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 611пст
Спеціальність 121 – Інженерія програмного забезпечення
(код та найменування)
Хвостовця К.К.
(прізвище й ініціали студента(ки))
Керівник: Лучшев П.О.
(прізвище й ініціали)
Національна шкала _____
Кількість балів: _____
Оцінка: ECTS _____
Члени комісії _____ Шевченко І.В
(підпис) (прізвище та ініціали)
_____ Дем’яненко В.А.
(підпис) (прізвище та ініціали)

Харків – 2024

ТИПОВЕ ЗАВДАННЯ НА КУРСОВУ РОБОТУ

Варіант 18 Платні навчальні курси

Критерій оцінювання	Бали	Штрафи
Покрокове виконання курсової роботи:		
Розділ 1 (крайній термін – **3 тиждень**)	0..15	-3
Розділ 2 (крайній термін – **6 тиждень**)	0..15	-3
Розділ 3 (крайній термін – **9 тиждень**)	0..15	-3
Розділ 4 (крайній термін – **12 тиждень**)	0..15	-3
Розділ 5 (крайній термін – **13 тиждень**)	0..5	-1
Оформлення пояснювальної записки (термін – **14 тиждень**)	0..3	-1
Підготовка до захисту курсової роботи:		
доповідь з презентацією (термін – **15 тиждень**)	0..1	
відеоролик з демонстрацією роботи програми (термін – **15 тиждень**)	0..1	
Захист курсової роботи:		
доповідь з презентацією	0..5	
демонстрація роботи програми і github-репозиторію	0..5	
відповіді на питання	0..5	
використання*: 1) інкапсуляції 2) статичних членів класів 3) інтерфейсів 4) абстрактних класів 5) спадкування 6) поліморфізму 7) .NET-delegates і events 8) сереалізації/десереалізації об’єктів 9) Collections.Generic 10) LINQ * мають бути представленні на захисті курсової роботи у презентації	0..15	
Всього за курсову роботу:	**0..100**	**0..14**

Зміст

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ

CRUD	–	(*англ*. Create Read Update Delete) основні функції управління даними «створення, читання, оновлення і вилучення».
ООП	–	об’єктно-орієнтоване програмування
…		…

ВСТУП

Чому тема є актуальною?
Хто є потенційним користувачем?
Навести лідерів серед програмних аналогів.
Зазначити стек-технологій, який буде використаний.

У сучасному світі, навчання та розвиток навичок стали важливою складовою успіху в будь-якій сфері діяльності. З цим пов'язано зростання популярності платних навчальних курсів, що надають можливість здобути нові знання та вдосконалити навички в зручний для користувача спосіб. Ця тема є актуальною через постійний розвиток технологій та зміну вимог ринку праці, що ставлять перед людьми необхідність постійного самовдосконалення.

Потенційними користувачами платних навчальних курсів є люди з різних сфер діяльності та рівнів кваліфікації, які мають бажання вдосконалювати свої навички, отримувати нові знання або перекваліфікуватися для успішної кар'єри чи особистого розвитку.

Серед лідерів серед програмних аналогів можна виокремити такі платформи, як Udemy та Coursera, які надають доступ до великого вибору курсів у різних галузях знань та мають високу репутацію серед користувачів.

У даній курсовій роботі буде розглянуто стек технологій, що використовуються для створення та розвитку платних навчальних курсів, включаючи такі складові, як Windows Forms, мову програмування C# та інші інструменти, що забезпечують ефективне функціонування та взаємодію з користувачами.

У даній курсовій роботі буде розглянуто стек технологій, що використовуються для створення та розвитку платних навчальних курсів, включаючи такі складові, як веб-фреймворки, мови програмування, системи управління контентом та інші інструменти, що забезпечують ефективне функціонування та взаємодію з користувачами.

Не забуваємо наводити правильно оформлені посилання. Для ручного оформлення посилань звернутися до [1], для автоматичного – до [2] із «Перелік джерел і посилань».

1 ФУНКЦІОНАЛЬНІ ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Аналіз функціональності програмних аналогів

1.1.1 Udemy

Посилання на сервіс: https://www.udemy.com/

Udemy - це платформа з онлайн-навчання, де користувачі можуть знайти та купити курси з різних тематик, що варіюються від програмування до мистецтва. Кожен курс на Udemy створений і ведеться незалежним викладачем або експертом відповідної галузі. Платформа пропонує більше 150 000 курсів у різних мовах.

Сервіс дозволяє:

- користувачам придбати доступ до широкого спектру онлайн-курсів з різних областей знань.
- Кожен курс має відеоуроки, текстові матеріали та тести для оцінки знань.
- Користувачі можуть самостійно вибирати курси, вивчати матеріали у зручний для них час та темп, а після успішного завершення отримувати сертифікати.

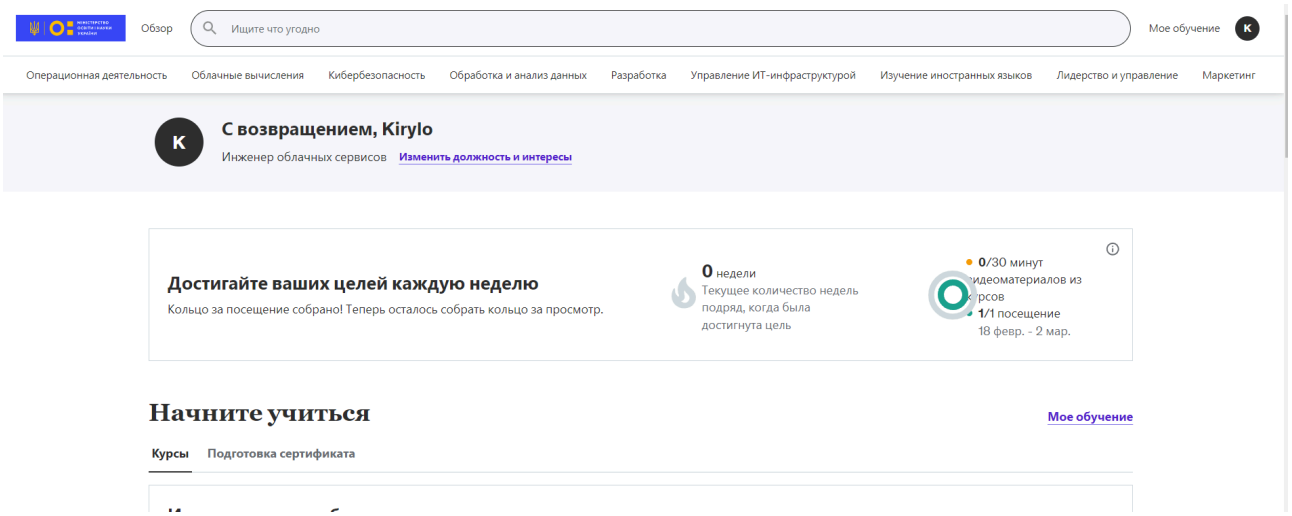

Рисунок 1.1 – Головна сторінка Udemy

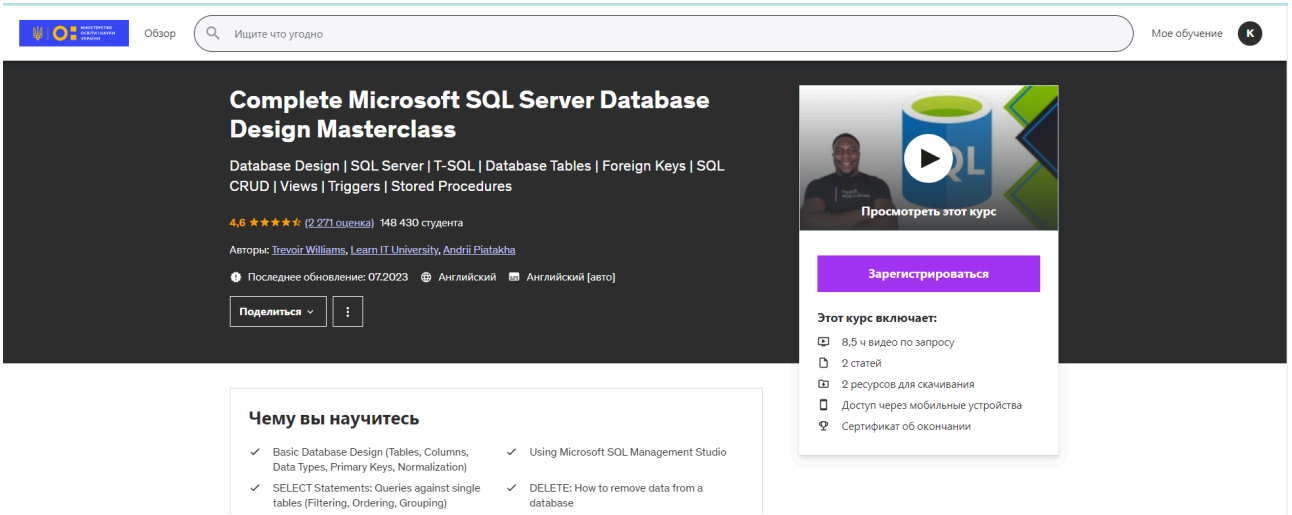

Рисунок 1.2 – Сторінка курсу у Udemy

1.1.2 Coursera

Посилання на сервіс: https://www.coursera.org/

Coursera - це платформа з онлайн-навчання, яка співпрацює з університетами та вищими навчальними закладами для надання курсів відомими викладачами та експертами. Курси на Coursera зазвичай розроблені університетами та інститутами світового класу, такими як Стенфордський, Єльський, та іншими. Платформа пропонує більше 4 000 курсів у різних галузях знань.

Сервіс дозволяє:

- користувачам отримувати доступ до курсів відомих університетів та коледжів з усього світу.
- Кожен курс має лекції, практичні завдання та тести для перевірки знань.
- Користувачі можуть вивчати матеріали у своєму темпі, отримувати сертифікати від університету та брати участь у спеціалізованих програмах для глибокого вивчення конкретних тематик.

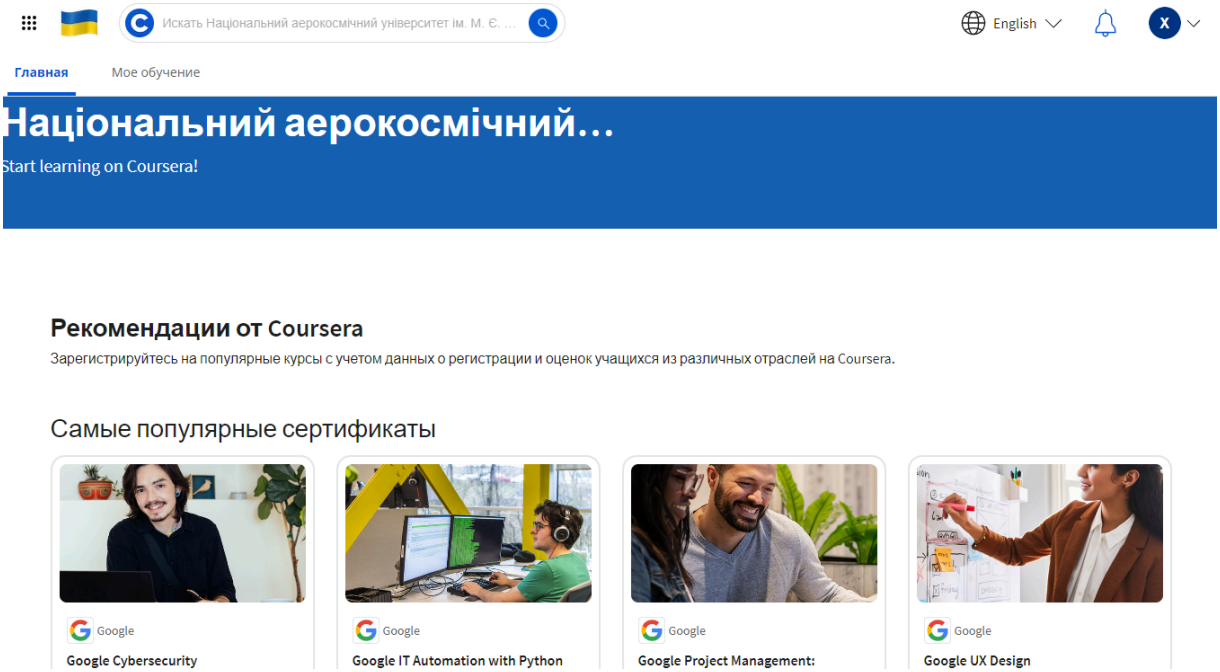

Рисунок 1.3 – Головна сторінка Coursera

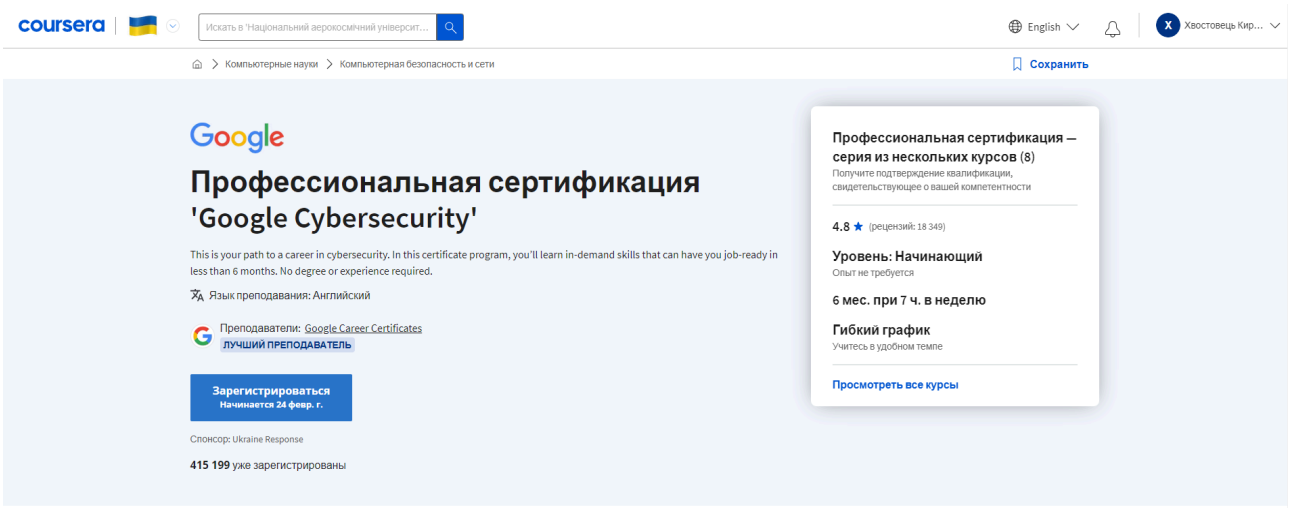

Рисунок 1.4 – Сторінка курсу у Coursera

1.2 Побудова Mind Map для заданої предметної області

Для систематизації інформації, побудуємо Mind Map для застосунку платних курсів Learnopia.

Mind Map була побудована за допомогою онлайн-інструментів MindMeister (https://www.mindmeister.com)

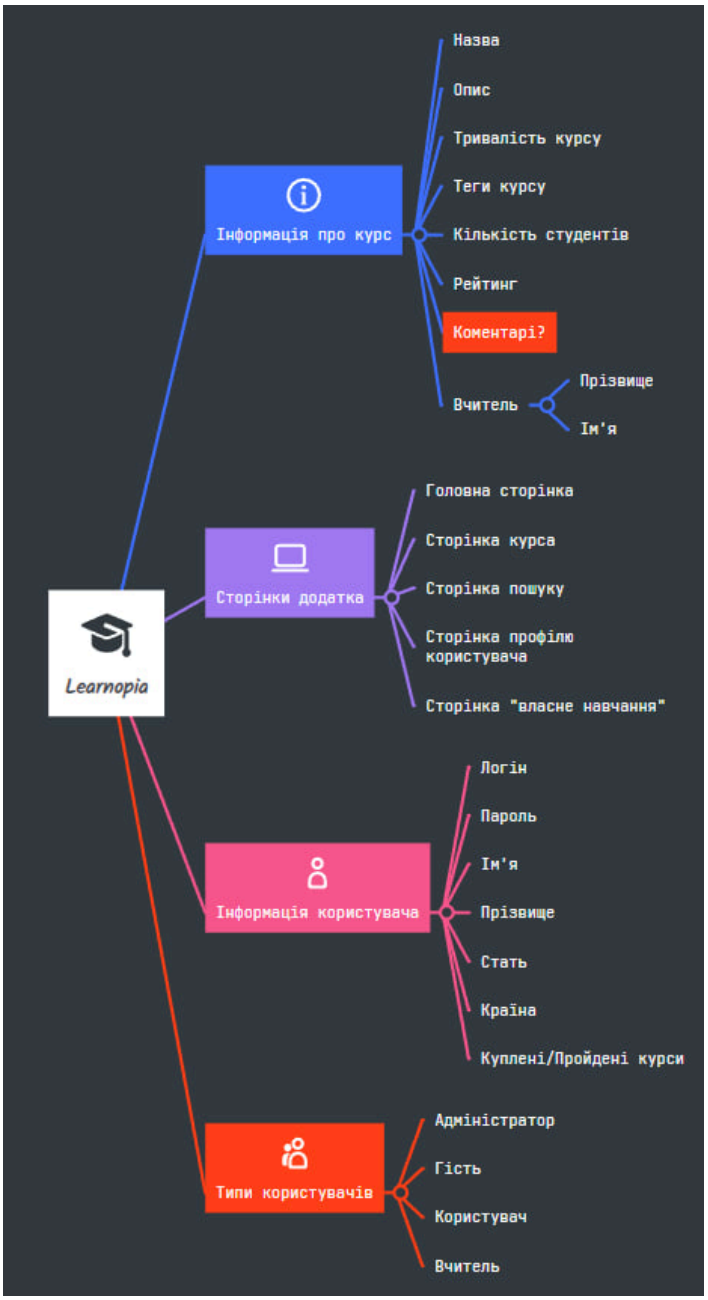

Рисунок 1.5 – Mind map платформи платних курсів Learnopia

Посилання на розроблений Mind map:
https://mm.tt/app/map/3174780711?t=aLYzuWDjnd

1.3 Виділення ролей користувачів і формулювання функціональних вимог

Були виділені три чотири користувачів: «Адміністратор», «Зареєстрований користувач», «Викладач », «Гість».

Опишемо функціональні вимоги кожного типу користувачів.

Таблиця 1.1 – Функціональні вимоги користувача з роллю «Адміністратор»

Ідентифікатор	Функціональні вимоги
А.1.	Адміністратор може додати/редагувати/видалити курси на платформі платних курсів. ***Примітка:*** якщо адміністратор додає курс, то у курса немає викладача
А.2.	Адміністратор може переглядати загальну статистику по усій платформі: активності користувачів, кількість і зареєстрованих користувачів, кількість придбаних курсів, кількість відвідувань тощо.
А.3.	Адміністратор може змінювати користувачів
А.4.	Можливість зберігати та завантажувати інформацію про курси та користувачів у форматі JSON для архівування або резервного копіювання

Таблиця 1.2 – Функціональні вимоги користувача з роллю «Зареєстрований користувач»

Ідентифікатор	Функціональні вимоги
ЗК.1.	Зареєстрований користувач може авторизуватися. ***Примітка:*** не можна авторизуватися з невірним логіном і паролем.
ЗК.2.	Зареєстрований користувач може переглядати курси.
ЗК.3.	Зареєстрований користувач може шукати курси.
ЗК.4.	Зареєстрований користувач може покупати курси ***Примітка:*** курс може бути безкоштовним.
ЗК.5.	Зареєстрований користувач може моніторити власний прогресс проходження курсів. ***Примітка:*** прогресс можна дивитися тільки у тих курсів, що проходяться
ЗК.6.	Зареєстрований користувач має особистотий профіль, який він може змінювати.
ЗК.7.	Зареєстрований користувач може оцінювати курси. ***Примітка:*** оцінювати можна тільки ті курси, що пройшов користувач

Таблиця 1.3 – Функціональні вимоги користувача з роллю «Гість»

Ідентифікатор	Функціональні вимоги
Г.1.	Гість може зареєструватися, вказавши логін і пароль. ***Примітка:*** 1) логін не має збігатися з існуючими; 2) довжина логіну не менше п’яти символів;

	3) довжина паролю не менше п’яти символів;
Г.2.	Гість може переглядати курси
Г.3.	Гість може шукати курси

Таблиця 1.4 – Функціональні вимоги користувача з роллю «Викладач»

Іденти-фікатор	**Функціональні вимоги**
В.1.	Викладач може зареєструватися, вказавши логін і пароль. ***Примітка:*** 1) логін не має збігатися з існуючими; 2) довжина логіну не менше п’яти символів; 3) довжина паролю не менше п’яти символів;
В.2.	Викладач може створити та редагувати свої курси
В.3.	Викладач може переглядати учасників курсів та їх прогресс проходження курсу ***Примітка:*** перегляд користувачів тільки власних курсів
В.4.	Можливість зберігати свої курси та інформацію про учасників у форматі JSON

1.4 Проєктування інтерфейсу користувача

Для реалізації замовником було обрано desktop-застосунок.

На основі сформульованих функціональних вимог були розроблені екранні форми, які наведені на рис. 1.6. – 1.11.

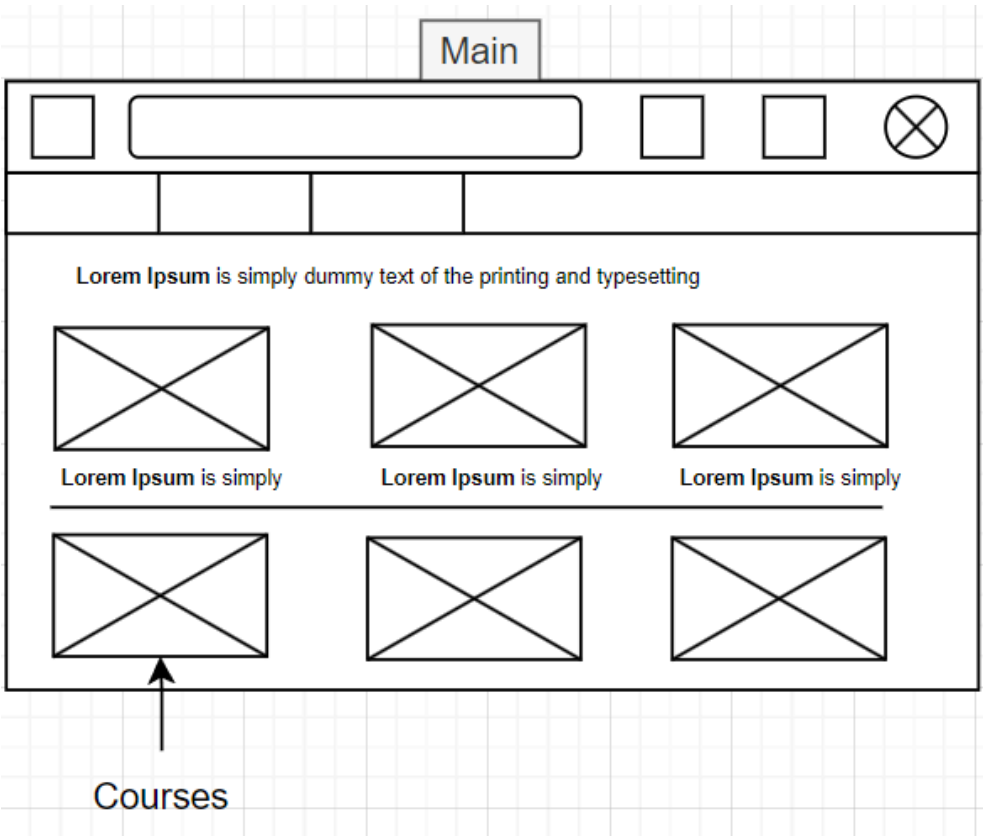

Рисунок 1.6 – Макет екранної форми «Головна сторінка»

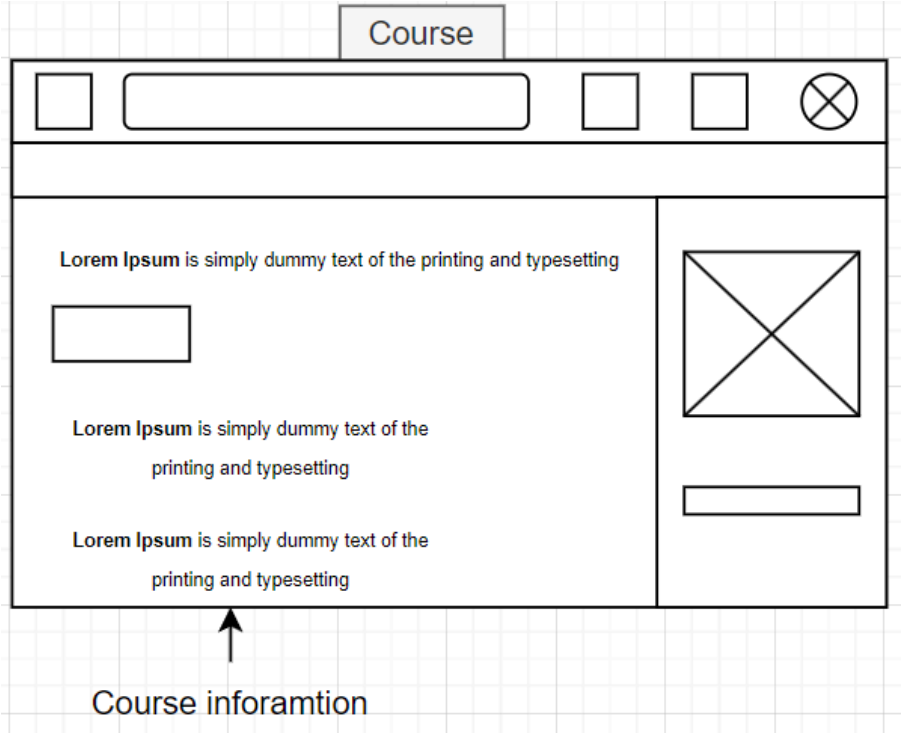

Рисунок 1.7 – Макет екранної форми «Сторінка курсу»

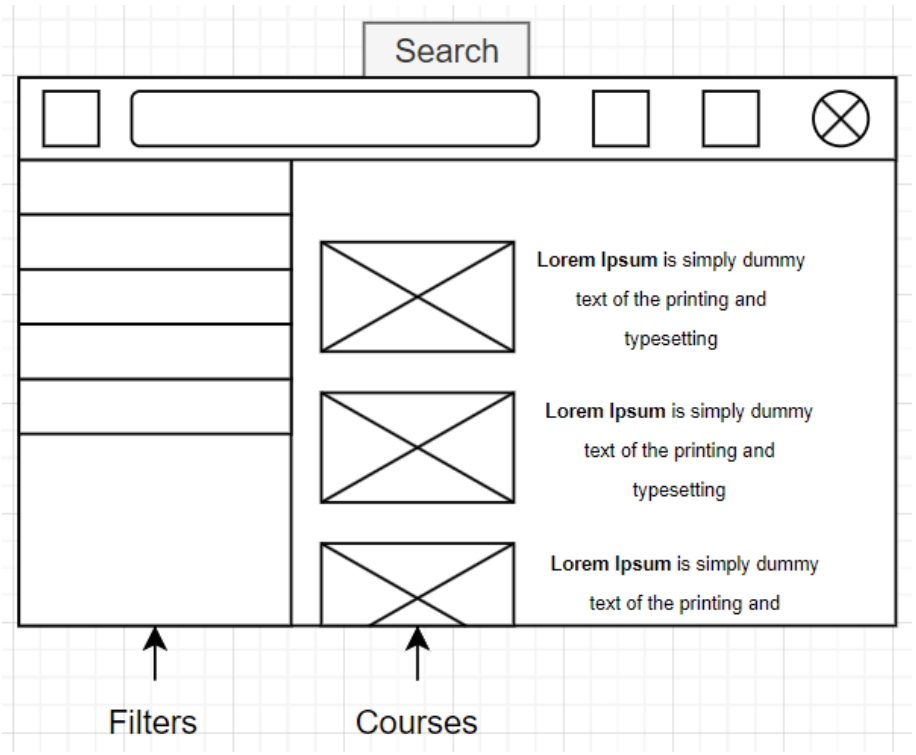

Рисунок 1.8 – Макет екранної форми «Пошук»

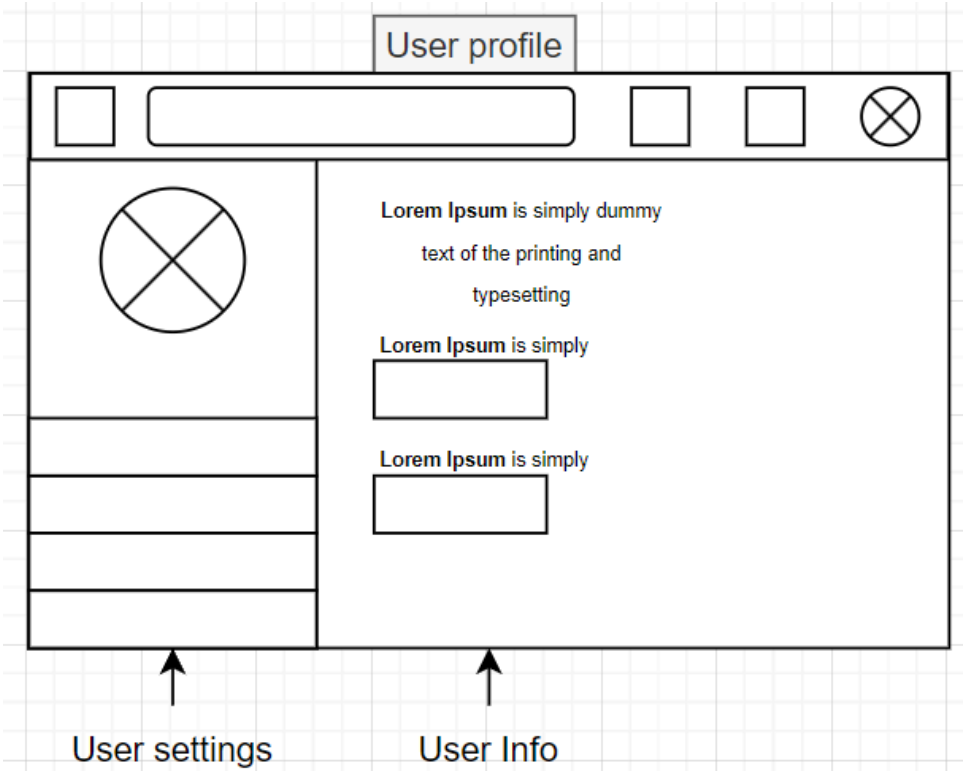

Рисунок 1.9 - Макет екранної форми «Профіль користувача»

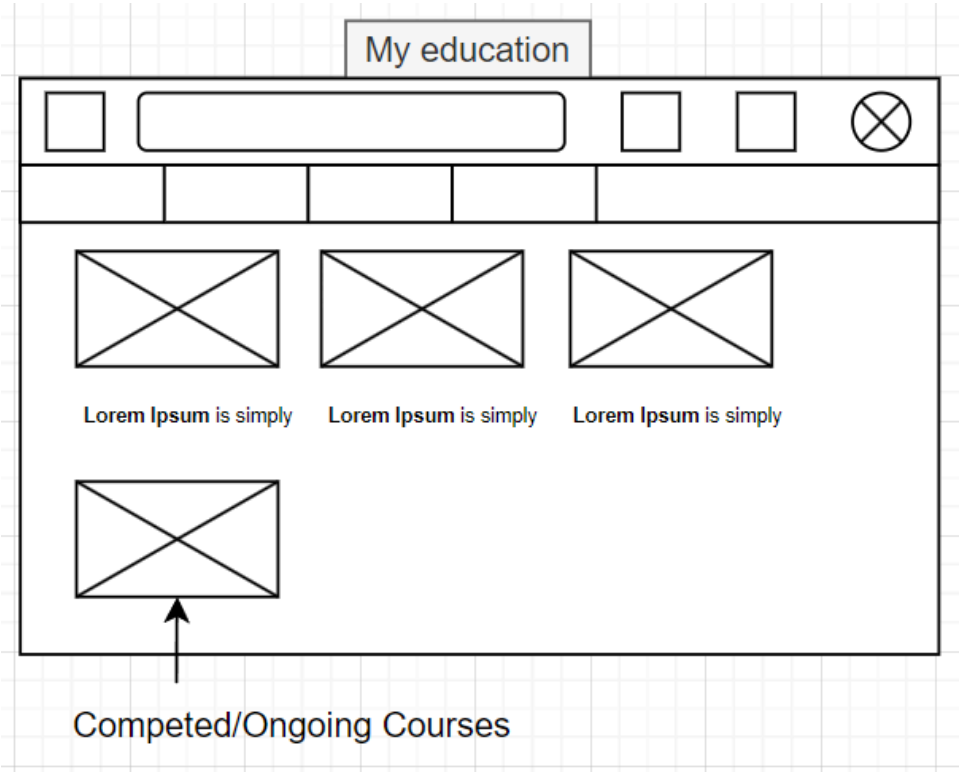

Рисунок 1.10 - Макет екранної форми «Моє навчання»

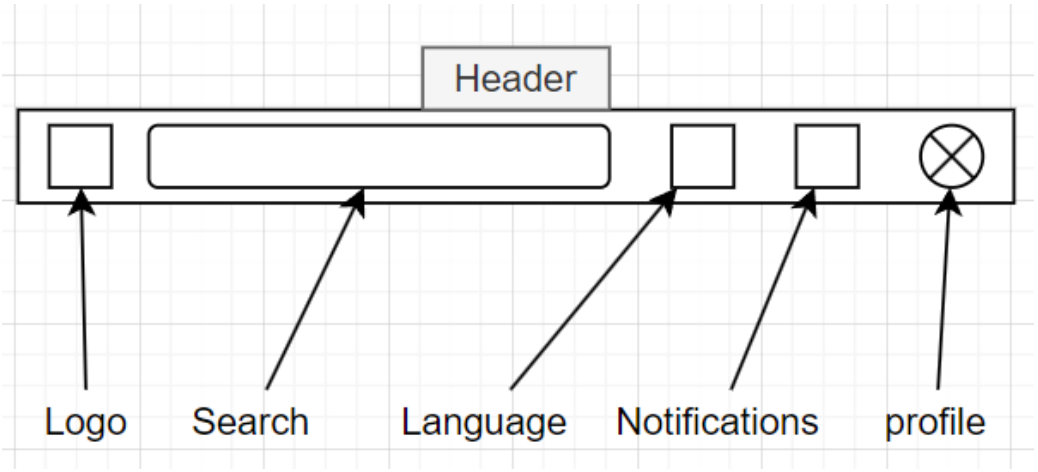

Рисунок 1.11 - Макет елементу «Хедер»

1.5 Розроблення функціональних тестів

1.5.1 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 1.5 – Функціональні тести для ролі користувача «Адміністратор»

Іденти-фікатор тесту	Послідовність дій користувача	Очікуваний результат
А.1. Адміністратор може додати/редагувати/видалити курси на платформі платних курсів.		
А.1.1	1. Ввести назву курса. 2. Ввести опис курса. 3. Додати тему курса. 4. Ввести ціну курса. 5. Натиснути кнопку «Додати».	Успішне додавання курса
А.1.2	1. У пошуку знайти потрібний курс. 2. Натиснути на сторінку курса. 3. Натиснути кнопку «Змінити» 4. Змінити назву курсу. 5. Натиснути кнопку «Зберегти».	Успішне редагування курсу
А.1.3	1. У пошуку знайти потрібний курс. 2. Натиснути на сторінку курса. 3. Натиснути кнопку «Видалити».	Успішне видалення курсу
А.2 Адміністратор може переглядати загальну статистику по усій платформі: активності користувачів, кількість і зареєстрованих користувачів, кількість придбаних курсів, кількість відвідувань тощо.		
А.2.1	1. Перейти до секції статистики. 2. Переглянути загальну кількість користувачів.	Коректне відображення статистичних даних
А.3 Адміністратор може змінювати користувачів		
А.3.1	1. Перейти до налаштувань адміністратора. 2. Обрати розділ користувачів. 3. Знайти портібного користувача 4. Змінити його логін 5. Натиснути кнопку «Змінити»	Успішна зміна облікового запису користувача
А.4 Можливість зберігати та завантажувати інформацію про курси та користувачів у форматі JSON для архівування або резервного копіювання		
А.4.1	6. Перейти до налаштувань адміністратора. 7. Обрати опцію «Архіювати». 8. Задати назву файла 9. Підтвердити дію.	Успішне архіювання інформації
А.4.2	10. Перейти до налаштувань адміністратора. 11. Обрати опцію «Завантажити архів». 12. Задати назву файла 13. Підтвердити дію.	Успішне завантаження інформації

1.5.2 Функціональні тести для ролі користувача «Зареєстрований користувач»

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

Таблиця 1.6 – Функціональні тести для ролі користувача «Зареєстрований користувач»

Іденти-фікатор тесту	Послідовність дій користувача	Очікуваний результат
ЗК.1. Зареєстрований користувач може авторизуватися		
ЗК.1.1	1. Ввести правильний логін. 2. Ввести правильний пароль 3. Натиснути кнопку "Увійти".	Успішна авторизація користувача
ЗК.1.2	1. Ввести правильний логін. 2. Ввести неправильний пароль 3. Натиснути кнопку "Увійти".	Помилка: Невірний пароль
ЗК.2 Зареєстрований користувач може переглядати курси.		
ЗК.2.1	1. Клікнути на потрібний курс	Перегляд інформації курса
ЗК.3. Зареєстрований користувач може шукати курси.		
ЗК.3.1	1. Перейти на сторінку пошуку 2. Ввести ключове слово у поле пошуку.	Успішне виведення результатів пошуку відповідно до введеного ключового слова.
ЗК.4. Зареєстрований користувач може покупати курси		
ЗК.4.1	1. Перейти на сторінку курса 2. Натиснути на кнопку "Зареєструватися". 3. Підтвердити покупку курса	Успішне придбання курсу
ЗК.5. Зареєстрований користувач може моніторити власний прогресс проходження курсів		
ЗК.5.1	1. Перейти до "Власного навчання" 2. Перейти до курсу, що проходиится 3. Натисни кнопку "Прогресс"	Успішне вивід свого прогресу курса
ЗК.6. Зареєстрований користувач має особистотий профіль, який він може змінювати		
ЗК.6.1	1. Перейти до особистого профілю. 2. Змінити своє ім’я 3. Натиснути кнопку "Зберегти".	Успішне оновлення інформації у профілі користувача.
ЗК.7. Зареєстрований користувач може оцінювати курси.		
ЗК.7.1	1. Перейти на потрібну сторінку курса 2. Натиснути кнопку "Поставити рейтинг" 3. Оцінити курс за 10-бальною системою 4. Натиснути кнопку "Підтвердити"	Успішне оцінювання курсу

1.5.3 Функціональні тести для ролі користувача «Гість»

На основі опису предметної області, а також функціональних вимог і екранних форм були розроблені функціональні тести для ролі користувача «Гість» (див. табл. 1.7).

Таблиця 1.7 – Функціональні тести для ролі користувача «Гість»

Іденти-фікатор тесту	Послідовність дій користувача	Очікуваний результат
Г.1.Гість може зареєструватися, вказавши логін і пароль		
Г.1.1	1. Ввести унікальний логін (не збігається з існуючими) довжиною не менше п'яти символів. 2. Ввести пароль довжиною не менше п'яти символів. 3. Ввести ім’я. 4. Ввести прізвище 5. Натиснути кнопку "Зареєструватися".	Успішна реєстрація гостя з новим.
Г.2. Гість може переглядати курси		
Г.2.1	2. Клікнути на потрібний курс	Перегляд інформації курса
Г.3.Гість може шукати курси		
Г.3.1	3. Перейти на сторінку пошуку 4. Ввести ключове слово у поле пошуку.	Успішне виведення результатів пошуку відповідно до введеного ключового слова.

1.5.4 Функціональні тести для ролі користувача «Викладач»

На основі опису предметної області, а також функціональних вимог і екранних форм були розроблені функціональні тести для ролі користувача «Викладач» (див. табл. 1.8).

Таблиця 1.7 – Функціональні тести для ролі користувача «Викладач»

Іденти-фікатор тесту	Послідовність дій користувача	Очікуваний результат
В.1. Викладач може зареєструватися, вказавши логін і пароль.		
В.1.1	1. Ввести унікальний логін (не збігається з існуючими) довжиною не менше п'яти символів. 2. Ввести пароль довжиною не менше п'яти символів. 3. Ввести ім’я. 4. Ввести прізвище 5. Натиснути кнопку "Зареєструватися".	Успішна реєстрація викладача з новим логіном і паролем.
В.2. Викладач може створити та редагувати свої курси		
В.2.1	1. Увійти до особистого кабінету викладача. 2. Обрати опцію "Створити новий курс". 3. Заповнити всі необхідні поля для курсу. 4. Натиснути кнопку "Зберегти зміни"	Успішне створення курсу викладачем
В.2.2	1. Увійти до особистого кабінету викладача. 2. Вибрати потрібний курс та клікнути на нього.	Успішне редагування курсу викладачем

	3. Змінити ім’я курса 4. Натиснути кнопку "Змінити"	
В.3. Викладач може переглядати учасників курсів та їх прогресс проходження курсу		
В.3.1	1. Перейти до списку власних курсів. 2. Вибрати потрібний курс..	Список учасників курсу та їх прогрес.
В.4. Можливість зберігати свої курси та інформацію про учасників у форматі JSON		
В.4.1	1. Увійти до особистого кабінету викладача. 2. Обрати опцію "Зберегти курс". 3. Підтвердити дію.	Успішне збереження даних у форматі JSON

2 МОДЕЛЮВАННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

2.1 Виділення і опис класів предметної області

Абстрактний клас «User» – користувач

Таблиця 2.1 – Виділення характеристик класу «User»

Назва характеристики	Тип значення	Обмеження
Логін	Рядок	Не пустий рядок
Пароль	Рядок	Не пустий рядок
Ім’я	Рядок	Не пустий рядок, довжина: хоча б 1 символ
Прізвище	Рядок	Не пустий рядок, довжина: хоча б 1 символ
Повідомлення	Масив рядків	
Статичний лічильник	Ціле число	Не менше нуля
Змінити пароль	Подія	

Клас «Authorised» – авторизований користувач

Таблиця 2.2 – Виділення характеристик класу «Authorised»

Назва характеристики	Тип значення	Обмеження
Придбані курси	Course[]	

Таблиця 2.3 – Виділення поведінки класу «Authorised»

Назва поведінки	Опис вхідних параметрів	Тип значення, що повертається	Примітка
Відкрити свої придбані курси	void		

Клас «Teacher» – вчитаель

Таблиця 2.4 – Виділення характеристик класу «Teacher»

Назва характеристики	Тип значення	Обмеження
Створені курси	Ціле число	Не менше нуля

Таблиця 2.5– Виділення поведінки класу «Teacher»

Назва поведінки	Опис вхідних параметрів	Тип значення, що повертається	Примітка
Створити курс	Void	True/False	
Видалити свій курс	ідинтифікатор	True/False	

Клас «Admin» – адміністратор

Таблиця 2.6 – Виділення характеристик класу «Admin»

Назва характеристики	Тип значення	Обмеження
Створені курси	Ціле число	Не менше нуля

Таблиця 2.7 – Виділення поведінки класу «Admin»

Назва поведінки	Опис вхідних параметрів	Тип значення, що повертається	Примітка
Подивитися статистику	Void	Void	
Створити користувача	Void	True/False	
Видалити користувача	ідинтифікатор	True/False	
Створити курс	Void	True/False	
Видалити свій курс	ідинтифікатор	True/False	

Клас «Course» – курс

Таблиця 2.8 – Виділення характеристик класу «Course»

Назва характеристики	Тип значення	Обмеження
Індифікатор	Ціле число	Не меньше нуля
Назва курса	Рядок	Не пустий рядок
Опис курса	Рядок	Не пустий рядок
Ціна	Грощі	
Чи виконаний курс	Логічне	
Модулів виконано	Ціле число	
Модулі	Масив модулів	Більше нуля
Теги	Масив енам теги	Більше нуля
Творець курсу	User	Не може бути null

Статичний лічильник	Ціле число	Не менше нуля

Таблиця 2.9 – Виділення поведінки класу «Course»

Назва поведінки	Опис вхідних параметрів	Тип значення, що повертається	Примітка
Розпочати N модуль	Номер модуля	void	…
Купити курс	Void	True/Flase	…

Клас «Module» – модуль курса

Таблиця 2.10 – Виділення характеристик класу «Module»

Назва характеристики	Тип значення	Обмеження
Назва модуля	Рядок	Не пустий рядок
Опис модуля	Рядок	Не пустий рядок
Кількість виконаних уроків	Ціле число	Не меньше 0
Чи виконаний модуль	Логічне	
Уроки	Масив уроків	
Статичний лічильник	Ціле число	Не меньше 0

Таблиця 2.11 – Виділення поведінки класу «Module»

Назва поведінки	Опис вхідних параметрів	Тип значення, що повертається	Примітка
Почати урок №N	Нормер урока	Lesson, що є у массиві уроків	

Клас «Lesson» – урок

Таблиця 2.12 – Виділення характеристик класу «Lesson»

Назва характеристики	Тип значення	Обмеження
Назва урока	Рядок	Не пустий рядок
Матеріали	Рядок	Не пустий рядок
Чи виконаний урок	Логічне	
Статичний лічильник	Ціле число	Не меньше 0

Таблиця 2.13 – Виділення поведінки класу «Lesson»

Назва поведінки	Опис вхідних параметрів	Тип значення, що повертається	Примітка
Виконати урок	Void	Void	
Завершити урок	Void	Void	

Клас «CoursesApp» – додаток курсів

Таблиця 2.14 – Виділення характеристик класу «CoursesApp»

Назва характеристики	Тип значення	Обмеження
Поточний користувач	User	Не може бути null

Таблиця 2.15 – Виділення поведінки класу «CoursesApp»

Назва поведінки	Опис вхідних параметрів	Тип значення, що повертається	Примітка
Пошук курсів	Ввід користувача	Course[]	
Пошук курсів	Теги	Course[]	
Завантажити курс	Курс	Void	

Інтерфейс «IStudyable» – вміння вчитися

Таблиця 2.16 – Виділення поведінки інтерфейсу «IStudyable»

Назва поведінки	Опис вхідних параметрів	Тип значення, що повертається	Примітка
Відкрити свої курси	Void	int[]	

Інтерфейс «ITeacheble» – вміння навчати

Таблиця 2.17 – Виділення поведінки інтерфейсу «ITeacheble»

Назва поведінки	Опис вхідних параметрів	Тип значення, що повертається	Примітка
Створити курс	Void	Void	
Видалити курс	Void	Void	

2.2 Встановлення зв’язків між класами

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

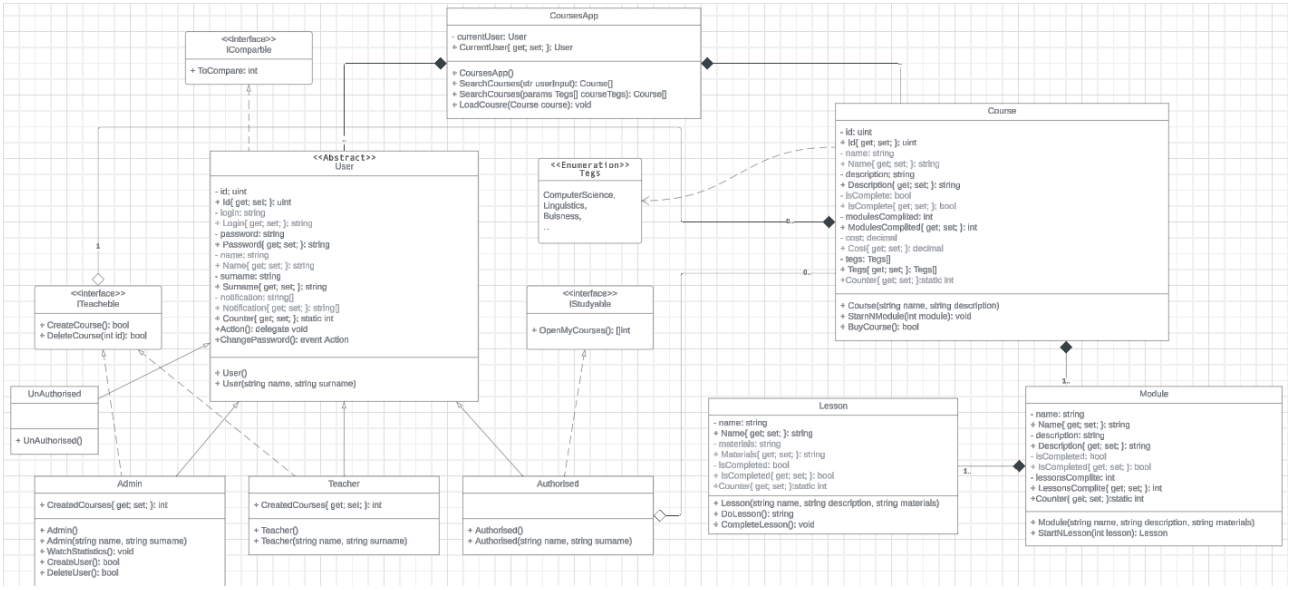

Рисунок 2.1 – Діаграма класів предметної області “Платні навчальні курси”

Всю діаграму можно умовно розділити на “користувачі” та “курси”.

Головний класс у розділі “користувачі” - абстрактний класс User (користувач): він реалізує вбудований інтерфейс IComparble.

Від абстрактного класу наслідуються чотири класси: UnAuthorised, Authorised, Teacher та Admin.

- UnAuthorised - найпростіший, використовується перед тим, як користувач залогиниться/авторизується. Не має якогось явного функціоналу.
- Authorised - звичайний користувач. Може покупати курси та проходити їх. Реалізує інтерфейс IStudyable.
- Teacher - вчитель. Відрізняється від звичайного користувача тим, що не може проходити курси. Реалізує інтерфейс ITeacheble.
- Admin - адміністратор. Також може створювати курси, але має більш обширний функціонал: може дивитися статистику курсів та створювати користувачів. Реалізує інтерфейс ITeacheble.

У розділі курси основний класс - Course (курс). Він має багато характеристик, одна з яких множина класу Module (модуль). У свою чергу один модуль містить у собі множину класу Lesson (урок, заняття).

Ці два розділи об’єднує у собі класс CoursesApp (клас застосунку).

Поміж класів є інтерфейси IStudyable та ITeacheble:

- IStudyable містить функціонал навчання.
- ITeacheble містить функціонал створення, редагування та моніторинг створених курсів

3 ПРОГРАМНА РЕАЛІЗАЦІЯ КЛАСІВ ПРЕДМЕТНОЇ ОБЛАСТІ І ЇХ ТЕСТУВАННЯ

3.1 Структура проєкту з реалізацією класів предметної області

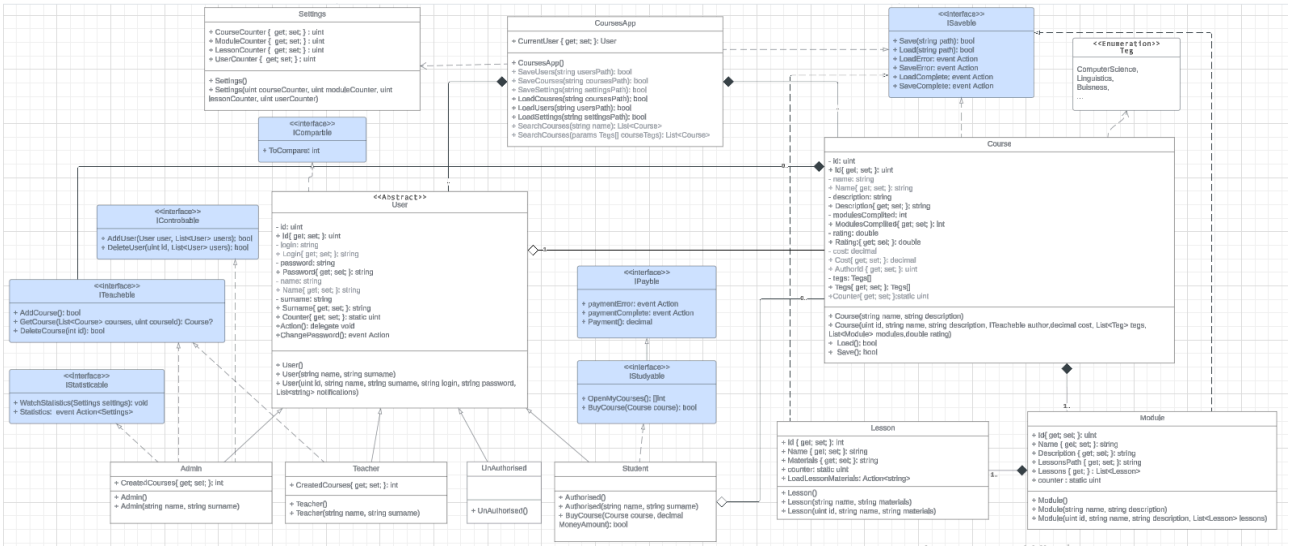

Рисунок 3.1 – Структура проєкту з реалізацією предметної області

3.2 Реалізація інтерфейсів і каркасів класів предметної області

Інтерфейс «IControlable» – здатність керувати користувачами

```
using CourseworkOOP.Entities.Users;
namespace CourseworkOOP.Iterfaces
{
    public interface IControlable
    {
        bool AddUser(List<User> users, User user);
        bool DeleteUser(List<User> users, uint id);
    }
}
```

Інтерфейс «IPayble» – здатність покупати курси

```
namespace CourseworkOOP.Iterfaces
{
    public interface IPayble
    {
        static event Action paymentError;
        static event Action paymentComplete;
        decimal Payment();
    }
}
```

Інтерфейс «ISaveble» – робота за файломи

```
namespace CourseworkOOP.Iterfaces
{
    public interface ISaveble
    {
        bool Save(string path);
```

```
        bool Load(string path);
        event Action LoadError;
        event Action SaveError;
        event Action LoadComplete;
        event Action SaveComplete;
    }
}
```

Інтерфейс «IStatisticable» – можливість перегляду статистики

```
using System;
using System.Collections.Generic;
using System.Configuration;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using CourseworkOOP.Entities;

namespace CourseworkOOP.Iterfaces
{
    internal interface IStatisticable
    {
        void WatchStatistics(Settings settings);
        event Action<Settings> Statistics;
    }
}
```

Інтерфейс «IStudyable» – здатність навчатися

```
using CourseworkOOP.Entities.Courses;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace CourseworkOOP.Iterfaces
{
    public interface IStudyable : IPayble
    {
        List<Course> Courses {  get; }
        bool BuyCourse(Course course);
    }
}
```

Інтерфейс «ITeacheble» – здатність навчати

```
using CourseworkOOP.Entities;
using CourseworkOOP.Entities.Courses;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace CourseworkOOP.Iterfaces
{
    public interface ITeacheble
    {
        bool AddCourse(List<Course> courses, Course newCourse);
        Course? GetCourse(List<Course> courses, uint courseId);
        bool DeleteCourse(List<Course> courses, uint courseId);
    }
}
```

Клас «Settings» – налаштування

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace CourseworkOOP.Entities
{
    public class Settings
    {
        public uint CourseCounter {  get; set; }
        public uint ModuleCounter { get; set; }
        public uint LessonCounter { get; set; }
        public uint UserCounter { get; set; }
        public Settings() { }
         public Settings(uint courseCounter, uint moduleCounter, uint lessonCounter,
uint userCounter)
    }
}
```

Клас «CoursesApp» – клас застосунку

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using CourseworkOOP.Entities.Users;
using CourseworkOOP.Entities.Courses;
using System.IO;
using System.Diagnostics;
using static System.Windows.Forms.LinkLabel;
using System.Security.Principal;
using System.Text.Json;
using CourseworkOOP.Iterfaces;


namespace CourseworkOOP.Entities
{
    public class CoursesApp : ISaveble
    {
        public event Action LoadError;
        public event Action SaveError;
        public event Action LoadComplete;
        public event Action SaveComplete;

        public User CurrentUser { get; set; }
        private List<User> users;
        public List<User> Users { get { return users; } }
        private List<Course> courses;
        public List<Course> Courses { get { return courses; } }

        public CoursesApp()

        public bool Save(string path = @"Data")

        public bool SaveUsers(string usersPath = @"Data\Users\UsersData.txt")

       public bool SaveCourses(string coursesPath = @"Data\Courses\CoursesData.txt")

        public bool SaveSettings(string dataPath = @"Data\Config\Settings.txt")

        public bool Load(string path = @"Data")

      public bool LoadCourses(string coursesPath = @"Data\Courses\CoursesData.txt")
```

```
        public bool LoadUsers(string usersPath = @"Data\Users\UsersData.txt")

        public bool LoadSettings(string dataPath = @"Data\Config\Settings.txt")

        public List<Course> SearchCourses(string stringToSearch)

        public List<Course> SearchCourses(params Teg[] tegs)

    }
}
```

Клас «User» – користувач

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace CourseworkOOP.Entities.Users
{
    public abstract class User: IComparable<User>
    {
        private uint id;
        public uint Id { get; set; }
        private string login;
        public string Login { get; set; }
        private string password;
        public string Password { get; set; }
        private string name;
        public string Name { get; set; }
        private string surname;
        public string Surname { get; set; }
        public static uint counter;
        public event Action<String> UpdatePassword;
        public short UserType { get; set; }

        public User()

        public User(string name, string surname)

   public User(uint id, string name, string surname, string login, string password)

        public int CompareTo(User? other)

    }
}
```

Клас «Admin» – адміністратор

```
using CourseworkOOP.Entities.Courses;
using CourseworkOOP.Iterfaces;
using Microsoft.VisualBasic.ApplicationServices;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace CourseworkOOP.Entities.Users
{
    public class Admin : User, ITeacheble, IControlable, IStatisticable
    {
        public event Action<Settings> Statistics;

       public Admin()
```

```
        public Admin(string name, string surname)

        public void WatchStatistics(Settings settings)

        public bool AddUser(List<User> users, User newUser)

        public bool DeleteUser(List<User> users, uint id)

        public bool AddCourse(List<Course> courses, Course newCourse)

        public bool DeleteCourse(List<Course> courses, uint courseId)

        public Course? GetCourse(List<Course> courses, uint courseId)
    }
}
```

Клас «Teacher» – вчитель

```
using CourseworkOOP.Entities.Courses;
using CourseworkOOP.Iterfaces;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Xml.Linq;

namespace CourseworkOOP.Entities.Users
{
    public class Teacher : User, ITeacheble
    {
        public Teacher()
        public Teacher(string name, string surname)

        public bool AddCourse(List<Course> courses, Course newCourse)

        public bool DeleteCourse(List<Course> courses, uint courseId)

        public Course? GetCourse(List<Course> courses, uint courseId)
    }
}
```

Клас «Student» – студент

```
using CourseworkOOP.Entities.Courses;
using CourseworkOOP.Iterfaces;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Text.Json.Serialization;
using System.Threading.Tasks;

namespace CourseworkOOP.Entities.Users
{
    public class Student : User, IStudyable
    {
        private List<Course> courses;
        [JsonIgnore]
        public List<Course> Courses { get => courses; }
        public List<uint> CoursesIds { get; set; }

        static public event Action paymentError;
        static public event Action paymentComplete;

        public Student()
```

```
            public Student(string name, string surname)

           public Student(string name, string surname, List<Course> courses):base (name,
surname)
            public bool BuyCourse(Course course)

            public bool BuyCourse(Course course, decimal MoneyAmount)

            public decimal Payment()

            public void LoadListOfCourses(List<Course> AllCourses)
        }
    }
```

Клас «UnAuthorised» – незарєстрований користувач

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace CourseworkOOP.Entities.Users
{
    public class UnAuthorised : User
    {
    }
}
```

Клас «Course» – курс

```
public class Course : ISaveble
{
    public uint Id { get; set; }
    public string Name { get; set; }
    public string Description { get; set; }
    public string ModulePath { get; set; }
    public uint AuthorId { get; set; }
    private double rating;
    public double Rating {get; set; }
    public int RatingsAmount {  get; set; }
    private decimal cost;
    public decimal Cost {get; set; }

    private List<Teg> tegs;

    public List<Teg> Tegs { get => tegs; set => tegs = value; }

    private List<Module> modules;
    [JsonIgnore]
    public List<Module> Modules { get => modules; }
    public static uint counter;

    public event Action LoadError;
    public event Action SaveError;
    public event Action LoadComplete;
    public event Action SaveComplete;

    public Course()

    public Course(string name, string description)

     public Course(string name, string description, string modulePath, uint author,
double rating, int ratingsAmount, decimal cost, List<Teg> tegs)
```

```
        public Course(uint id, string name, string description, string modulePath, uint
author, double rating, int ratingsAmount, decimal cost, List<Teg> tegs)

        public bool Save()

        public bool Save(string path)

        public bool Load()

        public bool Load(string path)
```

Клас «Module» – модуль

```
using CourseworkOOP.Iterfaces;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Reflection;
using System.Text;
using System.Text.Json;
using System.Text.Json.Serialization;
using System.Threading.Tasks;

namespace CourseworkOOP.Entities.Courses
{
    public class Module : ISaveble
    {
        public uint Id { get; set; }
        public string Name { get; set; }
        public string Description { get; set; }
        public string LessonsPath { get; set; }
        private List<Lesson> lessons;
        [JsonIgnore]
        public List<Lesson> Lessons { get => lessons; }
        public static uint counter;

        public event Action LoadError;
        public event Action SaveError;
        public event Action LoadComplete;
        public event Action SaveComplete;

        public Module()

        public Module(string name, string description)

       public Module(uint id,string name, string description, List<Lesson> lessons)

        public bool Save(string path)

        public bool Load(string path)
    }
}
```

Клас «Lesson» – урок

```
using CourseworkOOP.Iterfaces;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Text.Json;
using System.Threading.Tasks;

namespace CourseworkOOP.Entities.Courses
{
    public class Lesson : ISaveble
    {
```

```
        public uint Id { get; set; }
        public string Name { get; set; }
        public string Materials { get; set; }
        public static uint counter;

        public event Action LoadError;
        public event Action SaveError;
        public event Action LoadComplete;
        public event Action SaveComplete;
        public event Action<string> LoadLessonMaterials;

        public Lesson()

        public Lesson(string name, string materials)

        public Lesson(uint id, string name, string materials)

        public bool Save(string text)

        public bool Load(string text)
    }
}
```

3.3 Розроблення unit-тестів для класів предметної області

Розроблені unit-тести наведені у Додатку В.

Таблиця 3.1 – Покриття unit-тестами класу «Course»

Назва методу	Кількість розроблених unit-тестів
Rating setter	3
Cost setter	2

Таблиця 3.2 – Покриття unit-тестами класу «Admin»

Назва методу	Кількість розроблених unit-тестів
AddUser	1
DeleteUser	2
AddCourse	1
GetCourse	2
DeleteCourse	1

Таблиця 3.3 – Покриття unit-тестами класу «Teacher»

Назва методу	Кількість розроблених unit-тестів
AddCourse	1
GetCourse	2
DeleteCourse	2

Таблиця 3.3 – Покриття unit-тестами класу «Student»

Назва методу	Кількість розроблених unit-тестів
Student	1
BuyCourse	2

Таблиця 3.3 – Покриття unit-тестами класу «CoursesApp»

Назва методу	Кількість розроблених unit-тестів
SearchCourses	2

3.4 Повна реалізація класів предметної області

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

3.5 Результати unit-тестування класів предметної області

На рисунках 3.2 – 3.6 наведено скриншоти проходження розроблених unit-тестів для класів предметної області.

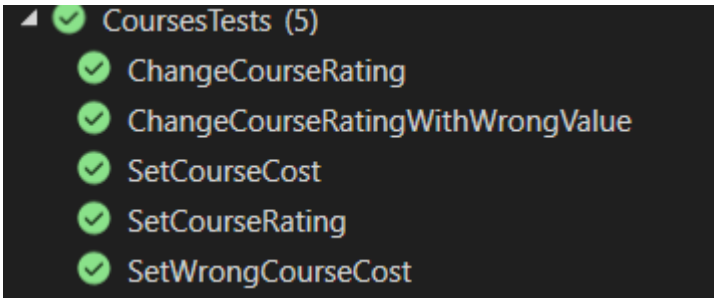

Рисунок 3.2 – Скриншот проходження unit-тестів для класу Course

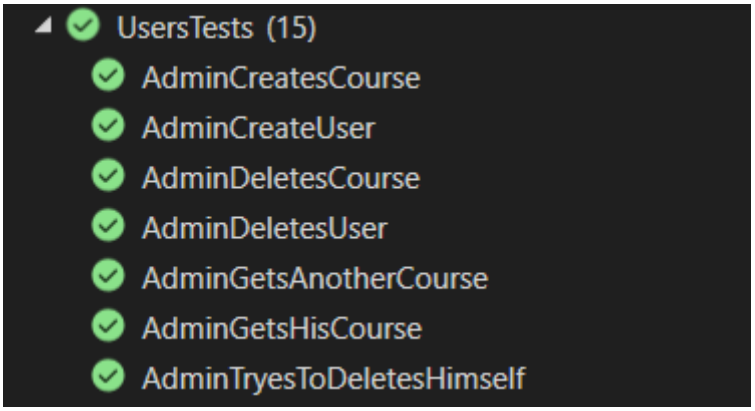

Рисунок 3.3 – Скриншот проходження unit-тестів для класу Admin

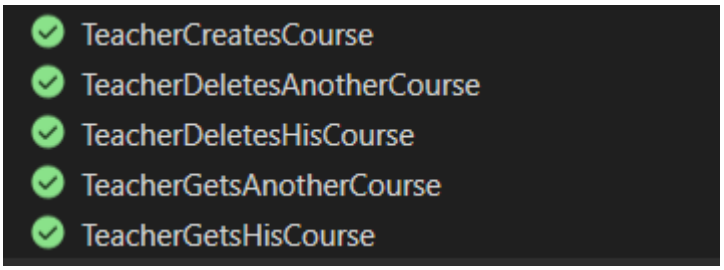

Рисунок 3.4 – Скриншот проходження unit-тестів для класу Teacher

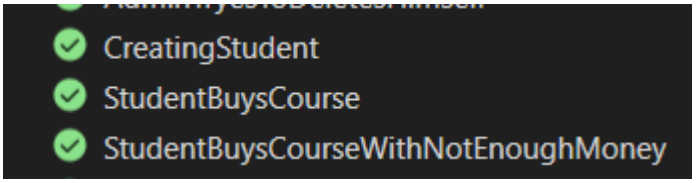

Рисунок 3.5 – Скриншот проходження unit-тестів для класу Student

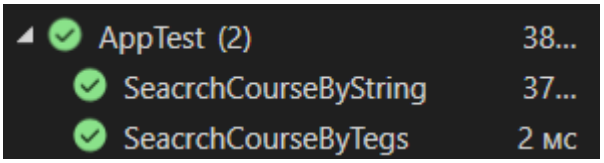

Рисунок 3.6 – Скриншот проходження unit-тестів для класу CoursesApp

4 ПРОЄКТУВАННЯ І ПРОГРАМНА РЕАЛІЗАЦІЯ КЛАСІВ ІНТЕРФЕЙСУ КОРИСТУВАЧА

4.1 Структура проєкту з реалізацією класів інтерфейсу користувача

Навести файлову/модульну структуру проєкту, яка містить реалізацію класів інтерфейсу користувача.

Рисунок 4.1 – Структура проєкту з інтерфейсом користувача

4.2 Виділення класів для реалізації інтерфейсу користувача

Описати виділені характеристики і методи для класів інтерфейсу користувача.

…

4.3 Програмна реалізація класів інтерфейсу користувача

У Додатку ___ наведено повну реалізацію програмного коду розроблених класів інтерфейсу користувача.

5 ФУНКЦІОНАЛЬНЕ ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

5.1 Функціональне тестування для ролі користувача «Адміністратор»

Для проведення функціонального тестування для ролі «Адміністратор» були використані розроблені у п. 1.4.1. функціональні тести. Результати проведення функціонального тестування наведені нижче.

Ідентифікатор тесту: A.1.1.

Вхідні дані:

1. Ввести дату сеансу: «20.01.2023».
2. Ввести час сеансу: «10:00».
3. Обрати зал у кінотеатрі: «2».
4. Обрати фільм: «Аватар: шлях води».
5. Натиснути кнопку «Додати».

Очікуваний результат: Успішне додавання сеансу

Отриманий результат:

Рисунок 5.1 – Скриншот виконання тесту A.1.1

Ідентифікатор тесту:

Вхідні дані:

Очікуваний результат:

Отриманий результат:

Рисунок 5.2 – Скриншот виконання тесту _____

5.2 Функціональне тестування для ролі користувача «Зареєстрований користувач»

Для проведення функціонального тестування для ролі «Зареєстрований користувач» були використані розроблені у п. 1.4.2. функціональні тести. Результати проведення функціонального тестування наведені нижче.

Ідентифікатор тесту:

Вхідні дані:

Очікуваний результат:

Отриманий результат:

Рисунок 5.3 – Скриншот виконання тесту _____

5.3 Функціональне тестування для ролі користувача «Гість»

Для проведення функціонального тестування для ролі «Гість» були використані розроблені у п. 1.4.3. функціональні тести. Результати проведення функціонального тестування наведені нижче.

Ідентифікатор тесту:

Вхідні дані:

Очікуваний результат:

Отриманий результат:

Рисунок 5.4 – Скриншот виконання тесту _____

ВИСНОВКИ

У результаті виконання курсової роботи було розроблено програмне забезпечення з використанням об’єктно-орієнтованої парадигми для вказати предметну область.

В ході роботи було…. (коротко описати що саме було зроблено під час виконання курсової роботи).

…

….

Примітка:

Посилання на github-репозиторій: https://github.com/PemzaZevsa/OOP_Cursova

Посилання на відеоролик: _____

ПЕРЕЛІК ДЖЕРЕЛ ТА ПОСИЛАНЬ

1. Воробйов, Ю. А. Правила оформлення навчальних і науково-дослідних документів [Текст] : навч. посіб. / Ю. А. Воробйов, Ю. О. Сисоєв. – 4-те вид., випр. і доп. – Харків : Нац. аерокосм. ун-т ім. М. Є. Жуковського «Харків. авіац. ін-т», 2019. – 88 с.
2. Автоматичне оформлення джерел по ВАК України [Електронний ресурс] – Режим доступу до ресурсу: http://vak.in.ua/do.php.
3. ChatGPT - штучний інтелект OpenAI [Електронний ресурс] // OpenAI – Режим доступу до ресурсу: https://chat.openai.com/
4. Coursera - глобальна платформа онлайн-навчання [Електронний ресурс] // Coursera Inc - Режим доступу до ресурсу: https://www.coursera.org/
5. Udemy - ринок онлайн-навчання та викладання [Електронний ресурс] // Udemy, Inc. - Режим доступу до ресурсу: https://www.udemy.com/
6. …

На всі наведені у цьому переліку публікації у тексті пояснювальної записці мають бути посилання.

ДОДАТОК А. Лістинг класів предметної області

Власноруч створений код програми.

ДОДАТОК Б. Лістинг класів інтерфейсу користувача

Власноруч створений код програми. Код, який було автоматично створено середовищем розробки, додавати не потрібно.

ДОДАТОК В. Лістинг класів unit-тестів

AppTests.cs

```
using System;
using CourseworkOOP.Entities.Users;
using CourseworkOOP.Entities.Courses;
using CourseworkOOP.Entities;
using Microsoft.VisualBasic.Devices;

namespace Tests
{
    [TestClass]
    public class AppTest
    {
        [TestMethod]
        public void SeacrchCourseByString()
        {
            CoursesApp coursesApp = new CoursesApp();
            Course course1 = new Course("The Test course","nothing");
            Course course2 = new Course("The Test courseword", "nothing");
            Course course3 = new Course("The word course", "nothing");
            Course course4 = new Course("The WoRD course 2", "nothing");
            coursesApp.Courses.Add(course1);
            coursesApp.Courses.Add(course2);
            coursesApp.Courses.Add(course3);
            coursesApp.Courses.Add(course4);

            List<Course> courses = coursesApp.SearchCourses("word");

            Assert.IsNotNull(courses);
            Assert.IsTrue(courses.Contains(course2));
            Assert.IsTrue(courses.Contains(course3));
            Assert.IsTrue(courses.Contains(course4));
        }
        [TestMethod]
        public void SeacrchCourseByTegs()
        {
            CoursesApp coursesApp = new CoursesApp();
                                Course course1 = new Course("The Test course",
"nothing","",3,4,0,10,new List<Teg>() { Teg.Cybersecurity, Teg.Development});
                  Course course2 = new Course("The Test courseword", "nothing", "", 3,
4,0, 10, new List<Teg>() { Teg.Development });
                 Course course3 = new Course("The word course", "nothing", "", 3, 4,0,
10, new List<Teg>() { Teg.Cybersecurity });
                  Course course4 = new Course("The WoRD course 2", "nothing", "", 3, 4,0,
10, new List<Teg>() { Teg.Marketing });
            coursesApp.Courses.Add(course1);
            coursesApp.Courses.Add(course2);
            coursesApp.Courses.Add(course3);
            coursesApp.Courses.Add(course4);

                 List<Course> courses = coursesApp.SearchCourses(Teg.Cybersecurity,
Teg.Development);

            Assert.IsNotNull(courses);
            Assert.IsTrue(courses.Contains(course1));
            Assert.IsTrue(courses.Contains(course2));
            Assert.IsTrue(courses.Contains(course3));
        }
    }
}
```

UsersTests.cs

```
using CourseworkOOP.Entities.Courses;
using CourseworkOOP.Entities.Users;
```

```
using CourseworkOOP.Entities;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Tests
{
    [TestClass]
    public class UsersTests
    {
        [TestMethod]
        public void CreatingStudent()
        {
            Course testCourse = new Course("TestCourse", "Testing");
            List<Course> courses = new List<Course>();
            courses.Add(testCourse);

            Student student = new Student("Steve", "Smith", courses);

            Assert.IsNotNull(student);
            Assert.AreEqual(student.Name, "Steve");
            Assert.AreEqual("TestCourse", student.Courses[0].Name);
        }
        [TestMethod]
        public void StudentBuysCourse()
        {
                Course testCourse = new Course((uint)0, "TestCourse", "Testing", "",
0,0,0, 400, null);
            Student student = new Student("Steve", "Smith");

            Assert.IsTrue(student.BuyCourse(testCourse, 400));
            Assert.IsTrue(student.BuyCourse(testCourse, 500));
        }
        [TestMethod]
        public void StudentBuysCourseWithNotEnoughMoney()
        {
             Course testCourse = new Course((uint)0, "TestCourse", "Testing", "", 0,
0, 0, 400, null);
            Student student = new Student("Steve", "Smith");

            Assert.IsFalse(student.BuyCourse(testCourse, 300));
        }
        [TestMethod]
        public void AdminCreatesCourse()
        {
            Admin admin = new Admin();
            CoursesApp coursesApp = new CoursesApp();
            Course course = new Course("test", "testing");

            admin.AddCourse(coursesApp.Courses, course);

            Assert.IsTrue(coursesApp.Courses.Contains(course));
        }
        [TestMethod]
        public void AdminGetsHisCourse()
        {
            Admin admin = new Admin();
            Course course = new Course("Test", "Course");
            CoursesApp coursesApp = new CoursesApp();
            uint courseId = course.Id;
            admin.AddCourse(coursesApp.Courses, course);

            Course? gotCourse = admin.GetCourse(coursesApp.Courses, courseId);

            Assert.IsTrue(coursesApp.Courses.Contains(course));
            Assert.IsNotNull(gotCourse);
        }
        [TestMethod]
        public void AdminGetsAnotherCourse()
```

```
{
    Admin admin = new Admin("", "");
    Teacher teacher = new Teacher("", "");
    Course course = new Course("Test", "Course");
    CoursesApp coursesApp = new CoursesApp();
    uint courseId = course.Id;
    teacher.AddCourse(coursesApp.Courses, course);

    Course? gotCourse = admin.GetCourse(coursesApp.Courses, courseId);

    Assert.IsTrue(coursesApp.Courses.Contains(course));
    Assert.IsNotNull(gotCourse);
}
[TestMethod]
public void AdminDeletesCourse()
{
    Admin admin = new Admin();
    CoursesApp coursesApp = new CoursesApp();
    Course course = new Course("test", "testing");
    admin.AddCourse(coursesApp.Courses, course);
    uint id = course.Id;

    admin.DeleteCourse(coursesApp.Courses, id);

    Assert.AreEqual(coursesApp.Courses.Count, 0);
}
[TestMethod]
public void AdminCreateUser()
{
    Admin admin = new Admin();
    CoursesApp coursesApp = new CoursesApp();
    Student user = new Student("Tom", "Tommer");

    admin.AddUser(coursesApp.Users, user);

    Assert.IsTrue(coursesApp.Users.Contains(user));
}
[TestMethod]
public void AdminTryesToDeletesHimself()
{
    Admin admin = new Admin();
    CoursesApp coursesApp = new CoursesApp();
    coursesApp.Users.Add(admin);
    uint id = admin.Id;

    admin.DeleteUser(coursesApp.Users, id);

    Assert.AreEqual(coursesApp.Users.Count, 1);
}
[TestMethod]
public void AdminDeletesUser()
{
    Admin admin = new Admin();
    CoursesApp coursesApp = new CoursesApp();
    Student user = new Student("Tom", "Tommer");
    admin.AddUser(coursesApp.Users, user);
    uint id = user.Id;

    admin.DeleteUser(coursesApp.Users, id);

    Assert.AreEqual(coursesApp.Users.Count, 0);
}
[TestMethod]
public void TeacherCreatesCourse()
{
    Teacher teacher = new Teacher("Bill", "Gates");
    Course course = new Course("Test", "Course");
    CoursesApp coursesApp = new CoursesApp();

    teacher.AddCourse(coursesApp.Courses, course);
```

```
            Assert.IsTrue(coursesApp.Courses.Any());
            Assert.IsTrue(coursesApp.Courses.Contains(course));
        }
        [TestMethod]
        public void TeacherGetsHisCourse()
        {
            Teacher teacher = new Teacher("Bill", "Gates");
            Course course = new Course("Test", "Course");
            CoursesApp coursesApp = new CoursesApp();
            uint courseId = course.Id;
            teacher.AddCourse(coursesApp.Courses, course);

            Course? gotCourse = teacher.GetCourse(coursesApp.Courses, courseId);

            Assert.IsTrue(coursesApp.Courses.Contains(course));
            Assert.IsNotNull(gotCourse);
        }
        [TestMethod]
        public void TeacherGetsAnotherCourse()
        {
            Teacher teacher = new Teacher("Bill", "Gates");
            Teacher anotherTeacher = new Teacher("Tim", "Cook");
            Course course = new Course("Test", "Course");
            CoursesApp coursesApp = new CoursesApp();
            anotherTeacher.AddCourse(coursesApp.Courses, course);
            uint courseId = course.Id;

            Course? gotCourse = teacher.GetCourse(coursesApp.Courses, courseId);

            Assert.IsTrue(coursesApp.Courses.Contains(course));
            Assert.IsNull(gotCourse);
        }
        [TestMethod]
        public void TeacherDeletesHisCourse()
        {
            Teacher teacher = new Teacher("Bill", "Gates");
            Course course = new Course("Test", "Course");
            CoursesApp coursesApp = new CoursesApp();
            teacher.AddCourse(coursesApp.Courses, course);

            teacher.DeleteCourse(coursesApp.Courses, course.Id);

            Assert.IsTrue(!coursesApp.Courses.Any());
            Assert.IsTrue(!coursesApp.Courses.Contains(course));
        }
        [TestMethod]
        public void TeacherDeletesAnotherCourse()
        {
            Teacher teacher = new Teacher("Bill", "Gates");
            Teacher anotherTeacher = new Teacher("Tim", "Cook");
            Course course = new Course("Test", "Course");
            CoursesApp coursesApp = new CoursesApp();

            anotherTeacher.AddCourse(coursesApp.Courses, course);
            teacher.DeleteCourse(coursesApp.Courses, course.Id);

            Assert.IsTrue(coursesApp.Courses.Any());
            Assert.IsTrue(coursesApp.Courses.Contains(course));
        }
    }
}
```

CoursesTests.cs

```
using CourseworkOOP.Entities.Courses;
using CourseworkOOP.Entities.Users;
using CourseworkOOP.Entities;
using System;
using System.Collections.Generic;
```

```
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Tests
{
    [TestClass]
    public class CoursesTests
    {
        [TestMethod]
        public void SetCourseRating()
        {
            Course testCourse = new Course("TestCourse", "Testing");

            testCourse.Rating = 10;

            Assert.IsNotNull(testCourse);
            Assert.AreEqual(testCourse.Rating,10d);
        }
        [TestMethod]
        public void ChangeCourseRating()
        {
            Course testCourse = new Course("TestCourse", "Testing");

            testCourse.Rating = 7;
            testCourse.Rating = 9;

            Assert.IsNotNull(testCourse);
            Assert.AreEqual(testCourse.Rating, 8d);
        }
        [TestMethod]
        public void ChangeCourseRatingWithWrongValue()
        {
            Course testCourse = new Course("TestCourse", "Testing");

            testCourse.Rating = 7;
            Action action = () => testCourse.Rating = 11;

            Assert.ThrowsException<ArgumentException>(action);
            Assert.AreEqual(testCourse.Rating, 7d);
        }
        [TestMethod]
        public void SetCourseCost()
        {
            Course testCourse = new Course("TestCourse", "Testing");

            testCourse.Cost = 199.99M;

            Assert.IsNotNull(testCourse);
            Assert.AreEqual(testCourse.Cost, 199.99M);
        }
        [TestMethod]
        public void SetWrongCourseCost()
        {
            Course testCourse = new Course("TestCourse", "Testing");

            testCourse.Cost = 199.99M;

            Action action = () => testCourse.Cost = -99.99M;

            Assert.ThrowsException<ArgumentException>(action);
            Assert.IsNotNull(testCourse);
            Assert.AreEqual(testCourse.Cost, 199.99M);
        }
    }
}
```

ДОДАТОК Г. Назва додатку

…