НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
Кафедра інформаційних систем та технологій

Тема: Mind-mapping software

Курсова робота
З дисципліни «Технології розроблення програмного забезпечення»

Керівник
доц. Амонс О.А.
«Допущений до захисту»
_____
(Особистий підпис керівника)
« » _____2025р.
Захищений з оцінкою
_____
(оцінка)

Виконавець
ст. Хитрова Н. А
залікова книжка № гр. ІА-33
_____
(особистий підпис виконавця)
« »_____2025р.

Члени комісії:

_____ _____
(особистий підпис) (розшифровка підпису)

_____ _____
(особистий підпис) (розшифровка підпису)

Київ – 2025

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»

(назва навчального закладу)

Кафедра ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

Дисципліна «Технології розроблення програмного забезпечення»

Курс 3 Група ІА-33 Семестр 5

ЗАВДАННЯ

на курсову роботу студента

Хитрова Наталія Антонівна

(прізвище, ім’я, по батькові)

1.Тема роботи Розробка настільного застосунку “Mind-mapping software”

2. Строк здачі студентом закінченої роботи

3. Вихідні дані до роботи:

4. Зміст розрахунково – пояснювальної записки (перелік питань, що підлягають розробці)

Огляд існуючих рішень та постановка задачі. Вимоги до системи. Сценарії використання. Концептуальна модель. Проєктування бази даних та вибір СКБД. Вибір мови програмування та середовища розробки. Архітектура системи. Реалізація компонентів і структура БД. Інструкція користувача.Висновки.

Додатки:

Лістинг коду системи

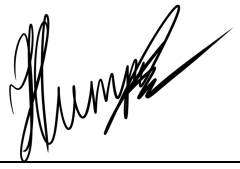
5.Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

Use-case діаграма. Use-case діаграма. UML діаграма класів. ER-діаграма бази даних. Діаграма компонентів системи. Діаграма розгортання

6. Дата видачі завдання

КАЛЕНДАРНИЙ ПЛАН

№, п/п	Назва етапів виконання курсової роботи	Строк виконання етапів роботи	Підписи або примітки
1.			
2.			
3.			
4.			
5.			
6.			
7.			
8.			
9.			
10.			
11.			
12.			
13.			
14.			
15.			
16.			
17.			
18.			

Студент  Наталія ХИТРОВА
(підпис) (Ім’я ПРІЗВИЩЕ)

Керівник_____ Олександр АМОНС
(підпис) (Ім’я ПРІЗВИЩЕ)

«____»_____ 2025 р.

ЗМІСТ

ВСТУП

У сучасному навчанні та проєктній діяльності зростає потреба в інструментах, що допомагають швидко структурувати інформацію, візуалізувати зв’язки між ідеями та спрощувати планування задач. Одним із найефективніших способів такої організації знань є ментальні карти, які дозволяють подати складну тему у вигляді наочної схеми. Тому розробка програмного застосунку для створення та редагування ментальних карт є актуальною задачею, що поєднує практичну цінність із можливістю застосувати сучасні підходи програмної інженерії.

Метою курсової роботи є проектування та реалізація десктопного застосунку Mind-mapping software для створення, редагування та збереження ментальних карт із підтримкою різних типів елементів, а також експортом результатів у зовнішні формати. Для забезпечення масштабованості, підтримуваності та розширюваності системи під час розробки було застосовано багаторівневу архітектуру, а також низку шаблонів проектування для організації взаємодії компонентів і керування поведінкою застосунку.

Об’єктом дослідження є процес розробки програмного забезпечення для візуалізації та керування структурованою інформацією. Предметом дослідження є архітектурні рішення та програмні засоби, що забезпечують реалізацію функцій редактора ментальних карт, збереження даних у базі, обробку дій користувача та інтеграцію з сервісами експорту/взаємодії по мережі.

Практична цінність роботи полягає у створенні робочого програмного продукту, який може використовуватись для навчання, планування та візуалізації ідей, а також у демонстрації застосування інженерних практик розробки (архітектура, шаблони проєктування, робота з БД та контроль версій) на реальному проєкті.

1 ПРОЄКТУВАННЯ СИСТЕМИ

1.1. Огляд існуючих рішень

Програмні засоби для створення ментальних карт (mind maps) широко використовуються для навчання, планування, брейнштормінгу та представлення знань у наочній формі. Сучасні рішення можна умовно поділити на три групи: настільні застосунки (desktop), хмарні веб-сервіси для колаборації, а також інструменти “ширшого профілю” (онлайн-дошки/планувальники), де mind map є одним із режимів роботи. Окремою тенденцією останніх років є додавання AI-функцій для генерації структури мапи з тексту або документів.

XMind є одним із найбільш популярних продуктів у цьому сегменті. До сильних сторін належать гнучкі можливості оформлення та широкий набір форматів експорту (зокрема PDF/PNG/SVG та інші), що спрощує підготовку матеріалів до друку або обміну.

MindManager орієнтований на професійне/корпоративне використання: акцент робиться на інтеграціях і обміні даними, зокрема через “outline”-експорт (наприклад OPML/RTF) та інші механізми, що дозволяє переносити структуру мапи в документи й системи керування інформацією.

Серед безкоштовних open-source рішень історично відомий FreeMind (Java-застосунок). Водночас вказують, що проєкт FreeMind не оновлювався тривалий час, а активніший розвиток отримав його форк Freeplane, який надає ширші можливості експорту (PDF, зображення, Markdown тощо).

MindMeister – один із найвідоміших онлайн-сервісів для mind mapping, орієнтований на роботу “в хмарі” та поширення результатів через експорт у популярні формати (PDF, зображення, документи тощо).
support.mindmeister.com

Coggle позиціонується як простий інструмент для швидкого створення мап і спільної роботи; також підтримує експорт у PDF/PNG та формат .mm (сумісний з деякими іншими mind-mapping застосунками).

Whimsical – приклад інструменту “дошка + діаграми”, де mind maps є частиною набору можливостей; серед ключових функцій заявлені multiplayer-редагування та експорт/копіювання як зображення (PNG).

Порівняння показує, що більшість сучасних рішень або фокусується на хмарній колаборації (спільне редагування, лінки на мапи, доступ за ролями), або розвивається як комплексні платформи (whiteboard/PM), де mind map один із модулів, або надає потужний desktop-функціонал з багатими можливостями експорту та оформлення.

Це формує базу вимог для власної системи: підтримка типових операцій редагування, робота з різними типами вузлів, експорт, збереження стану та можливість подальшого розширення.

1.2. Загальний опис проєкту

Основна ідея проекту полягає в тому, щоб надати користувачу зручний інструмент для структурування інформації у вигляді наочної схеми, де можна фіксувати ідеї, планувати навчальні або проектні задачі та швидко переглядати зв’язки між поняттями.

Застосунок орієнтований на студентів, викладачів і користувачів, яким потрібно організовувати знання або плани в візуальному форматі. Після входу в систему користувач працює зі своїм набором мап на головному екрані, може створювати нові мапи, відкривати та видаляти існуючі, змінювати назву й опис. Для зручної навігації реалізовано категорії (із можливістю задавати колір і призначати категорію мапі), позначення важливих мап, а також пошук і сортування, що спрощує роботу з великою кількістю матеріалів.

Ключовою частиною системи є редактор мапи з полотном. У ньому користувач може додавати й редагувати текстові вузли, вставляти зображення у вигляді окремих елементів мапи, переміщувати їх, змінювати розміри та видаляти. Додатково реалізовано інструменти малювання та стирання, що дає змогу робити позначки або прості ескізи прямо на мапі. Для підвищення зручності редагування передбачено механізм, який дозволяє скасовувати та повторювати ключові дії користувача.

Дані мапи та її елементи не зникають після закриття програми, оскільки застосунок використовує базу даних: інформація про користувачів, мапи, категорії та елементи зберігається в MySQL, а доступ до неї організовано через JDBC і шар репозиторіїв. Завдяки цьому при повторному відкритті мапи її вміст коректно відновлюється в редакторі.

Реалізація виконана мовою Java з використанням Swing/AWT для графічного інтерфейсу. Архітектура побудована за багаторівневим принципом, де інтерфейс користувача відокремлений від бізнес-логіки сервісів і від доступу до даних у репозиторіях. Під час розробки було застосовано шаблони проєктування, які зробили систему гнучкішою та зручнішою для розширення: Strategy використано для вибору способу експорту мапи, Command – для оформлення дій редагування та підтримки undo/redo, Abstract Factory – для створення інструментів редактора, Mediator – для організації взаємодії компонентів Dashboard без жорстких залежностей, а Visitor – для виконання операцій над елементами мапи без використання перевірок типів через instanceof. Окремо передбачено можливість експорту результату у зовнішні формати (наприклад зображення або документ), щоб користувач міг використовувати створені мапи поза межами програми.

Таким чином, результатом роботи став працездатний програмний продукт для створення та керування ментальними картами із збереженням у базі даних, розвиненим редактором, зручним керуванням мапами та архітектурою, що демонструє застосування сучасних підходів до проєктування програмних систем.

1.3. Вимоги до застосунків системи

У цьому розділі сформульовано вимоги до нашого застосунку Mind-mapping software, які визначають його основні можливості та обмеження. Вимоги поділено на функціональні, які описують, що саме повинна робити система і які сценарії має підтримувати та нефункціональні, що визначають характеристики якості, продуктивність, надійність, зручність, безпеку, обмеження середовища тощо. Сформульовані вимоги базуються на аналізі предметної області, огляді існуючих рішень і реалізованому наборі можливостей системи.

1.3.1. Функціональні вимоги до системи

FR-01. Реєстрація та авторизація. Система повинна надавати користувачу можливість зареєструватися (username/password) і виконати вхід у систему з перевіркою облікових даних у базі даних.

FR-02. Запуск і підключення до БД. Під час запуску система повинна встановлювати з’єднання з базою даних (MySQL/JDBC) та відкривати форму входу.

FR-03. Відображення списку мап користувача. Після входу система повинна відображати користувачу перелік його ментальних карт на головному екрані (Dashboard).

FR-04. Створення та відкриття мапи. Система повинна дозволяти створити нову мапу (із введенням назви) та відкривати вибрану мапу в редакторі.

FR-05. Пошук і сортування мап. Система повинна забезпечувати пошук мап за текстовим запитом.

FR-06. Робота з категоріями. Система повинна дозволяти створювати категорії, переглядати перелік категорій та призначати категорію мапі.

FR-07. Позначення мап як обраних. Система повинна дозволяти встановлювати та знімати позначку “обране/favorite” для мапи.

FR-08. Редагування метаданих мапи. Система повинна дозволяти змінювати та зберігати назву й опис мапи у відповідній панелі властивостей.

FR-09. Відновлення вмісту мапи. Під час відкриття мапи система повинна завантажувати з бази даних її елементи та збережені мазки (strokes) і відображати їх у редакторі.

FR-10. Додавання текстових вузлів і вузлів-зображень. У редакторі система повинна дозволяти додавати на полотно текстові вузли та вузли із зображенням (із вибором файлу).

FR-11. Вибір, переміщення та зміна розміру вузлів. Система повинна підтримувати виділення елемента, його переміщення мишею та зміну розміру (для елементів, де це передбачено), із фіксацією змін у даних мапи.

FR-12. Редагування тексту вузла. Система повинна дозволяти відкривати форму редагування тексту вузла і зберігати внесені зміни.

FR-13. Видалення вузлів з клавіатури. Система повинна дозволяти видалити виділений елемент мапи за допомогою клавіш Delete/Backspace.

FR-14. Undo/Redo у редакторі. Система повинна підтримувати скасування та повторення дій редагування (undo/redo) щонайменше для операцій з елементами (переміщення/зміна розміру/видалення) та для дій із мазками.

FR-15. Малювання та стирання на полотні. Система повинна надавати інструменти для малювання мазків, вибору параметрів малювання (колір, товщина) та стирання мазків ластиком, із подальшим збереженням та відновленням цих даних.

FR-16. Експорт мапи. Система повинна забезпечувати експорт мапи у формати PNG/JPG, а також формування текстового звіту/статистики у TXT.

1.3.2. Нефункціональні вимоги до системи

NFR-01. Зручність та зрозумілість інтерфейсу. Інтерфейс має бути інтуїтивним: основні дії повинні виконуватися без складних налаштувань, із логічним розташуванням елементів керування та передбачуваною поведінкою.

NFR-02. Реактивність інтерфейсу. Застосунок повинен залишатися швидким під час роботи: переміщення вузлів, масштабування та малювання на полотні мають виконуватись без помітних затримок у типових сценаріях використання.

NFR-03. Надійність та стабільність. Система не повинна аварійно завершувати роботу при некоректних діях користувача або помилках введення. У разі проблем з доступом до БД або файлів система має повідомляти про помилку зрозумілим повідомленням і, по можливості, дозволяти продовжити роботу.

NFR-04. Цілісність і збереження даних. Дані мапи, категорій і елементів повинні зберігатися так, щоб не виникало частково записаних або суперечливих станів (наприклад, елемент не може “існувати” без мапи). При повторному відкритті мапи відновлений стан має відповідати останньому збереженню.

NFR-05. Безпека облікових даних. Паролі користувачів не повинні зберігатися у відкритому вигляді; система має використовувати безпечний підхід до збереження облікових даних (хешування) та не розкривати конфіденційні дані через повідомлення про помилки.

NFR-06. Розмежування доступу. Користувач повинен мати доступ лише до власних мап і категорій. Дані різних користувачів мають бути логічно ізольовані на рівні операцій читання/запису.

NFR-07. Портативність. Застосунок має запускатися на типових настільних ОС (насамперед Windows; бажано без прив’язки до конкретної версії), за наявності встановленого Java Runtime Environment та доступного підключення до БД.

NFR-08. Розширюваність. Архітектура має дозволяти додавання нових інструментів редактора, нових форматів експорту та нових типів елементів мапи з мінімальними змінами існуючих модулів (переважно шляхом додавання нових класів/реалізацій, а не переписування наявного коду).

1.4. Сценарії використання системи

Сценарії використання відображають типові дії, які виконують актори системи і дозволяють формалізувати поведінку системи з точки зору кінцевого користувача.

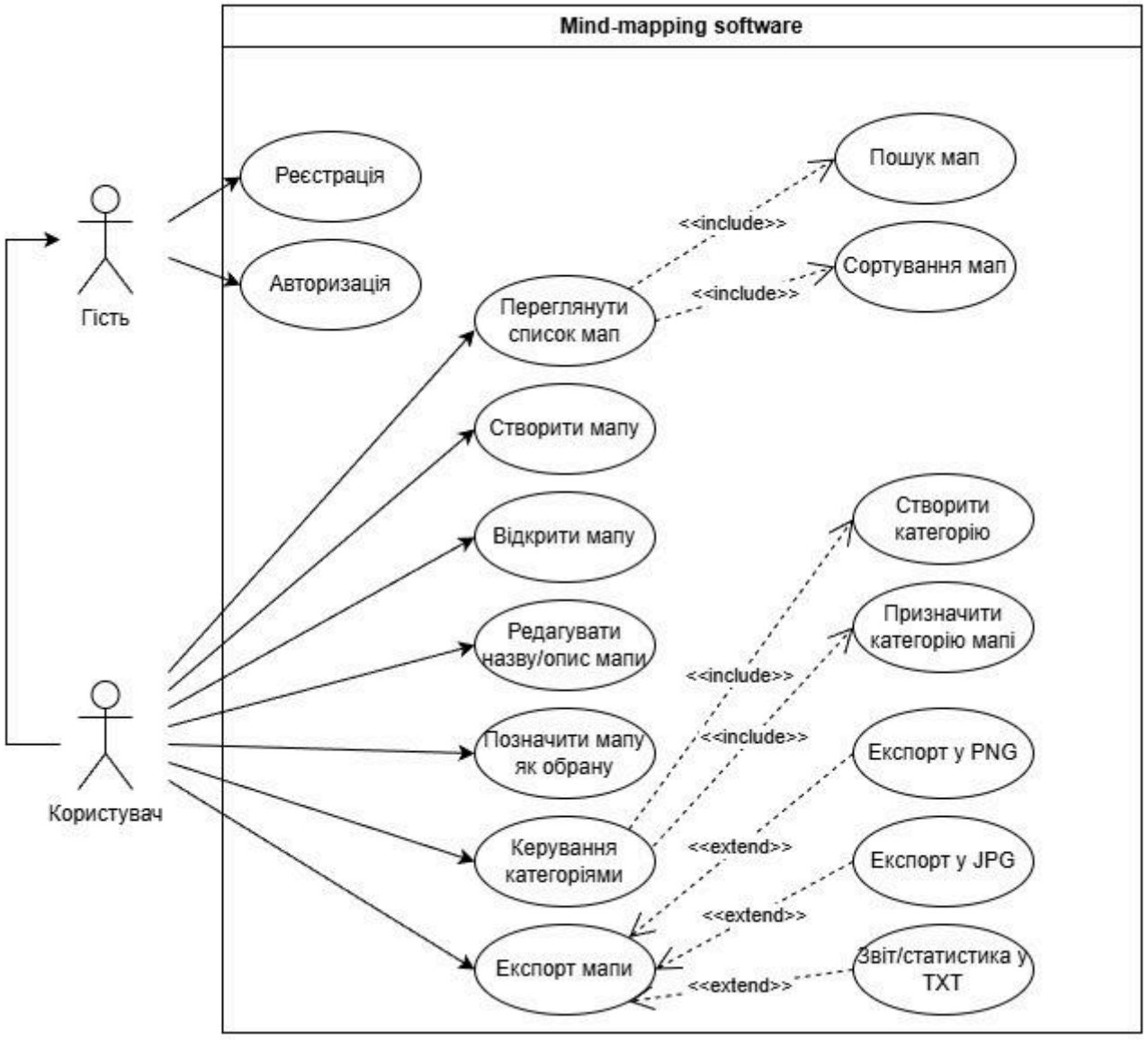


Рис. 1.4.1 Use-case діаграма підсистеми керування мапами

На рис. 1.4.1 наведено use-case діаграму підсистеми керування мапами (Dashboard) застосунку Mind-mapping software. Діаграма відображає основні сценарії взаємодії користувача із системою на етапі входу та роботи зі списком ментальних карт, а також допоміжні операції пошуку, сортування, категоризації та експорту.

У системі виділено два актори: Гість і Користувач. Актор Гість має доступ лише до базових сценаріїв початку роботи із застосунком, це реєстрація та авторизація. Реєстрація передбачає створення нового облікового запису, а

авторизація вхід у систему з подальшим переходом до головного екрана керування мапами.

Після успішного входу актор Користувач отримує доступ до основних функцій Dashboard. Центральним сценарієм є перегляд списку мап, який дозволяє ознайомитися з усіма ментальними картами, що належать користувачу. Цей сценарій деталізується двома включеними можливостями: пошук мап та сортування мап. Під час роботи зі списком користувач може застосовувати пошук і сортування як невід'ємні інструменти навігації, що особливо важливо при великій кількості збережених мап. Користувач також може створити мапу. Ініціювати додавання нової ментальної карти (зазвичай із введенням назви), а також відкрити мапу для переходу до редактора та подальшого редагування її вмісту. Окремим сценарієм передбачено редагування назви/опису мапи, що забезпечує підтримку актуальності метаданих та кращу організацію матеріалів. Для швидкого доступу до важливих матеріалів реалізовано сценарій позначити мапу як обрану. Це дозволяє користувачу виділяти пріоритетні мапи без зміни їх структури чи вмісту. Ще одним важливим напрямом є керування категоріями, яке використовується для класифікації мап. Даний сценарій включає дві функції: створити категорію та призначити категорію мапі. Таким чином користувач може формувати власну систему групування мап, а також швидко змінювати приналежність мап до категорій під час роботи. Окрім редагування та організації, система надає можливість експорту мапи у зовнішні формати. Базовий сценарій "Експорт мапи" розширюється конкретними варіантами експорту: експорт у PNG, експорт у JPG та формування звіту/статистики у TXT.

Отже, перша use-case діаграма описує повний набір сценаріїв роботи на рівні Dashboard: від входу в систему до створення, пошуку, сортування, організації ментальних карт за категоріями та експорту результатів у зручні формати.

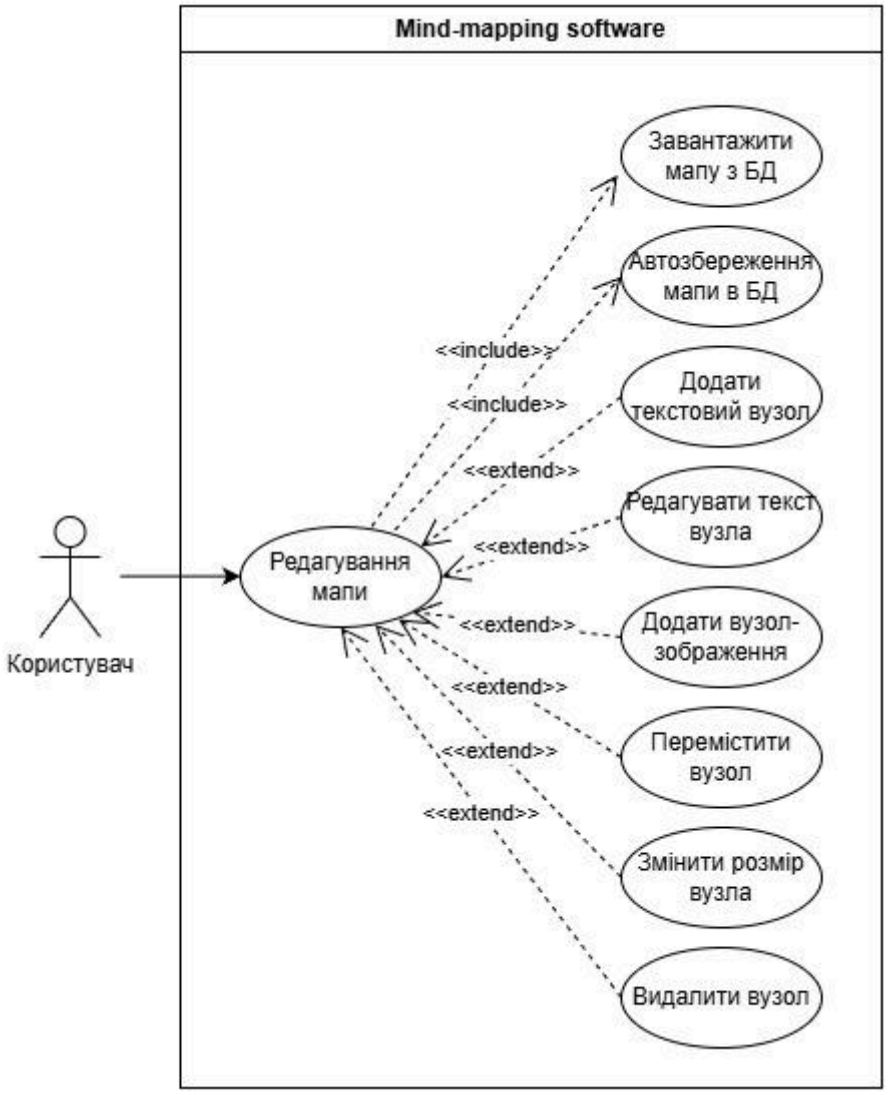


Рис. 1.4.2 Use-case діаграма підсистеми редактора ментальної карти

На рис. 1.4.2 наведено use-case діаграму підсистеми редактора ментальної карти (Canvas) застосунку Mind-mapping software. Вона відображає сценарії роботи користувача безпосередньо з вмістом мапи на полотні: завантаження даних, виконання операцій редагування та збереження результатів.

Єдиним актором на цій діаграмі є Користувач, оскільки доступ до редагування мап можливий лише після успішної авторизації. Центральним сценарієм виступає “Редагування мапи”, який описує процес внесення змін до ментальної карти в редакторі. Перед початком роботи система повинна відобразити актуальний стан мапи, тому в межах редагування виконується “Завантажити мапу з БД”. Цей варіант

використання позначений як включений, оскільки завантаження є обов’язковим етапом для відкриття мапи та коректного відображення її елементів у редакторі. У процесі роботи користувач виконує операції редагування, які реалізовані як розширення базового сценарію “Редагування мапи”. Це означає, що під час редагування користувач може за потреби обирати одну або кілька дій залежно від ситуації. Зокрема, система підтримує додавання текстового вузла, що дозволяє створювати нові елементи з текстовою інформацією, а також редагування тексту вузла для внесення змін у вже створені текстові елементи. Окрім текстових вузлів, реалізовано сценарій “Додати вузол-зображення”, який забезпечує вставку зображення як окремого елемента мапи (через вибір файлу). Для зміни структури мапи передбачено можливість переміщення вузла на полотні, що дозволяє змінювати розташування елементів і тим самим покращувати візуальну читабельність карти. Також користувач може змінювати розмір вузла, адаптуючи відображення елементів відповідно до обсягу інформації або зручності перегляду. У разі потреби система надає сценарій “Видалити вузол”, який дозволяє прибрати непотрібні елементи з мапи. Окремо на діаграмі відображено механізм збереження змін “Автозбереження мапи в БД”. Використання саме автозбереження означає, що після виконання ключових дій редагування система фіксує оновлений стан мапи в базі даних без необхідності ручного підтвердження користувачем. Це підвищує надійність роботи та зменшує ризик втрати даних у разі непередбачених ситуацій (наприклад, закриття програми).

Таким чином, друга use-case діаграма описує основний робочий цикл редактора: завантаження мапи, виконання набору операцій редагування (додавання/зміна/переміщення/видалення елементів) та збереження актуального стану мапи в базі даних, що забезпечує відновлення внесених змін при повторному відкритті.

Розробка сценаріїв варіантів використання:

Перед наведеними нижче сценаріями використання подано їх детальний опис у текстовому форматі. Кожен сценарій оформлено у вигляді специфікації use-case та містить: ідентифікатор і назву, акторів, передумови, основний потік подій, альтернативні/виняткові потоки та постумови. Такий формат дозволяє однозначно зафіксувати поведінку системи з точки зору користувача, перевірити повноту та несуперечливість функціональних вимог, а також використати сценарії як основу для подальшого тестування реалізованого застосунку.

UC-01 Авторизація користувача

ID: UC-01

Назва: Авторизація.

Актор: Гість / Зареєстрований користувач.

Передумова: Користувач має обліковий запис у системі.

Основний потік подій:

1. Користувач відкриває застосунок і бачить форму входу.
2. Користувач вводить логін та пароль.
3. Користувач натискає кнопку “Увійти”.
4. Система перевіряє облікові дані в БД.
5. У разі успіху система відкриває головний екран (Dashboard) зі списком мап користувача.

   Альтернативний потік (A1): Невірні облікові дані.

   3а. Користувач вводить неправильний логін або пароль.

   4а. Система відображає повідомлення про помилку та пропонує повторити введення.

   Альтернативний потік (A2): Помилка підключення до БД.

   4б. Система не може виконати запит до БД.

   4в. Система повідомляє про проблему з’єднання і не виконує вхід.

   Постумова: У разі успіху користувач авторизований і перебуває на Dashboard.

UC-02 Створення нової ментальної мапи

ID: UC-02

Назва: Створення нової мапи.

Актор: Зареєстрований користувач.

Передумова: Користувач авторизований і знаходиться на Dashboard.

Основний потік подій:

1. Користувач натискає кнопку “Створити мапу”.
2. Система відображає діалог для введення назви мапи.
3. Користувач вводить назву та підтверджує створення.
4. Система створює мапу в БД.
5. Система відкриває редактор створеної мапи.

   Альтернативний потік (А1): Назву не введено.

   3а. Користувач залишає назву порожньою та підтверджує створення.

   3б. Система повідомляє, що назва є обов’язковою, і повертає до введення.

   Постумова: Створено нову мапу та відкрито режим редагування.

UC-03 Перегляд списку мап, пошук і сортування

ID: UC-03

Назва: Перегляд списку мап.

Актор: Зареєстрований користувач.

Передумова: Користувач авторизований і знаходиться на Dashboard.

Основний потік подій:

1. Система відображає список мап користувача.
2. Користувач вводить рядок у поле пошуку (за потреби).
3. Система фільтрує список мап відповідно до запиту.
4. Користувач обирає режим сортування (за потреби).
5. Система відображає список, відсортований за обраним критерієм.

Альтернативний потік (А1): Нічого не знайдено.

3а. Після введення запиту список порожній.

3б. Система показує порожній результат/повідомлення та дозволяє змінити запит.

Постумова: Користувач бачить актуальний список мап згідно пошуку/сортування.

UC-04 Редагування назви та опису мапи

ID: UC-04

Назва: Редагування назви/опису мапи.

Актор: Зареєстрований користувач.

Передумова: Користувач обрав мапу на Dashboard.

Основний потік подій:

1. Користувач змінює назву та/або опис мапи у панелі властивостей.
2. Користувач натискає “Зберегти”.
3. Система зберігає оновлені метадані в БД.
4. Система оновлює відображення мапи у списку.

Альтернативний потік (А1): Некоректні дані.

2а. Користувач залишив назву порожньою.

2б. Система повідомляє про помилку та не виконує збереження.

Постумова: Метадані мапи оновлені та збережені.

UC-05 Керування категоріями

ID: UC-05

Назва: Створення категорії та призначення категорії мапі.

Актор: Зареєстрований користувач.

Передумова: Користувач авторизований і знаходиться на Dashboard.

Основний потік подій:

1. Користувач відкриває керування категоріями та натискає “Створити категорію”.
2. Система відображає форму введення назви та вибору кольору.
3. Користувач вводить назву категорії, обирає колір і підтверджує.
4. Система створює категорію в БД та оновлює список категорій.
5. Користувач обирає мапу та призначає їй створену категорію.
6. Система зберігає прив’язку мапи до категорії і оновлює відображення.

   Альтернативний потік (А1): Назву категорії не введено.

   3а. Користувач залишає назву порожньою.

   3б. Система повідомляє про помилку та просить виправити дані.

   Постумова: Категорію створено, мапу прив’язано до категорії.

UC-06 Позначення мапи як обраної

ID: UC-06

Назва: Позначити мапу як обрану.

Актор: Зареєстрований користувач.

Передумова: Користувач авторизований і вибрав мапу на Dashboard.

Основний потік подій:

1. Користувач встановлює/знімає позначку “обране” (зірочка).
2. Система зберігає новий стан “favorite” в БД.
3. Система оновлює відображення мапи у списку/панелі властивостей.

   Постумова: Мапа має актуальний статус “обране”.

UC-07 Відкриття мапи та завантаження даних у редактор

ID: UC-07

Назва: Відкрити мапу (завантаження з БД).

Актор: Зареєстрований користувач.

Передумова: Користувач знаходиться на Dashboard та має хоча б одну мапу.

Основний потік подій:

1. Користувач обирає мапу і виконує команду “Відкрити”.
2. Система завантажує з БД елементи мапи (вузли, зображення) та інші збережені дані.
3. Система відкриває редактор і відображає мапу на полотні.

   Альтернативний потік (А1): Мапу не вдалося завантажити.

   2а. Виникає помилка доступу до БД або дані пошкоджені.

   2б. Система повідомляє про помилку та не відкриває редактор.

   Постумова: Відкрито редактор із завантаженим станом мапи.

UC-08 Додавання та редагування текстового вузла

ID: UC-08

Назва: Додати текстовий вузол і змінити його текст.

Актор: Зареєстрований користувач.

Передумова: Відкрита мапа в редакторі.

Основний потік подій:

1. Користувач обирає дію “Додати текстовий вузол”.
2. Користувач клацає на полотні в місці додавання.
3. Система створює текстовий вузол та відображає його на полотні.
4. Користувач відкриває редагування тексту вузла (подвійним кліком) і вводить текст.
5. Користувач підтверджує редагування.
6. Система оновлює вузол і виконує автозбереження змін у БД.

   Альтернативний потік (А1): Скасування редагування.

   5а. Користувач закриває вікно редагування без підтвердження.

   5б. Система не змінює текст вузла.

   Постумова: Доданий/оновлений текстовий вузол збережено у складі мапи.

UC-09 Додавання вузла-зображення

ID: UC-09

Назва: Додати вузол-зображення.

Актор: Зареєстрований користувач.

Передумова: Відкрита мапа в редакторі.

Основний потік подій:

1. Користувач обирає інструмент “Додати зображення”.
2. Система відкриває діалог вибору файлу.
3. Користувач обирає зображення (PNG/JPG).
4. Система перевіряє формат і створює вузол типу ImageNode.
5. Система відображає зображення на полотні; користувач може перемістити його у потрібне місце.
6. Система зберігає зміни (автозбереження) як частину мапи.

   Альтернативний потік (A1): Непідтримуваний формат.

   3a. Користувач обирає файл іншого формату (не PNG/JPG).

   4a. Система повідомляє про помилку формату та не додає вузол.

   Постумова: Зображення додане до мапи та збережене в системі.

UC-10 Переміщення, зміна розміру та видалення вузла

ID: UC-10

Назва: Змінити розташування/розмір вузла або видалити вузол.

Актор: Зареєстрований користувач.

Передумова: Відкрита мапа в редакторі, на полотні є хоча б один вузол.

Основний потік подій (переміщення):

1. Користувач виділяє вузол.
2. Користувач перетягує вузол у нову позицію.
3. Система оновлює координати вузла та виконує автозбереження в БД.

Основний потік подій (зміна розміру):

4. Користувач змінює розмір вузла (як це передбачено інтерфейсом).
5. Система зберігає нові параметри розміру в БД.

Основний потік подій (видалення):

6. Користувач виділяє вузол і натискає Delete/Backspace.
7. Система видаляє вузол із мапи та зберігає зміни.

   Альтернативний потік (A1): Вузол не вибраний.

   1а. Користувач натискає Delete без виділення вузла.

   1б. Система не виконує дію.

   Постумова: Структура мапи оновлена; зміни збережені.

UC-11 Експорт мапи

ID: UC-11

Назва: Експорт мапи у файл (PNG/JPG/TXT).

Актор: Зареєстрований користувач.

Передумова: Користувач має мапу та бажає отримати її копію у файлі.

Основний потік подій:

1. Користувач обирає дію “Експорт мапи”.
2. Система пропонує доступні варіанти експорту (PNG/JPG або звіт/статистика TXT).
3. Користувач обирає формат і місце збереження.
4. Система формує файл експорту та зберігає його на пристрої.
5. Система повідомляє про успішний експорт.

   Альтернативний потік (A1): Помилка збереження файлу.

   4а. Немає доступу до папки або файл не може бути записаний.

   4б. Система повідомляє про помилку та пропонує обрати інше місце/ім’я файлу.

   Постумова: Створено експортований файл мапи у вибраному форматі.

1.5. Концептуальна модель системи

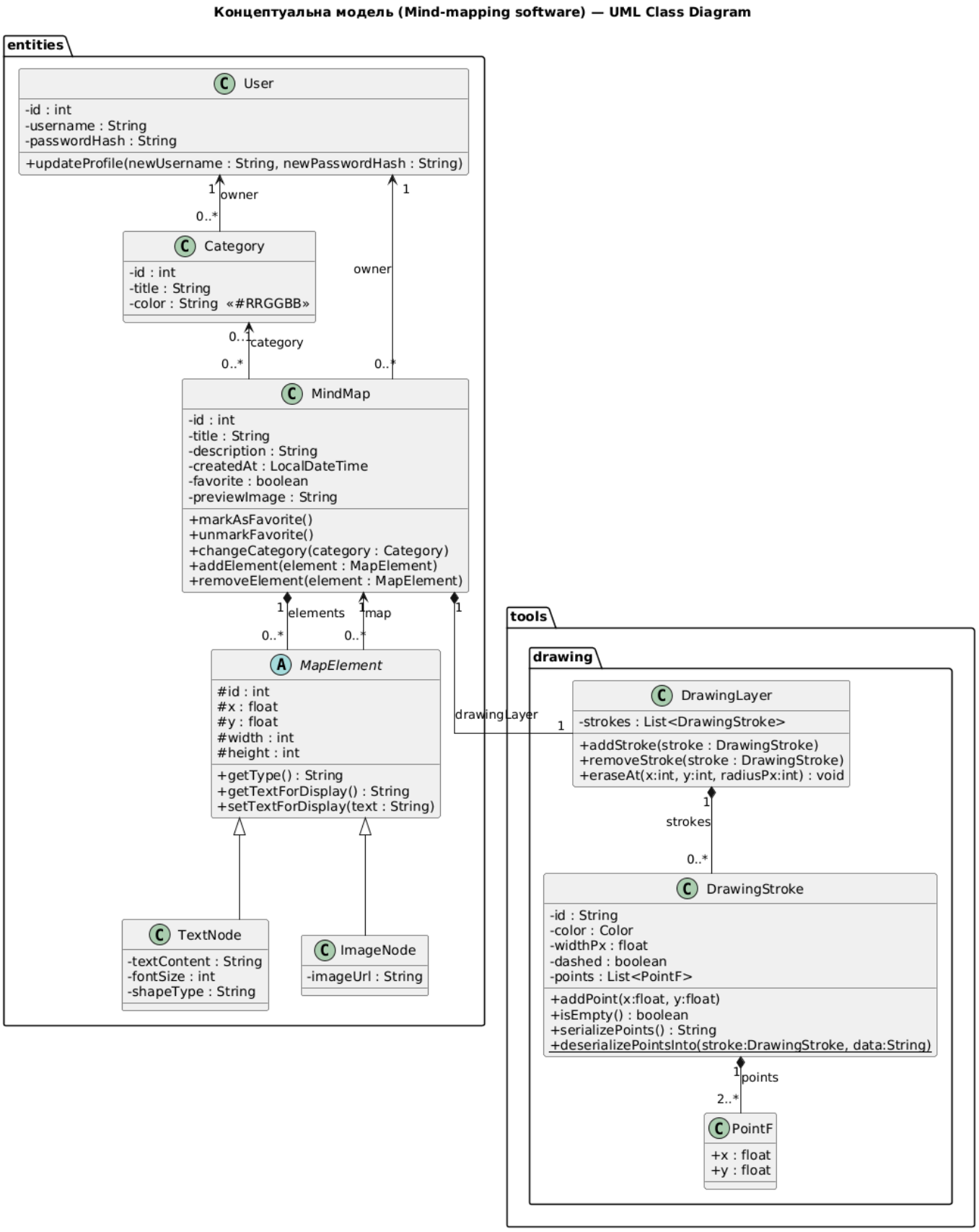


Рис. 1.5.1 Діаграма класів концептуальної моделі застосунку

На рисунку показано діаграму класів концептуальної моделі застосунку. Вона відображає, з яких основних об’єктів складається система та як вони пов’язані між собою під час роботи користувача з ментальними картами.

У центрі моделі знаходиться клас MindMap – це головний об’єкт, який представляє одну ментальну карту. У ньому зберігаються основні дані про карту: назва, опис, дата створення, ознака “обране”, а також допоміжні поля на кшталт прев’ю. MindMap також містить операції, які відповідають реальним діям користувача: позначити мапу як улюблену, змінити категорію, додати або прибрати елемент. Кожна мапа прив’язана до свого власника, класу User. User описує обліковий запис: ім’я користувача та дані для входу (замість збереження пароля у відкритому вигляді використовується хеш). Саме через користувача система “розуміє”, кому належать мапи і які дані потрібно показувати після авторизації. Щоб упорядкувати мапи на Dashboard, використовується клас Category. Категорія має назву та колір і теж належить користувачу. Коли користувач вибирає категорію для мапи, у MindMap просто зберігається посилання на цю Category. Якщо категорія не вибрана, то мапа залишається без категорії, але продовжує існувати й відображатися в списку. Вміст ментальної карти формується елементами на полотні редактора. Для цього в моделі є абстрактний клас MapElement, який задає спільну “основу” для всіх елементів: позицію на полотні та розміри. Це потрібно, щоб будь-який елемент можна було намалювати на екрані, перемістити або змінити розмір однаковим способом. Від MapElement наслідуються конкретні типи вузлів. TextNode використовується для текстових елементів: він містить текст і параметри його відображення. ImageNode представляє вузол із зображенням, у ньому зберігається шлях/посилання на файл зображення, який показується на полотні як окремий елемент мапи. Всі елементи належать конкретній мапі: тобто кожна MindMap зберігає список своїх MapElement, і при відкритті редактора цей список завантажується й відображається користувачу. Окремо в моделі показано підсистему малювання (пакет tools/drawing). Вона потрібна для того, щоб користувач міг робити вільні позначки прямо на мапі, як ручкою на дошці. За це відповідає DrawingLayer –

шар малювання, який “прикріплений” до конкретної мапи. DrawingLayer зберігає набір штрихів (DrawingStroke) і має методи для додавання штриха, видалення або стирання. Кожен штрих містить параметри лінії (колір, товщина, чи пунктирна) і набір точок PointF, з яких фактично і складається намальована траєкторія. Завдяки цьому при повторному відкритті мапи система може не тільки відновити вузли, а й відтворити все намальоване на полотні.

1.6. Вибір бази даних

Для роботи застосунку потрібне сховище, яке забезпечує надійне збереження даних користувачів, ментальних карт, категорій та елементів мапи між запусками програми. Оскільки дані мають чіткі зв’язки (наприклад, мапа належить користувачу, елементи належать мапі, мапа може мати категорію), доцільно використовувати реляційну СКБД із підтримкою обмежень цілісності та транзакцій.

У проекті обрано MySQL, оскільки вона добре підходить для навчального десктопного застосунку та легко інтегрується з Java. Для Java існує офіційний JDBC-драйвер MySQL Connector/J, який призначений саме для взаємодії застосунків з MySQL і документований виробником. Це спрощує реалізацію доступу до даних через стандартний підхід JDBC без додаткових платформних залежностей. Важливою причиною вибору MySQL також є підтримка механізмів цілісності даних. Зокрема, рушій InnoDB підтримує FOREIGN KEY-обмеження, що дозволяє контролювати коректність зв’язків між таблицями під час вставки/оновлення/видалення даних і запобігати появі “осиротілих” записів.

Як альтернативу можна розглядати SQLite. Її сильна сторона – це простота розгортання, оскільки SQLite є “serverless” і “zero-configuration” рішенням: не потребує окремого серверного процесу, встановлення та адміністрування. Проте для нашого проекту, де дані зберігаються як окрема база з користувачами та зв’язаними сутностями, практичніше використати повноцінну серверну СКБД (MySQL), яка типовим чином працює як окремий сервіс і добре підходить для структурованих даних із зв’язками. Ще однією поширеною альтернативою є PostgreSQL, яка відома

надійністю, цілісністю даних, багатим набором можливостей і розширюваністю. Однак для навчального десктопного проккту, де потрібні базові можливості збереження та вибірки, MySQL є простішою в налаштуванні у типовому середовищі та добре інтегрується через офіційний JDBC-драйвер, тому її вибір є виправданим.

1.7. Вибір мови програмування та середовища розробки

Для реалізації застосунку було обрано мову програмування Java, оскільки вона добре підходить для розробки настільних застосунків і дозволяє запускати програму на різних операційних системах завдяки виконанню коду на JVM.

Окрім портативності, Java має розвинену стандартну бібліотеку та екосистему, що дає змогу будувати як графічний інтерфейс, так і роботу з базою даних у межах одного технологічного стеку. Для створення графічного інтерфейсу буде використано Swing (частина Java Foundation Classes), який надає набір компонентів для побудови GUI та підтримує роботу з графікою, текстом і зображеннями, що є важливим для редактора ментальних карт із полотном. Завдяки цьому інтерфейс застосунку реалізується стандартними засобами Java без залежності від сторонніх UI-фреймворків.

Для доступу до даних використано JDBC (Java Database Connectivity), оскільки цей API забезпечує універсальний стандартний спосіб підключення Java-застосунків до реляційних баз даних і виконання SQL-запитів. Це дозволяє реалізувати збереження користувачів, мап, категорій та елементів мапи у MySQL, а також виконувати CRUD-операції через типові механізми java.sql.

Як середовище розробки було обрано IntelliJ IDEA, тому що це IDE, орієнтована на професійну розробку на Java і має сильну підтримку продуктивної роботи з кодом: підказки, статичний аналіз і вбудовані рефакторинги. Окремо IntelliJ IDEA підтримує типові інструменти збирання (наприклад, Maven/Gradle), що спрощує керування залежностями й запуск конфігурацій у майбутньому, якщо проєкт буде розширюватися

1.8. Проєктування розгортання системи

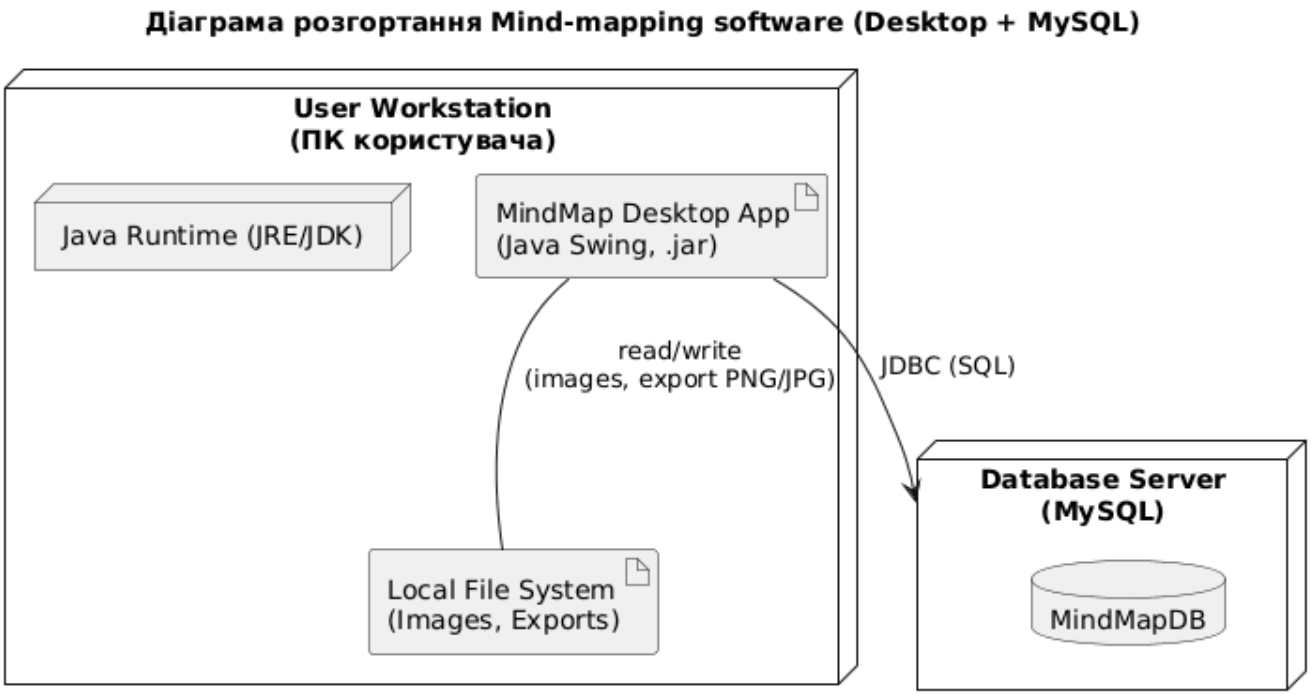


Рис. 1.8.1 Діаграма розгортання системи

На рис. 1.8.1 наведено діаграму розгортання системи Mind-mapping software у базовому варіанті Desktop + MySQL, який використовується як основний для роботи застосунку. Діаграма показує, на яких вузлах розміщуються компоненти системи та якими каналами вони взаємодіють під час виконання функцій.

Основним вузлом є робоча станція користувача (User Workstation), на якій запускається десктопний застосунок у вигляді Java-програми. Для виконання програми на цьому ж комп’ютері повинно бути встановлене середовище виконання Java Runtime, яке забезпечує запуск і роботу графічного інтерфейсу редактора ментальних карт.

Окрім виконання основної логіки, застосунок взаємодіє з локальною файловою системою. Це використовується у двох типових сценаріях: по-перше, під час додавання зображень у мапу (читання файлів з диска), а по-друге, під час експорту мапи у зовнішні формати (збереження згенерованих файлів PNG/JPG і текстового звіту/статистики). Таким чином, файлове сховище є необхідним

компонентом середовища, у якому працює застосунок, оскільки забезпечує обмін даними з файлами користувача.

Зберігання структурованих даних виконується на окремому вузлі Database Server (MySQL), де розміщена база даних MindMapDB. У ній зберігаються облікові записи користувачів, ментальні карти, категорії, вузли та дані, необхідні для відновлення стану мапи під час повторного відкриття.

Взаємодія між десктопним застосунком і сервером бази даних відбувається через JDBC (SQL): застосунок формує SQL-запити для читання та запису даних, а MySQL повертає результати або підтверджує виконання змін. Така схема розгортання є простою, зрозумілою та достатньою для основного сценарію використання: користувач запускає програму на власному ПК, працює з мапами в інтерфейсі, а всі зміни надійно зберігаються в MySQL і можуть бути відновлені при наступному запуску.

2 РЕАЛІЗАЦІЯ КОМПОНЕНТІВ СИСТЕМИ

2.1. Структура бази даних

Для збереження даних застосунку використовується база mindmapdb, в якій зберігаються облікові записи користувачів, метадані мап, їхні елементи (вузли) та результати малювання. Структура БД проектувалася так, щоб усі дані конкретного користувача були ізольовані та легко завантажувалися при вході в систему, а видалення мапи автоматично очищало всі пов’язані з нею записи.

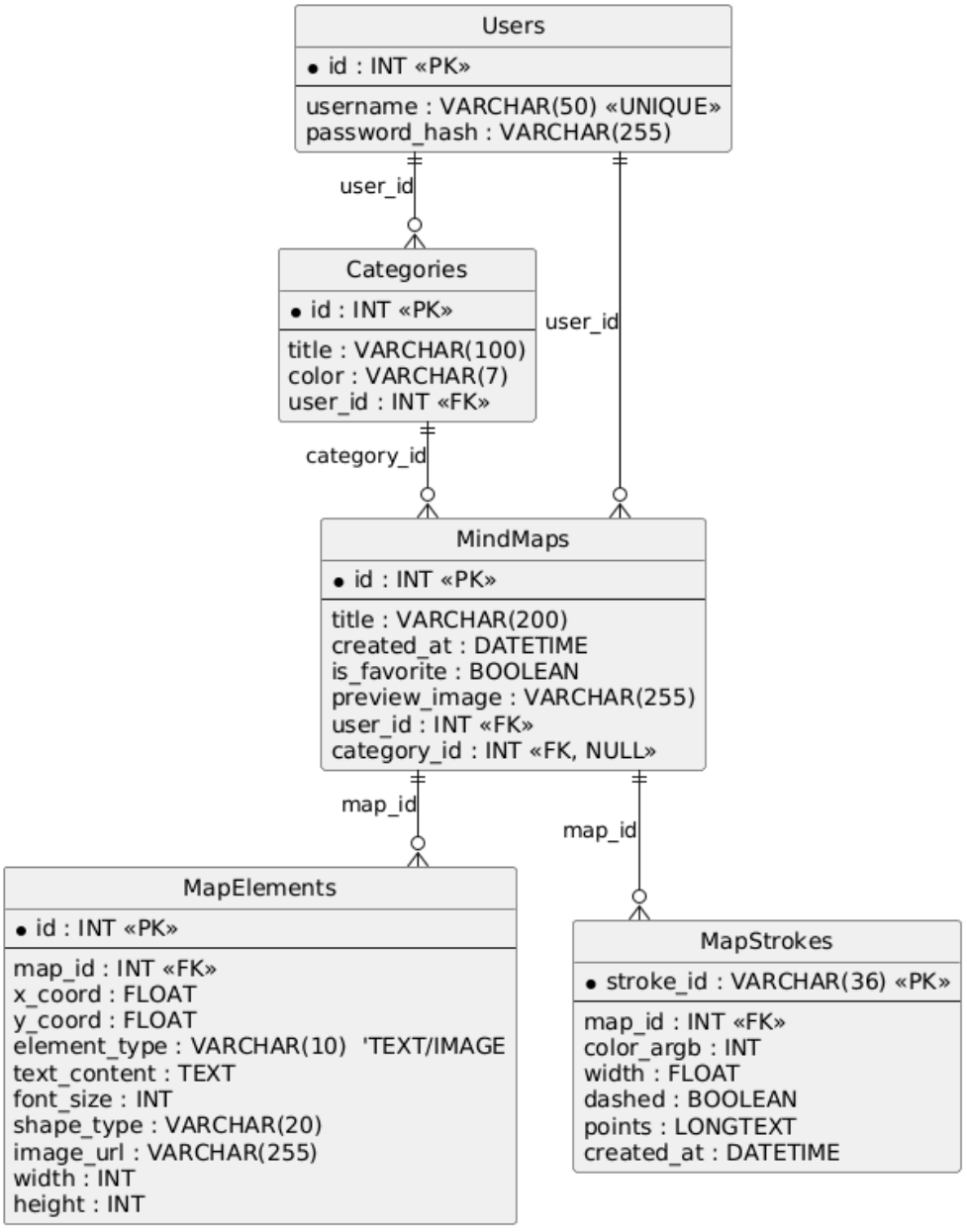


Рис. 2.1.1 ER-діаграма структури бази даних

Базовою сутністю є таблиця Users, у якій зберігаються облікові записи. Вона містить унікальне ім’я користувача (username) та хеш пароля (password_hash), що використовується для авторизації без збереження пароля у відкритому вигляді. Кожен користувач може створювати власні мапи та категорії, тому інші таблиці мають зовнішні ключі, які посилаються на Users.id.

Таблиця Categories використовується для класифікації мап. Вона містить назву категорії (title), її колір (color) та посилання на власника (user_id). Таким чином, категорії є персональними для користувача і не змішуються між різними акаунтами.

Таблиця MindMaps зберігає метадані мапи: назву (title), дату створення (created_at), ознаку “обране” (is_favorite) та шлях/посилання на прев’ю (preview_image). Кожна мапа належить конкретному користувачу через поле user_id. Зв’язок із категорією реалізований полем category_id, яке може бути порожнім (NULL), якщо категорія для мапи не вибрана. Це дозволяє користувачу працювати з мапами навіть без категоризації.

Вміст мапи зберігається у таблиці MapElements. Кожен запис відповідає одному елементу на полотні редактора і містить координати розміщення (x_coord, y_coord) та параметри відображення. Поле element_type визначає тип елемента (текстовий або зображення), а відповідні дані зберігаються у полях text_content, font_size, shape_type (для текстового вузла) або image_url (для вузла-зображення). Поля width і height дають змогу відновлювати розміри елемента після повторного відкриття мапи.

Окремо реалізовано збереження малювання – таблиця MapStrokes містить штрихи, намальовані користувачем у редакторі. Штрих прив’язаний до мапи через map_id і містить параметри лінії: колір (color_argb), товщину (width), тип лінії (dashed) та набір точок траєкторії (points), збережений у вигляді тексту. Це забезпечує можливість відновлювати ручні позначки разом із вузлами мапи.

SQL-скрипт для створення структури бази даних наведено в Додатку.

2.2. Архітектура системи

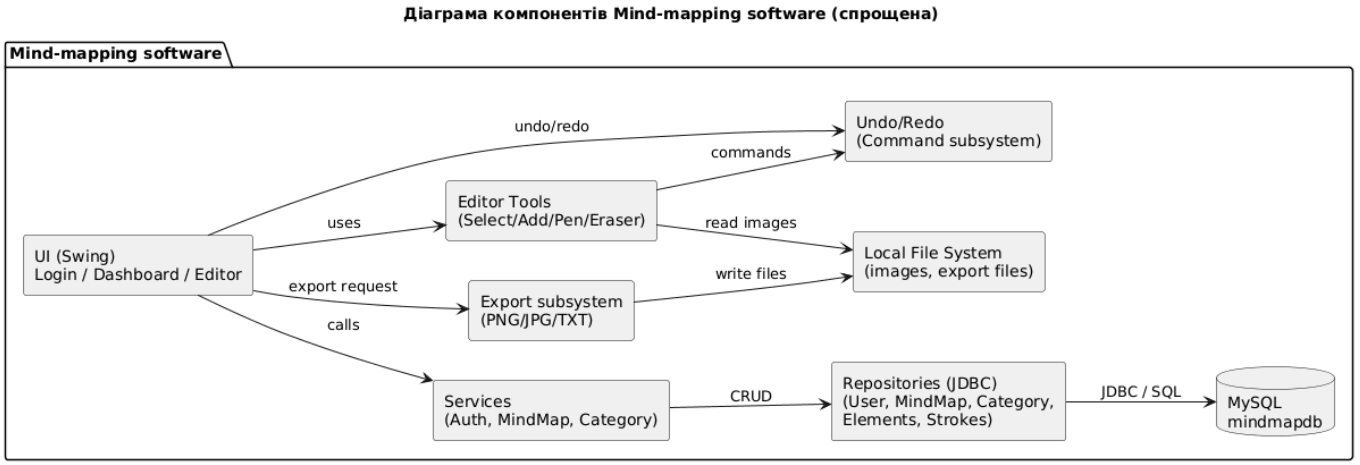


Рис. 2.2 Діаграма компонентів системи

Система побудована за принципом розподілу відповідальності: інтерфейс користувача ініціює дії, сервісний рівень реалізує бізнес-логіку, репозиторії відповідають за доступ до даних, а окремі підсистеми (інструменти, undo/redo, експорт) забезпечують розширюваність і зручність роботи з редактором ментальних карт.

На рис. 2.2 наведено спрощену діаграму компонентів системи. Вона демонструє основні модулі застосунку та взаємодію між ними під час виконання ключових функцій: авторизації, роботи зі списком мап, редагування мапи в редакторі, збереження даних та експорту.

Центральним компонентом є UI (Swing), який містить форми входу, головний екран (Dashboard) і редактор мапи. Інтерфейс приймає дії користувача (кнопки, меню, події миші/клавіатури) та ініціює відповідні операції в інших компонентах системи. Для виконання бізнес-логіки UI звертається до компонента Services, де зосереджені сервіси авторизації та керування мапами й категоріями. Саме сервісний рівень координує сценарії: створення/відкриття мапи, оновлення метаданих, завантаження елементів і збереження змін.

Доступ до бази даних винесений в окремий компонент Repositories (JDBC). Репозиторії інкапсулюють SQL-запити та операції CRUD для сутностей користувача, мап, категорій, елементів і штрихів малювання. Компонент репозиторіїв взаємодіє з

MySQL mindmapdb через JDBC-з’єднання, що забезпечує збереження даних між запусками застосунку та відновлення стану мап при повторному відкритті.

Функціональність редактора розширюється компонентом Editor Tools, який об’єднує інструменти взаємодії з полотном (вибір/додавання вузлів, малювання, ластик). Дії, які мають підтримку скасування та повторення, реалізуються через компонент Undo/Redo (Command subsystem): інструменти формують команди, які зберігаються в менеджері команд і можуть бути виконані у зворотному порядку при undo або повторно при redo. Це робить поведінку редактора передбачуваною та зручною для користувача.

Окремо виділено компонент Export subsystem, який відповідає за формування вихідних файлів мапи у потрібному форматі (PNG/JPG/TXT). Під час експорту або додавання зображень система взаємодіє з локальною файловою системою: інструменти зчитують файли зображень, а підсистема експорту записує згенеровані результати у вибране користувачем місце збереження.

2.2.1. Специфікація системи

Архітектура побудована за принципом розподілу відповідальності між основними рівнями: інтерфейс користувача, бізнес-логіка та доступ до даних. Такий підхід спрощує супровід і розвиток проєкту: інтерфейс не містить SQL-логіки, а робота з базою даних не залежить від деталей GUI.

Інтерфейсний рівень (UI) включає форми авторизації, головний екран керування мапами (Dashboard) та редактор мапи. UI приймає події користувача (кнопки, меню, події миші/клавіатури), відображає поточний стан даних та ініціює виконання операцій у сервісному рівні. У редакторі мапи UI відповідає за взаємодію з полотном: вибір вузлів, їх переміщення/зміна розміру, запуск інструментів малювання й стирання, а також виклик експортних операцій.

Сервісний рівень (Services) реалізує прикладну логіку системи. Зокрема, сервіс авторизації виконує перевірку облікових даних та реєстрацію користувачів, а сервіс мап координує операції створення, відкриття та оновлення мап, завантаження

елементів, застосування змін і збереження результатів. Окремий сервіс категорій забезпечує створення категорій та призначення їх мапам. Сервіси виступають “посередником” між UI та шаром даних: вони формують потрібні операції й забезпечують цілісність дій користувача (наприклад, оновлення метаданих мапи разом із її відображенням у списку).

Рівень доступу до даних (Repositories) ізолює роботу з базою даних та містить реалізації CRUD-операцій для основних сутностей: користувачів, мап, категорій, елементів мапи й штрихів малювання. Доступ здійснюється через JDBC-з’єднання з MySQL. Така організація дозволяє змінювати або оптимізувати SQL-запити без впливу на інші частини системи, а також зменшує зв’язність між компонентами.

Для роботи редактора виділено підсистему Editor Tools, яка об’єднує інструменти взаємодії з полотном (вибір, додавання текстових вузлів, додавання вузлів-зображень, малювання, ластик). Дії редагування, які мають підтримувати скасування та повторення, реалізуються через підсистему Undo/Redo (Command). Вона накопичує виконані команди та дозволяє відкотити зміни або повторити їх, забезпечуючи передбачувану поведінку редактора при активному редагуванні мапи.

Окремим компонентом є підсистема експорту, яка формує вихідні файли мапи у форматах PNG/JPG та текстовий звіт/статистику у TXT. Під час експорту або додавання зображень система взаємодіє з локальною файловою системою: зображення зчитуються з диска, а результати експорту записуються у вибрану користувачем директорію.

2.2.2. Вибір та обґрунтування патернів реалізації

У проєкті з самого початку було важливо розділити відповідальності між шарами: інтерфейс (Swing-форми) лише ініціює дії користувача, бізнес-логіка зосереджена в сервісах, а робота з MySQL винесена в окремі репозиторії. Для цього використано підхід **Repository**: у коді є базовий інтерфейс Repository<T,ID> та конкретні JDBC-репозиторії для користувачів, мап, категорій, елементів і штрихів

малювання. Це дозволяє сервісам працювати з даними через зрозумілий API CRUD, не «змішуючи» SQL з UI-логікою.

Для редактора мапи критичною була зручна зміна інструментів (Select/Text/Image/Pen/Eraser) і можливість додавати нові інструменти без переписування клієнтського коду. Саме тому застосовано **Abstract Factory**: інтерфейс ToolFactory задає сімейство інструментів, а DefaultToolFactory створює конкретні реалізації (SelectTool, AddTextNodeTool, AddImageNodeTool, PenTool тощо). У результаті CanvasPanel/MindMapEditorForm не “знає”, як саме створюється кожен інструмент — він просто просить фабрику видати потрібний Tool. Це зменшує зв’язність та спрощує розширення редактора.

Ключова вимога для графічного редактора це зробити відміну дії (Undo/Redo). Тут найкраще працює патерн **Command**: кожна дія користувача оформлена як окрема команда з методами execute() та undo() (додавання вузла, переміщення, зміна розміру, видалення, додавання штриха, стирання штрихів). Команди накопичуються в CommandManager у двох стеках, завдяки чому система може коректно «відкотити» або повторити будь-яку дію. Це особливо важливо для редагування мапи, де багато дрібних операцій і користувач очікує безпечного експериментування.

Експорт мапи реалізовано через **Strategy**, тому що формати експорту (PNG/JPG/звіт TXT) мають різні алгоритми, але однаковий сценарій запуску з UI. Використано ExportService, який працює з інтерфейсом ExportStrategy: форма лише обирає потрібну стратегію (наприклад, PNG або JPG), після чого викликає загальний метод export(...). Так ми уникаємо великої кількості умовної логіки всередині сервісу експорту й отримуємо можливість додавати нові формати без «поломки» існуючих.

Щоб не перетворювати Dashboard на «моноліт» із прямими зв’язками між усіма панелями (список мап, властивості, пошук/сортування, обране, категорії), застосовано **Mediator**: DashboardMediatorImpl централізовано обробляє події (вибір мапи, зміна пошуку, зміна сортування, перемикання “favorite”, призначення категорії) і координує оновлення інших UI-компонентів та синхронізацію з БД через

сервіс. Це робить UI-модулі простішими й дозволяє змінювати панелі без каскадних правок по всьому інтерфейсу.

Окремо для роботи з різними типами елементів (TextNode та ImageNode) застосовано **Visitor**. Через MapElementVisitor можна виконувати операції над елементами, не «роздуваючи» класи елементів і не створюючи великі instanceof-ланцюжки в кожній новій функції. У нашому випадку Visitor використано для коректного рендерингу контенту вузлів (текст із переносами, зображення з кешем і fallback) та підготовки статистики/звіту (TXT) як окремої операції над тими самими елементами мапи.

2.3. Інструкція користувача

2.3.1 Підготовка до роботи (перший запуск)

1. Встановіть Java (JRE/JDK) на комп’ютер.
2. Встановіть MySQL Server і переконайтесь, що сервіс запущено.
3. Створіть базу даних mindmapdb та таблиці, виконавши SQL-скрипт (його доцільно розмістити в додатку курсової як “Лістинг створення БД”).
4. Налаштуйте підключення до БД у проєкті (host/port/db/login/password) відповідно до ваших параметрів MySQL.
5. Запустіть застосунок:
   або з IDE (IntelliJ IDEA) через App.java,
   або зібраним .jar файлом (якщо у вас налаштована збірка).

2.3.2 Вхід у систему та реєстрація

1. Після запуску відкривається форма авторизації.
2. Якщо ви ще не маєте акаунта:
   натисніть «Реєстрація»,
   введіть username і password,
   підтвердьте створення акаунта.
3. Для входу:
   введіть логін і пароль,
   натисніть «Увійти».
4. Після успішного входу відкриється Dashboard зі списком ваших мап.

2.3.3 Робота з мапами на Dashboard

На головному екрані користувач може керувати своїми мапами:
Створення мапи:

1. Натисніть «Створити мапу».
2. Введіть назву мапи та підтвердьте.

3. Мапа з’явиться у списку та може бути відкрита в редакторі.

Відкриття мапи:

- Двічі клацніть по мапі у списку (або використайте кнопку/меню відкриття, якщо воно передбачене).

Пошук і сортування

1. У полі пошуку введіть текст – список мап відфільтрується.
2. Оберіть режим сортування – список буде впорядковано за вибраним критерієм.

Обране (Favorite):

- Для швидкого доступу до важливих мап позначайте їх як обрані (через відповідний перемикач).

Категорії:

1. Створіть категорію (назва + колір).
2. Призначте категорію обраній мапі.
3. За потреби змінюйте категорію – це впливає лише на організацію, а не на вміст мапи.

Редагування назви/опису:

- У панелі властивостей змініть назву/опис та збережіть (якщо ця функція реалізована в UI вашої версії).

2.3.4 Редагування мапи в редакторі (Canvas)

Після відкриття мапи запускається редактор, де доступні операції з вузлами та малюванням.

Додавання текстового вузла:

1. Оберіть інструмент додавання тексту.
2. Клацніть на полотні – з’явиться текстовий вузол.
3. Для зміни тексту: подвійний клік по вузлу → введіть текст → підтвердьте.

Додавання вузла-зображення:

1. Оберіть інструмент додавання зображення.
2. Вкажіть файл (PNG/JPG).
3. Зображення з’явиться як окремий вузол на полотні.

Переміщення та зміна розміру:

- Виділіть вузол і перетягніть його мишею.
- Якщо підтримується зміна розміру – змініть розмір маркерами/рамкою (залежить від вашої реалізації UI).

Видалення вузла:

- Виділіть вузол натиснув Delete/Backspace.

2.3.5 Малювання та стирання

Малювання:

1. Оберіть Pen або Dashed Pen.
2. Натисніть та ведіть мишею по полотну – створюється штрих.

Стирання:

1. Оберіть Eraser.

2. Проведіть по штрихах, які потрібно стерти.

2.3.6 Undo / Redo

- Для скасування останньої дії використовуйте Undo.
- Для повторення скасованої дії – Redo.

Це працює для дій редагування (вузли) та для дій з малюванням/стиранням (якщо вони включені в історію команд у вашій версії).

2.3.7 Експорт мапи

1. На Dashboard або в редакторі оберіть “Експорт мапи”.
2. Вкажіть формат:

- PNG або JPG (зображення мапи),
- TXT (звіт/статистика по мапі).

3. Оберіть місце збереження файл буде записаний у файлову систему.

Додатково: робота з SOA-модулем:

У межах роботи було реалізовано сервіс-орієнтований модуль (SOA): окремий локальний Web Service (MindMapSoaServerMain) та клієнтська частина (ApiClient, MindMapSoaClientDemo), які обмінюються даними по HTTP. Для захисту даних використовується спільний middleware-контракт: SoaConfig (порт/адреса/спільний секрет), FormCodec (кодування параметрів), CryptoUtils (двостороннє шифрування), а доступ до ендпоінтів після входу контролюється токеном (TokenService).

Передумови:

1. Запущено MySQL і створено БД mindmapdb (сервер використовує ті самі JDBC-репозиторії, що й основний застосунок).
2. Перевірте, що у SoaConfig вказано стандартний порт 8088 і базову адресу http://localhost:8088, а також спільний секрет однаковий для клієнта й сервера.

Запуск SOA-сервера:

1. Запустіть клас MindMapSoaServerMain (пакет soa/server).
2. Сервер підніме локальний HTTP-сервіс (типово на http://localhost:8088).
3. За потреби можна перевірити доступність сервісу через “health-check” (у реалізації він створюється як простий endpoint /).

Виконання запитів з клієнта (демо):

1. Запустіть клієнтський демо-клас MindMapSoaClientDemo (працює через ApiClient).
2. У консолі виконайте логін. Після успішного входу сервер видає токен, який клієнт автоматично додає до наступних запитів (список мап, створення мапи тощо).
3. Далі клієнт може:

- отримати список мап,
- створити нову мапу (у демо є відповідний сценарій із запитом на створення).

Перевірка результату в основному застосунку:

Оскільки SOA-сервер використовує ті ж сервіси та JDBC-репозиторії і працює з тією ж БД, створену через клієнт мапу можна побачити і в десктопному застосунку після оновлення списку/перезапуску.

ВИСНОВКИ

У межах курсової роботи було розроблено та реалізовано настільний застосунок Mind-mapping software для створення й редагування ментальних карт. Під час виконання роботи було послідовно пройдено етапи аналізу предметної області, формування вимог, проектування та реалізації компонентів системи.

У першому розділі було виконано проєктування системи: проведено огляд існуючих рішень, сформовано функціональні та нефункціональні вимоги, описано сценарії використання й побудовано UML-діаграми (use-case, діаграма класів предметної області, діаграма розгортання). Також обґрунтовано вибір технологій, реляційної бази даних MySQL, мови програмування Java та середовища розробки IntelliJ IDEA.

У другому розділі реалізовано основні компоненти застосунку: графічний інтерфейс (вхід у систему, Dashboard керування мапами, редактор з полотном), сервісний рівень бізнес-логіки та шар доступу до даних через JDBC-репозиторії. Спроєктовано і реалізовано структуру бази даних, яка забезпечує збереження користувачів, категорій, мап, елементів мапи та штрихів малювання з можливістю відновлення стану при повторному відкритті. Окремо реалізовано експорт мап у формати PNG/JPG та формування текстового звіту/статистики (TXT), а також інструменти редактора для роботи з вузлами й малюванням.

Для підвищення якості архітектури та спрощення розширення системи використано шаблони проєктування: Command (Undo/Redo для дій редактора), Strategy (вибір алгоритмів експорту), Abstract Factory (створення інструментів редактора), Mediator (координація компонентів Dashboard) та Visitor (обробка різних типів елементів мапи й підготовка статистики). Додатково, в межах навчальних завдань було реалізовано демонстраційний SOA-модуль із локальним веб-сервісом і клієнтом, що показує можливість розширення застосунку у напрямі клієнт-серверної взаємодії.

Результатом виконаної роботи є працездатний програмний продукт, який дозволяє користувачу авторизуватися, створювати та організовувати ментальні карти, редагувати їх у графічному редакторі (включно з малюванням), зберігати дані в базі та експортувати результати у файли. Реалізована архітектура є модульною та придатною для подальшого розвитку, зокрема для додавання нових інструментів редактора, форматів експорту та розширення мережевих можливостей.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

Best mind map software of 2025. Christian Cawley [Електронний ресурс]: https://www.techradar.com/best/best-mind-map-software

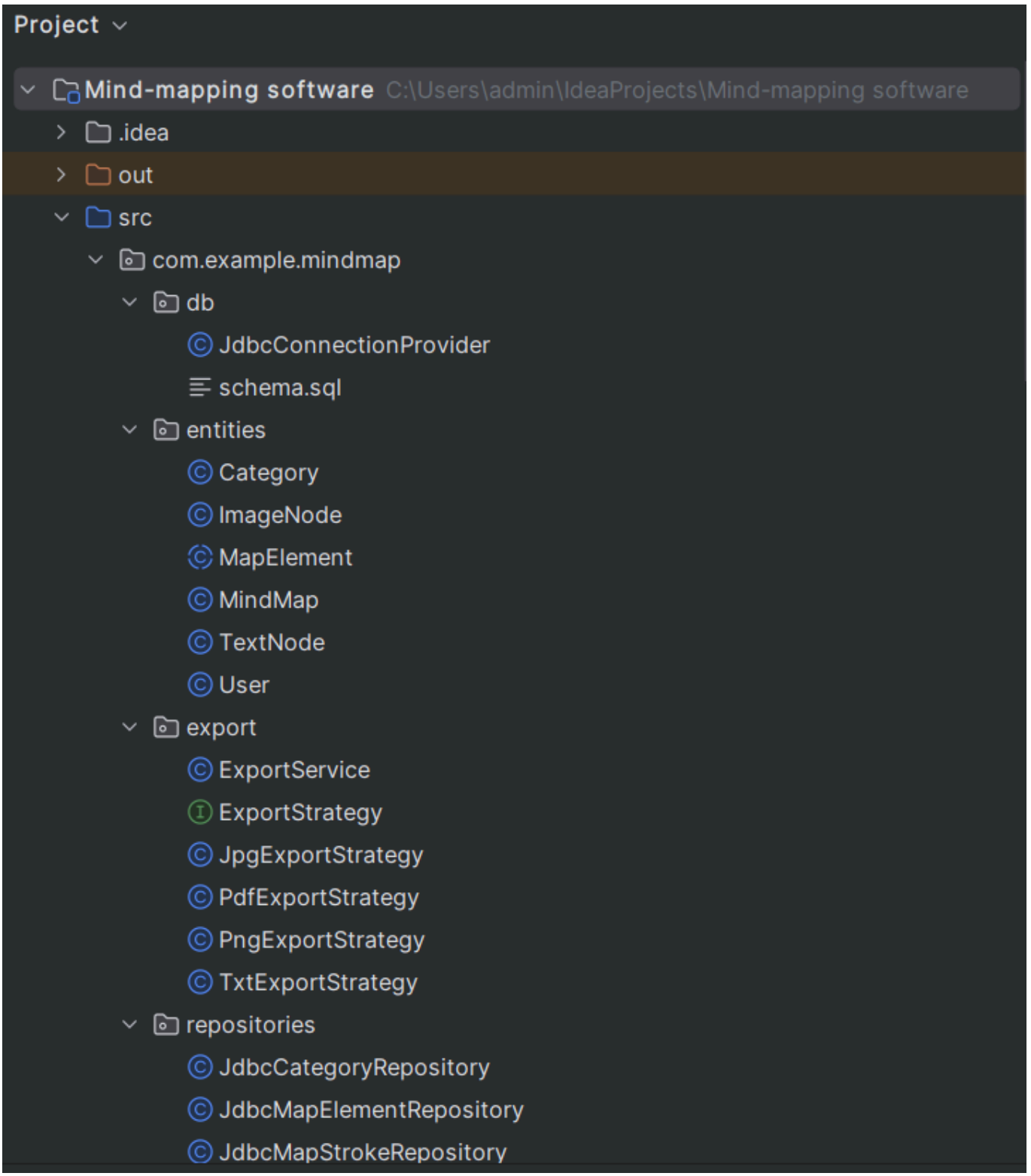
MySQL Connector/J Developer Guide [Електронний ресурс]: https://dev.mysql.com/doc/connector-j/en/

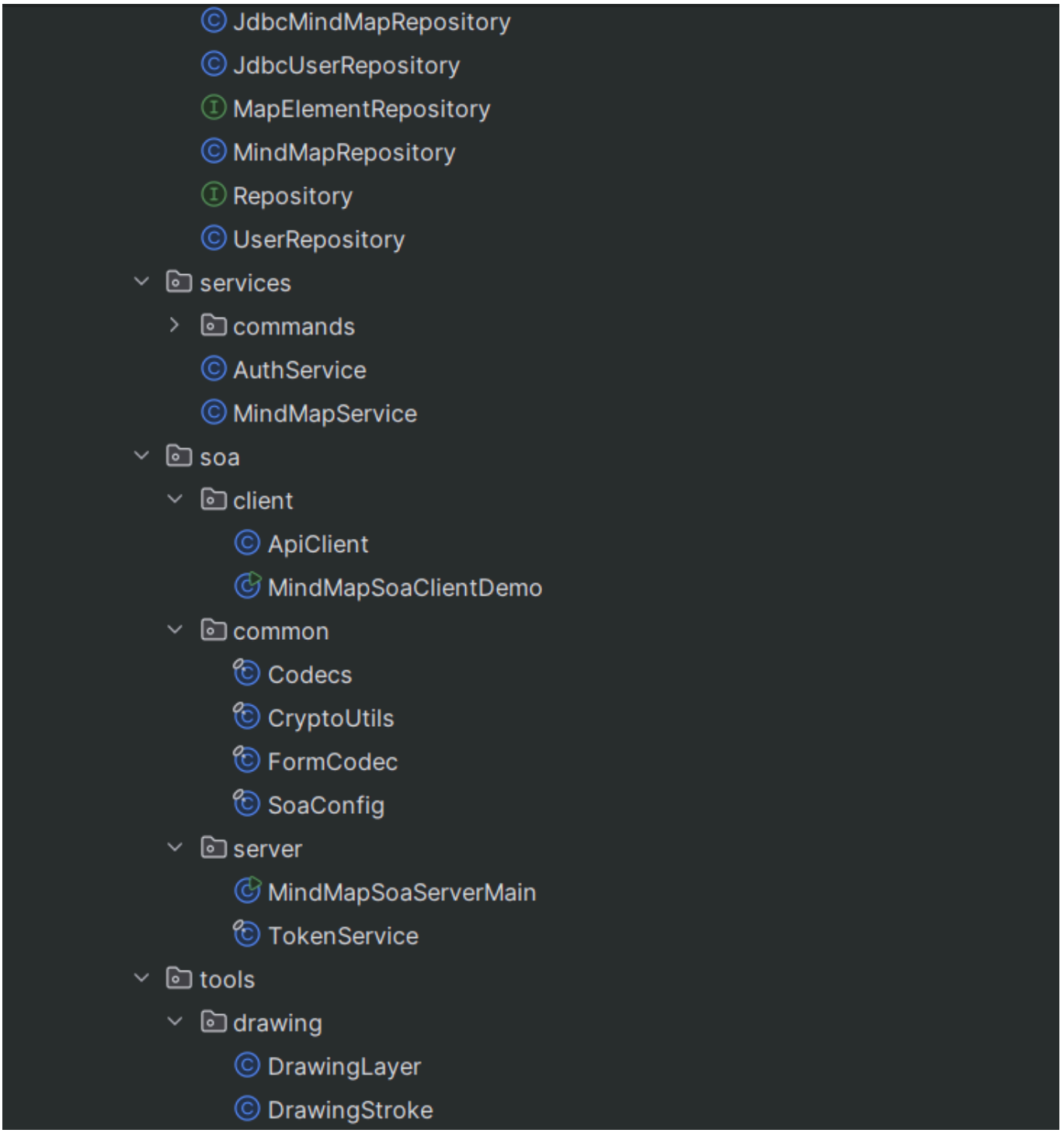
Java Developer's Guide [Електронний ресурс]: https://docs.oracle.com/en/database/oracle/oracle-database/19/jjdev/Java-overview.html

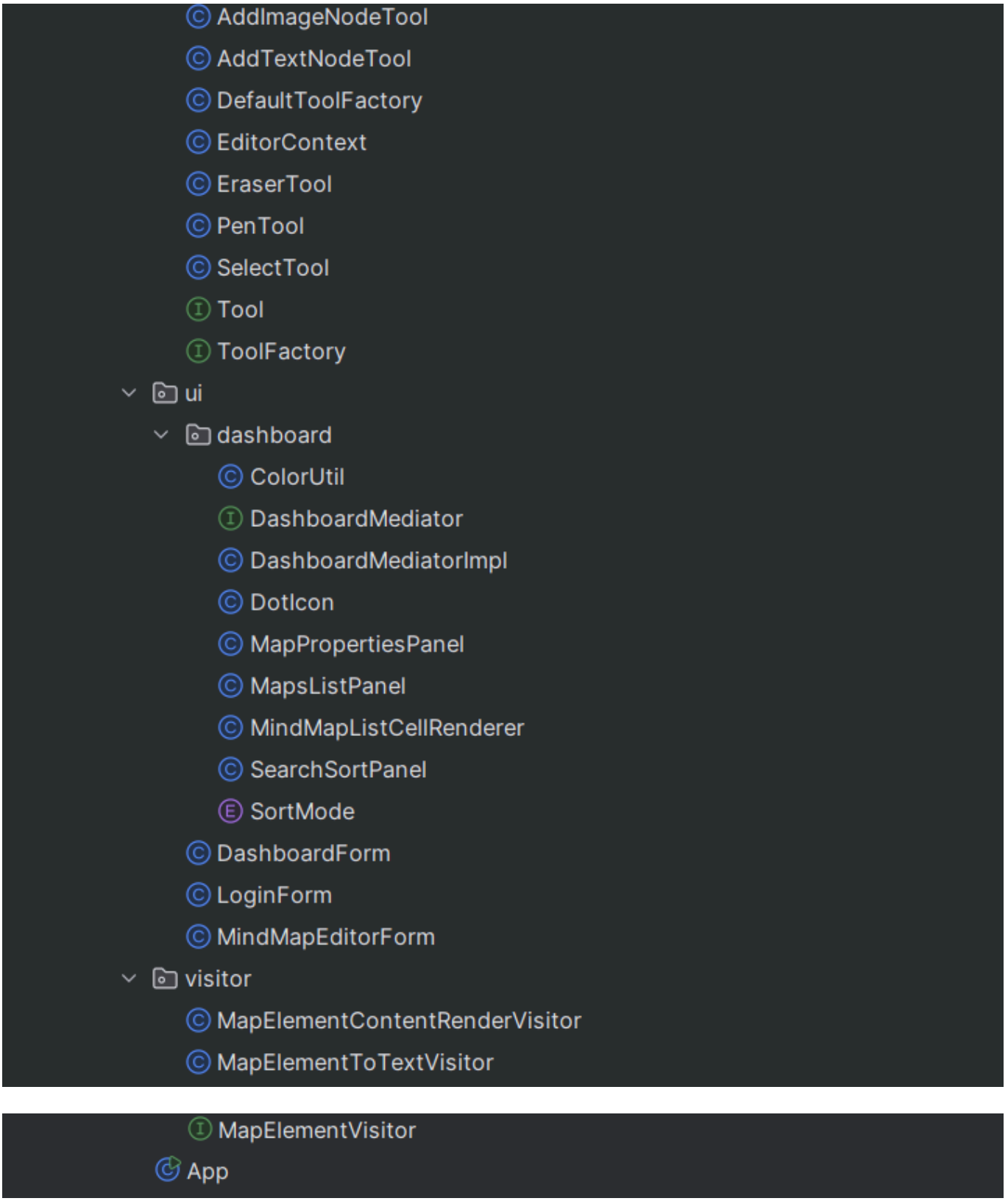
IntelliJ IDEA JetBrains IDEs. Features overview [Електронний ресурс]: https://www.jetbrains.com/idea/features/

ДОДАТКИ

Лістинг коду системи:




JdbcMindMapRepository
JdbcUserRepository
MapElementRepository
MindMapRepository
Repository
UserRepository
services
commands
AuthService
MindMapService
soa
client
ApiClient
MindMapSoaClientDemo
common
Codecs
CryptoUtils
FormCodec
SoaConfig
server
MindMapSoaServerMain
TokenService
tools
drawing
DrawingLayer
DrawingStroke



src/com/example/mindmap/App.java:

```
package com.example.mindmap;

import com.example.mindmap.db.JdbcConnectionProvider;
import com.example.mindmap.repositories.JdbcMapElementRepository;
```

```
import com.example.mindmap.repositories.JdbcMindMapRepository;
import com.example.mindmap.repositories.JdbcUserRepository;
import com.example.mindmap.services.AuthService;
import com.example.mindmap.services.MindMapService;
import com.example.mindmap.ui.LoginForm;

import javax.swing.*;

public class App {
   public static void main(String[] args) {

       // Системний стиль вигляду
       try {
           UIManager.setLookAndFeel(
                   UIManager.getSystemLookAndFeelClassName()
           );
       } catch (Exception ignored) {}

       // Підключення до БД
       JdbcConnectionProvider connectionProvider = new JdbcConnectionProvider();

       // Репозиторії
       JdbcUserRepository userRepo = new JdbcUserRepository(connectionProvider);
       JdbcMindMapRepository mapRepo = new JdbcMindMapRepository(connectionProvider);
       JdbcMapElementRepository elementRepo = new
JdbcMapElementRepository(connectionProvider);

       // Сервіси
       AuthService authService = new AuthService(userRepo);
       MindMapService mindMapService = new MindMapService(mapRepo, elementRepo);

       // Стартове вікно (логін/реєстрація)
       SwingUtilities.invokeLater(() ->
               new LoginForm(authService, mindMapService).setVisible(true)
       );
   }
}
```

src/com/example/mindmap/db/JdbcConnectionProvider.java:

```
package com.example.mindmap.db;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class JdbcConnectionProvider {

   private static final String URL = "jdbc:mysql://localhost:3306/mindmapdb";
   private static final String USER = "root";
   private static final String PASSWORD = ""; // АБО твій реальний пароль, якщо він є
```

```
    static {
        try {
            Class.forName("com.mysql.cj.jdbc.Driver"); // драйвер вже працює
        } catch (ClassNotFoundException e) {
            throw new RuntimeException("JDBC Driver not found", e);
        }
    }

    public Connection getConnection() throws SQLException {
        return DriverManager.getConnection(URL, USER, PASSWORD);
    }
}
```

src/com/example/mindmap/db/schema.sql:

```
-- 0. Створюємо базу даних, якщо її ще немає
CREATE DATABASE IF NOT EXISTS mindmapdb CHARACTER SET utf8mb4;
USE mindmapdb;

-- 1. Таблиця користувачів
CREATE TABLE IF NOT EXISTS Users (
    id INT PRIMARY KEY AUTO_INCREMENT,
    username VARCHAR(50) NOT NULL UNIQUE,
    password_hash VARCHAR(255) NOT NULL
);

-- 2. Таблиця категорій
CREATE TABLE IF NOT EXISTS Categories (
    id INT PRIMARY KEY AUTO_INCREMENT,
    title VARCHAR(100) NOT NULL,
    color VARCHAR(7),
    user_id INT NOT NULL,
    FOREIGN KEY (user_id) REFERENCES Users(id) ON DELETE CASCADE
);

-- 3. Таблиця мап
CREATE TABLE IF NOT EXISTS MindMaps (
    id INT PRIMARY KEY AUTO_INCREMENT,
    title VARCHAR(200) NOT NULL,
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
    is_favorite BOOLEAN DEFAULT FALSE,
    preview_image VARCHAR(255),
    user_id INT NOT NULL,
    category_id INT, -- Може бути NULL, якщо категорія не обрана
    FOREIGN KEY (user_id) REFERENCES Users(id) ON DELETE CASCADE,
    FOREIGN KEY (category_id) REFERENCES Categories(id) ON DELETE SET NULL
);

-- 4. Таблиця елементів мапи
CREATE TABLE IF NOT EXISTS MapElements (
    id INT PRIMARY KEY AUTO_INCREMENT,
    map_id INT NOT NULL,
    x_coord FLOAT NOT NULL,
    y_coord FLOAT NOT NULL,
```

```
    -- Колонка-дискримінатор (визначає тип: 'TEXT' або 'IMAGE')
    element_type VARCHAR(10) NOT NULL,

    -- Поля для TextNode
    text_content TEXT,
    font_size INT,
    shape_type VARCHAR(20),

    -- Поля для ImageNode
    image_url VARCHAR(255),
    width INT,
    height INT,

    FOREIGN KEY (map_id) REFERENCES MindMaps(id) ON DELETE CASCADE
);

CREATE TABLE IF NOT EXISTS MapStrokes (
    stroke_id VARCHAR(36) PRIMARY KEY,
    map_id INT NOT NULL,
    color_argb INT NOT NULL,
    width FLOAT NOT NULL,
    dashed BOOLEAN NOT NULL,
    points LONGTEXT NOT NULL,
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (map_id) REFERENCES MindMaps(id) ON DELETE CASCADE
);

CREATE INDEX idx_mapstrokes_map_id ON MapStrokes(map_id);
```

src/com/example/mindmap/entities/Category.java:

```
package com.example.mindmap.entities;

public class Category {
    private int id;
    private String title;
    private String color; // "#RRGGBB"
    private User owner;

    public Category() {}

    public Category(int id, String title, String color, User owner) {
        this.id = id;
        this.title = title;
        this.color = color;
        this.owner = owner;
    }

    public int getId() { return id; }
```

```
    public void setId(int id) { this.id = id; }

    public String getTitle() { return title; }
    public void setTitle(String title) { this.title = title; }

    public String getColor() { return color; }
    public void setColor(String color) { this.color = color; }

    public User getOwner() { return owner; }
    public void setOwner(User owner) { this.owner = owner; }

    @Override
    public String toString() {
        return title; // щоб ComboBox показував назву
    }
}
```

src/com/example/mindmap/entities/ImageNode.java:

```
package com.example.mindmap.entities;

public class ImageNode extends MapElement {

    private String imageUrl;

    public ImageNode() {
        super(0, 0, 0, null, 120, 80);
    }

    public ImageNode(int id, float x, float y, MindMap map,
                     String imageUrl, int width, int height) {
        super(id, x, y, map, width, height);
        this.imageUrl = imageUrl;
    }

    // Конструктор для редактора
    public ImageNode(float x, float y, String imageUrl) {
        super(0, x, y, null, 120, 80);
        this.imageUrl = imageUrl;
    }

    @Override
    public String getType() {
        return "IMAGE";
    }

    @Override
    public String getTextForDisplay() {
        return null;
    }

    @Override
    public void setTextForDisplay(String text) {
        // нічого не робимо
```

```
    }

    public String getImageUrl() {
        return imageUrl;
    }

    public void setImageUrl(String imageUrl) {
        this.imageUrl = imageUrl;
    }
}
```

src/com/example/mindmap/entities/MapElement.java:

```
package com.example.mindmap.entities;

public abstract class MapElement {

    protected int id;
    protected float x;
    protected float y;
    protected MindMap map;

    // розміри елемента (тепер є у всіх вузлів)
    protected int width;
    protected int height;

    // Старий конструктор залишаємо (для сумісності)
    public MapElement(int id, float x, float y, MindMap map) {
        this(id, x, y, map, 140, 45);
    }

    // конструктор з розмірами
    public MapElement(int id, float x, float y, MindMap map, int width, int height) {
        this.id = id;
        this.x = x;
        this.y = y;
        this.map = map;
        this.width = width;
        this.height = height;
    }

    public abstract String getType();
    public abstract String getTextForDisplay();
    public abstract void setTextForDisplay(String text);

    // --- getters / setters ---

    public int getId() { return id; }
    public void setId(int id) { this.id = id; }

    public float getX() { return x; }
    public void setX(float x) { this.x = x; }

    public float getY() { return y; }
```

```
    public void setY(float y) { this.y = y; }

    public MindMap getMap() { return map; }
    public void setMap(MindMap map) { this.map = map; }

    // розміри
    public int getWidthPx() { return width; }
    public void setWidthPx(int width) { this.width = Math.max(width, 40); }   // мінімум

    public int getHeightPx() { return height; }
    public void setHeightPx(int height) { this.height = Math.max(height, 30); } //
мінімум
}
```

src/com/example/mindmap/entities/MindMap.java:

```
package com.example.mindmap.entities;

import java.time.LocalDateTime;
import java.util.ArrayList;
import java.util.List;

public class MindMap {
    private int id;
    private String title;
    private String description;          // нове поле
    private LocalDateTime createdAt;
    private boolean favorite;
    private String previewImage;

    private User owner;
    private Category category;
    private final List<MapElement> elements = new ArrayList<>();

    public MindMap() {
        this.createdAt = LocalDateTime.now();
    }

    public MindMap(int id, String title, User owner) {
        this.id = id;
        this.title = title;
        this.owner = owner;
        this.createdAt = LocalDateTime.now();
    }

    public void markAsFavorite() {
        this.favorite = true;
    }

    public void unmarkFavorite() {
        this.favorite = false;
    }

    public void changeCategory(Category category) {
```

```
        this.category = category;
    }

    // Для сумісності з репозиторієм/медіатором
    public void setCategory(Category category) {
        changeCategory(category);
    }

    public void addElement(MapElement element) {
        elements.add(element);
    }

    public void removeElement(MapElement element) {
        elements.remove(element);
    }

    // getters / setters
    public int getId() { return id; }
    public void setId(int id) { this.id = id; }

    public String getTitle() { return title; }
    public void setTitle(String title) { this.title = title; }

    public String getDescription() { return description; }
    public void setDescription(String description) { this.description = description; }

    public LocalDateTime getCreatedAt() { return createdAt; }
    public void setCreatedAt(LocalDateTime createdAt) { this.createdAt = createdAt; }

    public boolean isFavorite() { return favorite; }
    public void setFavorite(boolean favorite) { this.favorite = favorite; }

    public String getPreviewImage() { return previewImage; }
    public void setPreviewImage(String previewImage) { this.previewImage = previewImage;
}

    public User getOwner() { return owner; }
    public void setOwner(User owner) { this.owner = owner; }

    public Category getCategory() { return category; }

    public List<MapElement> getElements() { return elements; }

    @Override
    public String toString() {
        return title;
    }
}
```

src/com/example/mindmap/entities/TextNode.java:

```
package com.example.mindmap.entities;

public class TextNode extends MapElement {
```

```
private String textContent;
private int fontSize;
private String shapeType;

public TextNode() {
    super(0, 0, 0, null, 140, 45);
}

public TextNode(int id, float x, float y, MindMap map,
                String textContent, int fontSize, String shapeType) {
    super(id, x, y, map, 140, 45);
    this.textContent = textContent;
    this.fontSize = fontSize;
    this.shapeType = shapeType;
}

//конструктор з розмірами (корисно для БД/ресайзу)
public TextNode(int id, float x, float y, MindMap map,
                String textContent, int fontSize, String shapeType,
                int width, int height) {
    super(id, x, y, map, width, height);
    this.textContent = textContent;
    this.fontSize = fontSize;
    this.shapeType = shapeType;
}

// Для редактора
public TextNode(float x, float y, String textContent) {
    super(0, x, y, null, 140, 45);
    this.textContent = textContent;
    this.fontSize = 14;
    this.shapeType = "RECT";
}

@Override
public String getType() {
    return "TEXT";
}

@Override
public String getTextForDisplay() {
    return textContent;
}

@Override
public void setTextForDisplay(String text) {
    this.textContent = text;
}

public String getTextContent() { return textContent; }
public void setTextContent(String textContent) { this.textContent = textContent; }

public int getFontSize() { return fontSize; }
```

```
    public void setFontSize(int fontSize) { this.fontSize = fontSize; }

    public String getShapeType() { return shapeType; }
    public void setShapeType(String shapeType) { this.shapeType = shapeType; }
}
```

src/com/example/mindmap/entities/User.java:

```
package com.example.mindmap.entities;

public class User {
    private int id;
    private String username;
    private String passwordHash;

    public User() {
    }

    public User(int id, String username, String passwordHash) {
        this.id = id;
        this.username = username;
        this.passwordHash = passwordHash;
    }

    public void updateProfile(String newUsername, String newPasswordHash) {
        this.username = newUsername;
        this.passwordHash = newPasswordHash;
    }

    // гетери/сетери
    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public String getPasswordHash() {
        return passwordHash;
    }

    public void setPasswordHash(String passwordHash) {
        this.passwordHash = passwordHash;
    }
}
```

src/com/example/mindmap/export/ExportService.java:

```
package com.example.mindmap.export;

import com.example.mindmap.entities.MindMap;

import javax.swing.*;
import java.io.File;
import java.io.IOException;

public class ExportService {

   private ExportStrategy strategy;

   public void setStrategy(ExportStrategy strategy) {
       this.strategy = strategy;
   }

   public void export(MindMap map, JComponent canvas, File targetFile) throws
IOException {
       if (strategy == null) {
           throw new IllegalStateException("Export strategy is not set");
       }
       strategy.export(map, canvas, targetFile);
   }
}
```

src/com/example/mindmap/export/ExportStrategy.java:

```
package com.example.mindmap.export;

import com.example.mindmap.entities.MindMap;

import javax.swing.*;
import java.io.File;
import java.io.IOException;

public interface ExportStrategy {
   void export(MindMap map, JComponent canvas, File targetFile) throws IOException;
}
```

src/com/example/mindmap/export/JpgExportStrategy.java:

```
package com.example.mindmap.export;

import com.example.mindmap.entities.MindMap;

import javax.imageio.ImageIO;
import javax.swing.*;
import java.awt.*;
import java.awt.image.BufferedImage;
import java.io.File;
```

```
import java.io.IOException;

public class JpgExportStrategy implements ExportStrategy {

   @Override
   public void export(MindMap map, JComponent canvas, File targetFile) throws
IOException {
       int w = canvas.getWidth();
       int h = canvas.getHeight();

       if (w <= 0 || h <= 0) {
           throw new IOException("Canvas has invalid size");
       }

       BufferedImage image = new BufferedImage(w, h, BufferedImage.TYPE_INT_RGB);
       Graphics2D g2 = image.createGraphics();
       // Білий фон, щоб не було чорного фону прозорості
       g2.setColor(Color.WHITE);
       g2.fillRect(0, 0, w, h);
       canvas.printAll(g2);
       g2.dispose();

       ImageIO.write(image, "jpg", targetFile);
   }
}
```

src/com/example/mindmap/export/PdfExportStrategy.java:

```
package com.example.mindmap.export;

import com.example.mindmap.entities.MindMap;

import javax.swing.*;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;

public class PdfExportStrategy implements ExportStrategy {

   @Override
   public void export(MindMap map, JComponent canvas, File targetFile) throws
IOException {
       // Спрощена реалізація: текстовий псевдо-PDF.
       // Для реального PDF потрібна окрема бібліотека.
       try (FileWriter writer = new FileWriter(targetFile)) {
           writer.write("Mind Map Export (pseudo PDF)\n");
           writer.write("Title: " + map.getTitle() + "\n");
           writer.write("Canvas size: " + canvas.getWidth() + "x" + canvas.getHeight()
+ "\n");
           writer.write("Note: For real PDF export a PDF library should be used.\n");
       }
   }
}
```

src/com/example/mindmap/export/PngExportStrategy.java:

```
package com.example.mindmap.export;

import com.example.mindmap.entities.MindMap;

import javax.imageio.ImageIO;
import javax.swing.*;
import java.awt.*;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.IOException;

public class PngExportStrategy implements ExportStrategy {

   @Override
   public void export(MindMap map, JComponent canvas, File targetFile) throws
IOException {
       int w = canvas.getWidth();
       int h = canvas.getHeight();

       if (w <= 0 || h <= 0) {
           throw new IOException("Canvas has invalid size");
       }

       BufferedImage image = new BufferedImage(w, h, BufferedImage.TYPE_INT_ARGB);
       Graphics2D g2 = image.createGraphics();
       canvas.printAll(g2);
       g2.dispose();

       ImageIO.write(image, "png", targetFile);
   }
}
```

src/com/example/mindmap/repositories/JdbcCategoryRepository.java:

```
package com.example.mindmap.repositories;

import com.example.mindmap.db.JdbcConnectionProvider;
import com.example.mindmap.entities.Category;
import com.example.mindmap.entities.User;

import java.sql.*;
import java.util.ArrayList;
import java.util.List;
import java.util.Optional;

public class JdbcCategoryRepository implements Repository<Category, Integer> {

   private final JdbcConnectionProvider connectionProvider;

   public JdbcCategoryRepository(JdbcConnectionProvider connectionProvider) {
```

```
        this.connectionProvider = connectionProvider;
    }

    @Override
    public Category save(Category entity) {
        if (entity.getId() == 0) {
            String sql = "INSERT INTO Categories(title, color, user_id) VALUES(?, ?,
?)";
            try (Connection conn = connectionProvider.getConnection();
                 PreparedStatement ps = conn.prepareStatement(sql,
Statement.RETURN_GENERATED_KEYS)) {

                ps.setString(1, entity.getTitle());
                ps.setString(2, entity.getColor());
                ps.setInt(3, entity.getOwner().getId());
                ps.executeUpdate();

                try (ResultSet rs = ps.getGeneratedKeys()) {
                    if (rs.next()) entity.setId(rs.getInt(1));
                }
                return entity;

            } catch (SQLException e) {
                throw new RuntimeException("Error inserting category", e);
            }
        } else {
            String sql = "UPDATE Categories SET title = ?, color = ? WHERE id = ? AND
user_id = ?";
            try (Connection conn = connectionProvider.getConnection();
                 PreparedStatement ps = conn.prepareStatement(sql)) {

                ps.setString(1, entity.getTitle());
                ps.setString(2, entity.getColor());
                ps.setInt(3, entity.getId());
                ps.setInt(4, entity.getOwner().getId());
                ps.executeUpdate();

                return entity;

            } catch (SQLException e) {
                throw new RuntimeException("Error updating category", e);
            }
        }
    }

    @Override
    public void delete(Integer id) {
        String sql = "DELETE FROM Categories WHERE id = ?";
        try (Connection conn = connectionProvider.getConnection();
             PreparedStatement ps = conn.prepareStatement(sql)) {

            ps.setInt(1, id);
            ps.executeUpdate();
```

```
        } catch (SQLException e) {
            throw new RuntimeException("Error deleting category", e);
        }
    }

    @Override
    public Optional<Category> getById(Integer id) {
        String sql = "SELECT id, title, color, user_id FROM Categories WHERE id = ?";
        try (Connection conn = connectionProvider.getConnection();
             PreparedStatement ps = conn.prepareStatement(sql)) {

            ps.setInt(1, id);
            try (ResultSet rs = ps.executeQuery()) {
                if (!rs.next()) return Optional.empty();

                Category c = new Category();
                c.setId(rs.getInt("id"));
                c.setTitle(rs.getString("title"));
                c.setColor(rs.getString("color"));
                c.setOwner(new User(rs.getInt("user_id"), null, null));
                return Optional.of(c);
            }

        } catch (SQLException e) {
            throw new RuntimeException("Error loading category", e);
        }
    }

    @Override
    public List<Category> getAll() {
        throw new UnsupportedOperationException("Not needed");
    }

    public List<Category> getAllByUser(User user) {
        String sql = "SELECT id, title, color, user_id FROM Categories WHERE user_id = ?
ORDER BY title ASC";
        List<Category> result = new ArrayList<>();

        try (Connection conn = connectionProvider.getConnection();
             PreparedStatement ps = conn.prepareStatement(sql)) {

            ps.setInt(1, user.getId());
            try (ResultSet rs = ps.executeQuery()) {
                while (rs.next()) {
                    Category c = new Category();
                    c.setId(rs.getInt("id"));
                    c.setTitle(rs.getString("title"));
                    c.setColor(rs.getString("color"));
                    c.setOwner(user);
                    result.add(c);
                }
            }
            return result;
```

```
        } catch (SQLException e) {
            throw new RuntimeException("Error loading categories by user", e);
        }
    }
}
```

src/com/example/mindmap/repositories/JdbcMapElementRepository.java:

```
package com.example.mindmap.repositories;

import com.example.mindmap.db.JdbcConnectionProvider;
import com.example.mindmap.entities.ImageNode;
import com.example.mindmap.entities.MapElement;
import com.example.mindmap.entities.TextNode;

import java.sql.*;
import java.util.ArrayList;
import java.util.List;

public class JdbcMapElementRepository implements MapElementRepository {

    private final JdbcConnectionProvider connectionProvider;

    public JdbcMapElementRepository(JdbcConnectionProvider connectionProvider) {
        this.connectionProvider = connectionProvider;
    }

    @Override
    public void save(MapElement element, int mapId) {
        String sql = """
            INSERT INTO MapElements
            (map_id, x_coord, y_coord, element_type, text_content, font_size,
shape_type, image_url, width, height)
            VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?)
        """;

        try (Connection conn = connectionProvider.getConnection();
             PreparedStatement ps = conn.prepareStatement(sql,
Statement.RETURN_GENERATED_KEYS)) {

            ps.setInt(1, mapId);
            ps.setFloat(2, element.getX());
            ps.setFloat(3, element.getY());
            ps.setString(4, element.getType()); // "TEXT" або "IMAGE"

            if (element instanceof TextNode text) {
                ps.setString(5, text.getTextContent());
                ps.setInt(6, text.getFontSize());
                ps.setString(7, text.getShapeType());

                ps.setNull(8, Types.VARCHAR); // image_url
            } else if (element instanceof ImageNode img) {
                ps.setNull(5, Types.LONGVARCHAR); // text_content
                ps.setNull(6, Types.INTEGER);     // font_size
```

```
            ps.setNull(7, Types.VARCHAR);     // shape_type

            ps.setString(8, img.getImageUrl());
        } else {
            ps.setNull(5, Types.LONGVARCHAR);
            ps.setNull(6, Types.INTEGER);
            ps.setNull(7, Types.VARCHAR);
            ps.setNull(8, Types.VARCHAR);
        }

        // ✅ width/height для будь-якого елемента
        ps.setInt(9, element.getWidthPx());
        ps.setInt(10, element.getHeightPx());

        ps.executeUpdate();

        ResultSet rs = ps.getGeneratedKeys();
        if (rs.next()) {
            element.setId(rs.getInt(1));
        }

    } catch (SQLException e) {
        throw new RuntimeException("Error saving map element", e);
    }
}

@Override
public void update(MapElement element) {
    String sql = """
        UPDATE MapElements
        SET x_coord = ?, y_coord = ?,
            text_content = ?, font_size = ?, shape_type = ?,
            image_url = ?, width = ?, height = ?
        WHERE id = ?
    """;

    try (Connection conn = connectionProvider.getConnection();
         PreparedStatement ps = conn.prepareStatement(sql)) {

        ps.setFloat(1, element.getX());
        ps.setFloat(2, element.getY());

        if (element instanceof TextNode text) {
            ps.setString(3, text.getTextContent());
            ps.setInt(4, text.getFontSize());
            ps.setString(5, text.getShapeType());

            ps.setNull(6, Types.VARCHAR); // image_url
        } else if (element instanceof ImageNode img) {
            ps.setNull(3, Types.LONGVARCHAR);
            ps.setNull(4, Types.INTEGER);
            ps.setNull(5, Types.VARCHAR);

            ps.setString(6, img.getImageUrl());
```

```
            } else {
                ps.setNull(3, Types.LONGVARCHAR);
                ps.setNull(4, Types.INTEGER);
                ps.setNull(5, Types.VARCHAR);
                ps.setNull(6, Types.VARCHAR);
            }

            ps.setInt(7, element.getWidthPx());
            ps.setInt(8, element.getHeightPx());

            ps.setInt(9, element.getId());

            ps.executeUpdate();

        } catch (SQLException e) {
            throw new RuntimeException("Error updating map element", e);
        }
    }

    @Override
    public void delete(int id) {
        String sql = "DELETE FROM MapElements WHERE id = ?";

        try (Connection conn = connectionProvider.getConnection();
             PreparedStatement ps = conn.prepareStatement(sql)) {

            ps.setInt(1, id);
            ps.executeUpdate();

        } catch (SQLException e) {
            throw new RuntimeException("Error deleting map element", e);
        }
    }

    @Override
    public List<MapElement> findByMapId(int mapId) {
        String sql = """
            SELECT id, x_coord, y_coord, element_type,
                   text_content, font_size, shape_type,
                   image_url, width, height
            FROM MapElements
            WHERE map_id = ?
        """;

        List<MapElement> elements = new ArrayList<>();

        try (Connection conn = connectionProvider.getConnection();
             PreparedStatement ps = conn.prepareStatement(sql)) {

            ps.setInt(1, mapId);
            ResultSet rs = ps.executeQuery();

            while (rs.next()) {
                int id = rs.getInt("id");
```

```
            String type = rs.getString("element_type");
            float x = rs.getFloat("x_coord");
            float y = rs.getFloat("y_coord");

            int width = rs.getInt("width");
            int height = rs.getInt("height");

            if ("TEXT".equalsIgnoreCase(type)) {
                String text = rs.getString("text_content");
                int fontSize = rs.getInt("font_size");
                String shapeType = rs.getString("shape_type");

                // Якщо в БД null -> rs.getInt дає 0, тому підстрахуємось:
                if (fontSize <= 0) fontSize = 14;
                if (shapeType == null || shapeType.isBlank()) shapeType = "RECT";
                if (width <= 0) width = 140;
                if (height <= 0) height = 45;

                TextNode node = new TextNode(
                        id,
                        x,
                        y,
                        null,
                        text,
                        fontSize,
                        shapeType,
                        width,
                        height
                );
                elements.add(node);

            } else if ("IMAGE".equalsIgnoreCase(type)) {
                String imageUrl = rs.getString("image_url");

                if (width <= 0) width = 120;
                if (height <= 0) height = 80;

                ImageNode node = new ImageNode(
                        id,
                        x,
                        y,
                        null,
                        imageUrl,
                        width,
                        height
                );
                elements.add(node);
            }
        }

    } catch (SQLException e) {
        throw new RuntimeException("Error loading map elements", e);
    }
```

```
        return elements;
    }
}
```

src/com/example/mindmap/repositories/JdbcMapStrokeRepository.java:

```
package com.example.mindmap.repositories;

import com.example.mindmap.db.JdbcConnectionProvider;
import com.example.mindmap.tools.drawing.DrawingStroke;

import java.awt.Color;
import java.sql.*;
import java.util.ArrayList;
import java.util.List;

public class JdbcMapStrokeRepository {

    private final JdbcConnectionProvider connectionProvider;

    public JdbcMapStrokeRepository(JdbcConnectionProvider connectionProvider) {
        this.connectionProvider = connectionProvider;
    }

    public List<DrawingStroke> findByMapId(int mapId) {
        String sql = """
            SELECT stroke_id, color_argb, width, dashed, points
            FROM MapStrokes
            WHERE map_id = ?
            ORDER BY created_at ASC
        """;

        List<DrawingStroke> result = new ArrayList<>();

        try (Connection conn = connectionProvider.getConnection();
             PreparedStatement ps = conn.prepareStatement(sql)) {

            ps.setInt(1, mapId);
            ResultSet rs = ps.executeQuery();

            while (rs.next()) {
                String id = rs.getString("stroke_id");
                int argb = rs.getInt("color_argb");
                float width = rs.getFloat("width");
                boolean dashed = rs.getBoolean("dashed");
                String points = rs.getString("points");

                DrawingStroke stroke = new DrawingStroke(id, new Color(argb, true),
width, dashed);
                DrawingStroke.deserializePointsInto(stroke, points);

                // пропускаємо "биті" мазки
                if (!stroke.isEmpty()) {
                    result.add(stroke);
```

```
                }
            }

        } catch (SQLException e) {
            throw new RuntimeException("Error loading strokes", e);
        }

        return result;
    }

    public void save(int mapId, DrawingStroke stroke) {
        String sql = """
            INSERT INTO MapStrokes (stroke_id, map_id, color_argb, width, dashed,
points)
            VALUES (?, ?, ?, ?, ?, ?)
        """;

        try (Connection conn = connectionProvider.getConnection();
             PreparedStatement ps = conn.prepareStatement(sql)) {

            ps.setString(1, stroke.getId());
            ps.setInt(2, mapId);
            ps.setInt(3, stroke.getColor().getRGB());
            ps.setFloat(4, stroke.getWidthPx());
            ps.setBoolean(5, stroke.isDashed());
            ps.setString(6, stroke.serializePoints());

            ps.executeUpdate();

        } catch (SQLException e) {
            throw new RuntimeException("Error saving stroke", e);
        }
    }

    public void deleteById(String strokeId) {
        String sql = "DELETE FROM MapStrokes WHERE stroke_id = ?";

        try (Connection conn = connectionProvider.getConnection();
             PreparedStatement ps = conn.prepareStatement(sql)) {

            ps.setString(1, strokeId);
            ps.executeUpdate();

        } catch (SQLException e) {
            throw new RuntimeException("Error deleting stroke", e);
        }
    }
}
```

src/com/example/mindmap/repositories/JdbcMindMapRepository.java:

```
package com.example.mindmap.repositories;

import com.example.mindmap.db.JdbcConnectionProvider;
```

```
import com.example.mindmap.entities.Category;
import com.example.mindmap.entities.MindMap;
import com.example.mindmap.entities.User;

import java.sql.*;
import java.time.LocalDateTime;
import java.util.ArrayList;
import java.util.List;
import java.util.Optional;

public class JdbcMindMapRepository implements Repository<MindMap, Integer> {

   private final JdbcConnectionProvider connectionProvider;

   public JdbcMindMapRepository(JdbcConnectionProvider connectionProvider) {
       this.connectionProvider = connectionProvider;
   }

   @Override
   public MindMap save(MindMap entity) {
       if (entity.getId() == 0) {
           String sql = "INSERT INTO MindMaps(title, description, created_at,
is_favorite, preview_image, user_id, category_id) " +
                   "VALUES(?, ?, ?, ?, ?, ?, ?)";

           try (Connection conn = connectionProvider.getConnection();
                PreparedStatement ps = conn.prepareStatement(sql,
Statement.RETURN_GENERATED_KEYS)) {

               ps.setString(1, entity.getTitle());
               ps.setString(2, entity.getDescription());
               ps.setTimestamp(3, Timestamp.valueOf(entity.getCreatedAt() != null ?
entity.getCreatedAt() : LocalDateTime.now()));
               ps.setBoolean(4, entity.isFavorite());
               ps.setString(5, entity.getPreviewImage());
               ps.setInt(6, entity.getOwner().getId());

               Integer catId = (entity.getCategory() == null) ? null :
entity.getCategory().getId();
               if (catId == null || catId == 0) ps.setNull(7, Types.INTEGER);
               else ps.setInt(7, catId);

               ps.executeUpdate();

               try (ResultSet rs = ps.getGeneratedKeys()) {
                   if (rs.next()) entity.setId(rs.getInt(1));
               }

               return entity;

           } catch (SQLException e) {
               throw new RuntimeException("Error inserting mindmap", e);
           }
```

```
        } else {
            String sql = "UPDATE MindMaps SET title = ?, description = ?, is_favorite =
?, preview_image = ?, category_id = ? WHERE id = ?";

            try (Connection conn = connectionProvider.getConnection();
                 PreparedStatement ps = conn.prepareStatement(sql)) {

                ps.setString(1, entity.getTitle());
                ps.setString(2, entity.getDescription());
                ps.setBoolean(3, entity.isFavorite());
                ps.setString(4, entity.getPreviewImage());

                Integer catId = (entity.getCategory() == null) ? null :
entity.getCategory().getId();
                if (catId == null || catId == 0) ps.setNull(5, Types.INTEGER);
                else ps.setInt(5, catId);

                ps.setInt(6, entity.getId());
                ps.executeUpdate();

                return entity;

            } catch (SQLException e) {
                throw new RuntimeException("Error updating mindmap", e);
            }
        }
    }

    @Override
    public void delete(Integer id) {
        String sql = "DELETE FROM MindMaps WHERE id = ?";
        try (Connection conn = connectionProvider.getConnection();
             PreparedStatement ps = conn.prepareStatement(sql)) {

            ps.setInt(1, id);
            ps.executeUpdate();

        } catch (SQLException e) {
            throw new RuntimeException("Error deleting mindmap", e);
        }
    }

    @Override
    public Optional<MindMap> getById(Integer id) {
        String sql = """
            SELECT m.id, m.title, m.description, m.created_at, m.is_favorite,
m.preview_image, m.user_id, m.category_id,
                   c.title AS category_title, c.color AS category_color
            FROM MindMaps m
            LEFT JOIN Categories c ON m.category_id = c.id
            WHERE m.id = ?
        """;

        try (Connection conn = connectionProvider.getConnection();
```

```
            PreparedStatement ps = conn.prepareStatement(sql)) {

           ps.setInt(1, id);

           try (ResultSet rs = ps.executeQuery()) {
               if (!rs.next()) return Optional.empty();

               MindMap map = new MindMap();
               map.setId(rs.getInt("id"));
               map.setTitle(rs.getString("title"));
               map.setDescription(rs.getString("description"));
               map.setOwner(new User(rs.getInt("user_id"), null, null));

               Timestamp ts = rs.getTimestamp("created_at");
               if (ts != null) map.setCreatedAt(ts.toLocalDateTime());

               map.setPreviewImage(rs.getString("preview_image"));

               if (rs.getBoolean("is_favorite")) map.markAsFavorite();
               else map.unmarkFavorite();

               int catId = rs.getInt("category_id");
               if (!rs.wasNull() && catId != 0) {
                   Category cat = new Category();
                   cat.setId(catId);
                   cat.setTitle(rs.getString("category_title"));
                   cat.setColor(rs.getString("category_color"));
                   cat.setOwner(map.getOwner());
                   map.setCategory(cat);
               }

               return Optional.of(map);
           }

       } catch (SQLException e) {
           throw new RuntimeException("Error loading mindmap by id", e);
       }
   }

   @Override
   public List<MindMap> getAll() {
       throw new UnsupportedOperationException("Not needed");
   }

   public List<MindMap> getAllByUser(User user) {
       String sql = """
           SELECT m.id, m.title, m.description, m.created_at, m.is_favorite,
m.preview_image, m.category_id,
                  c.title AS category_title, c.color AS category_color
           FROM MindMaps m
           LEFT JOIN Categories c ON m.category_id = c.id
           WHERE m.user_id = ?
       """;
```

```
        List<MindMap> maps = new ArrayList<>();

        try (Connection conn = connectionProvider.getConnection();
             PreparedStatement ps = conn.prepareStatement(sql)) {

            ps.setInt(1, user.getId());

            try (ResultSet rs = ps.executeQuery()) {
                while (rs.next()) {
                    MindMap map = new MindMap();
                    map.setId(rs.getInt("id"));
                    map.setTitle(rs.getString("title"));
                    map.setDescription(rs.getString("description"));
                    map.setOwner(user);

                    Timestamp ts = rs.getTimestamp("created_at");
                    if (ts != null) map.setCreatedAt(ts.toLocalDateTime());

                    map.setPreviewImage(rs.getString("preview_image"));

                    if (rs.getBoolean("is_favorite")) map.markAsFavorite();
                    else map.unmarkFavorite();

                    int catId = rs.getInt("category_id");
                    if (!rs.wasNull() && catId != 0) {
                        Category cat = new Category();
                        cat.setId(catId);
                        cat.setTitle(rs.getString("category_title"));
                        cat.setColor(rs.getString("category_color"));
                        cat.setOwner(user);
                        map.setCategory(cat);
                    }

                    maps.add(map);
                }
            }

            return maps;

        } catch (SQLException e) {
            throw new RuntimeException("Error loading mindmaps by user", e);
        }
    }
}
```

src/com/example/mindmap/repositories/JdbcUserRepository.java:

```
package com.example.mindmap.repositories;

import com.example.mindmap.db.JdbcConnectionProvider;
import com.example.mindmap.entities.User;
```

```
import java.sql.*;
import java.util.ArrayList;
import java.util.List;
import java.util.Optional;

public class JdbcUserRepository implements Repository<User, Integer> {

   private final JdbcConnectionProvider connectionProvider;

   public JdbcUserRepository(JdbcConnectionProvider connectionProvider) {
       this.connectionProvider = connectionProvider;
   }

   @Override
   public User save(User entity) {
       if (entity.getId() == 0) {
           // INSERT
           String sql = "INSERT INTO Users(username, password_hash) VALUES(?, ?)";
           try (Connection conn = connectionProvider.getConnection();
                PreparedStatement ps = conn.prepareStatement(sql,
Statement.RETURN_GENERATED_KEYS)) {

               ps.setString(1, entity.getUsername());
               ps.setString(2, entity.getPasswordHash());
               ps.executeUpdate();

               try (ResultSet rs = ps.getGeneratedKeys()) {
                   if (rs.next()) {
                       entity.setId(rs.getInt(1));
                   }
               }
           } catch (SQLException e) {
               throw new RuntimeException(e);
           }
       } else {
           // UPDATE
           String sql = "UPDATE Users SET username = ?, password_hash = ? WHERE id =
?";
           try (Connection conn = connectionProvider.getConnection();
                PreparedStatement ps = conn.prepareStatement(sql)) {

               ps.setString(1, entity.getUsername());
               ps.setString(2, entity.getPasswordHash());
               ps.setInt(3, entity.getId());
               ps.executeUpdate();
           } catch (SQLException e) {
               throw new RuntimeException(e);
           }
       }
       return entity;
   }

   @Override
   public void delete(Integer id) {
```

```
        String sql = "DELETE FROM Users WHERE id = ?";
        try (Connection conn = connectionProvider.getConnection();
             PreparedStatement ps = conn.prepareStatement(sql)) {
            ps.setInt(1, id);
            ps.executeUpdate();
        } catch (SQLException e) {
            throw new RuntimeException(e);
        }
    }

    @Override
    public Optional<User> getById(Integer id) {
        String sql = "SELECT id, username, password_hash FROM Users WHERE id = ?";
        try (Connection conn = connectionProvider.getConnection();
             PreparedStatement ps = conn.prepareStatement(sql)) {

            ps.setInt(1, id);
            try (ResultSet rs = ps.executeQuery()) {
                if (rs.next()) {
                    User u = new User(
                            rs.getInt("id"),
                            rs.getString("username"),
                            rs.getString("password_hash")
                    );
                    return Optional.of(u);
                }
            }
        } catch (SQLException e) {
            throw new RuntimeException(e);
        }
        return Optional.empty();
    }

    @Override
    public List<User> getAll() {
        String sql = "SELECT id, username, password_hash FROM Users";
        List<User> users = new ArrayList<>();
        try (Connection conn = connectionProvider.getConnection();
             PreparedStatement ps = conn.prepareStatement(sql);
             ResultSet rs = ps.executeQuery()) {

            while (rs.next()) {
                users.add(new User(
                        rs.getInt("id"),
                        rs.getString("username"),
                        rs.getString("password_hash")
                ));
            }
        } catch (SQLException e) {
            throw new RuntimeException(e);
        }
        return users;
    }
```

```
    public Optional<User> getByUsername(String username) {
        String sql = "SELECT id, username, password_hash FROM Users WHERE username = ?";
        try (Connection conn = connectionProvider.getConnection();
             PreparedStatement ps = conn.prepareStatement(sql)) {

            ps.setString(1, username);
            try (ResultSet rs = ps.executeQuery()) {
                if (rs.next()) {
                    return Optional.of(new User(
                            rs.getInt("id"),
                            rs.getString("username"),
                            rs.getString("password_hash")
                    ));
                }
            }
        } catch (SQLException e) {
            throw new RuntimeException(e);
        }
        return Optional.empty();
    }
}
```

src/com/example/mindmap/repositories/MapElementRepository.java:

```
package com.example.mindmap.repositories;

import com.example.mindmap.entities.MapElement;

import java.util.List;

public interface MapElementRepository {

    List<MapElement> findByMapId(int mapId);

    void save(MapElement element, int mapId);  // INSERT

    void update(MapElement element);           // UPDATE

    void delete(int id);
}
```

src/com/example/mindmap/repositories/MindMapRepository.java:

```
package com.example.mindmap.repositories;

import com.example.mindmap.entities.MindMap;
import com.example.mindmap.entities.User;

import java.util.*;
import java.util.stream.Collectors;

public class MindMapRepository implements Repository<MindMap, Integer> {

    private final Map<Integer, MindMap> storage = new HashMap<>();
```

```
    private int nextId = 1;

    @Override
    public MindMap save(MindMap entity) {
        if (entity.getId() == 0) {
            entity.setId(nextId++);
        }
        storage.put(entity.getId(), entity);
        return entity;
    }

    @Override
    public void delete(Integer id) {
        storage.remove(id);
    }

    @Override
    public Optional<MindMap> getById(Integer id) {
        return Optional.ofNullable(storage.get(id));
    }

    @Override
    public List<MindMap> getAll() {
        return new ArrayList<>(storage.values());
    }

    public List<MindMap> getAllByUser(User user) {
        return storage.values().stream()
                .filter(map -> map.getOwner() != null && map.getOwner().equals(user))
                .collect(Collectors.toList());
    }

    public List<MindMap> getFavorites(User user) {
        return storage.values().stream()
                .filter(map -> map.getOwner() != null
                        && map.getOwner().equals(user)
                        && map.isFavorite())
                .collect(Collectors.toList());
    }
}
```

src/com/example/mindmap/repositories/Repository.java:

```
package com.example.mindmap.repositories;

import java.util.List;
import java.util.Optional;

public interface Repository<T, ID> {
    T save(T entity);
    void delete(ID id);
    Optional<T> getById(ID id);
    List<T> getAll();
}
```

src/com/example/mindmap/repositories/UserRepository.java

```java
package com.example.mindmap.repositories;

import com.example.mindmap.entities.User;

import java.util.*;

public class UserRepository implements Repository<User, Integer> {

    private final Map<Integer, User> storage = new HashMap<>();
    private int nextId = 1;

    @Override
    public User save(User entity) {
        if (entity.getId() == 0) {
            entity.setId(nextId++);
        }
        storage.put(entity.getId(), entity);
        return entity;
    }

    @Override
    public void delete(Integer id) {
        storage.remove(id);
    }

    @Override
    public Optional<User> getById(Integer id) {
        return Optional.ofNullable(storage.get(id));
    }

    @Override
    public List<User> getAll() {
        return new ArrayList<>(storage.values());
    }

    public Optional<User> getByUsername(String username) {
        return storage.values().stream()
                .filter(u -> u.getUsername().equals(username))
                .findFirst();
    }
}
```

src/com/example/mindmap/services/commands/AddNodeCommand.java:

```java
package com.example.mindmap.services.commands;

import com.example.mindmap.entities.MindMap;
import com.example.mindmap.entities.MapElement;
import com.example.mindmap.services.MindMapService;
```

```
import java.util.List;

public class AddNodeCommand implements Command {

    private final MindMap mindMap;
    private final MindMapService mindMapService;
    private final List<MapElement> elements; // список CanvasPanel
    private final float x, y;
    private final String text;

    private MapElement createdElement;

    public AddNodeCommand(
            MindMap mindMap,
            MindMapService mindMapService,
            List<MapElement> elements,
            float x,
            float y,
            String text
    ) {
        this.mindMap = mindMap;
        this.mindMapService = mindMapService;
        this.elements = elements;
        this.x = x;
        this.y = y;
        this.text = text;
    }

    @Override
    public void execute() {
        createdElement = mindMapService.addTextNode(mindMap, x, y, text);
        elements.add(createdElement);
    }

    @Override
    public void undo() {
        elements.remove(createdElement);
        mindMapService.deleteElement(createdElement);
    }
}
```

src/com/example/mindmap/services/commands/AddStrokeCommand.java:

```
package com.example.mindmap.services.commands;

import com.example.mindmap.entities.MindMap;
import com.example.mindmap.services.MindMapService;
import com.example.mindmap.tools.drawing.DrawingLayer;
import com.example.mindmap.tools.drawing.DrawingStroke;

public class AddStrokeCommand implements Command {

    private final DrawingLayer layer;
    private final DrawingStroke stroke;
```

```
    private final MindMapService service;
    private final MindMap map;

    public AddStrokeCommand(DrawingLayer layer,
                            DrawingStroke stroke,
                            MindMapService service,
                            MindMap map) {
        this.layer = layer;
        this.stroke = stroke;
        this.service = service;
        this.map = map;
    }

    @Override
    public void execute() {
        layer.addStroke(stroke);
        service.saveStroke(map, stroke);
    }

    @Override
    public void undo() {
        layer.removeStroke(stroke);
        service.deleteStroke(stroke);
    }
}
```

src/com/example/mindmap/services/commands/Command.java:

```
package com.example.mindmap.services.commands;

public interface Command {
    void execute();
    void undo();
}
```

src/com/example/mindmap/services/commands/CommandManager.java:

```
package com.example.mindmap.services.commands;

import java.util.Stack;

public class CommandManager {

    private final Stack<Command> undoStack = new Stack<>();
    private final Stack<Command> redoStack = new Stack<>();

    public void executeCommand(Command command) {
        command.execute();
        undoStack.push(command);
        redoStack.clear(); // після нової дії redo вже не актуальні
    }

    public boolean canUndo() {
        return !undoStack.isEmpty();
```

```
    }

    public boolean canRedo() {
        return !redoStack.isEmpty();
    }

    public void undo() {
        if (!canUndo()) return;

        Command command = undoStack.pop();
        command.undo();
        redoStack.push(command);
    }

    public void redo() {
        if (!canRedo()) return;

        Command command = redoStack.pop();
        command.execute();
        undoStack.push(command);
    }
}
```

src/com/example/mindmap/services/commands/DeleteNodeCommand.java:

```
package com.example.mindmap.services.commands;

import com.example.mindmap.entities.MapElement;
import com.example.mindmap.entities.MindMap;
import com.example.mindmap.services.MindMapService;

import java.util.List;

public class DeleteNodeCommand implements Command {

    private final MindMap mindMap;
    private final MindMapService mindMapService;
    private final List<MapElement> elements;

    private final MapElement target;
    private float x, y;
    private String text;

    public DeleteNodeCommand(
            MindMap mindMap,
            MindMapService mindMapService,
            List<MapElement> elements,
            MapElement target
    ) {
        this.mindMap = mindMap;
        this.mindMapService = mindMapService;
        this.elements = elements;
        this.target = target;
    }
```

```
    @Override
    public void execute() {
        // Збережемо дані для undo
        x = target.getX();
        y = target.getY();
        text = target.getTextForDisplay();

        elements.remove(target);
        mindMapService.deleteElement(target);
    }

    @Override
    public void undo() {
        MapElement restored = mindMapService.addTextNode(mindMap, x, y, text);
        elements.add(restored);
    }
}
```

src/com/example/mindmap/services/commands/EraseStrokes Command.java:

```
package com.example.mindmap.services.commands;

import com.example.mindmap.entities.MindMap;
import com.example.mindmap.services.MindMapService;
import com.example.mindmap.tools.drawing.DrawingLayer;

import java.util.List;

public class EraseStrokesCommand implements Command {

    private final DrawingLayer layer;
    private final List<DrawingLayer.RemovedStroke> removed;
    private final MindMapService service;
    private final MindMap map;

    public EraseStrokesCommand(DrawingLayer layer,
                               List<DrawingLayer.RemovedStroke> removed,
                               MindMapService service,
                               MindMap map) {
        this.layer = layer;
        this.removed = removed;
        this.service = service;
        this.map = map;
    }

    @Override
    public void execute() {
        layer.removeAll(removed);
        for (DrawingLayer.RemovedStroke r : removed) {
            service.deleteStroke(r.stroke());
        }
    }
```

```
    @Override
    public void undo() {
        layer.restoreAll(removed);
        for (DrawingLayer.RemovedStroke r : removed) {
            service.saveStroke(map, r.stroke());
        }
    }
}
```

src/com/example/mindmap/services/commands/MoveNodeCommand.java:

```
package com.example.mindmap.services.commands;

import com.example.mindmap.entities.MapElement;
import com.example.mindmap.services.MindMapService;

public class MoveNodeCommand implements Command {

    private final MapElement element;
    private final float oldX, oldY;
    private final float newX, newY;
    private final MindMapService mindMapService;

    public MoveNodeCommand(
            MapElement element,
            float oldX,
            float oldY,
            float newX,
            float newY,
            MindMapService mindMapService
    ) {
        this.element = element;
        this.oldX = oldX;
        this.oldY = oldY;
        this.newX = newX;
        this.newY = newY;
        this.mindMapService = mindMapService;
    }

    @Override
    public void execute() {
        element.setX(newX);
        element.setY(newY);
        mindMapService.updateElement(element);
    }

    @Override
    public void undo() {
        element.setX(oldX);
        element.setY(oldY);
        mindMapService.updateElement(element);
    }
}
```

src/com/example/mindmap/services/commands/ResizeNodeCommand.java:

```
package com.example.mindmap.services.commands;

import com.example.mindmap.entities.MapElement;
import com.example.mindmap.services.MindMapService;

public class ResizeNodeCommand implements Command {

   private final MapElement element;
   private final int oldW, oldH;
   private final int newW, newH;
   private final MindMapService service;

   public ResizeNodeCommand(MapElement element, int oldW, int oldH, int newW, int newH,
MindMapService service) {
       this.element = element;
       this.oldW = oldW;
       this.oldH = oldH;
       this.newW = newW;
       this.newH = newH;
       this.service = service;
   }

   @Override
   public void execute() {
       element.setWidthPx(newW);
       element.setHeightPx(newH);
       service.updateElement(element);
   }

   @Override
   public void undo() {
       element.setWidthPx(oldW);
       element.setHeightPx(oldH);
       service.updateElement(element);
   }
}
```

src/com/example/mindmap/services/AuthService.java:

```
package com.example.mindmap.services;

import com.example.mindmap.entities.User;
import com.example.mindmap.repositories.JdbcUserRepository;

import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.util.Optional;

public class AuthService {

```

```
    private final JdbcUserRepository userRepository;

    public AuthService(JdbcUserRepository userRepository) {
        this.userRepository = userRepository;
    }

    // ЛОГІН
    public Optional<User> login(String username, String password) {
        String hashed = hashPassword(password);

        return userRepository.getByUsername(username)
                .filter(u -> u.getPasswordHash().equals(hashed));
    }

    // РЕЄСТРАЦІЯ
    public User register(String username, String password) {
        String hashed = hashPassword(password);          // тут хешуємо
        User user = new User(0, username, hashed);      // зберігаємо вже хеш
        return userRepository.save(user);
    }

    // ПРИВАТНИЙ МЕТОД ДЛЯ ХЕШУВАННЯ
    private String hashPassword(String plainText) {
        if (plainText == null) {
            return null;
        }
        try {
            MessageDigest digest = MessageDigest.getInstance("SHA-256");
            byte[] hashBytes = digest.digest(plainText.getBytes());
            StringBuilder sb = new StringBuilder();
            for (byte b : hashBytes) {
                sb.append(String.format("%02x", b)); // перетворюємо байти в HEX-рядок
            }
            return sb.toString();
        } catch (NoSuchAlgorithmException e) {
            throw new RuntimeException("SHA-256 not supported", e);
        }
    }
}
```

src/com/example/mindmap/services/MindMapService.java:

```
package com.example.mindmap.services;

import com.example.mindmap.db.JdbcConnectionProvider;
import com.example.mindmap.entities.*;
import com.example.mindmap.repositories.JdbcCategoryRepository;
import com.example.mindmap.repositories.JdbcMapElementRepository;
import com.example.mindmap.repositories.JdbcMapStrokeRepository;
import com.example.mindmap.repositories.JdbcMindMapRepository;
import com.example.mindmap.tools.drawing.DrawingStroke;

import java.util.List;
```

```
public class MindMapService {

    private final JdbcMindMapRepository mapRepository;
    private final JdbcMapElementRepository elementRepository;
    private final JdbcMapStrokeRepository strokeRepository;
    private final JdbcCategoryRepository categoryRepository;

    // старий конструктор лишаємо (як у тебе) :contentReference[oaicite:3]{index=3}
    public MindMapService(JdbcMindMapRepository mapRepository,
                          JdbcMapElementRepository elementRepository) {
        this(mapRepository, elementRepository,
                new JdbcMapStrokeRepository(new JdbcConnectionProvider()),
                new JdbcCategoryRepository(new JdbcConnectionProvider()));
    }

    public MindMapService(JdbcMindMapRepository mapRepository,
                          JdbcMapElementRepository elementRepository,
                          JdbcMapStrokeRepository strokeRepository) {
        this(mapRepository, elementRepository,
                strokeRepository,
                new JdbcCategoryRepository(new JdbcConnectionProvider()));
    }

    public MindMapService(JdbcMindMapRepository mapRepository,
                          JdbcMapElementRepository elementRepository,
                          JdbcMapStrokeRepository strokeRepository,
                          JdbcCategoryRepository categoryRepository) {
        this.mapRepository = mapRepository;
        this.elementRepository = elementRepository;
        this.strokeRepository = strokeRepository;
        this.categoryRepository = categoryRepository;
    }

    // --- Мапи ---
    public MindMap createMap(String title, User owner) {
        MindMap map = new MindMap();
        map.setTitle(title);
        map.setOwner(owner);
        mapRepository.save(map);
        return map;
    }

    public void updateMap(MindMap map) {
        mapRepository.save(map);
    }

    public List<MindMap> getMapsByUser(User user) {
        return mapRepository.getAllByUser(user);
    }

    // --- Категорії ---
    public List<Category> getCategoriesByUser(User user) {
        return categoryRepository.getAllByUser(user);
    }
```

```
    public Category createCategory(User user, String title, String colorHex) {
        Category c = new Category();
        c.setTitle(title);
        c.setColor(colorHex);
        c.setOwner(user);
        return categoryRepository.save(c);
    }

    // --- Елементи ---
    public List<MapElement> getElementsForMap(MindMap map) {
        return elementRepository.findByMapId(map.getId());
    }

    public MapElement addTextNode(MindMap map, float x, float y, String text) {
        TextNode node = new TextNode(x, y, text);
        elementRepository.save(node, map.getId());
        return node;
    }

    public MapElement addImageNode(MindMap map, float x, float y, String imagePath) {
        ImageNode node = new ImageNode(x, y, imagePath);
        elementRepository.save(node, map.getId());
        return node;
    }

    public MapElement addImageNode(MindMap map, float x, float y, String imagePath, int
width, int height) {
        ImageNode node = new ImageNode(x, y, imagePath);
        node.setWidthPx(width);
        node.setHeightPx(height);
        elementRepository.save(node, map.getId());
        return node;
    }

    public void updateElement(MapElement element) {
        elementRepository.update(element);
    }

    public void deleteElement(MapElement element) {
        if (element.getId() != 0) elementRepository.delete(element.getId());
    }

    // --- Strokes ---
    public List<DrawingStroke> getStrokesForMap(MindMap map) {
        return strokeRepository.findByMapId(map.getId());
    }

    public void saveStroke(MindMap map, DrawingStroke stroke) {
        strokeRepository.save(map.getId(), stroke);
    }

    public void deleteStroke(DrawingStroke stroke) {
        strokeRepository.deleteById(stroke.getId());
```

```
    }
}
```

src/com/example/mindmap/tools/drawing/DrawingLayer.java:

```
package com.example.mindmap.tools.drawing;

import java.awt.Shape;
import java.awt.geom.Ellipse2D;
import java.awt.geom.Path2D;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;
import java.util.List;

public class DrawingLayer {

    // Глобальні мазки по всій мапі
    private final List<DrawingStroke> strokes = new ArrayList<>();

    public List<DrawingStroke> getStrokes() {
        return strokes;
    }

    public void addStroke(DrawingStroke stroke) {
        strokes.add(stroke);
    }

    public void removeStroke(DrawingStroke stroke) {
        strokes.remove(stroke);
    }

    public void insertAt(int index, DrawingStroke stroke) {
        int i = Math.max(0, Math.min(index, strokes.size()));
        strokes.add(i, stroke);
    }

    public void removeAll(List<RemovedStroke> removed) {
        for (RemovedStroke r : removed) {
            strokes.remove(r.stroke());
        }
    }

    public void restoreAll(List<RemovedStroke> removed) {
        // вставляємо у зростаючому порядку індексів
        List<RemovedStroke> copy = new ArrayList<>(removed);
        copy.sort(Comparator.comparingInt(RemovedStroke::index));
        for (RemovedStroke r : copy) {
            insertAt(r.index(), r.stroke());
        }
    }
```

```
    /**
     * Стирає мазки, які перетинаються з кругом-ластиком у (x,y) з радіусом radiusPx.
     * ВАЖЛИВО: метод одразу видаляє зі списку і повертає, що саме видалили (для
undo).
     */
    public List<RemovedStroke> eraseAt(int x, int y, int radiusPx) {
        if (strokes.isEmpty()) return Collections.emptyList();

        Ellipse2D eraser = new Ellipse2D.Float(
                x - radiusPx, y - radiusPx,
                radiusPx * 2f, radiusPx * 2f
        );

        List<RemovedStroke> removed = new ArrayList<>();

        for (int i = strokes.size() - 1; i >= 0; i--) {
            DrawingStroke s = strokes.get(i);

            if (!s.getBounds().intersects(eraser.getBounds2D())) continue;

            Path2D path = s.buildPath();
            Shape thick = s.buildAwtStroke().createStrokedShape(path);

            if (thick.intersects(eraser.getBounds2D())) {
                removed.add(new RemovedStroke(s, i));
                strokes.remove(i);
            }
        }

        return removed;
    }

    public record RemovedStroke(DrawingStroke stroke, int index) {}
}
```

src/com/example/mindmap/tools/drawing/DrawingStroke.java:

```
package com.example.mindmap.tools.drawing;

import java.awt.*;
import java.awt.geom.Path2D;
import java.awt.geom.Rectangle2D;
import java.util.*;
import java.util.List;

public class DrawingStroke {
    private final String id;

    private final List<PointF> points = new ArrayList<>();
    private final Color color;
    private final float widthPx;
    private final boolean dashed;

    public DrawingStroke(Color color, float widthPx, boolean dashed) {
```

```
        this(UUID.randomUUID().toString(), color, widthPx, dashed);
    }

    public DrawingStroke(String id, Color color, float widthPx, boolean dashed) {
        this.id = id;
        this.color = color;
        this.widthPx = widthPx;
        this.dashed = dashed;
    }

    public String getId() { return id; }
    public Color getColor() { return color; }
    public float getWidthPx() { return widthPx; }
    public boolean isDashed() { return dashed; }

    public void addPoint(float x, float y) {
        points.add(new PointF(x, y));
    }

    public List<PointF> getPoints() {
        return Collections.unmodifiableList(points);
    }

    public boolean isEmpty() {
        return points.size() < 2;
    }

    public Path2D buildPath() {
        Path2D path = new Path2D.Float();
        if (points.isEmpty()) return path;

        PointF p0 = points.get(0);
        path.moveTo(p0.x, p0.y);

        for (int i = 1; i < points.size(); i++) {
            PointF p = points.get(i);
            path.lineTo(p.x, p.y);
        }
        return path;
    }

    public Stroke buildAwtStroke() {
        if (!dashed) {
            return new BasicStroke(widthPx, BasicStroke.CAP_ROUND,
BasicStroke.JOIN_ROUND);
        }
        float dashLen = Math.max(6f, widthPx * 3f);
        return new BasicStroke(
                widthPx,
                BasicStroke.CAP_ROUND,
                BasicStroke.JOIN_ROUND,
                10f,
                new float[]{dashLen, dashLen},
                0f
```

```
        );
    }

    public Rectangle2D getBounds() {
        return buildPath().getBounds2D();
    }

    /** Серіалізація точок у рядок: "x,y;x,y;..." (Locale.US щоб крапка) */
    public String serializePoints() {
        StringBuilder sb = new StringBuilder();
        for (int i = 0; i < points.size(); i++) {
            PointF p = points.get(i);
            if (i > 0) sb.append(';');
            sb.append(String.format(Locale.US, "%f,%f", p.x, p.y));
        }
        return sb.toString();
    }

    public static void deserializePointsInto(DrawingStroke stroke, String data) {
        if (data == null || data.isBlank()) return;
        String[] pairs = data.split(";");
        for (String pair : pairs) {
            String[] xy = pair.split(",");
            if (xy.length != 2) continue;
            try {
                float x = Float.parseFloat(xy[0]);
                float y = Float.parseFloat(xy[1]);
                stroke.addPoint(x, y);
            } catch (NumberFormatException ignored) {}
        }
    }

    public static class PointF {
        public final float x;
        public final float y;

        public PointF(float x, float y) {
            this.x = x;
            this.y = y;
        }
    }
}
```

src/com/example/mindmap/tools/AddImageNodeTool.java:

```
package com.example.mindmap.tools;

import com.example.mindmap.entities.MapElement;

import javax.swing.*;
import java.awt.Graphics2D;
import java.awt.event.KeyEvent;
import java.awt.event.MouseEvent;
import java.io.File;
```

```
public class AddImageNodeTool implements Tool {

   private final EditorContext ctx;

   public AddImageNodeTool(EditorContext ctx) {
       this.ctx = ctx;
   }

   @Override
   public void onMouseClicked(MouseEvent e) {
       JFileChooser chooser = new JFileChooser();
       chooser.setDialogTitle("Choose image");

       int res = chooser.showOpenDialog(null);
       if (res != JFileChooser.APPROVE_OPTION) return;

       File file = chooser.getSelectedFile();
       String path = file.getAbsolutePath();

       MapElement el = ctx.getMindMapService().addImageNode(
               ctx.getMindMap(),
               e.getX(),
               e.getY(),
               path,
               120,
               80
       );

       ctx.getElements().add(el);
       ctx.repaint();
   }

   @Override public void onMousePressed(MouseEvent e) {}
   @Override public void onMouseDragged(MouseEvent e) {}
   @Override public void onMouseReleased(MouseEvent e) {}
   @Override public void onKeyPressed(KeyEvent e) {}
   @Override public void paintOverlay(Graphics2D g2) {}
}
```

src/com/example/mindmap/tools/AddTextNodeTool.java:

```
package com.example.mindmap.tools;

import com.example.mindmap.services.commands.AddNodeCommand;

import java.awt.event.MouseEvent;

public class AddTextNodeTool implements Tool {

   private final EditorContext ctx;

   public AddTextNodeTool(EditorContext ctx) {
       this.ctx = ctx;
```

```
    }

    @Override
    public void onMouseClicked(MouseEvent e) {
        AddNodeCommand cmd = new AddNodeCommand(
                ctx.getMindMap(),
                ctx.getMindMapService(),
                ctx.getElements(),
                e.getX(),
                e.getY(),
                "New idea"
        );
        ctx.getCommandManager().executeCommand(cmd);
        ctx.repaint();
    }
}
```

src/com/example/mindmap/tools/DefaultToolFactory.java:

```
package com.example.mindmap.tools;

public class DefaultToolFactory implements ToolFactory {

    @Override
    public Tool createSelectTool(EditorContext ctx) {
        return new SelectTool(ctx);
    }

    @Override
    public Tool createAddTextNodeTool(EditorContext ctx) {
        return new AddTextNodeTool(ctx);
    }

    @Override
    public Tool createAddImageNodeTool(EditorContext ctx) {
        return new AddImageNodeTool(ctx);
    }

    @Override
    public Tool createPenTool(EditorContext ctx) {
        return new PenTool(ctx, false);
    }

    @Override
    public Tool createDashedPenTool(EditorContext ctx) {
        return new PenTool(ctx, true);
    }

    @Override
    public Tool createEraserTool(EditorContext ctx) {
        return new EraserTool(ctx);
    }
}
```

src/com/example/mindmap/tools/EditorContext.java:

```
package com.example.mindmap.tools;

import com.example.mindmap.entities.MapElement;
import com.example.mindmap.entities.MindMap;
import com.example.mindmap.services.MindMapService;
import com.example.mindmap.services.commands.CommandManager;
import com.example.mindmap.tools.drawing.DrawingLayer;

import javax.swing.*;
import java.awt.*;
import java.util.List;

public class EditorContext {

   private final JComponent canvas;
   private final MindMap mindMap;
   private final MindMapService mindMapService;
   private final CommandManager commandManager;
   private final List<MapElement> elements;

   // шар мазків
   private final DrawingLayer drawingLayer;

   // налаштування пензля/ластика
   private Color brushColor = Color.BLACK;
   private float brushWidthPx = 3f;
   private int eraserRadiusPx = 10;

   private MapElement selected;

   // Старий конструктор (для сумісності)
   public EditorContext(JComponent canvas,
                        MindMap mindMap,
                        MindMapService mindMapService,
                        CommandManager commandManager,
                        List<MapElement> elements) {
       this(canvas, mindMap, mindMapService, commandManager, elements, new
DrawingLayer());
   }

   // конструктор з drawingLayer
   public EditorContext(JComponent canvas,
                        MindMap mindMap,
                        MindMapService mindMapService,
                        CommandManager commandManager,
                        List<MapElement> elements,
                        DrawingLayer drawingLayer) {
       this.canvas = canvas;
       this.mindMap = mindMap;
       this.mindMapService = mindMapService;
       this.commandManager = commandManager;
```

```
        this.elements = elements;
        this.drawingLayer = drawingLayer;
    }

    public MindMap getMindMap() { return mindMap; }
    public MindMapService getMindMapService() { return mindMapService; }
    public CommandManager getCommandManager() { return commandManager; }
    public List<MapElement> getElements() { return elements; }
    public DrawingLayer getDrawingLayer() { return drawingLayer; }

    public MapElement getSelected() { return selected; }
    public void setSelected(MapElement el) { this.selected = el; repaint(); }

    public void repaint() { canvas.repaint(); }
    public void requestFocus() { canvas.requestFocusInWindow(); }

    // brush settings
    public Color getBrushColor() { return brushColor; }
    public void setBrushColor(Color brushColor) { this.brushColor = brushColor; }

    public float getBrushWidthPx() { return brushWidthPx; }
    public void setBrushWidthPx(float brushWidthPx) { this.brushWidthPx = Math.max(1f,
brushWidthPx); }

    public int getEraserRadiusPx() { return eraserRadiusPx; }
    public void setEraserRadiusPx(int eraserRadiusPx) { this.eraserRadiusPx =
Math.max(3, eraserRadiusPx); }

    public MapElement findElementAt(int x, int y) {
        for (int i = elements.size() - 1; i >= 0; i--) {
            MapElement el = elements.get(i);
            Rectangle r = new Rectangle((int) el.getX(), (int) el.getY(),
el.getWidthPx(), el.getHeightPx());
            if (r.contains(x, y)) return el;
        }
        return null;
    }
}
```

src/com/example/mindmap/tools/EraserTool.java:

```
package com.example.mindmap.tools;

import com.example.mindmap.services.commands.EraseStrokesCommand;
import com.example.mindmap.tools.drawing.DrawingLayer;

import java.awt.event.KeyEvent;
import java.awt.event.MouseEvent;
import java.util.ArrayList;
import java.util.HashSet;
import java.util.List;
import java.util.Set;

public class EraserTool implements Tool {
```

```
    private final EditorContext ctx;

    private final List<DrawingLayer.RemovedStroke> removedDuringDrag = new
ArrayList<>();
    private final Set<String> removedIds = new HashSet<>();

    private int lastX = -1;
    private int lastY = -1;

    public EraserTool(EditorContext ctx) {
        this.ctx = ctx;
    }

    @Override
    public void onMousePressed(MouseEvent e) {
        removedDuringDrag.clear();
        removedIds.clear();
        eraseAt(e.getX(), e.getY());
        ctx.repaint();
    }

    @Override
    public void onMouseDragged(MouseEvent e) {
        if (Math.abs(e.getX() - lastX) + Math.abs(e.getY() - lastY) < 2) return;
        eraseAt(e.getX(), e.getY());
        ctx.repaint();
    }

    @Override
    public void onMouseReleased(MouseEvent e) {
        if (removedDuringDrag.isEmpty()) return;

        // щоб redo працював: повертаємо назад, а команда знову видалить
        ctx.getDrawingLayer().restoreAll(removedDuringDrag);

        EraseStrokesCommand cmd = new EraseStrokesCommand(
                ctx.getDrawingLayer(),
                new ArrayList<>(removedDuringDrag),
                ctx.getMindMapService(),
                ctx.getMindMap()
        );
        ctx.getCommandManager().executeCommand(cmd);

        removedDuringDrag.clear();
        removedIds.clear();
        lastX = lastY = -1;
        ctx.repaint();
    }

    private void eraseAt(int x, int y) {
        lastX = x;
        lastY = y;
```

```
        int radius = ctx.getEraserRadiusPx();
        List<DrawingLayer.RemovedStroke> removedNow = ctx.getDrawingLayer().eraseAt(x,
y, radius);

        for (DrawingLayer.RemovedStroke r : removedNow) {
            if (removedIds.add(r.stroke().getId())) {
                removedDuringDrag.add(r);
            }
        }
    }

    @Override public void onMouseClicked(MouseEvent e) {}
    @Override public void onKeyPressed(KeyEvent e) {}
}
```

src/com/example/mindmap/tools/PenTool.java:

```
package com.example.mindmap.tools;

import com.example.mindmap.services.commands.AddStrokeCommand;
import com.example.mindmap.tools.drawing.DrawingStroke;

import java.awt.Graphics2D;
import java.awt.event.KeyEvent;
import java.awt.event.MouseEvent;

public class PenTool implements Tool {

    private final EditorContext ctx;
    private final boolean dashed;

    private DrawingStroke current;

    public PenTool(EditorContext ctx, boolean dashed) {
        this.ctx = ctx;
        this.dashed = dashed;
    }

    @Override
    public void onMousePressed(MouseEvent e) {
        current = new DrawingStroke(ctx.getBrushColor(), ctx.getBrushWidthPx(), dashed);
        current.addPoint(e.getX(), e.getY());
        ctx.repaint();
    }

    @Override
    public void onMouseDragged(MouseEvent e) {
        if (current == null) return;
        current.addPoint(e.getX(), e.getY());
        ctx.repaint();
    }

    @Override
    public void onMouseReleased(MouseEvent e) {
```

```
        if (current == null) return;

        if (!current.isEmpty()) {
            AddStrokeCommand cmd = new AddStrokeCommand(
                    ctx.getDrawingLayer(),
                    current,
                    ctx.getMindMapService(),
                    ctx.getMindMap()
            );
            ctx.getCommandManager().executeCommand(cmd);
        }

        current = null;
        ctx.repaint();
    }

    @Override
    public void paintOverlay(Graphics2D g2) {
        if (current == null || current.isEmpty()) return;

        g2.setColor(current.getColor());
        g2.setStroke(current.buildAwtStroke());
        g2.draw(current.buildPath());
    }

    @Override public void onMouseClicked(MouseEvent e) {}
    @Override public void onKeyPressed(KeyEvent e) {}
}
```

src/com/example/mindmap/tools/SelectTool.java:

```
package com.example.mindmap.tools;

import com.example.mindmap.entities.MapElement;
import com.example.mindmap.services.commands.MoveNodeCommand;
import com.example.mindmap.services.commands.ResizeNodeCommand;

import java.awt.*;
import java.awt.event.KeyEvent;
import java.awt.event.MouseEvent;

public class SelectTool implements Tool {

    private enum DragMode { NONE, MOVE, RESIZE_RIGHT, RESIZE_BOTTOM, RESIZE_CORNER }

    private final EditorContext ctx;

    private MapElement active;
    private DragMode dragMode = DragMode.NONE;

    // move
    private int offsetX, offsetY;
    private float startX, startY;
```

```
    // resize
    private int startW, startH;

    private static final int RESIZE_MARGIN = 7;

    public SelectTool(EditorContext ctx) {
        this.ctx = ctx;
    }

    @Override
    public void onMousePressed(MouseEvent e) {
        MapElement hit = ctx.findElementAt(e.getX(), e.getY());
        if (hit == null) {
            ctx.setSelected(null);
            active = null;
            dragMode = DragMode.NONE;
            return;
        }

        ctx.setSelected(hit);
        ctx.requestFocus();

        active = hit;
        startX = hit.getX();
        startY = hit.getY();
        startW = hit.getWidthPx();
        startH = hit.getHeightPx();

        Rectangle r = new Rectangle((int) hit.getX(), (int) hit.getY(),
hit.getWidthPx(), hit.getHeightPx());
        dragMode = detectDragMode(r, e.getX(), e.getY());

        offsetX = (int) (e.getX() - hit.getX());
        offsetY = (int) (e.getY() - hit.getY());
    }

    @Override
    public void onMouseDragged(MouseEvent e) {
        if (active == null) return;

        switch (dragMode) {
            case MOVE -> {
                active.setX(e.getX() - offsetX);
                active.setY(e.getY() - offsetY);
            }
            case RESIZE_RIGHT -> active.setWidthPx(e.getX() - (int) active.getX());
            case RESIZE_BOTTOM -> active.setHeightPx(e.getY() - (int) active.getY());
            case RESIZE_CORNER -> {
                active.setWidthPx(e.getX() - (int) active.getX());
                active.setHeightPx(e.getY() - (int) active.getY());
            }
            default -> {}
        }
```

```
        ctx.repaint();
    }

    @Override
    public void onMouseReleased(MouseEvent e) {
        if (active == null) return;

        // commit move
        float endX = active.getX();
        float endY = active.getY();
        if (dragMode == DragMode.MOVE && (endX != startX || endY != startY)) {
            MoveNodeCommand cmd = new MoveNodeCommand(active, startX, startY, endX,
endY, ctx.getMindMapService());
            ctx.getCommandManager().executeCommand(cmd);
        }

        // commit resize
        int endW = active.getWidthPx();
        int endH = active.getHeightPx();
        if ((dragMode == DragMode.RESIZE_RIGHT || dragMode == DragMode.RESIZE_BOTTOM ||
dragMode == DragMode.RESIZE_CORNER)
                && (endW != startW || endH != startH)) {
            ResizeNodeCommand cmd = new ResizeNodeCommand(active, startW, startH, endW,
endH, ctx.getMindMapService());
            ctx.getCommandManager().executeCommand(cmd);
        }

        active = null;
        dragMode = DragMode.NONE;
    }

    private DragMode detectDragMode(Rectangle r, int x, int y) {
        boolean nearRight = Math.abs(x - (r.x + r.width)) <= RESIZE_MARGIN;
        boolean nearBottom = Math.abs(y - (r.y + r.height)) <= RESIZE_MARGIN;

        if (nearRight && nearBottom) return DragMode.RESIZE_CORNER;
        if (nearRight) return DragMode.RESIZE_RIGHT;
        if (nearBottom) return DragMode.RESIZE_BOTTOM;
        return DragMode.MOVE;
    }
    @Override public void onKeyPressed(KeyEvent e) {}
    @Override public void paintOverlay(Graphics2D g2) {}

    @Override
    public void onMouseClicked(MouseEvent e) {
        if (e.getClickCount() != 2) return;

        MapElement hit = ctx.findElementAt(e.getX(), e.getY());
        if (!(hit instanceof com.example.mindmap.entities.TextNode)) return;

        String oldText = hit.getTextForDisplay();
        if (oldText == null) oldText = "";

        String newText = javax.swing.JOptionPane.showInputDialog(
```

```
                null, "Edit text:", oldText
        );
        if (newText == null) return;

        hit.setTextForDisplay(newText.trim());
        ctx.getMindMapService().updateElement(hit);
        ctx.repaint();
    }

}
```

src/com/example/mindmap/tools/Tool.java:

```
package com.example.mindmap.tools;

import java.awt.Graphics2D;
import java.awt.event.KeyEvent;
import java.awt.event.MouseEvent;

public interface Tool {
    default void onMousePressed(MouseEvent e) {}
    default void onMouseDragged(MouseEvent e) {}
    default void onMouseReleased(MouseEvent e) {}
    default void onMouseClicked(MouseEvent e) {}
    default void onKeyPressed(KeyEvent e) {}

    default void paintOverlay(Graphics2D g2) {}
}
```

src/com/example/mindmap/tools/ToolFactory.java:

```
package com.example.mindmap.tools;

public interface ToolFactory {
    Tool createSelectTool(EditorContext ctx);
    Tool createAddTextNodeTool(EditorContext ctx);
    Tool createAddImageNodeTool(EditorContext ctx);

    Tool createPenTool(EditorContext ctx);
    Tool createDashedPenTool(EditorContext ctx);
    Tool createEraserTool(EditorContext ctx);
}
```

src/com/example/mindmap/ui/dashboard/ColorUtil.java:

```
package com.example.mindmap.ui.dashboard;

import java.awt.*;

public class ColorUtil {
    public static String toHex(Color c) {
        return String.format("#%02x%02x%02x", c.getRed(), c.getGreen(), c.getBlue());
    }
```

```
    public static Color fromHex(String hex, Color fallback) {
        try {
            if (hex == null || hex.isBlank()) return fallback;
            return Color.decode(hex);
        } catch (Exception e) {
            return fallback;
        }
    }
}
```

src/com/example/mindmap/ui/dashboard/DashboardMediator.java:

```
package com.example.mindmap.ui.dashboard;

import com.example.mindmap.entities.Category;
import com.example.mindmap.entities.MindMap;

import java.awt.*;

public interface DashboardMediator {
    void onMapSelected(MindMap map);

    void onFavoriteToggled(boolean favorite);
    void onCategorySelected(Category category);

    void onCreateCategoryRequested(String title, Color color);

    // збереження назви/опису
    void onMetadataSaved(String newTitle, String newDescription);

    void onSearchChanged(String query);
    void onSortModeChanged(SortMode mode);

    void refresh();
}
```

src/com/example/mindmap/ui/dashboard/DashboardMediatorImpl.java:

```
package com.example.mindmap.ui.dashboard;

import com.example.mindmap.entities.Category;
import com.example.mindmap.entities.MindMap;
import com.example.mindmap.entities.User;
import com.example.mindmap.services.MindMapService;

import javax.swing.*;
import java.awt.*;
import java.util.ArrayList;
import java.util.Comparator;
import java.util.List;
import java.util.Locale;
import java.util.stream.Collectors;
```

```
public class DashboardMediatorImpl implements DashboardMediator {

   private final User user;
   private final MindMapService service;

   private MapsListPanel mapsListPanel;
   private MapPropertiesPanel propertiesPanel;
   @SuppressWarnings("unused")
   private SearchSortPanel searchSortPanel;

   private List<MindMap> allMaps = new ArrayList<>();
   private List<Category> categories = new ArrayList<>();

   private MindMap selected;
   private String search = "";
   private SortMode sortMode = SortMode.BY_CATEGORY;

   public DashboardMediatorImpl(User user, MindMapService service) {
       this.user = user;
       this.service = service;
   }

   public void register(MapsListPanel listPanel, MapPropertiesPanel propsPanel,
SearchSortPanel filterPanel) {
       this.mapsListPanel = listPanel;
       this.propertiesPanel = propsPanel;
       this.searchSortPanel = filterPanel;

       listPanel.setMediator(this);
       propsPanel.setMediator(this);
       filterPanel.setMediator(this);

       refresh();
   }

   @Override
   public void refresh() {
       int selectedId = (selected != null) ? selected.getId() : -1;

       categories = service.getCategoriesByUser(user);
       allMaps = service.getMapsByUser(user);

       // Пере-знайдемо selected після перечитування з БД
       if (selectedId != -1) {
           selected = allMaps.stream().filter(m -> m.getId() ==
selectedId).findFirst().orElse(null);
       }

       applyFilterAndSort();
       propertiesPanel.showMap(selected, categories);
   }

   @Override
   public void onMapSelected(MindMap map) {
```

```
        selected = map;
        propertiesPanel.showMap(selected, categories);
    }

    @Override
    public void onFavoriteToggled(boolean favorite) {
        if (selected == null) return;

        if (favorite) selected.markAsFavorite();
        else selected.unmarkFavorite();

        service.updateMap(selected);

        int keepId = selected.getId();
        refresh();
        mapsListPanel.selectById(keepId);
    }

    @Override
    public void onCategorySelected(Category category) {
        if (selected == null) return;

        selected.setCategory(category);
        service.updateMap(selected);

        int keepId = selected.getId();
        refresh();
        mapsListPanel.selectById(keepId);
    }

    @Override
    public void onCreateCategoryRequested(String title, Color color) {
        String hex = ColorUtil.toHex(color);
        Category created = service.createCategory(user, title, hex);

        if (selected != null) {
            selected.setCategory(created);
            service.updateMap(selected);
        }

        int keepId = (selected != null ? selected.getId() : -1);
        refresh();
        if (keepId != -1) mapsListPanel.selectById(keepId);
    }

    @Override
    public void onMetadataSaved(String newTitle, String newDescription) {
        if (selected == null) return;

        String t = (newTitle == null) ? "" : newTitle.trim();
        if (t.isBlank()) {
            JOptionPane.showMessageDialog(propertiesPanel, "Назва мапи обов'язкова");
            return;
        }
```

```
        selected.setTitle(t);
        selected.setDescription(newDescription == null ? "" : newDescription.trim());

        service.updateMap(selected);

        int keepId = selected.getId();
        refresh();
        mapsListPanel.selectById(keepId);
    }

    @Override
    public void onSearchChanged(String query) {
        search = (query == null) ? "" : query.trim();
        applyFilterAndSort();
    }

    @Override
    public void onSortModeChanged(SortMode mode) {
        sortMode = (mode == null) ? SortMode.BY_CATEGORY : mode;
        applyFilterAndSort();
    }

    private void applyFilterAndSort() {
        String q = search.toLowerCase(Locale.ROOT);

        List<MindMap> filtered = allMaps.stream()
                .filter(m -> q.isBlank() || (m.getTitle() != null &&
m.getTitle().toLowerCase(Locale.ROOT).contains(q)))
                .collect(Collectors.toList());

        Comparator<MindMap> byName =
                Comparator.comparing(m -> m.getTitle() == null ? "" : m.getTitle(),
String.CASE_INSENSITIVE_ORDER);

        Comparator<MindMap> byCategory =
                Comparator.comparing((MindMap m) -> {
                            if (m.getCategory() == null || m.getCategory().getTitle() ==
null) return "~~~";
                            return m.getCategory().getTitle().toLowerCase(Locale.ROOT);
                        })
                        .thenComparing(byName);

        Comparator<MindMap> base = (sortMode == SortMode.BY_NAME) ? byName : byCategory;


filtered.sort(Comparator.comparing(MindMap::isFavorite).reversed().thenComparing(base)
);

        mapsListPanel.setMaps(filtered);
    }
}
```

src/com/example/mindmap/ui/dashboard/DotIcon.java:

```
package com.example.mindmap.ui.dashboard;

import javax.swing.*;
import java.awt.*;

public class DotIcon implements Icon {
   private final int size;
   private final Color color;

   public DotIcon(Color color, int size) {
       this.color = color;
       this.size = size;
   }

   @Override public int getIconWidth() { return size; }
   @Override public int getIconHeight() { return size; }

   @Override
   public void paintIcon(Component c, Graphics g, int x, int y) {
       Graphics2D g2 = (Graphics2D) g.create();
       g2.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
RenderingHints.VALUE_ANTIALIAS_ON);
       g2.setColor(color);
       g2.fillOval(x, y, size, size);
       g2.setColor(new Color(0, 0, 0, 60));
       g2.drawOval(x, y, size, size);
       g2.dispose();
   }
}
```

src/com/example/mindmap/ui/dashboard/MapPropertiesPanel.java:

```
package com.example.mindmap.ui.dashboard;

import com.example.mindmap.entities.Category;
import com.example.mindmap.entities.MindMap;

import javax.swing.*;
import java.awt.*;
import java.util.List;

public class MapPropertiesPanel extends JPanel {

   private DashboardMediator mediator;

   private final JTextField titleField = new JTextField(18);
   private final JTextArea descriptionArea = new JTextArea(6, 18);
   private final JButton saveMetaBtn = new JButton("Save");

   private final JToggleButton favBtn = new JToggleButton("☆ Important");
   private final JComboBox<Category> categoryBox = new JComboBox<>();
   private final JButton newCategoryBtn = new JButton("New category");
```

```
    private boolean internalUpdate = false;

    private final Category NO_CATEGORY;

    public MapPropertiesPanel() {
        super(new BorderLayout());

        NO_CATEGORY = new Category();
        NO_CATEGORY.setId(0);
        NO_CATEGORY.setTitle("No category");
        NO_CATEGORY.setColor("#B0B0B0");

        JPanel card = new JPanel();
        card.setLayout(new BoxLayout(card, BoxLayout.Y_AXIS));
        card.setBorder(BorderFactory.createEmptyBorder(12, 12, 12, 12));
        card.setBackground(Color.WHITE);

        JLabel header = new JLabel("Map properties");
        header.setFont(new Font("Segoe UI", Font.BOLD, 16));

        // Title
        JLabel titleLbl = new JLabel("Title:");
        titleLbl.setFont(new Font("Segoe UI", Font.PLAIN, 12));

        // Description
        JLabel descLbl = new JLabel("Description:");
        descLbl.setFont(new Font("Segoe UI", Font.PLAIN, 12));

        descriptionArea.setLineWrap(true);
        descriptionArea.setWrapStyleWord(true);
        JScrollPane descScroll = new JScrollPane(descriptionArea);

        // Save meta button
        saveMetaBtn.setFocusPainted(false);
        saveMetaBtn.setCursor(Cursor.getPredefinedCursor(Cursor.HAND_CURSOR));
        saveMetaBtn.addActionListener(e -> {
            if (internalUpdate) return;
            if (mediator != null) mediator.onMetadataSaved(titleField.getText(),
descriptionArea.getText());
        });

        // Favorite
        styleToggle(favBtn);
        favBtn.addActionListener(e -> {
            if (internalUpdate) return;
            if (mediator != null) mediator.onFavoriteToggled(favBtn.isSelected());
        });

        // Category combobox rendering (with colored dot)
        categoryBox.setRenderer(new DefaultListCellRenderer() {
            @Override
            public Component getListCellRendererComponent(JList<?> list, Object value,
int index,
```

```
                                                    boolean isSelected, boolean
cellHasFocus) {
                JLabel lbl = (JLabel) super.getListCellRendererComponent(list, value,
index, isSelected, cellHasFocus);
                if (value instanceof Category c) {
                    lbl.setText(c.getTitle());
                    Color col = ColorUtil.fromHex(c.getColor(), new Color(140, 140,
140));
                    lbl.setIcon(new DotIcon(col, 10));
                }
                return lbl;
            }
        });

        categoryBox.addActionListener(e -> {
            if (internalUpdate) return;
            Category selected = (Category) categoryBox.getSelectedItem();
            if (selected != null && selected.getId() == 0) selected = null;
            if (mediator != null) mediator.onCategorySelected(selected);
        });

        // Create category dialog
        newCategoryBtn.setFocusPainted(false);
        newCategoryBtn.setCursor(Cursor.getPredefinedCursor(Cursor.HAND_CURSOR));
        newCategoryBtn.addActionListener(e -> {
            if (mediator == null) return;

            JTextField name = new JTextField(18);
            int ok = JOptionPane.showConfirmDialog(this, name, "Category name",
JOptionPane.OK_CANCEL_OPTION);
            if (ok != JOptionPane.OK_OPTION) return;

            String title = name.getText().trim();
            if (title.isBlank()) {
                JOptionPane.showMessageDialog(this, "Category title is required");
                return;
            }

            Color chosen = JColorChooser.showDialog(this, "Choose category color", new
Color(88, 101, 242));
            if (chosen == null) return;

            mediator.onCreateCategoryRequested(title, chosen);
        });

        // Layout
        card.add(header);
        card.add(Box.createVerticalStrut(10));

        card.add(titleLbl);
        card.add(titleField);
        card.add(Box.createVerticalStrut(8));

        card.add(descLbl);
```

```
        card.add(descScroll);
        card.add(Box.createVerticalStrut(8));

        JPanel saveRow = new JPanel(new FlowLayout(FlowLayout.LEFT, 0, 0));
        saveRow.setOpaque(false);
        saveRow.add(saveMetaBtn);
        card.add(saveRow);

        card.add(Box.createVerticalStrut(12));
        card.add(favBtn);
        card.add(Box.createVerticalStrut(12));

        JPanel catRow = new JPanel(new FlowLayout(FlowLayout.LEFT, 8, 0));
        catRow.setOpaque(false);
        catRow.add(new JLabel("Category:"));
        catRow.add(categoryBox);
        catRow.add(newCategoryBtn);

        card.add(catRow);

        add(card, BorderLayout.NORTH);
        setBackground(new Color(245, 247, 252));

        // default state: no map selected
        showMap(null, List.of());
    }

    public void setMediator(DashboardMediator mediator) {
        this.mediator = mediator;
    }

    public void setCategories(List<Category> categories) {
        internalUpdate = true;

        categoryBox.removeAllItems();
        categoryBox.addItem(NO_CATEGORY);
        for (Category c : categories) categoryBox.addItem(c);

        internalUpdate = false;
    }

    public void showMap(MindMap map, List<Category> categories) {
        internalUpdate = true;

        if (map == null) {
            titleField.setText("");
            descriptionArea.setText("");

            titleField.setEnabled(false);
            descriptionArea.setEnabled(false);
            saveMetaBtn.setEnabled(false);

            favBtn.setEnabled(false);
            favBtn.setSelected(false);
```

```
            favBtn.setText("☆ Important");

            categoryBox.setEnabled(false);
            newCategoryBtn.setEnabled(false);

            setCategories(categories);
            categoryBox.setSelectedItem(NO_CATEGORY);

            internalUpdate = false;
            return;
        }

        titleField.setEnabled(true);
        descriptionArea.setEnabled(true);
        saveMetaBtn.setEnabled(true);

        titleField.setText(map.getTitle() == null ? "" : map.getTitle());
        descriptionArea.setText(map.getDescription() == null ? "" :
map.getDescription());

        favBtn.setEnabled(true);
        favBtn.setSelected(map.isFavorite());
        favBtn.setText(map.isFavorite() ? "★ Important" : "☆ Important");

        categoryBox.setEnabled(true);
        newCategoryBtn.setEnabled(true);

        setCategories(categories);

        Category current = map.getCategory();
        if (current == null) {
            categoryBox.setSelectedItem(NO_CATEGORY);
        } else {
            Category found = null;
            for (int i = 0; i < categoryBox.getItemCount(); i++) {
                Category item = categoryBox.getItemAt(i);
                if (item != null && item.getId() == current.getId()) {
                    found = item;
                    break;
                }
            }
            categoryBox.setSelectedItem(found != null ? found : NO_CATEGORY);
        }

        internalUpdate = false;
    }

    private static void styleToggle(AbstractButton b) {
        b.setBackground(new Color(245, 245, 245));
        b.setFocusPainted(false);
        b.setBorder(BorderFactory.createEmptyBorder(8, 12, 8, 12));
        b.setCursor(Cursor.getPredefinedCursor(Cursor.HAND_CURSOR));
    }
}
```

src/com/example/mindmap/ui/dashboard/MapsListPanel.java:

```
package com.example.mindmap.ui.dashboard;

import com.example.mindmap.entities.MindMap;

import javax.swing.*;
import java.awt.*;
import java.util.List;

public class MapsListPanel extends JPanel {

    private DashboardMediator mediator;

    private final DefaultListModel<MindMap> model = new DefaultListModel<>();
    private final JList<MindMap> list = new JList<>(model);

    public MapsListPanel() {
        super(new BorderLayout());
        list.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
        list.setCellRenderer(new MindMapListCellRenderer());
        add(new JScrollPane(list), BorderLayout.CENTER);

        list.addListSelectionListener(e -> {
            if (e.getValueIsAdjusting()) return;
            if (mediator != null) mediator.onMapSelected(list.getSelectedValue());
        });
    }

    public void setMediator(DashboardMediator mediator) {
        this.mediator = mediator;
    }

    public void setMaps(List<MindMap> maps) {
        model.clear();
        for (MindMap m : maps) model.addElement(m);
    }

    public JList<MindMap> getList() {
        return list;
    }

    public MindMap getSelected() {
        return list.getSelectedValue();
    }

    public void selectById(int id) {
        for (int i = 0; i < model.size(); i++) {
            MindMap m = model.get(i);
            if (m.getId() == id) {
                list.setSelectedIndex(i);
                list.ensureIndexIsVisible(i);
                break;
```

```
            }
        }
    }
}
```

src/com/example/mindmap/ui/dashboard/MindMapListCellRenderer.java:

```
package com.example.mindmap.ui.dashboard;

import com.example.mindmap.entities.Category;
import com.example.mindmap.entities.MindMap;

import javax.swing.*;
import java.awt.*;

public class MindMapListCellRenderer extends DefaultListCellRenderer {

    @Override
    public Component getListCellRendererComponent(JList<?> list, Object value, int
index,
                                                  boolean isSelected, boolean
cellHasFocus) {
        JLabel lbl = (JLabel) super.getListCellRendererComponent(list, value, index,
isSelected, cellHasFocus);

        if (value instanceof MindMap map) {
            String star = map.isFavorite() ? "★ " : "☆ ";
            Category cat = map.getCategory();

            lbl.setText(star + map.getTitle() + (cat != null ? "  [" + cat.getTitle() +
"]" : ""));
            lbl.setFont(lbl.getFont().deriveFont(map.isFavorite() ? Font.BOLD :
Font.PLAIN));

            if (!isSelected) {
                lbl.setBackground(map.isFavorite() ? new Color(255, 250, 220) :
Color.WHITE);
            }

            if (cat != null) {
                Color c = ColorUtil.fromHex(cat.getColor(), new Color(140, 140, 140));
                lbl.setIcon(new DotIcon(c, 10));
            } else {
                lbl.setIcon(null);
            }
        }

        lbl.setBorder(BorderFactory.createEmptyBorder(6, 10, 6, 10));
        return lbl;
    }
}
```

src/com/example/mindmap/ui/dashboard/SearchSortPanel.java:

```
package com.example.mindmap.ui.dashboard;

import javax.swing.*;
import javax.swing.event.DocumentEvent;
import javax.swing.event.DocumentListener;
import java.awt.*;

public class SearchSortPanel extends JPanel {

   private DashboardMediator mediator;

   private final JTextField searchField = new JTextField(18);
   private final JComboBox<SortMode> sortBox = new JComboBox<>(SortMode.values());

   public SearchSortPanel() {
       super(new FlowLayout(FlowLayout.LEFT, 10, 6));

       add(new JLabel("Search:"));
       add(searchField);

       add(new JLabel("Sort:"));
       sortBox.setRenderer(new DefaultListCellRenderer() {
           @Override
           public Component getListCellRendererComponent(JList<?> list, Object value,
int index,
                                                         boolean isSelected, boolean
cellHasFocus) {
               super.getListCellRendererComponent(list, value, index, isSelected,
cellHasFocus);
               if (value == SortMode.BY_NAME) setText("By name");
               if (value == SortMode.BY_CATEGORY) setText("By category");
               return this;
           }
       });
       sortBox.setSelectedItem(SortMode.BY_CATEGORY);
       add(sortBox);

       searchField.getDocument().addDocumentListener(new DocumentListener() {
           private void fire() {
               if (mediator != null) mediator.onSearchChanged(searchField.getText());
           }
           @Override public void insertUpdate(DocumentEvent e) { fire(); }
           @Override public void removeUpdate(DocumentEvent e) { fire(); }
           @Override public void changedUpdate(DocumentEvent e) { fire(); }
       });

       sortBox.addActionListener(e -> {
           if (mediator != null) mediator.onSortModeChanged((SortMode)
sortBox.getSelectedItem());
       });
   }

   public void setMediator(DashboardMediator mediator) {
       this.mediator = mediator;
```

```
    }
}
```

src/com/example/mindmap/ui/dashboard/SortMode.java:

```
package com.example.mindmap.ui.dashboard;

public enum SortMode {
    BY_NAME,
    BY_CATEGORY
}
```

src/com/example/mindmap/ui/DashboardForm.java:

```
package com.example.mindmap.ui;

import com.example.mindmap.entities.MindMap;
import com.example.mindmap.entities.User;
import com.example.mindmap.services.AuthService;
import com.example.mindmap.services.MindMapService;
import com.example.mindmap.ui.dashboard.*;

import javax.swing.*;
import java.awt.*;

public class DashboardForm extends JFrame {

    private final User user;
    private final AuthService authService;
    private final MindMapService mindMapService;

    private final DashboardMediatorImpl mediator;

    private final MapsListPanel mapsListPanel = new MapsListPanel();
    private final MapPropertiesPanel propertiesPanel = new MapPropertiesPanel();
    private final SearchSortPanel searchSortPanel = new SearchSortPanel();

    public DashboardForm(User user, AuthService authService, MindMapService
mindMapService) {
        super("Mind Map - Dashboard");
        this.user = user;
        this.authService = authService;
        this.mindMapService = mindMapService;

        this.mediator = new DashboardMediatorImpl(user, mindMapService);

        initComponents();
        mediator.register(mapsListPanel, propertiesPanel, searchSortPanel);
    }

    private void initComponents() {
        setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
        setSize(980, 520);
```

```
        setLocationRelativeTo(null);

        JPanel mainPanel = new JPanel(new BorderLayout());
        mainPanel.setBackground(new Color(245, 247, 252));
        add(mainPanel);

        // ===== Top bar =====
        JPanel topBar = new JPanel(new BorderLayout());
        topBar.setBackground(Color.WHITE);
        topBar.setBorder(BorderFactory.createEmptyBorder(10, 16, 10, 16));
        mainPanel.add(topBar, BorderLayout.NORTH);

        JPanel left = new JPanel();
        left.setOpaque(false);
        left.setLayout(new BoxLayout(left, BoxLayout.Y_AXIS));

        JLabel hi = new JLabel("Hi, " + user.getUsername());
        hi.setFont(new Font("Segoe UI", Font.PLAIN, 13));
        hi.setForeground(new Color(120, 120, 145));

        JLabel title = new JLabel("Your mind maps");
        title.setFont(new Font("Segoe UI", Font.BOLD, 22));
        title.setForeground(new Color(40, 40, 70));

        left.add(hi);
        left.add(Box.createVerticalStrut(4));
        left.add(title);

        topBar.add(left, BorderLayout.WEST);

        JPanel right = new JPanel(new FlowLayout(FlowLayout.RIGHT, 10, 0));
        right.setOpaque(false);

        JButton newMapBtn = new JButton("New map");
        JButton logoutBtn = new JButton("Logout");

        styleNewMapButton(newMapBtn);
        styleLogoutButton(logoutBtn);

        right.add(newMapBtn);
        right.add(logoutBtn);
        topBar.add(right, BorderLayout.EAST);

        // ===== Center: split =====
        JPanel leftCenter = new JPanel(new BorderLayout());
        leftCenter.setOpaque(false);
        leftCenter.setBorder(BorderFactory.createEmptyBorder(10, 16, 16, 8));
        leftCenter.add(searchSortPanel, BorderLayout.NORTH);
        leftCenter.add(mapsListPanel, BorderLayout.CENTER);

        JPanel rightCenter = new JPanel(new BorderLayout());
        rightCenter.setOpaque(false);
        rightCenter.setBorder(BorderFactory.createEmptyBorder(10, 8, 16, 16));
        rightCenter.add(propertiesPanel, BorderLayout.NORTH);
```

```
        JSplitPane split = new JSplitPane(JSplitPane.HORIZONTAL_SPLIT, leftCenter,
rightCenter);
        split.setDividerLocation(560);
        split.setBorder(null);
        mainPanel.add(split, BorderLayout.CENTER);

        // ===== Handlers =====
        newMapBtn.addActionListener(e -> onCreateMap());
        logoutBtn.addActionListener(e -> onLogout());

        // Double click open
        mapsListPanel.getList().addMouseListener(new java.awt.event.MouseAdapter() {
            @Override
            public void mouseClicked(java.awt.event.MouseEvent e) {
                if (e.getClickCount() == 2) {
                    MindMap selected = mapsListPanel.getSelected();
                    if (selected != null) openEditor(selected);
                }
            }
        });
    }

    private void onCreateMap() {
        String title = JOptionPane.showInputDialog(this, "Enter map title:");
        if (title == null) return;
        if (title.isBlank()) {
            JOptionPane.showMessageDialog(this, "Назва мапи обов'язкова");
            return;
        }

        mindMapService.createMap(title.trim(), user);
        mediator.refresh();
    }

    private void onLogout() {
        dispose();
        SwingUtilities.invokeLater(() ->
                new LoginForm(authService, mindMapService).setVisible(true)
        );
    }

    private void openEditor(MindMap map) {
        SwingUtilities.invokeLater(() ->
                new MindMapEditorForm(map, mindMapService).setVisible(true)
        );
    }

    private void styleNewMapButton(JButton btn) {
        btn.setBackground(new Color(76, 175, 80));
        btn.setForeground(Color.BLACK);
        btn.setFocusPainted(false);
        btn.setBorder(BorderFactory.createEmptyBorder(10, 26, 10, 26));
        btn.setCursor(Cursor.getPredefinedCursor(Cursor.HAND_CURSOR));
```

```
        btn.setFont(new Font("Segoe UI", Font.BOLD, 15));
    }

    private void styleLogoutButton(JButton btn) {
        btn.setBackground(new Color(244, 67, 54));
        btn.setForeground(Color.BLACK);
        btn.setFocusPainted(false);
        btn.setBorder(BorderFactory.createEmptyBorder(10, 20, 10, 20));
        btn.setCursor(Cursor.getPredefinedCursor(Cursor.HAND_CURSOR));
        btn.setFont(new Font("Segoe UI", Font.BOLD, 15));
    }
}
```

src/com/example/mindmap/ui/LoginForm.java:

```
package com.example.mindmap.ui;

import com.example.mindmap.entities.User;
import com.example.mindmap.services.AuthService;
import com.example.mindmap.services.MindMapService;

import javax.swing.*;
import java.awt.*;
import java.util.Optional;

public class LoginForm extends JFrame {

    private final AuthService authService;
    private final MindMapService mindMapService;

    private JTextField usernameField;
    private JPasswordField passwordField;

    public LoginForm(AuthService authService, MindMapService mindMapService) {
        super("Mind Map - Login");
        this.authService = authService;
        this.mindMapService = mindMapService;
        initComponents();
    }

    private void initComponents() {
        setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
        setSize(430, 270);
        setLocationRelativeTo(null);
        setResizable(false);

        JPanel background = new JPanel(new GridBagLayout());
        background.setBackground(new Color(245, 247, 252));
        add(background);

        JPanel card = new JPanel(new BorderLayout());
        card.setBackground(Color.WHITE);
        card.setBorder(BorderFactory.createCompoundBorder(
                BorderFactory.createLineBorder(new Color(230, 230, 230)),
```

```
                BorderFactory.createEmptyBorder(18, 22, 18, 22)
        ));

        background.add(card);

        JLabel titleLabel = new JLabel("Mind Map", SwingConstants.CENTER);
        titleLabel.setFont(new Font("Segoe UI", Font.BOLD, 22));
        titleLabel.setForeground(new Color(50, 50, 80));
        card.add(titleLabel, BorderLayout.NORTH);

        JPanel formPanel = new JPanel(new GridBagLayout());
        formPanel.setOpaque(false);

        GridBagConstraints fgbc = new GridBagConstraints();
        fgbc.insets = new Insets(6, 6, 6, 6);
        fgbc.anchor = GridBagConstraints.WEST;
        fgbc.fill = GridBagConstraints.HORIZONTAL;

        fgbc.gridx = 0;
        fgbc.gridy = 0;
        formPanel.add(new JLabel("Username:"), fgbc);

        fgbc.gridx = 1;
        usernameField = new JTextField(18);
        formPanel.add(usernameField, fgbc);

        fgbc.gridx = 0;
        fgbc.gridy = 1;
        formPanel.add(new JLabel("Password:"), fgbc);

        fgbc.gridx = 1;
        passwordField = new JPasswordField(18);
        formPanel.add(passwordField, fgbc);

        card.add(formPanel, BorderLayout.CENTER);

        JPanel buttonsPanel = new JPanel(new FlowLayout(FlowLayout.RIGHT, 10, 0));
        buttonsPanel.setOpaque(false);

        JButton registerBtn = new JButton("Sign up");
        JButton loginBtn = new JButton("Login");

        stylePrimaryButton(loginBtn);
        styleSecondaryButton(registerBtn);

        buttonsPanel.add(registerBtn);
        buttonsPanel.add(loginBtn);
        card.add(buttonsPanel, BorderLayout.SOUTH);

        loginBtn.addActionListener(e -> onLogin());
        registerBtn.addActionListener(e -> onRegister());
    }

    private void onLogin() {
```

```
        String username = usernameField.getText().trim();
        String password = new String(passwordField.getPassword());

        if (username.isBlank() || password.isBlank()) {
            JOptionPane.showMessageDialog(this, "Please enter username and password");
            return;
        }

        Optional<User> userOpt = authService.login(username, password);
        if (userOpt.isPresent()) {
            User user = userOpt.get();

            SwingUtilities.invokeLater(() ->
                    new DashboardForm(user, authService,
mindMapService).setVisible(true)
            );
            dispose();
        } else {
            JOptionPane.showMessageDialog(this, "Invalid username or password");
        }
    }

    private void onRegister() {
        String username = usernameField.getText().trim();
        String password = new String(passwordField.getPassword());

        if (username.isBlank() || password.isBlank()) {
            JOptionPane.showMessageDialog(this, "Username and password are required");
            return;
        }

        authService.register(username, password);

        JOptionPane.showMessageDialog(this,
                "Welcome, " + username + "!\nNow you may log in.");
    }

    private void stylePrimaryButton(JButton btn) {
        btn.setBackground(new Color(88, 101, 242));
        btn.setForeground(Color.BLACK);
        btn.setFocusPainted(false);
        btn.setBorder(BorderFactory.createEmptyBorder(10, 24, 10, 24));
        btn.setCursor(Cursor.getPredefinedCursor(Cursor.HAND_CURSOR));
        btn.setFont(new Font("Segoe UI", Font.BOLD, 14));
    }

    private void styleSecondaryButton(JButton btn) {
        btn.setBackground(new Color(235, 238, 246));
        btn.setForeground(Color.BLACK);
        btn.setFocusPainted(false);
        btn.setBorder(BorderFactory.createEmptyBorder(10, 20, 10, 20));
        btn.setCursor(Cursor.getPredefinedCursor(Cursor.HAND_CURSOR));
        btn.setFont(new Font("Segoe UI", Font.PLAIN, 14));
    }
```

```
}
```

src/com/example/mindmap/ui/MindMapEditorForm.java:

```
package com.example.mindmap.ui;

import com.example.mindmap.entities.ImageNode;
import com.example.mindmap.entities.MindMap;
import com.example.mindmap.entities.MapElement;
import com.example.mindmap.entities.TextNode;
import com.example.mindmap.export.ExportService;
import com.example.mindmap.export.JpgExportStrategy;
import com.example.mindmap.export.PdfExportStrategy;
import com.example.mindmap.export.PngExportStrategy;
import com.example.mindmap.services.MindMapService;
import com.example.mindmap.services.commands.CommandManager;
import com.example.mindmap.services.commands.DeleteNodeCommand;
import com.example.mindmap.tools.DefaultToolFactory;
import com.example.mindmap.tools.EditorContext;
import com.example.mindmap.tools.SelectTool;
import com.example.mindmap.tools.Tool;
import com.example.mindmap.tools.ToolFactory;
import com.example.mindmap.tools.drawing.DrawingStroke;

import javax.imageio.ImageIO;
import javax.swing.*;
import java.awt.*;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.IOException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

public class MindMapEditorForm extends JFrame {

    private final MindMap mindMap;
    private final MindMapService mindMapService;
    private final CanvasPanel canvasPanel;
    private final ExportService exportService;
    private final CommandManager commandManager;

    public MindMapEditorForm(MindMap mindMap, MindMapService mindMapService) {
        super("Editing: " + mindMap.getTitle());
        this.mindMap = mindMap;
        this.mindMapService = mindMapService;
        this.exportService = new ExportService();
        this.commandManager = new CommandManager();

        setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);
        setSize(980, 680);
```

```
        setLocationRelativeTo(null);

        JPanel mainPanel = new JPanel(new BorderLayout());
        add(mainPanel);

        JPanel topBar = new JPanel(new BorderLayout());
        topBar.setBorder(BorderFactory.createEmptyBorder(8, 12, 8, 12));
        topBar.setBackground(Color.WHITE);

        JLabel titleLabel = new JLabel(mindMap.getTitle());
        titleLabel.setFont(new Font("Segoe UI", Font.BOLD, 18));
        topBar.add(titleLabel, BorderLayout.WEST);

        // ===== Right buttons =====
        JPanel rightButtons = new JPanel(new FlowLayout(FlowLayout.RIGHT, 8, 0));
        rightButtons.setOpaque(false);

        JButton undoBtn = new JButton("Undo");
        JButton redoBtn = new JButton("Redo");
        JButton exportBtn = new JButton("Export");

        styleButton(undoBtn, new Color(224, 224, 224));
        styleButton(redoBtn, new Color(224, 224, 224));
        styleButton(exportBtn, new Color(88, 101, 242));

        rightButtons.add(undoBtn);
        rightButtons.add(redoBtn);
        rightButtons.add(exportBtn);

        // ===== Tools =====
        JPanel toolsPanel = new JPanel(new FlowLayout(FlowLayout.CENTER, 8, 0));
        toolsPanel.setOpaque(false);

        JToggleButton selectToolBtn = new JToggleButton("Select");
        JToggleButton textToolBtn = new JToggleButton("Text");
        JToggleButton imageToolBtn = new JToggleButton("Image");
        JToggleButton penToolBtn = new JToggleButton("Pen");
        JToggleButton dashPenToolBtn = new JToggleButton("Dash pen");
        JToggleButton eraserToolBtn = new JToggleButton("Eraser");

        styleToggle(selectToolBtn);
        styleToggle(textToolBtn);
        styleToggle(imageToolBtn);
        styleToggle(penToolBtn);
        styleToggle(dashPenToolBtn);
        styleToggle(eraserToolBtn);

        ButtonGroup toolGroup = new ButtonGroup();
        toolGroup.add(selectToolBtn);
        toolGroup.add(textToolBtn);
        toolGroup.add(imageToolBtn);
        toolGroup.add(penToolBtn);
        toolGroup.add(dashPenToolBtn);
        toolGroup.add(eraserToolBtn);
```

```
        toolsPanel.add(selectToolBtn);
        toolsPanel.add(textToolBtn);
        toolsPanel.add(imageToolBtn);
        toolsPanel.add(penToolBtn);
        toolsPanel.add(dashPenToolBtn);
        toolsPanel.add(eraserToolBtn);

        // ===== Brush controls =====
        JPanel brushPanel = new JPanel(new FlowLayout(FlowLayout.LEFT, 8, 0));
        brushPanel.setOpaque(false);

        JButton colorBtn = new JButton("Color");
        styleButton(colorBtn, new Color(245, 245, 245));

        Integer[] sizes = {2, 3, 4, 6, 8, 10};
        JComboBox<Integer> penSizeBox = new JComboBox<>(sizes);
        penSizeBox.setSelectedItem(3);


        brushPanel.add(colorBtn);
        brushPanel.add(new JLabel("Pen:"));
        brushPanel.add(penSizeBox);

        JPanel center = new JPanel(new BorderLayout());
        center.setOpaque(false);
        center.add(toolsPanel, BorderLayout.CENTER);
        center.add(brushPanel, BorderLayout.WEST);

        topBar.add(center, BorderLayout.CENTER);
        topBar.add(rightButtons, BorderLayout.EAST);

        mainPanel.add(topBar, BorderLayout.NORTH);

        // ===== Canvas =====
        canvasPanel = new CanvasPanel(mindMap, mindMapService, commandManager);
        mainPanel.add(canvasPanel, BorderLayout.CENTER);

        // ===== Handlers =====
        selectToolBtn.addActionListener(e ->
canvasPanel.setTool(CanvasPanel.ToolType.SELECT));
        textToolBtn.addActionListener(e ->
canvasPanel.setTool(CanvasPanel.ToolType.ADD_TEXT));
        imageToolBtn.addActionListener(e ->
canvasPanel.setTool(CanvasPanel.ToolType.ADD_IMAGE));
        penToolBtn.addActionListener(e ->
canvasPanel.setTool(CanvasPanel.ToolType.PEN));
        dashPenToolBtn.addActionListener(e ->
canvasPanel.setTool(CanvasPanel.ToolType.DASH_PEN));
        eraserToolBtn.addActionListener(e ->
canvasPanel.setTool(CanvasPanel.ToolType.ERASER));

        selectToolBtn.setSelected(true);
        canvasPanel.setTool(CanvasPanel.ToolType.SELECT);
```

```
        undoBtn.addActionListener(e -> { commandManager.undo(); canvasPanel.repaint();
});
        redoBtn.addActionListener(e -> { commandManager.redo(); canvasPanel.repaint();
});
        exportBtn.addActionListener(e -> onExport());

        colorBtn.addActionListener(e -> {
            Color chosen = JColorChooser.showDialog(this, "Choose pen color",
canvasPanel.getBrushColor());
            if (chosen != null) canvasPanel.setBrushColor(chosen);
        });

        penSizeBox.addActionListener(e -> {
            Integer v = (Integer) penSizeBox.getSelectedItem();
            if (v != null) canvasPanel.setBrushWidth(v);
        });
    }

    private static void styleButton(AbstractButton b, Color bg) {
        b.setBackground(bg);
        b.setForeground(Color.BLACK);
        b.setFocusPainted(false);
        b.setBorder(BorderFactory.createEmptyBorder(6, 12, 6, 12));
        b.setCursor(Cursor.getPredefinedCursor(Cursor.HAND_CURSOR));
    }

    private static void styleToggle(AbstractButton b) {
        styleButton(b, new Color(245, 245, 245));
    }

    private void onExport() {
        String[] options = {"PNG", "JPG", "PDF", "Cancel"};
        int choice = JOptionPane.showOptionDialog(
                this, "Choose export format:", "Export map",
                JOptionPane.DEFAULT_OPTION, JOptionPane.QUESTION_MESSAGE,
                null, options, options[0]
        );
        if (choice == -1 || "Cancel".equals(options[choice])) return;

        String selectedFormat = options[choice];

        JFileChooser chooser = new JFileChooser();
        chooser.setDialogTitle("Save exported file");
        String extension = selectedFormat.toLowerCase();
        chooser.setSelectedFile(new File(mindMap.getTitle().replaceAll("\\s+", "_") +
"." + extension));

        int result = chooser.showSaveDialog(this);
        if (result != JFileChooser.APPROVE_OPTION) return;

        File targetFile = chooser.getSelectedFile();

        switch (selectedFormat) {
```

```
            case "PNG" -> exportService.setStrategy(new PngExportStrategy());
            case "JPG" -> exportService.setStrategy(new JpgExportStrategy());
            case "PDF" -> exportService.setStrategy(new PdfExportStrategy());
            default -> {
                JOptionPane.showMessageDialog(this, "Unsupported format");
                return;
            }
        }

        try {
            exportService.export(mindMap, canvasPanel, targetFile);
            JOptionPane.showMessageDialog(this, "Export successful:\n" +
targetFile.getAbsolutePath());
        } catch (IOException ex) {
            ex.printStackTrace();
            JOptionPane.showMessageDialog(this, "Export failed: " + ex.getMessage());
        } catch (Exception ex) {
            ex.printStackTrace();
            JOptionPane.showMessageDialog(this, "Unexpected error: " +
ex.getMessage());
        }
    }

    private static class CanvasPanel extends JPanel {

        enum ToolType { SELECT, ADD_TEXT, ADD_IMAGE, PEN, DASH_PEN, ERASER }

        private final MindMap mindMap;
        private final MindMapService mindMapService;
        private final CommandManager commandManager;

        private final List<MapElement> elements = new ArrayList<>();
        private MapElement selectedElement = null;

        private final ToolFactory toolFactory;
        private final EditorContext ctx;
        private Tool currentTool;

        private final Map<String, BufferedImage> imageCache = new HashMap<>();
        private static final int PADDING = 10;

        public CanvasPanel(MindMap mindMap, MindMapService mindMapService,
CommandManager commandManager) {
            this.mindMap = mindMap;
            this.mindMapService = mindMapService;
            this.commandManager = commandManager;

            setBackground(new Color(250, 250, 253));
            setFocusable(true);

            elements.addAll(mindMapService.getElementsForMap(mindMap));

            this.ctx = new EditorContext(this, mindMap, mindMapService, commandManager,
elements) {
```

```
                @Override public void setSelected(MapElement el) {
super.setSelected(el); selectedElement = el; }
                @Override public MapElement getSelected() { return selectedElement; }

                @Override
                public MapElement findElementAt(int x, int y) {
                    for (int i = elements.size() - 1; i >= 0; i--) {
                        MapElement el = elements.get(i);
                        Rectangle b = getElementBounds(el);
                        if (b.contains(x, y)) return el;
                    }
                    return null;
                }
            };

            //завантажуємо мазки з БД при відкритті мапи

ctx.getDrawingLayer().getStrokes().addAll(mindMapService.getStrokesForMap(mindMap));

            this.toolFactory = new DefaultToolFactory();
            this.currentTool = toolFactory.createSelectTool(ctx);

            MouseAdapter mouseHandler = new MouseAdapter() {
                @Override public void mousePressed(MouseEvent e) {
currentTool.onMousePressed(e); }
                @Override public void mouseDragged(MouseEvent e) {
currentTool.onMouseDragged(e); }
                @Override public void mouseReleased(MouseEvent e) {
currentTool.onMouseReleased(e); }

                @Override
                public void mouseClicked(MouseEvent e) {

                    if (currentTool instanceof SelectTool
                            && SwingUtilities.isLeftMouseButton(e)
                            && e.getClickCount() == 2) {

                        MapElement hit = ctx.findElementAt(e.getX(), e.getY());
                        if (hit instanceof TextNode textNode) {

                            String oldText = textNode.getTextForDisplay();
                            if (oldText == null) oldText = "";

                            String newText =
JOptionPane.showInputDialog(CanvasPanel.this, "Edit text:", oldText);
                            if (newText == null) return;

                            textNode.setTextForDisplay(newText.trim());
                            mindMapService.updateElement(textNode);
                            repaint();
                        }
                        return; // інакше tool теж може обробити double-click
                    }
```

```
                // звичайні кліки
                currentTool.onMouseClicked(e);
            }
        };

        addMouseListener(mouseHandler);
        addMouseMotionListener(mouseHandler);

        setupKeyBindings();
    }

    public Color getBrushColor() { return ctx.getBrushColor(); }
    public void setBrushColor(Color c) { ctx.setBrushColor(c); }
    public void setBrushWidth(int px) { ctx.setBrushWidthPx(px); }

    // залишили (може знадобитися пізніше), але UI для цього прибрали
    public void setEraserRadius(int px) { ctx.setEraserRadiusPx(px); }

    public void setTool(ToolType type) {
        switch (type) {
            case SELECT -> currentTool = toolFactory.createSelectTool(ctx);
            case ADD_TEXT -> currentTool = toolFactory.createAddTextNodeTool(ctx);
            case ADD_IMAGE -> currentTool =
toolFactory.createAddImageNodeTool(ctx);
            case PEN -> currentTool = toolFactory.createPenTool(ctx);
            case DASH_PEN -> currentTool = toolFactory.createDashedPenTool(ctx);
            case ERASER -> currentTool = toolFactory.createEraserTool(ctx);
        }
        requestFocusInWindow();
    }

    private void setupKeyBindings() {
        InputMap im = getInputMap(JComponent.WHEN_IN_FOCUSED_WINDOW);
        ActionMap am = getActionMap();

        im.put(KeyStroke.getKeyStroke("DELETE"), "delete-selected");
        im.put(KeyStroke.getKeyStroke("BACK_SPACE"), "delete-selected");

        am.put("delete-selected", new AbstractAction() {
            @Override
            public void actionPerformed(java.awt.event.ActionEvent e) {
                deleteSelectedElement();
            }
        });
    }

    private void deleteSelectedElement() {
        if (selectedElement == null) return;
        DeleteNodeCommand cmd = new DeleteNodeCommand(mindMap, mindMapService,
elements, selectedElement);
        commandManager.executeCommand(cmd);
        selectedElement = null;
        repaint();
    }
```

```
        private Rectangle getElementBounds(MapElement el) {
            return new Rectangle((int) el.getX(), (int) el.getY(), el.getWidthPx(),
el.getHeightPx());
        }

        private BufferedImage getCachedImage(String path) {
            if (path == null || path.isBlank()) return null;
            if (imageCache.containsKey(path)) return imageCache.get(path);
            try {
                BufferedImage img = ImageIO.read(new File(path));
                imageCache.put(path, img);
                return img;
            } catch (IOException ex) {
                imageCache.put(path, null);
                return null;
            }
        }

        private List<String> wrapText(String text, FontMetrics fm, int maxWidth) {
            List<String> lines = new ArrayList<>();
            if (text == null) return lines;

            String[] words = text.split("\\s+");
            StringBuilder line = new StringBuilder();

            for (String w : words) {
                if (line.length() == 0) { line.append(w); continue; }
                String test = line + " " + w;
                if (fm.stringWidth(test) <= maxWidth) line.append(" ").append(w);
                else { lines.add(line.toString()); line = new StringBuilder(w); }
            }
            if (line.length() > 0) lines.add(line.toString());
            if (lines.isEmpty() && !text.isEmpty()) lines.add(text);
            return lines;
        }

        private void paintAllStrokes(Graphics2D g2) {
            List<DrawingStroke> strokes = ctx.getDrawingLayer().getStrokes();
            if (strokes.isEmpty()) return;

            for (DrawingStroke s : strokes) {
                g2.setColor(s.getColor());
                g2.setStroke(s.buildAwtStroke());
                g2.draw(s.buildPath());
            }
        }

        @Override
        protected void paintComponent(Graphics g) {
            super.paintComponent(g);
            Graphics2D g2 = (Graphics2D) g.create();
            g2.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
RenderingHints.VALUE_ANTIALIAS_ON);
```

```
            for (MapElement el : elements) {
                Rectangle r = getElementBounds(el);

                g2.setColor(Color.WHITE);
                g2.fillRoundRect(r.x, r.y, r.width, r.height, 12, 12);

                if (el instanceof ImageNode imgNode) {
                    BufferedImage img = getCachedImage(imgNode.getImageUrl());
                    if (img != null) {
                        g2.drawImage(img, r.x, r.y, r.width, r.height, null);
                    } else {
                        g2.setColor(new Color(200, 60, 60));
                        g2.drawLine(r.x + 6, r.y + 6, r.x + r.width - 6, r.y + r.height
- 6);
                        g2.drawLine(r.x + r.width - 6, r.y + 6, r.x + 6, r.y + r.height
- 6);
                        g2.setColor(new Color(80, 80, 80));
                        g2.drawString("Image not found", r.x + 10, r.y + 20);
                    }
                } else if (el instanceof TextNode textNode) {
                    String text = textNode.getTextForDisplay();
                    if (text == null) text = "";

                    g2.setColor(new Color(40, 40, 60));
                    FontMetrics fm = g2.getFontMetrics();

                    int maxTextWidth = Math.max(20, r.width - PADDING * 2);
                    List<String> lines = wrapText(text, fm, maxTextWidth);

                    int neededH = Math.max(45, lines.size() * fm.getHeight() + PADDING
* 2);
                    if (textNode.getHeightPx() < neededH) textNode.setHeightPx(neededH);

                    int y = r.y + PADDING + fm.getAscent();
                    for (String lineStr : lines) {
                        g2.drawString(lineStr, r.x + PADDING, y);
                        y += fm.getHeight();
                        if (y > r.y + r.height - PADDING) break;
                    }
                }

                if (el == selectedElement) {
                    g2.setStroke(new BasicStroke(2f));
                    g2.setColor(new Color(255, 140, 0));
                } else {
                    g2.setStroke(new BasicStroke(1f));
                    g2.setColor(new Color(180, 180, 200));
                }
                g2.drawRoundRect(r.x, r.y, r.width, r.height, 12, 12);
            }

            // мазки по всій мапі зверху
            paintAllStrokes(g2);
```

```
            // прев'ю інструмента (поточний мазок)
            currentTool.paintOverlay(g2);

            g2.dispose();
        }
    }
}
```

src/com/example/mindmap/soa/client/ApiClient.java:

```
package com.example.mindmap.soa.client;

import com.example.mindmap.soa.common.CryptoUtils;
import com.example.mindmap.soa.common.FormCodec;
import com.example.mindmap.soa.common.SoaConfig;

import java.io.InputStream;
import java.io.OutputStream;
import java.net.HttpURLConnection;
import java.net.URL;
import java.nio.charset.StandardCharsets;
import java.util.LinkedHashMap;
import java.util.Map;

public class ApiClient {

    public Map<String,String> postEncrypted(String path, Map<String,String> formParams)
{
        try {
            String url = SoaConfig.BASE_URL + path;

            String plain = FormCodec.encode(formParams);
            String ivB64 = CryptoUtils.randomIvB64();
            String ctB64 = CryptoUtils.encryptToB64(ivB64, plain);

            HttpURLConnection conn = (HttpURLConnection) new URL(url).openConnection();
            conn.setRequestMethod("POST");
            conn.setDoOutput(true);
            conn.setRequestProperty("Content-Type", "text/plain; charset=utf-8");
            conn.setRequestProperty("X-IV", ivB64);

            try (OutputStream os = conn.getOutputStream()) {
                os.write(ctB64.getBytes(StandardCharsets.UTF_8));
            }

            int code = conn.getResponseCode();
            InputStream is = (code >= 200 && code < 300) ? conn.getInputStream() :
conn.getErrorStream();

            String respCtB64 = new String(is.readAllBytes(), StandardCharsets.UTF_8);
            String respIvB64 = conn.getHeaderField("X-IV");
```

```
            if (respIvB64 == null || respIvB64.isBlank()) {
                throw new RuntimeException("Missing X-IV in response");
            }

            String respPlain = CryptoUtils.decryptFromB64(respIvB64, respCtB64);
            return FormCodec.decode(respPlain);

        } catch (Exception e) {
            throw new RuntimeException("API call failed: " + e.getMessage(), e);
        }
    }

    public static Map<String,String> mapOf(String... kv) {
        Map<String,String> m = new LinkedHashMap<>();
        for (int i = 0; i + 1 < kv.length; i += 2) {
            m.put(kv[i], kv[i+1]);
        }
        return m;
    }
}
```

src/com/example/mindmap/soa/client/MindMapSoaClientDemo.java:

```
package com.example.mindmap.soa.client;

import com.example.mindmap.entities.MindMap;
import com.example.mindmap.entities.User;
import com.example.mindmap.soa.common.Codecs;

import java.nio.charset.StandardCharsets;
import java.util.*;
import java.util.Base64;

public class MindMapSoaClientDemo {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        ApiClient api = new ApiClient();

        System.out.print("Username: ");
        String username = sc.nextLine().trim();

        System.out.print("Password: ");
        String password = sc.nextLine();

        // LOGIN
        Map<String,String> loginResp = api.postEncrypted("/api/auth/login",
                ApiClient.mapOf("username", username, "password", password));

        if (!"1".equals(loginResp.get("ok"))) {
            System.out.println("Login failed: " + loginResp.getOrDefault("message",
"unknown"));
            return;
        }
```

```
        String token = loginResp.get("token");
        int userId = Integer.parseInt(loginResp.get("userId"));
        System.out.println("OK! token=" + token);

        User owner = new User();
        owner.setId(userId);
        owner.setUsername(username);

        // LIST MAPS
        Map<String,String> mapsResp = api.postEncrypted("/api/maps/list",
                ApiClient.mapOf("token", token));

        if (!"1".equals(mapsResp.get("ok"))) {
            System.out.println("Maps list failed: " + mapsResp.getOrDefault("message",
"unknown"));
            return;
        }

        String dataB64 = mapsResp.getOrDefault("data", "");
        String lines = new String(Base64.getDecoder().decode(dataB64),
StandardCharsets.UTF_8);

        List<MindMap> maps = new ArrayList<>();
        for (String line : lines.split("\n")) {
            line = line.trim();
            if (line.isBlank()) continue;
            maps.add(Codecs.decodeMap(line, owner));
        }

        System.out.println("Your maps (" + maps.size() + "):");
        for (MindMap m : maps) {
            System.out.println(" - [" + m.getId() + "] " + m.getTitle() +
(m.isFavorite() ? " ★" : ""));
        }

        // CREATE MAP
        System.out.print("\nCreate new map? (y/n): ");
        String ans = sc.nextLine().trim().toLowerCase(Locale.ROOT);
        if (ans.equals("y")) {
            System.out.print("New map title: ");
            String title = sc.nextLine().trim();

            Map<String,String> createResp = api.postEncrypted("/api/maps/create",
                    ApiClient.mapOf("token", token, "title", title));

            if (!"1".equals(createResp.get("ok"))) {
                System.out.println("Create failed: " +
createResp.getOrDefault("message", "unknown"));
                return;
            }

            String createdLine = new
String(Base64.getDecoder().decode(createResp.get("data")), StandardCharsets.UTF_8);
```

```
            MindMap created = Codecs.decodeMap(createdLine, owner);
            System.out.println("Created map: [" + created.getId() + "] " +
created.getTitle());
        }
    }
}
```

src/com/example/mindmap/soa/common/SoaConfig.java:

```
package com.example.mindmap.soa.common;

public final class SoaConfig {
    private SoaConfig() {}

    public static final int PORT = 8088;
    public static final String BASE_URL = "http://localhost:" + PORT;

    // ВАЖЛИВО: однаковий секрет має бути і на сервері, і на клієнті
    //для реального продукту — роблять інакше (key exchange / TLS).
    public static final String SHARED_SECRET = "mindmap-trpz-shared-secret";
}
```

src/com/example/mindmap/soa/common/CryptoUtils.java:

```
package com.example.mindmap.soa.common;

import javax.crypto.Cipher;
import javax.crypto.spec.GCMParameterSpec;
import javax.crypto.spec.SecretKeySpec;
import java.nio.charset.StandardCharsets;
import java.security.MessageDigest;
import java.security.SecureRandom;
import java.util.Base64;

public final class CryptoUtils {
    private CryptoUtils() {}

    private static final SecureRandom RNG = new SecureRandom();
    private static final int GCM_TAG_BITS = 128;
    private static final int IV_BYTES = 12;

    private static byte[] deriveKeyBytes(String sharedSecret) {
        try {
            MessageDigest sha = MessageDigest.getInstance("SHA-256");
            return sha.digest(sharedSecret.getBytes(StandardCharsets.UTF_8)); // 32
bytes => AES-256
        } catch (Exception e) {
            throw new RuntimeException("Cannot derive key", e);
        }
    }

    private static SecretKeySpec key() {
        return new SecretKeySpec(deriveKeyBytes(SoaConfig.SHARED_SECRET), "AES");
    }
```

```
    public static String randomIvB64() {
        byte[] iv = new byte[IV_BYTES];
        RNG.nextBytes(iv);
        return Base64.getEncoder().encodeToString(iv);
    }

    public static String encryptToB64(String ivB64, String plainText) {
        try {
            byte[] iv = Base64.getDecoder().decode(ivB64);
            Cipher cipher = Cipher.getInstance("AES/GCM/NoPadding");
            cipher.init(Cipher.ENCRYPT_MODE, key(), new GCMParameterSpec(GCM_TAG_BITS,
iv));
            byte[] ct = cipher.doFinal(plainText.getBytes(StandardCharsets.UTF_8));
            return Base64.getEncoder().encodeToString(ct);
        } catch (Exception e) {
            throw new RuntimeException("Encrypt failed", e);
        }
    }

    public static String decryptFromB64(String ivB64, String cipherTextB64) {
        try {
            byte[] iv = Base64.getDecoder().decode(ivB64);
            byte[] ct = Base64.getDecoder().decode(cipherTextB64);
            Cipher cipher = Cipher.getInstance("AES/GCM/NoPadding");
            cipher.init(Cipher.DECRYPT_MODE, key(), new GCMParameterSpec(GCM_TAG_BITS,
iv));
            byte[] pt = cipher.doFinal(ct);
            return new String(pt, StandardCharsets.UTF_8);
        } catch (Exception e) {
            throw new RuntimeException("Decrypt failed", e);
        }
    }
}
```

src/com/example/mindmap/soa/common/FormCodec.java:

```
package com.example.mindmap.soa.common;

import java.net.URLDecoder;
import java.net.URLEncoder;
import java.nio.charset.StandardCharsets;
import java.util.LinkedHashMap;
import java.util.Map;

public final class FormCodec {
    private FormCodec() {}

    public static String encode(Map<String, String> params) {
        StringBuilder sb = new StringBuilder();
        boolean first = true;
        for (Map.Entry<String, String> e : params.entrySet()) {
            if (!first) sb.append("&");
            first = false;
```

```
            sb.append(urlEnc(e.getKey())).append("=").append(urlEnc(e.getValue() ==
null ? "" : e.getValue()));
        }
        return sb.toString();
    }

    public static Map<String, String> decode(String form) {
        Map<String, String> out = new LinkedHashMap<>();
        if (form == null || form.isBlank()) return out;

        String[] pairs = form.split("&");
        for (String p : pairs) {
            if (p.isBlank()) continue;
            int idx = p.indexOf('=');
            if (idx < 0) {
                out.put(urlDec(p), "");
            } else {
                String k = urlDec(p.substring(0, idx));
                String v = urlDec(p.substring(idx + 1));
                out.put(k, v);
            }
        }
        return out;
    }

    private static String urlEnc(String s) {
        return URLEncoder.encode(s, StandardCharsets.UTF_8);
    }

    private static String urlDec(String s) {
        return URLDecoder.decode(s, StandardCharsets.UTF_8);
    }
}
```

src/com/example/mindmap/soa/common/Codecs.java:

```
package com.example.mindmap.soa.common;

import com.example.mindmap.entities.*;
import com.example.mindmap.tools.drawing.DrawingStroke;

import java.awt.Color;
import java.nio.charset.StandardCharsets;
import java.util.Base64;

public final class Codecs {
    private Codecs() {}

    private static String b64(String s) {
        if (s == null) s = "";
        return Base64.getEncoder().encodeToString(s.getBytes(StandardCharsets.UTF_8));
    }
```

```
    private static String unb64(String b64) {
        if (b64 == null || b64.isBlank()) return "";
        return new String(Base64.getDecoder().decode(b64), StandardCharsets.UTF_8);
    }

    // MAP|id|titleB64|descB64|favorite|catId|catTitleB64|catColor
    public static String encodeMap(MindMap m) {
        Category c = m.getCategory();
        int catId = (c == null) ? 0 : c.getId();
        String catTitle = (c == null) ? "" : c.getTitle();
        String catColor = (c == null) ? "" : c.getColor();

        return "MAP|" + m.getId()
                + "|" + b64(m.getTitle())
                + "|" + b64(m.getDescription())
                + "|" + (m.isFavorite() ? "1" : "0")
                + "|" + catId
                + "|" + b64(catTitle)
                + "|" + (catColor == null ? "" : catColor);
    }

    public static MindMap decodeMap(String line, User owner) {
        // tolerant parsing
        String[] p = line.split("\\|", -1);
        if (p.length < 5 || !"MAP".equals(p[0])) throw new IllegalArgumentException("Bad
map line: " + line);

        MindMap m = new MindMap();
        m.setId(Integer.parseInt(p[1]));
        m.setTitle(unb64(p[2]));
        m.setDescription(unb64(p[3]));
        m.setFavorite("1".equals(p[4]));
        m.setOwner(owner);

        if (p.length >= 8) {
            int catId = Integer.parseInt(p[5].isBlank() ? "0" : p[5]);
            if (catId != 0) {
                Category c = new Category();
                c.setId(catId);
                c.setTitle(unb64(p[6]));
                c.setColor(p[7]);
                c.setOwner(owner);
                m.setCategory(c);
            }
        }
        return m;
    }

    // CAT|id|titleB64|color
    public static String encodeCategory(Category c) {
        return "CAT|" + c.getId() + "|" + b64(c.getTitle()) + "|" + (c.getColor() ==
null ? "" : c.getColor());
    }
```

```
   public static Category decodeCategory(String line, User owner) {
       String[] p = line.split("\\|", -1);
       if (p.length < 4 || !"CAT".equals(p[0])) throw new IllegalArgumentException("Bad
category line: " + line);

       Category c = new Category();
       c.setId(Integer.parseInt(p[1]));
       c.setTitle(unb64(p[2]));
       c.setColor(p[3]);
       c.setOwner(owner);
       return c;
   }

   // TEXT|id|x|y|w|h|font|shapeB64|textB64
   // IMAGE|id|x|y|w|h|pathB64
   public static String encodeElement(MapElement el) {
       if (el instanceof TextNode t) {
           return "TEXT|" + t.getId()
                   + "|" + t.getX()
                   + "|" + t.getY()
                   + "|" + t.getWidthPx()
                   + "|" + t.getHeightPx()
                   + "|" + t.getFontSize()
                   + "|" + b64(t.getShapeType())
                   + "|" + b64(t.getTextForDisplay());
       }
       if (el instanceof ImageNode im) {
           return "IMAGE|" + im.getId()
                   + "|" + im.getX()
                   + "|" + im.getY()
                   + "|" + im.getWidthPx()
                   + "|" + im.getHeightPx()
                   + "|" + b64(im.getImageUrl());
       }
       throw new IllegalArgumentException("Unknown element type: " + el);
   }

   public static MapElement decodeElement(String line) {
       String[] p = line.split("\\|", -1);
       if (p.length < 1) throw new IllegalArgumentException("Bad element line: " +
line);

       switch (p[0]) {
           case "TEXT" -> {
               int id = Integer.parseInt(p[1]);
               float x = Float.parseFloat(p[2]);
               float y = Float.parseFloat(p[3]);
               int w = Integer.parseInt(p[4]);
               int h = Integer.parseInt(p[5]);
               int font = Integer.parseInt(p[6]);
               String shape = unb64(p[7]);
               String text = unb64(p[8]);
```

```
                // конструктор, який вже є в твоєму коді (використовується і в JDBC
репозиторії)
                return new TextNode(id, x, y, null, text, font, shape, w, h);
            }
            case "IMAGE" -> {
                int id = Integer.parseInt(p[1]);
                float x = Float.parseFloat(p[2]);
                float y = Float.parseFloat(p[3]);
                int w = Integer.parseInt(p[4]);
                int h = Integer.parseInt(p[5]);
                String path = unb64(p[6]);

                return new ImageNode(id, x, y, null, path, w, h);
            }
            default -> throw new IllegalArgumentException("Bad element type: " + p[0]);
        }
    }

    // STROKE|idB64|argb|width|dashed|pointsB64
    public static String encodeStroke(DrawingStroke s) {
        String id = s.getId() == null ? "" : s.getId();
        String points = s.serializePoints();
        return "STROKE|" + b64(id)
                + "|" + s.getColor().getRGB()
                + "|" + s.getWidthPx()
                + "|" + (s.isDashed() ? "1" : "0")
                + "|" + b64(points);
    }

    public static DrawingStroke decodeStroke(String line) {
        String[] p = line.split("\\|", -1);
        if (p.length < 6 || !"STROKE".equals(p[0])) throw new
IllegalArgumentException("Bad stroke line: " + line);

        String id = unb64(p[1]);
        int argb = Integer.parseInt(p[2]);
        int width = Integer.parseInt(p[3]);
        boolean dashed = "1".equals(p[4]);
        String points = unb64(p[5]);

        DrawingStroke s = new DrawingStroke(id, new Color(argb, true), width, dashed);
        DrawingStroke.deserializePointsInto(s, points);
        return s;
    }
}
```

src/com/example/mindmap/soa/server/MindMapSoaServerMain.java:

```
package com.example.mindmap.soa.server;

import com.example.mindmap.db.JdbcConnectionProvider;
import com.example.mindmap.entities.Category;
import com.example.mindmap.entities.MapElement;
import com.example.mindmap.entities.MindMap;
```

```
import com.example.mindmap.entities.User;
import com.example.mindmap.repositories.JdbcCategoryRepository;
import com.example.mindmap.repositories.JdbcMapElementRepository;
import com.example.mindmap.repositories.JdbcMapStrokeRepository;
import com.example.mindmap.repositories.JdbcMindMapRepository;
import com.example.mindmap.repositories.JdbcUserRepository;
import com.example.mindmap.services.AuthService;
import com.example.mindmap.services.MindMapService;
import com.example.mindmap.soa.common.Codecs;
import com.example.mindmap.soa.common.CryptoUtils;
import com.example.mindmap.soa.common.FormCodec;
import com.example.mindmap.soa.common.SoaConfig;
import com.example.mindmap.tools.drawing.DrawingStroke;
import com.sun.net.httpserver.HttpExchange;
import com.sun.net.httpserver.HttpServer;

import java.io.IOException;
import java.io.OutputStream;
import java.net.InetSocketAddress;
import java.nio.charset.StandardCharsets;
import java.util.*;
import java.util.Base64;

public class MindMapSoaServerMain {

   private static final class Req {
       final Map<String, String> form;
       final Integer userId;
       Req(Map<String, String> form, Integer userId) {
           this.form = form;
           this.userId = userId;
       }
   }

   public static void main(String[] args) throws Exception {

       // --- wiring: JDBC репозиторії + сервіси ---
       JdbcConnectionProvider provider = new JdbcConnectionProvider();

       JdbcUserRepository userRepo = new JdbcUserRepository(provider);
       JdbcMindMapRepository mapRepo = new JdbcMindMapRepository(provider);
       JdbcMapElementRepository elementRepo = new JdbcMapElementRepository(provider);
       JdbcMapStrokeRepository strokeRepo = new JdbcMapStrokeRepository(provider);
       JdbcCategoryRepository categoryRepo = new JdbcCategoryRepository(provider);

       AuthService authService = new AuthService(userRepo);

       MindMapService mindMapService = new MindMapService(mapRepo, elementRepo,
strokeRepo, categoryRepo);

       TokenService tokens = new TokenService(60 * 60); // 1 година

       // --- HTTP server ---
```

```
        HttpServer server = HttpServer.create(new InetSocketAddress(SoaConfig.PORT),
0);

        // health-check (не шифрований)
        server.createContext("/", ex -> {
            if (!"GET".equalsIgnoreCase(ex.getRequestMethod())) {
                writePlain(ex, 405, "Method Not Allowed");
                return;
            }
            writePlain(ex, 200, "MindMap SOA server is running on port " +
SoaConfig.PORT);
        });

        // -------- AUTH --------

        server.createContext("/api/auth/login", ex -> {
            if (!"POST".equalsIgnoreCase(ex.getRequestMethod())) { writePlain(ex, 405,
"Method Not Allowed"); return; }

            Req r = readReq(ex, tokens, false);
            if (r == null) return;

            String username = r.form.getOrDefault("username", "").trim();
            String password = r.form.getOrDefault("password", "");

            Optional<User> uOpt = authService.login(username, password);
            if (uOpt.isEmpty()) { writeEncryptedError(ex, 401, "Invalid credentials");
return; }

            User u = uOpt.get();
            String token = tokens.issueToken(u.getId());

            Map<String,String> resp = new LinkedHashMap<>();
            resp.put("ok", "1");
            resp.put("userId", String.valueOf(u.getId()));
            resp.put("username", u.getUsername());
            resp.put("token", token);
            writeEncryptedForm(ex, 200, resp);
        });

        server.createContext("/api/auth/register", ex -> {
            if (!"POST".equalsIgnoreCase(ex.getRequestMethod())) { writePlain(ex, 405,
"Method Not Allowed"); return; }

            Req r = readReq(ex, tokens, false);
            if (r == null) return;

            String username = r.form.getOrDefault("username", "").trim();
            String password = r.form.getOrDefault("password", "");

            try {
                User created = authService.register(username, password);
                String token = tokens.issueToken(created.getId());
```

```
            Map<String,String> resp = new LinkedHashMap<>();
            resp.put("ok", "1");
            resp.put("userId", String.valueOf(created.getId()));
            resp.put("username", created.getUsername());
            resp.put("token", token);
            writeEncryptedForm(ex, 200, resp);
        } catch (Exception e) {
            writeEncryptedError(ex, 400, "Register failed: " + e.getMessage());
        }
    });

    // -------- MAPS --------

    server.createContext("/api/maps/list", ex -> {
        if (!"POST".equalsIgnoreCase(ex.getRequestMethod())) { writePlain(ex, 405,
"Method Not Allowed"); return; }

        Req r = readReq(ex, tokens, true);
        if (r == null) return;

        User owner = new User();
        owner.setId(r.userId);

        List<MindMap> maps = mindMapService.getMapsByUser(owner);

        StringBuilder lines = new StringBuilder();
        for (MindMap m : maps) lines.append(Codecs.encodeMap(m)).append("\n");

        Map<String,String> resp = new LinkedHashMap<>();
        resp.put("ok", "1");
        resp.put("data",
Base64.getEncoder().encodeToString(lines.toString().getBytes(StandardCharsets.UTF_8)))
;
        writeEncryptedForm(ex, 200, resp);
    });

    server.createContext("/api/maps/create", ex -> {
        if (!"POST".equalsIgnoreCase(ex.getRequestMethod())) { writePlain(ex, 405,
"Method Not Allowed"); return; }

        Req r = readReq(ex, tokens, true);
        if (r == null) return;

        String title = r.form.getOrDefault("title", "").trim();
        if (title.isBlank()) { writeEncryptedError(ex, 400, "Title required");
return; }

        User owner = new User();
        owner.setId(r.userId);

        MindMap created = mindMapService.createMap(title, owner);

        Map<String,String> resp = new LinkedHashMap<>();
        resp.put("ok", "1");
```

```
            resp.put("data",
Base64.getEncoder().encodeToString(Codecs.encodeMap(created).getBytes(StandardCharsets
.UTF_8)));
            writeEncryptedForm(ex, 200, resp);
        });

        // -------- CATEGORIES --------

        server.createContext("/api/categories/list", ex -> {
            if (!"POST".equalsIgnoreCase(ex.getRequestMethod())) { writePlain(ex, 405,
"Method Not Allowed"); return; }

            Req r = readReq(ex, tokens, true);
            if (r == null) return;

            User owner = new User();
            owner.setId(r.userId);

            List<Category> categories = mindMapService.getCategoriesByUser(owner);

            StringBuilder lines = new StringBuilder();
            for (Category c : categories)
lines.append(Codecs.encodeCategory(c)).append("\n");

            Map<String,String> resp = new LinkedHashMap<>();
            resp.put("ok", "1");
            resp.put("data",
Base64.getEncoder().encodeToString(lines.toString().getBytes(StandardCharsets.UTF_8)))
;
            writeEncryptedForm(ex, 200, resp);
        });

        server.createContext("/api/categories/create", ex -> {
            if (!"POST".equalsIgnoreCase(ex.getRequestMethod())) { writePlain(ex, 405,
"Method Not Allowed"); return; }

            Req r = readReq(ex, tokens, true);
            if (r == null) return;

            User owner = new User();
            owner.setId(r.userId);

            String title = r.form.getOrDefault("title", "").trim();
            String color = r.form.getOrDefault("color", "").trim();
            if (title.isBlank()) { writeEncryptedError(ex, 400, "title required");
return; }

            Category created = mindMapService.createCategory(owner, title, color);

            Map<String,String> resp = new LinkedHashMap<>();
            resp.put("ok", "1");
            resp.put("data",
Base64.getEncoder().encodeToString(Codecs.encodeCategory(created).getBytes(StandardCha
rsets.UTF_8)));
```

```
            writeEncryptedForm(ex, 200, resp);
        });

        // -------- ELEMENTS --------

        server.createContext("/api/elements/list", ex -> {
            if (!"POST".equalsIgnoreCase(ex.getRequestMethod())) { writePlain(ex, 405,
"Method Not Allowed"); return; }

            Req r = readReq(ex, tokens, true);
            if (r == null) return;

            int mapId = Integer.parseInt(r.form.getOrDefault("mapId", "0"));

            MindMap map = new MindMap();
            map.setId(mapId);

            List<MapElement> els = mindMapService.getElementsForMap(map);

            StringBuilder lines = new StringBuilder();
            for (MapElement el : els)
lines.append(Codecs.encodeElement(el)).append("\n");

            Map<String,String> resp = new LinkedHashMap<>();
            resp.put("ok", "1");
            resp.put("data",
Base64.getEncoder().encodeToString(lines.toString().getBytes(StandardCharsets.UTF_8)))
;
            writeEncryptedForm(ex, 200, resp);
        });

        server.createContext("/api/elements/update", ex -> {
            if (!"POST".equalsIgnoreCase(ex.getRequestMethod())) { writePlain(ex, 405,
"Method Not Allowed"); return; }

            Req r = readReq(ex, tokens, true);
            if (r == null) return;

            String elB64 = r.form.get("el");
            if (elB64 == null || elB64.isBlank()) { writeEncryptedError(ex, 400, "el
required"); return; }

            String line = new String(Base64.getDecoder().decode(elB64),
StandardCharsets.UTF_8);
            MapElement el = Codecs.decodeElement(line);
            mindMapService.updateElement(el);

            Map<String,String> resp = new LinkedHashMap<>();
            resp.put("ok", "1");
            writeEncryptedForm(ex, 200, resp);
        });

        // -------- STROKES --------
```

```
        server.createContext("/api/strokes/list", ex -> {
            if (!"POST".equalsIgnoreCase(ex.getRequestMethod())) { writePlain(ex, 405,
"Method Not Allowed"); return; }

            Req r = readReq(ex, tokens, true);
            if (r == null) return;

            int mapId = Integer.parseInt(r.form.getOrDefault("mapId", "0"));

            MindMap map = new MindMap();
            map.setId(mapId);

            List<DrawingStroke> strokes = mindMapService.getStrokesForMap(map);

            StringBuilder lines = new StringBuilder();
            for (DrawingStroke s : strokes)
lines.append(Codecs.encodeStroke(s)).append("\n");

            Map<String,String> resp = new LinkedHashMap<>();
            resp.put("ok", "1");
            resp.put("data",
Base64.getEncoder().encodeToString(lines.toString().getBytes(StandardCharsets.UTF_8)))
;
            writeEncryptedForm(ex, 200, resp);
        });

        server.start();
        System.out.println("MindMap SOA server started on " + SoaConfig.BASE_URL);
    }

    // ====== Request reading (ONE TIME) ======

    private static Req readReq(HttpExchange ex, TokenService tokens, boolean
requireAuth) throws IOException {
        Map<String,String> form;
        try {
            form = readEncryptedForm(ex);
        } catch (Exception e) {
            writeEncryptedError(ex, 400, "Bad encrypted request: " + e.getMessage());
            return null;
        }

        Integer userId = null;
        if (requireAuth) {
            String token = form.get("token");
            userId = tokens.validate(token);
            if (userId == null) {
                writeEncryptedError(ex, 401, "Invalid/expired token");
                return null;
            }
        }

        return new Req(form, userId);
    }
```

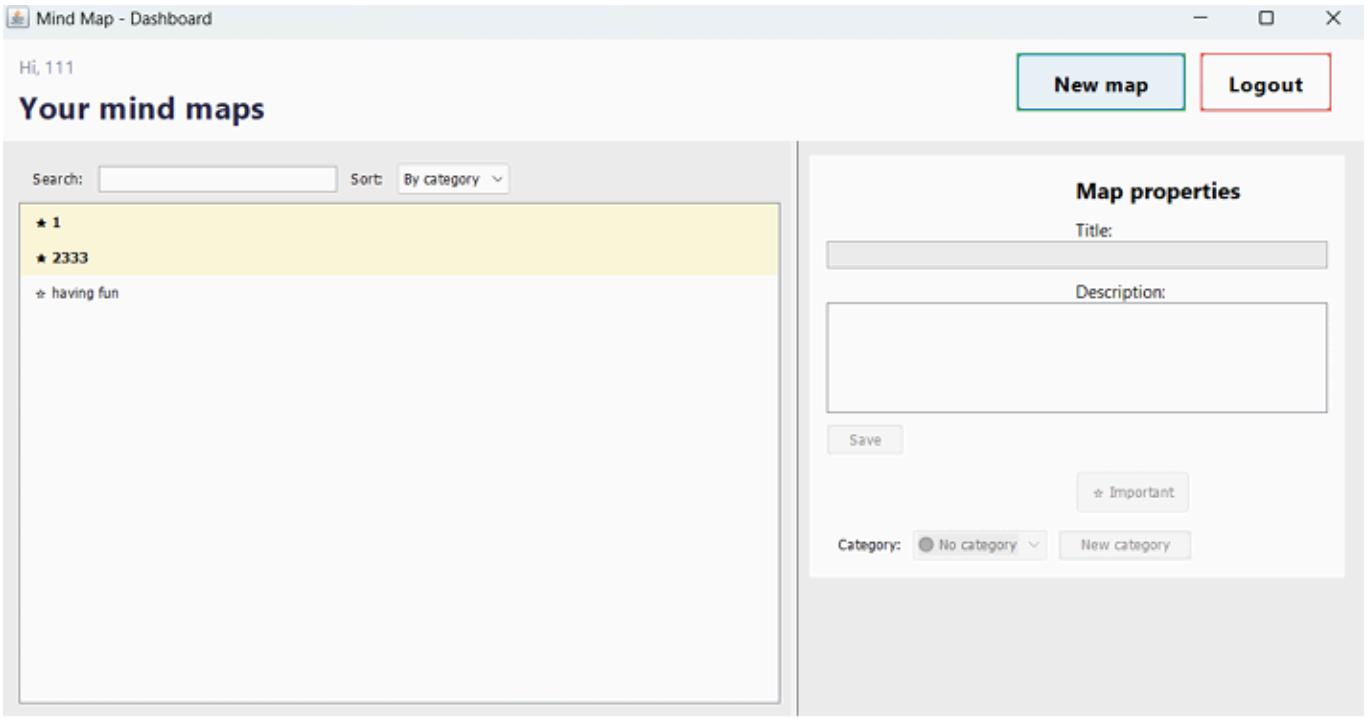
```
   private static Map<String,String> readEncryptedForm(HttpExchange ex) throws
IOException {
       String ivB64 = ex.getRequestHeaders().getFirst("X-IV");
       if (ivB64 == null || ivB64.isBlank()) throw new RuntimeException("Missing X-IV
header");

       String bodyB64 = new String(ex.getRequestBody().readAllBytes(),
StandardCharsets.UTF_8);
       String plain = CryptoUtils.decryptFromB64(ivB64, bodyB64);
       return FormCodec.decode(plain);
   }

   private static void writeEncryptedForm(HttpExchange ex, int status,
Map<String,String> params) throws IOException {
       String plain = FormCodec.encode(params);
       String ivB64 = CryptoUtils.randomIvB64();
       String ctB64 = CryptoUtils.encryptToB64(ivB64, plain);

       ex.getResponseHeaders().set("Content-Type", "text/plain; charset=utf-8");
       ex.getResponseHeaders().set("X-IV", ivB64);

       byte[] bytes = ctB64.getBytes(StandardCharsets.UTF_8);
       ex.sendResponseHeaders(status, bytes.length);
       try (OutputStream os = ex.getResponseBody()) { os.write(bytes); }
   }

   private static void writeEncryptedError(HttpExchange ex, int status, String msg)
throws IOException {
       Map<String,String> resp = new LinkedHashMap<>();
       resp.put("ok", "0");
       resp.put("message", msg);
       writeEncryptedForm(ex, status, resp);
   }

   private static void writePlain(HttpExchange ex, int status, String text) throws
IOException {
       byte[] bytes = text.getBytes(StandardCharsets.UTF_8);
       ex.getResponseHeaders().set("Content-Type", "text/plain; charset=utf-8");
       ex.sendResponseHeaders(status, bytes.length);
       try (OutputStream os = ex.getResponseBody()) { os.write(bytes); }
   }
}
```

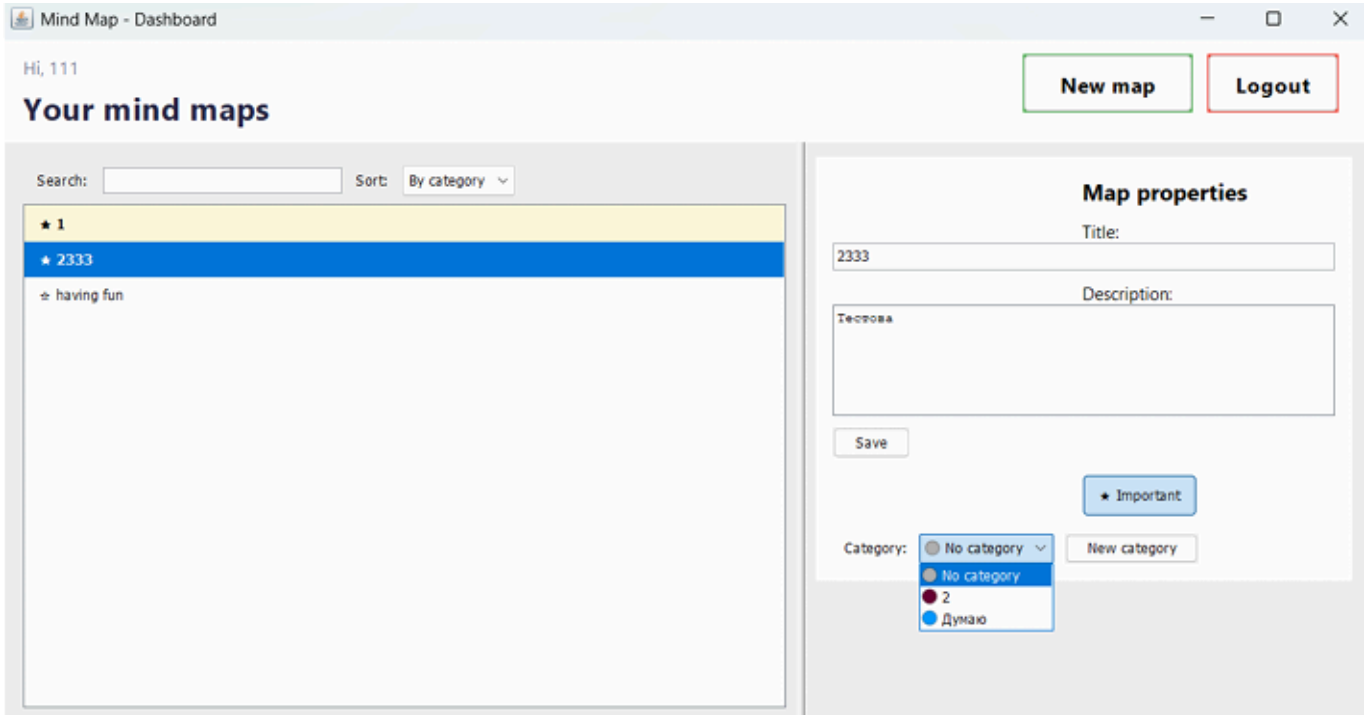
src/com/example/mindmap/soa/server/TokenService.java:

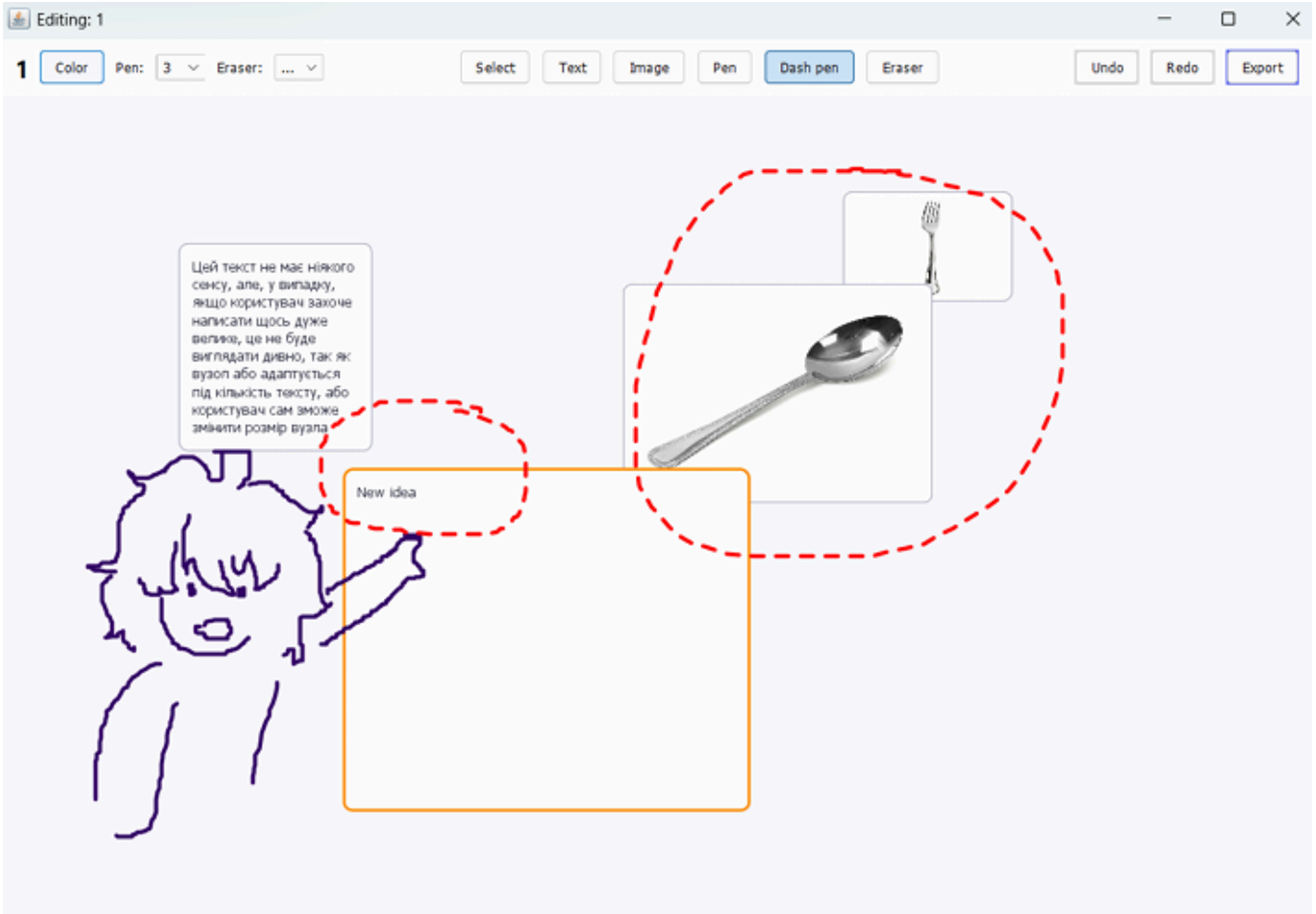
```
package com.example.mindmap.soa.server;

import java.security.SecureRandom;
import java.time.Instant;
import java.util.Base64;
import java.util.Map;
import java.util.concurrent.ConcurrentHashMap;
```

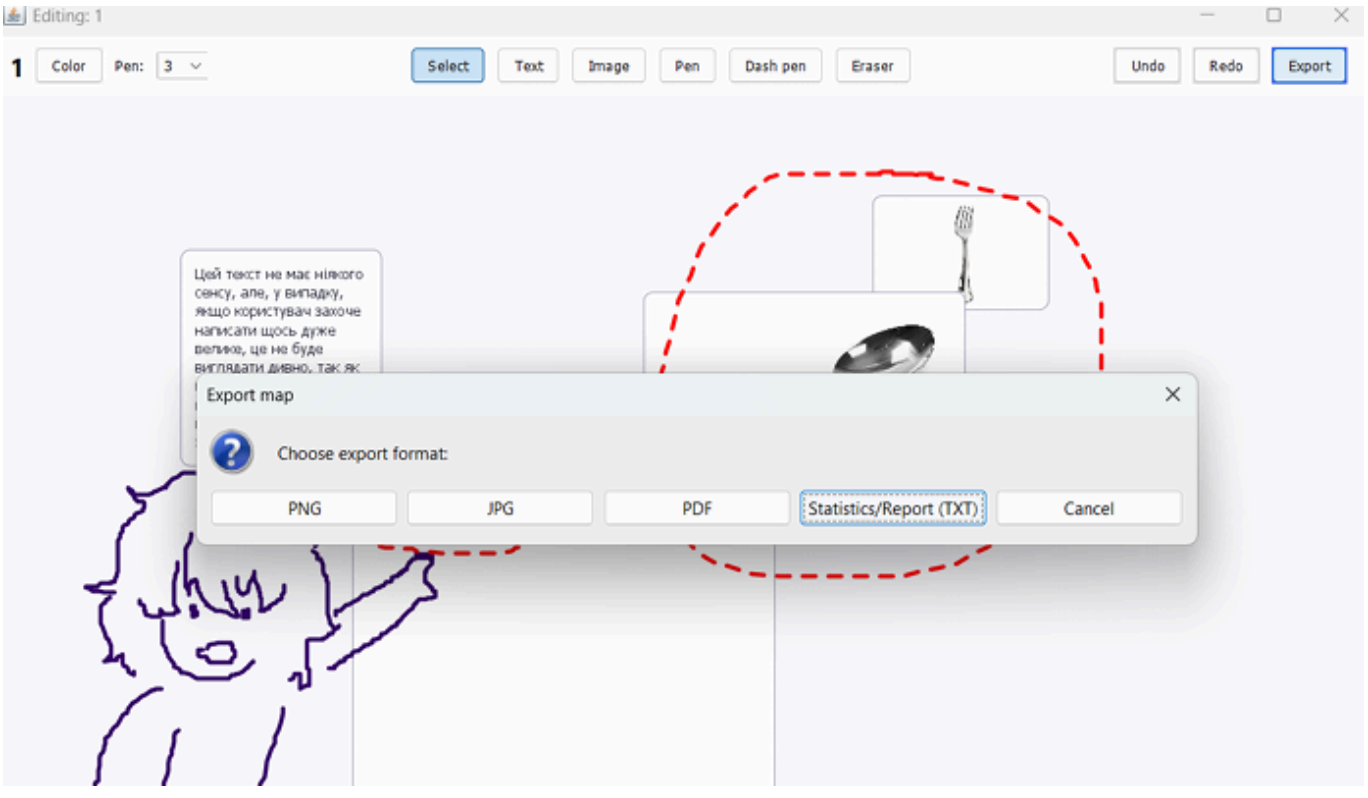
```
public final class TokenService {

    private static final class Session {
        final int userId;
        final long expiresAtEpochSec;
        Session(int userId, long expiresAtEpochSec) {
            this.userId = userId;
            this.expiresAtEpochSec = expiresAtEpochSec;
        }
    }

    private final SecureRandom rng = new SecureRandom();
    private final Map<String, Session> sessions = new ConcurrentHashMap<>();
    private final long ttlSeconds;

    public TokenService(long ttlSeconds) {
        this.ttlSeconds = ttlSeconds;
    }

    public String issueToken(int userId) {
        byte[] buf = new byte[32];
        rng.nextBytes(buf);
        String token = Base64.getUrlEncoder().withoutPadding().encodeToString(buf);
        long exp = Instant.now().getEpochSecond() + ttlSeconds;
        sessions.put(token, new Session(userId, exp));
        return token;
    }

    public Integer validate(String token) {
        if (token == null || token.isBlank()) return null;
        Session s = sessions.get(token);
        if (s == null) return null;
        if (Instant.now().getEpochSecond() > s.expiresAtEpochSec) {
            sessions.remove(token);
            return null;
        }
        return s.userId;
    }
}
```

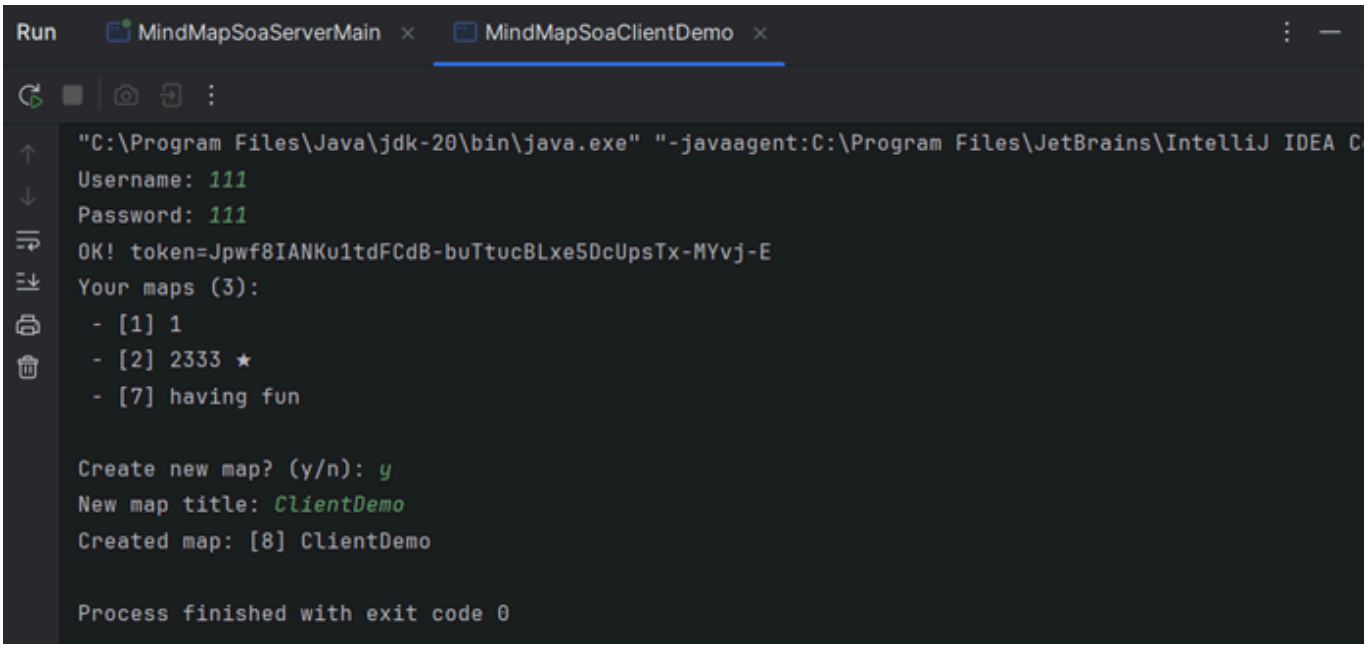
Скріншоти системи:

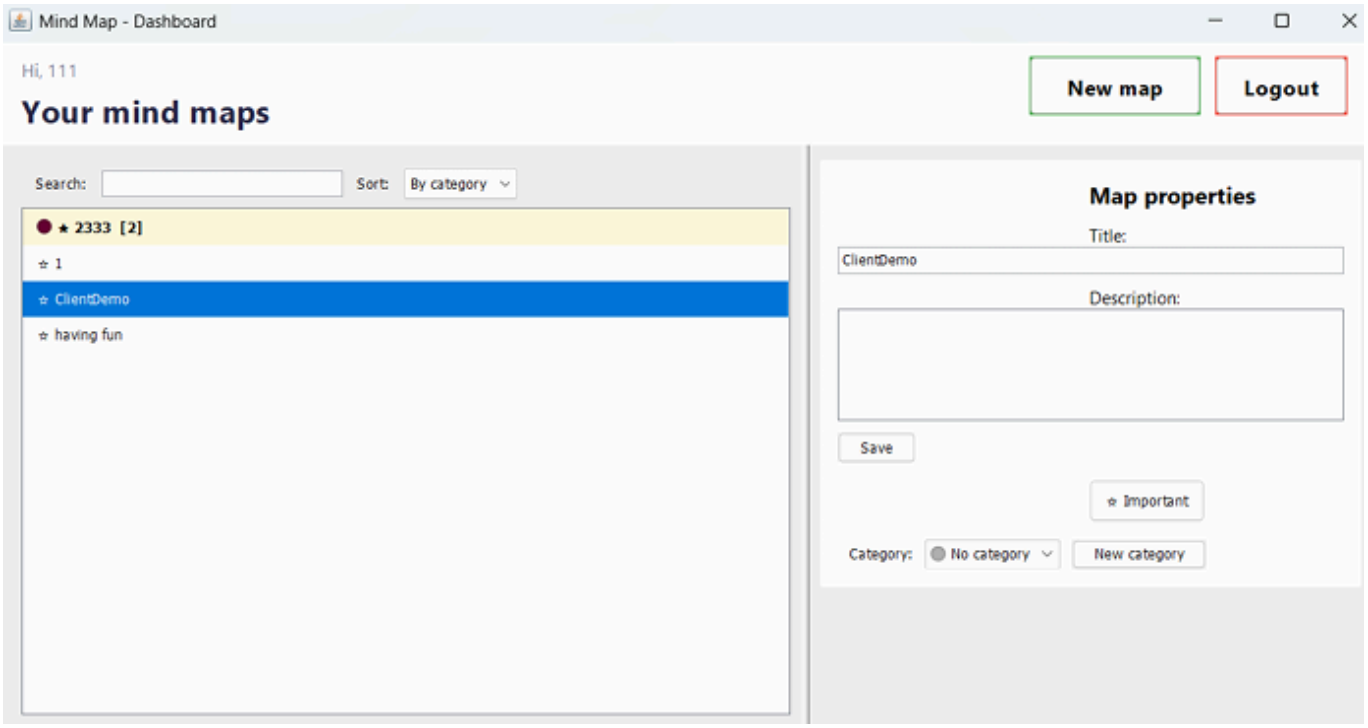

Mind Map - Dashboard
Hi, 111
Your mind maps
New map
Logout
Search:
Sort:
By category
★ 1
★ 2333
☆ having fun
Map properties
Title:
Description:
Save
☆ Important
Category:
No category
New category


Mind Map - Dashboard
Hi, 111
Your mind maps
New map
Logout
Search:
Sort:
By category
★ 1
★ 2333
☆ having fun
Map properties
Title:
2333
Description:
Тестова
Save
★ Important
Category:
No category
New category
No category
2
Думаю


Editing: 1
1
Color
Pen:
3
Eraser:
...
Select
Text
Image
Pen
Dash pen
Eraser
Undo
Redo
Export
Цей текст не має ніякого сенсу, але, у випадку, якщо користувач захоче написати щось дуже велике, це не буде виглядати дивно, так як вузол або адаптується під кількість тексту, або користувач сам зможе змінити розмір вузла
New idea


Editing: 1
1
Color
Pen:
3
Select
Text
Image
Pen
Dash pen
Eraser
Undo
Redo
Export
Цей текст не має ніякого сенсу, але, у випадку, якщо користувач захоче написати щось дуже велике, це не буде виглядати дивно, так як
Export map
Choose export format:
PNG
JPG
PDF
Statistics/Report (TXT)
Cancel


Run
MindMapSoaServerMain
MindMapSoaClientDemo
"C:\Program Files\Java\jdk-20\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA C
Username: 111
Password: 111
OK! token=Jpwf8IANKu1tdFCdB-buTtucBLxe5DcUpsTx-MYvj-E
Your maps (3):
- [1] 1
- [2] 2333 ★
- [7] having fun
Create new map? (y/n): y
New map title: ClientDemo
Created map: [8] ClientDemo
Process finished with exit code 0


Mind Map - Dashboard
Hi, 111
Your mind maps
New map
Logout
Search:
Sort:
By category
★ 2333 [2]
☆ 1
☆ ClientDemo
☆ having fun
Map properties
Title:
ClientDemo
Description:
Save
☆ Important
Category:
No category
New category