

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»

Кафедра інженерії програмного забезпечення

КУРСОВА РОБОТА
(ПОЯСНЮВАЛЬНА ЗАПИСКА)

з дисципліни: «Бази даних»

на тему:

«База даних онлайн енциклопедії аніме й манги»

студента II курсу групи ВТ-21-1
спеціальності 121 «Інженерія програмного
забезпечення»

Джуса Ігоря Олександровича
(прізвище, ім'я та по-батькові)

Керівник доцент каф. КН, Кравченко С. М.

Дата захисту: " 31 " травня 2023 р.

Національна шкала _____

Кількість балів: _____

Оцінка: ECTS _____

Члени комісії

_____	<u>І. І. Сугоняк</u>
(підпис)	(прізвище та ініціали)
_____	<u>О. В. Коротун</u>
(підпис)	(прізвище та ініціали)
_____	<u>С. М. Кравченко</u>
(підпис)	(прізвище та ініціали)
_____	<u>О.В. Чижмотря</u>
(підпис)	(прізвище та ініціали)

ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»

Факультет інформаційно-комп'ютерних технологій

Кафедра Інженерії програмного забезпечення

Освітній рівень: бакалавр

Спеціальність 121 «Інженерія програмного забезпечення»

«ЗАТВЕРДЖУЮ»

В. о. зав. кафедри

_____ А.В. Морозов

“ ____ ” _____ 2023 р.

ЗАВДАННЯ

НА КУРСОВУ РОБОТУ СТУДЕНТУ

Джусу Ігорю Олександровичу

1. Тема роботи: «База даних онлайн енциклопедії аніме й манги», керівник курсової роботи: ст. викл., Кравченко С. М.
2. Строк подання студентом: “29” травня 2023р.
3. Вихідні дані до роботи: розробити базу даних для онлайн енциклопедії аніме й манги.
4. Зміст розрахунково-пояснювальної записки(перелік питань. Які підлягають розробці)
 1. Постановка завдання
 2. Аналіз аналогічних розробок
 3. Алгоритми роботи програми
 4. Опис роботи програми
 5. Програмне дослідження
5. Перелік графічного матеріалу:
 1. Посилання на репозиторій: <https://github.com/KinDofHell/AnimeArchive-pet-project-.git>
 2. Презентація до КП
6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посади консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1, 2, 3, 4	Кравченко С.М., доцент каф. КН		

7. Дата видачі завдання “15” березня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів курсової роботи	Строк виконання етапів роботи	Примітки
1	Постановка задачі	01.03.2023	Виконано
2	Пошук, огляд та аналіз аналогічних розробок	13.03.2023	Виконано
3	Формулювання технічного завдання	14.04.2023	Виконано
4	Опрацювання літературних джерел	20.04.2023	Виконано
5	Проектування структури	25.04.2023	Виконано
6	Написання програмного коду	04.04.2023	Виконано
7	Налагодження	12.04.2023	Виконано
8	Написання пояснювальної записки	14.05.2023	Виконано

Студент

(підпис)

І. О. Джус

(прізвище та ініціали)

Керівник роботи

(підпис)

доцент каф. КН, Кравченко С. М.

(прізвище та ініціали)

Реферат

Курсова робота присвячена розробці бази даних для онлайн-енциклопедії аніме й манги.

Пояснювальна записка до курсової роботи на тему «розробка бази даних для онлайн енциклопедії аніме й манги» складається з вступу, чотирьох розділів, висновків, а також списку використаних джерел, де у свою чергу містяться посилання на ресурси, які надають безкоштовні послуги різних типів, і додатків.

Основна частина представлена 39 сторінками друкованого тексту.

Пояснювальна записка(звіт) у повному обсязі складає 62 сторінки, з яких 18 сторінок – додатки з лістингом програми.

Перший розділ було присвячено аналізу поставлених умов, дослідженню та пошуку аналогів і ознайомленню з додатковою інформацією.

Другий розділ увібрав у себе планування та проєктування програмної системи, представлення схем та діаграм.

Третій розділ слугував для дослідження роботи онлайн-енциклопедії, а також представлення та опису методів та алгоритмів.

Четвертий розділ розкрив адміністрування бази даних та безпекові моменти, які дозволяють запобігти некоректним діям.

Висновок містить змістовний, об'єктивний та лаконічний розгляд виконаної роботи, врахування отриманих та закріплених навичок, зазначення користі від тієї чи іншої частини роботи.

Ключові слова: NodeJS, MongoDB, Express, React, БД, онлайн-енциклопедія, веб-сайт, адміністратор, публікації, користувачі.

					Житомирська політехніка.23.121.08.000 - ПЗ		
Змн.	Арк.	№ докум.	Підпис	Дата	Розробка бази даних онлайн енциклопедії аніме та манг		
Розроб.		Джус І. О.					
Перевір.							
Керівник		Кравченко С. М.					
Н. контр.							
Зав. каф.					Літ. Арк. Аркушів		
						4	63
					ФІКТ Гр. ВТ-21-1[1]		

Зміст

ВСТУП	6
РОЗДІЛ 1. АНАЛІЗ ПРОБЛЕМАТИКИ, ЗАСОБІВ ТА МЕТОДІВ ВИРІШЕННЯ ЗАДАЧІ	7
1.1. Аналіз умов задачі, методи реалізації.....	7
1.2 Аналіз існуючих онлайн-енциклопедій за тематикою курсової роботи	8
1.3 Обґрунтування вибору засобів реалізації.....	10
РОЗДІЛ 2. ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	12
2.1. Проектування загального алгоритму роботи програми.....	12
2.2. Проектування структури бази даних за напрямком курсової роботи	14
РОЗДІЛ 3. ОПИС РОБОТИ З ПРОГРАМНИМ ДОДАТКОМ ТА ЙОГО ТЕСТУВАННЯ	17
3.1. Опис роботи з програмним додатком	17
3.2. Реалізація операцій обробки даних в БД за напрямком курсової роботи.	24
РОЗДІЛ 4. АДМІНІСТРУВАННЯ БАЗ ДАНИХ ТА БЕЗПЕКА	32
4.1. Розробка заходів захисту інформації в БД.....	32
ВИСНОВКИ	38
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	39
ДОДАТКИ	40
Технічне завдання	41

		Джус І. О.			Житомирська політехніка.23.121.08.000 - ПЗ	Арк.
		Кравченко С. М.				5
Змн.	Арк.	№ докум.	Підпис	Дата		

ВСТУП

Актуальність теми.

Актуальність роботи полягає в оптимізації взаємодії з іноземними культурами, а саме з японськими мангами та аніме, які користуються популярністю серед молоді. Продумана та адаптована під сучасні потреби база даних дозволить зберігати великі обсяги інформації про даний вид творчості та передбачить зв'язки для швидкого доступу між різними типами та видами інформації, яка в результаті предстане єдиним продуктом.

Метою курсової роботи є повноцінне застосування засобів для розробки бази даних у ході виконання курсового проекту на тему «База даних онлайн енциклопедії аніме й манги».

Об'єктом дослідження є можливість реалізації бази даних MongoDB з використанням платформи Node.js, його фреймворку Express та оформлення клієнтської частини за допомогою фреймворка React.

Предметом дослідження є опрацювання та вивчення способів реалізації курсового проекту за допомогою MongoDB, його застосування.

		Джус І. О.			Житомирська політехніка.23.121.08.000 - ПЗ	Арк.
		Кравченко С. М.				
Змн.	Арк.	№ докум.	Підпис	Дата		6

РОЗДІЛ 1. АНАЛІЗ ПРОБЛЕМАТИКИ, ЗАСОБІВ ТА МЕТОДІВ ВИРІШЕННЯ ЗАДАЧІ

1.1. Аналіз умов задачі, методи реалізації

Після детального розбіру умов та аналізу теми загалом було визначено певні аспекти, які вказують на подальші методи та способи розробки. Для реалізації бази даних онлайн енциклопедії аніме та манги був обраний варіант з розробкою веб-сайту за допомогою платформи Node.js та фреймворку Express, які забезпечують зручність та швидкість. У ролі бази даних буде виступати MongoDB.

Після визначення засобів розробки стало можливим передбачити найголовніший та необхідний функціонал курсово проекту:

1. Перегляд усього асортименту аніме та манг.
2. Сортування за новизною, популярністю та категоріями.
3. Перегляд інформації про конкретний продукт.
4. Формування власного списку аніме й манг шляхом відмічання їх.
5. Швидкий доступ до нещодавно доданих аніме та манг.
6. Перегляд тематичних новин.
7. Перегляд персонажів, зв'язаних зі своїм проектом.
8. Адміністрування(додавання, редагування та видалення)
9. Авторизація та реєстрація, ролі користувачів.

Для виконання поставлених задач буде використане середовище для розробки «VS Code» та програма для адміністрування бази даних «MongoDBCompass».

		Джус І. О.			Житомирська політехніка.23.121.08.000 - ПЗ	Арк.
		Кравченко С. М.				7
Змн.	Арк.	№ докум.	Підпис	Дата		

1.2 Аналіз існуючих онлайн-енциклопедій за тематикою курсової роботи

Після ретельного та об'єктивного пошуку подібних онлайн-енциклопедій було знайдено декілька популярних веб-сайтів та досліджено їхній функціонал та інші можливості.

1. Fandom(рис. 1.2.1)

Дана онлайн-енциклопедія дозволяє переглядати інформацію про незлічену кількість медіа-проектів, персонажів пов'язаних з ними та додаткову інформацію, яка може зустрічатися під час вивчення матеріалів.

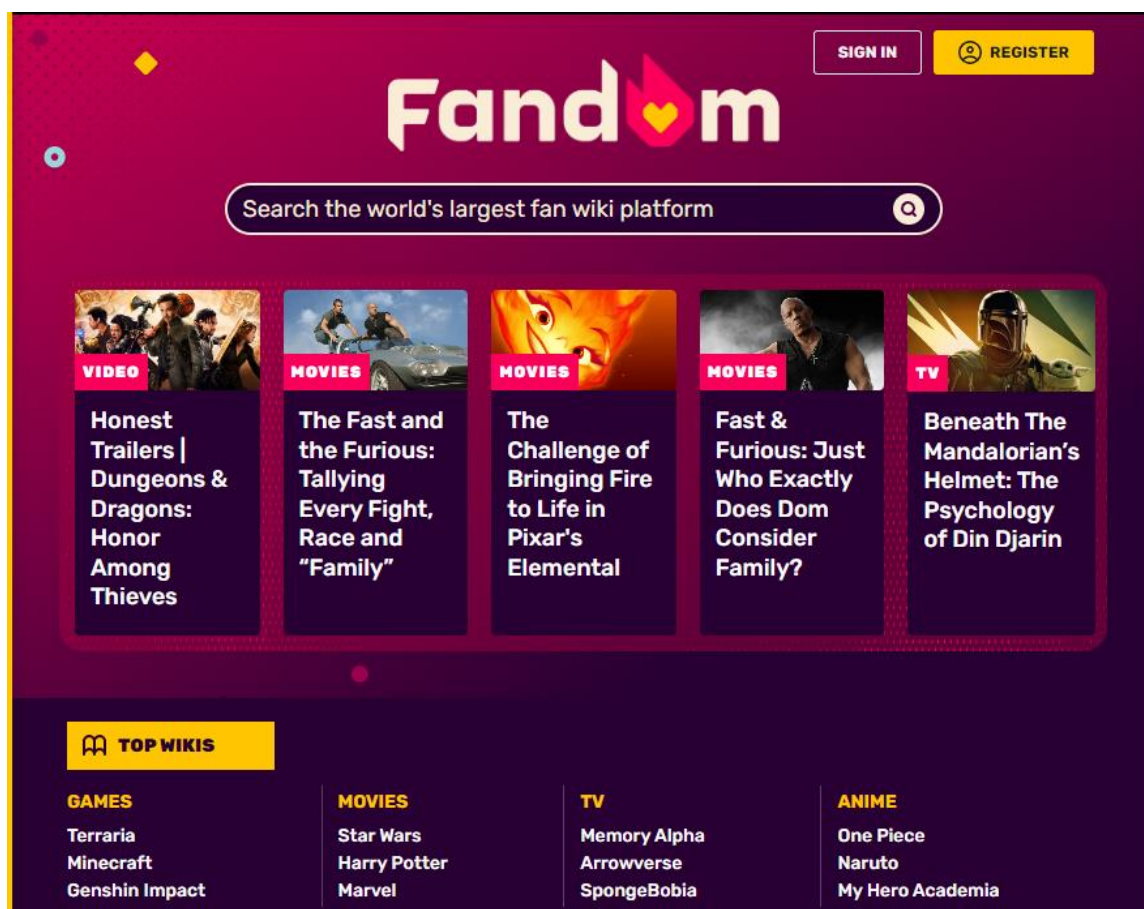


Рис. 1.2.1. Fandom

2. Wikipedia(рис.1.2.2)

Wikipedia є однією з найбільших онлайн-енциклопедій на просторах Інтернету, вона містить неймовірну кількість інформації про різні медіа-проекти, наукові терміни та кожен річ у нашому житті. Попри поважний

		Джус І. О.			Житомирська політехніка.23.121.08.000 - ПЗ	Арк.
		Кравченко С. М.				8
Змн.	Арк.	№ докум.	Підпис	Дата		

«вік» даної платформи — вона все ще допомагає мільярдам людей знайти потрібну інформацію.



Рис.1.2.2. Wikipedia

Після ознайомлення з аналогами можна перейти до прицільного порівняння їх особливостей з особливостями, які будуть реалізовані в курсовій роботі. Таким чином варто відзначити, що великою проблемою наведених онлайн-енциклопедій є застарілий та важкий для прочитання дизайн та стиль подання інформації, а також занадто великий обсяг інформації, яка, зазвичай, виявляється непотрібною. Також з досвіду користувача можна зазначити, що відносно реєстрації на сайті користувач не отримує ніяких зручних привілеїв.

Тож з врахуванням проаналізованих недоліків у курсовому проекті буде передбачено розумний розподіл та вибір інформації, її зрозуміле та оптимальне представлення користувачу, виділення важливої інформації за допомогою окремих полів у базі даних. Ще однією перевагою буде можливість зареєстрованим користувачам формувати власні списки з вибраних дописів на сайті для зберігання важливої для себе інформації.

		Джус І. О.			Житомирська політехніка.23.121.08.000 - ПЗ	Арк.
		Кравченко С. М.				9
Змн.	Арк.	№ докум.	Підпис	Дата		

1.3 Обґрунтування вибору засобів реалізації

На щастя, у світі існує багато технологій для розробки, використання та підтримки бази даних, але найефективнішим чином це відбуватиметься лише за умови раціонального вибору. У ході даної курсової роботи він постав між реляційною(MS SQL Server) та нереляційною(MongoDB) СУБД, через що з'явилася потреба у детальному аналізі цих двох технологій, який наведено у таблиці 1.

Таблиця 1

Аналіз СУБД

Особливості	MS SQL Server	MongoDB
Тип даних	Реляційна база даних	Документ-орієнтована база даних
Мова запитів	Transact-SQL (T-SQL)	MongoDB Query Language (MQL)
Схема	Суворі схема даних	Гнучка схема даних
Масштабованість	Вертикальна і горизонтальна масштабованість	Горизонтальна масштабованість
Підтримка ACID	Повна підтримка	Зазвичай підтримується в обмеженій формі
Реплікація	Підтримується	Підтримується
Індексування	Широкий набір індексів	Індексування за ключами та текстові індексування
Масштабованість читань	Висока масштабованість читань	Висока масштабованість читань

Масштабованість записів	Обмежена масштабованість записів	Висока масштабованість записів
Підтримка JSON	Підтримується з MS SQL Server 2016	Вбудована підтримка JSON
Геопросторові дані	Підтримується з допомогою розширення	Вбудована підтримка геопросторових запитів
Запити JOIN	Підтримується	Підтримується в обмеженій формі
Застосунки	Підходить для традиційних веб-застосунків	Підходить для складних, масштабованих застосунків
Індексування	Широкий набір індексів	Індексування за ключами та текстове індексування

Таким чином було визначено, що для поточного курсового проекту найкраще підійде MongoDB через свою масштабованість, швидкість та відносно легку взаємодію з веб-проєктами.

Висновок. Був проведений детальний аналіз поставленої умови для визначення обсягу роботи та складності реалізації. Окремо узгодилися кожні обов'язкові та додаткові можливості онлайн-енциклопедії, шляхом дослідження обрано оптимальне середовище та засоби розробки. Проведено огляд аналогів, виконано їх порівняння та обдумано декілька побачених ідей для реалізації.

		Джус І. О.			Житомирська політехніка.23.121.08.000 - ПЗ	Арк.
		Кравченко С. М.				11
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 2. ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1. Проектування загального алгоритму роботи програми

Робота в даному розділі є однією з найголовніших етапів курсової роботи, тому що вона полягає у реалізації отриманої після аналізу інформації технічним чином. Корисним на цьому етапі є побудова та представлення схем та блок-схем, які б пояснювали принципи роботи методів, або програмного додатку загалом. Також повинні бути описані основні можливості та функціонал.

Для повноцінного та всеосяжного використання розробленої та побудованої бази даних онлайн-енциклопедія аніме та манги передбачатиме ролі розподілу можливостей між користувачами додатку, а також загальні дії. Детальніше наведено у переліку функціоналу нижче:

Будь-який користувач може:

1. Переглядати сторінки із завантаженим контентом;
2. Читати окремі публікації;
3. Переглядати підбірки з нещодавно доданого контенту на сайті, а також популярного;

Авторизований користувач отримує можливість формувати власні списки з публікацій на сайті для ведення статистики, збирання «колекції» та інших цілей.

Користувачі із визначеними ролями(менеджери тематичного контенту, новин) здатні:

1. Редагувати публікації;
2. Видаляти публікації;
3. Додавати нові публікації;

У свою чергу адміністратор, окрім усіх вищезазначених можливостей має змогу додавати нових менеджерів та видаляти існуючих, а також користуватися

		Джус І. О.			Житомирська політехніка.23.121.08.000 - ПЗ	Арк.
		Кравченко С. М.				12
Змн.	Арк.	№ докум.	Підпис	Дата		

сторінкою з інформацією щодо кількості завантаженого контенту на сайті. Отже перш ніж детально розібрати проектування бази даних, було б доцільно представити блок-схему, яка описує загальний принцип роботи програмного додатку(рис. 2.1.1).

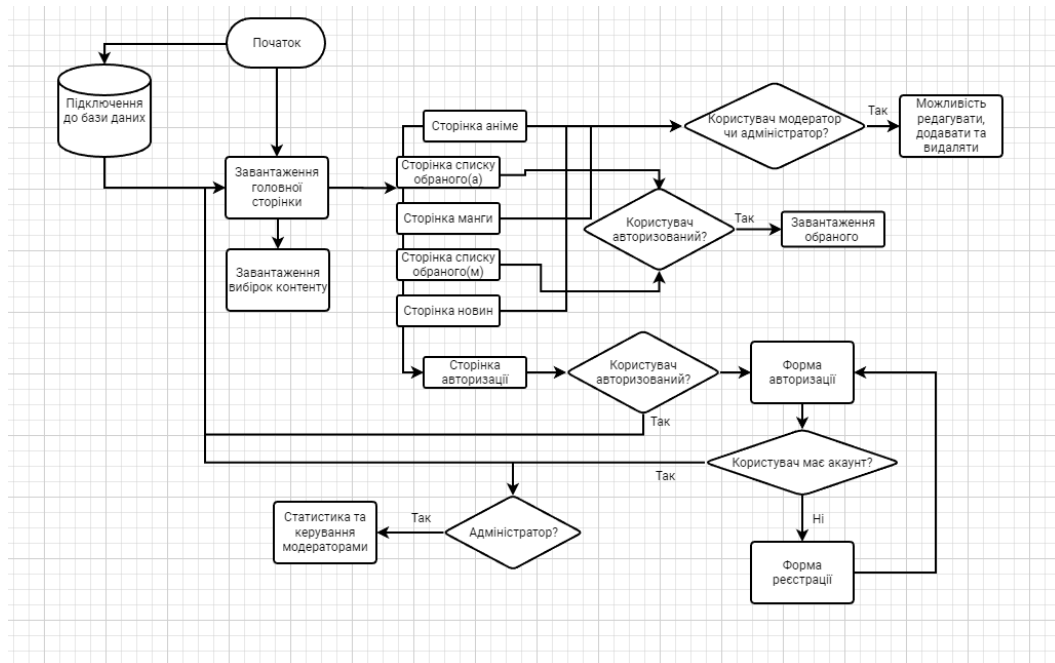


Рис. 2.1.1. Принцип роботи веб-сайту

Алгоритми реєстрації і авторизації виглядають наступним чином(рис. 2.1.2).

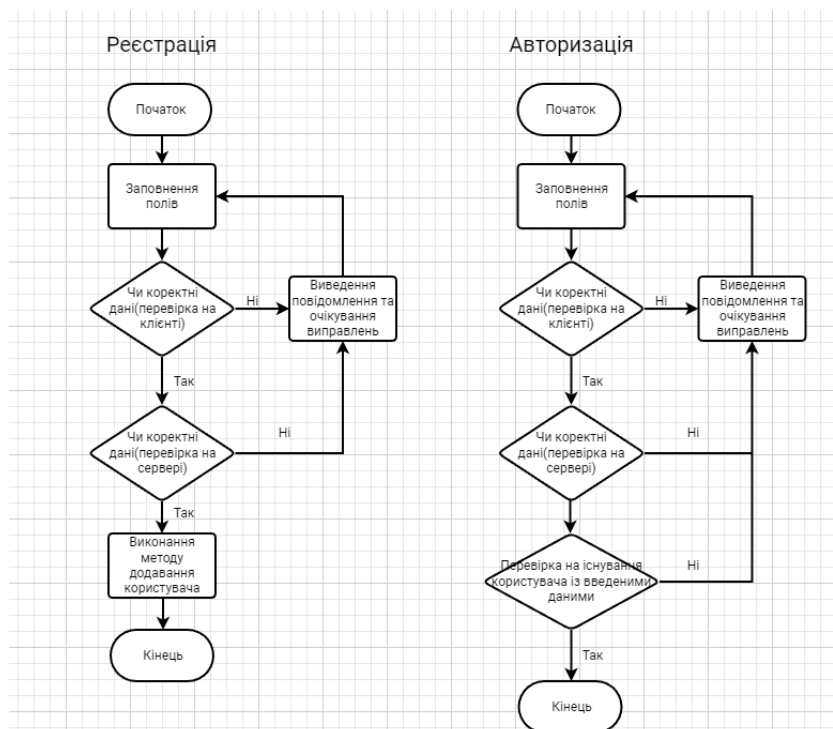


Рис. 2.1.2. Алгоритм авторизації та реєстрації

2.2. Проектування структури бази даних за напрямком курсової роботи

Важливою та фундаментальною складовою курсової роботи є база даних, тому варто навести таблиці та схеми, які показують її конструкцію та будову. Логічно було б почати із діаграми зв'язків колекцій в базі даних (рис. 2.2.1)

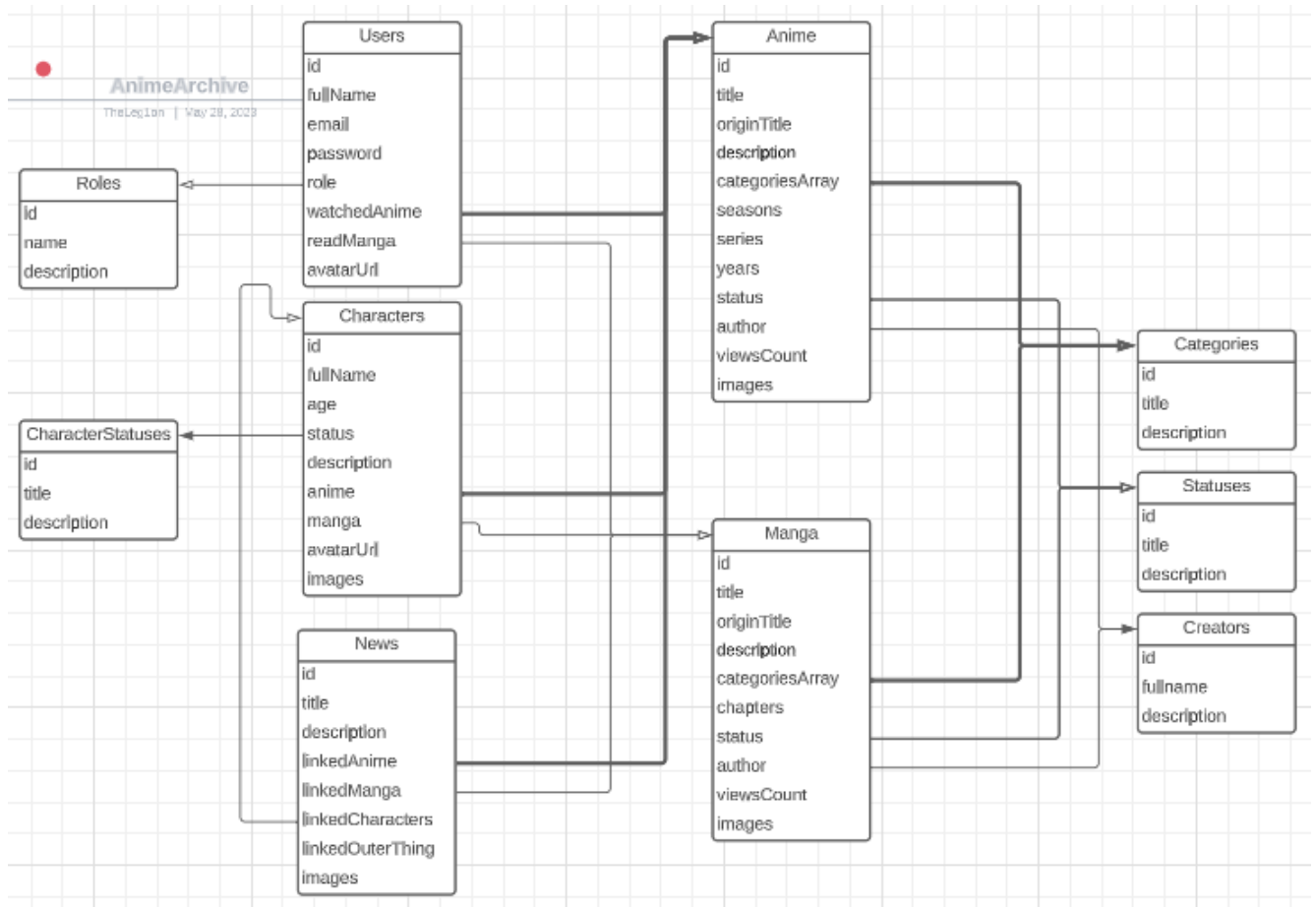


Рис. 2.2.1. Діаграма зв'язків між колекціями у MongoDB

Якщо говорити про список всіх колекцій, задіяних у даній курсовій роботі, то виглядатиме він наступним чином(рис. 2.2.2):

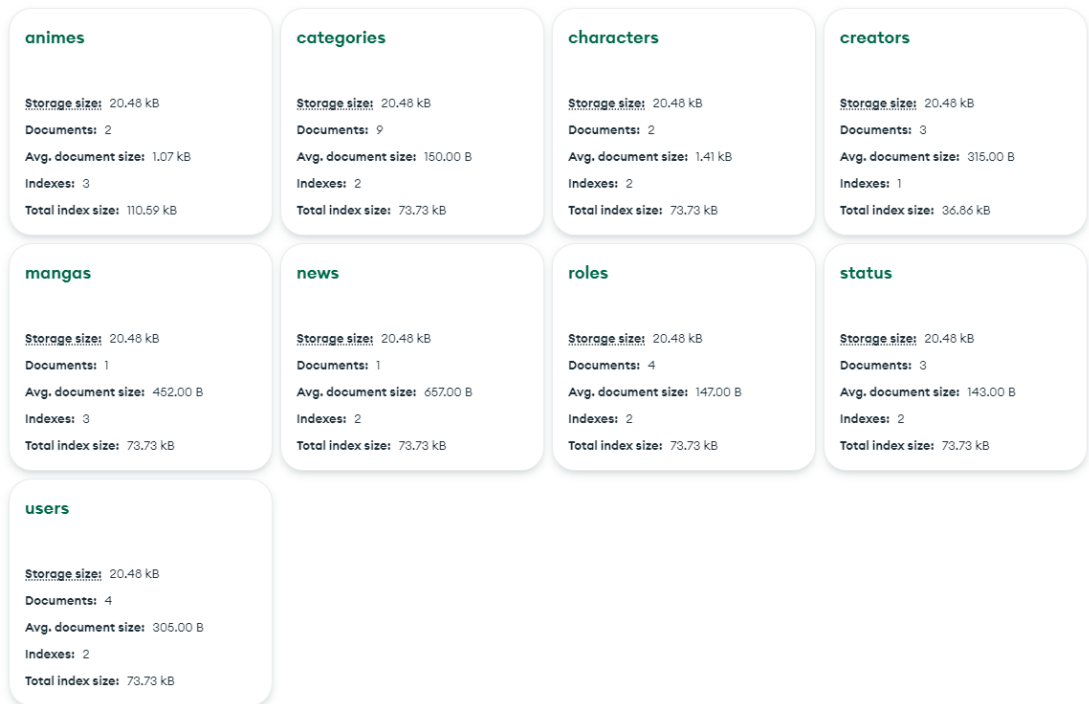


Рис. 2.2.2. Список колекцій у MongoDB
Таблиці представлення структури деяких колекцій:

Таблиця 2

Колекція animes

Назва поля	Тип
id	ObjectId
title	String
originTitle	String
description	String
categoriesArray	[ObjectId]
seasons	Number
series	Number
years	[String]
status	ObjectId
author	ObjectId
characters	[ObjectId]
viewsCount	Number
images	[String]

Таблиця 3

Колекція categories

Назва поля	Тип
id	ObjectId
title	String
description	String

Колекція characters

Назва поля	Тип
id	ObjectId
fullName	String
age	String
sex	String
race	String
status	String
partnersArray	[ObjectId]
appearance	String
personality	String
viewsCount	Number
images	[String]

Висновок. У даному розділі було спроектовано та розроблено всі основні та допоміжні алгоритми і методи, що формують або підтримують роботу онлайн-енциклопедії. Наведені: структурна діаграма, таблиці конструкції колекцій та блок-схеми алгоритмів деяких абстрактних частин додатку, а також його загалом.

РОЗДІЛ 3. ОПИС РОБОТИ З ПРОГРАМНИМ ДОДАТКОМ ТА ЙОГО ТЕСТУВАННЯ

3.1. Опис роботи з програмним додатком

Онлайн-енциклопедія аніме та манг розміщена за адресою <https://anime-archive-pet-project.vercel.app/>, після переходу за якою користувач опиняється на головній сторінці(рис. 3.1.1). Головна сторінка дає змогу будь-якому користувачу(мається на увазі авторизованому чи ні) побачити останні з нещодавно доданих аніме та манг, а також останні новини та найпопулярнішого(серед користувачів веб-сайту) персонажа.

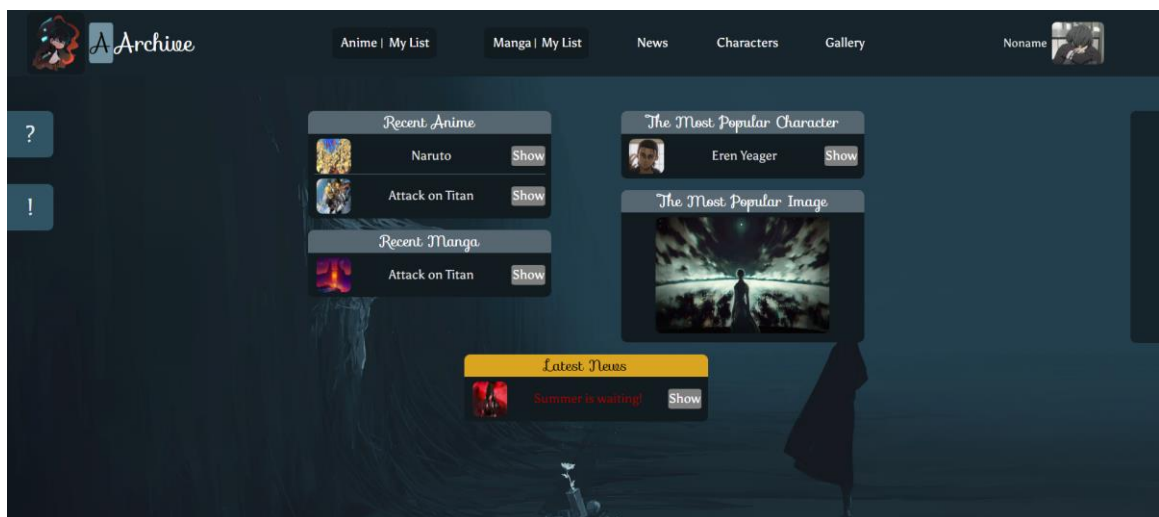


Рис. 3.1.1. Головна сторінка

Сторінка списку усіх наявних на сайті аніме(рис. 3.1.2) представляє набір контейнерів з короткою інформацією про публікацію, а також можливістю переходу на сторінку «продукту»(рис. 3.1.3).

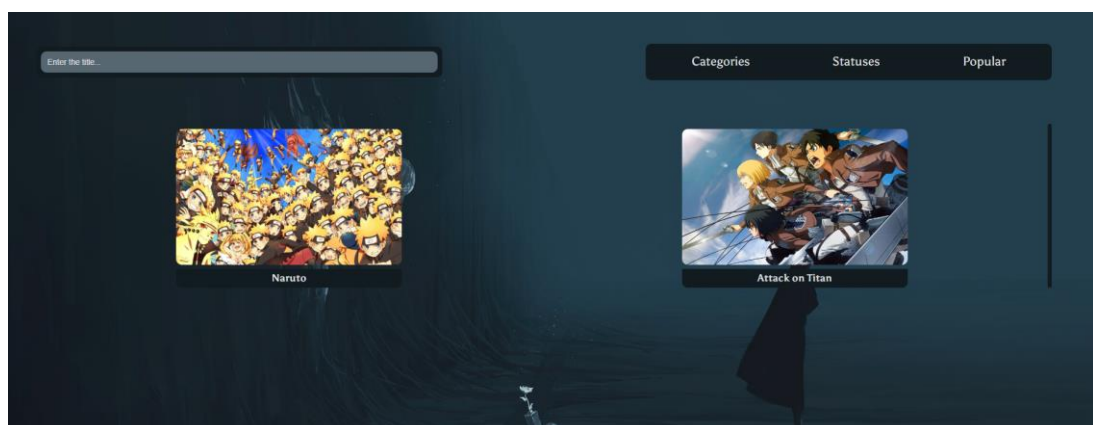


Рис. 3.1.2. Сторінка з усіма аніме

		Джус І. О.			Житомирська політехніка.23.121.08.000 - ПЗ	Арк.
		Кравченко С. М.				17
Змн.	Арк.	№ докум.	Підпис	Дата		

Сторінка конкретного(-ї) аніме/манги/новини є представленням детальної інформації та пов'язаних з нею сутностей(рис. 3.1.3). Тут відображаються пов'язані персонажі, категорії та автор з інших колекцій.

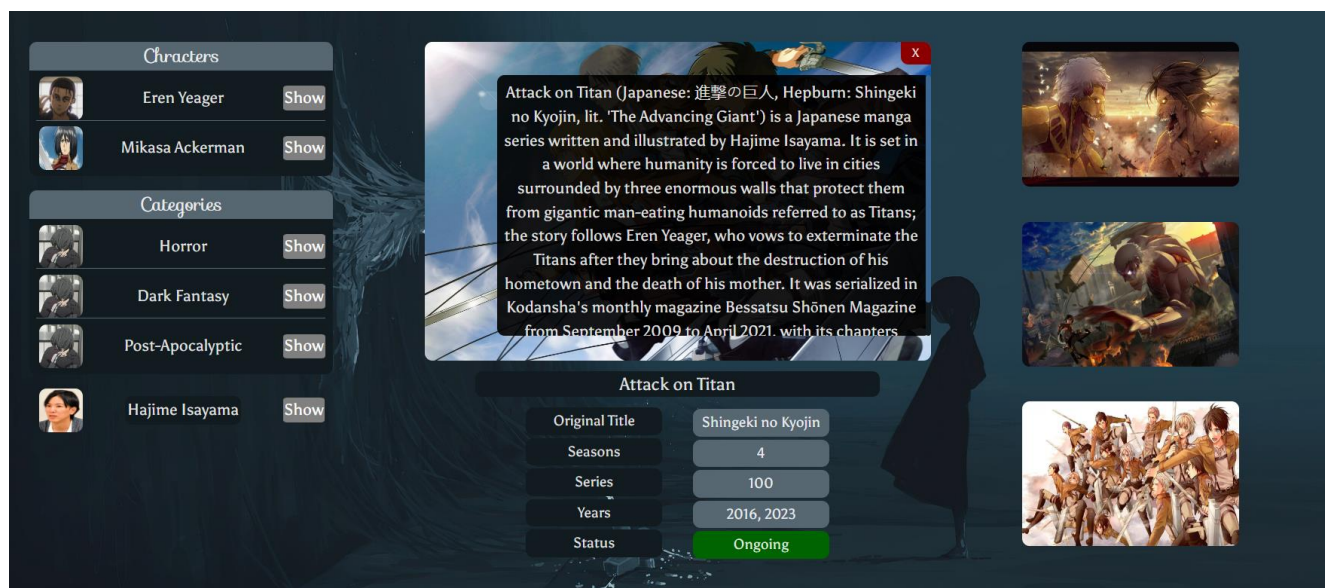


Рис. 3.1.3. Сторінка конкретної публікації

Якщо говорити про новини, то зв'язків з іншими колекціями тут передбачено більше(а саме пов'язані аніме, манги та персонажі)(рис. 3.1.4).

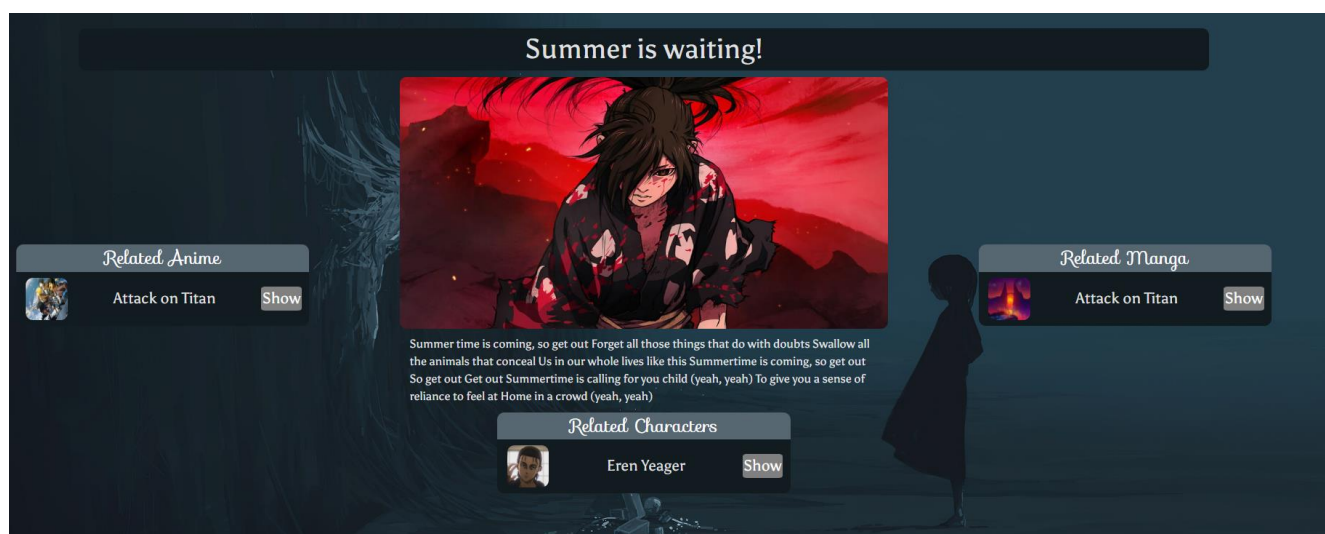


Рис. 3.1.4. Сторінка новини

Сторінка персонажа виглядає наступним чином(рис. 3.1.5):

		Джус І. О.			Житомирська політехніка.23.121.08.000 - ПЗ	Арк.
		Кравченко С. М.				18
Змн.	Арк.	№ докум.	Підпис	Дата		

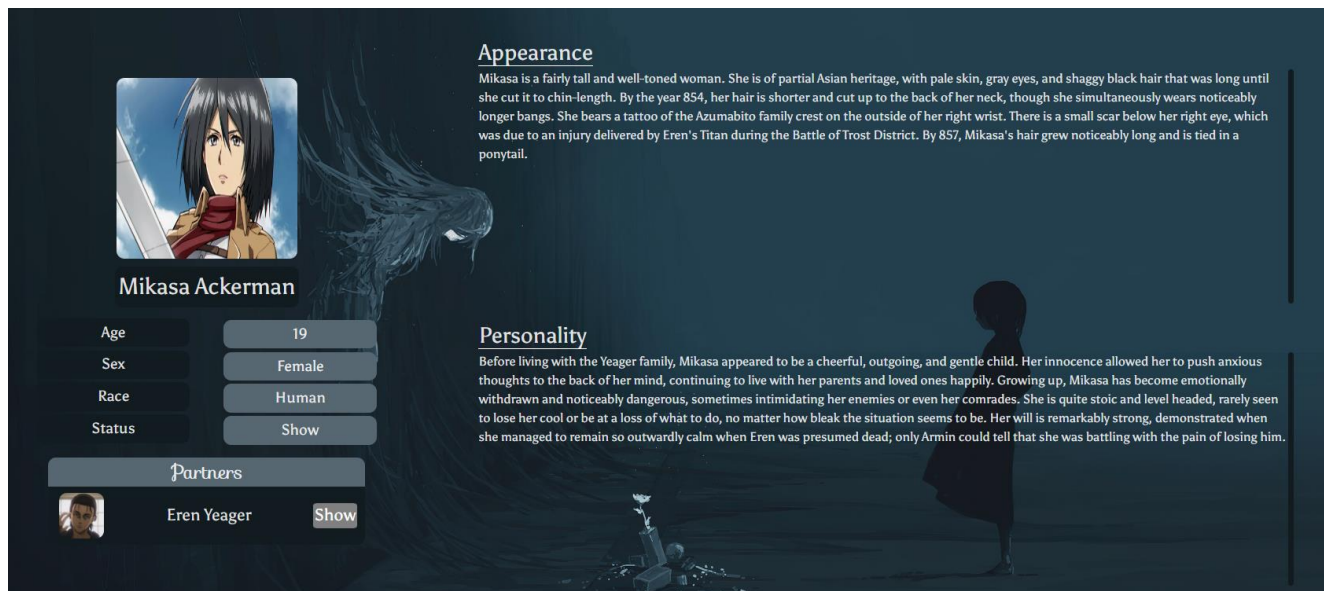


Рис. 3.1.5. Сторінка персонажа

Тепер перейдемо до використання веб-сайту як авторизований користувач та почнемо з реєстрації(рис. 3.1.6):

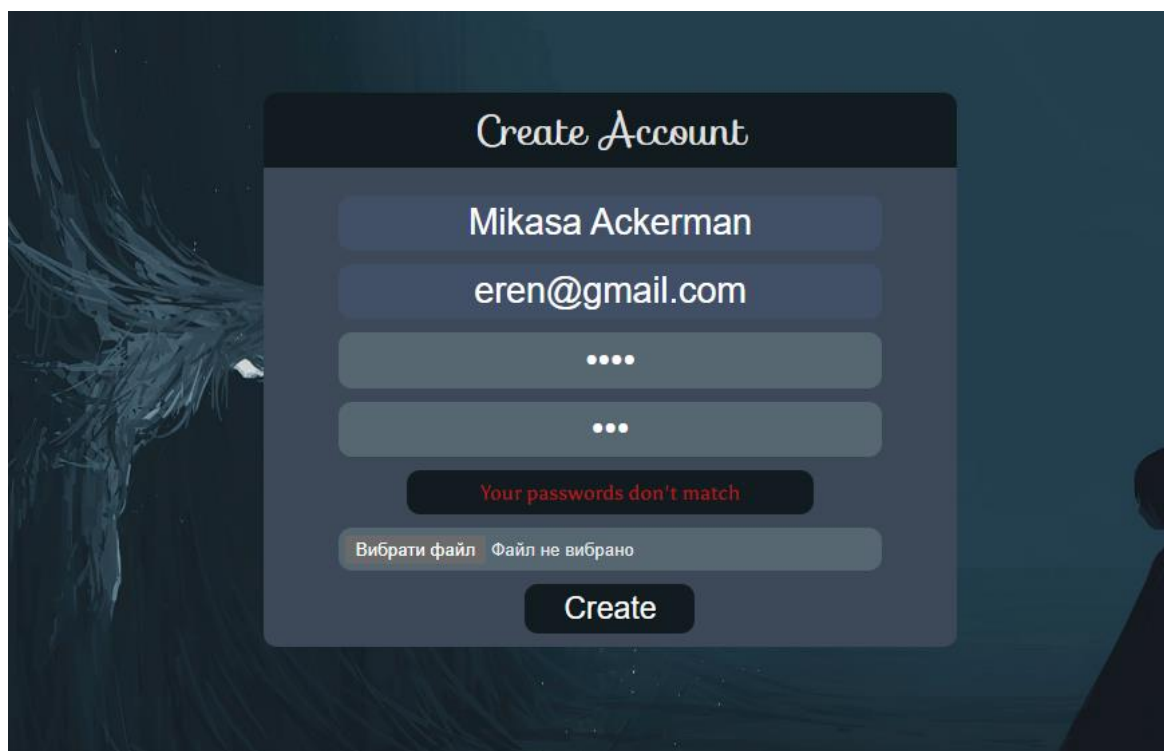


Рис. 3.1.6. Форма реєстрації

Як можна помітити – форма попереджає нас про те, що паролі не сходяться, тому створення користувача неможливе. Для прикладу можна показати й помилку щодо короткого паролю(рис. 3.1.7):

		Джус І. О.			Житомирська політехніка.23.121.08.000 - ПЗ	Арк.
		Кравченко С. М.				19
Змн.	Арк.	№ докум.	Підпис	Дата		

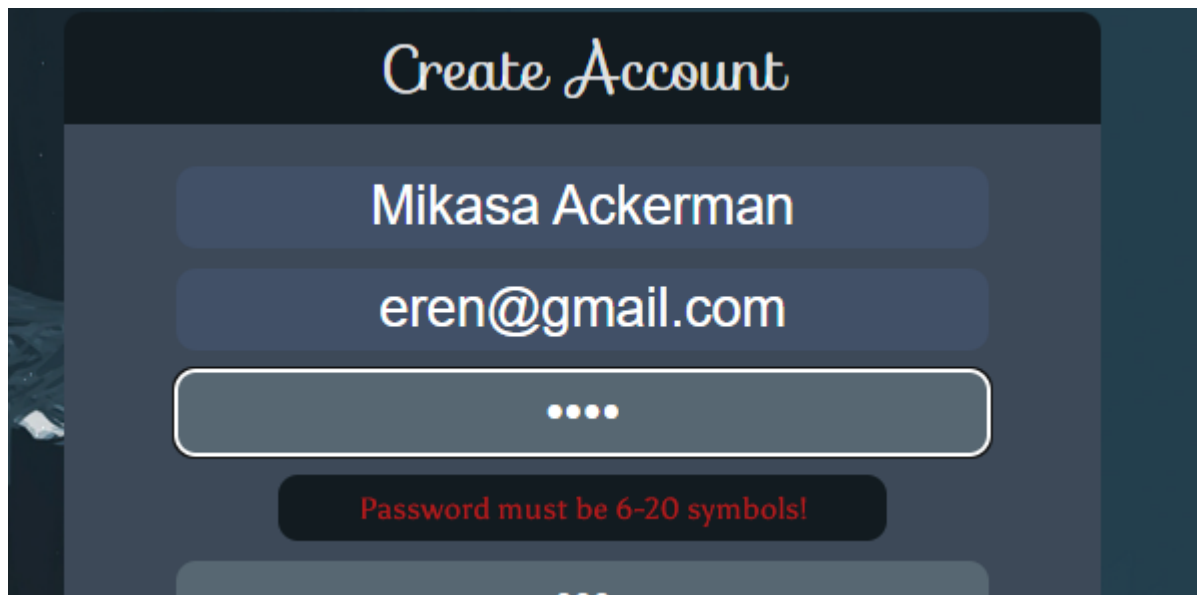


Рис. 3.1.7. Пароль закорткий

Форма авторизації виглядає так(рис. 3.1.8):

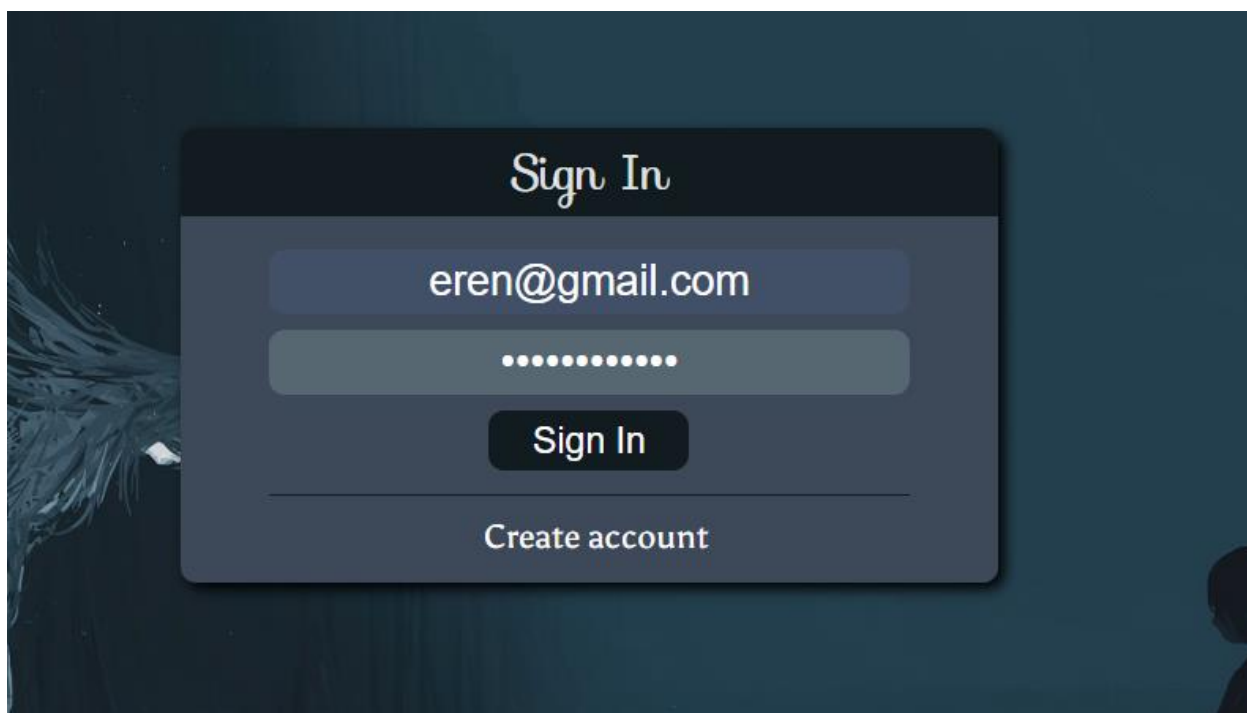


Рис. 3.1.8. Форма авторизації

Після успішної авторизації на хедері відображається ім'я користувача та його аватар(або ж стандартний аватар, якщо користувач не завантажував свою картинку)(рис. 3.1.9).



Рис. 3.1.9. Користувач авторизований

		Джус І. О.			Житомирська політехніка.23.121.08.000 - ПЗ	Арк.
		Кравченко С. М.				20
Змн.	Арк.	№ докум.	Підпис	Дата		

В авторизованого користувача з'являється можливість додавати публікації із загального списку у свій власний за допомогою спеціальної кнопки на сторінці:

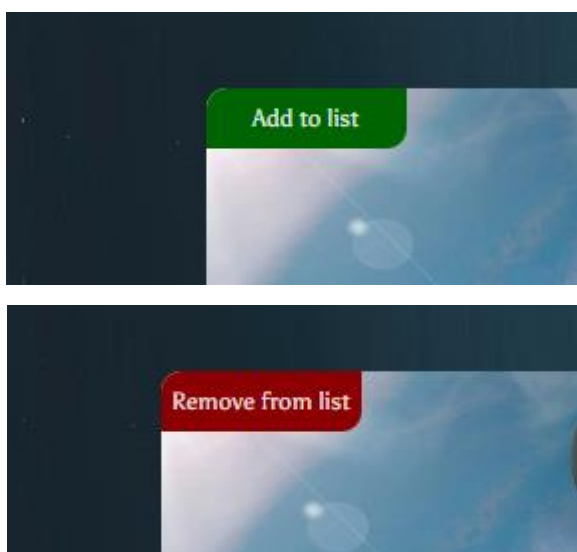


Рис. 3.1.10. Кнопка додавання до списку та вилучення з нього
Після цього на окремій сторінці з'являється обране у нашому випадку аніме:

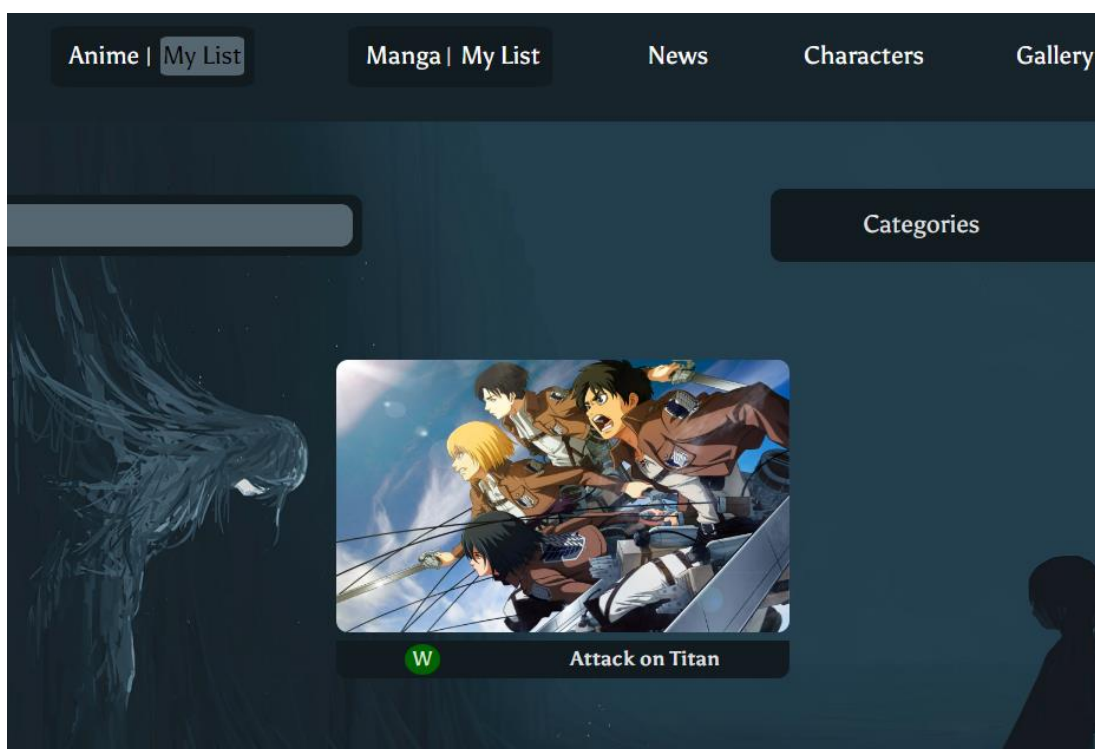


Рис. 3.1.11. Сторінка обраного

Тепер розглянемо вигляд веб-сайту за модератора, який займається керуванням аніме та манг. При авторизації як модератор, користувач отримує можливість видалити та редагувати, наприклад, аніме відразу зі сторінки всіх наявних(рис. 3.1.12).

		Джус І. О.			Житомирська політехніка.23.121.08.000 - ПЗ	Арк.
		Кравченко С. М.				21
Змн.	Арк.	№ докум.	Підпис	Дата		

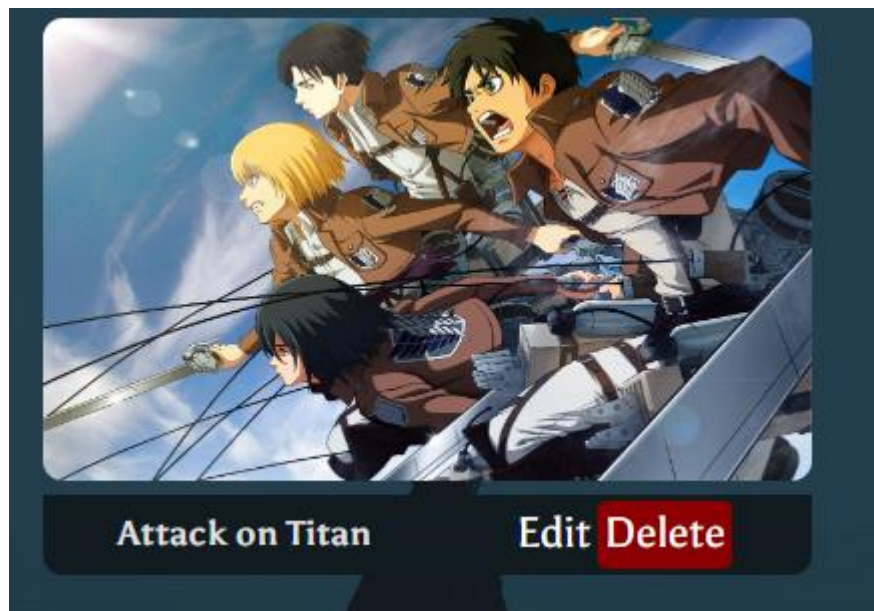


Рис. 3.1.12. Кнопки керування

При редагуванні завантажується форма із заповненими полями поточними даними:

Рис. 3.1.13. Форма редагування

Ця ж форма використовується для додавання нового аніме:

		Джус І. О.			Житомирська політехніка.23.121.08.000 - ПЗ	Арк.
		Кравченко С. М.				22
Змн.	Арк.	№ докум.	Підпис	Дата		

Рис. 3.1.14. Форма додавання(також присутні кнопки для переходу на додавання допоміжних сутностей(категорії, статуси і т.д.))

Якщо говорити про абсолютний доступ(користувача з роллю адміністратора), то окрім можливості керувати абсолютно всіма публікаціями на сайті також є доступ до сторінки з цифровою статистикою(мається на увазі кількість уже завантажених на сайт аніме, манг, новин), і переглядати та за потреби видаляти існуючих модераторів, або додавати нових(рис. 3.1.15).

Рис. 3.1.15. Сторінка адміністратора

3.2. Реалізація операцій обробки даних в БД за напрямком курсової роботи.

Після проектування та визначення логіки роботи веб-додатку необхідно розпочати реалізацію вищезазначеного за допомогою коду. Таким чином, у цьому пункті будуть розглянуті методи, що забезпечують роботу веб-сайту.

Варто почати з моделей представлення колекцій у програмному коді, де визначаються назви полів, їхні типи та посилання на інші колекції, якщо такі передбачено. Для прикладу розглянемо модель Anime:

```
const AnimeSchema = new mongoose.Schema({
  title: {
    type: String,
    required: true,
    unique: true,
  },
  originTitle: {
    type: String,
    unique: true,
  },
  description: {
    type: String,
    required: true,
  },
  categoriesArray: {
    type: [mongoose.Schema.Types.ObjectId],
    ref: "Categories",
    required: true,
  },
  seasons: {
    type: Number,
    required: true,
  },
  series: {
    type: Number,
    required: true,
  },
  years: {
    type: Array,
    required: true,
  },
  status: {
    type: mongoose.Schema.Types.ObjectId,
    ref: "Status",
    required: true,
  },
  author: {
    type: mongoose.Schema.Types.ObjectId,
    ref: "Creator",
    required: true,
  },
});
```



```

    },
    characters: {
      type: [mongoose.Schema.Types.ObjectId],
      ref: "Characters",
    },
    viewsCount: {
      type: Number,
      default: 0,
    },
    images: {
      type: [String],
    },
  },
  {
    timestamps: true,
  }
});

export default mongoose.model("Anime", AnimeSchema);

```

Враховуючи, що основний вид взаємодії сайту з базою даних полягає в CRUD(create, read, update, delete), розглянемо саме його реалізацію та почнемо з методу додавання нових аніме в базу даних, який передбачено у контролері AnimeContrlr:

```

export const createAnime = async (req: any, res: any) => {
  try {
    const document = new AnimeModel({
      title: req.body.title,
      originTitle: req.body.originTitle,
      description: req.body.description,
      categoriesArray: req.body.categoriesArray,
      seasons: req.body.seasons,
      series: req.body.series,
      years: req.body.years.split(","),
      status: req.body.status,
      author: req.body.author,
      characters: req.body.characters,
      images: req.body.images,
    });

    const anime = await document.save();
    res.json(anime);
  } catch (err) {
    console.warn(err);
    res.status(500).json({
      message: "Creating anime failed",
    });
  }
};

```

Даний метод приймає дані із запиту(а той надходить від форми додавання) і формує документ, після чого зберігає його до бази даних, або ж відловлює помилку та повідомляє про це.

		Джус І. О.			Житомирська політехніка.23.121.08.000 - ПЗ	Арк.
		Кравченко С. М.				25
Змн.	Арк.	№ докум.	Підпис	Дата		

Метод оновлення документу працює за рахунок пошуку його по айді та заміни старих даних новими із запиту:

```
export const updateAnime = async (req: any, res: any) => {
  try {
    const animeID = req.params.id;
    await AnimeModel.updateOne(
      {
        _id: animeID,
      },
      {
        title: req.body.title,
        originTitle: req.body.originTitle,
        description: req.body.description,
        categoriesArray: req.body.categoriesArray,
        seasons: req.body.seasons,
        series: req.body.series,
        years: req.body.years.split(","),
        status: req.body.status,
        author: req.body.author,
        characters: req.body.characters,
        images: req.body.images,
      }
    );

    res.json({
      success: true,
    });
  } catch (err) {
    console.warn(err);
    res.status(500).json({
      message: "Cannot update anime",
    });
  }
};
```

Видалення записів з колекції теж відбувається за рахунок пошуку по айді, проте логіка ускладнена наявністю збережених картинок, які теж потрібно видаляти. Через це метод має цикл, де знаходить всі картинки з назвами в збереженому масиві та видаляє їх:

```
export const removeAnime = async (req: any, res: any) => {
  try {
    const animeID = req.params.id;
    const anime = await AnimeModel.findById(animeID);
    if (anime) {
      if (anime.images) {
        for (let i = 0; i < anime.images.length; i++) {
          if (fs.existsSync(`../server/${anime.images[i]}`)) {
            fs.unlink(`../server/${anime.images[i]}`, (err) => {
```

		Джус І. О.			Житомирська політехніка.23.121.08.000 - ПЗ	Арк.
		Кравченко С. М.				
Змн.	Арк.	№ докум.	Підпис	Дата		26

```

        if (err) console.warn(err);
    });
}
}
}
}
await AnimeModel.findByIdAndRemove(animeID);
res.status(204).json({
    success: true,
});
} catch (err) {
    console.warn(err);
    res.status(500).json({
        message: "Cannot find anime",
    });
}
};

```

На останок розглянемо методи читання усіх записів та одного, який
шукається за айді:

```
export const getAllAnime = async (req: any, res: any) => {
  try {
    const anime = await AnimeModel.find()
      .populate("categoriesArray")
      .populate("status")
      .sort({ $natural: -1 })
      .exec();
    res.json(anime);
  } catch (err) {
    console.warn(err);
    res.status(500).json({
      message: "Cannot receive anime",
    });
  }
};
```

Можна помітити додаткові методи під час вибірки під назвою `populate` та `sort`. Перший слугує для видобування повної інформації за айді, який зберігається в полі, а други – сортує документи за параметром. У нашому випадку сортування має ефект реверсу, щоб останні додані аніме відображалися першими.

```
export const getOneAnime = async (req: any, res: any) => {
  try {
    const animeID = req.params.id;
    const anime = await AnimeModel.findOneAndUpdate(
      {
        _id: animeID,
      },
      { $inc: { viewsCount: 1 } }
    )
    .populate("categoriesArray")
    .populate("author")
    .populate("status")
  }
}
```

```

        .populate("characters")
        .exec();
    res.json(anime);
  } catch (err) {
    console.warn(err);
    res.status(500).json({
      message: "Cannot receive anime",
    });
  }
};

```

Що дійсно варто прокоментувати в цьому методі, так це збільшення значення поля `viewsCount` за допомогою `$inc`. Справа в тому, що дане поле відслідковує популярність(кількість переглядів даного запису) для подальших сортувань та фільтрацій.

Як неодноразово було згадано, окрім стандартного CRUDу передбачені також запити до бд на вибір певної кількості(у нашому випадку 3-х) останніх записів. Усе це контролює влаштований метод MongoDB під назвою `limit`, який обмежує кількість вибраних документів:

```

export const getRecentAnime = async (req: any, res: any) => {
  try {
    const anime = await AnimeModel.find()
      .limit(3)
      .sort({ $natural: -1 })
      .exec();
    res.json(anime);
  } catch (err) {
    console.log(err);
    res.status(500).json({
      message: "Cannot receive anime",
    });
  }
};

```

Інші моделі та контролери побудовані за аналогічним алгоритмом, і не потребують бути окремо представленими, чого не скажеш про контролер для взаємодії з моделлю користувачів. У даному блоці наявні декілька особливих методів, які дозволяють авторизованому користувачу додавати до списків обрані аніме та манги, а також вилучати їх:

```

export const updateUserWatched = async (req: any, res: any) => {
  try {
    await UserModel.updateOne(
      {
        _id: req.userID,

```

		Джус І. О.			Житомирська політехніка.23.121.08.000 - ПЗ	Арк.
		Кравченко С. М.				28
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    },
    {
      $push: {
        watchedAnime: req.body.watchedAnime,
        ReadManga: req.body.ReadManga,
      },
    }
  );
  res.json({
    success: true,
  });
} catch (err) {
  console.log(err);
  res.status(500).json({
    message: "Cannot toggle anime",
  });
}
};

```

Принцип роботи методу полягає в тому, щоб при натисканні кнопки айді обраного аніме чи манги додавалося за допомогою \$push до масиву в документі в колекції користувача. Таким чином списки обраного можна буде переглядати на будь-яких пристроях(мається на увазі, що обрані публікації зберігаються не в localStorage та подібному, а в самій колекції users).

Метод вилучення відрізняється лише параметром \$pull, який вилучає з масиву айді публікації.

```

export const removeFromWatched = async (req: any, res: any) => {
  try {
    await UserModel.updateOne(
      {
        _id: req.userID,
      },
      {
        $pull: {
          watchedAnime: req.body.watchedAnime,
          ReadManga: req.body.ReadManga,
        },
      }
    );
    res.json({success: true});
  } catch (err) {
    console.log(err);
    res.status(500).json({message: "Cannot remove anime"});
  }
};

```

Після розгляду методів взаємодії з базою даних можна показати маршрути та так звані кінцеві точки запитів, до яких звертається клієнта частина веб-додатку. Наприклад, роут запиту на додавання нового аніме:

		Джус І. О.			Житомирська політехніка.23.121.08.000 - ПЗ	Арк.
		Кравченко С. М.				29
Змн.	Арк.	№ докум.	Підпис	Дата		

```
app.post("/anime", CheckAuth, CheckProductModerator,
  AnimeValidation.animeValidation, handleValidationErrors,
  AnimeController.createAnime
);
```

Про методи перевірки прав користувача та валідацію вхідних даних детальніше буде в розділі 4, при аналізі адміністрування бази даних та безпекових заходів у веб-додатку.

Статистика та облік формуються за допомогою агрегатів, які підраховують кількість кількості всіх наявних аніме/манг/новин/користувачів на сайті, а також рахується кількість доданих у поточний день:

```
const getAggregateCreatedToday = async (model: Model<any>) => {
  const countDocsCreatedToday: any[] = await model.aggregate([
    {
      $match: {
        createdAt: {
          $gte: today,
        },
      },
    },
    {
      $group: {
        _id: null,
        count: { $sum: 1 },
      },
    },
  ]);
  if (!countDocsCreatedToday[0]) return 0;
  else return countDocsCreatedToday[0].count;
};

const getAggregate = async (model: Model<any>) => {
  const count: any[] = await model.aggregate([
    {
      $group: {
        _id: null,
        count: { $sum: 1 },
      },
    },
  ]);
  return count[0].count;
};
```

Рядок підключення до бази даних:

```
mongoose.connect(`${process.env.MONGO_BD_CONNECTION}`)
  .then(() => console.log("DB Connected"))
  .catch((error: Error) => console.error(error));
```

Самі дані рядка знаходяться у спеціальному файлі оточення(рис. 3.2.1):

		Джус І. О.			Житомирська політехніка.23.121.08.000 - ПЗ	Арк.
		Кравченко С. М.				30
Змн.	Арк.	№ докум.	Підпис	Дата		

```
PORT=4000
MONGO_DB_CONNECTION=mongodb+srv://admin:admin@cluster0.skrw0ja.mongodb.net/ar
SECRET_KEY=asdlkfjs2342
```

Рис. 3.2.1. Вміст файлу оточення

Висновок. У даному розділі було проведено представлення готового веб-сайту, показано та пояснено роботу з інтерфейсом та можливості сайту. Також увага була прикута до опису та пояснення методів та функцій, а також інших частин коду, на якому базується курсова робота.

		Джус І. О.			Житомирська політехніка.23.121.08.000 - ПЗ	Арк.
		Кравченко С. М.				
Змн.	Арк.	№ докум.	Підпис	Дата		31

РОЗДІЛ 4. АДМІНІСТРУВАННЯ БАЗ ДАНИХ ТА БЕЗПЕКА

4.1. Розробка заходів захисту інформації в БД

У даному розділі увага буде присвячена адмініструванню бази даних та безпеці під час роботи з даними, а також їхнього зберігання. Метою таких заходів є завчасне виявлення можливих слабких місць веб-сайту та вживання заходів щодо запобігання помилок у роботі.

Перш за все варто розглянути найбільш вразливий момент, а саме вхідні дані, які потрібно валідувати на сервері та відправляти до бази даних саме у визначеному до цього виді, а в разі невідповідності – відхиляти такі запити. Для цього були передбачені функції валідацій, що зв'язані з моделями та перевіряють всі важливі та критичні дані. Для прикладу розглянемо валідацію даних при додавання аніме:

```
export const animeValidation = [
  body("title", "Invalid title").isString().isLength({ min: 4, max: 20
}),
  body("originTitle", "Invalid origin title").isString().isLength({ min:
4 }),
  body("description", "Invalid description").isString().isLength({ min:
10 }),
  body("categoriesArray", "Invalid categories").isArray(),
  body("seasons", "Invalid seasons").isNumeric(),
  body("series", "Invalid series").isNumeric(),
  body("years", "Invalid years").isString(),
  body("status", "Invalid status").isString(),
  body("author", "Invalid author").isString(),
  body("images", "Invalid images").optional().isArray(),
];
```

Бібліотека «express-validator» дозволяє перевіряти різні параметри для вхідних полів(починаючи з типів і закінчуючи розмірами), а також визначати важливість, чи можливість знехтувати заповненням поля(метод optional).

Після перевірки результати надсилаються на метод-обробник валідації, і вже той є останнім рубіжом, який визначає, створиться новий запис у базі даних, чи запит буде відхилено через помилки.

```
export default (req: any, res: any, next: any) => {
  const errors = validationResult(req);
  if (!errors.isEmpty()) return res.status(400).json(errors.array());
  next();
};
```

		Джус І. О.			Житомирська політехніка.23.121.08.000 - ПЗ	Арк.
		Кравченко С. М.				
Змн.	Арк.	№ докум.	Підпис	Дата		32

Також не менш важливою складовою безпеки є перевірка ролі користувача додатку та визначення на основі цього доступу до певних операцій. Для цього були створені спеціальні методи-перевірки, що звертаються до даних користувача і загалом перевіряють, чи авторизований користувач.

Почнемо з перевірки на авторизацію, яка приймає токен і в разі успішного видобування з нього інформації про користувача – дозволяє продовжити суміжні операції(мається на увазі виконання методу next):

```
export default (req: any, res: any, next: any) => {
  const token = (req.headers.authorization || "").replace(/Bearer\s?/, "");

  if (token) {
    try {
      const decoded: any = jwt.verify(token, process.env.SECRET_KEY);

      req.userID = decoded._id;
      next();
    } catch (error) {
      console.log(error);
      return res.status(403).json({
        message: "Error getting access",
      });
    }
  } else {
    return res.status(403).json({
      message: "Access denied",
    });
  }
};
```

Тепер, для прикладу, розглянемо перевірку модератора новин. Даний метод працює за схожим алгоритмом, але тепер видобуває з токена саме інформацію про роль та порівнює її зі значеннями, визначеними на рівні додатку. Якщо ролі співпадають – користувач отримує доступ:

```
export default async (req: any, res: any, next: any) => {
  const token = (req.headers.authorization || "").replace(/Bearer\s?/, "");

  if (token) {
    try {
      const decoded: any = jwt.verify(token, process.env.SECRET_KEY);

      const role = decoded.role;
      if (role === "newsModerator" || role === "admin") next();
    } else {

```

		Джус І. О.			Житомирська політехніка.23.121.08.000 - ПЗ	Арк.
		Кравченко С. М.				33
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        return res.status(403).json({
            message: "Access denied. You're not news manager",
        });
    } catch (error) {
        console.log(error);
        return res.status(403).json({
            message: "Error getting access",
        });
    }
} else {
    return res.status(403).json({
        message: "Access denied",
    });
}
};

```

З цікаво можна відмітити, що метод також перевіряє, чи часом не є адміністратором поточний користувач, у разі чого він матиме повний доступ.

У межах безпеки паролі користувачів теж повинні бути спеціально оброблені, а саме не зберігатися у вхідному вигляді. З цим допоможе бібліотека «bcrypt», яка хешує рядки та повертає їх саме у такому вигляді:

```
passwordHash: "$2b$10$H6Swfr2qECCcXcL8GEwXOVehsevS/yqvGXNfDR1LVzWAK93RJ65W"
```

Рис. 4.1.1. Хешований пароль

У ході адміністрування бази даних також було опрацьовані та випробувані користувачі на рівні самої бази даних. Їх створення відбувається на офіційній сторінці MongoDB Atlas у розділі проекту, у якому розміщені кластери з базами даних, за вкладкою «Database Access». При створенні користувача можемо додати йому вбудовану роль, або створити власну з більш гнучкими параметрами доступу, як показано на рисунку 4.1.2.

		Джус І. О.			Житомирська політехніка.23.121.08.000 - ПЗ	Арк.
		Кравченко С. М.				
Змн.	Арк.	№ докум.	Підпис	Дата		34

Add Custom Role

Some actions require all clusters in a project to run a minimum version of MongoDB. [More information](#)

Name your custom role and associate its actions and roles with databases and collections.

Custom Role Name

productModerator

Action or Role	Database	Collection
find, insert, remove, updat...	animeArchive	animes
	animeArchive	mangas
	animeArchive	creators
	animeArchive	categories
	animeArchive	roles
	animeArchive	characters
+ Add a database or collection		
+ Add an action or role		

Рис. 4.1.2. Створення ролі в MongoDB Atlas

Створення користувача:

Password Authentication

productModerator

.....

SHOW

[Autogenerate Secure Password](#)

[Copy](#)

Database User Privileges

Configure role based access control by assigning database \$user a mix of one or multiple custom roles, and multiple specific privileges. A user will gain access to the roles assigned to them, not just the actions those roles share in common. **You must assign at least one role or privilege.** [Learn more about roles.](#)

Built-in Role

Select one [built-in role](#) for this user.

Add Built In Role

Custom Roles

Select your [pre-defined custom role\(s\)](#). Create a custom role in the [Custom Roles](#) page.

productModerator

Рис. 4.1.3. Створення користувача з роллю «productModerator»

		Джус І. О.			Житомирська політехніка.23.121.08.000 - ПЗ	Арк.
		Кравченко С. М.				35
Змн.	Арк.	№ докум.	Підпис	Дата		

Database Access

Database Users

Custom Roles

+ ADD NEW DATABASE USER










User Name ↕	Authentication Method ▲	MongoDB Roles	Resources	Actions
 admin	SCRAM	readWriteAnyDatabase@admin	1 Cluster, 0 Federated Database Instances	<div> EDIT</div> <div> DELETE</div>
 productModerator	SCRAM	productModerator@admin	All Resources	<div> EDIT</div> <div> DELETE</div>
 readOnly	SCRAM	readAnyDatabase@admin	All Resources	<div> EDIT</div> <div> DELETE</div>

Рис. 4.1.4. Декілька користувачів

Тепер можна протестувати створеного користувача за допомогою MongoDB Compass. Очікується, що модератора аніме та манг матиме доступ лише до цих колекцій(і колекцій з допоміжними даними).

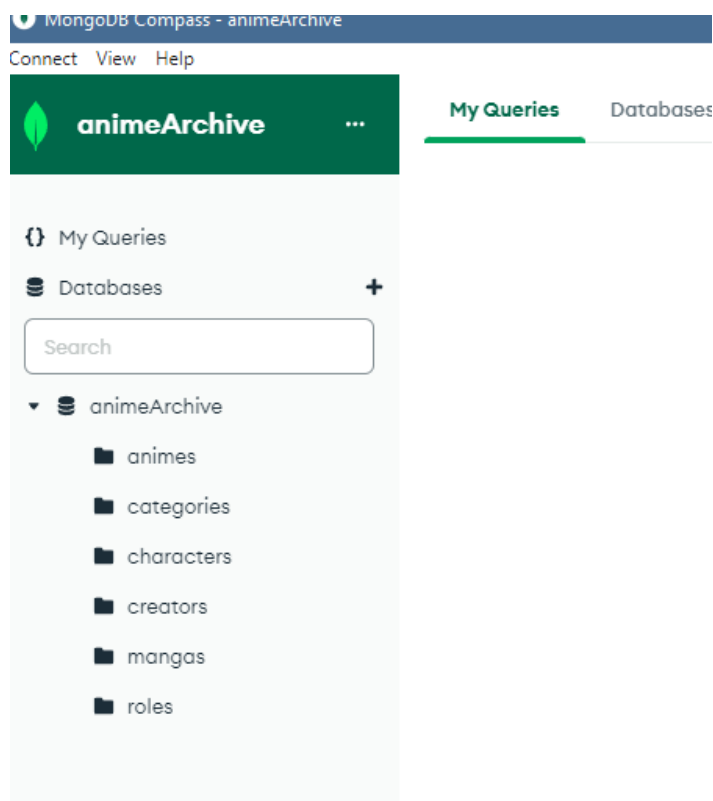


Рис. 4.1.5. Колекції, до яких модератор має доступ

Тепер протестуємо користувача з мінімальним доступом(лише читати колекції) і спробуємо щось видалити:

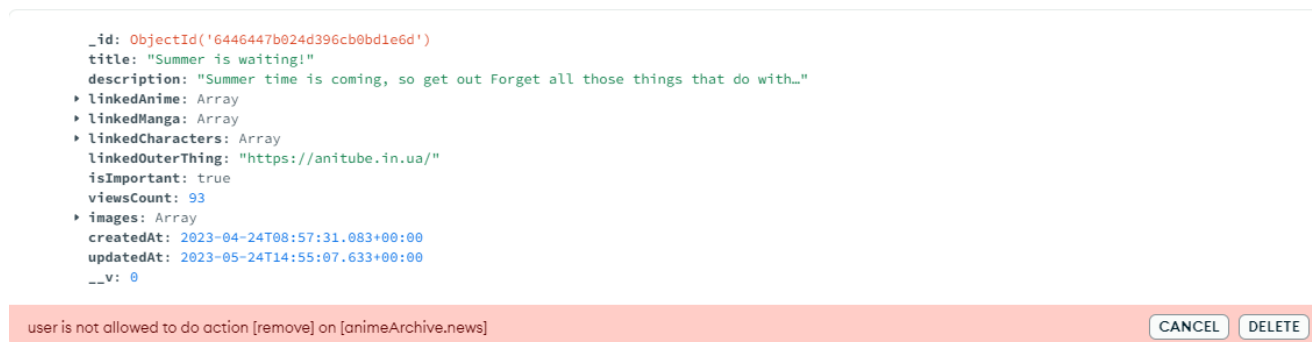


Рис. 4.1.6. Користувачу відмовлено у видаленні новини

Детально про ролі вказано у матриці доступу ролей.

Таблиця 5

Матриця доступу ролей

Таблиці / ролі	Матриця доступу				
	Неавторизований користувач	Авторизований користувач	Модератор аніме та манг	Модератор новин	Адміністратор
animes	1	1	5	1	5
mangas	1	1	5	1	5
creators	1	1	5	1	5
categories	1	1	5	1	5
roles	1	1	5	1	5
characters	1	1	5	1	5
status	1	1	5	1	5
news	1	1	1	5	5
users	2	1, 3	1, 3	1, 3	5

Де 0 – немає доступу, 1 – читання, 2 – вставка, 3 – редагування, 5 – повний доступ.

Висновок. У цьому розділі було досліджено та представлено адміністрування бази даних, а також пояснено аспекти захисту додатку да даних.

		Джус І. О.			Житомирська політехніка.23.121.08.000 - ПЗ	Арк.
		Кравченко С. М.				37
Змн.	Арк.	№ докум.	Підпис	Дата		

ВИСНОВКИ

Протягом виконання курсової роботи, насамперед, були закріплені навички роботи з мовою програмування JavaScript, використано платформу Node.js, фреймворки Express, React та створено і розроблено базу даних на основі MongoDB.

У розділі першому було проведено глобальний та детальний аналіз поставленої задачі. Розкладання теми на підтеми для коректнішого визначення обсягу роботи. Визначено обов'язковий та додатковий функціонал онлайн-енциклопедії аніме та манг, обрано середовище розробки та обговорено вимоги до програмного додатку. Після самостійного опрацювання всі питання та неоднозначні моменти були узгоджені з керівником.

Розділ другий був присвячений проєктуванню та розробці клієнтської та серверної частини веб-сайту, а також моделюванню бази даних, створення моделей та колекцій. Перш за все побудувалися блок-схеми та діаграми, після чого почалася безпосередня розробка.

Третій розділ було присвячено презентації інтерфейсу та додатку загалом. Провівся опис роботи з веб-сайтом, його функціонал та можливості. Скріншоти давали змогу побачити, про який етап йшла мова і точно відображали словесний опис. Протягом всього розділу представлялися методи та блоки коду, що було реалізовані для виконання тих чи інших функцій для підтримки веб-сайту.

Четвертий розділ було присвячено адмініструванню бази даних та безпеці. Тут були проаналізовані заходи захисту збережених даних та перевірки коректності нових. Також була описана робота з користувачами та ролями на рівні бази даних, протестовано їхні можливості та отримано бажаний результат.

Курсова робота надихнула на вивчення нових методів розробки задля покращення та розвитку у напрямку створення веб-сайтів та баз даних.

		Джус І. О.			Житомирська політехніка.23.121.08.000 - ПЗ	Арк.
		Кравченко С. М.				
Змн.	Арк.	№ докум.	Підпис	Дата		38

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Documentation | node.js. Node.js. URL: <https://nodejs.org/en/docs> (date of access: 25.05.2023).
2. Express - Node.js web application framework. Express - Node.js web application framework. URL: <https://expressjs.com/> (date of access: 25.05.2023).
3. Fandom. Fandom. URL: <https://www.fandom.com/> (date of access: 25.05.2023).
4. MongoDB atlas database | multi-cloud database service. MongoDB. URL: <https://www.mongodb.com/atlas/database> (date of access: 25.05.2023).
5. React. React. URL: <https://react.dev/> (date of access: 25.05.2023).
6. Redux toolkit | redux toolkit. Redux Toolkit | Redux Toolkit. URL: <https://redux-toolkit.js.org/> (date of access: 25.05.2023).
7. Team M. D. MongoDB documentation. MongoDB: The Developer Data Platform | MongoDB. URL: <https://www.mongodb.com/docs/> (date of access: 25.05.2023).
8. Wikipedia. Wikipedia. URL: <https://www.wikipedia.org/> (date of access: 25.05.2023).

		Джус І. О.			Житомирська політехніка. 23.121.08.000 - ПЗ	Арк.
		Кравченко С. М.				
Змн.	Арк.	№ докум.	Підпис	Дата		39

ДОДАТКИ

		Джус І. О.			Житомирська політехніка.23.121.08.000 - ПЗ	Арк.
		Кравченко С. М.				
Змн.	Арк.	№ докум.	Підпис	Дата		40

Технічне завдання

1. Загальне положення

1.1 Найменування програмного засобу

Повне найменування програмної системи: "База даних онлайн енциклопедії аніме й манги. Коротка назва програмної системи - "Anime Archive".

1.2. Призначення розробки та область застосування

Програма призначена для зберігання та зручного представлення інформації про аніме та манги, розподіл зацікавивших продуктів між авторизованими користувачами.

Дана програма має бути використана як веб-сайт, що забезпечує доступ з будь-якого пристрою та куточку світу.

1.3. Найменування розробника та замовника.

Розробник даного продукту - студент групи ВТ-21-1 Джус Ігор Олександрович(надалі "розробник").

Замовник програмного продукту – кафедра інженерії програмного забезпечення Державного університету «Житомирська політехніка» в межах виконання курсової з дисципліни «Бази даних» Кравченко Світлана Миколаївна.

2. Підстава для розробки

2.1. Документ на підставі якого ведеться розробка

Робота ведеться на підставі навчального плану за напрямом 121 «Інженерія програмного забезпечення».

3. Вимоги до програми

		Джус І. О.			Житомирська політехніка.23.121.08.000 - ПЗ	Арк.
		Кравченко С. М.				41
Змн.	Арк.	№ докум.	Підпис	Дата		

3.1. Вимоги до функціональних характеристик.

3.1.1. Загальні вимоги

Веб-сайт має забезпечувати:

- можливість дистанційної роботи з робочих станцій локальної мережі.
- постійний доступ користувачів веб-сайту;
- адміністрування для модераторів;
- організацію управління сайтом;
- можливість доступу до бази даних;

3.1.1. Склад виконуваних функцій

Розробити базу даних онлайн енциклопедії аніме й манги з таким функціоналом:

1. Перегляд усього асортименту аніме та манг.
2. Сортування за новизною, популярністю та категоріями.
3. Перегляд інформації про конкретний продукт.
4. Формування власного списку аніме й манг шляхом відмічання їх.
5. Швидкий доступ до нещодавно доданих аніме та манг.
6. Перегляд тематичних новин.
7. Перегляд персонажів, зв'язаних зі своїм проектом.
8. Адміністрування(додавання, редагування та видалення)
9. Авторизація та реєстрація, ролі користувачів.

3.1.2. Організація вхідних і вихідних даних

Вхідними даними є інформація про продукти (назва, жанри, автор, роки виходу, опис, фотографії тощо) та новини(заголовок, короткий опис, текст новини, фото).

		Джус І. О.			Житомирська політехніка.23.121.08.000 - ПЗ	Арк.
		Кравченко С. М.				
Змн.	Арк.	№ докум.	Підпис	Дата		42

Введення оперативних даних повинно виконуватися з використанням діалогових екранних форм, побудованих на основі візуальних компонентів. Введення даних виконується на основі затверджених форм документів: анкета, заява, інформаційна довідка та в режимі online оператором зі слів користувача.

3.1.3. Часові характеристики і розмір пам'яті, необхідної для роботи програми.

Час реакції програми на дії користувача не повинен перевищувати 0,5 с. Час виконання команд меню не більше 1 с. Відображення масивів даних за запитом не більше 3 хвилин. Доступність БД – 90% цілодобово. Операції з'єднання з БД не більше 1 хвилини. Обсяг оперативної пам'яті, необхідний для роботи програми не менше 1Гб. Дисковий простір, необхідний для збереження програми і файлів даних не більше 300 Мб.

3.2. Вимоги до надійності.

3.2.1. Вимоги до надійного функціонування

Програма повинна нормально функціонувати при безперебійній роботі ПК. Доступність БД 90% при одночасному доступі 30 користувачів. При апаратних збоях, відновлення нормальної роботи програми повинне виконуватися після апаратного збою робочої станції – перезавантаження ОС ПК, запуск виконуваного файлу програми.

3.2.2. Контроль вхідної і вихідної інформації

Для контролю коректності вхідної інформації: використання механізму авто заповнення та вибору за переліком. Визначені некоректні дії повинні супроводжуватись повідомленнями про помилку і блокуванням операцій оновлення даних. В системі має бути передбачений захист від загального блокування.

		Джус І. О.			Житомирська політехніка.23.121.08.000 - ПЗ	Арк.
		Кравченко С. М.				
Змн.	Арк.	№ докум.	Підпис	Дата		43

Додаток А:

Вихідний код проекту(наведена лише серверна частина, оскільки темою курсової роботи є реалізація бази даних та сервера для взаємодії з нею):

Контролер аніме:

```
import AnimeModel from "../model/Anime.js";

import fs from "fs";
import { Query } from "mongoose";

export const createAnime = async (req: any, res: any) => {
  try {
    const document = new AnimeModel({
      title: req.body.title,
      originTitle: req.body.originTitle,
      description: req.body.description,
      categoriesArray: req.body.categoriesArray,
      seasons: req.body.seasons,
      series: req.body.series,
      years: req.body.years.split(","),
      status: req.body.status,
      author: req.body.author,
      characters: req.body.characters,
      images: req.body.images,
    });

    const anime = await document.save();
    res.json(anime);
  } catch (err) {
    console.warn(err);
    res.status(500).json({
      message: "Creating anime failed",
    });
  }
};

export const updateAnime = async (req: any, res: any) => {
  try {
    const animeID = req.params.id;
    await AnimeModel.updateOne(
      {
        _id: animeID,
      },
      {
        title: req.body.title,
        originTitle: req.body.originTitle,
        description: req.body.description,
        categoriesArray: req.body.categoriesArray,
        seasons: req.body.seasons,
        series: req.body.series,
      }
    );
  }
};
```

		Джус І. О.			Житомирська політехніка.23.121.08.000 - ПЗ	Арк.
		Кравченко С. М.				
Змн.	Арк.	№ докум.	Підпис	Дата		44

```

        years: req.body.years.split(","),
        status: req.body.status,
        author: req.body.author,
        characters: req.body.characters,
        images: req.body.images,
    }
    );

    res.json({
        success: true,
    });
} catch (err) {
    console.warn(err);
    res.status(500).json({
        message: "Cannot update anime",
    });
}
};

export const removeAnime = async (req: any, res: any) => {
    try {
        const animeID = req.params.id;
        const anime = await AnimeModel.findById(animeID);
        if (anime) {
            if (anime.images) {
                for (let i = 0; i < anime.images.length; i++) {
                    if (fs.existsSync(`../server/${anime.images[i]}`)) {
                        fs.unlink(`../server/${anime.images[i]}`, (err) => {
                            if (err) console.warn(err);
                        });
                    }
                }
            }
        }
        await AnimeModel.findByIdAndRemove(animeID);
        res.status(204).json({
            success: true,
        });
    } catch (err) {
        console.warn(err);
        res.status(500).json({
            message: "Cannot find anime",
        });
    }
};

export const getAllAnime = async (req: any, res: any) => {
    try {
        const anime = await AnimeModel.find()
            .populate("categoriesArray")
            .populate("status")
            .sort({ $natural: -1 })
    }
};

```

		Джус І. О.			Житомирська політехніка.23.121.08.000 - ПЗ	Арк.
		Кравченко С. М.				
Змн.	Арк.	№ докум.	Підпис	Дата		45

```

        .exec();
        res.json(anime);
    } catch (err) {
        console.warn(err);
        res.status(500).json({
            message: "Cannot receive anime",
        });
    }
};

export const getOneAnime = async (req: any, res: any) => {
    try {
        const animeID = req.params.id;
        const anime = await AnimeModel.findOneAndUpdate(
            {
                _id: animeID,
            },
            { $inc: { viewsCount: 1 } }
        )
        .populate("categoriesArray")
        .populate("author")
        .populate("status")
        .populate("characters")
        .exec();
        res.json(anime);
    } catch (err) {
        console.warn(err);
        res.status(500).json({
            message: "Cannot receive anime",
        });
    }
};

export const getRecentAnime = async (req: any, res: any) => {
    try {
        const anime = await AnimeModel.find()
            .limit(3)
            .sort({ $natural: -1 })
            .exec();
        res.json(anime);
    } catch (err) {
        console.log(err);
        res.status(500).json({
            message: "Cannot receive anime",
        });
    }
};

export const getPopularAnime = async (req: any, res: any) => {
    try {
        const anime = await AnimeModel.find()
            .populate("status")

```

		Джус І. О.			Житомирська політехніка.23.121.08.000 - ПЗ	Арк.
		Кравченко С. М.				
Змн.	Арк.	№ докум.	Підпис	Дата		46

```

        .sort({ viewsCount: -1 })
        .exec();
    res.json(anime);
} catch (err) {
    console.warn(err);
    res.status(500).json({
        message: "Cannot receive anime",
    });
}
};

```

Контролер персонажа:

```

import CharacterModel from "../model/Character.js";
import fs from "fs";

export const createCharacter = async (req: any, res: any) => {
    try {
        const document = new CharacterModel(req.body);

        const character = await document.save();
        res.json(character);
    } catch (err) {
        console.warn(err);
        res.status(500).json({
            message: "Creating character failed",
        });
    }
};

export const updateCharacter = async (req: any, res: any) => {
    try {
        const characterID = req.params.id;
        await CharacterModel.updateOne(
            {
                _id: characterID,
            },
            req.body
        );
        res.json({
            success: true,
        });
    } catch (err) {
        console.warn(err);
        res.status(500).json({
            message: "Cannot update character",
        });
    }
};

export const removeCharacter = async (req: any, res: any) => {

```

		Джус І. О.			Житомирська політехніка. 23.121.08.000 - ПЗ	Арк.
		Кравченко С. М.				47
Змн.	Арк.	№ докум.	Підпис	Дата		

```

try {
  const characterID = req.params.id;
  const character = await CharacterModel.findById(characterID);
  if (character) {
    if (character.images) {
      for (let i = 0; i < character.images.length; i++) {
        if (fs.existsSync(`../server${character.images[i]}`)) {
          fs.unlink(`../server${character.images[i]}`, (err) => {
            if (err) console.warn(err);
          });
        }
      }
    }
  }
  await CharacterModel.findByIdAndRemove(characterID);
  res.status(204).json({
    success: true,
  });
} catch (err) {
  console.warn(err);
  res.status(500).json({
    message: "Cannot find character",
  });
}
};

export const getAllCharacter = async (req: any, res: any) => {
  try {
    const character = await CharacterModel.find().sort({ $natural:
-1 }).exec();
    res.json(character);
  } catch (err) {
    console.warn(err);
    res.status(500).json({
      message: "Cannot receive character",
    });
  }
};

export const getOneCharacter = async (req: any, res: any) => {
  try {
    const characterID = req.params.id;
    const character = await CharacterModel.findOneAndUpdate(
      {
        _id: characterID,
      },
      { $inc: { viewsCount: 1 } }
    )
      .populate("partnersArray")
      .exec();
    res.json(character);
  } catch (err) {

```



```

        console.warn(err);
        res.status(500).json({
            message: "Cannot receive character",
        });
    }
};

export const getPopularCharacter = async (req: any, res: any) => {
    try {
        const character = await CharacterModel.find()
            .sort({ viewsCount: -1 })
            .limit(1)
            .exec();
        res.json(character);
    } catch (err) {
        console.warn(err);
        res.status(500).json({
            message: "Cannot receive character",
        });
    }
};

export const getPopularCharacters = async (req: any, res: any) => {
    try {
        const character = await CharacterModel.find()
            .sort({ viewsCount: -1 })
            .exec();
        res.json(character);
    } catch (err) {
        console.warn(err);
        res.status(500).json({
            message: "Cannot receive characters",
        });
    }
};

```

Контролер користувача:

```

import jwt from "jsonwebtoken";
import bcrypt from "bcrypt";
import dotenv from "dotenv";

import UserModel from "../model/User.js";
import fs from "fs";

dotenv.config();

export const registerUser = async (req: any, res: any) => {
    try {
        const password = req.body.password;
        const salt: string = await bcrypt.genSalt();
    }
};

```

		Джус І. О.			Житомирська політехніка.23.121.08.000 - ПЗ	Арк.
		Кравченко С. М.				49
Змн.	Арк.	№ докум.	Підпис	Дата		

```

const hash: string = await bcrypt.hash(password, salt);

const document: any = new UserModel({
  email: req.body.email,
  fullName: req.body.fullName,
  role: req.body.role,
  avatarUrl: req.body.avatarUrl,
  passwordHash: hash,
});
const user = await document.save();

res.json({
  user,
});
} catch (err) {
  console.warn(err);
  res.status(500).json({
    message: "Register failed",
  });
}
};

export const loginUser = async (req: any, res: any) => {
  try {
    const user: any = await UserModel.findOne({ email:
req.body.email })
      .populate("role")
      .exec();

    if (!user) {
      return res.status(404).json({
        message: "Login or email not found",
      });
    }

    const isValidPass = await bcrypt.compare(
      req.body.password,
      user._doc.passwordHash
    );

    if (!isValidPass) {
      return res.status(400).json({
        message: "Login or email not found",
      });
    }

    const token = jwt.sign(
      {
        _id: user._id,
        role: user.role.name,
        watchedAnime: user.watchedAnime,
        readManga: user.ReadManga,

```

		Джус І. О.			Житомирська політехніка.23.121.08.000 - ПЗ	Арк.
		Кравченко С. М.				
Змн.	Арк.	№ докум.	Підпис	Дата		50

```

    },
    process.env.SECRET_KEY,
    {
      expiresIn: "2d",
    }
  );

  const { passwordHash, ...userData } = user._doc;
  res.json({
    ...userData,
    token,
  });
} catch (err) {
  console.log(err);
  return res.status(404).json({
    message: "Login failed",
  });
}
};

export const profileUser = async (req: any, res: any) => {
  try {
    const user: any = await UserModel.findById(req.userID)
      .populate("role")
      .exec();

    if (!user) {
      return res.status(404).json({
        message: "User not found",
      });
    }

    const { passwordHash, ...userData } = user._doc;

    res.json(userData);
  } catch (err) {
    console.log(err);
    res.status(500).json({
      message: "Access denied",
    });
  }
};

export const updateUserWatched = async (req: any, res: any) => {
  try {
    await UserModel.updateOne(
      {
        _id: req.userID,
      },
      {
        $push: {
          watchedAnime: req.body.watchedAnime,

```

		Джус І. О.			Житомирська політехніка.23.121.08.000 - ПЗ	Арк.
		Кравченко С. М.				
Змн.	Арк.	№ докум.	Підпис	Дата		51

```

        ReadManga: req.body.ReadManga,
      },
    }
  );
  res.json({
    success: true,
  });
} catch (err) {
  console.log(err);
  res.status(500).json({
    message: "Cannot toggle anime",
  });
}
};

export const removeFromWatched = async (req: any, res: any) => {
  try {
    await UserModel.updateOne(
      {
        _id: req.userID,
      },
      {
        $pull: {
          watchedAnime: req.body.watchedAnime,
          ReadManga: req.body.ReadManga,
        },
      }
    );
    res.json({
      success: true,
    });
  } catch (err) {
    console.log(err);
    res.status(500).json({
      message: "Cannot remove anime",
    });
  }
};

export const updateUserInfo = async (req: any, res: any) => {
  try {
    await UserModel.updateOne(
      {
        _id: req.userID,
      },
      req.body
    );
    res.json({
      success: true,
    });
  } catch (err) {
    console.log(err);
  }
};

```

```

    res.status(500).json({
      message: "Cannot update user info",
    });
  }
};

export const removeUser = async (req: any, res: any) => {
  try {
    const userID = req.params.id;
    const user = await UserModel.findByIdAndRemove(userID);
    if (fs.existsSync(`../server${user.avatarUrl}`)) {
      fs.unlink(`../server${user.avatarUrl}`, (err) => {
        if (err) console.warn(err);
      });
    }
    res.status(204).json({
      success: true,
    });
  } catch (err) {
    console.log(err);
    res.status(500).json({
      message: "Cannot find user",
    });
  }
};

```

Модель аніме:

```

import mongoose from "mongoose";

const AnimeSchema = new mongoose.Schema(
  {
    title: {
      type: String,
      required: true,
      unique: true,
    },
    originTitle: {
      type: String,
      unique: true,
    },
    description: {
      type: String,
      required: true,
    },
    categoriesArray: {
      type: [mongoose.Schema.Types.ObjectId],
      ref: "Categories",
      required: true,
    },
    seasons: {

```

		Джус І. О.			Житомирська політехніка.23.121.08.000 - ПЗ	Арк.
		Кравченко С. М.				
Змн.	Арк.	№ докум.	Підпис	Дата		53

```

        type: Number,
        required: true,
    },
    series: {
        type: Number,
        required: true,
    },
    years: {
        type: Array,
        required: true,
    },
    status: {
        type: mongoose.Schema.Types.ObjectId,
        ref: "Status",
        required: true,
    },
    author: {
        type: mongoose.Schema.Types.ObjectId,
        ref: "Creator",
        required: true,
    },
    characters: {
        type: [mongoose.Schema.Types.ObjectId],
        ref: "Characters",
    },
    viewsCount: {
        type: Number,
        default: 0,
    },
    images: {
        type: [String],
    },
},
{
    timestamps: true,
}
);

export default mongoose.model("Anime", AnimeSchema);

```

Модель персонажа:

```

import mongoose from "mongoose";

const CharacterSchema = new mongoose.Schema(
    {
        fullName: {
            type: String,
            required: true,
            unique: true,
        },
    },
    {
        timestamps: true,
    }
);

```

		Джус І. О.			Житомирська політехніка. 23.121.08.000 - ПЗ	Арк.
		Кравченко С. М.				
Змн.	Арк.	№ докум.	Підпис	Дата		54

```

age: {
  type: String,
  required: true,
  default: "Not given",
},
sex: {
  type: String,
  required: true,
  default: "Not given",
},
race: {
  type: String,
  required: true,
},
status: {
  type: String,
  required: true,
},
partnersArray: {
  type: [mongoose.Schema.Types.ObjectId],
  ref: "Characters",
},
appearance: {
  type: String,
  required: true,
},
personality: {
  type: String,
  required: true,
},
viewsCount: {
  type: Number,
  default: 0,
},
images: {
  type: [String],
},
},
{
  timestamps: true,
}
);

export default mongoose.model("Characters", CharacterSchema);

```

Модель користувача

```

import mongoose from "mongoose";

const UserSchema = new mongoose.Schema(
  {

```

		Джус І. О.			Житомирська політехніка.23.121.08.000 - ПЗ	Арк.
		Кравченко С. М.				
Змн.	Арк.	№ докум.	Підпис	Дата		55

```

    fullName: {
      type: String,
      required: true,
    },
    email: {
      type: String,
      required: true,
      unique: true,
    },
    passwordHash: {
      type: String,
      required: true,
    },
    role: {
      type: mongoose.Schema.Types.ObjectId,
      ref: "Roles",
      default: "642c6ba083145c6516e56f9e",
    },
    watchedAnime: {
      type: [mongoose.Schema.Types.ObjectId],
      ref: "Anime",
    },
    ReadManga: {
      type: [mongoose.Schema.Types.ObjectId],
      ref: "Manga",
    },
    avatarUrl: String,
  },
  {
    timestamps: true,
  }
);

export default mongoose.model("User", UserSchema);

```

Стартовий файл index.ts

```

import express from "express";
import cors from "cors";
import mongoose from "mongoose";
import dotenv from "dotenv";
import multer from "multer";

import { handleValidationErrors } from "../utils/imports.js";
import {
  CheckAuth,
  CheckProductModerator,
  CheckAdmin,
  CheckNewsModerator,
} from "../utils/checks/imports.js";

```

		Джус І. О.			Житомирська політехніка.23.121.08.000 - ПЗ	Арк.
		Кравченко С. М.				
Змн.	Арк.	№ докум.	Підпис	Дата		56


```

import {
  CategoryController,
  CreatorController,
  AnimeController,
  UserController,
  StatusController,
  RoleController,
  MangaController,
  NewsController,
  CharacterController,
  AdminInfoController,
} from "../controllers/imports.js";

import {
  UserValidation,
  AnimeValidation,
  CategoryValidation,
  CreatorValidation,
  StatusValidation,
  RoleValidation,
  MangaValidation,
  NewsValidation,
  CharacterValidation,
} from "../validations/imports.js";

//env config
dotenv.config();

//db connection
mongoose
  .connect(`${process.env.MONGO_BD_CONNECTION}`)
  .then(() => console.log("DB Connected"))
  .catch((error: Error) => console.error(error));

//app-express creating
const app = express();

const storage = multer.diskStorage({
  destination: (_, __, cb) => {
    cb(null, "uploads");
  },
  filename: (_, file, cb) => {
    cb(null, file.originalname);
  },
});

const upload = multer({ storage: storage });

app.use(express.json());
app.use(cors());
app.use("/uploads", express.static("uploads"));

```

		Джус І. О.			Житомирська політехніка.23.121.08.000 - ПЗ	Арк.
		Кравченко С. М.				
Змн.	Арк.	№ докум.	Підпис	Дата		57

```

//routes
//uploads
app.post("/upload", upload.array("image"), (req: any, res: any) =>
{
    res.json({
        message: "success",
    });
});

//admin
app.get(
    "/admin-info",
    CheckAuth,
    CheckAdmin,
    AdminInfoController.getNumberOfEntities
);

//user
app.post(
    "/register",
    UserValidation.registerValidation,
    handleValidationErrors,
    UserController.registerUser
);
app.post(
    "/login",
    UserValidation.loginValidation,
    handleValidationErrors,
    UserController.loginUser
);
app.get("/profile", CheckAuth, UserController.profileUser);
app.patch("/user", CheckAuth, UserController.updateUserWatched);
app.patch("/watched-list", CheckAuth,
UserController.removeFromWatched);
app.delete(
    "/user-delete/:id",
    CheckAuth,
    CheckAdmin,
    UserController.removeUser
);

//role
app.post(
    "/role",
    CheckAuth,
    CheckAdmin,
    RoleValidation.roleValidation,
    handleValidationErrors,
    RoleController.createRole
);
app.delete("/role/:id", CheckAuth, CheckAdmin,
RoleController.removeRole);

```

		Джус І. О.			Житомирська політехніка.23.121.08.000 - ПЗ	Арк.
		Кравченко С. М.				
Змн.	Арк.	№ докум.	Підпис	Дата		58

```

app.patch(
    "/role/:id",
    CheckAuth,
    CheckAdmin,
    RoleValidation.roleValidation,
    handleValidationErrors,
    RoleController.updateRole
);

app.get("/role/", CheckAuth, CheckAdmin,
RoleController.getAllRoles);
app.get("/role/:id", CheckAuth, CheckAdmin,
RoleController.getOneRole);

//category
app.post(
    "/category",
    CheckAuth,
    CheckProductModerator,
    CategoryValidation.categoryValidation,
    handleValidationErrors,
    CategoryController.createCategory
);
app.delete(
    "/category/:id",
    CheckAuth,
    CheckProductModerator,
    CategoryController.removeCategory
);
app.patch(
    "/category/:id",
    CheckAuth,
    CheckProductModerator,
    CategoryValidation.categoryValidation,
    handleValidationErrors,
    CategoryController.updateCategory
);
app.get("/category/", CategoryController.getAllCategories);
app.get("/category/:id", CategoryController.getOneCategory);

//creator
app.post(
    "/creator",
    CheckAuth,
    CheckProductModerator,
    CreatorValidation.creatorValidation,
    handleValidationErrors,
    CreatorController.createCreator
);
app.delete(
    "/creator/:id",
    CheckAuth,

```

		Джус І. О.			Житомирська політехніка.23.121.08.000 - ПЗ	Арк.
		Кравченко С. М.				
Змн.	Арк.	№ докум.	Підпис	Дата		59

```

    CheckProductModerator,
    CreatorController.removeCreator
);
app.patch(
    "/creator/:id",
    CheckAuth,
    CheckProductModerator,
    CreatorValidation.creatorValidation,
    handleValidationErrors,
    CreatorController.updateCreator
);
app.get("/creator/", CreatorController.getAllCreators);
app.get("/creator/:id", CreatorController.getOneCreator);

//status
app.post(
    "/status",
    CheckAuth,
    CheckProductModerator,
    StatusValidation.statusValidation,
    handleValidationErrors,
    StatusController.createStatus
);
app.delete(
    "/status/:id",
    CheckAuth,
    CheckProductModerator,
    StatusController.removeStatus
);
app.patch(
    "/status/:id",
    CheckAuth,
    CheckProductModerator,
    StatusValidation.statusValidation,
    handleValidationErrors,
    StatusController.updateStatus
);
app.get("/status/", StatusController.getAllStatuses);
app.get("/status/:id", StatusController.getOneStatus);
//character
app.post(
    "/character",
    CheckAuth,
    CheckProductModerator,
    CharacterValidation.characterValidation,
    handleValidationErrors,
    CharacterController.createCharacter
);
app.delete(
    "/character/:id",
    CheckAuth,
    CheckProductModerator,

```

		Джус І. О.			Житомирська політехніка.23.121.08.000 - ПЗ	Арк.
		Кравченко С. М.				
Змн.	Арк.	№ докум.	Підпис	Дата		60

```

CharacterController.removeCharacter
);
app.patch(
    "/character/:id",
    CheckAuth,
    CheckProductModerator,
    CharacterValidation.characterValidation,
    handleValidationErrors,
    CharacterController.updateCharacter
);
app.get("/character", CharacterController.getAllCharacter);
app.get("/character/:id", CharacterController.getOneCharacter);
app.get("/character-popular/",
CharacterController.getPopularCharacter);
app.get("/characters-popular/",
CharacterController.getPopularCharacters);
//anime
app.post(
    "/anime",
    CheckAuth,
    CheckProductModerator,
    AnimeValidation.animeValidation,
    handleValidationErrors,
    AnimeController.createAnime
);
app.delete(
    "/anime/:id",
    CheckAuth,
    CheckProductModerator,
    AnimeController.removeAnime
);
app.patch(
    "/anime/:id",
    CheckAuth,
    CheckProductModerator,
    AnimeValidation.animeValidation,
    handleValidationErrors,
    AnimeController.updateAnime
);
app.get("/anime/", AnimeController.getAllAnime);
app.get("/anime/popular", AnimeController.getPopularAnime);
app.get("/anime/:id", AnimeController.getOneAnime);
app.get("/anime-recent/", AnimeController.getRecentAnime);

//manga
app.post(
    "/manga",
    CheckAuth,
    CheckProductModerator,
    MangaValidation.mangaValidation,
    handleValidationErrors,
    MangaController.createManga

```

		Джус І. О.			Житомирська політехніка.23.121.08.000 - ПЗ	Арк.
		Кравченко С. М.				
Змн.	Арк.	№ докум.	Підпис	Дата		61

```

);
app.delete(
  "/manga/:id",
  CheckAuth,
  CheckProductModerator,
  MangaController.removeManga
);
app.patch(
  "/manga/:id",
  CheckAuth,
  CheckProductModerator,
  MangaValidation.mangaValidation,
  handleValidationErrors,
  MangaController.updateManga
);
app.get("/manga/", MangaController.getAllManga);
app.get("/manga/popular", MangaController.getPopularManga);
app.get("/manga/:id", MangaController.getOneManga);
app.get("/manga-recent/", MangaController.getRecentManga);
app.post(
  "/news",
  CheckAuth,
  CheckNewsModerator,
  NewsValidation.newsValidation,
  handleValidationErrors,
  NewsController.createNews
);
app.delete(
  "/news/:id",
  CheckAuth,
  CheckNewsModerator,
  NewsController.removeNews
);
app.patch(
  "/news/:id",
  CheckAuth,
  CheckNewsModerator,
  NewsValidation.newsValidation,
  handleValidationErrors,
  NewsController.updateNews
);
app.get("/news/", NewsController.getAllNews);
app.get("/news/popular", NewsController.getPopularNews);
app.get("/news/:id", NewsController.getOneNews);
app.get("/news-recent/", NewsController.getRecentNews);
const server = app.listen(process.env.PORT, () => {
  console.log("Server is ON");
});
server.on("error", (e) => console.error("Error", e));

```