МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний аерокосмічний університет ім. М. Є. Жуковського

«Харківський авіаційний інститут»

Факультет програмної інженерії та бізнесу

Кафедра інженерії програмного забезпечення

**КУРСОВА РОБОТА**

з курсу: «Об’єктно-орієнтоване програмування»

на тему: «Розроблення програмного забезпечення з використанням

об’єктно-орієнтованої парадигми»

Виконав: студент (ка) 2 курсу групи 621п

Спеціальність 121 – Інженерія програмного забезпечення

(код та найменування)

Овчаров А. В.

(прізвище й ініціали студента(ки))

Керівник: Лучшев П. О.

(прізвище й ініціали)

Національна шкала \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кількість балів: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Оцінка: ECTS \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Члени комісії \_\_\_\_\_\_\_\_ Лучшев П.О.

(підпис) (прізвище та ініціали)

\_\_\_\_\_\_\_\_ Вдовітченко О.В.

(підпис) (прізвище та ініціали)

Харків – 2023

**ТИПОВЕ ЗАВДАННЯ НА КУРСОВИЙ ПРОЄКТ**

Варіант 14. Курси дистанційного навчання

|  |  |  |
| --- | --- | --- |
| **Задачі** | **Бали** | **Штрафи5)** |
| Розділ 1. Функціональні вимоги до програмного забезпечення (термін – **3 тиждень**) | 0..15 | -3 |
| Розділ 2. Моделювання предметної області(термін – **6 тиждень**) | 0..15 | -3 |
| Розділ 3.Програмна реалізація класів предметної області і їх тестування(термін – **9 тиждень**) | 0..15 | -3 |
| Розділ 4. Програмна реалізація класів інтерфейсу користувача (термін – **12 тиждень**) | 0..15 | -3 |
| Розділ 5.Функціональне тестування програмного забезпечення(термін – **13 тиждень**) | 0..5 | -1 |
| Оформлення пояснювальної записки (термін – **14 тиждень**) | 0..5 | -1 |
| **Захист курсової роботи** |  |  |
| Оформлення презентації (термін – **15 тиждень**) | 0..5 6) |  |
| Оформлення відеоролику роботи програми (термін – **15 тиждень**) | 0..5 6) |  |
| Використання:   1. інкапсуляції, 2. статичних членів класів, 3. інтерфейсів, 4. абстрактних класів, 5. спадкування, 6. поліморфізму, 7. .NET-delegates і events, 8. сереалізації/десереалізації, 9. generic collections, 10. LINQ | 0..20 6) |  |
| **Всього за курсову роботу** | **0..100** | **0..14** |

5*)* Штрафні (від’ємні) бали нараховуються за порушення зазначених у таблиці «Система оцінювання» крайніх термінів здачі.

6) Бали, які студент отримує під час захисту роботи.

Процедура **захисту** курсової роботи перед комісією відбувається на **16-му тижні.**

Зміст

[1 ФУНКЦІОНАЛЬНІ ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 7](#_Toc134844349)

[1.1 Детальний опис заданої предметної області 7](#_Toc134844350)

[1.2 Виділення ролей користувачів і формулювання функціональних вимог 8](#_Toc134844351)

[1.3 Проєктування інтерфейсу користувача 9](#_Toc134844355)

[1.4 Розроблення функціональних тестів 13](#_Toc134844356)

[1.4.1 Функціональні тести для ролі користувача «Адміністратор» 13](#_Toc134844357)

[1.4.2 Функціональні тести для ролі користувача «Вчитель» 14](#_Toc134844358)

[1.4.3 Функціональні тести для ролі користувача «Ученик» 15](#_Toc134844359)

[2 МОДЕЛЮВАННЯ ПРЕДМЕТНОЇ ОБЛАСТІ 17](#_Toc134844360)

[2.1 Виділення і опис класів предметної області 17](#_Toc134844361)

[2.2 Встановлення зв’язків між класами 22](#_Toc134844362)

[3 ПРОГРАМНА РЕАЛІЗАЦІЯ КЛАСІВ ПРЕДМЕТНОЇ ОБЛАСТІ І ЇХ ТЕСТУВАННЯ 24](#_Toc134844363)

[3.1 Структура проєкту з реалізацією класів предметної області 24](#_Toc134844364)

[3.2 Реалізація інтерфейсів і каркасів класів предметної області 24](#_Toc134844365)

[3.3 Розроблення unit-тестів для класів предметної області 27](#_Toc134844366)

[3.4 Повна реалізація класів предметної області 29](#_Toc134844367)

[3.5 Результати unit-тестування класів предметної області 29](#_Toc134844368)

Зміст

**СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ**

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
| ІТ-технології | – | це система методів, процесів та способів використання обчислювальної техніки і систем зв'язку для створення, збору, передачі, пошуку, оброблення та поширення інформації з метою ефективної організації діяльності людей |

**ВСТУП**

Розвиток інтернет технологій у сучасних умовах дозволяє використовувати їх у різних галузях діяльності, включаючи освіту. Онлайн-освіта є альтернативою класичному способу навчання, а саме очному. Завдяки розвитку IT-технологій [2], онлайн-навчання дозволяє легко вибрати не тільки зручний час та місце для навчання, а й власний темп, легко зв'язатися з викладачем за допомогою чату, пошти, форуму або аудіо-відео зв'язку на платформі для дистанційного навчання. Багатьом сором'язливим людям легше наважитися навчатися на онлайн-курсах. Їм навіть не обов'язково скористатися камерою для розмови з викладачем. Розробка сервісу для курсів дистанційного навчання може бути дуже хорошою ідеєю.

Основними перевагами такого сервісу є:

* Гнучкість та доступність: студенти можуть навчатися у зручний для себе час та місце, без необхідності відвідувати фізичні аудиторії.
* Розширення аудиторії: такий сервіс може бути доступним для студентів з будь-якої точки світу, що дозволяє розширити аудиторію та залучити більше студентів.
* Економія часу та коштів: студенти не потребують витрачати час на дорогу до університету, що дозволяє їм економити час та кошти.
* Актуальність матеріалів: сервіс може легко оновлювати матеріали та курси, щоб відповідати сучасним вимогам та потребам ринку.

# ФУНКЦІОНАЛЬНІ ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

## 1.1 Детальний опис заданої предметної області

Сервіс матиме два типи користувачів:

* Учень
* Викладач

Викладачі зможуть розміщувати інформацію про свої курси та про себе, таку як:

1. ФІО, та контактна інформація (телефон, Telegram), свої досягнення(досвід роботи, про себе(нагороди, спосіб викладання, тощо)
2. Назва, напрям курсу, його довжина(в годинах), посилання на матеріали, ціна.

Учні матимуть змогу переглянути список викладачів, та їх курсів. Вибрати кращий для них варіант. Учень матиме змогу поставили оцінку викладачу, яку буде видно в його профілі. Ще користувач “Учень” зможе бачити на які курси він записаний і зможе редагувати це.

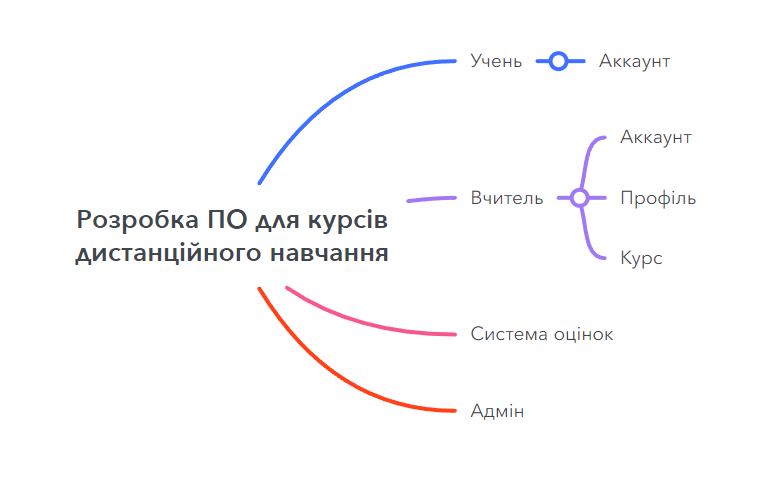


Рисунок 1.1 – Mind Map

## Виділення ролей користувачів і формулювання функціональних вимог

Були виділені три ролі користувачів: «Адміністратор», «Вчитель», «Учень».

Всі ролі обов’язково реєстровані та авторизовані.

Опишемо функціональні вимоги кожного типу користувачів.

Таблиця 1.1 – Функціональні вимоги користувача з роллю «Адміністратор»

|  |  |
| --- | --- |
| **Іденти-фікатор** | **Функціональні вимоги** |
| А.1. | Адміністратор може авторизуватись. |
| А.2. | Адміністратор може видалити курс. |
| А.3. | Адміністратор може заблокувати вчителя. |
| А.4. | Адміністратор може заблокувати учня. |

Таблиця 1.2 – Функціональні вимоги користувача з роллю «Вчитель»

|  |  |
| --- | --- |
| **Іденти-фікатор** | **Функціональні вимоги** |
| В.1. | Вчитель може зареєструватися  ***Примітка:***   1. довжина логіну більше 3х символів 2. не можна зареєструвати два однакових логіна 3. довжина паролю більше 8ми символів 4. телефон має бути в форматі 0ХХ-ХХХ-ХХ-ХХ 5. тег Telegram має бути в форматі @nickname 6. про себе не довше 300 символів |
| В.2. | Вчитель може авторизуватися.  ***Примітка:***  не можна авторизуватися з невірним логіном і паролем. |
| В.3 | Вчитель може додати/редагувати/видалити інформацію про курс.  ***Примітка:***   1. Назва обов'язково 2. Напрям курсу обов'язково 3. ціна обов’язково 4. додатково не довше 300 символів |

Таблиця 1.3 – Функціональні вимоги користувача з роллю «Учень»

|  |  |
| --- | --- |
| **Іденти-фікатор** | **Функціональні вимоги** |
| У.1. | Учень може зареєструватися  ***Примітка:*** довжина логіну більше 3 символівне можна зареєструвати два однакових логінадовжина паролю більше 8 символів |
| У.2. | Клієнт може авторизуватися.  ***Примітка:***  не можна авторизуватися з невірним логіном і паролем. |
| У.3. | Учень може переглядати інформацію про Вчителя |
| У.4. | Учень може записуватись на курс |
| У.5. | Учень може ставити оцінку курсу |

## Проєктування інтерфейсу користувача

Для реалізації замовником було обрано *desktop*-застосунок.

На основі сформульованих функціональних вимог були розроблені екранні форми, які наведені на рис. 1.2 – 1.4).

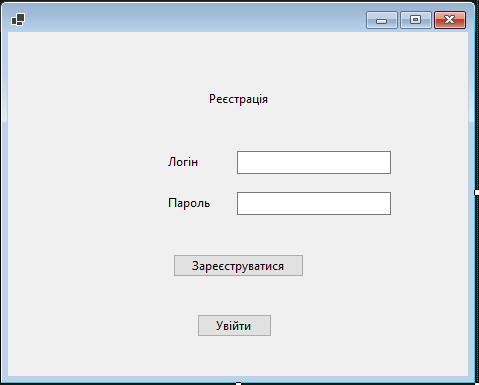


Рисунок 1.2 – Екранна форма реєстрації

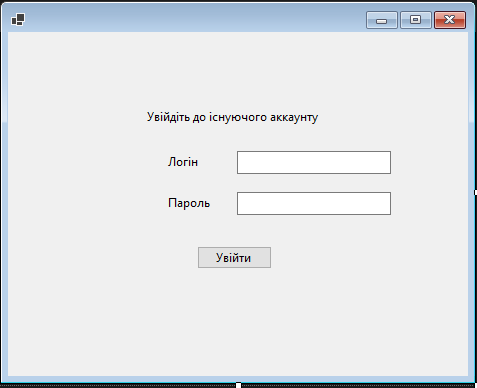


Рисунок 1.3 – Екранна форма входу

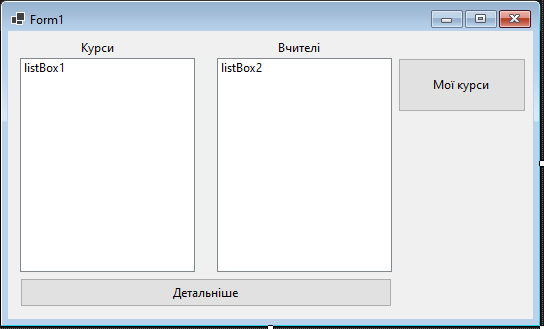


Рисунок 1.4 – Екранна форма учня

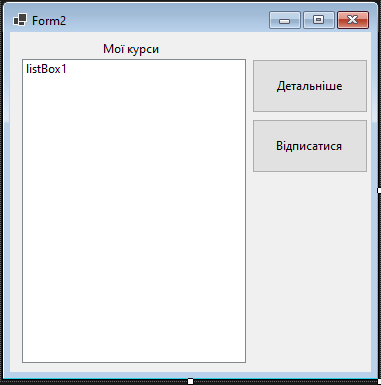


Рисунок 1.5 – Екранна форма списку “Мої курси”

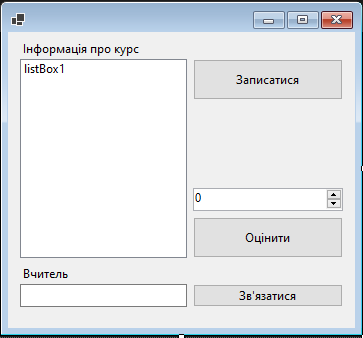


Рисунок 1.5 – Екранна форма інформація про курс

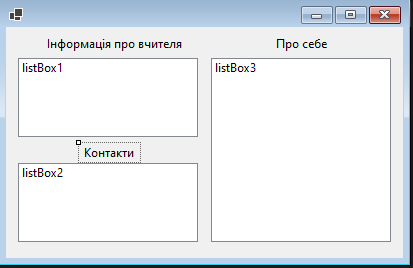


Рисунок 1.6 – Екранна форма інформація про вчителя

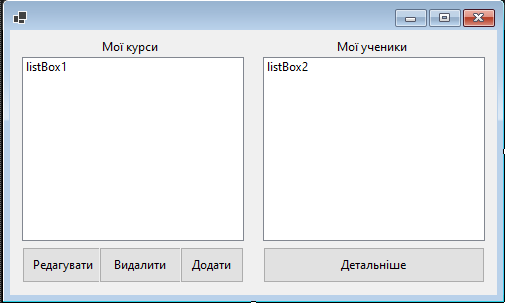


Рисунок 1.7 – Екранна форма Вчителя

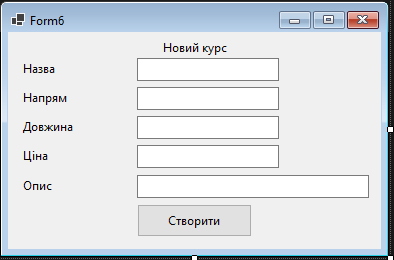


Рисунок 1.7 – Екранна форма Вчителя додавання курсу

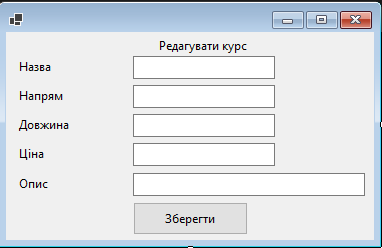


Рисунок 1.7 – Екранна форма Вчителя редагування курсу

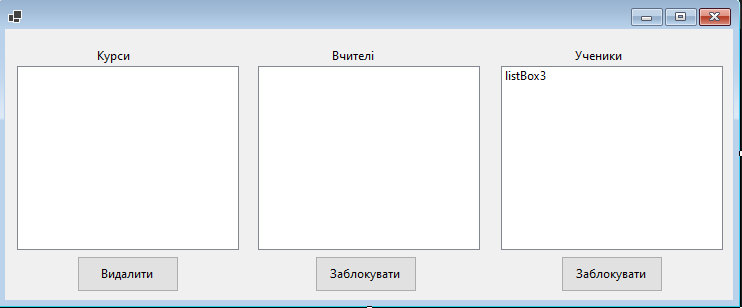


Рисунок 1.10 – Екранна форма адміністратора

## Розроблення функціональних тестів

### Функціональні тести для ролі користувача «Адміністратор»

На основі опису предметної області, а також функціональних вимог і екранних форм були розроблені функціональні тести для ролі користувача «Адміністратор» (див. табл. 1.4).

Таблиця 1.4 – Функціональні тести для ролі користувача «Адміністратор»

|  |  |  |
| --- | --- | --- |
| **Іденти-фікатор тесту** | **Послідовність дій користувача** | **Очікуваний результат** |
| *А.1. Адміністратор може авторизуатись.* | | |
| А.1.2 | 1.ввести логін  2.ввести пароль  3.натиснути “увійти” | Успішна авторизація |
| *А.2. Адміністратор може заблокувати Вчителя* | | |
| А.2.2 | 1.обрати Вчителя  2.натиснути “заблокувати” | Успішне блокування |
| *А.3. Адміністратор може видалити курс* | | |
| А.3.2 | 1.обрати курс  2.натиснути “видалити” | Успішне видалення курсу |
| *А.4. Адміністратор може заблокувати Учня* | | |
| А.4.2 | 1.обрати Учня  2.натиснути “заблокувати” | Успішне блокування |

### Функціональні тести для ролі користувача «Вчитель»

На основі опису предметної області, а також функціональних вимог і екранних форм були розроблені функціональні тести для ролі користувача «Вчитель» (див. табл. 1.5).

Таблиця 1.5 – Функціональні тести для ролі користувача «Вчитель»

|  |  |  |
| --- | --- | --- |
| **Іденти-фікатор**  **тесту** | **Послідовність дій користувача** | **Очікуваний результат** |
| *В.1. Вчитель може зареєструватись.* | | |
| В.1.2 | 1.ввести логін  2.ввести пароль  3.натиснути “зареєструватися” | Успішно створено аккаунт Вчитель |
| *В.2. Вчитель може авторизуатись.* | | |
| В.2.2 | 1.ввести логін  2.ввести пароль  3.натиснути “увійти” | Успішна авторизація |
| *В.3. Вчитель може додати курс* | | |
| В.3.2 | 1.натиснути “додати”  2.ввести дані  3.натиснути “створити” | Успішне додавання курсу |
| *В.4. Вчитель може редагувати курс* | | |
| В.4.2 | 1.обрати курс  2.натиснути “редагувати”  3.ввести дані  4.натиснути “зберегти” | Успішне редагування курсу |
| *В.5. Вчитель може видалити курс* | | |
| В.5.2 | 1.обрати курс  2.натиснути “видалити” | Успішне видалення курсу |
| *В.6. Вчитель може переглянути деталі профілю ученика* | | |
| В.6.2 | 1.обрати ученика  2.натиснути “детальніше” | Отримано інформацію про ученика |

### Функціональні тести для ролі користувача «Ученик»

На основі опису предметної області, а також функціональних вимог і екранних форм були розроблені функціональні тести для ролі користувача «Ученик» (див. табл. 1.6).

Таблиця 1.6 – Функціональні тести для ролі користувача «Ученик»

|  |  |  |
| --- | --- | --- |
| **Іденти-фікатор**  **тесту** | **Послідовність дій користувача** | **Очікуваний результат** |
| У*.1. Ученик може зареєструватись.* | | |
| У.1.2 | 1.ввести логін  2.ввести пароль  3.натиснути “зареєструватися” | Успішно створено аккаунт *Ученик* |
| *У.2. Ученик може авторизуатись.* | | |
| У.2.2 | 1.ввести логін  2.ввести пароль  3.натиснути “увійти” | Успішна авторизація |
| *У.3. Ученик може переглянути детальну інформацію про вчителя.* | | |
| У.3.2 | 1.обрати вчителя  2.натиснути “детальніше” | Отримана інформація про учителя |
| *У.4. Ученик може переглянути детальну інформацію про курс.* | | |
| У.4.2 | 1.обрати курс  2.натиснути “детальніше” | Отримана інформація про курс |
| *У.5. Ученик може записатися на курс.* | | |
| У.5.2 | 1.натиснути записатися | Успішно додано до списку *“*мої курси*”* |
| *У.6. Ученик може оцінити курс.* | | |
| У.6.2 | 1.обрати оцінку  2. натиснути “оцінити” | Успішно оцінено |
| *У.7. Ученик може отримати інформацію про вчителя.* | | |
| У.7.2 | 1.натиснути “улюблене”  2.обрати мережу АЗС  3.натиснути “видалити” | Отримано інформацію про вчителя |
| У.7.3 | 1.обрати вчителя  2.натиснути “детальніше” | Отримано інформацію про вчителя |
| *У.8. Ученик може відписатися від курсу* | | |
| У.8.2 | 1.обрати курс  2. натиснути “відписатися” | Успішно додано до списку *“*мої курси*”* |

# МОДЕЛЮВАННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

## Виділення і опис класів предметної області

**Клас «User» – користувач**

|  |  |  |
| --- | --- | --- |
| **Назва характеристики** | **Тип значення** | **Обмеження** |
| Логін | Рядок | Не пустий рядок, довжина більше 3 символів |
| Пароль | Рядок | Не пустий рядок, довжина більше 8 символів |

Таблиця 1.1 – Виділення характеристик класу «User»

|  |  |  |  |
| --- | --- | --- | --- |
| **Назва поведінки** | **Опис вхідних параметрів** | **Тип значення, що повертається** | **Бізнес-правила** |
| Авторизація | Логін, пароль | true/false – успішно авторизовано / невірний логін або пароль | 1. Перевірити чи вірні вхідні дані. 2. Якщо вхідні дані вірні, то повернути true, інакше - false. |

Таблиця 1.2 – Виділення поведінки класу «User »

**Клас «Admin» –адміністратор**

Таблиця 2.1 – Виділення характеристик класу «Admin»

|  |  |  |
| --- | --- | --- |
| **Назва характеристики** | **Тип значення** | **Обмеження** |
| Логін | Рядок | Не пустий рядок, довжина більше 3 символів |
| Пароль | Рядок | Не пустий рядок, довжина більше 8 символів |

Таблиця 2.2 – Виділення поведінки класу «Admin»

|  |  |  |  |
| --- | --- | --- | --- |
| **Назва поведінки** | **Опис вхідних параметрів** | **Тип значення, що повертається** | **Бізнес-правила** |
| Видалити Курс | Назва курсу | true/false – успішно видалено / помилка при видаленні. | 1. Перевірити чи існує курс. 2. Якщо курс існує - видалити. |
| Заблокувати Учителя | Ім’я Учителя | true/false – успішно заблоковано/ помилка при блокуванні. | 1. Перевірити чи існує Учитель. 2. Якщо Учитель існує - видалити. |
| Заблокувати Учителя | Ім’я Ученика | true/false – успішно заблоковано/ помилка при блокуванні. | 1. Перевірити чи існує Учень. 2. Якщо Учень існує - видалити. |

**Клас «Teacher» – вчитель**

Таблиця 3.1 – Виділення характеристик класу «Teacher»

|  |  |  |
| --- | --- | --- |
| **Назва характеристики** | **Тип значення** | **Обмеження** |
| Логін | Рядок | Не пустий рядок, довжина більше 3 символів |
| Пароль | Рядок | Не пустий рядок, довжина більше 8 символів |

Таблиця 3.2 – Виділення поведінки класу «Teacher»

|  |  |  |  |
| --- | --- | --- | --- |
| **Назва поведінки** | **Опис вхідних параметрів** | **Тип значення, що повертається** | **Примітка** |
| Додати Курс | Назва, напрям курсу, довжина курсу, ціна і опис | true/false – курс додано / помилка при додаванні курсу; | 1. Перевірити чи дійсні вхідні дані. 2. Якщо пройдені перевірки, то додати мережу. |
| Видалити Курс | Назва курсу | true/false – успішно видалено / помилка при видаленні. | 1. Перевірити чи існує курс. 2. Якщо курс існує - видалити. |
| Редагувати Курс | Назва, напрям курсу, довжина курсу, ціна і опис | true/false – дані оновлено / помилка при оновленні даних; | 1. Перевірити чи дійсні вхідні дані. 2. Якщо пройдені перевірки, то оновити дані. |

**Клас «Student» – ученик**

Таблиця 4.1 – Виділення характеристик класу «Student»

|  |  |  |
| --- | --- | --- |
| **Назва характеристики** | **Тип значення** | **Обмеження** |
| Логін | Рядок | Не пустий рядок, довжина більше 3 символів |
| Пароль | Рядок | Не пустий рядок, довжина більше 8 символів |

Таблиця 4.2 – Виділення характеристик класу «Student»

|  |  |  |  |
| --- | --- | --- | --- |
| **Назва поведінки** | **Опис вхідних параметрів** | **Тип значення, що повертається** | **Бізнес-правила** |
| Поставити оцінку курсу | Назва курсу,  оцінка | true/false – оцінку додано/ помилка при додаванні оцінки; | 1. Перевірити чи дійсні вхідні дані. 2. Якщо пройдені перевірки, то додати оцінку. |
| Записатися на курс | Назва курсу | true/false – успішно додано курс/помилка при додаванні курсу. | 1. Перевірити чи існує курс. 2. Якщо курс існує - додати. |
| Покинути курс | Назва курсу | true/false – успішно покинуто/помилка при покиданні. | 1. Перевірити чи існує курс. 2. Якщо курс існує - відписатися. |

**Клас «Course» – курс**

Таблиця 5.1 – Виділення характеристик класу «Course»

|  |  |  |
| --- | --- | --- |
| **Назва характеристики** | **Тип значення** | **Обмеження** |
| Назва | Рядок | Не пустий рядок, довжина: хоча б 1 символ |
| Напрям | Рядок | Не пустий рядок, довжина: хоча б 1 символ |
| Довжина | Рядок | Не пустий рядок, довжина: хоча б 1 символ |
| Ціна | Число з плаваючою точкою | Не пустий рядок, довжина: хоча б 2 символи |
| Опис | Рядок | Не пустий рядок, довжина: хоча б 1 символ, максимум 300 символів. |

Таблиця 5.2 – Виділення поведінки класу «Course»

|  |  |  |  |
| --- | --- | --- | --- |
| **Назва поведінки** | **Опис вхідних параметрів** | **Тип значення, що повертається** | **Примітка** |
| Обчислення оцінки | Оцінка | float/false - успішно обчисленно/помилка при обчисленні | 1. Перевірити чи дійсні вхідні дані. 2. Якщо пройдені перевірки, то провести обчислення. 3. Якщо обчислення пройшли вдало - повернути дані. |

**Клас "User"** (користувач) використовується для збереження інформації про користувача курсів дистанційного навчання. У цьому класі визначені такі характеристики:

* "Login" - рядок, що містить логін користувача. Мінімальна довжина логіну має бути більше 3 символів.
* "Password" - рядок, що містить пароль користувача. Мінімальна довжина паролю має бути більше 8 символів.

Також у класі "User" визначена поведінка "Authorize", яка використовується для перевірки вірності введеного логіну та паролю користувача. Цей метод приймає два вхідні параметри - логін та пароль, та повертає значення true, якщо введені дані відповідають збереженим у класі "User" значенням "Login" та "Password", і значення false, якщо введені дані невірні.

**Клас "Admin"** в нашій системі представляє адміністратора, який має повний доступ до управління курсами, учителями та учнями.

Характеристики класу "Admin":

* Логін - рядок, обмеження: не пустий рядок, довжина більше 3 символів;
* Пароль - рядок, обмеження: не пустий рядок, довжина більше 8 символів.

Поведінка класу "Admin":

**Видалити курс** - вхідні параметри: назва курсу, тип значення, що повертається: true/false - успішно видалено/помилка при видаленні. Бізнес-правило: перевірити, чи існує курс, якщо курс існує, то його видалити;

**Заблокувати учителя** - вхідні параметри: ім'я учителя, тип значення, що повертається: true/false - успішно заблоковано/помилка при блокуванні. Бізнес-правило: перевірити, чи існує учитель, якщо учитель існує, то його заблокувати;

**Заблокувати учня** - вхідні параметри: ім'я учня, тип значення, що повертається: true/false - успішно заблоковано/помилка при блокуванні. Бізнес-правило: перевірити, чи існує учень, якщо учень існує, то його заблокувати.

**Клас «Student»** описує учня і містить наступні характеристики:

* Логін: рядок, що вказує на ім'я користувача, не може бути порожнім і повинен містити більше 3 символів.
* Пароль: рядок, що вказує на пароль користувача, не може бути порожнім і повинен містити більше 8 символів.

Клас містить такі методи:

**Поставити оцінку курсу** - приймає назву курсу та оцінку, перевіряє чи є такий курс та чи є валідна оцінка, якщо так - додає оцінку до списку оцінок для цього курсу і повертає true, якщо ні - повертає false.

**Записатися на курс** - приймає назву курсу, перевіряє чи існує такий курс, якщо так - додає цей курс до списку курсів, на які записаний учень і повертає true, якщо ні - повертає false.

**Покинути курс** - приймає назву курсу, перевіряє чи є такий курс в списку курсів, на які записаний учень, якщо так - видаляє цей курс зі списку і повертає true, якщо ні - повертає false.

**Клас «Course»** представляє курс, який може бути доступний для студентів. У даному класі можна визначити наступні характеристики:

* Назва: рядок, що відображає назву курсу. Значення не може бути порожнім та повинно містити хоча б один символ.
* Напрям: рядок, що відображає напрям курсу. Значення не може бути порожнім та повинно містити хоча б один символ.
* Довжина: рядок, що відображає тривалість курсу. Значення не може бути порожнім та повинно містити хоча б один символ.
* Ціна: число з плаваючою точкою, що відображає вартість курсу. Значення не може бути порожнім та повинно містити хоча б два символи.
* Опис: рядок, що відображає опис курсу. Значення не може бути порожнім, повинно містити хоча б один символ і не може перевищувати 300 символів.

Крім того, клас має такі поведінки:

**Обчислення оцінки**: приймає оцінку, яку студент отримав за курс, і обчислює її середню оцінку. Повертає обчислену оцінку або false, якщо обчислення було неможливим. Перед обчисленням перевіряється, щоб вхідні дані були дійсними.

## 2.2 Встановлення зв’язків між класами

На рисунку 2.1 наведена діаграма класів, яка була побудова з використанням сервісу <https://www.lucidchart.com>.

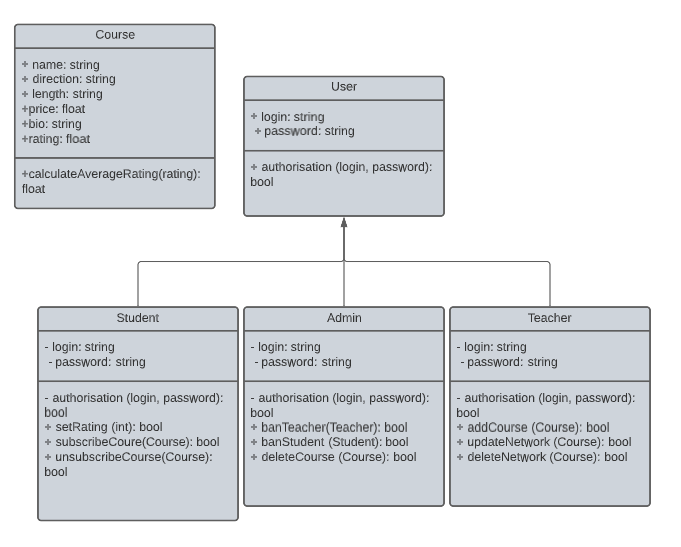


Рисунок 2.1 – Діаграма класів …

Тож, було визначено такі зв’язки між класами:  
Класи **User**, **Teacher**, **Student** та **Admin** мають зв'язок успадкування (ієрархію класів), де клас **User** є базовим класом, а класи **Teacher**, **Student** та **Admin** успадковують властивості та методи класу **User**, а також мають власні унікальні властивості та методи, які визначають їх роль та можливості в системі.

Клас **Course** може має зв'язок агрегації з класами **Teacher** та **Student**, тому що кожен курс повинен мати відповідального викладача (об'єкт класу **Teacher**), який його веде, та список студентів (об'єктів класу **Student**), які зареєструвалися на цей курс.

Крім того, класи **Teacher** та **Admin** мають можливість створювати та редагувати курси (об'єкти класу Course), тоді як клас Student може зареєструватися на курс, покинути його та отримувати оцінки за курс.

Отже, зв'язки між класами можуть бути наступними:

* успадкування (ієрархія класів) між класами User, Teacher, Student та Admin;
* агрегація між класом Course та класами Teacher та Student;
* можливість створення та редагування курсів (об'єктів класу Course) класами Teacher та Admin;
* можливість реєстрації на курс, покидання його та отримання оцінок класом Student.

Класи **User** буде абстрактним, оскільки він не можуть бути безпосередньо створений, оскільки містять загальні характеристики та методи, які можуть бути успадковані класами Student, Admin, Teacher які розширюють їх функціонал.

У проекті можна виділити декілька власних інтерфейсів, які допомагають забезпечити чітке відокремлення різних компонентів системи і знизити залежність між ними:

* ICourse - це інтерфейс, який визначає функціональність Курсу. Він містить методи для отримання інформації про курс.
* IAccount - це інтерфейс, який визначає функціональність облікового запису користувача. Він містить методи для авторизації та реєстрації .
* IRating - це інтерфейс, який визначає функціональність оцінок користувачів. Він містить методи для додавання та отримання оцінок.
* ICourseManager - це інтерфейс, який визначає функціональність управління курсом. Він містить методи для додавання та видалення курсів, редагування інформації про курс та отримання списку курсів.

Крім власних інтерфейсів, в проекті використовуються також .NET-інтерфейси, такі як IEnumerable, IComparable, ICloneable тощо. Вони допомагають забезпечити стандартну функціональність, яка доступна у всіх проектах, що використовують .NET.

# 3 ПРОГРАМНА РЕАЛІЗАЦІЯ КЛАСІВ ПРЕДМЕТНОЇ ОБЛАСТІ І ЇХ ТЕСТУВАННЯ

## 3.1 Структура проєкту з реалізацією класів предметної області

OnlineCourses/

| |-- ClassFolder/

| | |-- User.cs

| | |-- Admin.cs

| | | --Student.cs  
 | | | -- Teacher.cs

| | | -- Course.cs

| | | -- CourseList.cs

| `-- ...

| |-- StaticMethod/

| | |-- CoursesMethod.cs

| `-- ...

`-- Program.cs

Рисунок 3.1 – Структура проєкту з реалізацією предметної області

## Реалізація інтерфейсів і каркасів класів предметної області

**Клас «User» – базовий клас користувача**

abstract class User

{

props{get; set;}

}

**Клас «Admin» – адміністратор**

public class Admin : User

{

props {get; set;}

ctor Admin(string login, string password, string role)  
{}

**Клас «Student» – учень**

public class Client : User

{

props {get; set;}

ctor Client(string login, string password, string eMail, string role, string name, string surname)

{}

public static void Rate(string name)

{}

}

**Клас «Teacher» – учитель**

public class Owner : User

{

props {get; set;}

ctor Owner(string login, string password, string eMail, string role, string name, string surname)

{}

}

**Клас «Course» – курс**

public class Course

{

props {get; set;}

ctor Course(string name, string subject, int length, double price, string bio, int teacherId)

{}

}

**Клас «CourseList» – курс**

public class CourseList  
{

props {get; set;}

ctor CourseList(int studentId)

public static void Serialise(List<CourseList> courses)

{}  
 public static List<CourseList> Deserialise()

{}

public static void Add(int id, string name)

{}

private static void RemoveCourseList(int id)

{}

public static void RemoveCourse(int id, string name)

{}

public static bool IsCourseInCourseList(int id, string name)

{}

}

**Клас «CourseList» – статичні методи проєкту**

public static void SerialiseUserList(List<User> userlist)

{}  
 public static List<User> DeserialiseUserList()

{}  
 public static void SerialiseCourseList(List<Course> userlist)

{}  
 public static List<Course> DeserialiseCourseList()

{}  
 public static bool IsLogin(string login)

{}  
 public static bool IsAuthenticated(string login, string password)

{}  
 public static int GetId(string login)

{}  
 public static string GetRole(string login)

{}  
 public static void DeleteCourse(string name)

{}

public static int GetCourseId(string name)

{}  
 public static void UpdateCourse(Course course)

{}

public static Course GetCourse(string name)

{}

public static User GetTeacher(string name, string surname)

{}  
 public static User GetStudent(string name, string surname)

{}

public static User GetUser(int id)

{}  
 public static CourseList GetList(int \_id)

{}

}

## Розроблення unit-тестів для класів предметної області

Розроблені unit-тести наведені у Додатку B.

|  |  |
| --- | --- |
| **Назва методу** | **Кількість розроблених тестів** |
| Constructor\_SetsPropertiesCorrectly | 6 |
| Constructor\_SetsUniqueId | 1 |
| Constructor\_SetsUniqueIdPositive | 1 |

Таблиця 3.1 – Покриття unit-тестами класу «User»

|  |  |
| --- | --- |
| **Назва методу** | **Кількість розроблених тестів** |
| Constructor\_SetsRoleToTeacher | 1 |
| Constructor\_SetsPropertiesInheritedFromUser | 5 |

Таблиця 3.2 – Покриття unit-тестами класу «Teacher»

|  |  |
| --- | --- |
| **Назва методу** | **Кількість розроблених тестів** |
| Constructor\_SetsRoleToStudent | 1 |
| Rate\_IncrementsCourseRating | 1 |

Таблиця 3.3 – Покриття unit-тестами класу «Student»

|  |  |
| --- | --- |
| **Назва методу** | **Кількість розроблених тестів** |
| Constructor\_SetsRoleToAdmin | 1 |
| Constructor\_SetsAdminLoginAndPassword | 2 |
| Constructor\_SetsPropertiesInheritedFromUser | 5 |

Таблиця 3.4 – Покриття unit-тестами класу «Admin»

|  |  |
| --- | --- |
| **Назва методу** | **Кількість розроблених тестів** |
| Constructor\_SetsRatingTo100 | 1 |

Таблиця 3.5 – Покриття unit-тестами класу «Course»

|  |  |
| --- | --- |
| **Назва методу** | **Кількість розроблених тестів** |
| Serialise\_Deserialise\_RoundTrip | 3 |
| Add\_AddsCourseToList\_IfAlreadyExists | 3 |
| RemoveCourse\_RemovesCourseFromList | 3 |
| IsCourseInCourseList\_ReturnsTrue\_IfCourseExists | 1 |

Таблиця 3.6 – Покриття unit-тестами класу «CourseList»

|  |  |
| --- | --- |
| **Назва методу** | **Кількість розроблених тестів** |
| SerialiseUserList\_SerializesUserListToFile | 14 |
| SerialiseCourseList\_WritesCourseListToFile | 2 |
| IsLogin\_ExistingLogin\_ReturnsTrue | 1 |
| IsAuthenticated\_ShouldReturnTrue\_WhenValidCredentialsAreProvided | 1 |
| IsAuthenticated\_ShouldReturnFalse\_WhenInvalidCredentialsAreProvided | 1 |
| GetUser\_WithValidId\_ReturnsUserObject | 2 |
| GetTeacher\_ReturnsCorrectUser\_WhenNameAndSurnameMatch | 5 |
| GetTeacher\_ReturnsNull\_WhenNoUserMatchesNameAndSurname | 1 |
| GetRole\_UserExists\_ReturnsRole | 1 |
| GetList\_ReturnsCorrectCourseList\_ForExistingStudent | 3 |
| GetCourse\_ReturnsCorrectCourse | 8 |
| DeserialiseUserList\_ShouldReturnListOfUsers | 8 |
| SerialiseUserList\_SerializesUsers\_Successfully | 2 |
| DeserialiseUserList\_DeserializesUsers\_Successfully | 1 |
| SerialiseCourseList\_SerializesCourses\_Successfully | 2 |
| DeserialiseCourseList\_DeserializesCourses\_Successfully | 1 |
| DeleteCourse\_RemovesCourseFromList | 2 |

Таблиця 3.7 – Покриття unit-тестами класу «CoursesMethod»

## Повна реалізація класів предметної області

У Додатку A наведено повну реалізацію програмного коду розроблених класів предметної області.

## 3.5 Результати unit-тестування класів предметної області

На рисунках 3.8 – 3.4 наведено скриншоти проходження розроблених unit-тестів для класів предметної області.

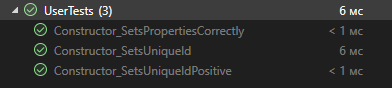


Рисунок 3.8 – Скриншот проходження unit-тестів для класу User



Рисунок 3.9 – Скриншот проходження unit-тестів для класу Teacher



Рисунок 3.10 – Скриншот проходження unit-тестів для класу Student

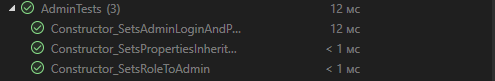


Рисунок 3.11 – Скриншот проходження unit-тестів для класу Admin



Рисунок 3.12 – Скриншот проходження unit-тестів для класу Course

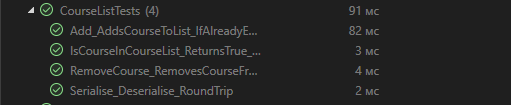
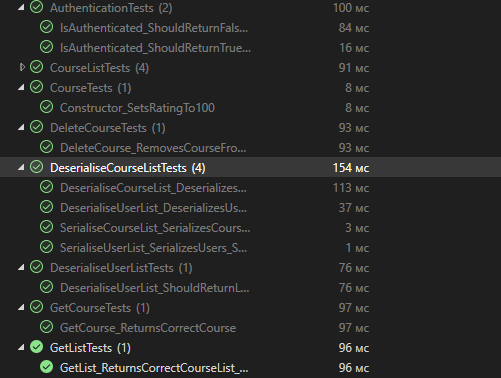


Рисунок 3.13 – Скриншот проходження unit-тестів для класу CourseList



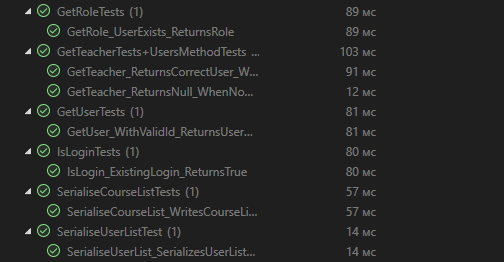


Рисунок 3.13 – Скриншот проходження unit-тестів для класу CoursesMethod