МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"

ІКНІ
Кафедра **ПЗ**

ЗВІТ

До лабораторної роботи № 6
На тему: *"Основні типи та функції доступу до БД MongoDB"*
З дисципліни: *"Програмування в Інтернет"*

Лектор:
асистент каф. ПЗ
Степанов Д. С.

Виконав:
ст. гр. ПЗ-22
Гамела Б. А.

Перевірила:
старша викладачка каф. ПЗ
Грицай О. Д.

«___» _____ 2025 р.

∑ = ____ _____

Львів – 2025

Тема роботи: Основні типи та функції доступу до БД MongoDB.

Мета роботи: Засвоїти елементи створення, модифікації, читання та занесення даних з таблиць БД засобами Node.js.

Теоретичні відомості

MongoDB — це NoSQL база даних, яка зберігає дані у вигляді JSON-подібних документів, об'єднаних у колекції. На відміну від реляційних баз даних (наприклад, MySQL), MongoDB не вимагає фіксованої схеми, що забезпечує гнучкість структури даних. Документи в колекціях можуть мати різні поля, що робить її ідеальною для high-load проєктів із великими обсягами даних.

Основні особливості MongoDB:

- **Гнучкість**: Документи в одній колекції можуть мати різні набори полів.
- **Документо-орієнтована модель**: Дані зберігаються як об'єкти, подібні до JSON.
- **Масштабованість**: Підходить для обробки великих обсягів даних.

Mongoose — це ODM (Object Document Mapper) бібліотека для Node.js, яка спрощує роботу з MongoDB. Вона дозволяє визначати схеми з чіткими типами даних і забезпечує зручний інтерфейс для створення, читання, оновлення та видалення даних.

Типи даних у Mongoose:

- String, Number, Date, Buffer, Boolean, Mixed, ObjectId, Array.
- Для кожного типу можна налаштувати: значення за замовчуванням, валідацію, обов'язковість поля, геттери/сеттери, індекси.
- Додаткові опції для String: конвертація регістру, обрізання, регулярні вирази, переліки допустимих значень.
- Для Number і Date: мінімальні/максимальні значення.

Особливості типів даних:

- **Buffer**: Для зберігання двійкових даних (зображення, PDF).
- **Mixed**: Дозволяє зберігати дані будь-якого типу, але обмежує валідацію.
- **ObjectId**: Унікальний ідентифікатор, часто використовується для посилань між документами.
- **Array**: Підтримує масиви з операціями JavaScript (push, pop тощо).

Встановлення та підключення:

- MongoDB встановлюється з офіційного сайту (Community Server).
- Mongoose встановлюється через npm: npm install mongoose --save.
- Підключення до бази: mongoose.connect('mongodb://localhost/database').

Робота з даними:

- **Схеми та моделі**: Схеми визначають структуру документів, моделі — це класи для роботи з даними.
- **Операції**: Створення (save), пошук (find, findOne, findById), оновлення (findByIdAndUpdate, save), валідація даних.
- **Посилання**: Використання ObjectId для зв'язків між колекціями (наприклад, книга посилається на автора).

Приклад: Схема для автора з полями імені, біографії, соціальних мереж і валідацією URL, а також схема книги з посиланням на автора та масивом рейтингів. Дані зберігаються та оновлюються через методи Mongoose.

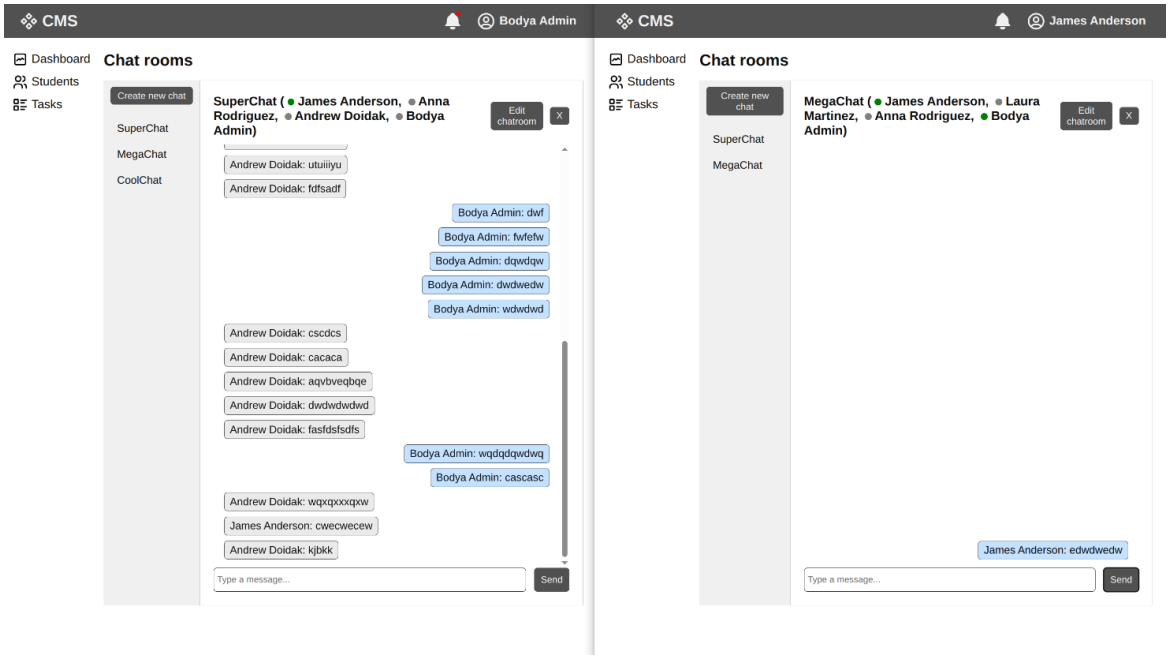
MongoDB і Mongoose забезпечують гнучке та ефективне управління даними в Node.js-додатках, особливо для проєктів реального часу, таких як чати.

Формулювання завдання

1. Розробити web-сторінку згідно макета (wireframe).
2. Сторінка повинна відповідати наступним вимогам.

Результати виконання (спільні 5-6 лабораторна)

Репозиторій проекту на github.com: PI-Labs



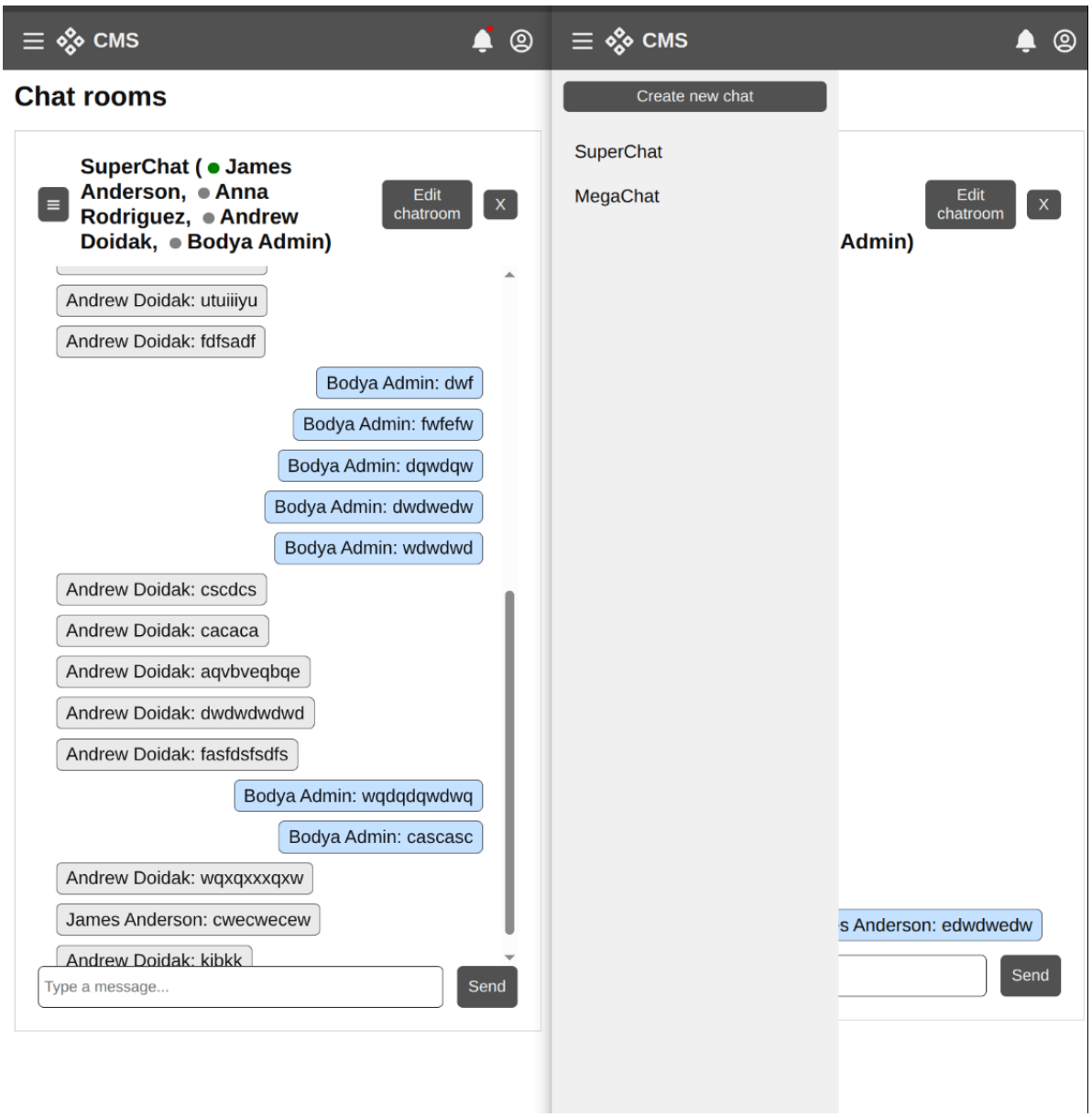


Рис. 1-2. Екран чатів

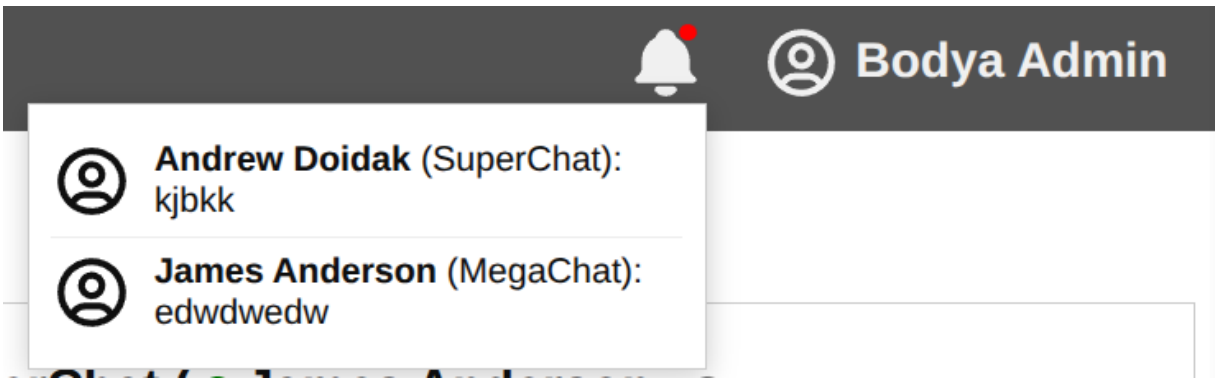


Рис. 3. Сповіщення про нові повідомлення

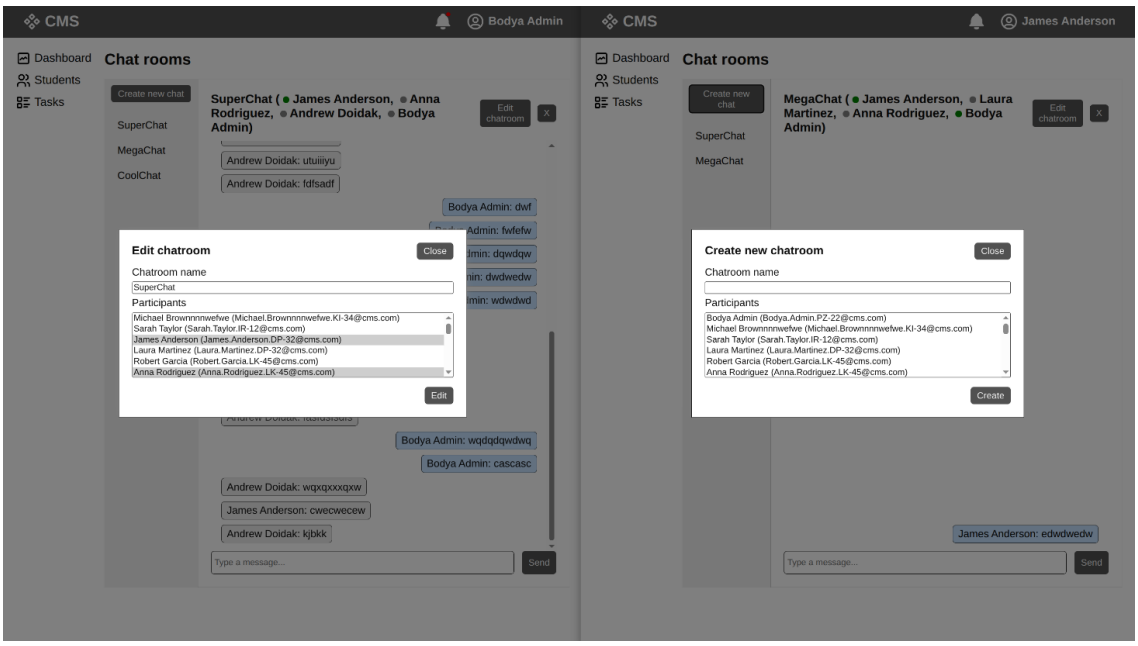


Рис. 4. Модальні вікна

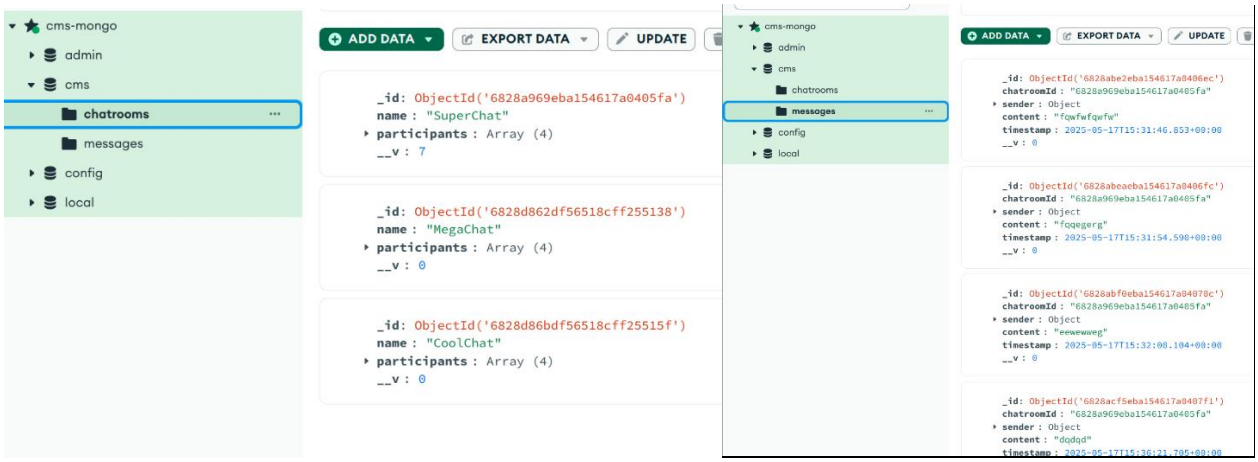


Рис. 5. База даних

Висновки (спільні 5-6 лабораторна)

Під час виконання лабораторних робіт №5 та №6 я поглибив знання з серверної веб-розробки з використанням Node.js, MongoDB та бібліотеки Socket.io для створення застосунків реального часу. Було розроблено чат-додаток, який включає клієнтську частину з формою для надсилання повідомлень та відображення чатів, а також серверну частину, що обробляє повідомлення, перевіряє їх адресатів (окремий студент, група чи всі) і зберігає дані в MongoDB. Використано Mongoose для роботи з базою даних, включаючи створення схем (повідомлення, чати), валідацію даних і зв'язки між колекціями. Для авторизації застосовано JWT, а Socket.io забезпечило миттєву доставку повідомлень і оновлення статусу учасників.

Робота розвинула навички асинхронного програмування в Node.js, роботи з NoSQL базами даних через Mongoose, реалізації реального часу комунікації та інтеграції клієнт-серверних технологій. Ці компетенції сприятимуть подальшому розвитку в створенні масштабованих веб-додатків.