачити їх актуальні курси

Учень може:

- усе, що може незареєстрований користувач (крім реєстрації/авторизації)
- залишати заявки на курс
- писати коментарі з оцінкою для курсу
- редагувати/видаляти профіль та змінювати пароль
- переглядати календар практичних та теоретичних занять
- переглядати свої заявки, видаляти їх, якщо вони неопрацьовані адміністратором

Лектор може:

- усе, що може незареєстрований користувач (крім реєстрації/авторизації)
- редагувати/видаляти профіль та змінювати пароль
- переглядати особистий календар та бачити повний розклад занять
- створювати/редагувати/видаляти (неопубліковані) курси
- створювати/редагувати/видаляти заняття
- переглядати деталі своїх курсів включно з таблицею зі студентами, які проходять або завершили проходження курсу
- переглядати деталі уроків включно з таблицями зі студентами, які проходять, завершили проходження або ще не починали проходити заняття і кругова діаграма з цими даними (кожен сектор відповідає за кількість студентів, які …)
- переглядати деталі проходження курсу конкретним студентом, включно з переглядом уроків, які він пройшов, почав та ще не починав та круговою діаграмою проходження курсу (кожен сектор відповідає за кількість уроків, які студент …)

Інструктор може:

- усе, що може незареєстрований користувач (крім реєстрації/авторизації)
- редагувати/видаляти профіль та змінювати пароль
- переглядати особистий календар та бачити створені/заброньовані (включно з курсом та студентом) слоти практичних занять
- створювати/видаляти (якщо слот не забронювали) слоти для практичних занять
- створювати/редагувати/видаляти (неопубліковані)/архівувати (опубліковані)/розархівовувати (архівовані або готуються до архівування) курси
- переглядати деталі своїх курсів включно з таблицею зі студентами, які проходять або завершили проходження курсу
- переглядати деталі проходження курсу конкретним студентом, включно з деталями пройдених та заброньованих занять (дата, тривалість заняття, місце початку, інструктор), кількість занять для завершення курсу, секторна діаграма з кількістю заброньованих, пройдених та необхідних для проходження занять)

Адміністратор може:

- усе, що може незареєстрований користувач (окрім реєстрації/ авторизації)
- редагувати/видаляти профіль та змінювати пароль
- створювати та архівувати/розархівовувати (тоді користувач не має можливості увійти в систему до його розархівування, аналог видалення, однак з можливим поверненням до роботи) лекторів та інструкторів
- підтверджувати/відхиляти заявки на курси від студентів, а також переглядати усі заявки на курси списком та у їх статистику у вигляді лінійної діаграми (кількість створених заявок за день)
- переглядати уроки теоретичних курсів
- переглядати інформацію про усіх співробітників у вигляді таблиці (нікнейм, ім’я, прізвище, електронна пошта, кількість курсів, статус (активний або

заархівований), категорія машини (для інструктора), машина (для інструктора), вид коробки передач (для інструктора)), а також додаткову інформацію про них, включно з календарем занять та курсами з усіма статусами з можливістю перейти до статистичних деталей певного курсу

- переглядати інформацію про усі курси у вигляді таблиці (назва, автор, категорія навчання, необхідна категорія для початку навчання, тип курсу, категорія курсу, ціна, статус, кількість активних студентів, кількість студентів, які пройшли курс), а також також додаткову інформацію про них, включно з таблицею зі студентами, які у процесі проходження курсу та які вже завершили його, списком уроків з можливістю перейти до статистики конкретного уроку, у якій наявні додаткові дані про урок, таблиці зі студентами, які проходять, пройшли та ще не починали проходити заняття і кругова діаграма з цими даними (кожен сектор відповідає за кількість студентів, які …)
- переглядати інформацію про усіх студентів у вигляді таблиці (нікнейм, ім’я, прізвище, електронна пошта, категорії посвідчень), а також додаткову інформацію про них, включно з календарем занять, курсами, які вже пройшов студент та які ще проходить
- редагувати головну сторінку, а саме: верхній напис (зазвичай використовується для назви автошколи), основний напис, акцентний напис, опис, опис у нижньому меню (footer), посилання на Instagram, Facebook, X (Twitter) сторінки, фото, список електронних поштових скриньок, список телефонів, список статистичної інформації та секцію поширені питання – відповіді

До того ж, якщо учень записаний на практичний курс він додатково може:

- записуватися/виписуватися з практичного заняття (за умови, що ще є можливість виписатися)

- бачити статистику проходження курс (деталі пройдених та заброньованих занять (дата, тривалість заняття, місце початку, інструктор), кількість занять для завершення курсу, секторна діаграма з кількістю заброньованих, пройдених та необхідних для проходження занять)
- переглядати розклад інструктора включно з усіма заброньованими слотами

До того ж, якщо учень записаний на теоретичний курс, він додатково може:

- бачити статистику проходження курс з переглядом уроків, які він пройшов, почав та ще не починав та круговою діаграмою проходження курсу (кожен сектор відповідає за кількість уроків, які студент …)

2.6 Безпека

2.6.1 Авторизація та аутентифікація

Для забезпечення аутентифікації та авторизації був використаний фреймворк Spring Security [21]. Аутентифікація полягає у перевірці чи користувач є тим, за кого себе видає. Авторизація визначає чи є доступ у певного користувача до певного ресурсу.

Для авторизації у додатку використовуються JSON Web Token (скорочено JWT). Після успішної аутентифікації сервер повертає клієнту два токени: токен доступу (access token) та токен для оновлення токену доступу (refresh token). Токен доступу має обмежений та короткий термін дії (у застосунку - 15 хвилин). Клієнтська частина застосунку надсилає його у заголовку «Authorization» з префіксом «Bearer » при кожному запиті до захищених ресурсів серверу. Сервер отримує переданий токен з заголовку, перевіряє правильність підпису, термін дії, отримує дані про користувача, який здійснив запит, додає їх у SecurityContextHolder та передає обробку запиту далі. Токен оновлення має набагато більший термін дії (у застосунку - 24 години) та використовується для отримання нового токену доступу без необхідності введення користувачем логіну та паролю. Цей токен зберігається у HTTP-only cookies. При запиті на оновлення токену доступу сервер

перевіряє токен оновлення на правильність підпису, термін дії та чи не був він скасований достроково (наприклад, при виході з системи), отримує дані про користувача та повертає новий токен доступу та той самий токен оновлення.

Підхід до використання токену доступу разом з токеном оновлення допомагають підтримувати баланс між зручністю користувача та безпекою застосунку [22]. Тобто це дає можливість суттєво зменшити час дії токену доступу без необхідності постійного (кожен раз після закінчення дії токену доступу) запиту на введення облікових даних акаунту від користувача, що підвищує його User Experience. Якщо зловмисник отримає токен доступу, тоді доступ до платформи буде відбуватися ще певний невеликий проміжок часу поки цей токен не буде протермінований. При підозрі на викрадення токену оновлення його можна скасувати достроково, тобто в базі даних застосунку поле токену «revoked» буде змінено на true, і надалі при запиті на оновлення токену буде повертатится помилка.

2.6.2 Зберігання паролів

Паролі зберігаються у базі даних у зашифрованому вигляді. Це підвищує рівень безпеки застосунку, адже при несанкціонованому доступі до бази даних, пароль користувача не буде розкритий. Для їх шифрування використано BCryptPasswordEncoder (імплементація інтерфейсу PasswordEncoder у фреймворку Spring Security), який використовує хеш функцію bcrypt [23]. І саме хеш (результат виконання функції) зберігається у базі даних. Ця функція була створена Нільсом Провосом та Девідом Мазьєром на основі шифру Blowfish. Її перевагами є використання солі (рандомно згенерованого рядку), яка додається до кожного паролю перед хешуванням, тому навіть ідентичні паролі матимуть різний хеш. До того ж, цю функцію можна навмисно сповільнювати шляхом збільшення кількості ітерацій алгоритму. Саме тому вона є стійкою до атак з використанням грубої сили (brute force, тобто підбирання паролів), райдужної таблиці (rainbow table) перебору за словником (dictionary attack).

Ідея використання іншої відомої криптографічної функції для хешування SH256 була відхилена через більшу швидкість обчислення хешу (тобто ця функція є більш вразливою до атак методом перебору (brute force), адже швидше буде проведено обрахування) та гірші, порівняно з bcrypt, алгоритми додавання солі [24].

РОЗДІЛ 3. Практична реалізація застосунку

Основні сторінки інтерфейсу для усіх користувачів спільні, однак для кожної ролі існують унікальні сторінки. Основними кольорами застосунку є білий та блакитний.

3.1 Головний екран застосунку

Головним екраном застосунку є рекламний банер з інформацією про школу (див. Рисунок 10), QA секцією з поширеними питаннями та нижнє меню з контактами (див. Рисунок 11).

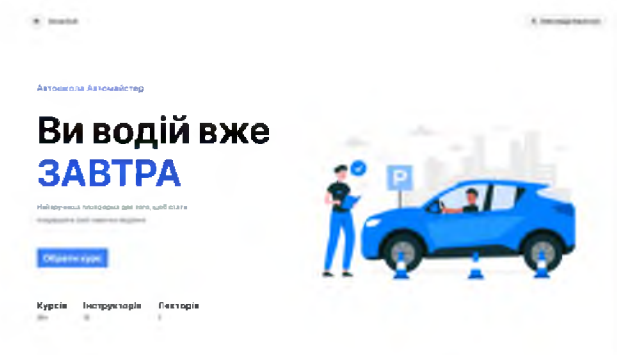

Рисунок 10. Головний екран застосунку

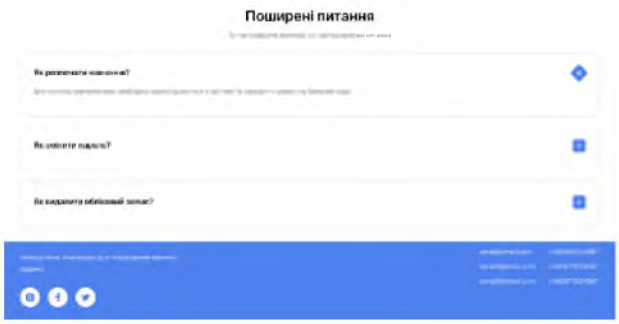

Рисунок 11. Секція QA

Цю інформацію може змінювати адміністратор через спеціальне меню для редагування (див. Рисунок 12).

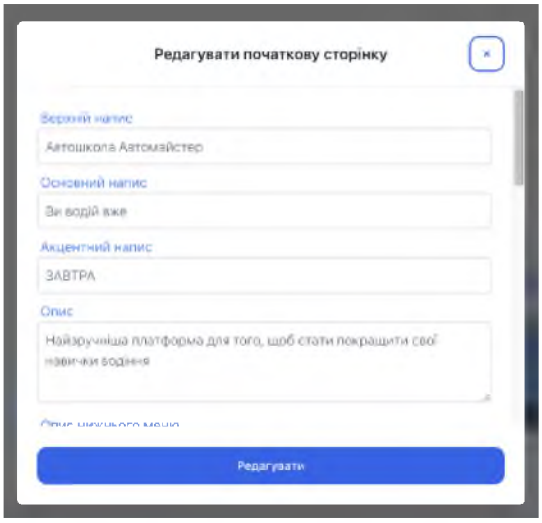

Рисунок 12. Меню для редагування головної сторінки

Перехід між сторінками застосунку здійснюється за допомогою бокового меню (див Рисунок 13). Користувачі з різними ролями мають різне наповнення цього меню.

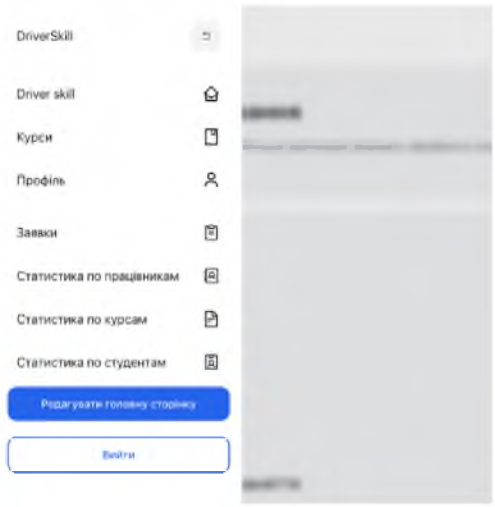

Рисунок 13. Бокове меню

3.2 Вибір курсу для навчання

Для вибору курсів для навчання необхідно перейти на сторінку зі списком доступних курсів (див. Рисунок 14). Вона доступна і для авторизованого користувача, і для неавторизованого. Список можна фільтрувати та сортувати.

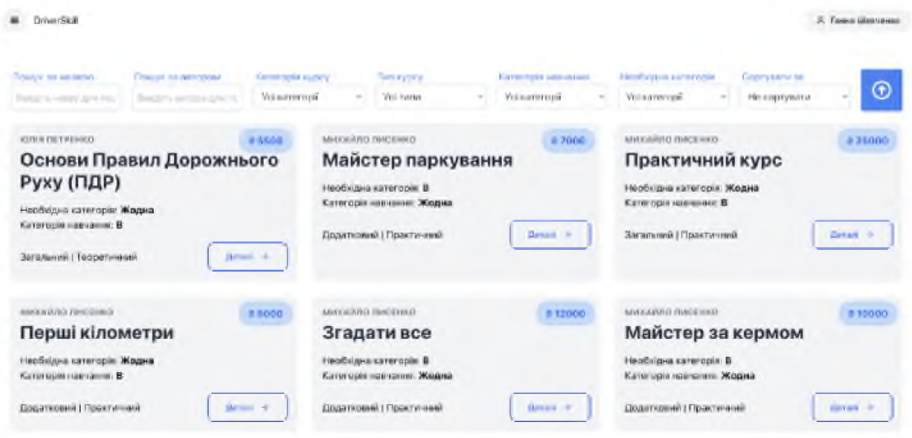

Рисунок 14. Сторінка зі списком доступних курсів

При перегляді детальної інформації про курс виникає різниця функціональності між авторизованим та неавторизованим користувачем. Авторизований може залишити заявку на курс та написати коментар, а неавторизований бачить спеціальний банер з закликом авторизуватися/зареєструватися. Для запису на курс необхідно залишити заявку, яка буде опрацьована адміністратором. Різниця між переглядом сторінки теоретичного та практичного курсу полягає у тому, що для теоретичного курсу можна побачити усі заняття, їх назви, описи та дати (див. Рисунок 15), а для практичного – лише кількість занять (див. Рисунок 16). QA секція є у обох видів курсів.

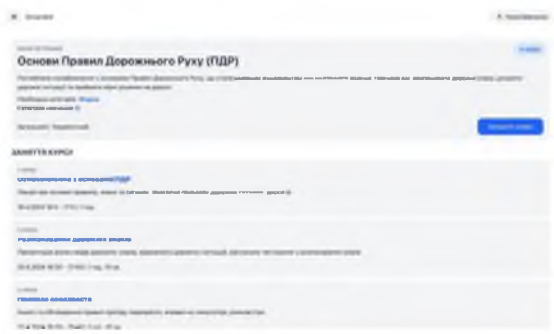

Рисунок 15. Сторінка теоретичного курсу (вигляд авторизованого користувача)

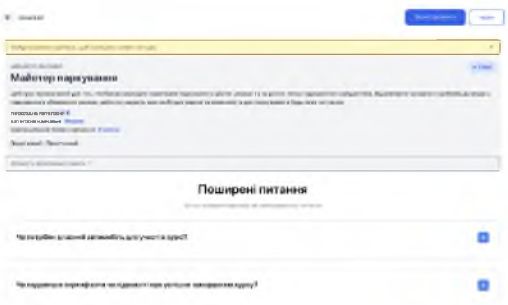

Рисунок 16. Сторінка практичного курсу (вигляд неавторизованого користувача

Зручним є перегляд відгуків на курс та його рейтингу (див. Рисунок 17). Користувачі бачать не лише обрахований рейтинг, але й відсоток відгуків з певною оцінкою.

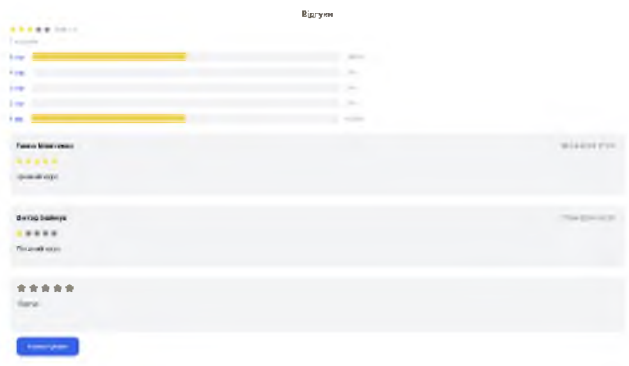

Рисунок 17. Перегляд відгуків на курс

3.3 Реєстрація та логін

Для реєстрації на курси та їх проходження користувачу необхідно зареєструватися (див. Рисунок 18) або увійти у систему (див. Рисунок 19). При реєстрації на пошту користувача буде надіслано лист з посилання при переході за яким він активує свій акаунт. Таким чином відбувається перевірка, що дана електронна пошта є реальною та належить саме цій людині. Також при заповнені відбувається перевірка на введені правильних даних та заповненні обов'язкових полів.

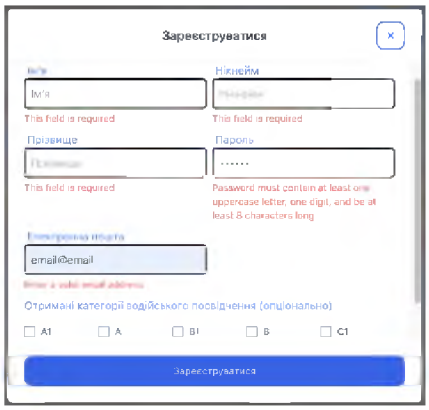

Рисунок 18. Форма реєстрації

Рисунок 19. Форма входу

За потреби користувач може скинути свій пароль. Після введення електронної пошти, на яку зареєстрований акаунт, на неї прийде електронний лист з шестизначним кодом перевірки. Його та новий пароль необхідно ввести у формі (див. Рисунок 20).

Рисунок 20. Форма зміни паролю

3.4 Заявки на курс

Заявки на курс бачить адміністратор (через спеціальний розділ) та студент (у профілі) (див. Рисунок 21). Неопрацьовані заявки студент може видаляти. Скасовані заявки позначаються червоним кольором, погоджені – синім, а створені - зеленим.

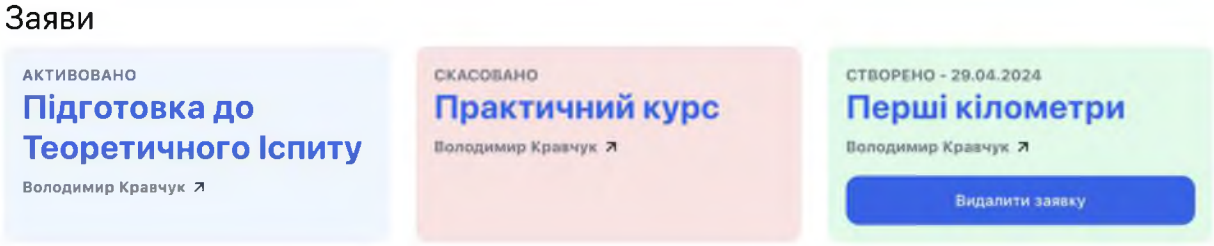

Рисунок 21. Заявки на курси у профілі студента

Адміністратор може переглядати розділ зі створеними заявками та з усіма заявками (див. Рисунок 22).

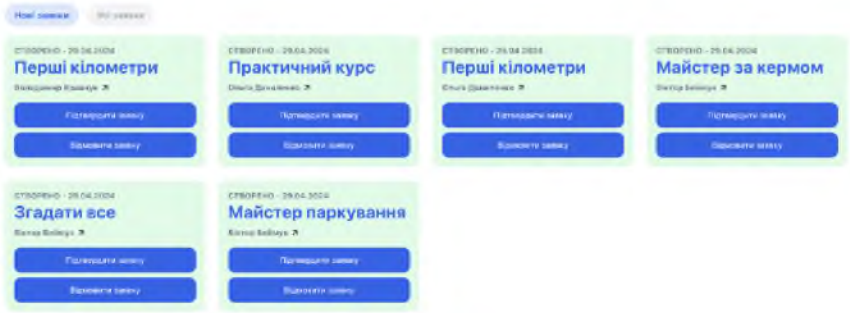

Рисунок 22. Заявки на курс у адміністратора

При натиску на ім’я студента, який створив заявку, є можливість переглянути інформацію про нього, включно з контактами та категоріями посвідчення (див. Рисунок 23).

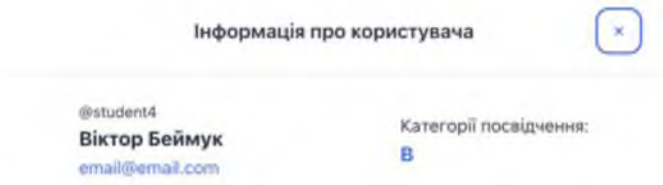

Рисунок 23. Додаткова інформація про студента, який створив заявку

У розділі з усіма заявками, окрім усіх заявок (створені, активовані та скасовані), є графік, який систематизує дані щодо кількості створених заявок по днях (див. Рисунок 24). Ця інформації є корисною для маркетингового відділу, а також при плануванні та аналізі показників.

Рисунок 24. Графік створених заявок по днях

3.5 Перегляд статистики

Адміністратор має можливість переглядати детальну статистику по курсам, працівникам та студентам через спеціальні сторінки застосунку.

При перегляді статистики по працівникам (лектори та інструктори окремо), адміністратор бачить таблицю з загальною інформацією про них (див. Рисунок 25), а при натиску на рядок – профіль працівника, включно з календарем (див. Рисунок 26) та курсами працівника з усіма статусами (див. Рисунок 27). Також архівувати працівника можливо через профіль. Для зручнішого пошуку є можливість використання пошуку, фільтрів та сортування.

Рисунок 25. Таблиця з працівниками.

Рисунок 26. Профіль працівника та календар

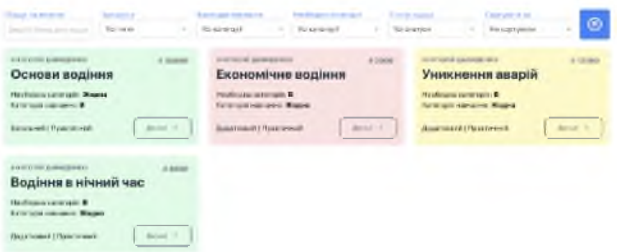

Рисунок 27. Список курсів працівника

При перегляді статистики про студентів адміністратор бачить таблицю з загальною інформацією про них (див. Рисунок 28), а при натиску на рядок – профіль студента, включно з календарем та курсами, які він проходить та пройшов (див. Рисунок 29). Для зручнішого пошуку є можливість використання пошуку, фільтрів та сортування.

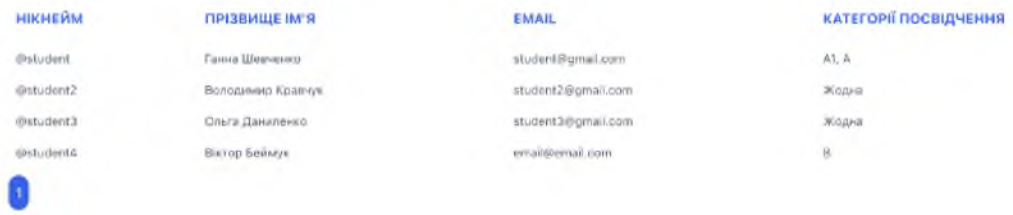

Рисунок 28. Таблиця зі студентами

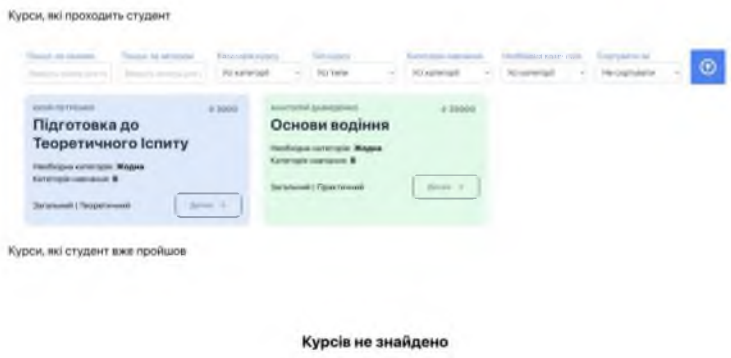

Рисунок 29. Курси студента у профілі

При перегляді статистики по курсам (практичні та теоретичні окремо), адміністратор бачить таблицю з загальною інформацією про них (див. Рисунок 30), а при натиску на рядок – детальну інформацію, включно з таблицею зі студентами, які проходять/пройшли курс (див. Рисунок 31), списком теоретичних уроків (для теоретичного курсу) з можливістю переглянути їх або їх статистику (див. Рисунок 32) та кількість практичних занять (для практичного курсу).

Рисунок 30. Таблиця з курсами

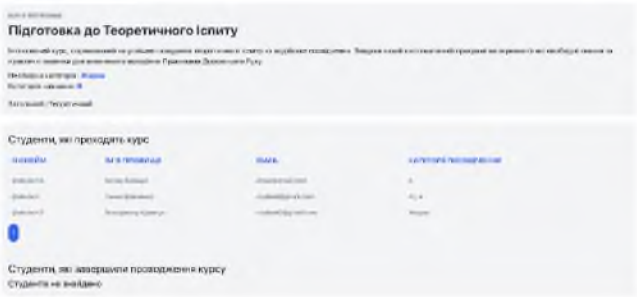

Рисунок 31. Список студентів, які проходять/пройшли курс

Рисунок 32. Список теоретичних уроків до курсу

Лектор має можливість переглядати детальну статистику своїх курсів, як загальну по курсу або заняття, так і стосовно певного студенту щодо певного курсу (які заняття він не пройшов, почав проходити та закінчив). Для наочного перегляду додано кругові діаграми.

Загальна статистика по обрану курсу полягає у перегляді таблиці зі студентами, які пройшли або проходять курс (див. Рисунок 33).

Рисунок 33. Список студентів, які проходять/пройшли курс

Загальна статистика по заняття полягає у перегляді таблиці та кругової діаграми з даними студентів, які почали проходити заняття, пройшли його або ще не починали (див. Рисунок 34).

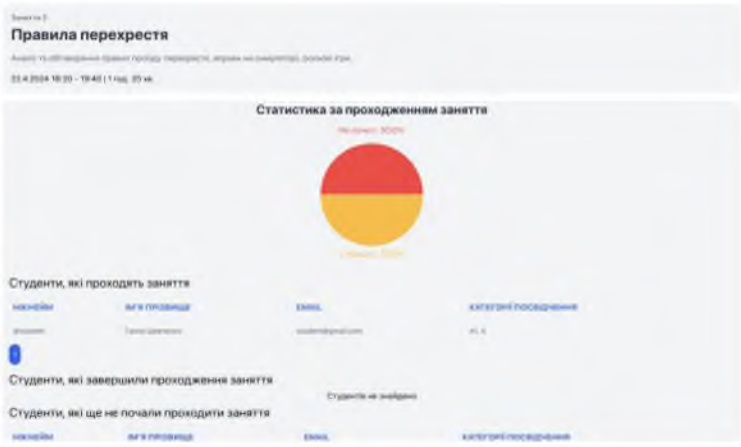

Рисунок 34. Статистика заняття

Статистика стосовно певного студенту щодо певного курсу полягає у перегляді списку (див. Рисунок 35) та кругової діаграми з заняттями (див. Рисунок 36), які він почав проходити, пройшов або ще не починав. Пройдене заняття має зелений фон, в процесі проходження – жовтий, не почате – червоним.

Рисунок 35. Статистика студента щодо занять курсу

Рисунок 36. Кругова діаграма статистики студента

При переході до профілю користувача з статистичних таблиць видно загальну інформацію та список курсів даного лектора, на які записаний та які пройшов студент.

Інструктор має можливість переглядати детальну статистику своїх курсів, як загальну по курсу, так і стосовно певного студенту щодо певного курсу (пройдені/заброньовані). Для наочного перегляду додано кругові діаграми.

Загальна статистика по обрану курсу полягає у перегляді таблиці зі студентами, які пройшли або проходять курс (див. Рисунок 37).

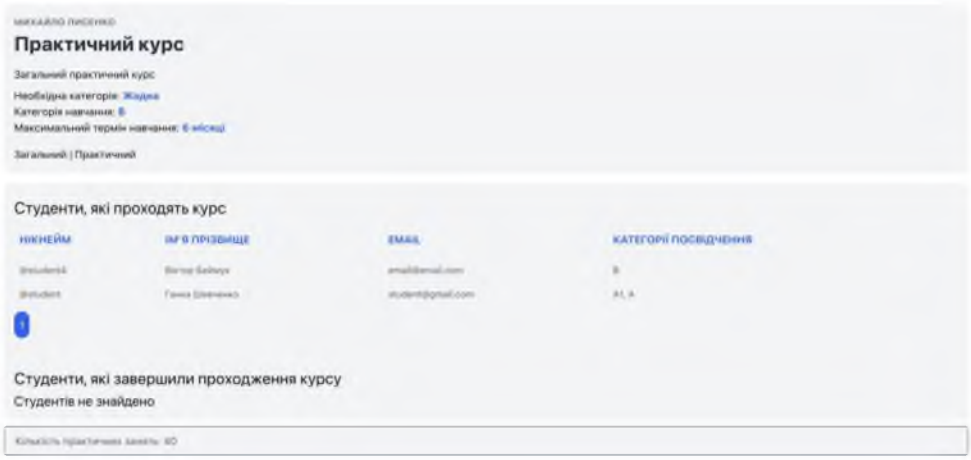

Рисунок 37. Список студентів, які проходять/пройшли курс

Статистика стосовно певного студенту щодо певного курсу полягає у перегляді списку та кругової діаграми з заняттями, які він пройшов, тільки записався або ще необхідно записатися (див. Рисунок 38).

Рисунок 38. Статистика проходження практичного курсу студентом

При переході до профілю користувача з статистичних таблиць видно загальну інформацію та список курсів даного інструктора, на які записаний та які пройшов студент.

Студент має можливість переглядати статистику свого проходження теоретичного та практичного курсу. Статистика проходження практичного курсу полягає у можливості перегляду списку заброньованих, пройдених занять, їх кількості до завершення курсу та кругової інформації з цією інформацією (див. Рисунок 39).

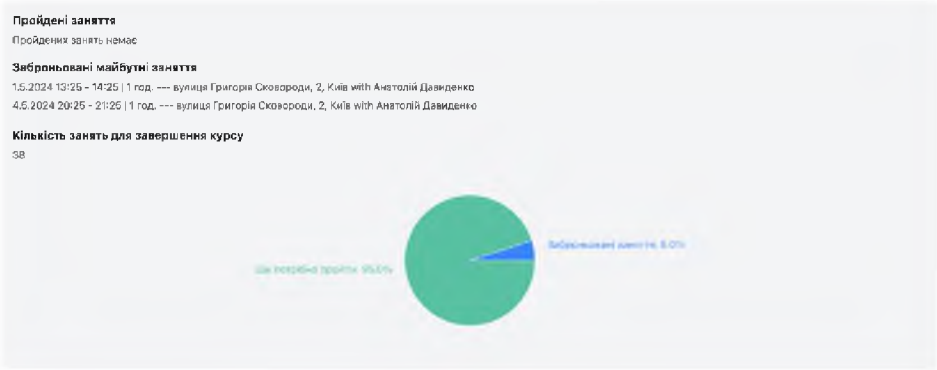

Рисунок 39. Статистика проходження практичного курсу студентом

Статистика проходження теоретичного курсу полягає у перегляді списку (див. Рисунок 40) та кругової діаграми з заняттями, які він почав проходити, пройшов або ще не починав (див. Рисунок 41). При першому відкритті заняття воно позначається як почате (жовтий колір фону), після того як час синхронного заняття пройде та студент відмітить його як пройдене фон змінюється на зелений. Перевагами цього підходу є полегшення відслідковування занять особливо у великих курсів.

Рисунок 40. Статистика студента щодо занять курсу

Рисунок 41. Кругова діаграма статистики студента

3.6 Профіль користувача

Профіль користувача складається з різних розділів залежно від його ролі. Спільними можливостями для всіх ролей є зміна паролю, загальної інформації, видалення профілю (якщо користувач є студентом або адміністратором), а також перегляд своїх даних (див. Рисунок 42).

Рисунок 42. Інформація про користувача у профілі

Студент може переглянути свої заявки на курси, а також видалити неопрацьовану заявку (див. Рисунок 43).

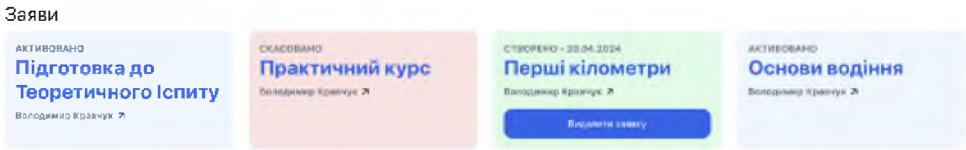

Рисунок 43. Заявки у профілі студента

Студент, лектор та інструктор можуть переглянути свій календар подій (див. Рисунок 44). У лектора відображаються його теоретичні уроки з усіх курсів, у

інструктора – слоти практичних занять, у студента – і лекційні заняття курсів на які він записаний, і заброньовані слоти для практичних занять.

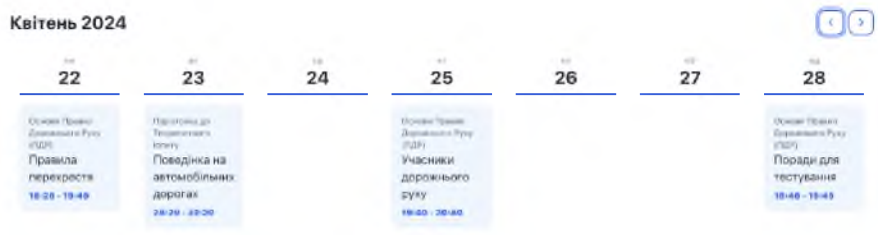

Рисунок 44. Календар у профілі

3.7 Процес навчання

Студент бачить список своїх курсів для навчання (окремо теоретичні та практичні) у спеціальному меню (див. Рисунок 45). Для зручності можна використати пошук за назвою.

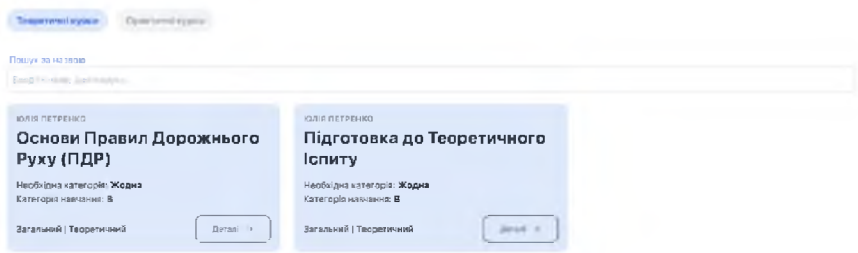

Рисунок 45. Список курсів студента

Під час навчання на практичному курсі студенту необхідно записуватися на практичні заняття через календар на сторінці навчального курсу (див. Рисунок 46). Для цього у доступних слотах інструктора є кнопка «Записатися». Також передбачено виписування з занять (через профіль) за умови, що мінімальна кількість годин до заняття для скасування ще не вичерпалася. Якщо слот заброньований – він підсвічується сірим, якщо доступний – жовтим. Для відслідковування кількості пройдених/заброньованих занять студент може переглядати статистичні дані на сторінці курсу.

Рисунок 46. Запис на практичні заняття

Під час навчання на теоретичному курсі студент необхідно опрацьовувати матеріал кожного заняття. Ці матеріали знаходяться на сторінці заняття (див. Рисунок 47).

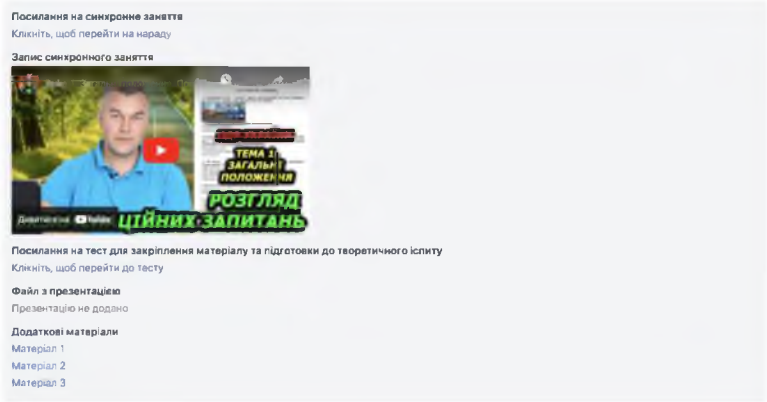

Рисунок 47. Сторінка заняття теоретичного курсу

За бажанням студент може відповісти на контрольні запитання до теми, які додав викладач (див. Рисунок 48). Відслідковувати прогрес курсу студент може через головну сторінку навчального курсу.

Рисунок 48. Контрольні запитання до заняття

3.8 Створення та управління курсами

Для управління курсами створено окреме меню вебзастосунку (див. Рисунок 49), яке доступне лише користувачам з роллю лектор або інструктор. Саме з нього відбувається редагування, опублікування та перегляд курсів, перехід до статистики та архівування/розархівування (лише для практичних курсів). У цьому меню знаходяться курси користувача з усіма статусами. Для зручнішого перегляду було додано фільтри та пошук.

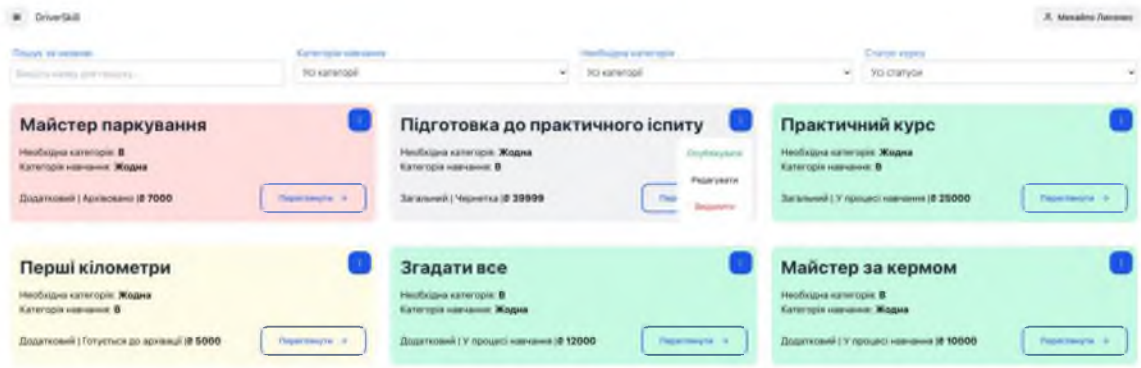

Рисунок 49. Сторінка для управління курсами

Створення та редагування курсів відбувається через модальні меню. Для теоретичного (див. Рисунок 50) та практичного (див. Рисунок 51) курсу вони відрізняються деякими полями введення. Додавання та редагування теоретичних занять також відбувається через модальні форми.

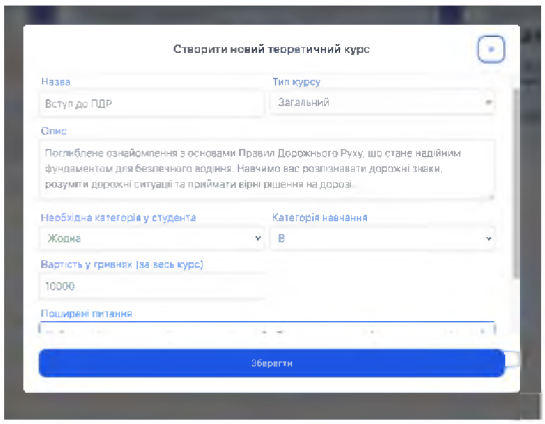

Рисунок 50. Форма створення теоретичного курсу

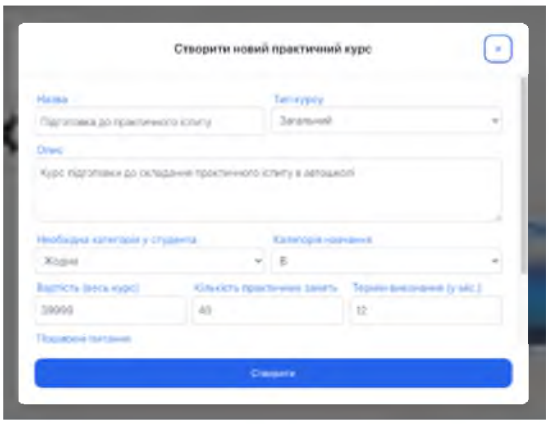

Рисунок 51. Форма створення практичного курс

ВИСНОВКИ

Під час виконання курсової роботи було створено вебзастосунок для систематизації навчального процесу в автошколах.

Було розглянуто та проаналізовано інші LMS (learning management system), що надають можливості організації дистанційного навчання та виокремлені основні функції, які необхідно реалізувати у застосунку.

На основі аналізу існуючих LMS-систем було сформовано постановку задачі та обрано інструментарій для розробки вебзастосунку, розроблено схему бази даних, обрано необхідний архітектурний шаблон. Архітектурним шаблоном даного застосунку було обрано клієнт-серверний шаблон. Під час розробки серверної частини застосунку була використана мова програмування Java та фреймворк Spring Boot. Для розробки клієнтської частини застосунку була використана мова JavaScript та бібліотека React.

Створений застосунок має ряд переваг, оскільки його можна використовувати як єдине джерело інформації про автошколу, адже у ньому наявні не лише функції для створення курсів, їх управління та навчання на них, введення обліку даних, можливість отримання статистичної інформації, перегляд прогресу навчання, але й загальна інформація про школу, її актуальні набори на курси тощо.

Також застосунок має хороші перспективи на майбутнє, оскільки можна зручно та швидко поповнити його новими функціоналами, наприклад такими як: експорт статистичних даних у файл, персональні та групові чати між викладачами та студентами всередині застосунку, можливість проведення окремих заходів, лекцій, майстер класів і т.п.

ДЖЕРЕЛА

1. Що таке Moodle. URL: https://moodle.org/mod/page/view.php?id=8174
2. Офіційний сайт KUTS. URL: https://kuts.ua/lms/
3. Офіційна сторінка платформи «Microsoft Teams». URL: https://www.microsoft.com/fi-fi/microsoft-teams/group-chat-software?rtc=1
4. All Major Software Architecture Patterns Explained. URL: https://medium.com/@learnwithwhiteboard_digest/all-major-software-architecture-patterns-explained-8ed6b50a16e3
5. Spring Boot Architecture. URL: https://www.javatpoint.com/spring-boot-architecture
6. Saharawat V. Architecture of Spring Boot: Examples, Pattern, Layered, Controller Layer. URL: https://pwskills.com/blog/architecture-of-spring-boot-examples-pattern-layered-controller-layer/
7. What are HTTP methods? URL: https://blog.postman.com/what-are-http-methods/
8. Офіційна документація React. URL: https://react.dev/
9. Advantages And Disadvantages of React JS. URL: https://medium.com/@reactmasters.in/advantages-and-disadvantages-of-react-js-e6c80b25763b
10. Офіційна документація Tailwind CSS. URL: https://tailwindcss.com/
11. Prajeet Kumar Thakur. Tailwind CSS: Pros And Cons. URL: https://medium.com/readytowork-org/tailwind-css-pros-and-cons-f1a8fdb1fb47
12. Офіційна документація Spring Boot. URL: https://spring.io/projects/spring-boot
13. Apache Maven. URL: https://www.geeksforgeeks.org/apache-maven/
14. Офіційна документація Spring Data JPA. URL: https://spring.io/projects/spring-data-jpa

15. Fadatare R. Difference between Spring Data JPA and JDBC. URL: https://www.javaguides.net/2023/12/what-is-difference-between-spring-data-jpa-and-jdbc.html
16. Дудка М. Spring JDBC проти Spring Data JPA. URL: https://dou.ua/forums/topic/46082/
17. Офіційна документація MySQL. URL: https://dev.mysql.com/doc/
18. Офіційна документація Cloudinary. URL: https://cloudinary.com/documentation/java_integration
19. Офіційна документація Swagger. URL: https://swagger.io/docs/
20. Organize and automate API requests in Postman Collections. URL: https://learning.postman.com/docs/collections/collections-overview/
21. Офіційна документація Spring Security. URL: https://docs.spring.io/spring-security/reference/index.html
22. Ruck D. What Are Refresh Token And How Can They Boost Your Security? URL: https://stateful.com/blog/refresh-tokens-security
23. Офіційна документація Spring Security BCryptPasswordEncoder. URL: https://docs.spring.io/spring-security/reference/features/authentication/password-storage.html#authentication-password-storage-bcrypt
24. What is bcrypt and how does it work? URL: https://medium.com/@valevpn/what-is-bcrypt-and-how-does-it-work-bef43ee8762d