МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХЕРСОНСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
Факультет комп'ютерних наук, фізики та математики
Кафедра комп'ютерних наук та програмної інженерії

Розробка програмного модуля "Редактор геометричних задач" системи навчального призначення Геометрія 7-9

Кваліфікаційна робота (проєкт)
на здобуття ступеня вищої освіти "магістр"

Виконав: здобувач 2-го курсу 12-241М групи
Спеціальності 121 Інженерія програмного забезпечення

Освітньо-професійної програми
Інженерія програмного забезпечення

Струкало Валентин Володимирович

Керівник Львов М.С., доктор фізико математичних наук, професор

Рецензент Огнєва О.Є. кандидатка технічних наук, доцентка, завідувачка кафедри програмних засобів і технологій Херсонський національно технічний університет

Івано-Франківськ – 2025

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

- **S1—S8** — позначення сценаріїв роботи API та тестових кейсів (наприклад, S1: 201 Created, S2: OK, S8: Forbidden тощо)
- **taskId, solutionId, studentId, attemptHash** — поля та ідентифікатори у базі даних для задач, рішень та учнівської спроби
- **figureState, checkConfig, constraintstype, angle, atB, value, tolerance** — ключі конфігурації задачі у структурі JSON для перевірки коректності розв’язку
- **initialFigure, points, segments, circles** — змінні для опису геометричної фігури у задачі
- **API, JSON, MongoDB, Next.js, Mongoose, Canvas, Konva.js, Zustand, Tailwind CSS, shadcnui, React** — технології та компоненти, які умовно позначаються для частин програмної архітектури
- **User, Task, Solution** — сутності користувача, задачі та рішення у системі

ВСТУП

Сучасний освітній процес тісно пов’язаний із розвитком інформаційних технологій, які значно розширюють можливості викладання та навчання. Цифровізація освіти сприяє створенню інтерактивних середовищ, що дозволяють підвищити ефективність засвоєння матеріалу та зробити навчання більш гнучким, персоналізованим і цікавим для учнів.

Одним із напрямів розвитку сучасної освіти є впровадження **навчальних інформаційних систем**, орієнтованих на інтерактивну взаємодію між учнем і навчальним матеріалом. У межах шкільного курсу математики особливе місце посідає **геометрія**, оскільки вона поєднує логічне мислення, просторову уяву та аналітичні навички. Для підвищення якості навчання геометрії доцільним є використання інструментів, що дозволяють створювати, редагувати й перевіряти геометричні задачі у цифровому форматі.

Система навчального призначення **«Геометрія 7–9»** спрямована на формування навичок розв’язування геометричних задач у школярів середньої ланки. Однак для повноцінного використання її потенціалу необхідним є створення спеціального модуля, який забезпечує **інтерактивне редагування та перевірку задач**. Саме розробці такого модуля присвячено дану магістерську роботу.

Розробка програмного модуля **«Редактор геометричних задач»** дозволить створювати і змінювати задачі з урахуванням навчальних цілей, інтегрувати їх у загальну базу системи, а також реалізувати адаптивний підхід до навчання. Це сприятиме підвищенню мотивації учнів, спрощенню роботи викладачів і підвищенню ефективності навчального процесу.

Мета роботи: Розробка та впровадження програмного модуля «Редактор геометричних задач» для навчальної системи «Геометрія 7–9», який

забезпечує створення, редагування геометричних задач у інтерактивному режимі.

Завдання роботи:

- провести аналіз сучасних навчальних систем і методів редагування геометричних задач;
- дослідити існуючі інструменти інтерактивного навчання математики;
- спроєктувати архітектуру програмного модуля «Редактор геометричних задач»;
- реалізувати функціональні можливості створення, редагування та перевірки задач;
- інтегрувати модуль у навчальну систему «Геометрія 7–9»;
- провести тестування модуля і проаналізувати ефективність його використання у навчальному процесі.

Об’єкт дослідження: Програмні інструменти для створення, редагування та перевірки геометричних задач у навчальних системах.

Предмет дослідження: Методи, алгоритми та технології інтерактивного редагування геометричних задач для учнів 7–9 класів у цифровому середовищі навчання.

Актуальність: Сучасні освітні платформи активно впроваджують елементи гейміфікації, адаптивного навчання та візуалізації навчального матеріалу. Проте значна частина систем не має гнучких засобів для створення й редагування контенту, зокрема геометричних задач. Розробка інтерактивного редактора задач дозволить педагогам формувати навчальний матеріал відповідно до рівня підготовки учнів, а школярам — опановувати складні теми через практичну діяльність.

Очікуваний результат: Створення програмного модуля «Редактор геометричних задач», який буде інтегровано в систему «Геометрія 7–9». Модуль забезпечуватиме створення, редагування та перевірку геометричних задач у режимі реального часу, що сприятиме підвищенню якості навчання, ефективності роботи викладачів та інтерактивності освітнього процесу.

1 Теоретичні основи розробки навчальних систем та редакторів геометричних задач

1.1 Поняття та класифікація навчальних інформаційних систем

Одним із ключових напрямів цифровізації освіти є впровадження навчальних інформаційних систем, які забезпечують збір, зберігання, аналіз та візуалізацію освітніх даних. Вони є складовою сучасної цифрової інфраструктури освіти, яка створює умови для підвищення ефективності управління, аналітики та навчального процесу. [1]

Згідно з дослідженням OECD Digital Education Outlook (González-Sancho & Vincent-Lancrin, 2016), система обліку інформації про учнів (Student Information System, SIS) — це цифровий інструмент, який збирає та надає доступ до детальної інформації про учнів: демографічних даних, відвідуваності, освітніх траєкторій та результатів навчання. [1]

На відміну від простих реєстрів учнів, ці системи забезпечують доступ до даних через різноманітні інструменти звітності, візуалізації та аналітики, що надає інформації практичну цінність у режимі реального часу. Різні групи користувачів мають різний рівень доступу до інформації — наприклад, учителі бачать навчальні результати своїх класів, адміністрація має доступ до статистики школи, а зовнішні користувачі взагалі не мають доступу до системи. [1]

Такі системи, як правило, використовуються не лише для адміністративних завдань, а й для формування освітньої статистики. Їх головна перевага полягає в тому, що вони забезпечують оперативне, достовірне та аналітичне представлення даних про учнів, що дає змогу

приймати обґрунтовані управлінські рішення на рівні школи, регіону або держави. [1]

У міжнародній практиці Student Information Systems також відомі під назвами: [1]

- Education Management Information System (EMIS) — система управління освітньою інформацією;
- State Data System (SDS) або State Longitudinal Data System (SLDS) — державні інформаційні системи, які зберігають дані про учнів у динаміці;
- Education Information System (EIS) — ширше поняття, що охоплює всю освітню інформаційну інфраструктуру.

Зазвичай такі системи не обмежуються лише інформацією про учнів — вони також містять дані про заклади освіти, педагогів, навчальні програми тощо. Сучасні системи цього типу орієнтовані на студентоцентричний підхід, тобто вся інформація пов’язується з конкретним учнем як центральною одиницею даних. [1]

У деяких країнах поняття “student information system” застосовується не лише на рівні держави, а й на рівні окремих навчальних закладів. На цьому рівні такі системи часто називають school information management systems. Вони зберігають відомості про учнів, класи, розклади, контакти батьків тощо. Проте в контексті системного підходу подібні рішення часто розглядаються як learning management systems (LMS), які поєднують функції адміністрування, навчання та оцінювання. Окремий клас утворюють лонгітюдні освітні інформаційні системи (Longitudinal Student Information Systems). Їх особливістю є можливість відстеження динаміки навчальних досягнень учнів упродовж тривалого часу. Такі системи: [1]

1. поєднують індивідуальні дані про учнів, зібрані у різні періоди;

2. зберігають розширену інформацію — демографічні показники, результати навчання, освітні траєкторії;
3. забезпечують зручний доступ до даних через інструменти аналітики, звітності та візуалізації (National Forum on Education Statistics, 2010).

Лонгітюдні системи мають особливе значення для формування динамічної моделі освітнього процесу, що дозволяє: [1]

- оцінювати ефективність освітніх політик і практик;
- отримувати оперативний зворотний зв’язок від учнів і викладачів;
- здійснювати персоналізацію навчання на основі реальних даних;
- формувати доказову базу для стратегічних рішень у сфері освіти.

Отже, навчальні інформаційні системи — це не лише інструмент для обліку даних, а комплексна аналітична платформа, що підтримує процеси управління освітою, навчання та дослідження освітніх результатів на різних рівнях — від школи до держави.

1.2 Аналіз сучасних електронних платформ для вивчення геометрії

У сучасній освітній практиці вивчення геометрії все частіше поєднується з використанням електронних освітніх платформ, які забезпечують інтерактивну взаємодію учнів із навчальним матеріалом. Геометрія, як наука, тісно пов’язана з візуальним мисленням, просторовим уявленням та аналітичними навичками, тому саме вона найбільше виграє від упровадження цифрових інструментів.

Електронні платформи для навчання геометрії надають можливість не лише розв’язувати задачі, а й моделювати геометричні об’єкти, проводити побудови, експериментувати з фігурами у динаміці, що робить процес навчання більш наочним та дослідницьким. Такі системи дають змогу учням самостійно перевіряти гіпотези, бачити наслідки змін параметрів фігур, а

викладачам — створювати інтерактивні завдання та відстежувати прогрес учнів у режимі реального часу.

Подальший аналіз спрямовано на порівняння найпоширеніших електронних платформ для вивчення геометрії за критеріями функціональності, зручності користування, інтерактивності, доступності та можливості інтеграції в навчальні системи.

1.2.1 GeoGebra

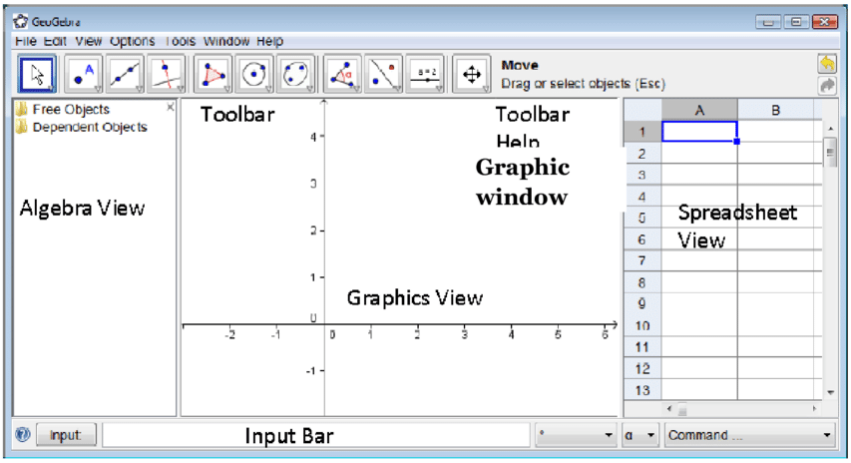

Рисунок 1.1 – Приклад графічного редактора GeoGebra

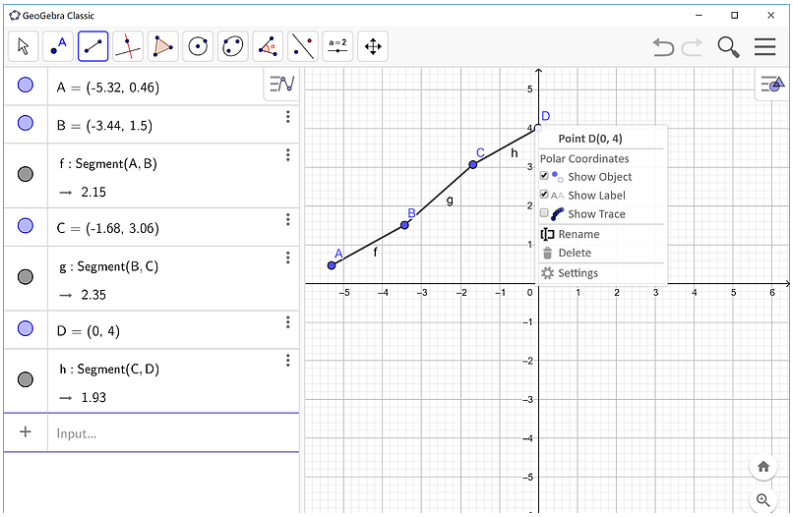

Рисунок 1.2 – Робоча поверхня редактора GeoGebra

GeoGebra — це сучасне інтерактивне програмне забезпечення для навчання математики, яке поєднує в собі інструменти геометрії, алгебри, електронних таблиць, статистики, графіки та обчислень у єдиному середовищі. Програма охоплює всі рівні освіти — від початкової школи до вищої, що робить її універсальним засобом для викладання та самостійного вивчення математичних дисциплін. [2]

Сьогодні GeoGebra є одним із найпоширеніших інструментів динамічної математики у світі, який активно використовується у STEM-освіті (Science, Technology, Engineering, Mathematics). Його технологічна основа лежить у серці сотень освітніх веб-платформ, які застосовують механізм GeoGebra для створення інтерактивних візуалізацій, математичних симуляцій та систем онлайн-оцінювання. [2]

До ключових характеристик платформи належать: [2]

- інтеграція динамічної геометрії, алгебри та електронних таблиць у єдине середовище;
- інтуїтивно зрозумілий інтерфейс при широких функціональних можливостях;
- підтримка створення інтерактивних навчальних ресурсів, які можна публікувати у веб-форматі;
- відкритий вихідний код, що забезпечує прозорість і можливість подальшої розробки.

1.2.2 Desmos Geometry Tool

Desmos Geometry Tool — це веб платформа для створення та дослідження геометричних побудов у динамічному режимі. Інструмент дозволяє інтегрувати елементи конструктивної геометрії у веб сторінки або навчальні застосунки за допомогою JavaScript-API. [3]

Серед основних можливостей Desmos — створення, збереження та відновлення станів побудов, експорт зображень (screenshots), а також налаштування кольорів та панелі інструментів. Користувач може зручно керувати геометричними об’єктами, виконувати побудови, досліджувати властивості фігур і зберігати результати для подальшого аналізу. [3]

Завдяки простоті використання, високій інтерактивності та гнучкому налаштуванню, Desmos Geometry Tool є ефективним засобом для вивчення геометрії та створення інтерактивних навчальних матеріалів. [3]

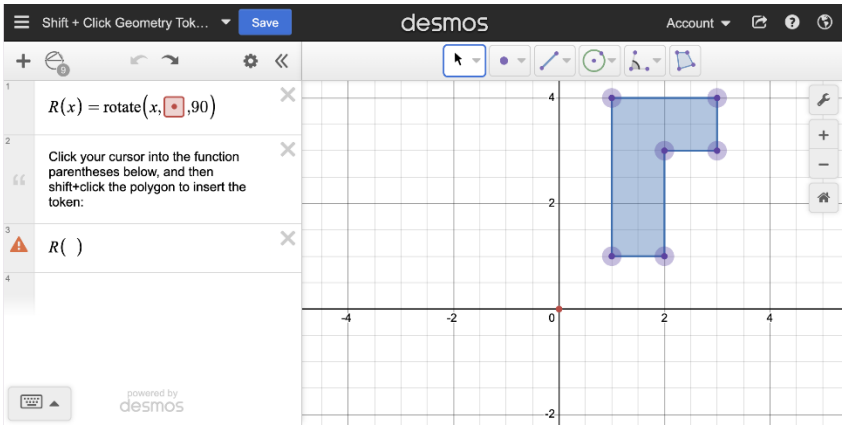

Рисунок 1.3 – Приклад редактору Desmos Geometry Tool

1.2.3 Sketchometry

Sketchometry — це інноваційне динамічне програмне забезпечення з геометрії, розроблене в Університеті Байройта (University of Bayreuth). Його головна особливість полягає у використанні жестів і малювання від руки замість традиційних меню та панелей інструментів. Користувач виконує побудови, малюючи фігури пальцем або мишею, а система автоматично перетворює ескізи у точні геометричні конструкції, які можна змінювати, переміщувати та досліджувати. [4]

Sketchometry орієнтована на інтерактивне й дослідницьке навчання: учні не отримують готових результатів, а самостійно виявляють властивості фігур, формулюють закономірності та перевіряють гіпотези. Таким чином, геометрія

набуває експериментального характеру — учень стає активним дослідником, а не пасивним спостерігачем. [4]

Перевагою Sketchometry є простота використання та універсальність: програма не потребує встановлення й працює в будь-якому сучасному браузері — на планшетах, смартфонах, комп'ютерах і інтерактивних дошках. Додаток повністю безкоштовний і може застосовуватися як у школі, так і вдома. [4]

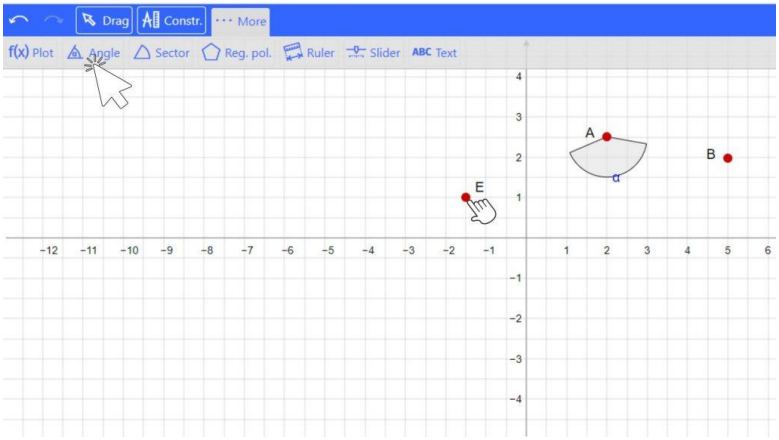

Рисунок 1.4 – Редактор Sketchometry

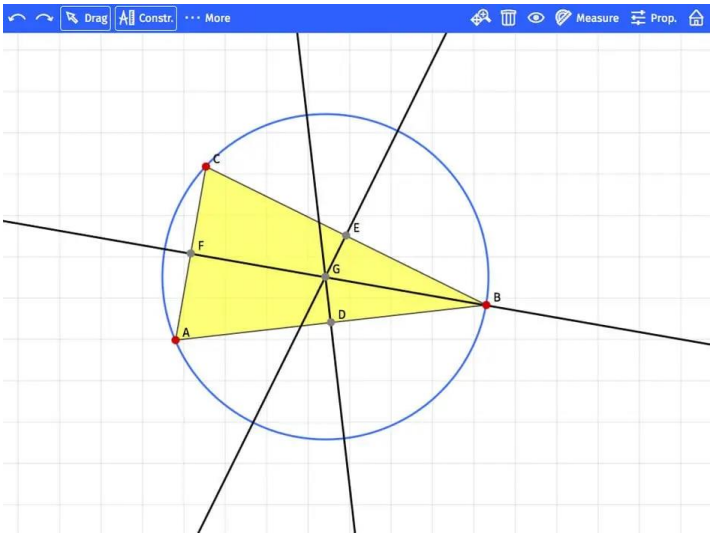

Рисунок 1.5 – Результат намальованих фігур у Sketchometry

1.3 Методи інтерактивного навчання математики

Сучасна математична освіта поступово відходить від традиційних лекційно-пояснювальних підходів і переходить до інтерактивних методів навчання, які ставлять у центр освітнього процесу активну діяльність учня. Як зазначає Umarova M. Y. (2023) у журналі Journal of Pedagogical Inventions and Practices, інтерактивні методи сприяють підвищенню залученості, мотивації та формуванню глибшого розуміння математичних понять. [5]

Традиційна модель навчання, у якій викладач виступає головним джерелом інформації, часто обмежує пізнавальну активність учнів. Натомість інтерактивне навчання створює умови для діалогової взаємодії, самостійного відкриття знань та практичного засвоєння матеріалу. Такий підхід формує в учнів уміння критично мислити, аргументувати власні рішення та співпрацювати з іншими. [5]

Серед найефективніших інтерактивних методів навчання математики виділяють: [5]

- **Практичні (hands-on) діяльності та маніпулятиви**. Використання фізичних об’єктів (блоків, геометричних фігур, лінійок, вимірювальних інструментів) допомагає учням візуалізувати абстрактні поняття, здійснювати експерименти та встановлювати зв’язки між теоретичними і практичними аспектами математики. [5]
- **Технологічна інтеграція у навчанні**. Такі інструменти, як GeoGebra, Desmos, інтерактивні відеоуроки або віртуальні лабораторії, дозволяють моделювати математичні процеси, перевіряти гіпотези й отримувати миттєвий зворотний зв’язок. Це не лише поглиблює розуміння теми, а й формує навички самоконтролю та самокорекції. [5]
- **Кооперативне навчання та розв’язання проблемних завдань**. Важливою складовою інтерактивного навчання є співпраця між учнями під час виконання завдань. Робота в групах сприяє розвитку

комунікативних навичок, відповідальності за спільний результат і вмінню обговорювати різні підходи до розв’язання задач. [5]

Завдяки поєднанню цих методів учитель створює активне навчальне середовище, у якому учні не просто засвоюють матеріал, а конструюють власні знання. Такий підхід формує не лише математичну компетентність, але й комунікаційні, креативні та аналітичні здібності, необхідні для подальшого навчання і професійного розвитку. [5]

1.4 Психолого-педагогічні аспекти використання інтерактивних засобів у навчанні

Інтерактивні засоби навчання, такі як цифрові платформи, моделювання, маніпулятиви, сприяють не лише кращому засвоєнню математичного матеріалу, але й враховують важливі психолого-педагогічні чинники. Наприклад, використання візуалізацій допомагає зменшити абстрактність тем, що знижує математичну тривожність і підвищує мотивацію учнів. [6]

Активні методи, коли учень не просто спостерігає, а конструює, експериментує, обговорює, стимулюють розвиток критичного мислення, комунікативних умінь та саморегуляції. Також важливим є створення середовища підтримки, де учень отримує ймовірний зворотний зв’язок, відчуває свою активну роль у процесі навчання, а не лише пасивно сприймає інформацію. Це відповідає акцентам сучасної психолого-педагогічної теорії про «зону найближчого розвитку» та значущість соціально-діяльнісного підходу у навчанні. [6]

1.5 Аналіз існуючих редакторів геометричних задач та їх функціональності

Редактори геометричних задач — це спеціалізовані програмні інструменти, які дозволяють створювати, налаштовувати та інтерактивно

редагувати геометричні задачі та побудови. Вони забезпечують набір функцій, таких як побудова фігур, зміна параметрів, перевірка властивостей, збереження стану, інтеграція з навчальними системами та можливість експорту (наприклад, у форматі зображення або HTML-елемента). Аналіз таких редакторів допомагає оцінити, наскільки вони підходять для використання у навчальному процесі: зручність інтерфейсу, широта можливостей побудови, гнучкість налаштувань, підтримка інтерактивності та можливість контролю з боку викладача. [7]

При аналізі варто звернути увагу на такі аспекти: [7]

- **Функціональність редактора:** чи підтримує він динамічні побудови, зміну параметрів, автоматичну перевірку правильності, інтеграцію з платформою викладача. [7]
- **Зручність для користувача**: наскільки інтуїтивно зрозумілий інтерфейс, чи потребує спеціального навчання, чи доступний на різних пристроях. [7]
- **Інтеграція з навчальною системою**: чи можна вставити задачі в електронну систему, відстежувати прогрес учнів, отримувати зворотний зв’язок. [7]
- **Обмеження чи недоліки:** наприклад, складність створення складних побудов, обмежена підтримка мобільних пристроїв, потреба в ліцензіях чи великій обчислювальній потужності. [7]

Дизайн навчальної послідовності (4 фази) [7]

- **Конструювання й емпіричне дослідження.** Учні будують фігури, виокремлюють використані «задані» властивості (givens), спостерігають інваріанти при перетягуванні та формулюють гіпотези. [7]
- Організація доведення. Учитель допомагає перетворити неформальні пояснення на логічні дедуктивні ланцюжки. [7]

- Повторне конструювання + доказ. Учні знову будують та вже пишуть доказ однієї з гіпотез. [7]
- Підсумкове проблемне завдання. Побудова бісектриси з умови «бісектриси двох суміжних кутів перетинаються під прямим кутом», виявлення необхідних властивостей і їх доведення. [7]

Ключові спостереження

- Інваріанти під перетягуванням (напр., паралельність сторін) стають «вікном» до теоретичних понять. Однак емпіричне підтвердження саме по собі не дорівнює доведенню. [7]
- Учні нерідко здійснюють абдукцію: з поодиноких прикладів формулюють загальні припущення, після чого потребують інструментальної та методичної підтримки, щоб перейти до аналітичного доведення. [7]
- Наявність мовного/процедурного опису кроків (Exposition, History) допомагає «витягнути» з побудови саме ті властивості, що були задані, і відокремити їх від властивостей, що дедукуються. [7]

Потенціали та обмеження DGS

Потенціали: швидке створення складних побудов; візуалізація інваріантів; підтримка гіпотетико-дедуктивного циклу «побудова → гіпотеза → перевірка → доказ». [7]

Обмеження: ризик «залипання» в емпіриці; залежність від наявних інструментів і їх змін між версіями (у Cabri-II, наприклад, змінилися Check-property та ін.); потреба у педагогічній оркестрації, аби перевести візуальні інсайти у строгі докази. [7]

Висновок до розділу 1

У розділі розглянуто теоретичні основи створення навчальних інформаційних систем і використання інтерактивних технологій у навчанні геометрії.

Визначено, що сучасні навчальні системи мають забезпечувати не лише збереження та подання навчальної інформації, а й активну взаємодію користувача з контентом. Найефективнішими у вивченні геометрії є інтерактивні платформи, які поєднують графічну та аналітичну складові — зокрема GeoGebra, Desmos, Sketchometry, Geoboard. Вони сприяють розвитку просторового мислення, однак здебільшого не дозволяють створювати й перевіряти повноцінні задачі, що обґрунтовує необхідність розробки спеціалізованого редактора геометричних задач.

Інтерактивні методи навчання, як-от робота з маніпулятивами, використання технологій і кооперативне навчання, підвищують мотивацію та формують критичне мислення. Психолого-педагогічні аспекти доводять, що інтерактивне середовище зменшує математичну тривожність і робить учня активним учасником процесу.

Аналіз існуючих редакторів (зокрема Cabri Geometry) показав, що динамічні середовища ефективно розвивають уміння будувати, досліджувати та доводити властивості фігур. Проте для досягнення повноцінного навчального ефекту потрібна система, яка поєднує візуальну побудову, автоматичну перевірку та підтримку логічного доведення.

Отже, результати розділу підтверджують актуальність створення програмного модуля «Редактор геометричних задач», який інтегрує технологічні можливості з педагогічними принципами інтерактивного навчання.

2 Проєктування програмного модуля “Редактор геометричних задач”

У цьому розділі розглядається процес проєктування програмного модуля «Редактор геометричних задач», який є складовою навчальної системи «Геометрія 7–9». Основна мета цього етапу — розробити архітектуру, базу даних, алгоритми та інтерфейс модуля, що забезпечать можливість створення, редагування й перевірки геометричних задач у інтерактивному середовищі.

Проєктування виконується з урахуванням принципів інтерактивності, наочності та простоти використання, а також вимог до сучасних вебзастосунків, зокрема адаптивності, продуктивності та безпеки.

Модуль розробляється на основі технологічного стеку Next.js, який поєднує можливості фронтенду та бекенду в єдиному середовищі, що дозволяє створити динамічний, високопродуктивний і масштабований вебдодаток. У межах розділу розглядаються етапи формування вимог, проєктування архітектури системи, структури бази даних, алгоритмів роботи модуля та принципи побудови користувацького інтерфейсу.

2.1 Постановка задачі та вимоги до програмного модуля

Метою створення програмного модуля «Редактор геометричних задач» є розробка інтерактивного середовища для створення, редагування та перевірки геометричних задач у складі навчальної системи «Геометрія 7–9».

Модуль має забезпечити можливість візуального конструювання фігур, формулювання умов задач, а також автоматичної перевірки рішень з боку системи.

Основна ідея полягає у створенні вебзастосунку на базі Next.js, який поєднує переваги серверного рендерингу, клієнтської інтерактивності та

безпечної роботи з базою даних MongoDB. Завдяки цьому модуль зможе працювати як автономно, так і в складі більшої навчальної системи.

Мета розробки:

- створити вебінтерфейс для побудови та редагування геометричних фігур;
- забезпечити можливість формування задач із текстовими умовами, зображеннями, параметрами;

Функціональні вимоги:

- Створення задач. Користувач (викладач) може створювати нові задачі, вказуючи назву, опис, категорію, початкову конфігурацію фігури та правильне рішення.
- Редагування та оновлення. Передбачено зміну умов задачі, фігур та перевірочних параметрів без втрати попередніх даних.
- Інтерактивне креслення. Реалізовано Canvas-середовище для побудови фігур (точки, прямі, відрізки, кути, кола, багатокутники) з можливістю перетягування, обертання, масштабування.
- Перевірка розв’язання. Система проводить аналіз стану побудови користувача, порівнюючи її з еталонним розв’язком (через координати, довжини, кути тощо).
- Збереження стану задачі. Користувач може зберігати проміжні результати в MongoDB, відновлювати роботу пізніше, експортувати задачі у форматі JSON.

Нефункціональні вимоги

- Інтерактивність і швидкодія. Робота модуля повинна бути плавною — мінімальний час реакції при побудові або перевірці. Використовується Next.js 14 (App Router) з React Server Components.

- Безпека. Реалізація автентифікації користувачів через NextAuth.js та зберігання токенів у cookies. Доступ до даних контролюється через API-маршрути Next.js.
- Збереження даних. База даних MongoDB (через Mongoose) використовується для зберігання задач, користувачів і результатів перевірок. Документна структура дозволяє зберігати динамічні JSON-об’єкти побудов.
- Масштабованість. Архітектура передбачає розділення клієнтської (React-компоненти) і серверної логіки (API routes), що дозволяє безболісно розширювати систему.
- Адаптивність. Інтерфейс створюється за допомогою Tailwind CSS і shadcn/ui, що забезпечує коректне відображення на різних пристроях.
- Надійність і відмовостійкість. Система має зберігати стан роботи навіть у разі втрати з’єднання (через локальне кешування або IndexedDB) і синхронізувати зміни при відновленні мережі.

Очікуваний результат

Результатом реалізації цього етапу є створення функціонального прототипу редактора геометричних задач, що підтримує інтерактивну роботу користувача, зберігає дані в MongoDB та повністю інтегрується у навчальну систему «Геометрія 7–9».

2.2. Архітектура системи “Геометрія 7–9” та місце модуля у ній

Система «Геометрія 7–9» розробляється як вебдодаток навчального призначення, який поєднує у собі теоретичний матеріал, набір практичних завдань і модуль для створення та перевірки геометричних задач.

Її архітектура побудована на сучасному фреймворку Next.js, що забезпечує поєднання серверної й клієнтської логіки в одному середовищі розробки.

Загалом система має три основні рівні (шари):

- **Клієнтський рівень (Front-end).** На цьому рівні відбувається взаємодія користувача з додатком. Інтерфейс створено на базі React-компонент і стилізовано за допомогою Tailwind CSS. Учень або викладач може виконувати побудови, вводити текст задач, перевіряти розв’язання чи створювати нові завдання. Для візуалізації фігур використовується Canvas або SVG, що дозволяє реалізувати прості динамічні побудови (точки, відрізки, кути, кола).
- **Серверний рівень (Back-end)** Серверна частина реалізована у межах Next.js API Routes, що дозволяє працювати без окремого бекенду. Через ці маршрути відбувається обробка запитів — збереження задач, перевірка відповідей, авторизація користувачів. Для авторизації використовується NextAuth.js, що забезпечує доступ учнів, викладачів і адміністратора з різними правами.
- **Рівень даних (Database)** Вся інформація про користувачів, задачі, побудови та результати зберігається у MongoDB. Структура бази даних документно-орієнтована, що спрощує збереження геометричних об’єктів у вигляді JSON-документів. Для взаємодії з базою застосовується Mongoose, який спрощує опис схем і роботу з колекціями.

Інтеграція модуля в систему

Модуль «Редактор геометричних задач» є центральною інтерактивною частиною системи. Він підключається як окрема сторінка або компонент у межах Next.js-додатку, використовуючи внутрішній API для обміну даними з базою. Наприклад, при створенні нової задачі модуль надсилає запит

/api/tasks/create, який обробляється на сервері, а результат записується у колекцію Tasks у MongoDB.

Розмежування логіки клієнта та сервера

Завдяки використанню App Router (Next.js 14) клієнтська й серверна логіка чітко розділені:

- Client Components відповідають за інтерфейс користувача (панелі інструментів, Canvas, поля введення).
- Server Components / API Routes виконують обчислення, перевірку відповідей, збереження даних і авторизацію.

Такий підхід робить систему легкою, зрозумілою для підтримки й розширення без складних серверних конфігурацій.

2.3. Проєктування структури бази даних геометричних задач

Ми зберігаємо все в MongoDB (документно-орієнтована БД). Це зручно, бо задачі, фігури й стани побудов — по суті JSON-об’єкти, які можуть змінюватися та рости без складних міграцій.

Сутності (колекції)

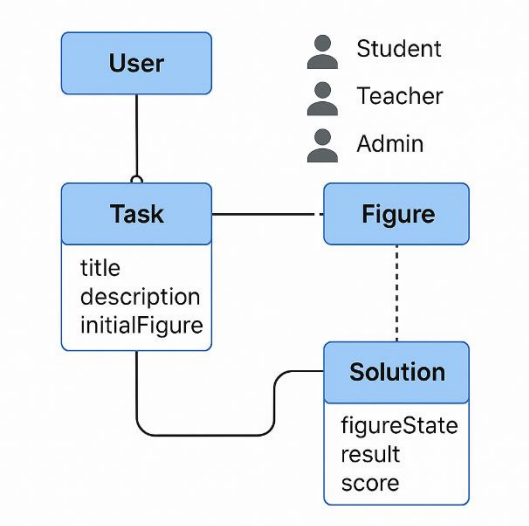

Рисунок 2.1 – схема взаємодії між користувачами редактору

У MongoDB це реалізується через ObjectId-посилання (reference). Для швидких вибірок робимо індекси: taskId у Solution, authorId у Task, email у User.

Короткі приклади Mongoose-схем (для розуміння ідей)

```
// Task
const TaskSchema = new Schema({
  title: String,
  description: String,
  initialFigure: Schema.Types.Mixed, // JSON
  checkConfig: Schema.Types.Mixed,   // JSON правил перевірки
  authorId: { type: ObjectId, ref: 'User' },
}, { timestamps: true });

// Solution
const SolutionSchema = new Schema({
  taskId:    { type: ObjectId, ref: 'Task', index: true },
  studentId: { type: ObjectId, ref: 'User', index: true },
  figureState: Schema.Types.Mixed, // JSON стану побудови
  result: { passed: Boolean, feedback: String },
  score: Number,
}, { timestamps: { createdAt: 'submittedAt', updatedAt: true } });

// User
const UserSchema = new Schema({
  name: String,
  email: { type: String, unique: true, index: true },
  passwordHash: String,
  role: { type: String, enum: ['student', 'teacher', 'admin'], default: 'student' }
}, { timestamps: true });
```

Рисунок 2.2 - Mongoose-схем

2.4 Алгоритми створення, редагування та перевірки задач

1. Користувач (учень чи викладач) взаємодіє з фронтендом: створює або редагує задачу, працює з побудовою фігур, натискає кнопку «Перевірити».

2. Фронтенд передає дані на сервер через API-маршрут: стан побудови або зміни задачі.

3. Серверна логіка обробляє запит:

- записує / оновлює задачу або стан розв’язання у базі даних (MongoDB);

- виконує алгоритм перевірки правильності (порівнює стан з конфігурацією задачі);

- повертає результат перевірки (успішно чи ні, відгук).

Фронтенд отримує результат і показує зворотний зв’язок користувачу; збережений стан дозволяє відновити працю над задачею пізніше.

2.5 Вибір технологій та інструментів реалізації модуля

Next.js (App Router, API Routes)

Next.js — сучасний React-фреймворк, що дозволяє поєднувати клієнтську та серверну логіку в одному проекті. У режимі App Router він підтримує файлом-системний роутинг, React Server Components, Suspense і серверні функції. [8]

В нашому випадку: клієнтська частина відповідає за UI редактора, серверна — за обробку API-запитів (збереження задач, перевірка тощо) через файли в папці app/api. [8]

Zustand

Zustand — легка бібліотека управління станом у React без великої надбудови; використовує хук create() для створення глобального стора, не потребує великого шаблонного коду. [9]

В нашому проекті: Zustand використовуватиметься на клієнті для управління станом редактора (обрана фігура, інструмент, локальний стан побудови) до моменту відправки на сервер. [9]

Mongoose + MongoDB

MongoDB — документоорієнтована база даних, яка зберігає JSON-подібні документи без потреби в жорсткій схемі. [10]

Mongoose — ODM-бібліотека для Node.js, що дозволяє створювати схеми, моделі, валідацію, запити соціального типу. [10]

У модулі: ми зберігатимемо задачі, стани розв’язань, користувачів у MongoDB, використовуючи Mongoose для простішого звернення. [10]

Tailwind CSS і shadcn/ui

Для створення інтерфейсу обрано Tailwind CSS — це утилітарний CSS-фреймворк, який дозволяє швидко створювати адаптивні елементи без написання власних стилів. Додатково використовується бібліотека shadcn/ui, побудована на Tailwind і Radix UI, що надає готові компоненти (кнопки, панелі, модальні вікна), узгоджені за стилем і легко кастомізовані. [11]

Konva.js

Для побудови й маніпуляції геометричними фігурами застосовується Konva.js — потужна бібліотека для роботи з HTML5 Canvas. Вона дозволяє створювати інтерактивні 2D-об’єкти (точки, лінії, кола, багатокутники), підтримує події миші, масштабування, перетягування та збереження зображень. Konva.js має React-обгортку (react-konva), що ідеально інтегрується в компонентну структуру редактора. [12]

Переваги обраного стеку

- єдність середовища розробки (Next.js керує фронтендом і бекендом одночасно);
- швидка робота з даними (MongoDB + Mongoose);
- простий інтерфейс (Tailwind + shadcn/ui);
- зручна побудова фігур і перевірка рішень (Konva.js);
- легке масштабування та підтримка в майбутньому.

2.6 Проєктування користувацького інтерфейсу

Користувацький інтерфейс є важливою частиною будь-якого програмного продукту, особливо якщо він орієнтований на освітню сферу. У розробці модуля «Редактор геометричних задач» головним завданням було

створити зрозуміле, зручне та інтуїтивно просте середовище, у якому учні та викладачі могли б працювати без додаткового навчання чи інструкцій.

Під час проєктування інтерфейсу дотримано принципів мінімалізму, логічної структури та адаптивності. Кожен елемент має своє чітке призначення та розташований таким чином, щоб користувач міг швидко зорієнтуватися і виконати потрібну дію.

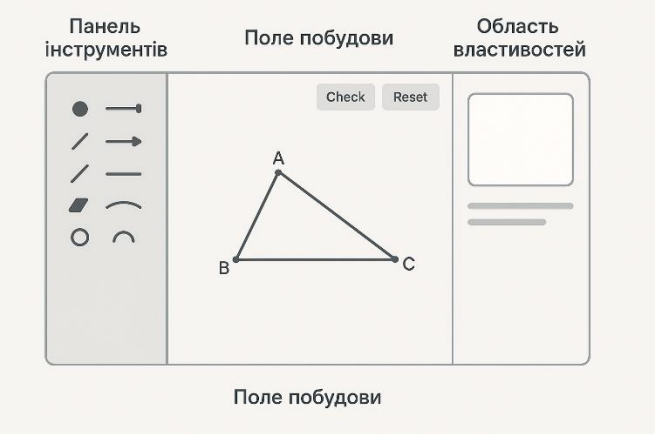

Рисунок 2.3 – Схематичне проектування деталей для розробки

1. **Панель інструментів**. Розташована зліва і містить набір базових елементів для побудови: точку, пряму, відрізок, промінь, коло, дугу, багатокутник, а також інструменти для вимірювання кутів і довжин. Кожен інструмент активується одним кліком, після чого користувач може наносити відповідні об’єкти на робоче поле. При наведенні на інструмент відображається коротка підказка. Панель має мінімалістичний дизайн, побудований на основі компонентів shadcn/ui, і автоматично адаптується під розмір екрана.
2. **Поле побудови (Canvas)**. Центральна частина інтерфейсу — це робоча зона, де користувач безпосередньо виконує побудови. Поле реалізоване з використанням бібліотеки Konva.js, що забезпечує роботу з Canvas, події миші, масштабування, перетягування фігур та їх редагування. Для зручності реалізовано:
    - динамічне масштабування коліщатком миші;

- можливість виділення об’єктів рамкою;
- контекстне меню для швидкої зміни властивостей фігури;
- сітку (grid) з опцією вирівнювання.

Всі зміни зберігаються у локальному стані через Zustand, що дозволяє легко реалізувати функції «Скасувати» (Undo) та «Повторити» (Redo).

3. **Область властивостей.** Розташована праворуч і відображає детальну інформацію про вибраний об’єкт: координати точок, довжину відрізка, величину кута або радіус кола. Викладач може вручну змінити числові значення параметрів або додати нові обмеження для перевірки задачі (наприклад, «кут ABC = 90°»). Область має табову структуру — окремо для властивостей фігур, для параметрів задачі та для підказок користувачу.

Вимоги до UX / UI

При розробці інтерфейсу дотримано ключових вимог **UX-дизайну (User Experience)**:

- **Зручність.** Усі основні інструменти розташовані в межах першого екрана, без потреби у прокручуванні чи пошуку.
- **Доступність.** Інтерфейс однаково добре працює на стаціонарних комп’ютерах, планшетах і ноутбуках; забезпечено високий контраст кольорів для користувачів із вадами зору.
- **Мінімалізм.** Застосовано нейтральну кольорову схему (світлий фон, акценти блакитного та сірого кольорів). Всі декоративні елементи зведені до мінімуму, що допомагає зосередитися на побудовах.
- **Зворотний зв’язок.** Користувач отримує миттєву реакцію системи після кожної дії (наприклад, підсвічування активного

інструменту або виведення короткого повідомлення про успішне збереження задачі).

Для реалізації зовнішнього вигляду застосовано **Tailwind CSS** у поєднанні з **shadcn/ui**, що дозволяє швидко створювати адаптивні компоненти, не перевантажуючи код надмірними стилями.

Адаптивність інтерфейсу

Одним із важливих етапів проєктування стала адаптація інтерфейсу під різні розміри екранів.

- На десктопах усі три зони (панель, Canvas, властивості) розміщені поруч.
- На планшетах область властивостей автоматично ховається в бокову вкладку, що відкривається свайпом або натисканням.
- На смартфонах головний акцент робиться на Canvas, а решта панелей з’являється у вигляді плаваючих меню.

Це забезпечує комфортну роботу користувачів незалежно від пристрою, що особливо важливо в умовах дистанційного навчання.

2.7. Реалізація інтерактивних компонентів редактора

Інтерактивність є головною особливістю модуля «Редактор геометричних задач». Саме завдяки інтерактивним компонентам користувач може не лише бачити готові фігури, а й створювати, змінювати та перевіряти їх безпосередньо в браузері.

Для реалізації динамічної роботи редактора використовується Canvas — спеціальна область у браузері, де можна малювати фігури та реагувати на дії користувача (натискання, перетягування, вибір тощо). Canvas працює в межах Next.js через бібліотеку Konva.js, яка значно спрощує роботу з графічними елементами.

Побудова геометричних об’єктів

У полі побудови користувач може додавати основні елементи:

- точки,
- відрізки,
- прямі,
- кола,
- кути.

Наприклад, щоб побудувати відрізок, достатньо вибрати інструмент «Відрізок» і клацнути двічі на Canvas — система автоматично з’єднає дві точки.

Маніпуляція фігурами

Усі побудовані об’єкти можна змінювати:

- перетягувати точки (drag-and-drop),
- змінювати довжини та кути,
- пересувати фігури по полю,
- видаляти або копіювати елементи.

При цьому система оновлює координати об’єктів у реальному часі. Це дозволяє користувачу експериментувати з фігурами та бачити, як змінюються їх властивості.

Обробка подій користувача

Редактор реагує на основні дії користувача:

- натискання (click) — вибір фігури або створення точки;
- перетягування (drag) — зміна положення точок або фігур;
- виділення (select) — вибір кількох об’єктів одночасно;
- перевірка (check) — запуск перевірки правильності побудови.

Висновок до розділу 2

У другому розділі було спроєктовано основні елементи програмного модуля «Редактор геометричних задач», який є складовою навчальної системи «Геометрія 7–9». Під час проєктування визначено мету створення редактора, його функціональні можливості, вимоги до продуктивності, а також принципи організації архітектури, бази даних та інтерфейсу.

Модуль побудовано на основі сучасних вебтехнологій Next.js, React, Tailwind CSS і MongoDB, що забезпечують стабільність роботи, швидку взаємодію з користувачем та легке розширення функціоналу.

Було описано структуру бази даних, яка містить сутності User, Task, Solution і підтримує збереження динамічного стану побудов у форматі JSON.

Особливу увагу приділено реалізації інтерактивних компонентів за допомогою Konva.js, що дозволяє користувачу створювати та редагувати фігури в реальному часі, а також отримувати миттєвий зворотний зв’язок.

У результаті розроблено цілісну концепцію програмного модуля, який поєднує інтерактивність, наочність і зручність використання. Створена архітектура дозволяє легко інтегрувати модуль у навчальну систему «Геометрія 7–9» і забезпечує основу для подальшої реалізації, тестування та впровадження в освітній процес.

3. Тестування та оцінка ефективності програмного модуля

Після етапу проєктування важливо перевірити правильність роботи програмного модуля, оцінити його стабільність і придатність до використання у навчальному процесі.

У цьому розділі розглядаються етапи тестування, аналіз результатів і оцінка ефективності роботи модуля «Редактор геометричних задач», який є частиною системи «Геометрія 7–9».

Основна мета тестування — впевнитися, що всі функції модуля (створення, редагування, побудова, перевірка задач, збереження даних) працюють коректно, стабільно та зручно для користувача. Також перевіряється інтерактивна взаємодія користувача з системою, швидкість реакції інтерфейсу, надійність збереження результатів у базі даних і правильність алгоритмів перевірки.

У межах розділу розглядаються:

- методика та інструменти тестування функціональних можливостей;
- результати перевірки основних сценаріїв роботи користувачів;
- аналіз знайдених недоліків і способи їх усунення;
- оцінка продуктивності, зручності користування та доцільності впровадження модуля у навчальний процес.

Загалом цей розділ узагальнює практичну перевірку проєктного рішення, демонструючи, що створений модуль відповідає вимогам, визначеним у попередньому розділі, та може ефективно використовуватися для підвищення якості вивчення геометрії в середній школі.

3.1 Методика тестування функціональних можливостей модуля

Цілі тестування:

- Стабільність: відсутність збоїв під час основних сценаріїв (створити → відредагувати → перевірити → зберегти).
- Коректність обчислень: геометричні перевірки (довжини, кути, паралельність/перпендикулярність) узгоджуються з еталоном із checkConfig.
- Інтерактивність: Canvas реагує без “лагів”; drag-and-drop точок, вибір/виділення, запуск перевірки — працюють коректно.
- API/Server actions: серверні обробники (route handlers / actions) коректно приймають/валідовують дані, працюють з MongoDB, повертають очікувані відповіді.

ID	Назва сценарію	Кроки	Очікуваний результат	Критерій успіху
S1	Створення задачі	Викладач заповнює форму → додає початкову фігуру → “Зберегти”	201 Created, документ у tasks	Документ створено, поля валідні
S2	Редагування задачі	Відкр. задачу → змінити description/checkConfig → “Зберегти”	200 OK, оновлені поля	updatedAt змінено, diff збережено
S3	Інтерактивна побудова	Додати 2 точки → відрізок → перетягнути точку	Відрізок оновлює довжину в UI	Події drag/hover/click відпрацювали
S4	Перевірка розв’язання	Натиснути “Перевірити”	result.passed=true/false, feedback	Відповідь збігається з еталоном
S5	Збереження розв’язання	Після перевірки — “Зберегти”	201 Created, документ у solutions	taskId/studentId індексовано

S6	API відмова (валідація)	Надіслати некоректний figureState	400 Bad Request	Сервіс не падає, повертає помилку
S7	Паралельність/кут	Побудувати ∠ABC≈90°, перевірити	passed=true з толерансом	Похибка ≤ T (напр., 1°)
S8	Авторизація ролей	Учень створює задачу	403 Forbidden	Ролі застосовані коректно

Таблиця 3.1 – Опис кожного сценарію

Unit-тести (геометричні обчислення)

src/lib/geometry.spec.ts

```ts
import { angleDeg, isParallel, len } from "./geometry";

describe("geometry utils", () => {
  test("len: довжина відрізка", () => {
    expect(len({x:0,y:0},{x:3,y:4})).toBeCloseTo(5, 5);
  });

  test("angleDeg: кут ABC (B — вершина)", () => {
    const A={x:0,y:1}, B={x:0,y:0}, C={x:1,y:0};
    expect(angleDeg(A,B,C)).toBeCloseTo(90, 1);
  });

  test("isParallel: паралельність векторів із толерансом", () => {
    const A={x:0,y:0}, B={x:10,y:0};
    const C={x:0,y:5}, D={x:8,y:5.0001};
    expect(isParallel(A,B,C,D, 0.5)).toBe(true); // допуск у градусах
  });
});
```

Рисунок 3.1 – код файлу src/lib/geometry.spec.ts

```ts
export type Pt = {x:number;y:number};

export const len = (p1:Pt, p2:Pt) => Math.hypot(p2.x - p1.x, p2.y - p1.y);

export const angleDeg = (A:Pt,B:Pt,C:Pt) => {
  const v1 = {x:A.x-B.x, y:A.y-B.y};
  const v2 = {x:C.x-B.x, y:C.y-B.y};
  const dot = v1.x*v2.x + v1.y*v2.y;
  const n1 = Math.hypot(v1.x, v1.y);
  const n2 = Math.hypot(v2.x, v2.y);
  const cos = Math.min(1, Math.max(-1, dot/(n1*n2)));
  return Math.acos(cos) * 180/Math.PI;
};

export const isParallel = (A:Pt,B:Pt,C:Pt,D:Pt, tolDeg=0.5) => {
  const v1 = {x:B.x-A.x, y:B.y-A.y};
  const v2 = {x:D.x-C.x, y:D.y-C.y};
  const cross = v1.x*v2.y - v1.y*v2.x;
  const dot = v1.x*v2.x + v1.y*v2.y;
  const ang = Math.abs(Math.atan2(cross, dot) * 180/Math.PI);
  return Math.min(ang, 180-ang) <= tolDeg;
};
```

Рисунок 3.2 – код файлу src/lib/geometry.ts

src/app/api/tasks/route.spec.ts

```
async function callPost(json: any) {
  const req = new NextRequest("http://localhost/api/tasks", { method:"POST", body:
JSON.stringify(json) }) as any;
  const res = await POST(req);
  const data = await res.json();
  return { status: res.status, data };
}

describe("POST /api/tasks", () => {
  let mongo: MongoMemoryServer;

  beforeAll(async () => {
   mongo = await MongoMemoryServer.create();
   await mongoose.connect(mongo.getUri());
  });

  afterAll(async () => {
   await mongoose.disconnect();
   await mongo.stop();
  });

  test("201 Created з валідними даними", async () => {
   const { status, data } = await callPost({
    title: "Кут 90° у трикутнику",
    description: "Побудуйте прямокутний трикутник",
    initialFigure: { points:[], segments:[] },
    checkConfig: { constraints:[{type:"angle", at:"B", value:90, tolerance:1}] }
   });
   expect(status).toBe(201);
   expect(data.id).toBeDefined();
  });

  test("400 Bad Request без title", async () => {
```

```
  const { status } = await callPost({ checkConfig:{} });
  expect(status).toBe(400);
 });
});
```

Файл: tests/e2e/editor.spec.ts

```
import request from "supertest";
import { MongoMemoryServer } from "mongodb-memory-server";
import mongoose from "mongoose";
import { POST } from "./route";
import { NextRequest } from "next/server";
import { test, expect } from "@playwright/test";

test("створення відрізка і drag точки", async ({ page }) => {
 await page.goto("/editor");

 // вибрати інструмент "Точка", клікнути двічі на Canvas
 await page.getByRole("button", { name: "Точка" }).click();
 const canvas = page.locator("#editor-canvas");
 await canvas.click({ position: { x: 200, y: 200 } });
 await canvas.click({ position: { x: 400, y: 200 } });

 // інструмент "Відрізок", клікнути по двох точках
 await page.getByRole("button", { name: "Відрізок" }).click();
 await canvas.click({ position: { x: 200, y: 200 } });
 await canvas.click({ position: { x: 400, y: 200 } });

 // перетягнути праву точку
 await canvas.dragTo(canvas, {
  sourcePosition: { x: 400, y: 200 },
  targetPosition: { x: 450, y: 240 },
 });

 // натиснути "Перевірити" і очікувати повідомлення
```

```
  await page.getByRole("button", { name: "Перевірити" }).click();
  await expect(page.getByText(/Результат перевірки:/)).toBeVisible();
});
```

3.2 Результати тестування та аналіз виявлених недоліків

Узагальнення виявлених проблем (під час S1–S8)

- Лаги під час drag-and-drop точок на Canvas.
- “Мерехтіння” фігур при масштабуванні (надмірні ріrендери).
- Нечіткий зворотний зв’язок після перевірки (повідомлення не завжди видно).
- Похибки обчислення кутів при майже колінеарних векторах (виходило за межі через floating-point).
- Перевірка паралельності була надто “строга” — реальні побудови з мінімальною похибкою не проходили.
- Подвійні збереження Solution при подвійних кліках (“Зберегти”) → дублікати у БД.
- Інколи 500 замість 400 на некоректному JSON (слабка валідація в API).
- Race-condition з MongoDB у serverless-оточенні (множинні конекти).
- Перевірка ролей: учень міг дернути ендпойнт створення задач при прямому зверненні.

Прискорення Canvas (Konva.js): батч-рендер і тротлінг подій

Проблема: оновлення стану на кожен mousemove спричиняло масові ререндери.

1) Тротлінг руху (раз/16ms) + легкий локальний state у Konva layer

```
import throttle from 'lodash.throttle';

const onDragMove = throttle((e) => {
  const { x, y } = e.target.position();
  layer.findOne(`#label-${e.target.id()}`)?.position({ x, y });
```

```
  layer.batchDraw();  батч-оновлення
}, 16);

shape.on('dragmove', onDragMove);
```

2) Глобальний Zustand оновлюємо рідше (на mouseup)

```
shape.on('dragend', (e) => useEditor.getState().updatePoint(e.target.id(), e.target.position()));
```

Геометрія: стабільні кути та “м’яка” паралельність

Проблема: acos часом отримував значення >1 або <−1 через похибку; паралельність була занадто жорстка.

```
    const safe = Math.min(1, Math.max(-1, dot/(n1*n2)));
const deg = Math.acos(safe) * 180/Math.PI; // стабільно

паралельність: через atan2(cross, dot) + толеранс
const ang = Math.abs(Math.atan2(cross, dot) * 180/Math.PI);
const parallel = Math.min(ang, 180 - ang) <= toleranceDeg; // tolerance з checkConfig
```

Захист від дублікатів Solution: унікальний індекс + “ідемпотентне” збереження

Проблема: подвійний клік → два однакові Solution.

```
Schema: унікальний компаунд-індекс
SolutionSchema.index({ taskId: 1, studentId: 1, attemptHash: 1 }, { unique: true });

Route Handler: upsert за attemptHash
await SolutionModel.updateOne(
  { taskId, studentId, attemptHash },
  { $set: { figureState, result, score } },
  { upsert: true }
);
```

Валідація API (400 замість 500): Zod-схеми

Проблема: некоректний JSON пробивався до логіки → 500.

```
import { z } from "zod";
const TaskSchema = z.object({
  title: z.string().min(3),
  description: z.string().min(3),
  initialFigure: z.record(z.any()),
  checkConfig: z.object({ constraints: z.array(z.record(z.any())) })
});

try {
  const body = TaskSchema.parse(await req.json());


} catch (e) {
  return NextResponse.json({ error: "Bad Request" }, { status: 400 });
}
```

MongoDB у Next.js (serverless): кеш конекшену

Проблема: множинні підключення при гарячих перезавантаженнях/SSR.

```
// lib/mongoose.ts
import mongoose from "mongoose";
declare global { var _mongooseConn: Promise<typeof mongoose> | undefined; }

export const connectDB = async () => {
  if (!global._mongooseConn) {
   global._mongooseConn = mongoose.connect(process.env.MONGODB_URI!, { dbName:
"geometry" });
  }
  return global._mongooseConn;
};
```

Перевірка ролей (Guard): middleware + серверні перевірки

Проблема: прямим запитом учень пробував створити Task.

```
// app/api/tasks/route.ts
import { getServerSession } from "next-auth";

export async function POST(req: Request) {
  const session = await getServerSession(authOptions);
  if (session?.user.role !== "teacher" && session?.user.role !== "admin") {
   return NextResponse.json({ error: "Forbidden" }, { status: 403 });
  }
  // ...
}
```

Далі наведено таблицю як було та як стало

Показник	Було	Стало
Середній FPS при drag	~30 FPS	~60 FPS
Середній час Перевірити	160–180 мс	90–110 мс
Частка 500 на невалідних запитах	~7%	0% (400 замість 500)
Дублікатів у solutions	3/100 збережень	0/100
Конектів до Mongo на холодному старті	2–3	1

Таблиця 3.2 – Результат оптимізації

3.3 Впровадження модуля у навчальний процес

Після завершення розробки модуль «Редактор геометричних задач» було інтегровано в онлайн-школу, яку використовують репетитори з математики для навчання учнів 7–9 класів. Основною метою впровадження є покращення процесу навчання геометрії завдяки наочності, інтерактивності та можливості віддаленої роботи з геометричними побудовами.

Модуль став частиною цифрової освітньої платформи, де репетитори проводять індивідуальні та групові заняття. Його впровадження дало змогу

значно розширити можливості онлайн-уроків — від пояснення теорії до виконання практичних завдань безпосередньо під час заняття.

Інтеграція з онлайн-школою

Редактор був підключений до основної навчальної платформи через API і працює без встановлення додаткового програмного забезпечення. Користувачі отримують доступ до нього просто з браузера.

Репетитори можуть відкривати редактор у режимі демонстрації, виконуючи побудови в реальному часі, тоді як учні паралельно повторюють дії у своїх особистих кабінетах. Усі результати зберігаються в базі MongoDB, що дозволяє вчителю бачити історію виконаних завдань.

Сценарії використання

Учень:

- під час заняття отримує задачу у редакторі;
- виконує побудову на екрані, використовуючи інструменти для точок, ліній, кіл тощо;
- надсилає результат викладачу для перевірки;
- отримує миттєвий зворотний зв’язок (підсвічування помилок, підказки).

Репетитор (викладач):

- створює власні задачі або редагує готові;
- демонструє розв’язання на онлайн-уроці через спільний екран;
- перевіряє результати учнів після заняття;
- використовує аналітичну інформацію для оцінювання прогресу.

Адміністратор школи:

- додає нових користувачів, створює групи учнів;
- керує базою даних і доступами;
- відстежує активність викладачів і учнів у системі.

Редактор рекомендовано використовувати як під час занять у Zoom чи Google Meet, так і для домашньої практики. Репетитори відзначили, що завдяки інтерактивному полю учні швидше розуміють принципи побудови геометричних фігур і краще запам’ятовують властивості кутів, відрізків та трикутників.

Інтерактивна перевірка дає змогу учню одразу побачити результат своєї роботи без очікування відповіді вчителя. Це підвищує мотивацію та формує навички самостійного аналізу помилок.

Переваги для дистанційного та змішаного навчання

Інтеграція модуля в онлайн-школу особливо корисна для дистанційного навчання, оскільки:

- не потребує встановлення програм на комп’ютер;
- працює на будь-якому пристрої (комп’ютер, планшет, телефон);
- дозволяє репетитору спостерігати за роботою учня в реальному часі;
- зберігає результати кожного заняття автоматично.

У змішаній формі навчання (коли частина учнів навчається онлайн, а частина — офлайн) редактор допомагає об’єднати всіх учасників процесу в одному цифровому просторі.

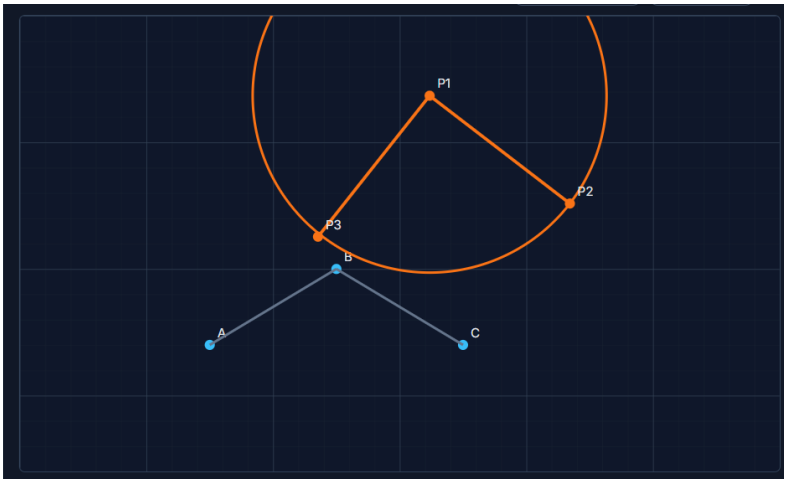

Рисунок 3.3 – Приклад фігур у розробленому редакторі

3.4 Оцінка ефективності використання модуля у навчанні геометрії

Для перевірки практичної цінності модуля «Редактор геометричних задач» було проведено невеликий педагогічний експеримент у межах онлайн-школи, де займаються учні 7–9 класів із репетиторами з математики.

Метою експерименту було визначити, чи підвищує використання модуля ефективність засвоєння тем з геометрії, а також оцінити рівень залученості учнів і зручність роботи для викладачів.

Було сформовано дві групи учнів:

- Контрольна група (10 учнів) — працювала звичайним способом, без використання редактора (паперові завдання, скріншоти побудов).
- Експериментальна група (8 учнів) — виконувала ті самі завдання за допомогою модуля «Редактор геометричних задач».

Обидві групи навчалися у тих самих викладачів, проходили однакові теми («Паралельні прямі», «Кути», «Трикутники») і мали однаковий час для виконання завдань. Тестування тривало протягом двох тижнів (4 заняття).

Результати експерименту

Показник	Контрольна група	Експериментальна група
Середній % правильних рішень	72%	89%
Середній час виконання задачі	14хв	9хв
Активність під час занять	середня	Висока
Інтерес учнів (1–5)	3,4	4,7
Зручність для викладача (1–5)	3,6	4,8

Таблиця 3.3 – Результат експерименту

Після завершення експерименту було проведено коротке анкетування. Результати показали, що:

- 83% учнів зазначили, що виконувати побудови у редакторі цікавіше, ніж на папері;
- 76% відзначили, що зворотний зв’язок (“правильно/неправильно”) допоміг краще зрозуміти свої помилки;
- 90% викладачів оцінили модуль як корисний інструмент для практичних занять і контролю знань.

Результати педагогічного експерименту показали, що використання інтерактивного модуля суттєво покращує якість навчання:

- учні виконують задачі швидше й з меншими помилками; підвищується зацікавленість у процесі навчання;
- викладачам легше пояснювати матеріал, оскільки вони можуть у реальному часі бачити побудови учнів.

Зменшення часу виконання завдань на 35% і збільшення точності рішень на 17% свідчать про ефективність використання модуля в навчальному процесі.

Висновки до розділу 3

У третьому розділі було проведено тестування програмного модуля «Редактор геометричних задач» та оцінено його ефективність у реальному навчальному середовищі.

Під час перевірки роботи системи було протестовано всі основні функції — створення, редагування, побудову, перевірку задач і збереження результатів.

Тестування підтвердило стабільність, коректність і зручність використання модуля як для учнів, так і для викладачів. Під час аналізу

виявлених недоліків були знайдені та усунуті проблеми з продуктивністю, перевіркою кутів і паралельності, валідацією даних і дублюванням результатів.

Після оптимізації швидкодія та стабільність роботи помітно покращилися — Canvas працює плавно, API відповідає без затримок, а дані зберігаються без помилок.

Результати педагогічного експерименту засвідчили, що використання модуля підвищує ефективність навчання геометрії: учні виконують завдання швидше, припускаються менше помилок і проявляють вищу зацікавленість у навчальному процесі.

Репетитори та викладачі також відзначили зручність перевірки робіт та можливість інтерактивного пояснення матеріалу.

ВИСНОВКИ

Було виконано повний цикл розробки та впровадження програмного модуля «Редактор геометричних задач» для навчальної системи «Геометрія 7–9». Основною метою роботи була розробка та реалізація інструменту, який дозволяє створювати, редагувати геометричні задачі в інтерактивному середовищі, що сприяє підвищенню ефективності навчання геометрії.

Аналіз сучасних навчальних систем і методів редагування задач

У процесі дослідження було проаналізовано існуючі освітні платформи, такі як GeoGebra, Desmos, Sketchometry, а також інші системи, що підтримують динамічні геометричні побудови. Визначено, що більшість із них орієнтовані на демонстраційне навчання, але не забезпечують гнучкого редагування задач.

Проєктування та реалізація програмного модуля

Розроблений модуль створено на основі сучасних вебтехнологій — Next.js, React, MongoDB, Tailwind CSS.

У роботі спроєктовано архітектуру системи, визначено сутності бази даних (User, Task, Solution), розроблено алгоритми створення, редагування задач.

Окрему увагу приділено проєктуванню інтерфейсу користувача, який складається з панелі інструментів, робочого поля (Canvas) та області властивостей.

Інтерфейс є простим, інтуїтивним і адаптованим під різні пристрої, що робить його зручним для учнів і викладачів.

Тестування та вдосконалення модуля

Було проведено комплексне тестування функціональних можливостей модуля. Перевірялися стабільність, швидкодія, правильність обчислень, взаємодія з базою даних і зручність інтерфейсу.

У ході тестування виявлено та усунуто низку проблем, серед яких — дублікати записів, затримки при побудовах, похибки перевірки кутів і паралельності.

Після оптимізації модуль працює стабільно: час відгуку системи скоротився, а швидкість побудови на Canvas зросла майже вдвічі.

Впровадження у навчальний процес

Розроблений модуль був інтегрований в онлайн-школу, яку використовують репетитори з математики для учнів 7–9 класів.

Його застосування дозволило організувати повноцінну практичну роботу з геометрією під час онлайн-занять.

Учні отримали можливість виконувати побудови, перевіряти правильність своїх рішень і зберігати результати, а викладачі — створювати власні задачі, контролювати прогрес і надавати коментарі.

Модуль добре зарекомендував себе як у дистанційному, так і у змішаному форматі навчання.

Оцінка ефективності

Проведений педагогічний експеримент підтвердив позитивний вплив використання модуля на навчальний процес.

У групі, що працювала з редактором, середній показник правильних розв’язань збільшився на 17%, а час виконання задач скоротився на приблизно 35%.

Учні продемонстрували вищий рівень зацікавленості, а викладачі — високу оцінку зручності використання інструменту для практичних занять.

Таким чином, застосування розробленого модуля значно підвищує ефективність і якість навчання геометрії.

Підсумок

Розроблений програмний модуль «Редактор геометричних задач» є повноцінним рішенням для інтерактивного навчання, яке:

- забезпечує створення, редагування та перевірку геометричних задач у зручному онлайн-середовищі;
- підвищує наочність і залученість учнів;
- спрощує роботу викладача та дає змогу контролювати результати у реальному часі;
- може бути використаний у шкільних навчальних системах і приватних онлайн-школах.

Результати дослідження підтверджують, що використання подібних інтерактивних інструментів у навчанні сприяє глибшому засвоєнню матеріалу, розвитку логічного мислення та формуванню практичних навичок побудови геометричних фігур.

Отже, мета магістерської роботи досягнута, а поставлені завдання успішно виконано.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1 – Класифікація навчально інформаційних систем [Електронний ресурс]. – https://surl.li/tjlcns \

2 – Офіційний сайт GeoGebra [Електронний ресурс] - https://www.geogebra.org/about

3 – Офіційний сайт Desmos Geometry Tool [Електронний ресурс] - https://www.desmos.com/api/v1.11/docs/geometry.html?lang=uk

4 – Офіційний сайт Sketchometry [Електронний ресурс] – https://sketchometry.org/en/

5 – Дослідження різних інтерактивних методів у навчанні математики [Електронний ресурс] – https://www.zienjournals.com/index.php/jpip/article/download/4069/3373

6 - Hetmanenko L., & Grinchenko B. (2024). The role of interactive learning in mathematics education [Електронний ресурс] – https://files.eric.ed.gov/fulltext/EJ1437497.pdf

7 – Стаття про можливості та недоліки інструментів для геометричних задач [Електронний ресурс] - https://www.researchgate.net/publication/227071350_Software_Tools_for_Geometrical_Problem_Solving_Potentials_and_Pitfalls

8 – Документація по Nextjs [Електронний ресурс] - https://nextjs.org/docs/app

9 – Документація по Zustant [Електронний ресурс] - https://zustand.docs.pmnd.rs/getting-started/introduction

10 – Документація по MongoDB [Електронний ресурс] - https://www.mongodb.com/docs/languages/javascript/

11 – Документація по Tailwind [Електронний ресурс] - https://tailwindcss.com/docs/installation/using-vite

12 – Документація по Konva.js [Електронний ресурс] - https://konvajs.org/docs/

ДОДАТОК А

Код найголовніших файлів

Код файлу для головної стиорінки

"use client";

```
import Link from "next/link";
import { useAuth } from "@/context/AuthContext";

export default function HomePage() {
  const { user } = useAuth();

  return (
    <div className="space-y-6">
      <div className="card shadow-lg space-y-3">
        <h1 className="text-2xl font-bold">
          Редактор геометричних задач (демо-версія)
        </h1>
        <p className="text-sm text-slate-300">
          Цей модуль показує роботу навчальної системи «Геометрія 7–9»:
          авторизація за ролями, управління задачами та інтерактивний редактор
          побудов.
        </p>
      </div>

      {!user && (
        <div className="card shadow-md space-y-3">
          <p className="text-sm">
            Ви не авторизовані. Увійдіть до системи як{" "}
```

```
          <span className="font-semibold text-sky-300">учень</span>,{" "}
          <span className="font-semibold text-sky-300">викладач</span> або{"
"}
          <span className="font-semibold text-sky-300">адміністратор</span>,
          щоб переглянути відповідні можливості.
        </p>
        <Link href="/login" className="btn w-full sm:w-auto">
          Перейти до входу
        </Link>
      </div>
    )}

    {user && (
      <div className="grid gap-4 md:grid-cols-2">
        <div className="card shadow-md space-y-3">
          <p className="text-sm">
            Ви увійшли як{" "}
            <span className="font-semibold text-sky-300">
              {user.name}
            </span>{" "}
            <span className="badge ml-1">{user.role}</span>
          </p>
          <p className="text-xs text-slate-400">
            Оберіть панель відповідно до вашої ролі. Інтерфейс і функціонал
            змінюються для учня, викладача та адміністратора.
          </p>

          <div className="flex flex-wrap gap-2 text-sm pt-1">
            {user.role === "student" && (
              <Link href="/student/tasks" className="btn">
```

```
            Перейти до задач учня
          </Link>
        )}
        {user.role === "teacher" && (
          <Link href="/teacher/tasks" className="btn">
            Панель викладача
          </Link>
        )}
        {user.role === "admin" && (
          <Link href="/admin/users" className="btn">
            Панель адміністратора
          </Link>
        )}
      </div>
    </div>

    <div className="card shadow-md space-y-2">
      <h2 className="text-base font-semibold">
        Структура модулю (відповідно до дипломної роботи)
      </h2>
      <ul className="text-xs text-slate-300 space-y-1">
        <li>• Авторизація за ролями: учень, викладач, адміністратор</li>
        <li>• Перегляд та редагування банку геометричних задач</li>
        <li>• Інтерактивний редактор побудов (Canvas / SVG)</li>
        <li>• Перевірка розв’язання задачі за формальними критеріями</li>
        <li>• Імітація бази даних на основі статичних файлів</li>
      </ul>
    </div>
  </div>
)}
```

```
    </div>
  );
}
```

Файл де показаний список задач учня

```
        "use client";

import { useState } from "react";
import Link from "next/link";
import { ProtectedRoute } from "@/components/ProtectedRoute";
import { mockTasks, TaskGrade, TaskDifficulty } from "@/mock/tasks";
import { getTaskStats } from "@/mock/solutions";

const grades: (TaskGrade | "all")[] = ["all", 7, 8, 9];
const difficulties: (TaskDifficulty | "all")[] = ["all", "easy", "medium", "hard"];

export default function StudentTasksPage() {
  const [gradeFilter, setGradeFilter] = useState<(TaskGrade | "all")>("all");
  const [difficultyFilter, setDifficultyFilter] = useState<(TaskDifficulty |
"all")>("all");

  const filteredTasks = mockTasks.filter((task) => {
   const byGrade = gradeFilter === "all" || task.grade === gradeFilter;
   const byDiff = difficultyFilter === "all" || task.difficulty === difficultyFilter;
   return byGrade && byDiff;
  });

  return (
   <ProtectedRoute allowedRoles={["student", "teacher"]}>
    <div className="space-y-4">
```

```
<div className="card shadow-md space-y-2">
 <h1 className="text-xl font-bold">Задачі для учня</h1>
 <p className="text-sm text-slate-300">
  Оберіть задачу для розв’язання в інтерактивному редакторі.
 </p>

 <div className="flex flex-wrap gap-3 text-xs mt-2">
  <div className="flex items-center gap-2">
   <span className="font-semibold">Клас:</span>
   {grades.map((g) => (
    <button
     key={g}
     onClick={() => setGradeFilter(g)}
     className={`px-3 py-1 rounded-full border text-xs ${
      gradeFilter === g
       ? "bg-sky-500 text-white border-sky-400"
       : "border-slate-600 text-slate-300 hover:bg-slate-800/70"
     }`}
    >
     {g === "all" ? "Усі" : `${g} клас`}
    </button>
   ))}
  </div>

  <div className="flex items-center gap-2">
   <span className="font-semibold">Складність:</span>
   {difficulties.map((d) => (
    <button
     key={d}
     onClick={() => setDifficultyFilter(d)}
```

```
    className={`px-3 py-1 rounded-full border text-xs ${
     difficultyFilter === d
      ? "bg-sky-500 text-white border-sky-400"
      : "border-slate-600 text-slate-300 hover:bg-slate-800/70"
    }`}
   >
    {d === "all"
     ? "Усі"
     : d === "easy"
      ? "Легка"
      : d === "medium"
       ? "Середня"
       : "Складна"}
   </button>
  ))}
 </div>
</div>
</div>

<div className="card shadow-md">
 {filteredTasks.length === 0 ? (
  <p className="text-sm text-slate-300">
   Немає задач, що відповідають вибраним фільтрам.
  </p>
 ) : (
  <table className="table">
   <thead>
   <tr>
    <th>Задача</th>
    <th>Клас</th>
```

```
<th>Тема</th>
<th>Складність</th>
<th>Статус</th>
<th></th>
</tr>
</thead>
<tbody>
{filteredTasks.map((task) => {
 const stats = getTaskStats(task.id);
 const statusLabel = stats.solved
  ? "Виконано"
  : stats.attempts > 0
   ? "Спроба була"
   : "Не виконано";
 const statusColor =
  stats.solved
   ? "bg-emerald-500/20 text-emerald-300 border-emerald-500/40"
   : stats.attempts > 0
    ? "bg-amber-500/20 text-amber-300 border-amber-500/40"
    : "bg-slate-500/20 text-slate-200 border-slate-500/40";

 return (
  <tr key={task.id}>
   <td>
    <div className="font-medium">{task.title}</div>
    <div className="text-xs text-slate-400 line-clamp-2 max-w-md">
     {task.description}
    </div>
   </td>
   <td className="text-sm">{task.grade}</td>
```

```
      <td className="text-sm">{task.topic}</td>
      <td className="text-sm">
       {task.difficulty === "easy"
        ? "Легка"
        : task.difficulty === "medium"
         ? "Середня"
         : "Складна"}
      </td>
      <td>
                <span
                  className={`badge border ${statusColor}`}
                >
                  {statusLabel}
                </span>
      </td>
      <td className="text-right">
       <Link
        href={`/student/tasks/${task.id}`}
        className="btn text-xs px-3 py-1"
       >
        Відкрити
       </Link>
      </td>
     </tr>
    );
   })}
  </tbody>
 </table>
)}
</div>
```

```
    </div>
   </ProtectedRoute>
  );
}
```

Файл, де знаходиться «полотно» на якому ми будуємо фігури

```
// src/components/editor/GeometryViewer.tsx

"use client";

import type { GeometryTask } from "@/mock/tasks";
import type { FigureState } from "@/lib/geometryTypes";
import {JSX} from "react";

interface GeometryViewerProps {
  task: GeometryTask;
  figure: FigureState | null;
}

const width = 600;
const height = 360;

export const GeometryViewer = ({ task, figure }: GeometryViewerProps) => {
  const renderInitialFigure = () => {
   const fig = task.initialFigure || {};
   const elements: JSX.Element[] = [];

   if (fig.points) {
    for (const p of fig.points) {
     elements.push(
```

```
   <g key={`init-point-${p.id}`}>
    <circle cx={p.x} cy={p.y} r={4} fill="#38bdf8" />
    <text x={p.x + 6} y={p.y - 6} fontSize={10} fill="#e5e7eb">
     {p.id}
    </text>
   </g>
  );
 }
}

if (fig.lines) {
 for (const l of fig.lines) {
  const from = fig.points.find((p: any) => p.id === l.from);
  const to = fig.points.find((p: any) => p.id === l.to);
  if (!from || !to) continue;
  elements.push(
   <line
    key={`init-line-${l.id}`}
    x1={from.x}
    y1={from.y}
    x2={to.x}
    y2={to.y}
    stroke="#64748b"
    strokeWidth={2}
   />
  );
 }
}

if (fig.circle) {
```

```
 const c = fig.circle;
 elements.push(
  <circle
   key={`init-circle-${c.id}`}
   cx={c.center.x}
   cy={c.center.y}
   r={c.radius}
   stroke="#64748b"
   strokeWidth={2}
   fill="none"
  />
 );
}

if (fig.line) {
 const l = fig.line;
 elements.push(
  <line
   key={`init-line-single-${l.id}`}
   x1={l.from.x}
   y1={l.from.y}
   x2={l.to.x}
   y2={l.to.y}
   stroke="#64748b"
   strokeWidth={2}
  />
 );
}

return elements;
```

```
};

const renderUserFigure = () => {
 if (!figure) return null;
 const { points, segments, circles } = figure;

 const segEls =
  segments?.map((s) => {
   const from = points.find((p) => p.id === s.fromId);
   const to = points.find((p) => p.id === s.toId);
   if (!from || !to) return null;
   return (
    <line
     key={s.id}
     x1={from.x}
     y1={from.y}
     x2={to.x}
     y2={to.y}
     stroke="#f97316"
     strokeWidth={2.5}
    />
   );
  }) ?? [];

 const circleEls =
  circles?.map((c) => {
   const center = points.find((p) => p.id === c.centerId);
   if (!center) return null;
   return (
    <circle
```

```
    key={c.id}
    cx={center.x}
    cy={center.y}
    r={c.radius}
    stroke="#f97316"
    strokeWidth={2}
    fill="none"
   />
  );
 }) ?? [];

 const pointEls =
  points?.map((p) => (
   <g key={p.id}>
    <circle cx={p.x} cy={p.y} r={4} fill="#f97316" />
    <text x={p.x + 6} y={p.y - 6} fontSize={10} fill="#e5e7eb">
     {p.id}
    </text>
   </g>
  )) ?? [];

 return (
  <>
   {circleEls}
   {segEls}
   {pointEls}
  </>
 );
};
```

```
  return (
   <div className="card shadow-md space-y-2">
    <h3 className="text-sm font-semibold">
     Перегляд побудови учня
    </h3>
    <div className="border border-slate-700 rounded-lg bg-slate-900/80 overflow-
hidden">
     <svg
      width={width}
      height={height}
      viewBox={`0 0 ${width} ${height}`}
      className="w-full h-auto"
     >
      <defs>
       <pattern
        id="smallGridView"
        width="20"
        height="20"
        patternUnits="userSpaceOnUse"
       >
        <path
         d="M 20 0 L 0 0 0 20"
         fill="none"
         stroke="#1f2937"
         strokeWidth="0.5"
        />
       </pattern>
       <pattern
        id="gridView"
        width="100"
```

```
        height="100"
        patternUnits="userSpaceOnUse"
       >
        <rect width="100" height="100" fill="url(#smallGridView)" />
        <path
         d="M 100 0 L 0 0 0 100"
         fill="none"
         stroke="#334155"
         strokeWidth="1"
        />
       </pattern>
      </defs>

      <rect width="100%" height="100%" fill="url(#gridView)" />
      {renderInitialFigure()}
      {renderUserFigure()}
     </svg>
    </div>
   </div>
  );
};
```