

Опис продукту (Func spec, UML, Identity Management)

Навчальний застосунок “Математика: повторення”

Застосунок призначений для теоретичного опрацювання шкільного курсу алгебри з 5 по 11 класи.

1. Вступ

1.1. Мета

Метою цього документа є визначення вимог до створення вебсервісу “Математика: повторення”, призначеного для повторення шкільного курсу математики (алгебра та геометрія) для учнів 5–11 класів. Сервіс надає інтерактивні можливості для теоретичного вивчення, проходження тестів, а також інтеграцію з Telegram-ботом для зручної взаємодії. “Математика: повторення” стане ефективним інструментом підготовки до державних іспитів, забезпечуючи персоналізований підхід до навчання та контролю знань.

1.2. Позиціонування продукту

“Математика: повторення” створений для задоволення актуальної потреби в сучасних освітніх інструментах, які допомагають учням якісно підготуватися до державних іспитів, таких як ЗНО чи НМТ. Застосунок забезпечує зручний доступ до теоретичного матеріалу з алгебри та геометрії, дозволяючи користувачам працювати у власному темпі. Особливістю сервісу є інтеграція з Telegram-ботом, що спрощує отримання домашніх завдань, нагадувань і аналітики для вчителів. Крім того, “Математика: повторення” пропонує швидкий доступ до архіву тестів ЗНО/НМТ, можливість зберігати нотатки та відстежувати активність учнів. Цей інструмент має зробити

процес навчання більш гнучким, структурованим і доступним як для учнів, так і для вчителів.

1.3. Цільова аудиторія

Основною аудиторією продукту є учні 5–11 класів, які прагнуть покращити свої знання з математики. Також вчителі, які шукають інструмент для організації навчального процесу, моніторингу активності та контролю знань учнів і батьки, які хочуть контролювати навчальний прогрес своїх дітей.

2. Опис продукту

2.1. Особливості продукту

“Математика: повторення” — це інтерактивний вебсервіс, створений для допомоги учням 5–11 класів у повторенні та поглибленні знань з математики. Продукт поєднує доступ до теоретичних матеріалів, тестів, персональної статистики та додаткового інструментарію, що робить процес навчання зручним і ефективним.

Основні можливості застосунку включають:

1. Вивчення математичних тем: сервіс надає доступ до широкого спектра тем, включаючи алгебру, геометрію, тригонометрію, диференціальне та інтегральне числення, ймовірність і статистику.
2. Теоретичний матеріал: для кожної теми доступні визначення, теореми, формули та приклади, що супроводжуються покроковими поясненнями.
3. Словник математичних термінів: інтегрований словник забезпечує швидкий доступ до визначень основних термінів.

4. Підготовка до тестувань: застосунок пропонує можливість проходження онлайн-тестів, зокрема у форматі ЗНО/НМТ. Сервіс також допомагає у регулярному закріпленні знань упродовж навчального року.
5. Персоналізована статистика: сервіс веде облік навчального часу, відстежуючи, скільки часу користувачі проводять у програмі. Це дозволяє вчителям отримувати детальні звіти про активність учнів, а самим учням — контролювати свій прогрес.
6. Нотатки: користувачі мають змогу зберігати власні записи, ідеї чи важливу інформацію прямо в застосунку. Всі нотатки зберігаються автоматично та впорядковуються за датою.
7. Доступність матеріалів: теоретичні матеріали з алгебри доступні офлайн, тоді як для доступу до геометричних матеріалів, що містять фото, та проходження тестів необхідне підключення до Інтернету.
8. Інтеграція з Telegram-ботом: бот сповіщає користувачів про нові завдання, нагадує про активність, пропущені завдання та нові теми і дозволяє швидко отримувати зворотний зв'язок.
9. Збір відгуків: продукт передбачає функціонал для збору відгуків через кнопку зворотного зв'язку, що дозволяє розробникам отримувати актуальну інформацію для покращення сервісу.
10. Ролі користувачів: підтримує дві основні ролі користувачів: учня і вчителя. Учні мають можливість проходити тести, переглядати навчальні матеріали, отримувати завдання через Telegram-бот і відстежувати власну активність. Вчителі можуть створювати класи,

додавати учнів, контролювати їх активність, призначати домашні завдання і отримувати статистичні звіти.

2.2. Функціональні вимоги

Незарєєстровані користувачі можуть:

1. Переглядати теоретичні матеріали за темами.
2. Отримувати зворотний зв'язок через кнопку у застосунку.
3. Мати доступ до матеріалів офлайн (для алгебри та геометрії).

Зарєєстровані користувачі можуть:

1. Проходити тести та отримувати результати.
2. Зберігати нотатки і переглядати їх.
3. Переглядати персональну статистику активності та прогресу.
4. Отримувати сповіщення через Telegram-бот.
5. Вчителі можуть створювати та управляти класами.
6. Вчителі можуть призначати домашні завдання учням.
7. Вчителі можуть створювати класи і додавати учнів
8. Вчителі можуть контролювати активність своїх учнів та отримувати звіти.

2.3. Нефункціональні вимоги

1. Продуктивність

- a) Час відгуку сервісу на запит користувача не повинен перевищувати 1 секунди для 95% запитів.
- b) Час завантаження головної сторінки не більше 3 секунд при стандартному навантаженні.
- c) Система повинна обробляти не менше 1000 одночасних активних користувачів без зниження продуктивності.

2. Безпека

- a) Всі дані користувачів повинні зберігатися у зашифрованому вигляді за допомогою алгоритму AES-256.
- b) Паролі зберігаються у вигляді хешів за допомогою алгоритму bcrypt.
- c) Всі зовнішні API-запити мають здійснюватися через HTTPS
- d) Ввести механізм багатофакторної автентифікації (MFA) для вчителів та адміністраторів.

3. Надійність

- a) Доступність сервісу повинна становити не менше 99.9% часу на місяць.
- b) Впровадити автоматичне резервне копіювання бази даних щодня, зберігаючи копії протягом 30 днів.

- c) Визначити процедуру відновлення після збоїв, що не перевищує 2 години.

4. Масштабованість

- a) Система повинна підтримувати горизонтальне масштабування для обробки збільшення кількості користувачів.
- b) Вебсервіс має витримувати збільшення навантаження до 2000 одночасних активних користувачів без критичних збоїв.

5. Кросплатформеність

- a) Сервіс має бути доступний на основних браузерях (Chrome, Safari, Edge).
- b) Забезпечити коректне відображення інтерфейсу на екранах з різною роздільною здатністю та розмірами дисплеїв

6. Юзабіліті

- a) Для нових користувачів повинна бути доступна інтерактивна інструкція користування.
- b) Проводити регулярне тестування інтерфейсу для покращення зручності використання не рідше ніж раз на півроку.

7. Документованість

- a) Всі публічні API повинні бути задокументовані у форматі OpenAPI.
- b) Керівництво користувача повинно бути доступне у вигляді онлайн-документації.

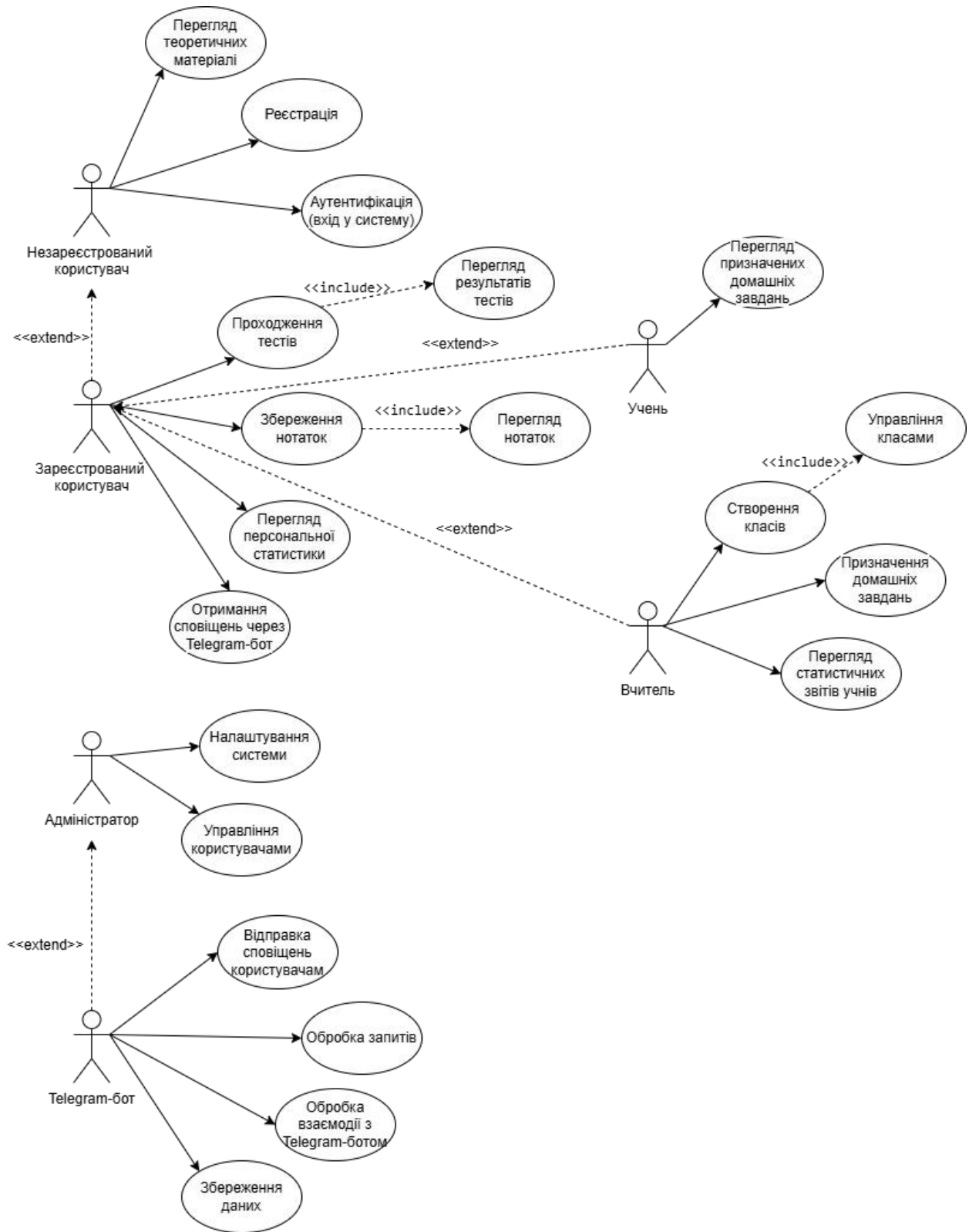
8. Відновлюваність:

- a) У випадку збоїв система повинна забезпечити відновлення останніх збережених даних протягом 15 хвилин після аварії.
- b) Впровадити систему сповіщень про збої для адміністраторів через електронну пошту або Telegram-бота.

9. Вимоги до тестування:

- a) Покриття тестами має становити не менше 80% для основного коду.
- b) Регресійне тестування повинно проводитися перед кожним основним релізом.

2.4. Діаграма прецедентів (Use-case diagram)



Назва	Функціональність	Опис
Перегляд теоретичних матеріалів	Надання доступу до навчальних матеріалів.	Користувачі можуть переглядати теоретичні дані з математики для підготовки до тестів.
Реєстрація	Створення нового облікового запису.	Користувач вводить персональні дані для реєстрації у системі.
Аутентифікація (вхід у систему)	Перевірка облікових даних.	Користувач вводить логін і пароль для доступу до персонального акаунту.
Проходження тестів	Виконання тестових завдань.	Користувачі проходять інтерактивні тести для перевірки знань.
Збереження нотаток	Додавання особистих нотаток	Користувач може зберігати свої записи під час навчання.
Перегляд нотаток	Доступ до збережених нотаток	Перегляд, редагування або видалення власних нотаток.
Перегляд результатів тестів	Аналіз успішності	Користувачі можуть переглядати результати виконаних тестів
Перегляд персональної статистики	Моніторинг успішності	Відображення персональних статистичних даних щодо пройдених тестів
Перегляд призначених домашніх завдань	Доступ до завдань від вчителя	Учні можуть переглядати завдання, які їм призначено
Управління класами	Організація навчальних груп	Вчитель може створювати, редагувати та видаляти класи
Створення класів	Додавання нових навчальних груп	Вчитель створює класи для групування учні
Призначення домашніх завдань	Призначення завдань для виконання	Викладачі можуть призначати домашні завдання конкретним класам

		або учням
Перегляд статистичних звітів учнів	Аналіз успішності учнів	Викладачі можуть переглядати детальні статистичні дані про успішність учнів
Отримання сповіщень через Telegram-бот	Отримання повідомлень	Учні отримують повідомлення про завдання чи результати через Telegram-бот.
Відправка сповіщень користувачам	Надсилення важливої інформації	Адміністратор може надсилати сповіщення всім користувачам
Обробка запитів	Опрацювання запитів користувачів	Система обробляє запити, отримані через вебінтерфейс чи Telegram-бот
Обробка взаємодії з Telegram-ботом	Інтеграція з ботом	Система забезпечує обробку даних і відповідей, отриманих через Telegram-бот
Збереження даних	Постійне збереження інформації	Система забезпечує надійне збереження інформації у базі даних
Налаштування системи	Конфігурація платформи	Адміністратор може змінювати налаштування системи відповідно до потреб
Управління користувачами	Додання та видалення користувачів	Адміністратор може керувати обліковими записами користувачів

2.5. User Stories

1. Як учень, я хочу переглядати теоретичні матеріали, щоб підготуватися до тестів.
2. Як учень, я хочу пройти реєстрацію в системі, щоб отримати доступ до функціональності платформи.
3. Як учень, я хочу мати можливість аутентифікуватися, щоб захистити свої дані та результати.
4. Як учень, я хочу проходити тести, щоб перевірити свої знання з математики.
5. Як учень, я хочу зберігати нотатки, щоб мати швидкий доступ до важливої інформації.
6. Як учень, я хочу переглядати свої нотатки, щоб повторювати матеріал.
7. Як учень, я хочу переглядати результати тестів, щоб оцінити свою успішність.
8. Як учень, я хочу бачити свою персональну статистику, щоб відстежувати прогрес.
9. Як учень, я хочу отримувати сповіщення через Telegram-бот, щоб не пропускати важливі новини.
10. Як учень, я хочу переглядати призначені домашні завдання, щоб знати, що потрібно виконати.
11. Як викладач, я хочу створювати класи, щоб організувати учнів у навчальні групи.

12. Як викладач, я хочу призначати домашні завдання, щоб перевірити знання учнів.

13. Як викладач, я хочу переглядати статистичні звіти учнів, щоб оцінювати їхній прогрес.

14. Як викладач, я хочу надсилати сповіщення учням, щоб інформувати їх про важливі події.

15. Як адміністратор, я хочу налаштовувати систему, щоб забезпечити її ефективне функціонування.

16. Як адміністратор, я хочу управляти користувачами, щоб додавати або видаляти облікові записи.

17. Як адміністратор, я хочу обробляти запити користувачів, щоб вирішувати їхні проблеми.

18. Як адміністратор, я хочу забезпечити обробку взаємодії з Telegram-ботом, щоб підтримувати комунікацію з користувачами.

19. Як адміністратор, я хочу зберігати дані, щоб забезпечити їх надійне збереження.

2.6. Технологічний стек

“Математика: повторення” реалізований на основі мікросервісної архітектури з використанням сучасних технологій.

Аутентифікація: JWT-токен з обмеженим часом життя для періодичного перелогінювання користувачів. У майбутньому планується додати підтримку двох токенів для покращення безпеки.

Архітектура: Мікросервісний підхід.

Спілкування: REST API (клієнт-сервер).

База даних: PostgreSQL.

Back-end: .NET Core 8.

Front-end: React + TypeScript.

2.7. Зацікавлені сторони

1. Учні: отримання доступу до теоретичних матеріалів, проходження тестів, перегляд результатів і статистики, отримання сповіщень через Telegram-бот.
2. Вчителі: створення класів, призначення домашніх завдань, перегляд статистичних звітів учнів, контроль за успішністю.
3. Адміністратор системи: управління користувачами, налаштування системи, забезпечення стабільної роботи сервісу, відправка сповіщень.
4. Розробники: розробка, тестування та підтримка функціональності системи.

2.8. Обмеження та ризики

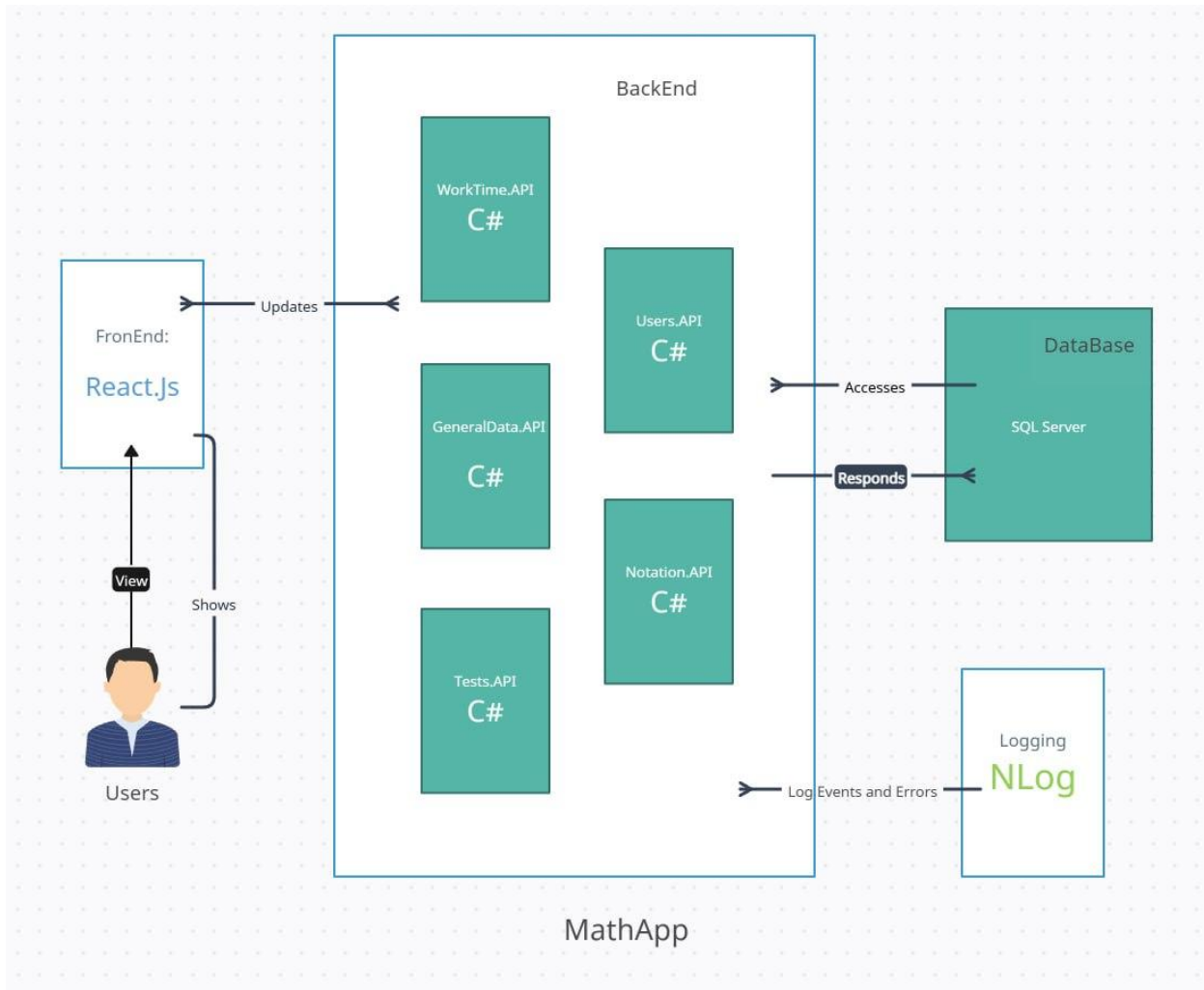
Обмеження: застосунок використовує мікросервісну архітектуру зі статичним JWT токеном, який має обмежений час життя. Це означає, що користувачі мають періодично перелогінюватися для підтримки активної сесії. Система наразі не підтримує механізм із двома токенами (access і refresh), що обмежує рівень безпеки, але це планується реалізувати в майбутньому. Взаємодія клієнта з сервером відбувається через REST API, що обмежує систему у випадках, коли потрібна асинхронна або подієва комунікація.

Ризики: основним ризиком є можливість втрати даних або збоїв у роботі бази даних PostgreSQL у разі високого навантаження або неправильної конфігурації. Також існує ризик безпеки, пов'язаний із використанням лише одного токена для аутентифікації, що може призвести до потенційних атак, якщо токен буде скомпрометовано. Крім того, відсутність системи резервного копіювання на початкових етапах може створити загрозу втрати важливої інформації.

Архітектура: microservice/SPA/serverless/cloud/etc

Архітектура

Як архітектуру для проєкту MathTouch обрано мікросервісну архітектуру, оскільки застосунок містить декілька незалежних сервісів, кожен з яких відповідає за окрему функціональність. Це дозволяє легко масштабувати систему, розгортати сервіси окремо та спрощує підтримку й оновлення застосунку.



Компоненти:

Frontend (React.js)

- Відображає інтерфейс користувача
- Отримує дані з бекенду та оновлює свій стан
- Передає оновлення користувача до відповідних API

Backend (C# .NET, набір мікросервісів)

- WorkTime API – керує обліком робочого часу користувачів
- Users API – відповідає за реєстрацію, автентифікацію та управління користувачами
- GeneralData API – обробляє загальні дані застосунку
- Notation API – відповідає за роботу з записами користувачів
- Tests API – керує тестами та їх перевіркою
- Усі сервіси працюють незалежно та взаємодіють через HTTP-запити

Database (SQL Server)

- Зберігає всі дані користувачів, результати тестів, записи робочого часу та інші важливі дані
- База даних напряму взаємодіє тільки з відповідними API

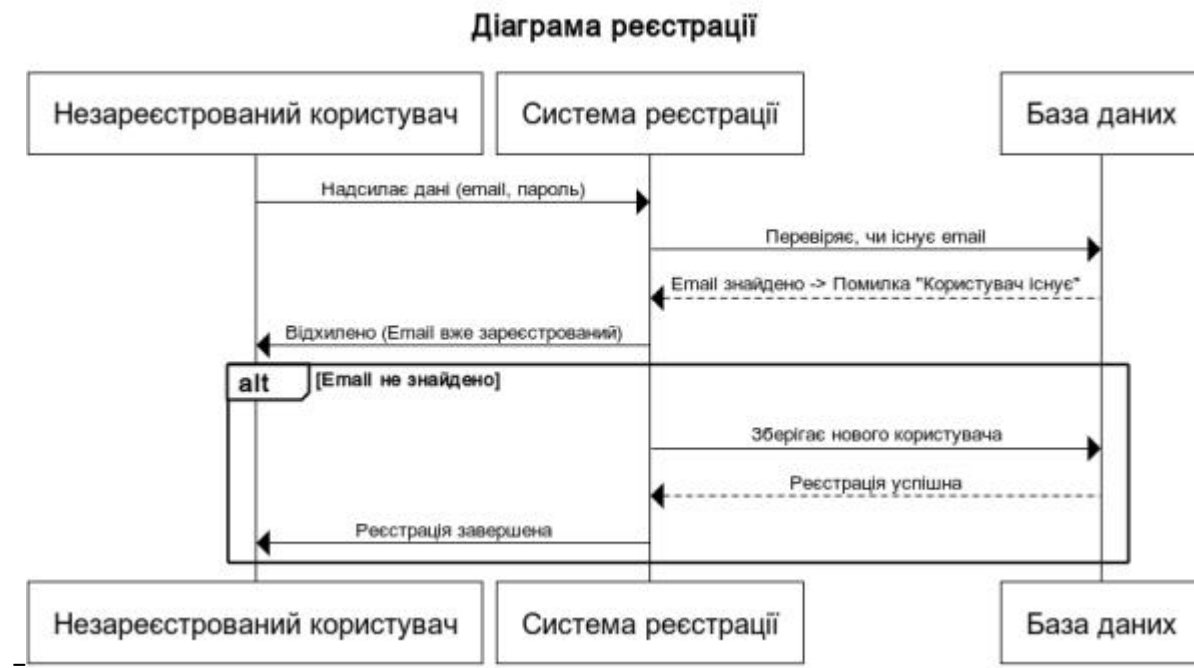
Logging (NLog)

- Відповідає за логування подій та помилок у системі
- Дозволяє відстежувати виконання сервісів та потенційні проблеми

Діаграми послідовностей

1. Діаграма реєстрації

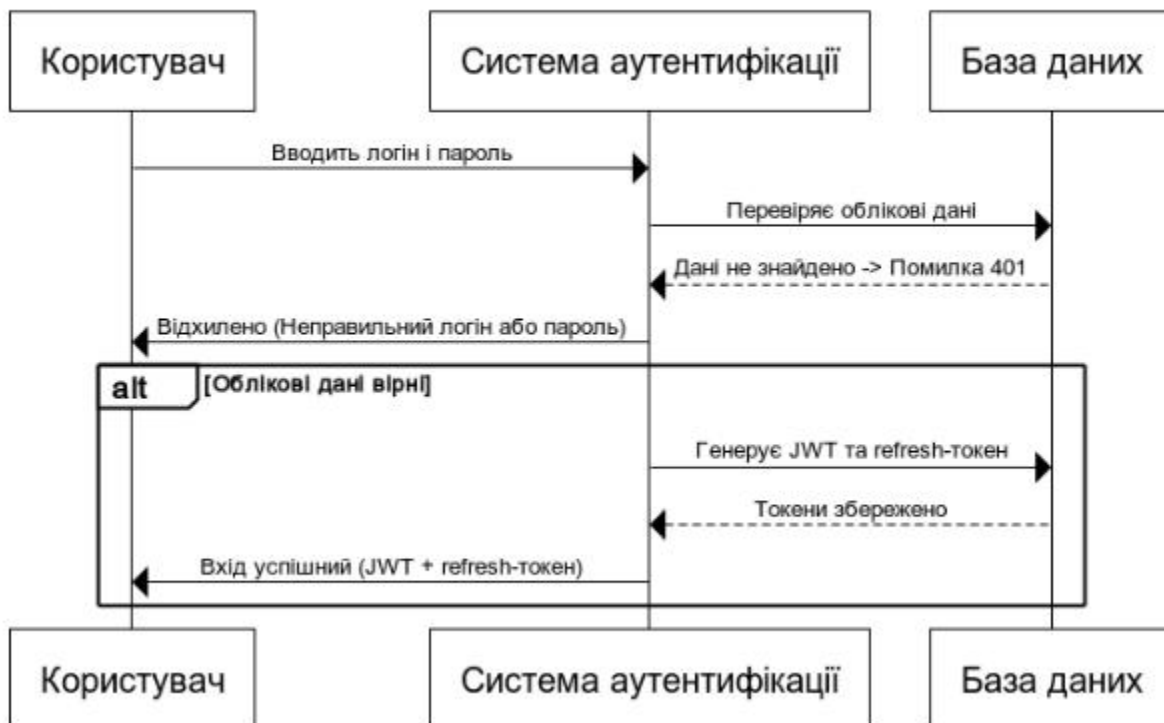
Ця діаграма демонструє процес реєстрації нового користувача. Спочатку система перевіряє, чи email вже існує в базі. Якщо він знайдений, користувач отримує помилку. Якщо email новий, дані зберігаються в базі, і реєстрація успішно завершується.



2. Аутентифікація (вхід у систему)

Ця діаграма описує процес входу користувача. Користувач вводить логін та пароль. Якщо дані невірні, система повертає помилку 401. Якщо все правильно, система створює JWT та refresh-токен, які будуть використовуватися для автентифікації.

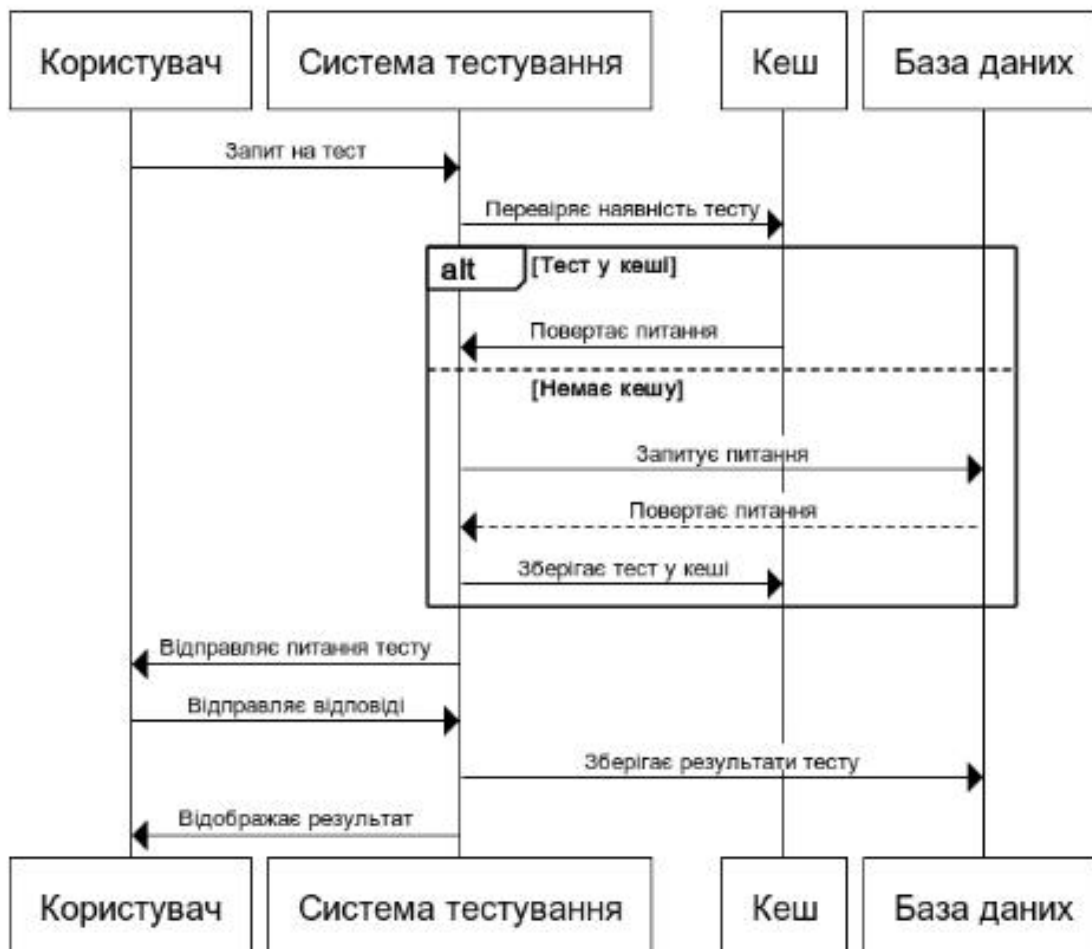
Діаграма входу в систему



3. Проходження тесту

Користувач запитує тест. Система спочатку перевіряє, чи є питання в кеші, щоб зменшити навантаження на базу. Якщо кешу немає, питання беруться з бази, а потім записуються в кеш. Відповіді користувача зберігаються у базі.

Діаграма проходження тесту



4. Отримання статистики

Вчитель запитує статистику успішності класу. Система надсилає запит у базу, отримує агреговані дані та повертає їх вчителю у вигляді звіту.

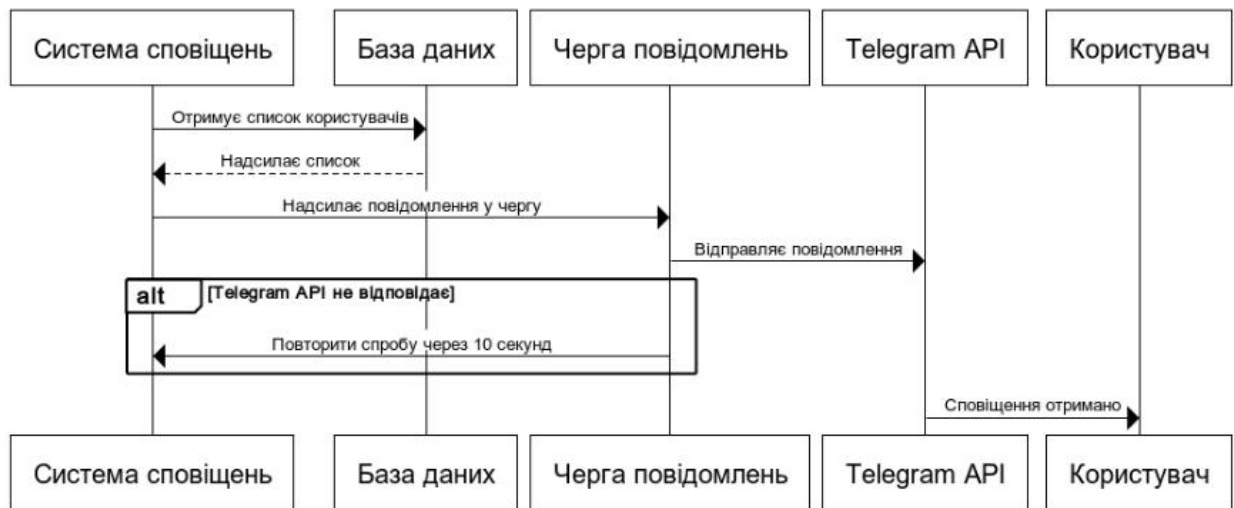
Діаграма отримання статистики



5. Надсилання сповіщень через Telegram-бот

Система отримує список користувачів, які повинні отримати повідомлення. Щоб уникнути перевантаження, повідомлення додаються в чергу. Потім система відправляє їх через Telegram API. Якщо Telegram API не відповідає, відправка повторюється через 10 секунд.

Діаграма надсилання сповіщень

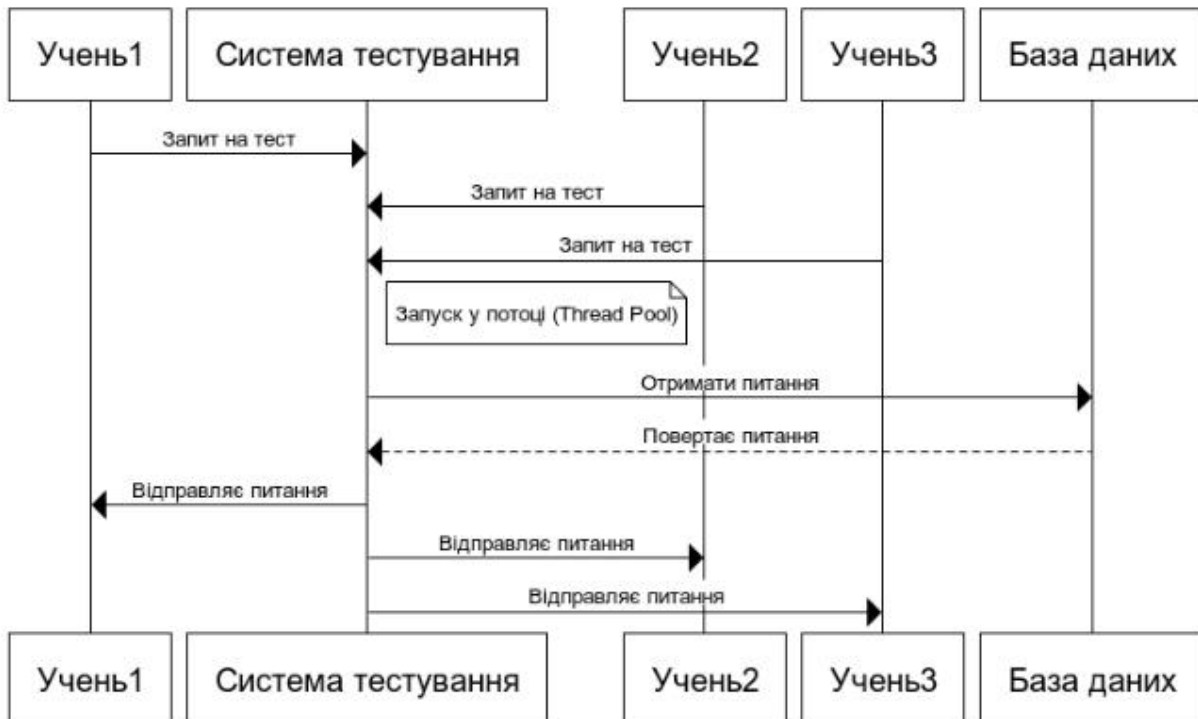


Concurrency Patterns usage

1. Обробка кількох запитів одночасно (Thread Pool)

Учні одночасно запускають тест. Сервер обробляє їхні запити через пул потоків, що дозволяє уникнути перевантаження.

Thread Pool: Одночасне проходження тесту



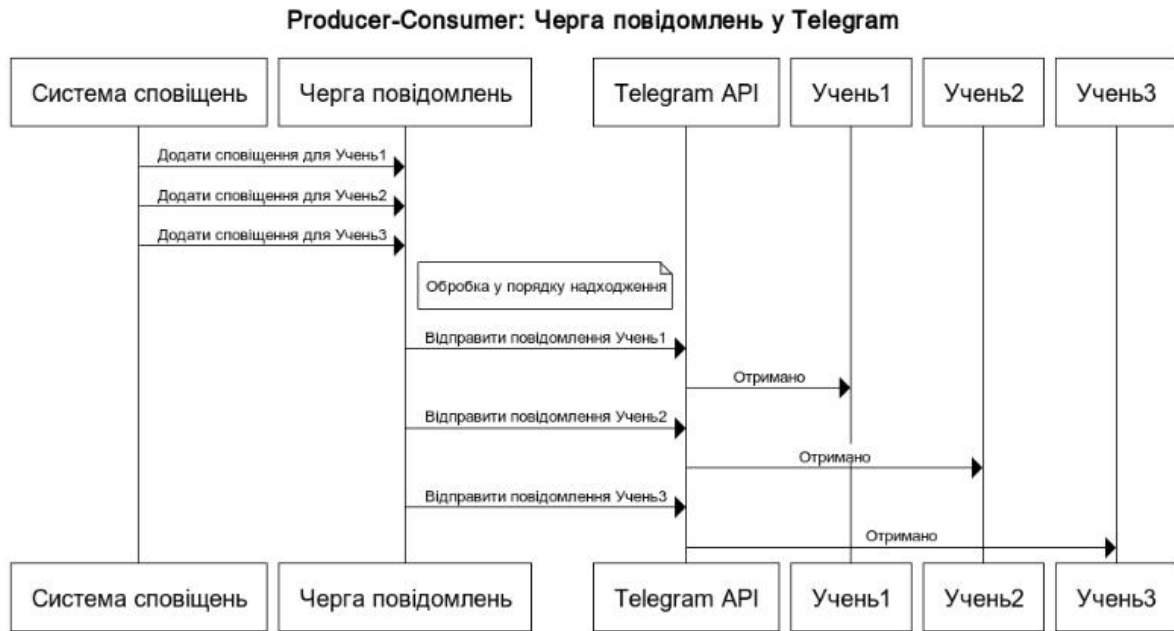
2. Асинхронне оновлення статистики (Future/Promise)

Учень завершує тест, а оновлення статистики запускається у фоновому режимі. Це дозволяє швидко повертати результат тесту, не чекаючи обробки статистики



3. Черга повідомлень у Telegram-боті (Producer-Consumer)

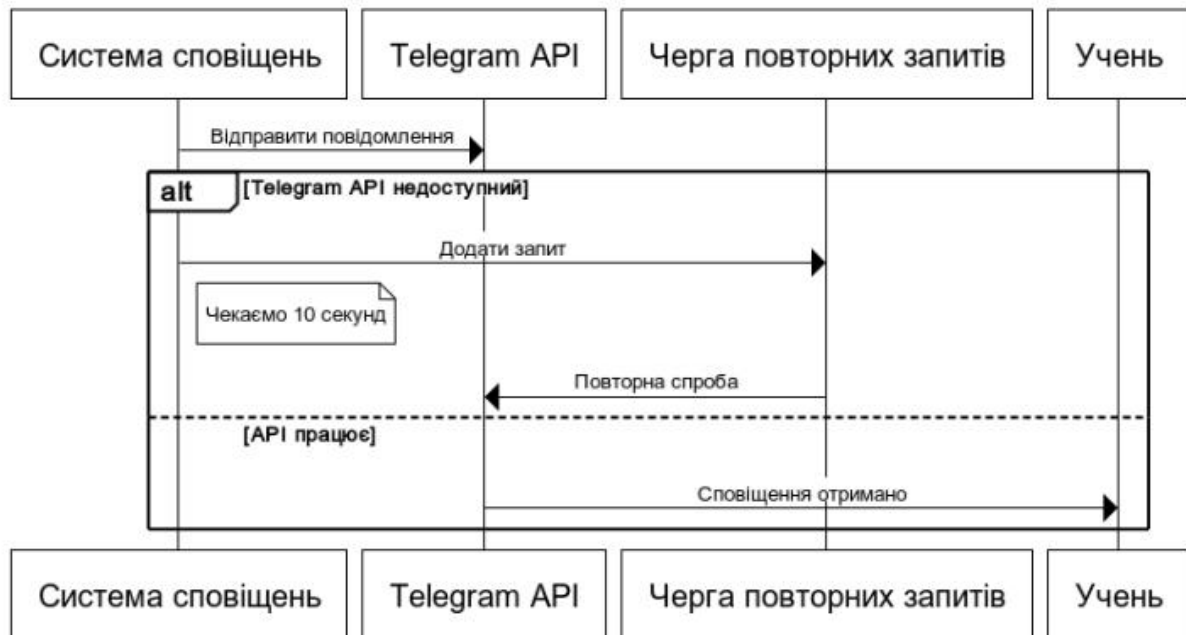
Щоб не перевантажувати Telegram API, повідомлення додаються у чергу і відправляються поступово.



4. Захист від перевантаження Telegram API (Circuit Breaker)

Якщо Telegram API не працює, система перериває запити і повторює спробу через певний час, щоб уникнути перевантаження.

Circuit Breaker: Захист від перевантаження Telegram API



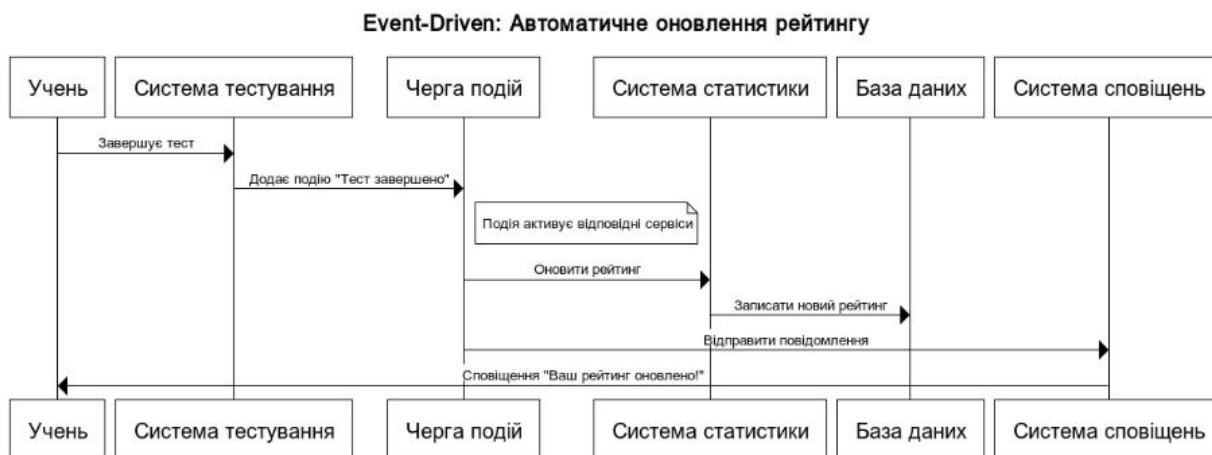
5. Асинхронна обробка домашніх завдань (Asynchronous Messaging)

Коли вчитель призначає завдання, воно відправляється учням асинхронно, щоб не блокувати інші операції.



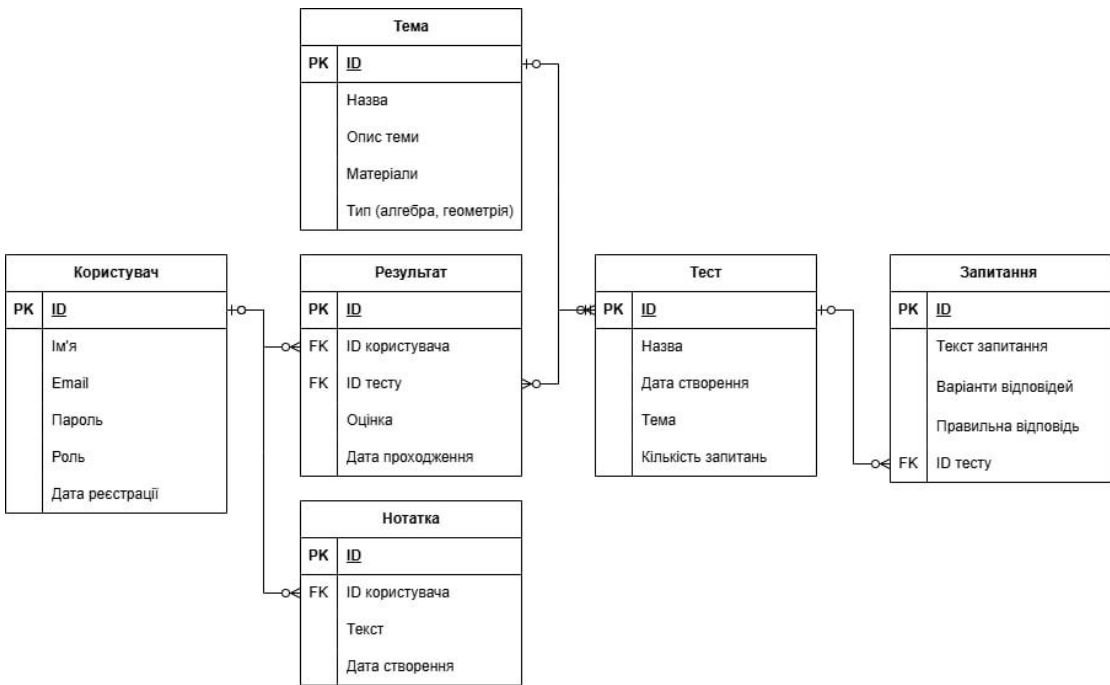
6. Автоматичне оновлення рейтингу (Event-Driven Architecture)

Коли учень завершує тест, система автоматично оновлює його рейтинг та надсилає сповіщення.



Data model

ER-діаграма



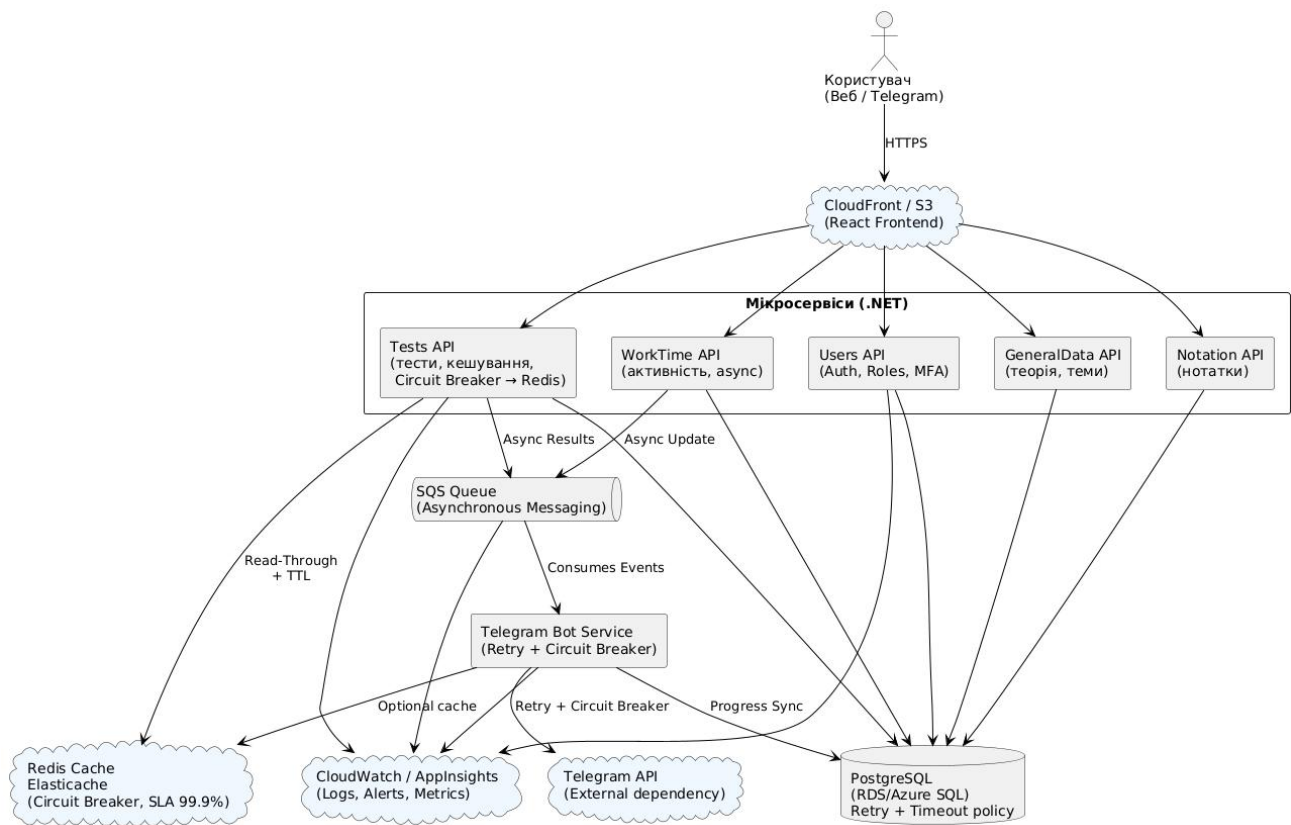
Класифікація даних по Data Retention Policy

Категорія даних	Політика збереження
Облікові дані (ім'я, email, роль)	Зберігаються доки акаунт активний. Видаляються через 6 місяців після видалення акаунту.
Результати тестів	Зберігаються 2 роки і потім архівуються.
Логи подій	Зберігаються 1 рік, після чого видаляються.
Запитання та тести	Зберігаються постійно, доки адміністратор не видалить.
Прогрес користувача	Зберігається 2 роки після останньої активності, потім видаляється.

Telegram ID користувачів	Видаляється одразу після видалення акаунту.
--------------------------	---

Resiliency model

CID-діаграма (Component Interaction Diagram)



Компонент А	Компонент В	Інтерація
Користувач (Учень)	Frontend (React)	Отримання теорії, тестів, нотаток
Frontend	Users API	Логін, реєстрація, отримання JWT
Tests API	Redis Cache	Отримання питань тестів з кешу
WorkTime API	Queue (SQS)	Відправка статистики до черги
Telegram Bot Service	Telegram API	Надсилання повідомлень про завдання/успіхи

RMA Workbook (Resiliency, Mitigation, Availability)

Інтерація 1: Користувач → Frontend

Тип помилки	Пояснення	Mitigation
React SPA не завантажується	Користувач бачить пусту сторінку через збій S3/CDN	Кешування на стороні браузера, fallback сторінка
Втрата з'єднання	Клієнт не має інтернету або поганий зв'язок	Offline support для теоретичних матеріалів
API-запит зависає	Frontend чекає відповідь, але бекенд не відповідає	Loading state + timeout + error feedback

Інтерація 2: Frontend → Users API

Тип помилки	Пояснення	Mitigation
Сервер недоступний	Users API зупинений або відсутній через деплой	Health checks + Auto-restart + LB
JWT не створено	Проблема в токен генерації / криптографії	Перевірка токенів + fallback login flow
Rate-limit перевищено	Користувач робить занадто багато запитів (брутфорс)	IP Throttling, CAPTCHA, Rate Limiting

Інтерація 3: Tests API → Redis Cache

Тип помилки	Пояснення	Mitigation
Redis недоступний	Сервер Redis упав або мережевий збій	Fallback → PostgreSQL, TTL backfill
Застарілий кеш	Питання з кешу неактуальні	Cache Invalidation Policy
Перевищення об'єму	Redis переповнений, LRU очищує корисні дані	Обмеження розміру кешу, пріоритезація тем

Інтерація 4: WorkTime API → SQS Queue

Тип помилки	Пояснення	Mitigation
Queue переповнена	Велика кількість подій — черга затримується	Dead Letter Queue, Auto-scaling consumers
Затримка підтвердження	Event залишається в черзі, бо worker не підтверджує	Visibility Timeout + Retry Policy
Queue не працює	SQS недоступна (мережа/авторизація)	Fallback → локальний буфер, backoff policy

Інтерація 5: Telegram Bot → Telegram API

Тип помилки	Пояснення	Mitigation
Telegram API недоступний	Telegram має збій або API перевантажене	Circuit Breaker + Retry + Backoff
Невірний токен бота	Бот має старий/відкликаний	Моніторинг авторизації,

	ТОКЕН	автооновлення токена
Втрата повідомлень	Повідомлення не дійшло до Telegram через помилку	Queue Retry + Logging невдалих спроб

Security model

Основні флови

Флов 1: Учень проходить тест через веб-інтерфейс

Кроки:

1. Учень заходить на сайт.
2. Логується - отримує JWT-токен.
3. Обирає тест → отримує питання.
4. Відповідає → надсилає відповіді.
5. Отримує оцінку.

Флов 2: Адміністратор створює новий тест

Кроки:

1. Адмін заходить на сайт.
2. Логується - отримує токен з правами “admin”.
3. Вводить питання і варіанти відповідей.
4. Система зберігає цей тест у базу.

Threat Modeling (за STRIDE)

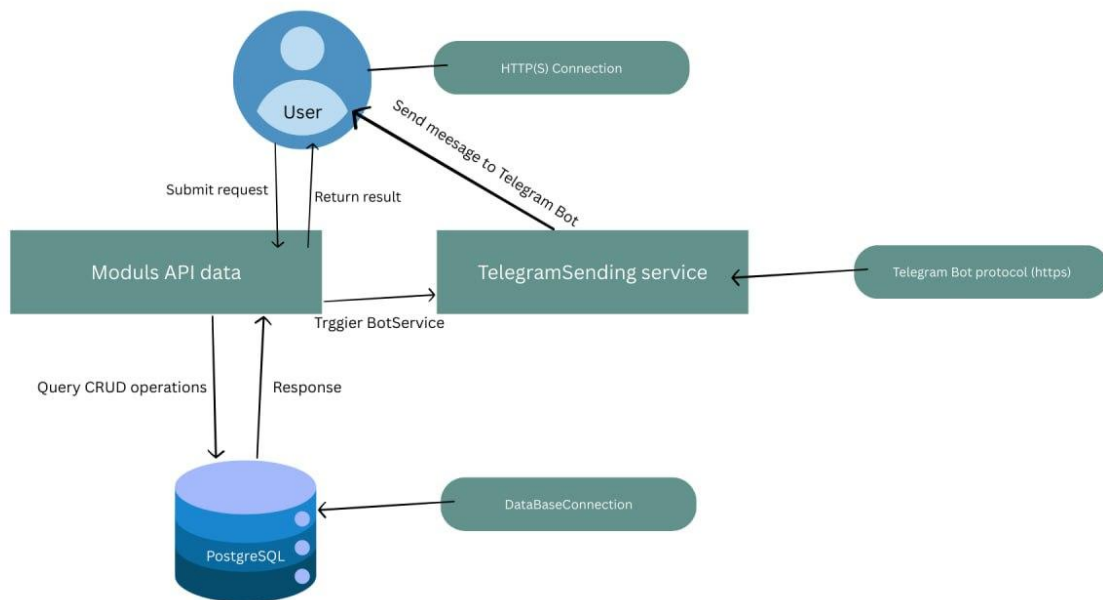
Тип	Флов	Загроза	Пояснення
S	Обидва	Викрадення або підробка JWT	Несанкціонований доступ до функцій
T	Флов 1	Підміна запиту (спроба надіслати відповіді іншого	Сфальсифіковані

		користувача)	результати
R	Обидва	Відсутність логування ключових дій	Неможливо відслідкувати зміни
I	Флов 1	Витік питань тесту через погано захищене API	Зловживання системою
D	Обидва	Перевантаження одного з сервісів (DoS)	Сервіс недоступний
E	Флов 2	Підміна ролі (Elevation of Privilege)	Звичайний користувач діє як адміністратор
T	Флов 2	Підміна структури тесту	Некоректне збереження, злам системи
S	Флов 1	Немає обмеження сесій або rate limiting	Атаки brute-force
I	Обидва	Надмірне логування → витік токенів чи даних	Витік інформації через лог-файли
T	Флов 2	SQL-ін'єкції	Потенційно відкриває доступ до всієї інформації або її модифікації

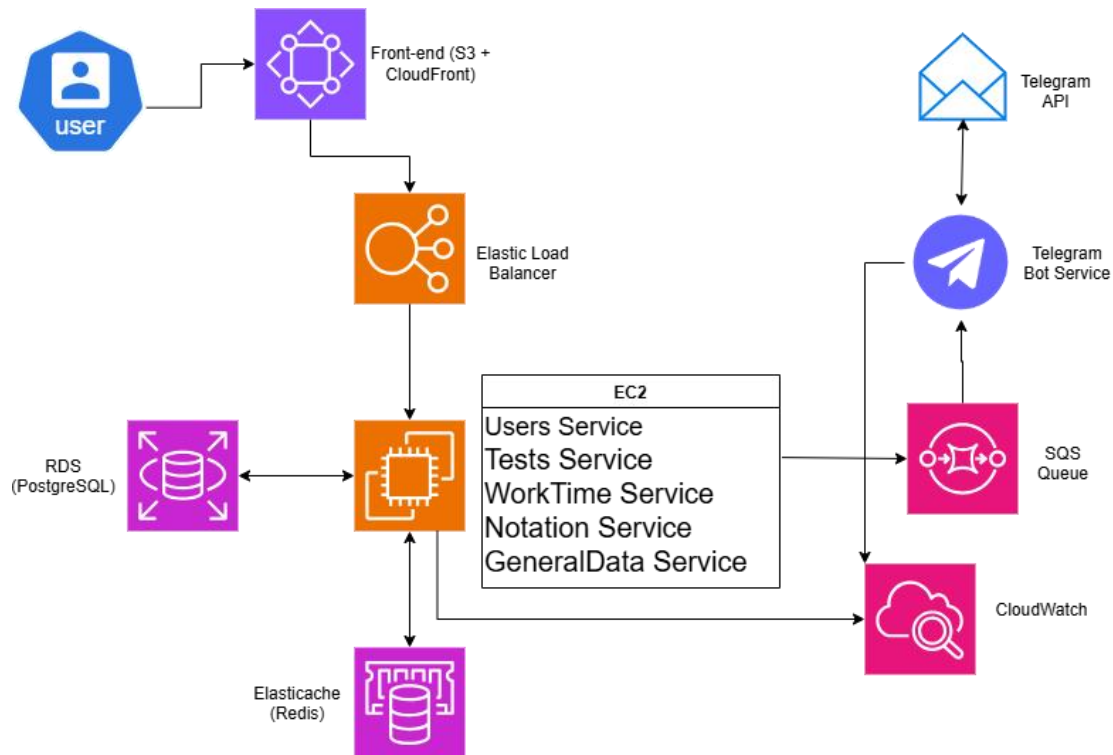
Mitigation Plan (на 10 загроз)

Загроза	Захист
Викрадення або підробка JWT	Використання HTTPS; обмеження

	часу дії токена (наприклад, 15 хв)
Підміна запиту (спроба надіслати відповіді іншого користувача)	Валідація, що ID користувача з JWT = ID, який надсилає відповіді
Відсутність логування ключових дій	Впровадити audit-лог з записом ID користувача, IP та часу
Витік питань тесту через погано захищене API	Дозвіл бачити питання тільки тим, хто проходить тест
Перевантаження одного з сервісів (DoS)	Встановити таймаути, ліміти запитів (наприклад, express-rate-limit)
Підміна ролі (Elevation of Privilege)	Ролі в JWT, перевірка на бекенді для кожної дії
Підміна структури тесту	Валідація JSON payload на бекенді
Немає обмеження сесій або rate limiting	Rate limiting + CAPTCHA або затримка після кількох спроб
Надмірне логування → витік токенів чи даних	Не зберігати токени, паролі, відповіді в логах
SQL-ін'єкції	Використовувати ORM або хоча б параметризовані запити (не через конкатенацію!)



Deployment model



Компоненти:

- ✓ Користувач (User): уособлює учня, вчителя або адміністратора, які взаємодіють із системою через вебзастосунок або Telegram-бот.
- ✓ Front-end (Хостинг статичного сайту): S3 Bucket із підключеним CloudFront для розміщення вебінтерфейсу React-додатку. Забезпечує швидку доставку контенту користувачам через мережу доставки вмісту (CDN).
- ✓ Elastic Load Balancer (Балансувальник навантаження): розподіляє вхідні запити між кількома інстансами серверів бекенду для забезпечення масштабованості та високої доступності системи.
- ✓ Back-end (Мікросервіси на EC2): група мікросервісів (Users Service, Tests Service, Notation Service, GeneralData Service, WorkTime Service)

розгорнутих на віртуальних машинах EC2 з автоматичним масштабуванням. Відповідають за бізнес-логіку, обробку тестів, облік статистики та управління користувачами.

- ✓ PostgreSQL (Керована реляційна база даних): Amazon RDS на базі PostgreSQL для зберігання структурованих даних: інформації про користувачів, результати тестів, нотатки та активність.
- ✓ Elasticache (Redis) (Кеш-сервер): використовується для кешування популярних запитань тестів, зменшення навантаження на базу даних та прискорення відповіді системи.
- ✓ SQS Queue (Черга повідомлень): сервіс чергування повідомлень для асинхронної обробки задач, таких як надсилання сповіщень через Telegram-бот або оновлення статистики після проходження тестів.
- ✓ Telegram Bot Service (Окремий сервіс бота): мікросервіс, що обробляє повідомлення від користувачів у Telegram, надсилає сповіщення, отримує статуси виконання завдань.
- ✓ CloudWatch (Моніторинг і логування): система моніторингу продуктивності, збору логів та відправки сповіщень про помилки або збої в роботі сервісів.

Робочий процес (Workflow):

- ✓ Взаємодія користувача: кінцевий користувач надсилає запити через браузер або Telegram-клієнт до застосунку, розміщеного у Front-end на S3.

- ✓ Обробка на бекенді: запити потрапляють через Elastic Load Balancer до відповідних мікросервісів на EC2.
- ✓ Інтеграція сервісів мікросервісами:
 - звернення до PostgreSQL для зберігання або отримання даних (результати тестів, інформація про користувачів);
 - звернення до ElastiCache Redis для даних кеш (питання тестів, короткочасні результати);
 - публікація подій у SQS Queue для асинхронної обробки (оновлення статистики, відправка сповіщень через Telegram-бот);
 - спілкування із Telegram Bot Service для обробки повідомлень користувачів та надсилання завдань;
 - логування важливих подій та помилок у CloudWatch для аналізу та моніторингу системи.

Analytics model

Таблиця аналітичної моделі

Метрика	Вимірювання / Формат	Пов'язана функціональність	Призначення (Insight)
Кількість активних користувачів щодня	int (DAU)	Вхід/використання сайту	Загальний рівень залучення
Середній час сесії	хвилини (float)	Час роботи з теорією/тестами/ нотатками	Якість залучення
Кількість пройдених тестів	int	Проходження тестів	Оцінка використання тестового функціоналу
Середній бал за тести	float (0–100%)	Результати тестів	Середній рівень знань
Частка учнів, які завершують тест	% (completion rate)	Проходження тестів	Показник утримання під час проходження
Кількість створених нотаток	int	Збереження нотаток	Активність у глибшому вивченні матеріалу
Кількість переглянутих теоретичних сторінок	int	Теоретичний матеріал	Якість підготовки до тесту
Відсоток користувачів, які відкривали Telegram-бот	%	Telegram-інтеграція	Охоплення каналів комунікації
Кількість отриманих push/бот-сповіщень	int	Telegram-бот	Ефективність комунікації
Відсоток виконаних домашніх завдань	%	Призначення та перегляд домашніх завдань	Відповідальність учнів

Частка повторного повернення наступного дня	%	Вхід/використання	Рівень утримання
Кількість створених класів	int	Управління класами (вчитель)	Залучення вчителів
Відсоток учнів з прогресом >50%	%	Прогрес користувача	Ефективність навчання
Частка користувачів, які залишили відгук	%	Форма зворотного зв'язку	Якість взаємодії з продуктом
Середня затримка відповіді API	мс (milliseconds)	Усі дії з бекендом	Продуктивність системи

Funnel 1: Проходження тесту

Мета: відстежити, як користувачі доходять до завершення тесту.

Кроки:

1. Увійшов у систему
2. Перейшов у розділ “Тести”
3. Вибрав тему та відкрив тест
4. Завершив тест
5. Переглянув результат

Використані метрики:

- 1: DAU (щоденна активність користувачів)
- 3: кількість пройдених тестів
- 5: completion rate (відсоток завершених тестів)
- 4: середній бал

13: прогрес користувача

Funnel 2: Залучення через Telegram-бот

Мета: оцінити ефективність Telegram-комунікації.

Кроки:

1. Отримав бот-сповіщення
2. Перейшов за посиланням із повідомлення
3. Увійшов у застосунок
4. Пройшов тест або переглянув нову тему

Використані метрики:

8: відсоток користувачів, які відкривали Telegram-бот

9: кількість бот-сповіщень

1: DAU (вхід у застосунок)

3: кількість пройдених тестів

7: перегляд теоретичних сторінок

Funnel 3: Залучення вчителя до створення класу і призначення завдань

Мета: проаналізувати, наскільки легко вчителі переходять від реєстрації до активного використання функцій навчання.

Кроки:

1. Зареєструвався як вчитель
2. Увійшов у профіль

3. Створив перший клас
4. Додав до класу хоча б одного учня
5. Призначив домашнє завдання
6. Переглянув звіт про виконання завдань

Використані метрики:

- 1: DAU (вхід)
- 12: кількість створених класів
- 10: відсоток виконаних завдань
- 13: прогрес учнів
- 14: відгуки

Monitoring&alerting model

Monitoring: таблиця операційних метрик

Метрика	Вимірювання	Пов'язаний ресурс	Як збирається	Призначення
CPU usage	%	EC2 (Back-end servic)	AWS CloudWatch	Навантаження на сервер
Memory usage	%	EC2	CloudWatch Agent	Споживання пам'яті
Response time (avg)	мс	API Gateway / EC2	CloudWatch + custom logs	Продуктивність API
Error rate	%	Back-end, Tests API	NLog + CloudWatch Logs	Стабільність роботи
Request count	int	API Gateway / Load Balancer	CloudWatch	Навантаження користувачів
Redis hit/miss ratio	float	ElastiCache (Redis)	Redis Monitoring	Ефективність кешу
DB query time	мс	PostgreSQL (Amazon RDS)	Performance Insights / pg_stat_statements	Повільні запити
Queue length	int	SQS (черги)	AWS SQS metrics	Затримки в обробці подій
Telegram bot success rate	%	Telegram Bot Service	Custom metrics + logs	Доставлення сповіщень
Uptime сервісів	%	EC2, Load Balancer	CloudWatch status checks	Доступність сервісу
Кількість логінів на годину	int	Users API	NLog + CloudWatch	Піки активності
Queue	сек	Worker + SQS	CloudWatch +	Затримки в

processing delay			логи	обробці
Вільне місце в БД	GB	PostgreSQL	RDS storage monitoring	Запас пам'яті
Помилки автентифікації	int	Users API	NLog	Аномалії входу
Telegram API response time	мс	Telegram Bot Service	Custom monitoring	Якість зовнішнього API

Alerting: критичні пороги і сповіщення

Метрика	Min / Max значення	Тип події	Критичність	План дій (Mitigation)
CPU usage	> 85% (10 хв)	Threshold breach	Висока	Auto-scale EC2, перевірити код
Response time (avg)	> 800 мс	Latency alert	Висока	Перевірити Redis, DB, увімкнути кеш
Error rate	> 5%	Error spike	Критична	Перевірити логи NLog, викотити rollback
Queue length (SQS)	> 1000	Queue threshold	Висока	Додати воркери, перевірити затримки
Redis miss ratio	> 0.5	Efficiency drop	Середня	Оновити кеш, змінити політику TTL
Uptime сервісів	< 99% за добу	Availability	Критична	Перевірити EC2, балансер, DNS
DB query time	> 1500 мс	Performance	Висока	Індексувати запити, оптимізувати SQL
Telegram bot success rate	< 90%	API failure	Середня	Перевірити токен, Telegram API
Помилки автентифікації	> 50 на годину	Security alert	Висока	Заблокувати IP, ввести CAPTCHA

Storage left in DB	< 5 GB	Capacity limit	Середня	Розширити storage, очистити старі дані
--------------------	--------	----------------	---------	--

Збір метрик:

- AWS CloudWatch: CPU, memory, disk, API latency
- NLog + CloudWatch Logs: події, помилки, логіни
- Redis: custom Redis CLI stats
- PostgreSQL: pg_stat_statements, RDS Insights
- Telegram Bot: custom success/error counter
- Метрики з SQS: Queue length, delay