Специфікація програмного забезпечення

Генератор тестових завдань на базі ШІ

Software system requirement specification

1.0

28.04.2025

Юкленчук Роман Олексійович  
Єременко Андрій Віталійович

Жуйков Михайло Олександрович

Береговий Даніїл Олександрович

**ІСТОРІЯ ЗМІН**

|  |  |  |  |
| --- | --- | --- | --- |
| Дата | Опис | Автор | Коментарі |
| 26.04.2025 | Створено пункти 1.1-1.5 | Юкленчук Роман Олексійович |  |
| 27.04.2025 | Створено пункти 2.1-2.5 | Єременко Андрій Віталійович |  |
| 28.04.2025 | Створено пункти 3.1-3.2 | Жуйков Михайло Олександрович |  |
| 29.04.2025 | Створено пункти 3.3-3.5 | Береговій Даніїл Олександрович |  |

**ЗАТВЕРДЖЕННЯ ДОКУМЕНТУ**

Наступну специфікацію вимог до програмного забезпечення було прийнято та схвалено:

|  |  |  |  |
| --- | --- | --- | --- |
| Підпис | Друковане ім’я | Назва | Дата |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**1 ВСТУП**

1.1 Огляд продукту

Даний програмний продукт є вебплатформою для автоматизованої генерації тестових завдань на основі завантаженого тексту за допомогою засобів штучного інтелекту. Система призначена для викладачів, учнів, студентів та інших користувачів, які потребують швидкого створення якісних тестів на основі навчального або будь-якого іншого текстового матеріалу.

Користувач платформи завантажує текст (наприклад, лекцію, статтю або підручник), після чого система обробляє цей вміст та за допомогою моделей штучного інтелекту формує набір питань з варіантами відповідей. Користувач може переглядати, редагувати, зберігати та проходити згенеровані тести. Також підтримується персоналізація тестування — повторні запитання базуються на попередніх відповідях, з акцентом на теми, де користувач допускав помилки.

Програмний продукт забезпечує зручний вебінтерфейс, інтеграцію з базою даних PostgreSQL для зберігання користувачів і тестів, а також Redis — для швидкого доступу до згенерованих питань. Архітектура передбачає масштабованість, модульність і підтримку майбутніх розширень, зокрема покращених алгоритмів генерації та аналізу відповідей.

1.2 Мета

Метою розробки даного програмного забезпечення є створення інтелектуальної вебплатформи, яка дозволяє автоматично генерувати тестові завдання на основі завантаженого текстового матеріалу за допомогою технологій штучного інтелекту.

Система має забезпечити:

* швидкий та зручний інструмент для формування тестів без необхідності ручного створення кожного запитання;
* підвищення ефективності навчального процесу за рахунок персоналізованого тестування;
* збереження та керування згенерованими тестами для подальшого використання;
* можливість редагування згенерованих питань відповідно до потреб користувача;
* інтерактивне проходження тестів з урахуванням попередніх результатів користувача.

У підсумку платформа має стати корисним інструментом для викладачів, студентів, учнів та будь-яких користувачів, які зацікавлені у швидкому створенні якісного навчального контенту у формі тестів.

1.3 Межі

Програмне забезпечення призначене для генерації тестових завдань на основі наданого текстового матеріалу з використанням технологій штучного інтелекту. Основними функціональними межами системи є:

* джерело даних — текст, завантажений користувачем через інтерфейс платформи. Система не шукає інформацію в інтернеті й не обробляє мультимедійний контент (зображення, відео);
* генерація питань здійснюється на основі одного документа за один сеанс. Масова генерація або обробка декількох файлів одночасно не підтримується;
* типи завдань — на поточному етапі підтримуються лише питання з одним або кількома варіантами відповідей (multiple choice). Інші формати (відкриті питання, true/false тощо) не реалізовані;
* аудиторія — система орієнтована на викладачів, учнів і студентів, які працюють із тестовими форматами перевірки знань;
* права доступу — розмежування користувачів за ролями (наприклад, адміністратор, викладач, студент) поки що не реалізовано;
* інтеграції — не передбачено прямої інтеграції з LMS (Learning Management Systems) або зовнішніми базами даних;
* збереження результатів тестування — реалізоване лише частково. Платформа зберігає історію згенерованих тестів і відповіді користувача, але не формує повну аналітику.

Ці межі визначають функціональний обсяг продукту на поточному етапі розробки. У подальших ітераціях можливе розширення функціональності, зокрема підтримка нових типів завдань, рольова модель доступу, інтеграція з навчальними платформами.

1.4 Посилання

У процесі створення цього документа та розробки програмного продукту були використані наступні джерела нормативної, технічної та методичної документації:

* FastAPI. FastAPI. URL: <https://fastapi.tiangolo.com/> (дата звернення: 26.05.2025);
* PostgreSQL: Documentation. PostgreSQL: The world's most advanced open source database. URL: <https://www.postgresql.org/docs/> (дата звернення: 10.05.2025);
* Redis. Docs. URL: <https://redis.io/docs/> (date of access: 10.05.2025);
* Using HTTP cookies - HTTP | MDN. MDN Web Docs. URL: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies> (дата звернення: 26.05.2025).

1.5 Означенні та абревіатури

У цьому документі використовуються наступні терміни, означення та абревіатури:

* штучний інтелект — підсистема, яка автоматично генерує тестові завдання на основі вхідного тексту за допомогою моделей обробки природної мови (NLP);
* тест — набір питань, сформований на основі одного текстового документа;
* питання (Question) — окрема одиниця тестування, що містить текст питання, варіанти відповідей, правильну відповідь, теги та зв'язок із тестом;
* UUID (Universally Unique Identifier) — унікальний ідентифікатор, що використовується для визначення сутностей у базі даних (наприклад, test\_id, question\_id, user\_id);
* UserTests — таблиця зв'язку між користувачем і тестом, яка зберігає інформацію про володіння тестом;
* redis — високошвидкісне сховище типу "ключ-значення", що використовується для зберігання тестових питань та пов’язаної інформації;
* postgresql — реляційна система управління базами даних, яка використовується для збереження даних про користувачів і тестування;
* API (Application Programming Interface) — інтерфейс взаємодії між клієнтом і сервером для отримання або відправки даних;
* UI (User Interface) — користувацький інтерфейс, через який відбувається взаємодія користувача із системою;
* JSON (JavaScript Object Notation) — формат зберігання і передачі даних, який використовується в запитах і відповідях між клієнтом і сервером;
* frontend — клієнтська частина застосунку, реалізована за допомогою вебінтерфейсу;
* backend — серверна частина системи, яка обробляє запити, зберігає та обробляє дані, реалізує логіку генерації тестів.

**2 ЗАГАЛЬНИЙ ОПИС**

2.1 Перспективи продукту

Розроблюване програмне забезпечення — це сучасна вебплатформа, яка використовує штучний інтелект для автоматичного створення тестів на основі завантаженого тексту. Основною метою є полегшення процесу підготовки тестових матеріалів для викладачів, студентів, учнів та інших зацікавлених користувачів. Застосування штучного інтелекту дає змогу значно зменшити витрати часу на створення якісних, тематично релевантних тестів.

У майбутньому система має значний потенціал для масштабування та розвитку:

* інтеграція з навчальними платформами (Moodle, Google Classroom тощо) для безшовного використання у навчальному процесі;
* підтримка різних форматів тестів, включно з відкритими запитаннями, тестами на відповідність, кросвордами тощо;
* адаптивне навчання — використання аналітики для формування індивідуальних маршрутів навчання на основі відповідей користувача.
* статистика й аналітика — надання детальної інформації про успішність проходження тестів, слабкі місця, популярні теми;
* мультимовна підтримка — можливість генерації тестів на різних мовах.
* розширена персоналізація — створення тестів відповідно до навчального рівня або цілей користувача.

Завдяки відкритій архітектурі з підтримкою API платформа може бути легко інтегрована у більші освітні екосистеми. Подальший розвиток включатиме також мобільну версію застосунку, розширену рольову модель для адміністраторів, викладачів і студентів, а також модуль рекомендацій на базі штучного інтелекту.

2.2 Функції продукту

Розроблювана система забезпечує низку ключових функцій, які охоплюють повний цикл роботи з тестами — від завантаження тексту до проходження користувачем та аналізу результатів. Основні функції продукту включають:

1. авторизація та управління користувачами
   1. реєстрація нового користувача з валідацією email;
   2. вхід до системи за допомогою email та пароля;
   3. рольова модель доступу (звичайний користувач, викладач, адміністратор);
2. завантаження тексту
   1. інтерфейс для завантаження текстових матеріалів (наприклад, уривків з підручників, лекцій тощо);
   2. підтримка форматів .txt, .docx, .pdf (у майбутніх версіях);
3. генерація тестів за допомогою ШІ
   1. автоматичне створення питань із завантаженого тексту за допомогою вбудованого модуля штучного інтелекту;
   2. створення тестів із варіантами відповідей, де кожен варіант має позначку правильності;
   3. тематичне маркування питань (теги) на основі контексту;
4. редагування та збереження тестів
   1. інтерфейс для перегляду, редагування й видалення згенерованих питань;
   2. можливість ручного додавання нових питань або редагування існуючих;
   3. збереження тесту у власному кабінеті користувача;
5. проходження тестів
   1. інтерактивний режим проходження з фіксацією відповідей користувача;
   2. підтримка логіки адаптивного тестування (на основі відповідей користувача пропонуються релевантні наступні питання);
   3. підрахунок результатів після завершення тесту;
6. аналіз результатів
   1. виведення результатів проходження у вигляді відсотків правильних відповідей;
   2. збір інформації про помилки для подальшої адаптації питань;
   3. перегляд історії проходжень користувача;
7. керування тестами та статистикою
   1. збереження й групування тестів за тегами або датою створення;
   2. виведення загальної статистики по тестах, тематиці й правильності відповідей;

2.3 Характеристики користувачів

Система розрахована на декілька категорій користувачів, кожна з яких має різні цілі використання та рівень технічної підготовки. Основні групи користувачів:

1. учні / студенти
   1. мета використання: проходження тестів, підготовка до контрольних, самоперевірка знань;
   2. рівень технічної підготовки: базовий;
   3. особливості взаємодії: інтуїтивно зрозумілий інтерфейс для проходження тестів та перегляду результатів;
2. викладачі / освітяни
   1. мета використання: створення тестів на основі власних текстів, аналіз результатів студентів;
   2. рівень технічної підготовки: середній;
   3. особливості взаємодії: завантаження навчальних матеріалів, редагування згенерованих питань, формування тестів для різних груп;
3. адміністратори системи
   1. мета використання: забезпечення стабільної роботи системи, контроль за даними користувачів, управління контентом;
   2. рівень технічної підготовки: високий;
   3. особливості взаємодії: доступ до розширених інструментів керування даними, статистикою та правами доступу.

2.4 Загальні обмеження

У процесі проєктування та розробки програмного забезпечення були визначені такі загальні обмеження, які впливають на функціональність, продуктивність і технічну реалізацію системи:

* залежність від стороннього ШІ-сервісу: генерація тестових питань здійснюється за допомогою зовнішнього AI-модуля. У разі недоступності цього сервісу функція генерації буде тимчасово недоступна;
* підтримка браузерів: веб-інтерфейс системи підтримується сучасними браузерами (Chrome, Firefox, Edge, Safari). Робота в застарілих версіях браузерів не гарантується;
* обмеження на збереження даних: для забезпечення продуктивності, згенеровані тести зберігаються в кеші Redis обмежений час. Довготривале збереження відбувається в PostgreSQL лише після підтвердження збереження тесту користувачем;
* ідентифікація користувачів: для доступу до розширених функцій необхідна реєстрація та аутентифікація користувача. Гостьовий режим має обмежений доступ;
* масштабованість: поточна реалізація орієнтована на середню кількість одночасних користувачів. Для розгортання на великій кількості користувачів потрібно масштабування інфраструктури;
* користувацький досвід: система повинна забезпечувати зручний та інтуїтивно зрозумілий інтерфейс для користувачів різного рівня технічної підготовки. Обмеження можуть виникати через необхідність уніфікації інтерфейсу для різних типів пристроїв та користувачів.

2.5 Припущення й залежності

Для забезпечення стабільної та ефективної роботи системи автоматичної генерації тестів на основі тексту необхідною умовою є постійне й надійне з’єднання з Інтернетом. Це зумовлено тим, що система активно взаємодіє з віддаленими сервісами, зокрема для автентифікації користувачів, обробки вхідного тексту за допомогою моделей штучного інтелекту, збереження результатів та передачі даних між компонентами платформи.

Окрім того, система залежить від коректної роботи бази даних PostgreSQL та сховища Redis, які забезпечують збереження користувацьких даних, тестових завдань та тимчасове кешування. Без доступу до цих служб неможлива повноцінна функціональність програмного забезпечення.

**3 КОНКРЕТНІ ВИМОГИ**

3.1 Вимоги до зовнішніх інтерфейсів

3.1.1 Інтерфейс користувача

Інтерфейс користувача системи має бути простим, інтуїтивно зрозумілим та кросплатформним, з фокусом на зручність взаємодії як для студентів, так і для викладачів. Інтерфейс реалізується як вебзастосунок.

Користувацький інтерфейс повинен забезпечувати такі функціональні зони:

* головна панель (Dashboard) — відображає список доступних курсів, поточних завдань, результатів тестування, повідомлень;
* модуль генерації тестів — дозволяє викладачеві створювати тести на основі завантажених матеріалів (PDF, DOCX, TXT), переглядати згенеровані питання, редагувати або підтверджувати їх;
* проходження тесту — інтерфейс для студента, що забезпечує покрокове проходження адаптивного або звичайного тестування з відображенням прогресу та таймером;
* модуль статистики та аналітики — надає викладачу доступ до графіків успішності, детальної статистики за результатами тестування, рейтингів та зведеної інформації по групах/студентах;
* система навігації — реалізована у вигляді бічного меню або вкладок, забезпечує доступ до всіх функціональних блоків;
* сторінка профілю користувача — можливість змінювати особисту інформацію, мову інтерфейсу, отримувати сертифікати.

Основні вимоги до UI:

* адаптивний дизайн: коректне відображення на смартфонах, планшетах, ПК;
* підтримка темної та світлої тем: користувач може перемикати;
* уніфікований стиль: всі елементи оформлені відповідно до єдиної дизайн-системи (типографіка, іконки, кольорова палітра);
* підтримка української та англійської мов;
* відповідність принципам WCAG (Web Content Accessibility Guidelines) для базової доступності (контрастність, навігація з клавіатури, alt-тексти).

Інтерфейс має бути розроблений відповідно до принципів мінімалістичного UX-дизайну: лише необхідні елементи, акценти на основних діях, уникнення перевантаження інформацією.

3.1.2 Апаратний інтерфейс

Оскільки програмний продукт реалізується як вебзастосунок, що працює через браузер на стороні клієнта, апаратна залежність має бути мінімізована. Взаємодія з апаратною частиною відсутня — жодні специфічні драйвери, контролери або фізичні пристрої не повинні використовуватися.

Програмне забезпечення не потребує встановлення на пристрій користувача і працює в будь-якому сучасному браузері, що підтримує HTML5, CSS3 та JavaScript (наприклад, Google Chrome, Microsoft Edge, Mozilla Firefox). Вся обробка ресурсоємних операцій повинна виконуватися на серверній стороні (backend), що дозволяє працювати з платформою навіть на малопотужних пристроях.

Мінімальні апаратні вимоги до клієнтського пристрою:

* процесор: будь-який двоядерний (1.6 GHz або вище);
* оперативна памʼять: від 2 ГБ;
* підключення до Інтернету: стабільне, зі швидкістю від 1 Мбіт/с;
* роздільна здатність екрана: не менше 1024×768 пікселів.

Серверна частина платформи повинна розгортатися у хмарному середовищі з використанням Kubernetes-кластера, що забезпечує масштабованість, відмовостійкість та автоматизоване управління навантаженням. Взаємодія з хмарною інфраструктурою здійснюється через стандартні API Kubernetes та інтерфейси обміну даними між мікросервісами (gRPC, HTTP REST).

Таким чином, апаратний інтерфейс зводиться до стандартної взаємодії із серверною інфраструктурою в хмарному середовищі, без жодної залежності від фізичних пристроїв користувача.

3.1.3 Програмний інтерфейс

Програмний інтерфейс системи має реалізовуватися шляхом взаємодії окремих компонентів (фронтенду, бекенду та мікросервісів) через чітко визначені API, побудовані за принципами мікросервісної архітектури. Усі компоненти мають взаємодіяти виключно через стандартизовані протоколи, що забезпечує гнучкість, масштабованість і легкість підтримки.

Формати даних:

* JSON — для HTTP REST API;
* Protocol Buffers — для gRPC викликів;
* YAML/JSON — для маніфестів Kubernetes.

Уся система повинна бути реалізована із дотриманням принципів низького звʼязування (low coupling) та високої когезії (high cohesion), що дозволить легко масштабувати, замінювати або оновлювати окремі компоненти без впливу на загальну архітектуру.

3.1.4 Комунікаційний протокол

У процесі розробки програмного комплексу передбачається використання набору комунікаційних протоколів, які забезпечуватимуть ефективну взаємодію між компонентами системи, а також із зовнішніми сервісами. Комунікація між клієнтською та серверною частинами, мікросервісами, а також із LLM API відбуватиметься виключно по захищених та стандартизованих протоколах.

Передбачені комунікаційні протоколи включають:

* HTTPS (HTTP/1.1, HTTP/2) - буде використовуватись для обміну даними між фронтендом та testify-backend. Всі REST API будуть доступні виключно через HTTPS з TLS-шифруванням;
* gRPC - застосовуватиметься для міжсервісної взаємодії (db-gateway, question-builder, key-store-service), що дозволить досягти високої продуктивності, підтримки стрімінгових запитів та стисненого бінарного формату передачі;
* kubernetes API (HTTPS) - для управління об’єктами типу Job, які створюватимуться під час генерації тестових завдань. Доступ здійснюватиметься через кластерний проксі з аутентифікацією сервісними акаунтами;
* DNS / Service Discovery (CoreDNS) - для забезпечення динамічного виявлення мікросервісів усередині Kubernetes-кластера;
* TCP/IP - як базовий транспортний рівень для всіх видів мережевої взаємодії в межах кластера.

Вибір зазначених протоколів зумовлений вимогами до безпеки, масштабованості та низької затримки системи. Усі зовнішні з’єднання передбачається захищати за допомогою TLS/SSL, а доступ до внутрішніх API обмежуватиметься на рівні мережевих політик Kubernetes.

3.1.5 Обмеження пам’яті

Для забезпечення стабільної роботи системи у середовищі Kubernetes передбачено встановлення чітких обмежень споживання оперативної пам’яті для кожного компонента. Це дозволяє уникати перевитрат ресурсів та забезпечити ізольованість контейнерів.

Орієнтовні ліміти пам’яті за мікросервісами визначено на основі експериментальних замірів у тестовому середовищі:

* backend-data-transform – до 1.0 ГБ (враховуючи операції з PDF, відео і текстовими витягами);
* text-segmenter – до 512 МБ (через специфіку роботи з лінгвістичними алгоритмами сегментації);
* question-builder – до 2.0 ГБ на один Job, враховуючи навантаження при взаємодії з зовнішнім LLM API;
* key-store-service – не більше 128 МБ, оскільки сервіс працює у фоновому режимі та не обробляє об’ємні дані;
* db-gateway – до 256 МБ, оскільки Redis-клієнт споживає мінімум пам’яті при простій маршрутизації;
* testify-backend – до 1.0 ГБ, залежно від кількості активних користувачів та запитів до бази даних;
* testify-frontend (в контейнеризованому середовищі) – до 512 МБ, з урахуванням ресурсів на рендерінг та маршрутизацію.

Загальний обсяг оперативної пам’яті для базового розгортання системи у кластері не повинен перевищувати 6–8 ГБ, за умови одночасної роботи не більше 10 користувачів та обмеженої кількості паралельних генерацій. Усі обмеження пам’яті встановлюються через параметри resources.limits.memory у відповідних Kubernetes-манифестах.

3.1.6 Операції

Розроблювана система забезпечує підтримку наступного базового набору операцій, що виконуються на рівні користувача та внутрішніх сервісів:

* завантаження навчальних матеріалів – користувач має можливість завантажувати PDF-файли, відеоматеріали або текстові документи через вебінтерфейс. Після завантаження ресурс автоматично передається на обробку;
* автоматичне вилучення тексту – модуль перетворення (backend-data-transform) виконує конвертацію завантажених ресурсів у текстовий формат для подальшої обробки;
* семантичне сегментування тексту – текст розбивається на логічно завершені фрагменти за допомогою сервісу text-segmenter;
* генерація тестових запитань – на основі сегментованого тексту формуються запитання, варіанти відповідей, правильні відповіді та відповідні тематичні теги з використанням LLM. Процес виконується у рамках K8S Job;
* збереження результатів генерації – усі сформовані запитання передаються через db-gateway у сховище Redis у структурованому вигляді;
* формування адаптивного тесту – на основі тегів і результатів попередніх відповідей система визначає наступні запитання з урахуванням складності та тематики;
* проходження тестування – користувач проходить тест, відповідає на запитання, отримує зворотний зв’язок у режимі реального часу.
* редагування та перегляд тестів – користувач має змогу переглядати, редагувати та повторно запускати тести (як згенеровані, так і збережені раніше);
* моніторинг та логування – усі ключові операції логуються в системі моніторингу, що дає змогу проводити аналіз та виявлення збоїв.

Система орієнтована на безперервну роботу в режимі хмарного середовища з мінімальним втручанням з боку адміністратора.

3.2 Атрибути програмного продукту

Програмний продукт матиме наступні ключові атрибути, які забезпечать його відповідність сучасним стандартам якості, масштабованості та зручності використання.

3.2.1 Надійність

Система забезпечуватиме безперебійну роботу сервісів за рахунок мікросервісної архітектури та оркестрації в Kubernetes, з механізмами автоматичного відновлення після збоїв та резервного збереження даних. Синхронізація баз даних і кешів мінімізуватиме ризики втрати інформації.

3.2.2 Продуктивність

Платформа гарантуватиме швидке завантаження та обробку ресурсів відповідно до встановлених SLA. Балансування запитів до LLM через сервіс управління API ключами оптимізує використання ресурсів і знизить затримки. Відгук інтерфейсу користувача не перевищуватиме 200 мс для основних операцій.

3.2.3 Масштабованість

Архітектура буде спроєктована з урахуванням горизонтального масштабування фронтенд- та бекенд-компонентів шляхом динамічного розгортання контейнерів у Kubernetes. Паралелізація модулів генерації запитань і сегментації тексту дозволить ефективно працювати з великими обсягами даних.

3.2.4 Безпека

Система впровадить аутентифікацію і авторизацію на основі стандартних протоколів JWT. Конфіденційні дані, включно з API ключами, будуть захищені за допомогою сучасних методів шифрування. Контроль доступу на рівні ролей гарантуватиме безпеку функцій створення і редагування тестів.

3.2.5 Зручність використання

Інтерфейс забезпечуватиме інтуїтивно зрозумілу навігацію та адаптивний дизайн, сумісний із мобільними і десктопними пристроями. Користувачі отримуватимуть оперативний зворотній зв’язок під час тестування і редагування питань, а кількість кроків для виконання основних дій буде мінімізована.

3.2.6 Портативність

Платформа розгортатиметься у хмарних середовищах на базі Kubernetes, що забезпечить переносимість між провайдерами. Фронтенд реалізовуватиметься як веб-додаток, сумісний із сучасними браузерами на основних операційних системах.

3.2.7 Підтримуваність

Кодова база буде модульною для спрощення підтримки та розвитку функціоналу. Інфраструктура описуватиметься у вигляді Infrastructure-as-Code, що дозволить автоматизувати розгортання та операції. Документація підтримуватиметься актуальною, включаючи описи API, архітектури та бізнес-логіки.

**4 ВИМОГИ ДО БАЗИ ДАНИХ**

У системі використовуватиметься розподілена модель зберігання даних з опорою на високопродуктивні бази даних, зокрема Redis для кешування і проміжного зберігання, а також персистентне сховище для довготривалого зберігання тестових завдань.

Основні вимоги до бази даних будуть включати:

* структурованість: Усі згенеровані запитання, варіанти відповідей, правильні відповіді та тематичні теги зберігатимуться у структурованому вигляді з підтримкою зв’язків між сутностями (наприклад, Question, Answer, Tag, Topic, User);
* швидкодія: Записи та запити до бази даних мають виконуватися з низькою затримкою. Для забезпечення швидкого доступу до найбільш популярного контенту буде реалізовано кешування з використанням Redis;
* висока доступність і відмовостійкість: База даних буде розгорнута у режимі з реплікацією та резервним копіюванням. Підтримуватиметься автоматичне переключення у випадку відмови основного вузла;
* сумісність: Всі компоненти, включаючи Python та Go сервіси, матимуть уніфікований доступ до бази даних через спеціалізований db-gateway сервіс, реалізований за допомогою gRPC;
* безпека: Доступ до бази даних буде обмежений через мережеві політики та механізми автентифікації між сервісами. Конфіденційні дані (наприклад, ідентифікатори користувачів або результати тестування) шифруватимуться як під час передачі, так і у стані зберігання;
* масштабованість: Архітектура зберігання дозволятиме горизонтальне масштабування за рахунок використання шардування або розподілу даних за ключами (наприклад, за тематиками або користувачами);
* гнучкість запитів: База даних має підтримувати складні фільтраційні запити — зокрема, пошук питань за тегами, темами, рівнем складності або історією відповідей користувача.

Схема бази даних має формуватися з урахуванням потреб адаптивного тестування, аудиту змін, повторного використання питань та масштабування під навантаження.