Міністерство освіти і науки України

Національний університет “Одеська політехніка”

Інститут комп’ютерних систем

Кафедра інженерії програмного забезпечення

**РОЗРАХУНКОВО – ГРАФІЧНА РОБОТА**

з дисципліни «Архітектура та проектування програмного забезпечення»

на тему: Make the Grade

Студента курсу III групи АС-225 спеціальності 121 – Інженерія програмного забезпечення

Перевертня В. В.

Керівник проф. В. В. Любченко

Одеса – 2024

Зміст

[**ВСТУП** 3](#_Toc180686888)

[**1 СПЕЦИФІКАЦІЯ ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ** 5](#_Toc180686889)

[**1.1 Функціональні вимоги** 5](#_Toc180686890)

[**1.2 Нефункціональні вимоги** 8](#_Toc180686891)

[**2 БАЗОВИЙ АРХІТЕКТУРНИЙ ПРОЕКТ** 11](#_Toc180686892)

[**2.1 Процесне подання** 11](#_Toc180686893)

[**2.2 Логічне подання** 22](#_Toc180686894)

[**2.3 Опис і обґрунтування прийнятих рішень** 22](#_Toc180686895)

[**3 ОЦІНЮВАННЯ БАЗОВОЇ АРХІТЕКТУРИ** 23](#_Toc180686896)

[**3.1 Оцінка архітектури методом АТАМ** 23](#_Toc180686897)

# **ВСТУП**

В рамках виконання розрахунково-графічної роботи виконується проектування кати «Make the Grade». Проаналізуємо опис кати.

|  |  |
| --- | --- |
| Дуже велика і густонаселена держава хоче нову систему для підтримки стандартизованого тестування у всіх державних шкільних системах для класів з 3 по 12.  **Користувачі**: понад 40,000 студентів, 2000 екзаменаторів, 50 адміністраторів.  **Вимоги**:   * Студенти зможуть користуватися додатком тільки в центрах тестування по всій державі, більшість з яких буде в школах, але не всі. * Студенти повинні мати можливість пройти тест, і результати в кінцевому підсумку мають бути зібрані в єдиному місці, яке представляє всі бали по штату (по школі, вчителю та студенту). * Тести будуть містити запитання з вибором відповідей, короткі відповіді та есе. * Система повинна мати систему звітності, щоб знати, які студенти пройшли тести і які бали вони отримали. * Запитання з короткими відповідями та есе будуть оцінюватися вручну вчителями, які потім додадуть оцінки за есе в систему.   **Додатковий контекст**:   * Процес затвердження змін, що передбачає участь трьох різних урядових агентств, є необхідним для зміни способу зберігання студентських оцінок для забезпечення безпеки. * Держава не володіє своїм центром хостингу, а передає його третій стороні. * Проект має захищати свій бюджет щороку. |  |
| Характеристика якості: Ефективність виконання, Атрибут: ємність. |
| Характеристика якості: Мобільність  Атрибут: Простота інтсал. |
| Характеристика якості:  Ефективність виконання  Атрибут: Використання обчислювальних ресурсів |
| Характеристика якості:  Функц. придатність  Атрибут: функц. повнота |
| Характеристика якості:  Функц. придатність  Атрибут: функц коректність |
| Характеристика якості:  Зручність використання  Атрибут: Працездатність |
| Характеристика якості:  Безпека  Атрибут: Цілісність  Характеристика якості:  Мобільність  Атрибут: Адапт. до змін середовища  Характеристика якості:  Безпека  Атрибут: конфіденційність |

Проаналізувавши кату, можна визначити до 5 функціональних вимог системи, які будуть реалізовувати основний функціонал зазначеної системи для тестування студентів у відповідних центра тестування. Перша та найважливіша вимога обумовлюється проведенням тестування студентів, де передбачаються різні завдання фіксація, перевірка, відправка результатів тестування, що передбачає основні вимоги до системи. Також не менш важливими є і інші функціональні вимоги, без яких першозгадана, не буде функціонувати коректно, це звісно можливість ручної перевірки певних завдань, генерація звітів для отримання даних по всій територіальній області, де буде працювати система.

# **1 СПЕЦИФІКАЦІЯ ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ**

## **1.1 Функціональні вимоги**

Для полегшення сприйняття функціональних вимог до системи, зазвичай використовують візуалізовану форму завдань, які має виконувати система а саме діаграму варіантів використання. За допомоги раніше згаданої діаграми(рис. 1.1) можна зрозуміти хто взаємодіє із системою та які функціональні вимоги будуть взаємодіяти із певними об’єктами («акторами»).

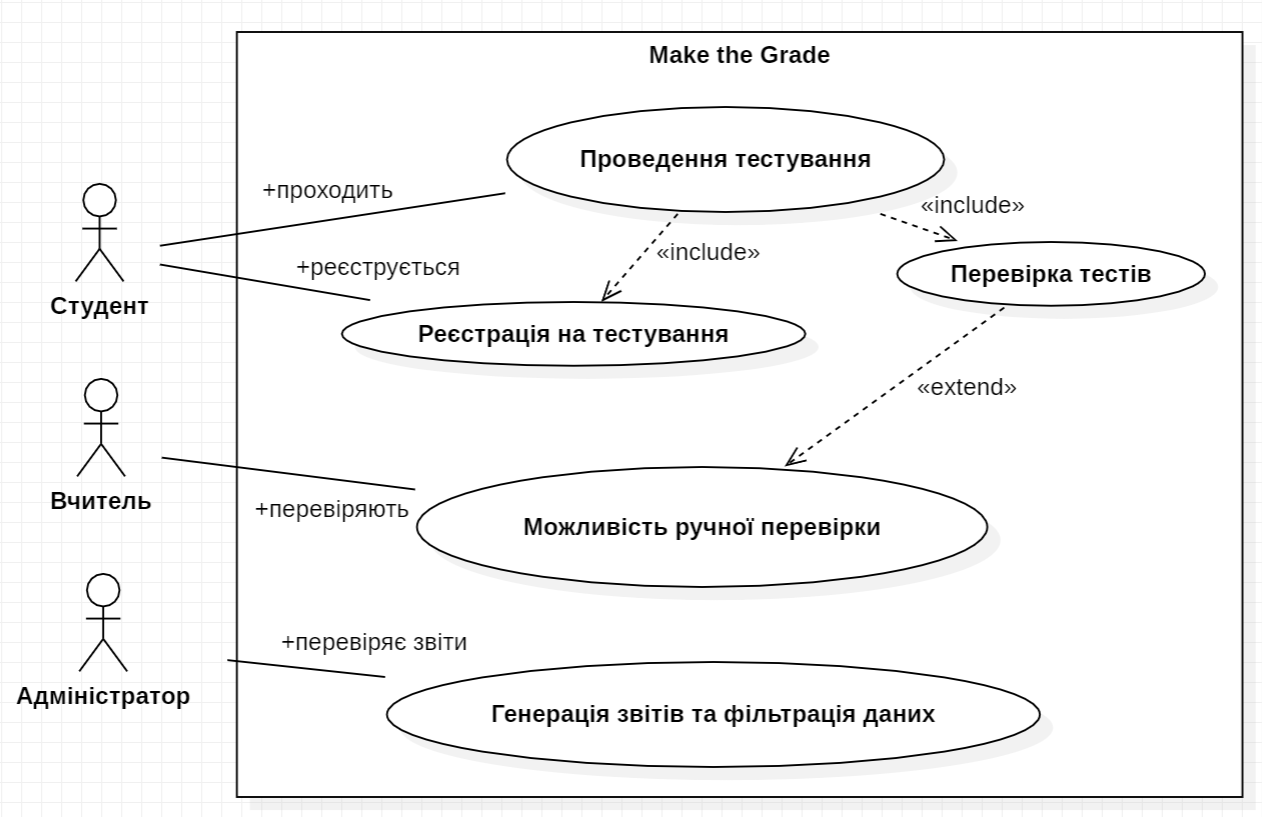


Рисунок 1.1 - Діаграма варіантів використання

Опис варіантів використання та user story

1. Проведення тестування

Користувач: Студент

Передумови: Користувач знаходиться у відповідному центрі тестування.

Сценарій:

1. Користувач намагається авторизуватися.
2. Система перевіряє дані та підтверджує авторизацію.
3. Якщо користувач ввів не валідні дані.

Система повідомляє користувача про помилку.

1. Система показує доступні для даного студента тести.
2. Користувач обирає потрібний тест.
3. Система починає проводити тестування.
4. Користувач проходить тестування та завершує.
5. ВВ «Перевірка тестів».
6. Система показує результат тесту користувачеві.
7. Перевірка тестів

Передумови: Студент пройшов тест

Сценарій:

1. Система отримує відповіді.
2. Система перевіряє типи тестів, на наявність завдань, які потрібно перевіряти користувачеві «Вчитель».
3. Система перевіряє тести.

Якщо тести містять завдання, які система не може перевірити

1. Система надсилає завдання на перевірку користувачеві «Вчитель».
2. Система перевіряє тести, які може перевірити.
3. Система фіксує результати та надсилає дані до заданого місця збереження інформації.
4. Можливість ручної перевірки

Користувач: Вчитель

Передумови: Користувач авторизований, як «Вчитель».

Сценарій:

1. Користувач переходить на сторінку із тестами які потрібно доперевірити.
2. Система показує список тестів які варто перевірити.
3. Якщо тести не знайдено
4. Система повідомляє користувача що наразі немає доступних тестів.
5. Користувач вибирає один із тестів, проводить перевірку та зберігає оновлені дані.
6. Система фіксує зміни та надсилає дані до заданого місця збереження інформації.

4. Генерація звітів та фільтрація даних

Користувач: адміністратор

Передумови: Користувач авторизований, як «Адміністратор».

Сценарій:

1. Користувач переходить на сторінку для генерації звіту.
2. Система запитує тип даних та фільтри за потреби.
3. Користувач вносить дані.
4. Система на базі отриманих даних отримує інформацію та генерує по ній звіт.

Якщо користувач не ввів дані для пошуку.

1. Система повідомить користувача про відсутність параметрів пошуку.

Інформація за запитами не знайдено.

1. Система повідомляє користувача про відсутність інформації за вказаними параметрами.
2. Користувач ознайомлюється із звітом.

5. Реєстрація на тестування

Користувач: студент

Сценарій:

1. Користувач переходить на сторінку для реєстрації на тест.
2. Система запитує дані про користувача
3. Користувач вводить дані про себе(ПІБ, школа, ідентифікаційні дані).
   1. Дані не є коректними
      1. Система просить повторно ввести дані.
   2. Не всі дані введені
      1. Система повідомляє користувача про те що не всі дані введені.
4. Система показує доступні тести для користувача.
   1. Доступних тестів немає.
      1. Система повідомляє користувача, що наразі доступних тестів немає.
5. Користувач обирає потрібний тест.
6. Система реєструє користувача та генерує унікальні дані для тестування.
7. Система повідомляє користувачеві його унікальні дані та реєструє інформацію у сховищі даних.

## **1.2 Нефункціональні вимоги**

1) Надійність:

1. Система повинна мати можливість автоматичного резервного копіювання даних у захищені сховища з інтервалом у 24 години для забезпечення цілісності даних студентів та екзаменаторів.
2. У випадку збою системи (нападу, непередбачені збої в апаратній частині), відновлення системи повинно відбуватися не більше ніж за 1 годину.
3. Система повинна бути доступною майже завжди, за винятком запланованих періодів обслуговування.
4. Усі резервні копії даних повинні бути перевірені на цілісність кожні 24 години для забезпечення їхньої придатності для відновлення.

2) Ефективність:

1. Система повинна обробляти одночасне використання понад 40,000 студентів, 2000 екзаменаторів і 50 адміністраторів без затримок в обробці даних або їх втрати.
2. Час відповіді для користувачів (студентів, вчителів, адміністраторів) не повинен перевищувати 3 секунд на кожну операцію в базовому режимі використання(без критичних умов пікового навантаження або під час відновлення).
3. Система повинна підтримувати до 1000 одночасних сесій тестування на кожному центрі тестування.
4. Максимальний час відгуку при пікових навантаженнях не повинен перевищувати 5 секунд на операцію із системою для кожного користувача.

3) Безпека:

1. Дані студентів, вчителів та адміністраторів повинні зберігатися в зашифрованому вигляді під час передачі та зберігання для захисту від несанкціонованого доступу.
2. Система повинна підтримувати багаторівневу автентифікацію (двофакторну автентифікацію) для адміністраторів та вчителів.
3. Усі операції з обробки та збереження оцінок повинні відповідати державним стандартам безпеки для захисту конфіденційності інформації.
4. Система повинна бути доступною тільки в центрах тестування з використанням контрольованого доступу на основі геолокації або специфічних мережевих налаштувань для шкіл та інших затверджених центрів.

4) Супроводжуваність

1. Код системи повинен бути задокументований так, щоб новий розробник міг почати внесення змін не більше ніж через 2 робочі дні після ознайомлення з проектом.
2. Усі зовнішні бібліотеки та залежності повинні бути оновлені не рідше одного разу на рік або в разі появи критичних оновлень безпеки.
3. Система повинна мати можливість оновлюватись без потреби мануального втручання у кожному тестувальному центрі.
4. Виявлення та виправлення будь-яких помилок або недоліків у системі не повинно перевищувати 5 робочих днів від моменту їх ідентифікації.

5) Юзабіліті

1. Інтерфейс користувача повинен бути інтуїтивно зрозумілим, що дозволить користувачам самостійно виконувати всі необхідні операції без додаткової допомоги або навчання.
2. Середній час завершення основної операції користувача (наприклад, реєстрації) не повинен перевищувати 5 хвилин.
3. Всі форми введення даних повинні містити інструкції або підказки для запобігання помилкам користувачів.
4. Користувачі які використовують розширений функціонал(наприклад, ручна перевірка робіт або отримання звіту) повинні мати змогу навчитися користуватися функціями не більше ніж за 5 годин.

# **2 БАЗОВИЙ АРХІТЕКТУРНИЙ ПРОЕКТ**

## **2.1 Процесне подання**

Опираючись на нефункціональні вимоги, були визначені відповідні архітектурні тактики, які допоможуть визначити необхідні характеристики системи. Проаналізовані нефункціональні вимоги та застосування для кожної тактики наведені у табл. 2.1, де вимоги розбиті на відповідні категорії, які у свою чергу притаманні тактикам.

Таблиця 2.1 – Визначення архітектурних тактик для системи.

|  |  |
| --- | --- |
| Нефункціональні вимоги | Тактики |
| Надійність: Система повинна мати можливість автоматичного резервного копіювання даних у захищені сховища з інтервалом у 24 години для забезпечення цілісності даних студентів та екзаменаторів. | Контрольна точка / відкат |
| Надійність: У випадку збою системи (непередбачені збої в апаратній частині), відновлення системи повинно відбуватися не більше ніж за годину. | Пасивна надмірність (теплий перезапуск) |
| Надійність: Система повинна бути доступною майже завжди, за винятком запланованих періодів обслуговування. | Активна надмірність (гаряче перезавантаження) |
| Надійність: Усі резервні копії даних повинні бути перевірені на цілісність кожні 24 години для забезпечення їхньої придатності для відновлення. | Моніторинг процесу |
| Ефективність: Система повинна обробляти одночасне використання понад 40,000 студентів, 2000 екзаменаторів і 50 адміністраторів без затримок в обробці даних або їх втрати. | Збільшення ресурсів |
| Ефективність: Час відповіді для користувачів (студентів, вчителів, адміністраторів) не повинен перевищувати 3 секунд на кожну операцію в базовому режимі використання(без критичних умов пікового навантаження або під час відновлення). | Введення паралелізму |
| Ефективність: Система повинна підтримувати до 1000 одночасних сесій тестування на кожному центрі тестування. | Введення паралелізму |
| Ефективність: Максимальний час відгуку при пікових навантаженнях не повинен перевищувати 5 секунд на операцію із системою для кожного користувача. | Збільшення ресурсів |
| Безпека: Дані студентів, вчителів та адміністраторів повинні зберігатися в зашифрованому вигляді під час передачі та зберігання для захисту від несанкціонованого доступу. | Шифрування  збереження конфіденційності даних |
|  |  |
| Продовження таблиці 2.1 | |
| Безпека: Система повинна підтримувати багаторівневу автентифікацію (двофакторну автентифікацію) для адміністраторів та вчителів. | Аутентифікація користувачів |
| Безпека: Усі операції з обробки та збереження оцінок повинні відповідати державним стандартам безпеки для захисту конфіденційності інформації. | Підтримка цілісності |
| Безпека: Система повинна бути доступною тільки в центрах тестування з використанням контрольованого доступу на основі геолокації або специфічних мережевих налаштувань для шкіл та інших затверджених центрів. | Обмеження доступу |
| Супроводжуваність: Код системи повинен бути задокументований так, щоб новий розробник міг почати внесення змін не більше ніж через 2 робочі дні після ознайомлення з проектом. | Підтримка семантичної узгодженості |
| Супроводжуваність: Усі зовнішні бібліотеки та залежності повинні бути оновлені не рідше одного разу на рік або в разі появи критичних оновлень безпеки. | Упередження очікуваних змін |
| Супроводжуваність: Система повинна мати можливість оновлюватись без потреби мануального втручання у кожному тестувальному центрі. | Підтримку існуючих інтерфейсів |
| Супроводжуваність: Виявлення та виправлення будь-яких помилок або недоліків у системі не повинно перевищувати 5 робочих днів від моменту їх ідентифікації. | Приховання інформації |
| Юзабіліті: Інтерфейс користувача повинен бути інтуїтивно зрозумілим, що дозволить користувачам самостійно виконувати всі необхідні операції без додаткової допомоги або навчання. | Підтримка моделі користувача |
| Юзабіліті: Середній час завершення основної операції користувача (наприклад, реєстрації) не повинен перевищувати 5 хвилин. | Підтримка моделі задачі |
| Юзабіліті: Всі форми введення даних повинні містити інструкції або підказки для запобігання помилкам користувачів. | Підтримка моделі системи |
| Юзабіліті: Користувачі які використовують розширений функціонал(наприклад, ручна перевірка робіт або отримання звіту) повинні мати змогу навчитися користуватися функціями не більше ніж за 5 годин. | Підтримка моделі користувача |

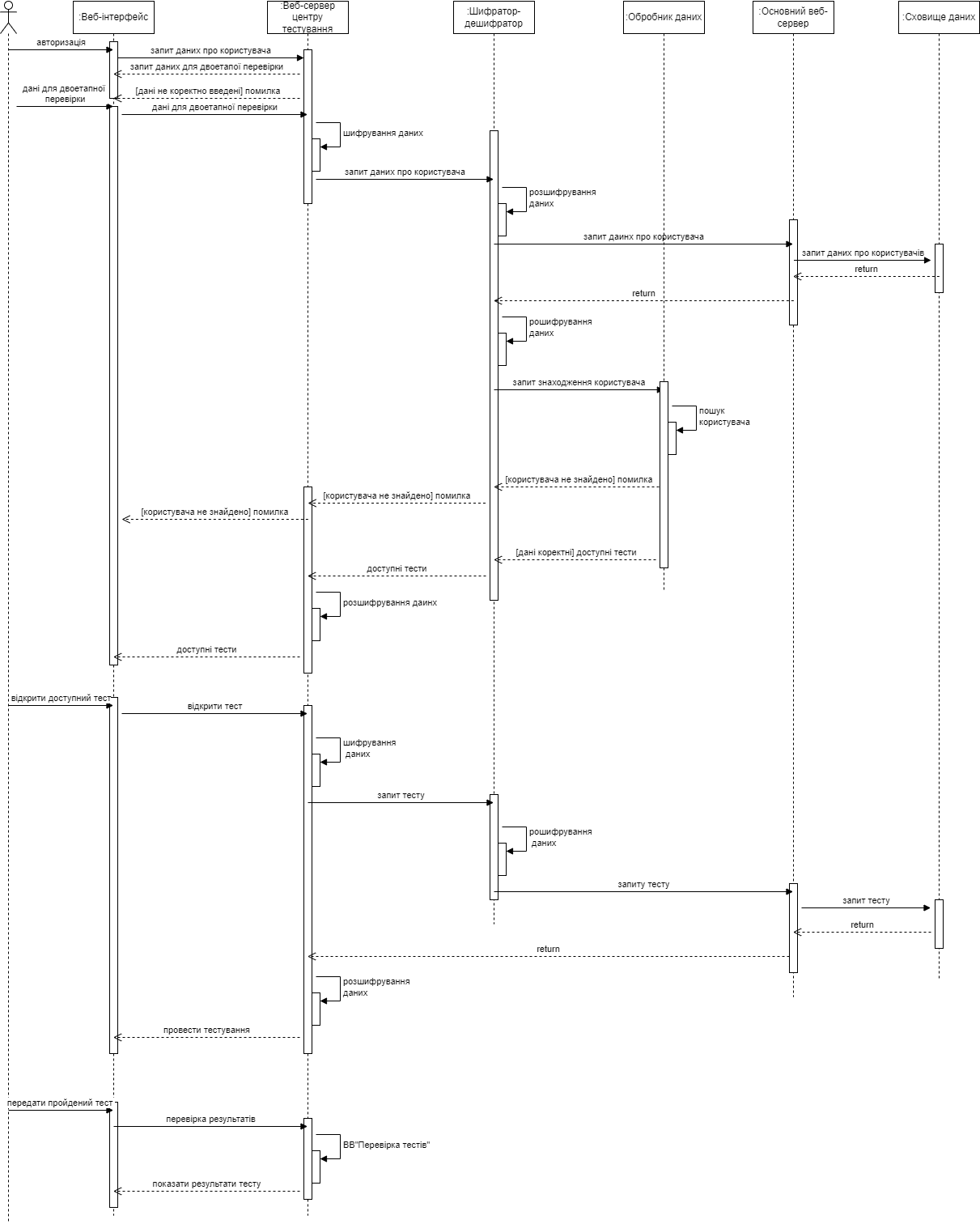


Рисунок 2.1 - Діаграма послідовності ВВ "Проведення тестування"

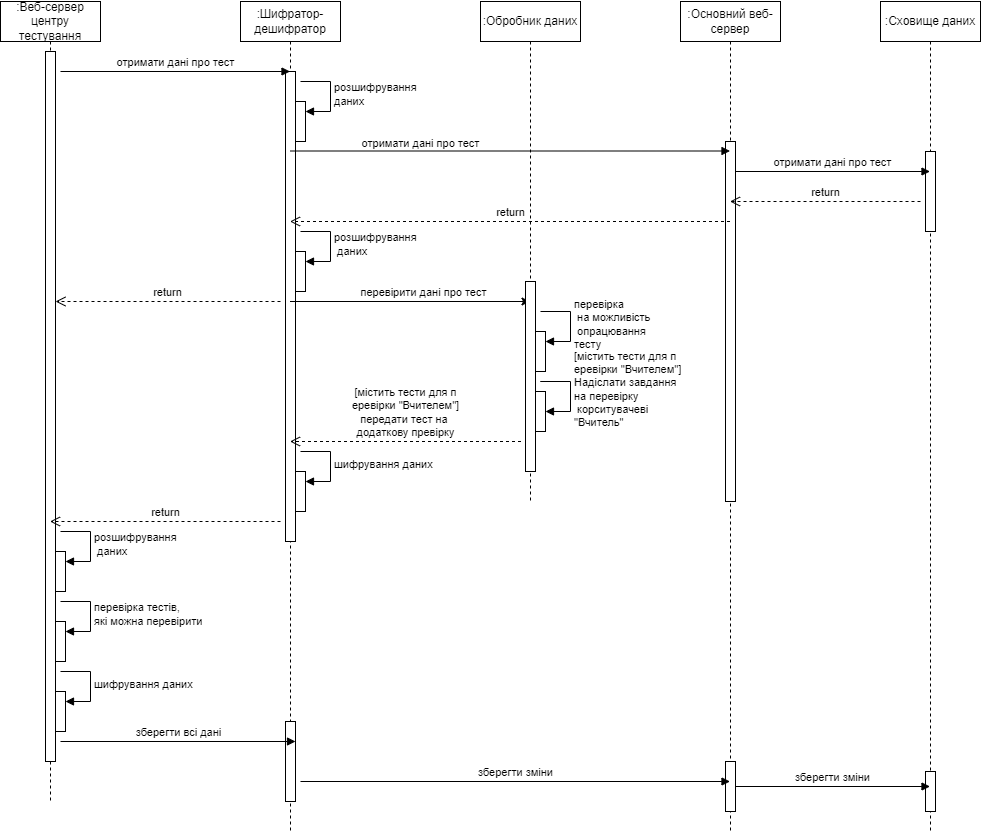


Рисунок 2.2 - Діаграма послідовності для ВВ " Перевірка тестів"

Визначення процесного подання системи, реалізовується за допомоги діаграм послідовності рисунки 2.1 – 2.9 ([посилання](https://drive.google.com/file/d/1kSJ69Qw4i_SPFJTbMzsqe7tZPUFOxRw-/view?usp=sharing) draw.io, де знаходяться діаграми, у разі неможливості їх прочитати зі збільшенням масштабу) де ми чітко можемо визначити підсистеми, методи цих підсистем та як вони будуть взаємодіяти між собою.

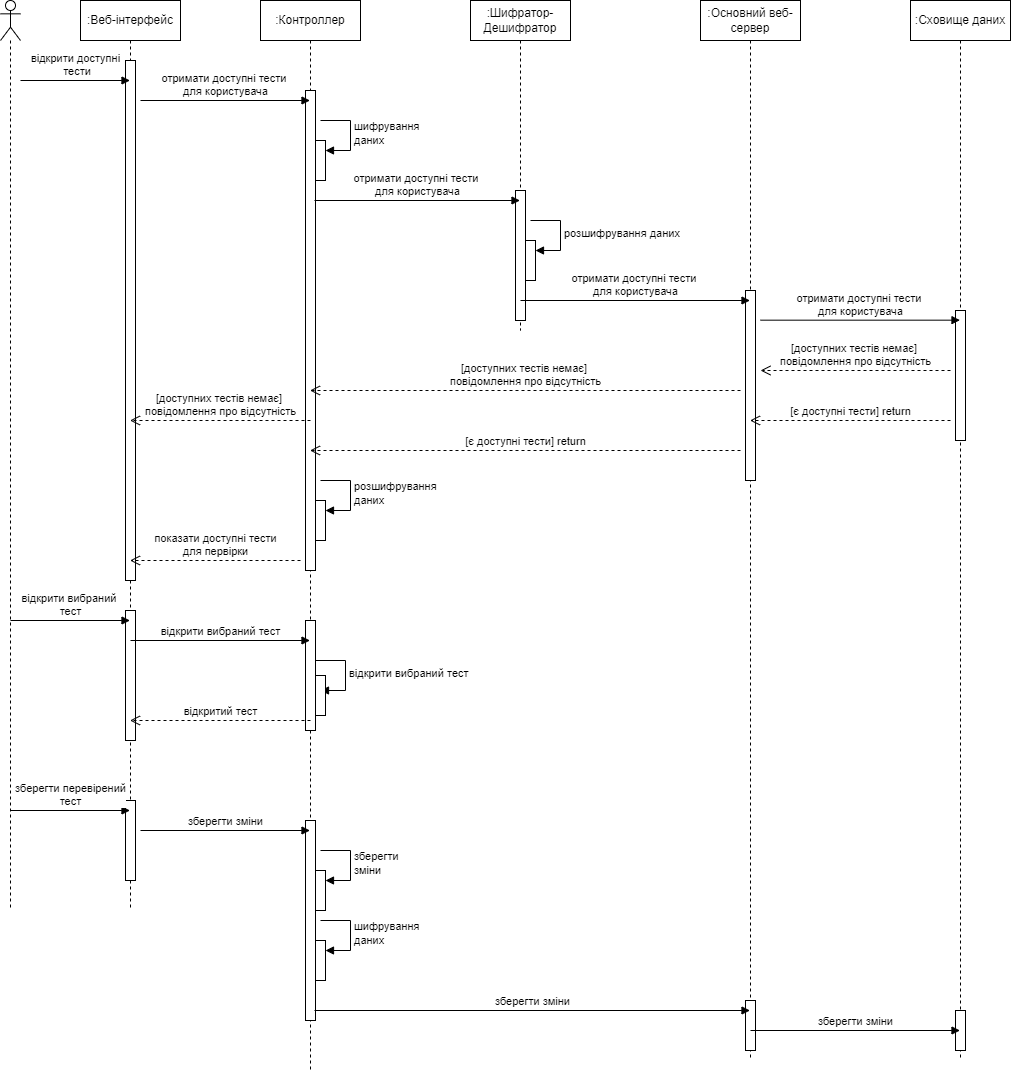


Рисунок 2.3 - Діаграма послідовності для ВВ " Можливість ручної перевірки"

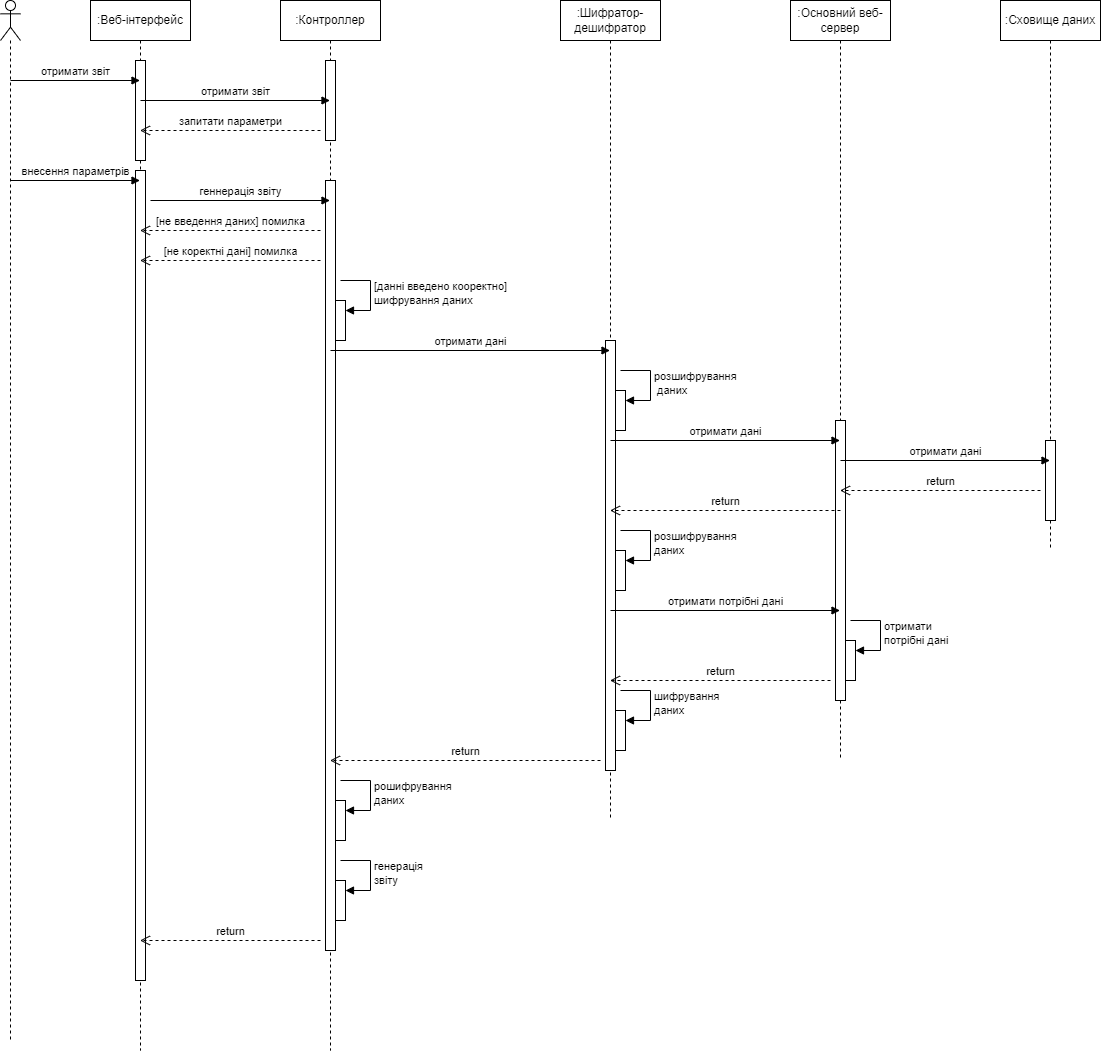


Рисунок 2 4 - Діаграма послідовності для ВВ "Генерація звітів та фільтрація даних"

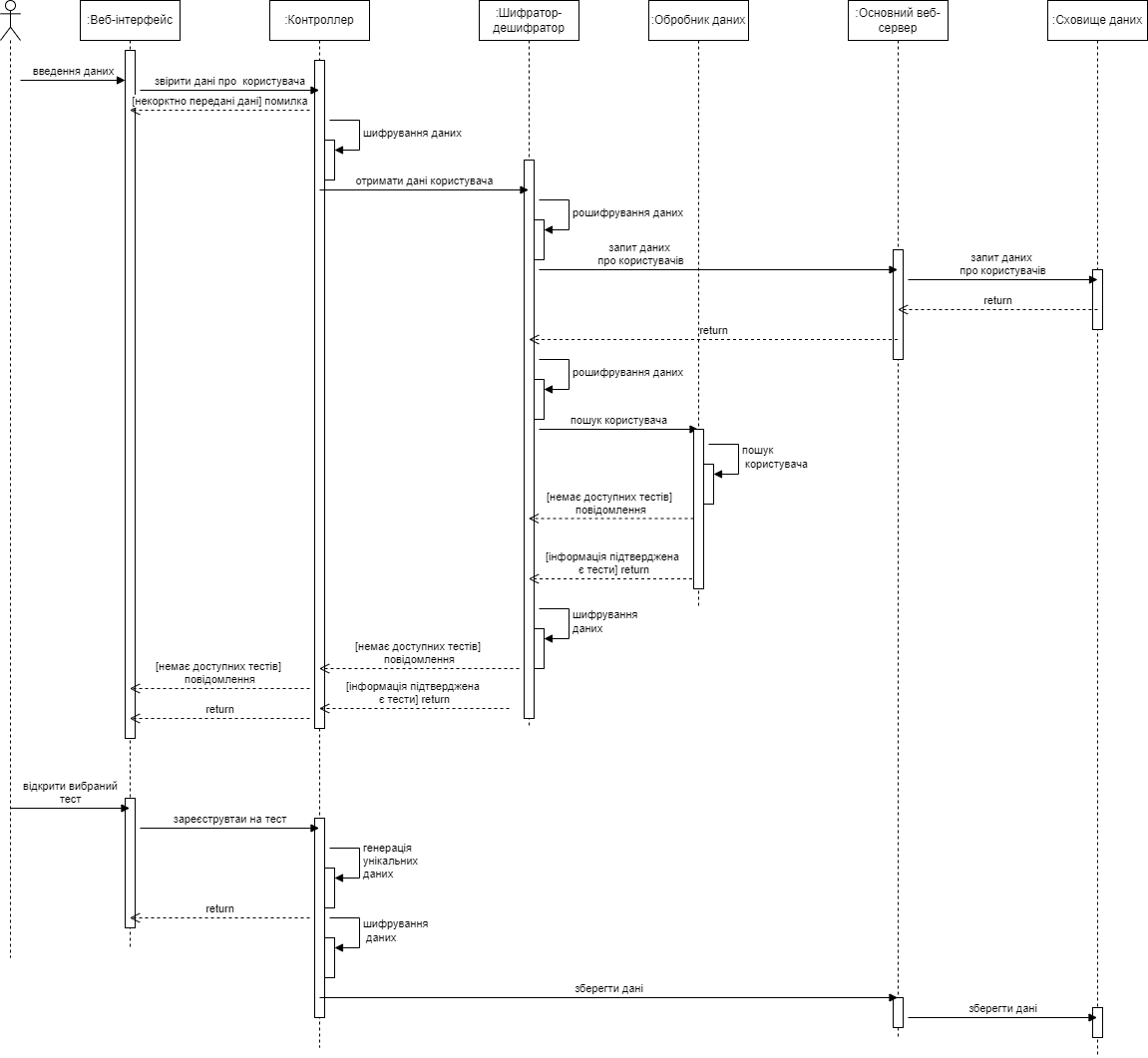


Рисунок 2.5 - діаграма послідовності для ВВ "Реєстрація на тестування"

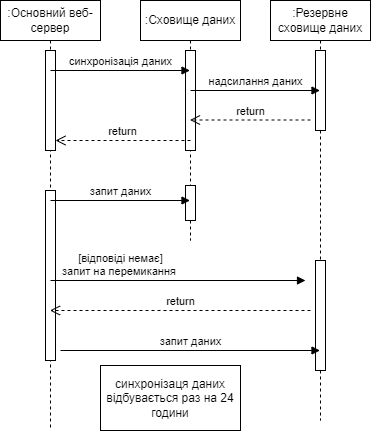


Рисунок 2.6 - діаграма послідовності для покриття нефункціональних вимог 1

Для покриття нефункціональних вимог та визначення окремих моментів у системі, було побудовано додатково ще 3 діаграми послідовності (рис. 2.6 – 2.8) які відображають нефункціональні вимоги, які раніше не були забезпечені у архітектурі, проте після проведення оцінки методом АТАМ та побудовою карт аналізу, було визначено що варто допрацювати.

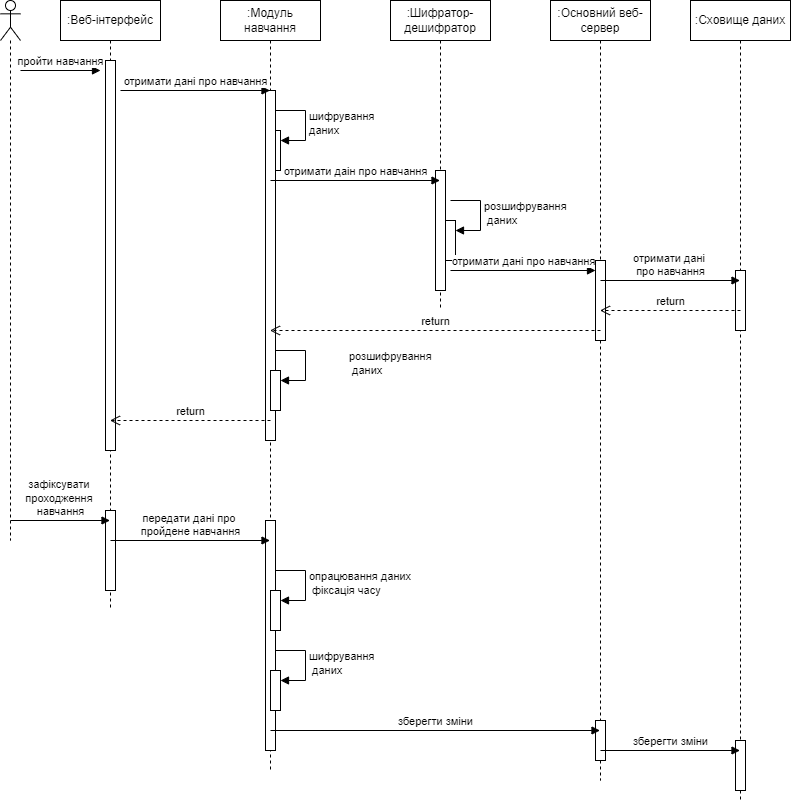


Рисунок 2.7 - діаграма послідовності для покриття нефункціональних вимог 2

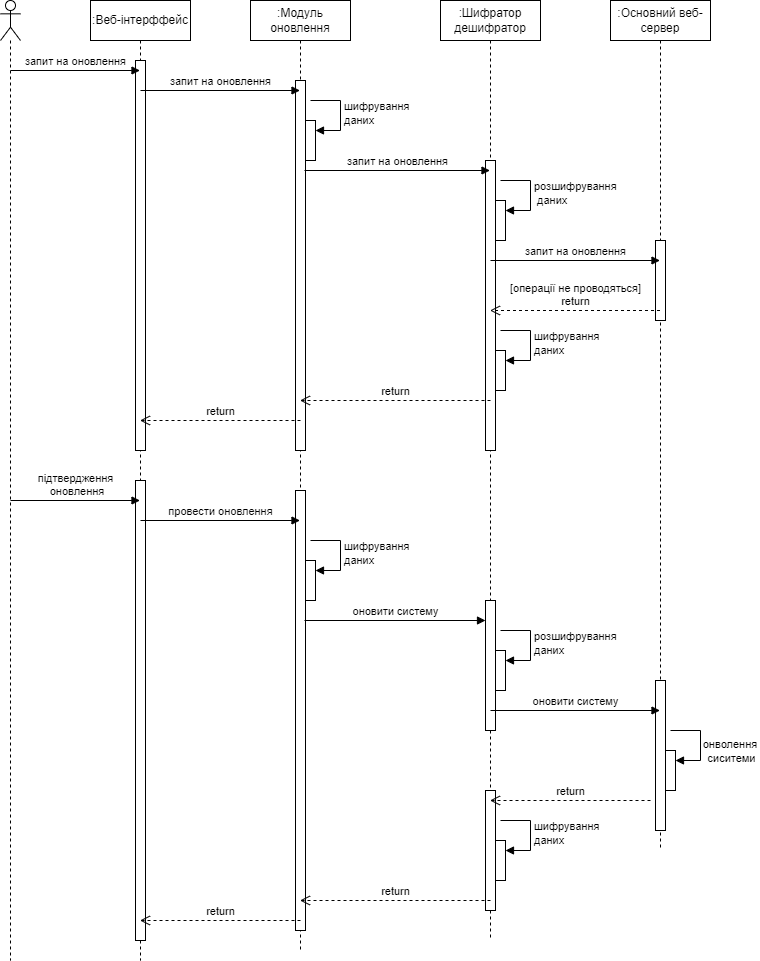


Рисунок 2.8 - діаграма послідовності для покриття нефункціональних вимог 3

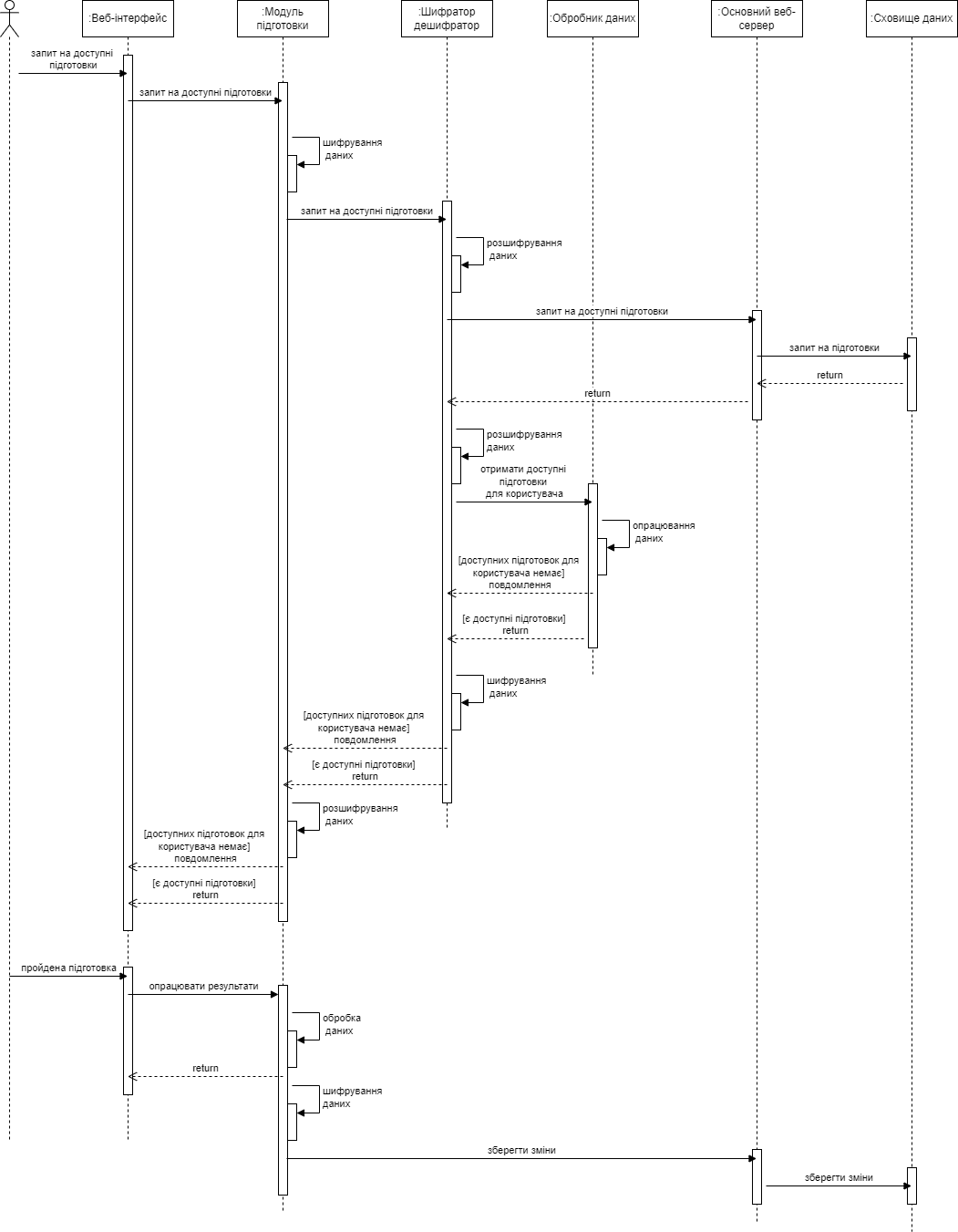


Рисунок 2.9 - забезпечення ВВ " Проведення підготовки до тестів "

## **2.2 Логічне подання**

Для реалізації архітектури системи, одним із важливих є створення логічного подання системи. На рисунку нижче(рис. 2.6), можна побачити декомпозиція системи на 11 підсистем та їх методи які застосовуються, методи та самі підсистеми збираються із процесного подання(рис. 2.1 – 2.9).

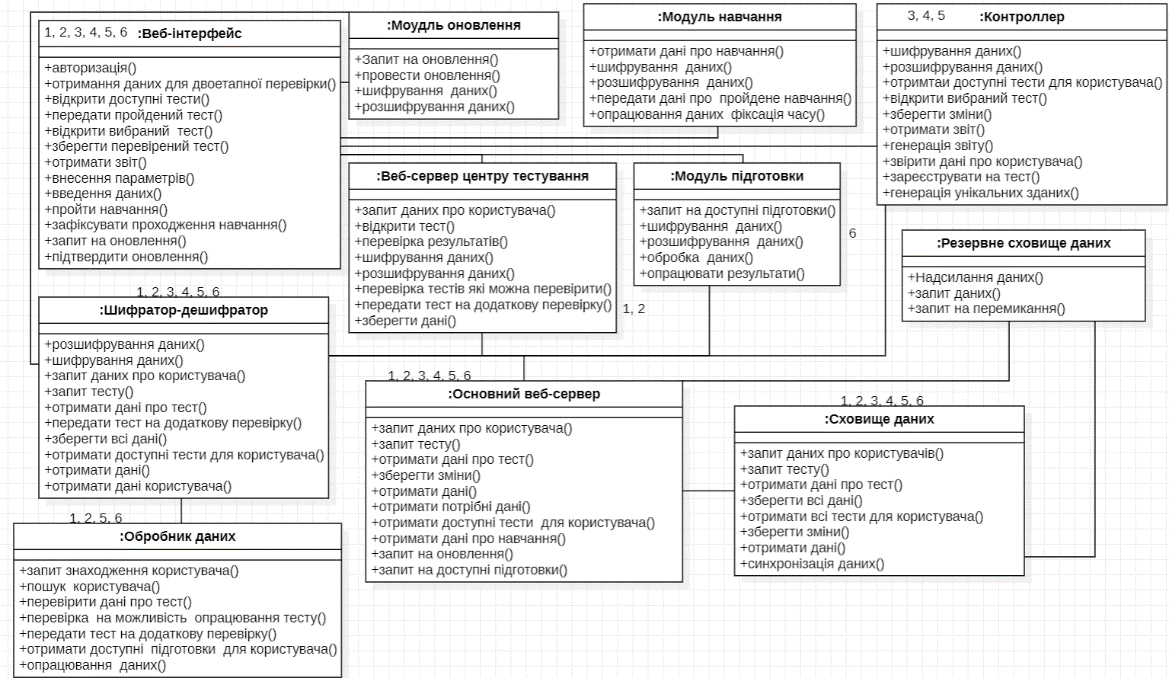


Рисунок 2.10 - Логічне подання системи

## **2.3 Опис і обґрунтування прийнятих рішень**

Для реалізація системи «Make the Grade», було декомпозовано систему на підсистеми представлені у логічному та процесному поданні. Були визначені архітектурні тактики до нефункціональних вимог, та застосовані у процесі архітектури системи. До прикладу для безпеки застосовувались тактики «Шифрування даних» та «Автентифікація користувачів», також було декомпозовано на декілька підсистем одну підсистему для зменшення навантаження відповідно тактикам. Загалом було реалізовано 9 діаграм послідовності відповідно до варіантів використання та одна діаграма(діаграма класів) для логічного подання де наведені всі підсистеми із зв’язками.

# **3 ОЦІНЮВАННЯ БАЗОВОЇ АРХІТЕКТУРИ**

## **3.1 Оцінка архітектури методом АТАМ**

Для визначення того, які пріоритети мають нефункціональні вимоги для системи, визначається «Дерево корисності»(табл. 3.1) де вказується оцінка та пріоритет для нефункціональної вимоги, що у свою чергу допоможе нам визначити, які вимоги варто оцінити методом АТАМ.

Таблиця 3.1 – Дерево корисності

|  |  |  |  |
| --- | --- | --- | --- |
| Характеристика | Сценарій | Оцінка | Пріоритет |
| Надійність | Н1: Система повинна мати можливість автоматичного резервного копіювання даних у захищені сховища з інтервалом у 24 години для забезпечення цілісності даних студентів та екзаменаторів | MH | 3 |
| Надійність | Н2: У випадку збою системи (непередбачені збої в апаратній частині), відновлення системи повинно відбуватися не більше ніж за 1 годину | HH | 1 |
| Надійність | Н3: Система повинна бути доступною майже завжди, за винятком запланованих періодів обслуговування | ML | - |
| Надійність | Н4: Усі резервні копії даних повинні бути  перевірені на цілісність кожні 24 години для  забезпечення їхньої придатності для відновлення | LM | - |
| Ефективність | Е1: Система повинна обробляти одночасне  використання понад 40,000 студентів, 2000  екзаменаторів і 50 адміністраторів без затримок в  обробці даних або їх втрати | HM | 2 |
| Ефективність | Е2: Час відповіді для користувачів  (студентів, вчителів, адміністраторів) не повинен  перевищувати 3 секунд на кожну операцію в базовому режимі використання(без критичних умов пікового навантаження або під час відновлення). | MH | 3 |
| Ефективність | Е3: Система повинна підтримувати до 1000 одночасних сесій тестування на кожному центрі тестування. | HM | 2 |
| Ефективність | Е4: Максимальний час відгуку при пікових  навантаженнях не повинен перевищувати 5 секунд на операцію із системою для кожного користувача. | MH | 3 |
| Безпека | Б1: Дані студентів, вчителів та адміністраторів  повинні зберігатися в зашифрованому вигляді під час передачі та зберігання для захисту від несанкціонованого доступу. | HH | 1 |
|  |  |  |  |
| Продовження таблиці 3.1 | | | |
| Безпека | Б2: Система повинна підтримувати  багаторівневу автентифікацію (двофакторну автентифікацію) для адміністраторів та вчителів. | MM | 4 |
| Безпека | Б3: Усі операції з обробки та збереження оцінок  повинні відповідати державним стандартам безпеки для захисту конфіденційності інформації. | MH | 3 |
| Безпека | Б4: Система повинна бути доступною тільки в  центрах тестування з використанням контрольованого доступу на основі геолокації або специфічних мережевих налаштувань для шкіл та інших затверджених центрів. | HH | 1 |
| Супроводжуваність | С1: Код системи повинен бути задокументований так, щоб новий розробник міг почати внесення змін не більше ніж через 2 робочі дні після ознайомлення з проектом. | LM | - |
| Супроводжуваність | С2: Усі зовнішні бібліотеки та залежності повинні бути оновлені не рідше одного разу на рік або в разі появи критичних оновлень безпеки. | MM | 4 |
| Супроводжуваність | С3: Система повинна мати можливість оновлюватись без потреби мануального  втручання у кожному тестувальному центрі. | ML | - |
| Супроводжуваність | С4: виявлення та виправлення будь-яких помилок або недоліків у системі не повинно перевищувати 5 робочих днів від моменту їх ідентифікації. | MM | 4 |
| Юзабіліті | Ю1: Інтерфейс користувача повинен бути інтуїтивно зрозумілим, що дозволить користувачам самостійно виконувати всі необхідні операції без додаткової допомоги або навчання. | HL | - |
| Юзабіліті | Ю2: Середній час завершення основної операції  користувача (наприклад, реєстрації) не повинен  перевищувати 5 хвилин. | LM | - |
| Юзабіліті | Ю3: Всі форми введення даних повинні містити  інструкції або підказки для запобігання помилкам  користувачів. | ML | - |
| Юзабіліті | Ю4: Користувачі які використовують  розширений функціонал(наприклад, ручна перевірка робіт або отримання звіту) повинні мати змогу навчитися користуватися функціями не більше ніж за 5 годин. | HM | 2 |

Для проведення чіткого аналізу нефункціональних вимог, використовуються карти аналізу де ми маємо базову інформацію про застосування вимоги у архітектурі, та чи вони покриваються. Нижче(табл. 3.2 - 5) наведені чотири карти аналізу для нефункціональних вимог різних рівнів, від найвищого пріоритету 1(HH), до найнижчого 4(MM).

Таблиця 3.2 – Карта аналізу, для пріоритету 1(Б1)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Сценарій # Б1 | | Дані студентів, вчителів та адміністраторів  повинні зберігатися в зашифрованому вигляді пі час передачі та зберігання для захисту від несанкціонованого доступу. | | | | |
| Атрибут(и) | Безпека | | | | | |
| Середовище | Система працює у нормальному стані | | | | | |
| Вплив | Запит користувача | | | | | |
| Реакція | Дані прийшли незмінені та конфіденційні | | | | | |
| Архітектурне рішення | | | Чутливість | Компроміс | Ризик | Не-ризик |
| Шифрування | | |  | К1 |  | НР1 |
| Аргументація | НР1 – дані зберігаються зашифрованими у сховищі. К1 – шифрування, розшифрування впливає на швидкість виконання запитів(Е3). | | | | | |
| Архітектурна діаграма | Рисунок 2.1 | | | | | |

Таблиця 3.3 – Карта аналізу, пріоритет 1(Н2)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Сценарій #Н2 | | У випадку збою системи (непередбачені збої  в апаратній частині), відновлення системи повинно відбуватися не більше ніж за 1 годину | | | | |
| Атрибут(и) | Надійність | | | | | |
| Середовище | Резервні компоненти | | | | | |
| Вплив | Відключення основного компоненту | | | | | |
| Реакція | Перемикання на додатковий компонент | | | | | |
| Архітектурне рішення | | | Чутливість | Компроміс | Ризик | Не-ризик |
| Пасивна надмірність(теплий перезапуск) | | |  | К2 |  | НР2 |
| Аргументація | НР2 – система буде готова до швидкого відновлення. К2 – У разі потреби додаткові компоненти можна буде залучати до роботи для підсилення роботи системи у пікових моментах навантаження(Е1) | | | | | |
| Архітектурна діаграма | Рисунок 2.6 | | | | | |

Таблиця 3.4 – Карта аналізу, для пріоритету 1(Б4)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Сценарій #Б4 | | Система повинна бути доступною тільки в  центрах тестування з використанням контрольованого доступу на основі геолокації або специфічних мережевих налаштувань для шкіл та інших затверджених центрів. | | | | |
| Атрибут(и) | Безпека | | | | | |
| Середовище | Онлайн | | | | | |
| Вплив | Спроба запиту отримати доступ до системи не у зазначеному місці | | | | | |
| Реакція | Система відхиляє запит | | | | | |
| Архітектурне рішення | | | Чутливість | Компроміс | Ризик | Не-ризик |
| Обмеження доступу | | |  | К3 | Р1 |  |
| Аргументація | Р1 – збільшення ресурсів для системи. К3 – Збільшить час відповіді(Е2) | | | | | |
| Архітектурна діаграма | Рисунок 2.1 | | | | | |

Таблиця 3.5 – Карта аналізу, для пріоритету 2(Е1)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Сценарій #Е1 | | Система повинна обробляти одночасне використання понад 40,000 студентів, 2000 екзаменаторів і 50 адміністраторів без затримок в обробці даних або їх втрати. | | | | |
| Атрибут(и) | Ефективність. | | | | | |
| Середовище | Нормальне функціонування. | | | | | |
| Вплив | Надходження понад 42050 запитів даних різними типами користувачів. | | | | | |
| Реакція | Дані надходять без затримки або їх втрати. | | | | | |
| Архітектурне рішення | | | Чутливість | Компроміс | Ризик | Не-ризик |
| Збільшення ресурсів | | |  | К4 |  | НР3 |
| Аргументація | НР2 – пришвидшує роботу та збільшує ресурс системи. К4 – збільшення ресурсів вплине на час відновлення після збою(Н2). | | | | | |
| Архітектурна діаграма | Рисунок 2.2 | | | | | |

Таблиця 3.6 – Карта аналізу, для пріоритету 2(Ю4)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Сценарій #Ю4 | | Користувачі які використовують розширений функціонал(наприклад, ручна перевірка робіт або отримання звіту) повинні мати змогу навчитися користуватися функціями не більше ніж за 5 годин. | | | | |
| Атрибут(и) | Юзабіліті | | | | | |
| Середовище | Система працює в нормальному стані | | | | | |
| Вплив | Користувач вирішує власні задачі | | | | | |
| Реакція | Користувач не припускає більше однієї помилки на одну функцію. | | | | | |
| Архітектурне рішення | | | Чутливість | Компроміс | Ризик | Не-ризик |
| Модель користувача | | |  |  | Р2 |  |
| Аргументація | Р2 – У користувача може зайняти більше часу навчання. | | | | | |
| Архітектурна діаграма | Рисунок 2.7 | | | | | |

Таблиця 3.7 – Карта аналізу, для пріоритету 2(Е3)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Сценарій #Е3 | | Система повинна підтримувати до 1000 одночасних сесій тестування на кожному центрі тестування. | | | | |
| Атрибут(и) | Ефективність | | | | | |
| Середовище | Нормальне функціонування | | | | | |
| Вплив | Надходження не менше ніж 1000 запитів на проводження тесту | | | | | |
| Реакція | Система проводить 1000 сесій тестування | | | | | |
| Архітектурне рішення | | | Чутливість | Компроміс | Ризик | Не-ризик |
| Збільшення ресурсів | | |  | К5 |  | НР4 |
| Аргументація | НР4 – Дозволить швидше опрацьовувати запити. К5 – збільшення ресурсів вплине на час відновлення після збою(Н2). | | | | | |
| Архітектурна діаграма | Рисунок 2.2 | | | | | |

Таблиця 3.9 – Карта аналізу, для пріоритету 3(Б3)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Сценарій #Б3 | | Усі операції з обробки та збереження оцінок  повинні відповідати державним стандартам  безпеки для захисту конфіденційності інформації | | | | |
| Атрибут(и) | Безпека | | | | | |
| Середовище | Онлайн | | | | | |
| Вплив | Зміна оцінок | | | | | |
| Реакція | Оцінки змінені та збережені без несанкціонованих змін чи читання | | | | | |
| Архітектурне рішення | | | Чутливість | Компроміс | Ризик | Не-ризик |
| Підтримка цілісності | | |  | К6 | Р2 |  |
| Аргументація | Р2 – оцінки можуть бути несанкціонована змінені або прочитані. К6 – зменшення швидкості системи у пікові навантаження(Е4) | | | | | |
| Архітектурна діаграма | Рисунки: 2.1, 2.2, 2.3 | | | | | |

Таблиця 3.10 – Карта аналізу, для пріоритету 3(Е4)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Сценарій #Е4 | | Максимальний час відгуку при пікових навантаженнях не повинен перевищувати 5  секунд на операцію із системою для кожного  користувача. | | | | |
| Атрибут(и) | Ефективність | | | | | |
| Середовище | Онлайн | | | | | |
| Вплив | Проведення операцій, кількість яких, схожа або перевищує норму(42 тис. користувачів одночасно). | | | | | |
| Реакція | Проведення кожної операції не довше ніж 5 секунд | | | | | |
| Архітектурне рішення | | | Чутливість | Компроміс | Ризик | Не-ризик |
| Введення паралелізму | | | Ч1 | К7 | Р3 |  |
| Аргументація | Р3 – швидка відповідь при пікових навантаженнях може призвести до втрати або пошкодження інформації – Ч1. К4 – впровадження паралелізму збільшить час відновлення системи(Н2). | | | | | |
| Архітектурна діаграма | Рисунок 2.2 | | | | | |

Таблиця 3.11 – Карта аналізу, для пріоритету 3(Н1)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Сценарій #Н1 | | Система повинна мати можливість  автоматичного резервного копіювання даних у  захищені сховища з інтервалом у 24 години для  забезпечення цілісності даних студентів та  екзаменаторів | | | | |
| Атрибут(и) | Надійність | | | | | |
| Середовище | Додаткові компоненти | | | | | |
| Вплив | Додавання та опрацювання даних | | | | | |
| Реакція | Фіксація змін раз на добу | | | | | |
| Архітектурне рішення | | | Чутливість | Компроміс | Ризик | Не-ризик |
| Контрольна точка / відкат | | |  |  |  | НР5 |
| Аргументація | НР5 – спростить та пришвидшить відновлення системи. | | | | | |
| Архітектурна діаграма | Рисунок 2.6 | | | | | |

Таблиця 3.12 – Карта аналізу, для пріоритету 3(Е2)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Сценарій #Е2 | | Час відповіді для користувачів не повинен перевищувати 3 секунд на кожну операцію в базовому режимі використання(без критичних умов пікового навантаження або під час відновлення). | | | | |
| Атрибут(и) | Ефективність | | | | | |
| Середовище | Нормальне функціонування | | | | | |
| Вплив | Запит від користувача | | | | | |
| Реакція | Відповідь надходить не довше ніж за 10 секунд | | | | | |
| Архітектурне рішення | | | Чутливість | Компроміс | Ризик | Не-ризик |
| Введення паралелізму | | |  | К8 |  | НР6 |
| Аргументація | НР6 - Пришвидшення роботи системи при паралельному підході зменшить навантаження на певні ресурси. К8 – впровадження паралелізму збільшить час відновлення системи | | | | | |
| Архітектурна діаграма | Рисунок 2.2 | | | | | |

Таблиця 3.13 – Карта аналізу, для пріоритету 4(Б2)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Сценарій #Б2 | | Система повинна підтримувати багаторівневу автентифікацію (двофакторну автентифікацію) для адміністраторів та вчителів. | | | | |
| Атрибут(и) | Безпека | | | | | |
| Середовище | Онлайн | | | | | |
| Вплив | Користувач авторизується | | | | | |
| Реакція | Користувач отримує доступ після проходження двоетапної перевірки | | | | | |
| Архітектурне рішення | | | Чутливість | Компроміс | Ризик | Не-ризик |
| Автентифікація користувачів | | |  | К9 |  | НР7 |
| Аргументація | НР7 - Збільшить ефективність безпеки системи.К9 - при пікових навантаження двофакторна перевірка ускладнить обробку даних системою(Е1) | | | | | |
| Архітектурна діаграма | Рисунок 2.1 | | | | | |

Таблиця 3.14 – Карта аналізу, для пріоритету 4(С2)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Сценарій #С2 | | Усі зовнішні бібліотеки та залежності повинні  бути оновлені не рідше одного разу на рік або в  разі появи критичних оновлень безпеки. | | | | |
| Атрибут(и) | Супроводжуваність | | | | | |
| Середовище | Розробників | | | | | |
| Вплив | Отримання нових оновлень системи | | | | | |
| Реакція | Система оновлюється автоматично | | | | | |
| Архітектурне рішення | | | Чутливість | Компроміс | Ризик | Не-ризик |
| Упередження очікуваних змін | | |  | К10 |  | НР8 |
| Аргументація | НР8 – оновлення призводять до покращення та збільшення ефективності системи. К10 - Не чітко заплановані та перевірені оновлення можуть призвести до збою системи. | | | | | |
| Архітектурна діаграма | Рисунок 2.8 | | | | | |

Таблиця 3.15 – Карта аналізу, для пріоритету 4(С4)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Сценарій #С4 | | виявлення та виправлення будь-яких помилок  або недоліків у системі не повинно перевищувати 5 робочих днів від моменту їх ідентифікації. | | | | |
| Атрибут(и) | Супроводжуваність. | | | | | |
| Середовище | Розробників. | | | | | |
| Вплив | Виявлення помилок у роботі системи. | | | | | |
| Реакція | Помилки виправляються не довше ніж 5 робочих днів. | | | | | |
| Архітектурне рішення | | | Чутливість | Компроміс | Ризик | Не-ризик |
| Зменшення зв’язаності компонентів | | |  | К11 |  | НР9 |
| Аргументація | НР9 - Кожен компонент незалежний від реалізації інших, що спрощує внесення змін до системи. К11 – відокремлення компонентів підвищить безпеку даних, так як не всі компоненти мають прямий доступ до даних. | | | | | |
| Архітектурна діаграма | Рисунок 2.1 | | | | | |

Після проведення аналізу було визначено що архітектура не покриває всі нефункціональні вимоги, тому було змінено та покрито всі визначені та потрібні (за пріоритетом) вимоги. Широкий розбір аналізу, тобто аналіз всіх(за пріоритетами) вимог, допоміг покращити та структурувати архітектуру. Змінені діаграми були оновлені та нові додані до даного документу(рис. 2.1 – 2.10).

## **3.2 Оцінка методом SAAM**

Для визначення чи покриті всі функціональні вимоги будується відповідна таблиця, де наводяться функціональні вимоги, їх опис та визначається чи реалізовані вони, чи ні.

Таблиця 3.16 – Оцінювання архітектури на покриття функціональних вимог.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Опис | Реалізовано\ні | Компонент | Потрібні зміни |
| 1 | Проведення тестування | Реалізовано |  |  |
| 2 | Реєстрація на тестування | Реалізовано |  |  |
| 3 | Перевірка тестів | Реалізовано |  |  |
| 4 | Можливість ручної перевірки | Реалізовано |  |  |
| 5 | Генерація звітів та фільтрація даних | Реалізовано |  |  |
| 6 | Проведення підготовки до тестів | Не реалізовано | Веб-інтерфейс,  основний веб-сервер, сховище даних, модуль підготовки | Додати відповідні методи. Додати модуль підготовки. |

Після проведення аналізу на забезпечення функціональних вимог архітектурою, було визначено, що архітектура не відповідає 6-му варіанту використання. Тому було реалізовано ще одну діаграму процесного подання (рис. 2.9) та виправлено логічне подання (рис. 2.10).

Вже коли визначено, що всі функціональні вимоги покриваються, варто проаналізувати наскільки окремі компоненти навантажені по застосуванню, тому на логічному поданні (рис. 2.10) можна побачити числа, які вказують, які саме компоненти взаємодіють у покритті функціональних вимог.

Після проведення аналізу ми визначили, що компонент «Основний веб-сервер», є досить таки навантаженим, тому було визначено новий компонент, та внесені зміни. Кінцевий варіант базової архітектури можна побачити вище (рис. 2.1 – 2.10).

# **4 ЗАСТОСУВАННЯ АРХІТЕКТУРНИХ СТИЛІВ**

## **4.1 Процесне подання із застосованими стилем та шаблонами**

Архітектурні стилі та шаблони, є певними готовими рішеннями, які застосовують для архітектури, таким чином архітектура будується відповідно певного стандарту, що у свою чергу забезпечує чітке покриття певних рішень, та при правильному застосуванні підвищує якість архітектури. Нижче (рис. 4.1 – 4.4) наведено процесне подання архітектури із застосування стилю MVC та шаблонів Broker, Proxy, Publisher-Subscriber.

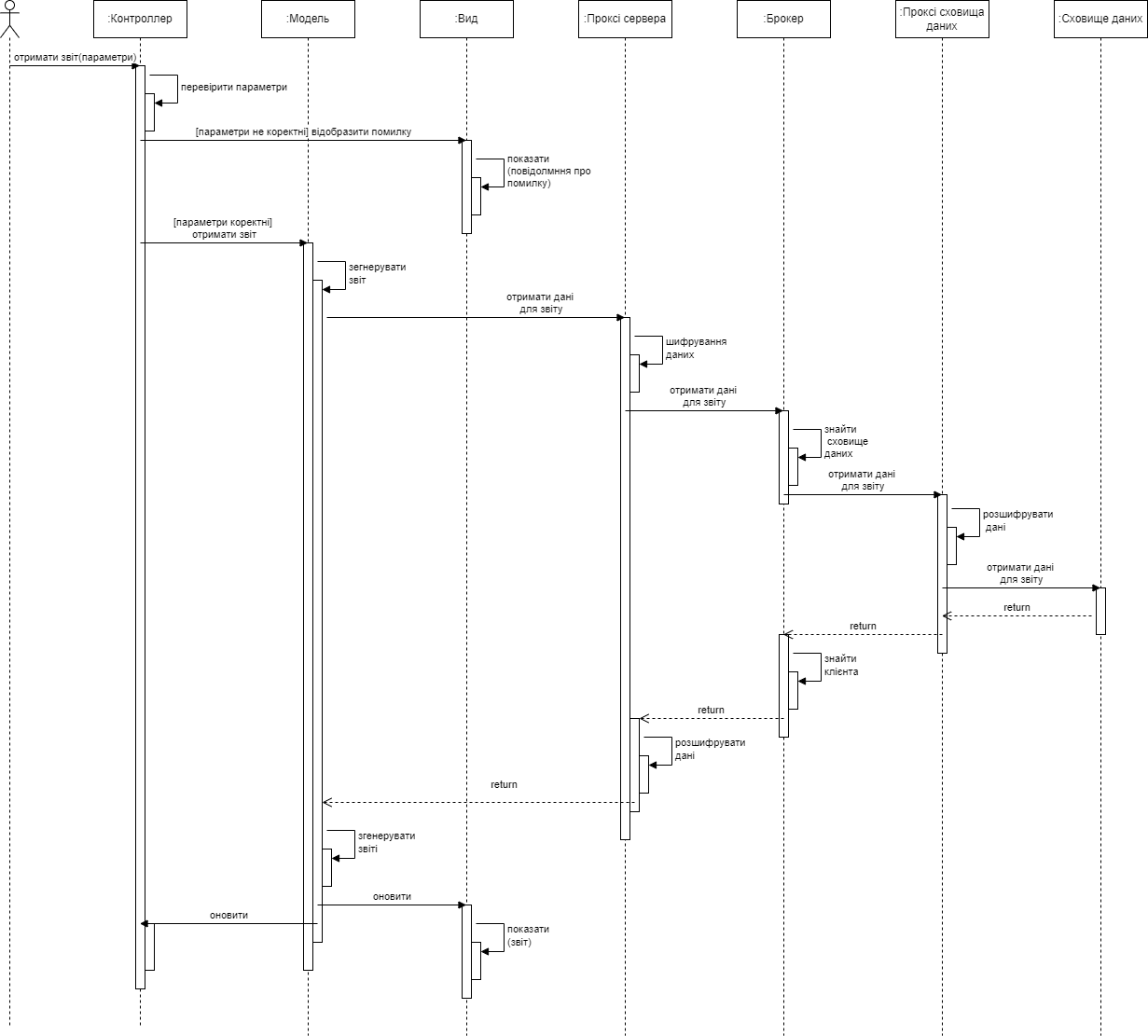


Рисунок 4.1 - забезпечення функціоналу «Отримання звіту»

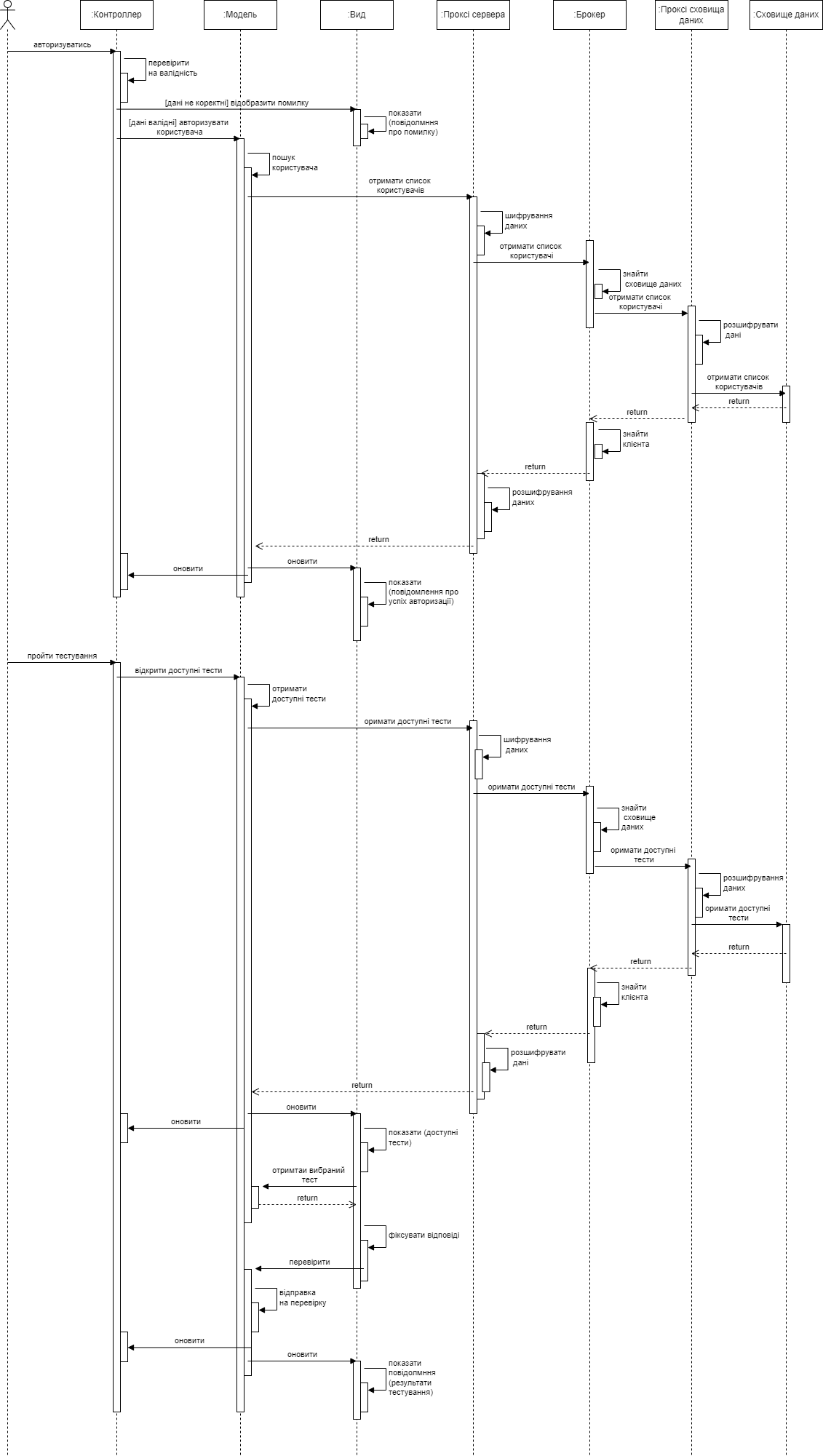


Рисунок 4.2 - забезпечення основного функціоналу "Проведення тестування"

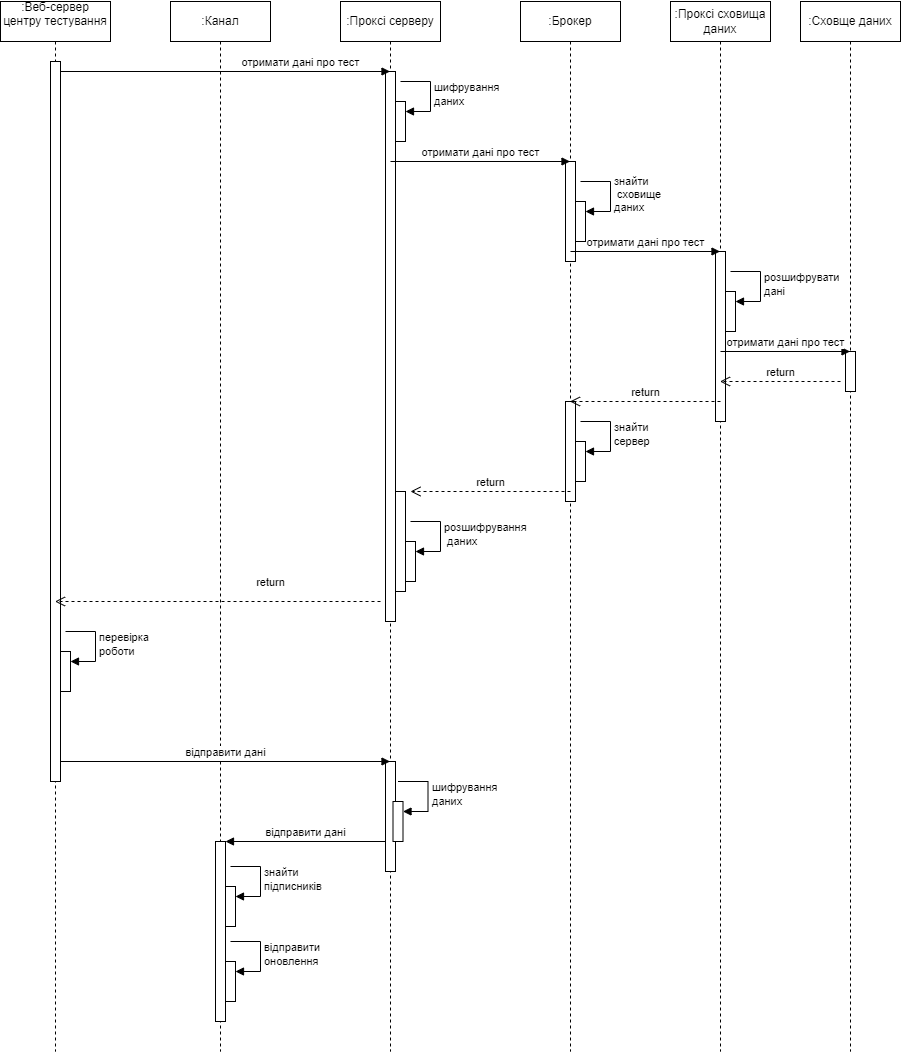


Рисунок 4.3 - забезпечення функціоналу "Автоматична перевірка"

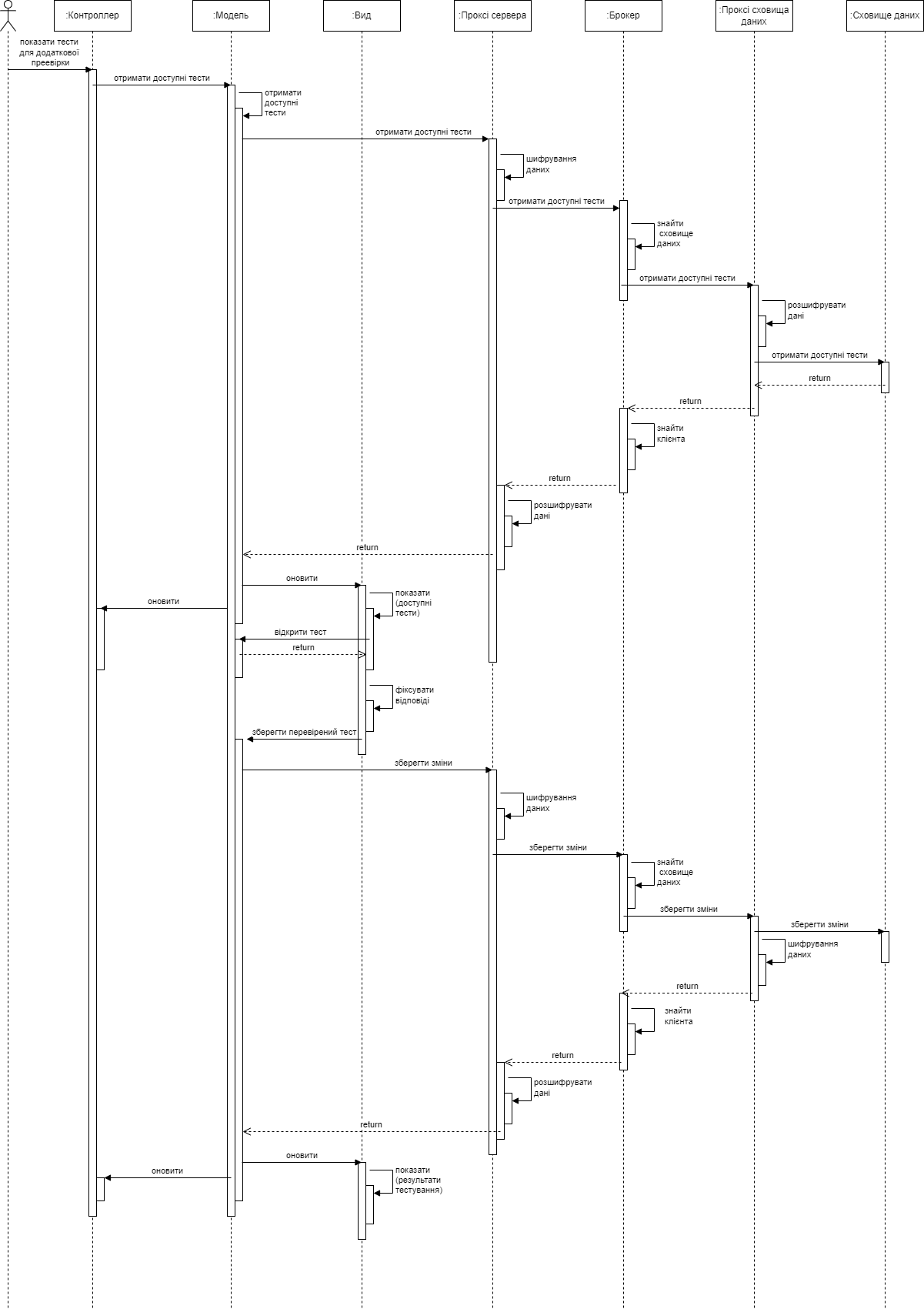
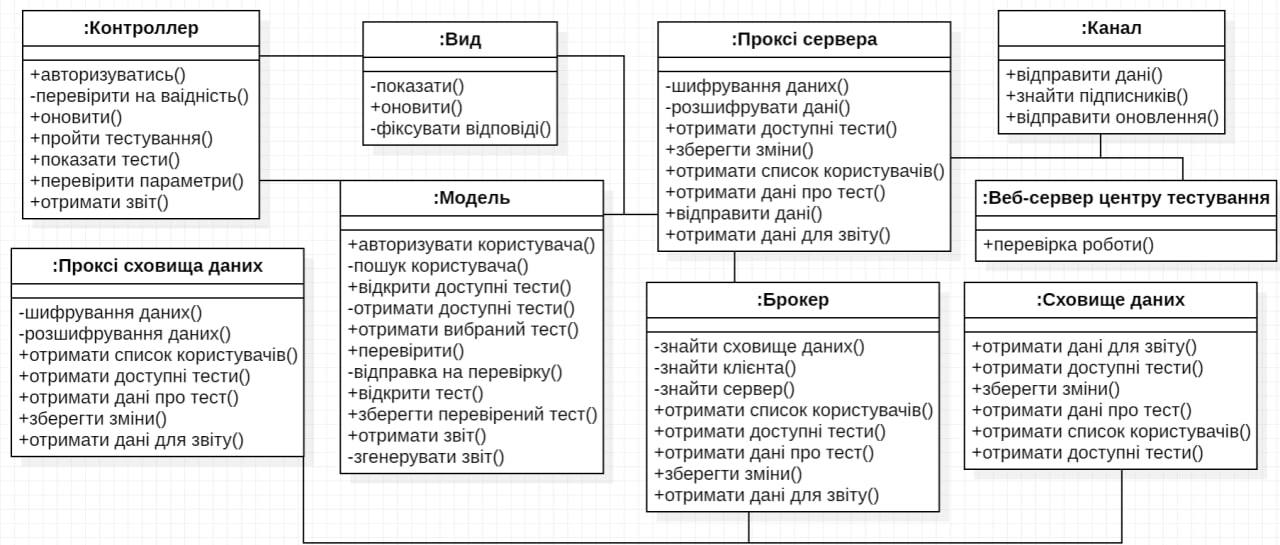


Рисунок 4.4 - забезпечення функціоналу "Ручна перевірка"

## **4.2 Логічне подання із застосуванням шаблонів та стилю**

Логічне подання дає побачити чіткіше, як система декомпозована на окремі компоненти. На діаграмі(рис. 4.5) зображені компоненти, які відповідають декомпозиції системи за стилем MVC та шаблонами.



віаіфаіваівфаіфваіва