НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО”

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

До захисту допущено:
Завідувач кафедри
_____ Сергій СТІРЕНКО
(підпис)
“___”_____2021 р.

Дипломний проєкт

на здобуття ступеня бакалавра
за освітньо-професійною програмою “Комп’ютерні системи та мережі”
спеціальності 123 “Комп’ютерна інженерія”

на тему: *“Веб-додаток для управління курсами образотворчого мистецтва”*

Виконала: студентка 4 курсу, групи ІВ-72
Гладка Тетяна Анатоліївна _____
(прізвище, ім’я, по батькові) (підпис)

Керівник *ст. викладач Алещенко О. В.* _____
(прізвище, ім’я, по батькові) (підпис)

Консультант нормо контроль проф., д.т.н, Сімоненко В. П. _____
(назва розділу) (прізвище, ім’я, по батькові) (підпис)

Рецензент _____ _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2021 р.

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМ. ІГОРЯ СІКОРСЬКОГО”

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Рівень вищої освіти – перший (бакалавр)

Освітньо-професійна програма

“Комп’ютерні системи та мережі”

спеціальність 123 “Комп’ютерна інженерія”

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Сергій СТІРЕНКО
(підпис)
“___”_____2021 р.

ЗАВДАННЯ
на бакалаврський дипломний проєкт студента

Гладкої Тетяни Анатоліївни
(прізвище, ім’я, по батькові)

1. Тема проєкту *“Веб-додаток для управління курсами образотворчого мистецтва”*
керівник проєкту *ст. викладач Алещенко О. В.*,
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від *7 травня* 2021 року №_____
2. Термін здачі студентом закінченого проекту (роботи) *9 червня* 2021 р.
3. Вихідні дані до проекту (роботи) *технічне завдання, теоретичні данні*
4. Зміст розрахунково-пояснювальної записки (перелік питань, які розробляються)
Розгляд існуючих рішень та технологій, програмна реалізація додатку
5. Перелік графічного матеріалу (з точним позначенням обов’язкових креслень)
Схема бази даних, діаграма класів моделей, алгоритм запису студента на курс
6. Консультанта проєкту (робота), з вказівкою розділів роботи, які до них

вносяться

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Нормоконтроль	Сімоненко В. П.		

3. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Найменування етапів дипломного проекту (роботи)	Строк виконання етапів проекту(роботи)	Примітки
1.	*Вивчення рекомендованої літератури*	*21.02.2021-04.03.2021*	
2.	*Вивчення та аналіз завдання*	*04.03.2021-15.03.2021*	
3.	*Розробка архітектури та загальної структури системи*	*15.03.2021-25.03.2021*	
4.	*Розробка структур окремих підсистем*	*25.03.2021-05.04.2021*	
5.	*Програмна реалізація системи*	*05.04.2021-15.04.2021*	
6.	*Оформлення пояснювальної записки*	*15.04.2021-20.05.2021*	
8.	*Передзахист*	*22.05.2021*	
9.	*Захист*	*16.06.2021*	

Студент-дипломник _____ Тетяна ГЛАДКА

(підпис)

Керівник проєкту _____. Олексій АЛЕЩЕНКО

(підпис)

Анотація

В даному бакалаврському дипломному проекті реалізовано систему для управління курсами образотворчого мистецтва.

Система дає змогу студентам виконувати пошук курсів з використанням різноманітих фільтрів, переглядати детальну інформацію про кожен та записуватись і проводити оплату онлайн на найбільш цікавий курс. А викладачу створювати, редагувати та видаляти власні курси, запрошувати колег викладачів. Користувачі також мають змогу переглядати онлайн-галерею та викладаючи фотографії власних робіт.

Серверна частина системи розроблена на мові Java на з використанням фреймворку Spring у інтегрованому середовищі розробки IntelliJ IDEA, а клієнтська – з використанням шаблонізатору Thymeleaf та мови JavaScript.

Annotation

In this bachelor's diploma project the system for management of courses of fine arts is developed.

The system allows students to search for courses using a variety of filters, view detailed information about each, and sign up and pay online for the most interesting course. And the teacher to create, edit and delete their own courses, invite fellow teachers. Users also have the opportunity to view the online gallery and post photos of their own work.

The server part of the system is developed in Java on using the Spring framework in the integrated development environment IntelliJ IDEA, and the client - using the Thymeleaf template engine and JavaScript.

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
			Документація загальна		
			Заново розроблена		
1	А4		Опис альбому	1	
2	А4	ІАЛЦ.467800.001 ВП	Відомість проекту	1	
3	А4	ІАЛЦ.467800.002 ТЗ	Технічне завдання	3	
4	А4	ІАЛЦ.467800.003 ПЗ	Пояснювальна записка	64	
5	А3	ІАЛЦ.467800.004 Д1	Схема бази даних	1	
6	А3	ІАЛЦ.467800.005 Д2	Діаграма класів моделей	1	
7	А3	ІАЛЦ.467800.006 Д3	Алгоритм запису студента на курс	1	

ТЕХНІЧНЕ ЗАВДАННЯ

до дипломного проєкту

на тему: «Веб-додаток для управління курсами образотворчого мистецтва»

Київ – 2021 р.

ЗМІСТ

1. Найменування та область застосування

Найменування: «Веб-додаток для управління курсами образотворчого мистецтва».

Область застосування: практичне використання студентами для пошуку та запису на курси образотворчого мистецтва та викладачами для створення власних курсів.

2. Підстави для розробки

Підставою для розробки веб-додатку для управління курсами образотворчого мистецтва є завдання на дипломне проектування, затверджене кафедрою обчислювальної техніки Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

3. Мета та призначення розробки

Метою даної розробки є побудова веб-додатку для управління курсами образотворчого мистецтва. Призначення даної розробки полягає в створенні програмного забезпечення, яке надає можливість виконувати пошук та запис на курси образотворчого мистецтва.

4. Джерела розробки

Джерелом розробки є науково-технічна література з технічних питань та статі в інтернеті за даною темою.

5. Технічні вимоги

5.1. Вимоги до розроблюваного продукту

Вимоги до розробки:

- простий та зрозумілий інтерфейс системи;
- пошук курсів з використанням фільтрів;

- перегляд детальної інформації по курсу, викладача, студентів;
- запис та оплата онлайн;
- створення та редагування курсів викладачами;
- особистий кабінет;
- онлайн-галерея.

5.2. Вимоги до програмного забезпечення

Вимоги до програмного забезпечення:

- OS: MS Windows, Linux, MacOS;
- доступ до мережі Інтернет;
- веб-браузери: Google Chrome, Firefox, Opera.

5.3. Вимоги до апаратного забезпечення

Вимоги до апаратного забезпечення:

- процесор Intel чи AMD;
- 2 Гб оперативної пам’яті.

6. Етапи розробки

Етап	Термін
Вивчення літератури	16.01.2021
Аналіз існуючих рішень	04.02.2021
Огляд та аналіз ресурсів для розробки	11.02.2021
Розробка програмного продукту	08.03.2021
Тестування	23.04.2021
Відлагодження і виправлення помилок	01.05.2021
Оформлення документації дипломного проєкту	06.06.2021

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломного проєкту

на тему: «Веб-додаток для управління курсами образотворчого мистецтва»

Київ – 2021 р.

ЗМІСТ

ВСТУП

У кожного з нас є можливість навчатися не виходячи зі своєї домівки, читаючи книгу чи проходячи онлайн-заняття. Проте іноді буває важко повноцінно розібратись самотужки, можна легко розгубитись в цьому величезному розмаїтті інформації.

Якщо стоїть мета - навчитися малювати - краще зупинити вибір на курсі з чітко вибудуваної програмою навчання, що розроблена провідними експертами. Навіть якщо є бажання освоїти одну конкретну техніку, масла або акварелі, або навчитися реалістично малювати простим олівцем. Можна вибрати від коротких, вечірніх та вихідних до академічних курсів, щоб не просто розвинути навички, а розкрити свій внутрішній творчий потенціал, навчитися керувати натхненням і зрозуміти, як використовувати свій талант і творчі здібності.

Практикуючі дизайнери що ведуть свої проекти, володіють власними дизайн-студіями можуть передати свій практичний досвід і навчити працювати в реаліях сучасного ринку. Показати як працювати з реальним об'єктом і взаємодіяти з реальним замовником. Такий досвід – вкрай цінний для новачка. Він додасть впевненості і буде сприяти швидкому старту в професії.

Тому було вирішено створити веб-додаток який полегшить майбутнім художникам та дизайнерам, чи просто людям захопленим мистецтвом, процес вибору найбільш вдалого курсу та наставника для навчання, самореалізації та розкриття індивідуального творчого потенціалу.

РОЗДІЛ 1 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1. Опис предметного середовища

Образотворче мистецтво – розділ мистецтв, вид художньої творчості, що практикується задля естетичної цінності та краси, а не функціональної цінності. Його метою якого є відтворення навколишнього світу у вигляді художніх образів на площині та в просторі.

Образотворче мистецтво відображає дійсність у наочних образах, відтворює об'єктивно наявні властивості реального світу: об'єм, колір, просторовість, матеріальну форму предмета, світлоповітряне середовище тощо. Проте образотворче мистецтво зображує не тільки те, що доступне безпосередньому зоровому сприйняттю, але й передає розвиток подій у часі, певну фабулу, розгорнуту оповідь. Воно розкриває духовний склад людини, її психологію [1].

Види образотворчого мистецтва:

- скульптура (робота виконана шляхом висікання, виливки чи ліплення);
- архітектура (мистецтва проектування будівель);
- декоративно-прикладне мистецтво (мистецтво прикрашати предмети побуту);
- фотомистецтво (мистецтво створення художньої фотографії);
- живопис (робота виконана в кольорі, фарбами, на площині);
- графіка (робота виконана лініями, штрихами, на площині без фарб);

1.1.1. Опис завдання

Метою даної роботи є розроблення веб-додатку для управління курсами образотворчого мистецтва. Дане програмне забезпечення призначене для

зручного пошуку та запису на різноманітні курси образотворчого мистецтва від фахівців своєї справи.

Функціональність програми буде надавати змогу учням отримати всю необхідну актуальну інформацію по курсу, переглянути учнів що вже там навчаються, та ознайомитись з детальною інформацією про викладача. Користуватись особистим кабінетом, виконати платіжку онлайн та завантажувати фотографії своїх робіт в галерею.

Окрім звичайних користувачів (учнів) також будуть наявні адміністратори (викладачі) яким будуть надані можливості для створення та редагування власних курсів.

1.1.2. Опис функціональних вимог

Опис курсу образотворчого мистецтва може містити:

- назву;
- загальний опис;
- зображення роботи по темі;
- додаткові зображення що ілюструють процес виконання роботи;
- відео-демонстрацію;
- план курсу розписаний на кожне заняття або по темах;
- матеріали що будуть необхідні;
- цільову аудиторію;
- складність курсу;
- дисципліну;
- дату та час проведення, дні тижня, тривалість курсу;
- ціну;
- кількість вільних місць;
- викладача що буде вести курс;
- студентів що вже записалися на курс.

Веб-застосування перш за все має надавати можливість пошуку та перегляду курсів образотворчого мистецтва. Оскільки всі курси є публічними,

а їх пошук та перегляд лише використовує дані та ніяк їх не змінює, то користувачам, які бажають просто ознайомитися з наявними курсами, нема потреби проходити процес реєстрації. Оскільки існує необхідність пошуку за даними курсу(назва, дисципліна, цільова аудиторія, викладач), то зрозумілим є, що користувач також повинен мати змогу переглядати деталі курсу.

Тобто виникає роль – гість (незареєстрований користувач).

Проте, якщо користувач матиме бажання записатися на курс, то додаток повинен ідентифікувати його при повторному вході. Для цього користувач повинен зареєструватися за допомогою власного емейлу та паролю, та зможе використовувати їх для входу в особистий кабінет. Тобто виникає наступна роль – студент (зареєстрований користувач).

Створення початкового списку курсів образотворчого мистецтва, з детальним описом кожного, потребує занадто багато часу та зусиль, якщо додавати їх в базу даних напряму. Окрім цього, список курсів буде оновлюватися з часом, змінюватися дата проведення та деталі курсу. Та звісно ж якщо у школи образотворчого мистецтва буде багато студентів, то буде необхідність в більшій кількості викладачів, що будуть створювати нові курси з різних спеціальностей. Тож необхідно створити користувацький інтерфейс для додавання та правки курсів на клієнтському рівні. Логічним буде те, що лише автор курсу може його змінювати та видаляти, для цього додаток має його ідентифікувати. Тобто виникає ще одна роль – викладач (зареєстрований користувач).

1.1.3. Опис нефункціональних вимог

Потрібно, щоб учні та викладачі могли ефективно використовувати додаток, тобто він має бути простим, зручним та функціональним.

З точки зору графічного інтерфейсу – кількість елементів повинна бути мінімальною, щоб користувач концентрував увагу на справді необхідній інформації. Необхідно, щоб дизайн був привабливим, це зробить

використання додатку більш приємним та розширить користувацьку аудиторію, і адаптивним, для перегляду застосунку з мобільного пристрою.

Також надійним – попри проблеми та навантаження він повинен стабільно працювати або надавати інформативні помилки при наявності.

1.2. Огляд наявних аналогів

1.2.1. Art School of San Francisco Bay

Вчать студентів малювати все, що завгодно, з уяви чи природи (люди, тварини чи прибульці, транспортні засоби чи пейзажі), використовуючи будь-які доступні мистецькі засоби. Навчають контролювати всі стадії творчого процесу, починаючи з нуля і закінчуючи штрихами, і застосовувати ці знання в інших (нехудожніх) аспектах свого життя.

Вчать не копіювати твори, зроблені кимось іншим, а допомагають налагодити кращий зв’язок і відкрити свій творчий потенціал.

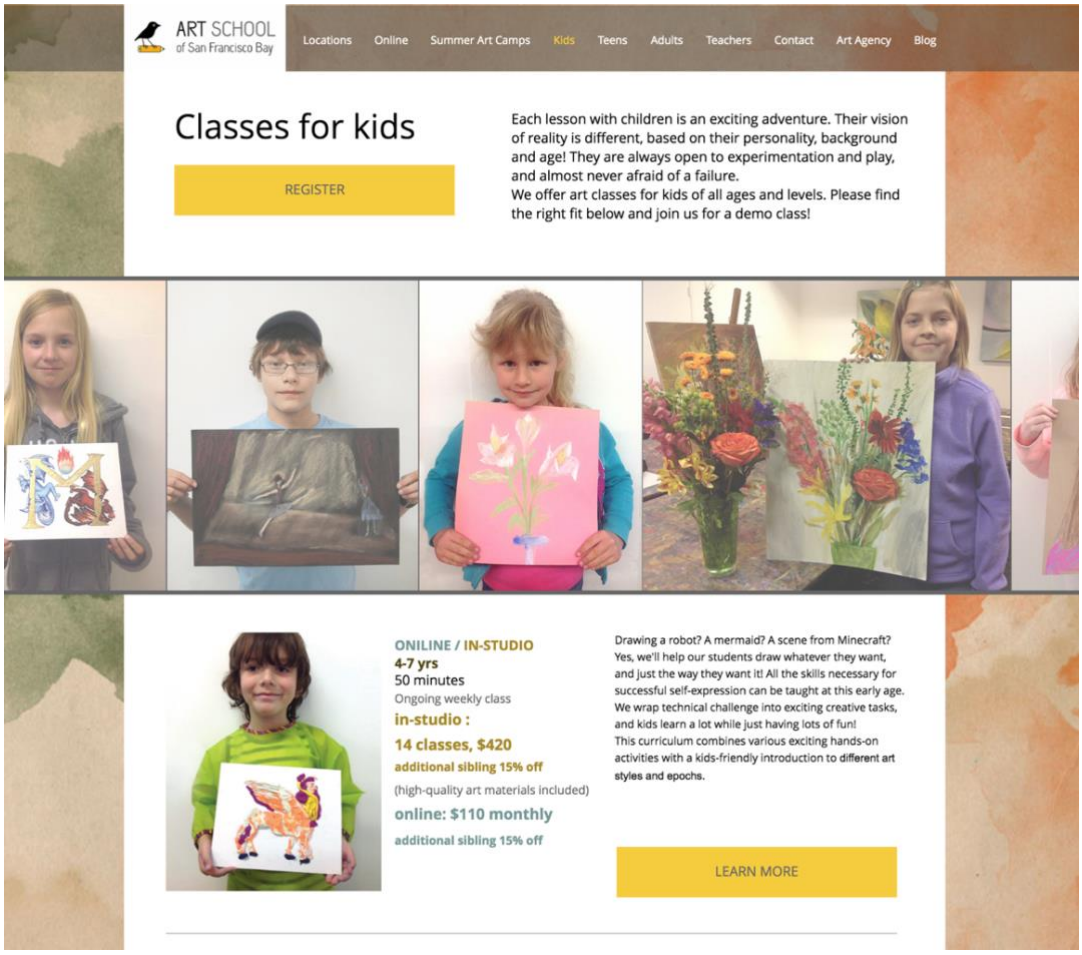


Рисунок 1.1 – Курси школи Art School of San Francisco Bay

Переваги:

- курси для дітей, підлітків та дорослих;
- літні табори для дітей;
- онлайн курси;
- повний опис курсу, з фотографіями та іноді відео;
- блог з фото та відео з курсів;
- план курсу розписаний по днях;
- детальний опис інформації про викладача;
- є можливість відвідати пробний урок;
- можна відправити форму для резервації місця на курс;
- карта показує місцезнаходження школи;
- відгуки про курси;
- форма для запитання онлайн;
- подарункові сертифікати;
- простий та зрозумілий UI/UX.

Недоліки:

- немає можливості пошуку курсів;
- невелика кількість курсів;
- відсутній особистий кабінет;
- не можна записатися та сплатити онлайн;
- немає форми для запитання онлайн;
- нераціональне використання місця на сайті, дуже великі заголовки та кнопки;
- деякі сторінки відображаються дуже повільно.

Школа мистецтв та галерея образотворчого мистецтва була створена в 1974 році як некомерційний культурно-ресурсний центр. Група художників та друзів, що прагнуть створити спільноту мистецьких шкіл, побачили

можливість в місцевій будівлі церкви 19 століття. Група перетворила стару церкву на центр художнього навчання та збагачення.

1.2.2. The Art School of Old Church

Є чотири студії, обладнані для малювання, живопису, скульптури, кераміки, ювелірних виробів, скляних намистин, кошика, графіки, цифрової фотографії та колажу. Заняття пропонуються цілий рік протягом чотирьох семестрів.

Є дві художні галереї: галерея Михайла Закіна, названа на честь нашого засновника, та галерея Кафе, в яких обидва експонуються різноманітні експонати протягом року.

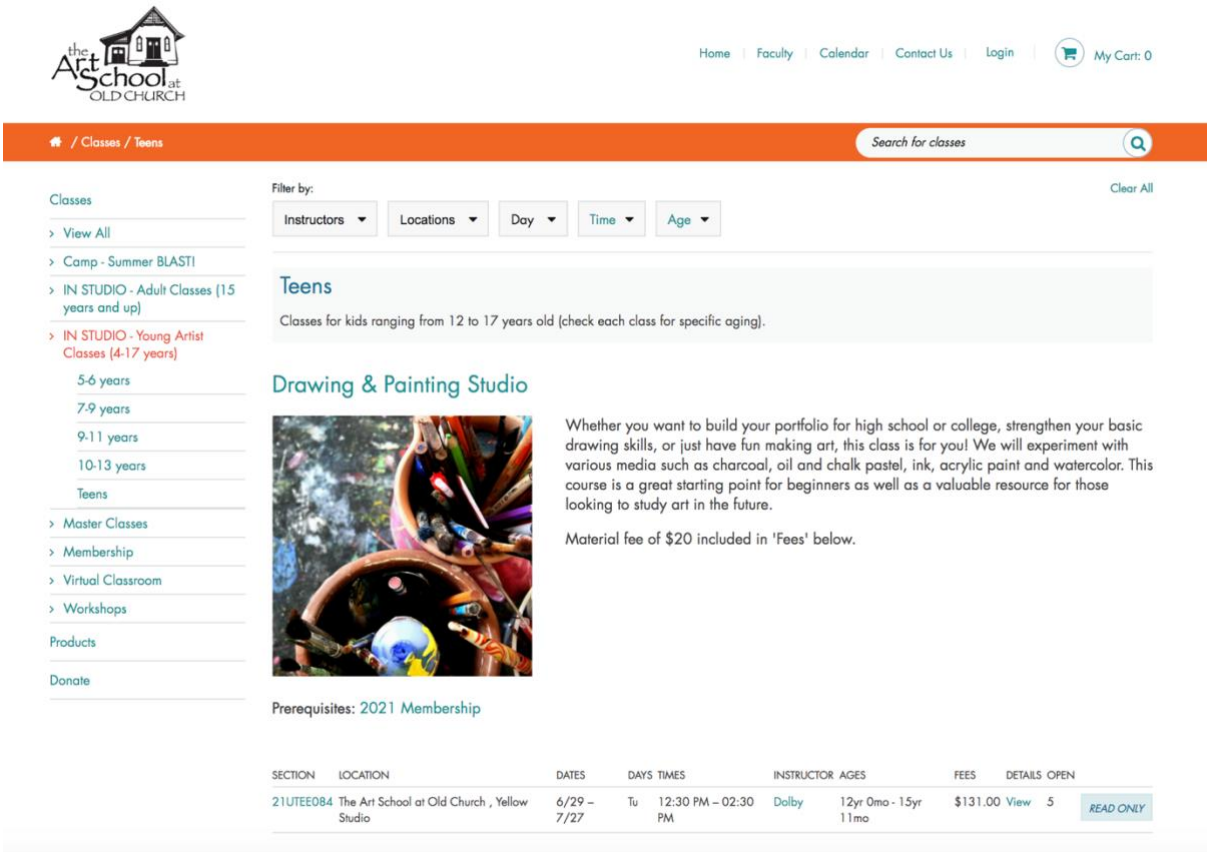


Рисунок 1.2 – Курси школи The Art School of Old Church

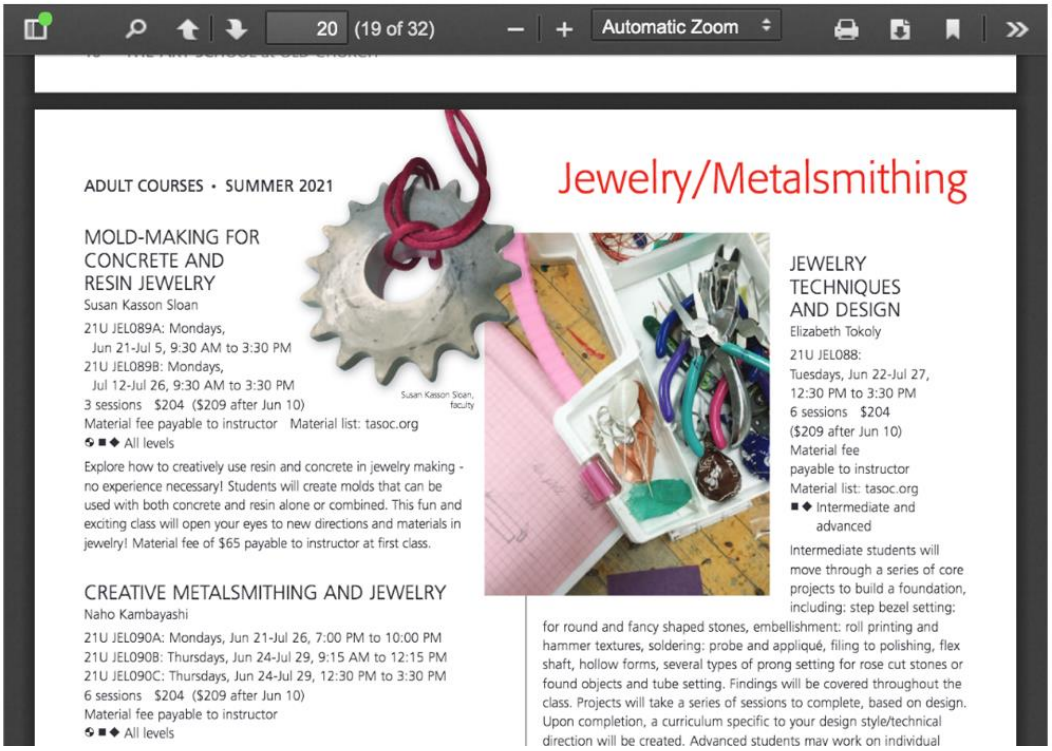


Рисунок 1.3 – Інтерактивний PDF каталог школи The Art School of Old Church

Переваги:

- курси для дітей, підлітків та дорослих;
- літні табори для дітей;
- онлайн курси;
- фільтри для пошуку;
- детальна інформація про курси з зображеннями;
- інтерактивний PDF каталог з курсами та фото;
- особистий кабінет;
- можливість сплатити онлайн;
- емейл-підписка;
- карта, що показує місцезнаходження школи;
- форма для регістрації на курс, запитання онлайн, зворотного зв’язку;
- подарункові сертифікати;
- невеличкий магазин з потрібними товарами;

- календар з подіями;
- онлайн галерея.

Недоліки:

- немає можливості пошуку курсів;
- деякі елементи не відповідають сторінкам;
- важко знайти необхідну інформацію;
- деякі сторінки відображаються дуже повільно;
- важкий, незручний та незрозумілий UI/UX.

1.2.3. Art Academy London

Інноваційна школа мистецтв була заснована в 2000 році, пропонує курси, розроблені та проведені практикуючими художниками. Навчають задумуватися про ширший теоретичний та історичний контекст вибраної дисципліни та застосовувати ці знання у своїх практичних роботах.

Дають доступ до лекцій та бесід видатних художників, відкритий доступ до друкарні та майстерні, а також пропонують завітати на майстер-класи у вихідні дні. Можна скористатися безкоштовними місцями на інших вечірніх, вихідних та коротких курсах (за умови наявності).

Можна відвідати багато різних програм від коротких вихідних курсів та майстер-класів до бакалаврських програм, що підтверджені “Відкритим університетом”.

Є система стипендій, що можна обговорити на співбесіді. Одержувачі стипендій будуть обрані та повідомлені про це після процесу співбесіди.

Проводяться регулярні дні відкритих дверей протягом року, що дає можливість отримати відповіді на всі запитання та отримати поглиблену інформацію про курси.

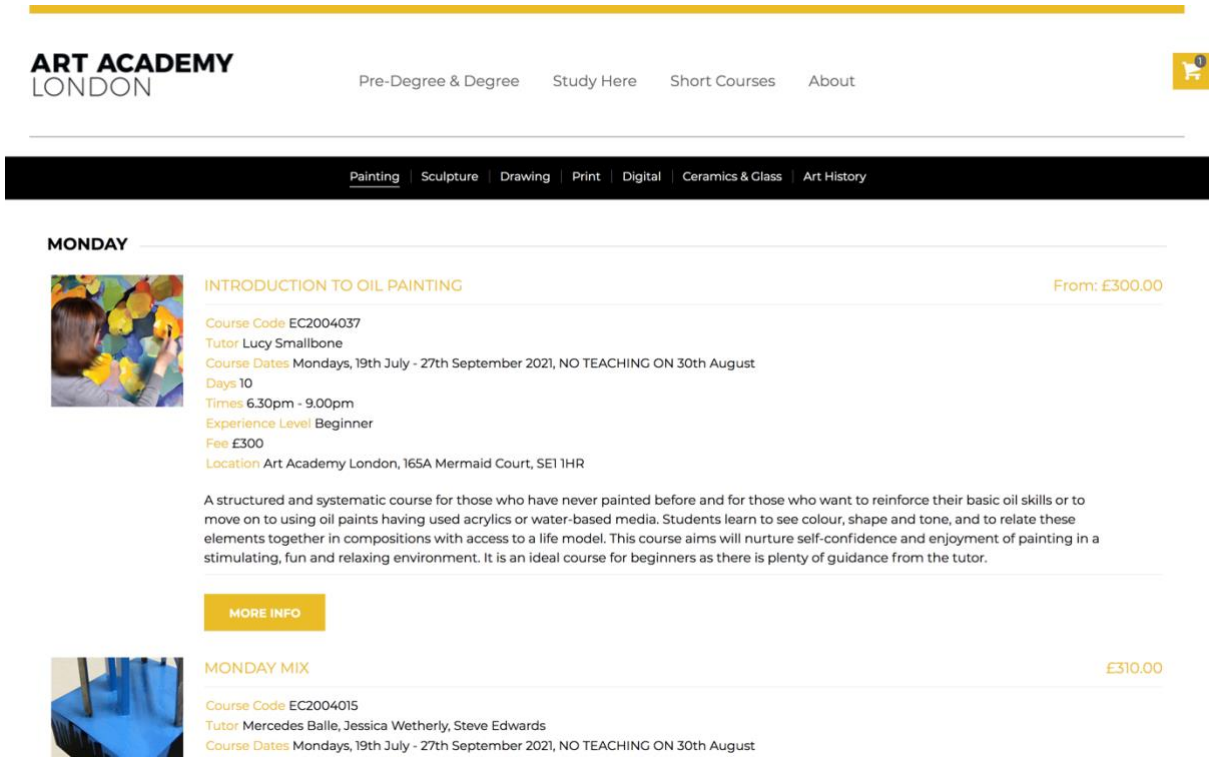


Рисунок 1.4 – Курси школи Art Academy London

Переваги:

- курси для дітей, підлітків та дорослих;
- літні табори для дітей;
- онлайн курси;
- бакалаврські та магістерські програми;
- програма сертифікації;
- дуже багато різних програм та курсів;
- повний опис курсу, з фотографіями;
- можливість пошуку по дисциплінах;
- вся інформація дуже детально описана та гарно структурована;
- інформація про викладача;
- карта з місцем проведення курсу;
- емейл-підписка;
- історії випускників;
- стипендії;
- дні відкритих дверей;

- зрозумілий UI/UX.

Недоліки:

- мало фільтрів для пошуку;
- немає фотографії викладача;
- відсутній особистий кабінет;
- немає форми для запитання онлайн;
- недостатня кількість фотографій;
- відсутня галерея з роботами студентів.

1.2.4. RISD

Школа дизайну Род-Айленда - приватна школа мистецтв та дизайну, була заснована у 1877 році Хелен Аделією Роу Меткалф, яка прагнула збільшити доступність дизайнерської освіти для жінок. Сьогодні RISD пропонує як курси для дітей так і бакалаврські та магістерські програми. Музей школи, в якому розміщені колекції мистецтва, є одним з найбільших музеїв мистецтв серед коледжів США.

Школа дизайну пов'язана з Університетом Брауна, з яким вона пропонує програми подвійного диплому на рівні аспірантів та студентів.

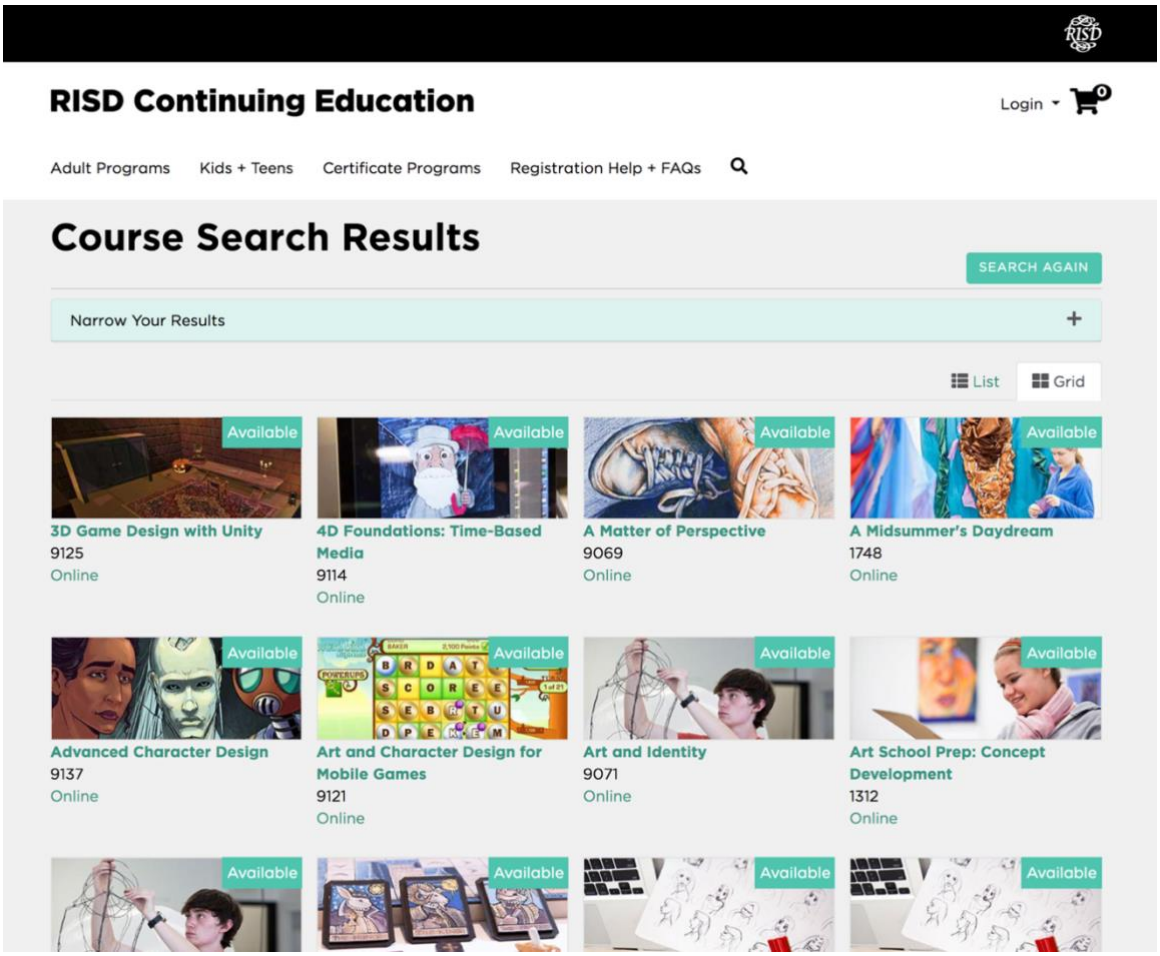


Рисунок 1.5 – Курси школи RISD Continuing Education

Переваги:

- курси для дітей, підлітків та дорослих;
- літні табори;
- онлайн курси;
- бакалаврські та магістерські програми;
- програма сертифікації;
- багато фільтрів для пошуку;
- інформація про викладача;
- особистий кабінет;
- можливість сплатити онлайн;
- можливість перестали емейлом інформацію про курс;
- можливість отримання емейлу про курс за декілька тижнів до початку або коли він буде відкритий для запису;

- можливість роздрукувати інформацію про курс;
- простий UI/UX.

Недоліки:

- немає фотографій викладачів;
- відсутній детальний план курсу;
- відсутня інформація про необхідні матеріали;
- немає форми для запитання онлайн;
- відсутня галерея з роботами студентів;
- непродумані та незручні деякі елементи дизайну.

1.2.5. Fine Arts Center

Центр образотворчого мистецтва Колорадо-Спрінгз в Колорадо-коледжі підтримує візуальне мистецтво, виконавське мистецтво та мистецьку освіту в межах громади Колорадо-Спрінгз. У галереях, класах, студіях та віртуальних майданчиках мета - збагатити життя, підтримати митців та виховати розуміння та оцінку багатьох культур людства.

Досвідчені художники та викладачі створюють мотиваційне та підтримуюче середовище для розкриття творчого потенціалу.

Навчання проходить в невеликих за розміром групах, що дає змогу більше навчатися безпосередньо у викладача.

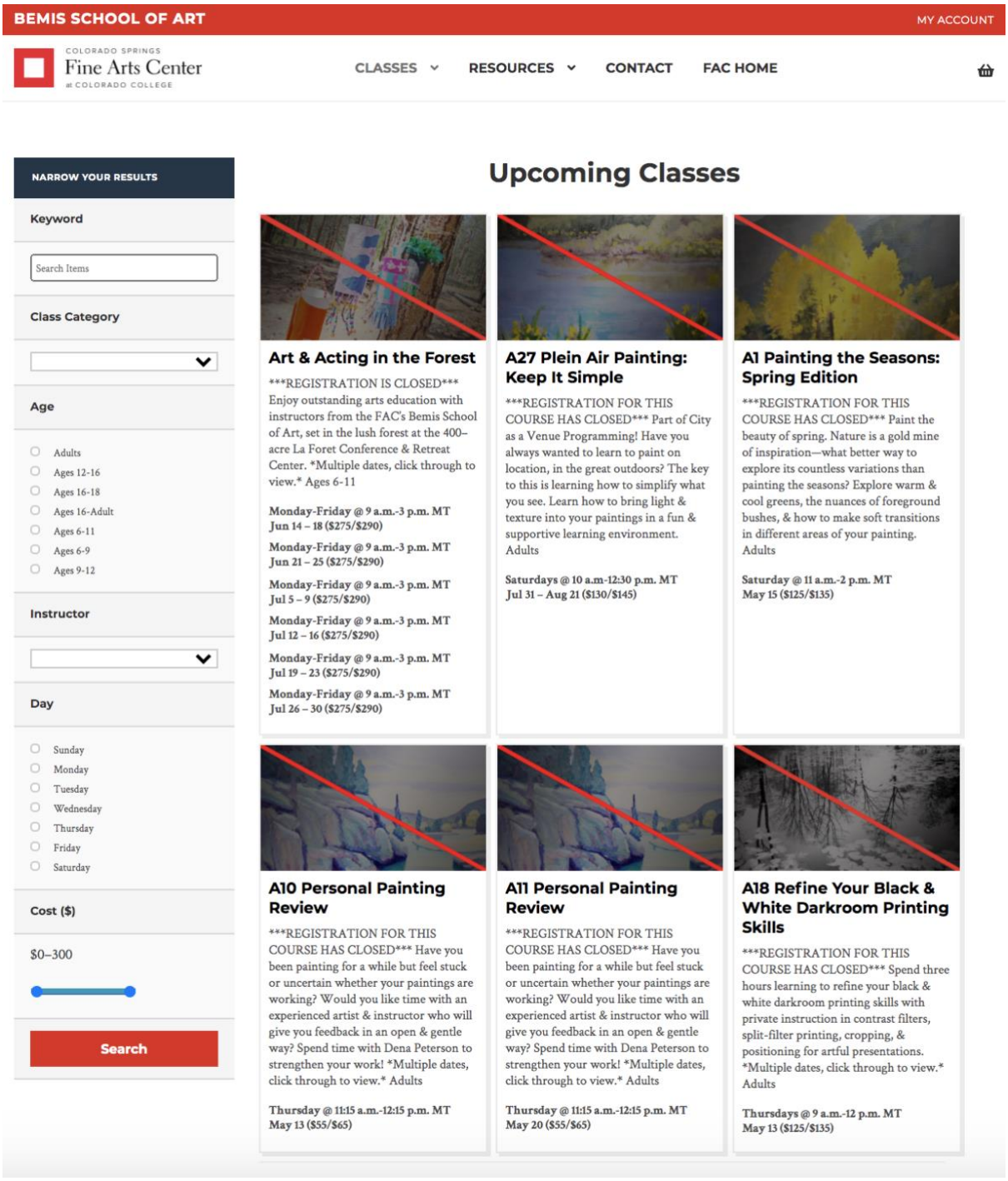


Рисунок 1.6 – Курси школи Fine Arts Center

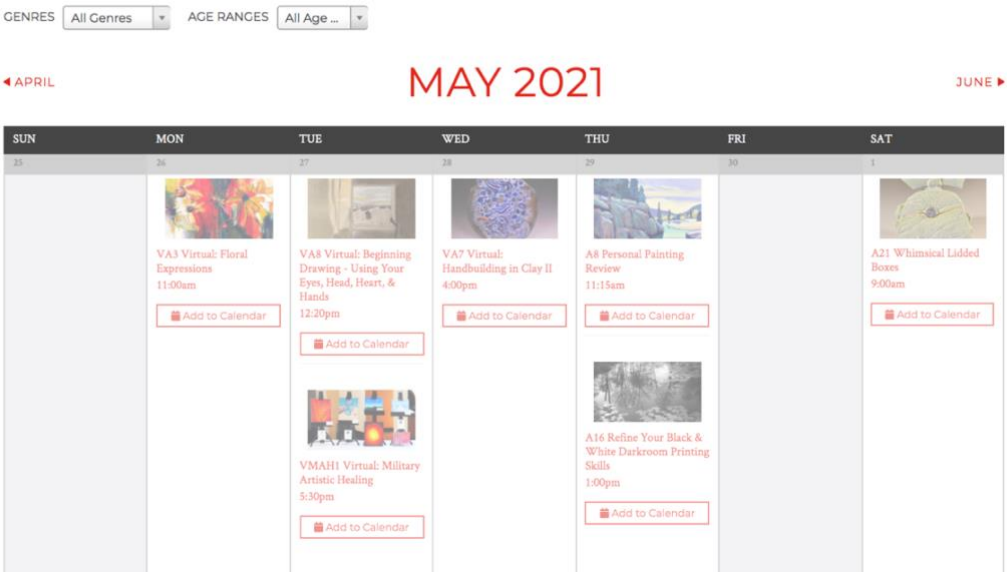


Рисунок 1.7 – Календар з розкладом курсів школи Fine Arts Center

Переваги:

- курси для дітей, підлітків та дорослих;
- календар з розкладом курсів;
- інтерактивний PDF каталог курсів;
- повний опис курсу, з фотографіями;
- особистий кабінет;
- можливість записатися та сплатити онлайн;
- подарункові сертифікати;
- інформація про викладача;
- зручний та зрозумілий UI/UX.

Недоліки:

- немає онлайн курсів;
- немає літніх таборів для дітей;
- відсутній детальний план курсу;
- відсутня інформація про необхідні матеріали;
- відсутня галерея з роботами студентів.

ВИСНОВКИ ДО РОЗДІЛУ 1

У даному розділі були описані мета проведення розробки, основі задачі та цілі.

Було виконано дослідження існуючих веб-додатків для управління курсами образотворчого мистецтва. Визначено наявні типи курсів, підходи до організації та оформлення, додаткові можливості, основні переваги та недоліки.

Виділені основні ролі користувачів та їх функції. Сформовано перелік функціональних та нефункціональних вимог до веб-додатку та проаналізовано їх відношення до основних сценаріїв роботи.

РОЗДІЛ 2 ОГЛЯД ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ ДОДАТКУ

2.1. Вибір засобів розробки

Веб-додаток реалізований на мові Java. Для реалізації серверної частини був вибраний фреймворк Spring Boot, що дозволяє створювати проект з мінімальною конфігурацією. Було реалізовано шаблон MVC, що розподіляє дані додатку на 3 рівні: модель, представлення та контролер. Клієнтська частина розроблена з використанням шаблонізатору Thymeleaf, адже він зручний для створення простого UI, і легко інтегрується з Spring Framework. Зберігання даних організовано з використанням бази даних MySQL. Для зв’язку між сервером та БД був використаний Spring Data JPA.

2.1.1. Java

Java – мова програмування що була розроблена для паралельної роботи, використання класів та об’єктно-орієнтованого програмування корпорацією Sun Microsystems в 1995 році.

Програми Java компілюються в байт-код, що далі виконується JVM (віртуальною машиною Java), тому вони можуть працювати на будь-якій комп'ютерній архітектурі.

Мова значно запозичила синтаксис із C і C++. Зокрема, взято за основу об'єктну модель C++, проте її модифіковано. Усунуто можливість появи деяких конфліктних ситуацій, що могли виникнути через помилки програміста та полегшено сам процес розробки об'єктно-орієнтованих програм. Ряд дій, які в C/C++ повинні здійснювати програмісти, доручено віртуальній машині. Передусім Java розроблялась як платформо-незалежна мова, тому вона має менше низькорівневих можливостей для роботи з апаратним забезпеченням, що в порівнянні, наприклад, з C++ зменшує швидкість роботи програм. За необхідності таких дій Java дозволяє викликати підпрограми, написані іншими мовами програмування [2].

Java широко використовується банківській сфері та великих компаніях,

завдяки своїй надійності та відмовостійкості. Підтримує зворотню сумісність версій.

Java перевірена часом, вдосконалена, та продовжує розвиватися спеціальною спільнотою розробників, архітекторів та ентузіастів, незважаючи на те, що її походження налічує майже два десятиліття.

Java можна використовувати для створення повноцінних програм, які можуть працювати на одному комп'ютері або розподіленими між серверами та клієнтами в мережі. Вона також може бути використана для створення невеликого програмного модуля або аплету для використання як частини веб-сторінки.

Вона призначена для розробки портативних високопродуктивних додатків для найширшого обсягу обчислювальних платформ, отже, забезпечує основні положення загальної доступності, а також взаємодії між платформами.

Java стала безцінною для розробників, дозволивши їм:

- писати програмне забезпечення на одній платформі та запускати його практично на будь-якій іншій платформі;
- створювати програми, які можуть працювати у веб-браузері та отримувати доступ до доступних веб-служб;
- розробляти серверні додатки для онлайн-форумів, магазинів, опитувань, обробки HTML-форм;
- поєднувати програми або сервіси для створення спеціально налаштованих програм або сервісів;
- писати потужні та ефективні програми для мобільних телефонів, віддалених процесорів, мікроконтролерів, бездротових модулів, датчиків, шлюзів, споживчих товарів та практично будь-якого іншого електронного пристрою.

2.1.2. Spring Framework

Spring Framework - це платформа Java, яка забезпечує всебічну підтримку інфраструктури для розробки програм Java [3].

Фреймворк допомагає розробляти системи зі слабкою зв’язаністю, що досягається за рахунок функцій Spring Inversion of Control (IoC), та високою зчепленістю за рахунок функцій Aspect oriented programming (AOP).

Spring пропонує ряд функцій, необхідних для розробки корпоративних додатків. Функції фреймворка розподілені по модулях, що надає можливість розробникам вибрати один або кілька модулів для інтеграції в залежності від необхідних функцій.

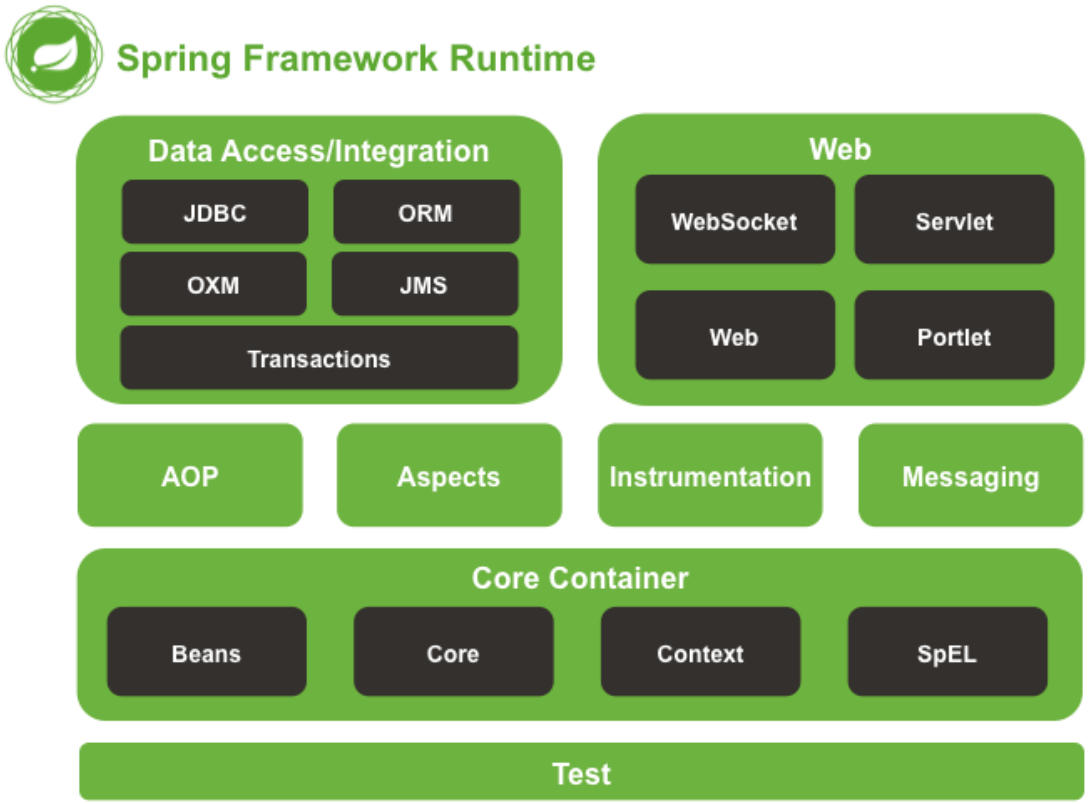


Рисунок 2.1 – Архітектура фреймворку Spring [3]

Core модуль надає функцію Dependency Injection (DI) також відому як Inversion of Control (IoC).

Beans модуль містить BeanFactory (шаблон фабрика), який створює та управляє життєвим циклом об’єктів (бінів), визначених у файлах конфігурації Spring або за допомогою анотацій. Завдяки ньому можлива інверсія контролю.

SpEL модуль забезпечує потужну мову виразів для запитів та маніпулювання графом об'єктів під час виконання.

Application Context надає інформацію про конфігурацію програми.

Так само, як BeanFactory, ApplicationContext завантажує біни, пов'язує їх разом і конфігурує їх певним чином. Але крім цього, ApplicationContext володіє додатковою функціональністю: розпізнання текстових повідомлень з файлів налаштування і відображення подій, які відбуваються в додатку різними способами.

Spring AOP забезпечує реалізацію аспектно-орієнтованого програмування, що дозволяє визначати методи-перехоплювачі для чіткого поділу коду, який реалізує функціональні можливості, які повинні бути розділені.

Spring Web допомагає у розробці веб-додатків. Цей модуль побудований на основі Application context модуля та пропонує веб-орієнтовані функції.

Spring MVC побудований на основі модуля Web та допоможе у розробці веб-додатків з застосуванням шаблону проектування MVC.

Spring JDBC дозволяє легко взаємодіяти з базою даних, надаючи абстракцію над низькорівневими завданнями JDBC, завдяки створенню з'єднань з базою даних, його звільнення і т. д.

Spring ORM допомагає інтегруватися інструментами об'єктно-реляційного відображення як Hibernate, iBatis і т. д.

2.1.2.1. Spring IoC

Контейнер IoC отримує метадані з XML-файлу, анотацій Java або коду Java. Контейнер отримує вказівки щодо того, які об’єкти створювати, настроювати та збирати з POJO, читаючи надані метадані конфігурації. Створені таким чином об’єкти мають назву Spring Beans.

Обов'язки контейнера IoC:

- створення бінів;
- зв’язування бінів з іншими;
- налаштування бінів;
- управління життєвим циклом біна від створення до знищення [4].

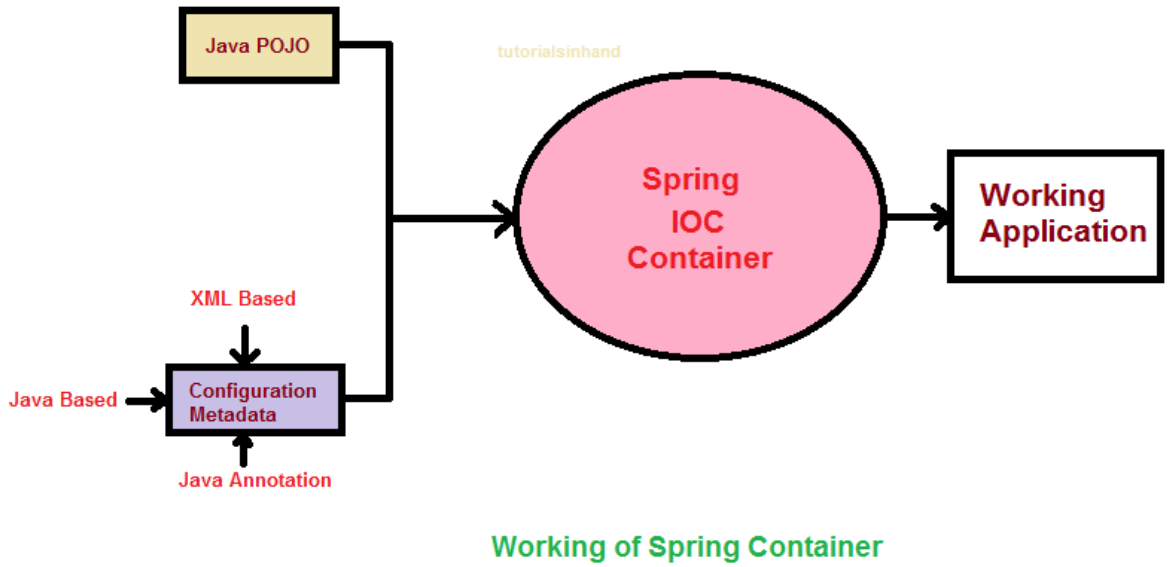


Рисунок 2.2 – Робота контейнера Spring IoC [4]

2.1.2.2. Spring Boot

Spring Boot - це проект, який побудований на основі Spring Framework. Він забезпечує простіший і швидший спосіб налаштування та запуску як простих, так і веб-програм [5].

Надає можливість почати роботу з мінімальними настройками без необхідності повної настройки конфігурації, так як та підключає багато налаштувань автоматично.

Переваги:

- легко розуміти і розробляти програми Spring;
- скорочує час розробки;
- підвищує продуктивність;
- менший поріг входження для початківців.

Цілі:

- уникнути складної конфігурації XML;
- спростити розробку готових до виробництва додатків;
- скоротити час розробки і запустити додаток самостійно;
- запропонувати більш простий спосіб почати роботу з додатком.

Функції:

- гнучкий спосіб настройки компонентів Java, конфігурацій XML і транзакцій бази даних;
- забезпечує потужну пакетну обробку і управляє кінцевими точками REST;
- все налаштовується автоматично, ручні настройки не потрібні;
- пропонує додаток на основі анотацій;
- спрощує управління залежностями;
- включає вбудований контейнер сервлетів (embedded Tomcat, Jetty);
- підтримує CLI.

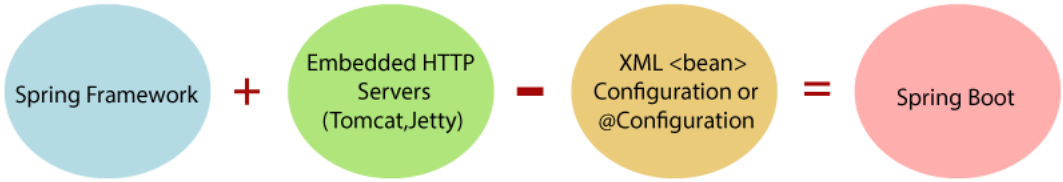


Рисунок 2.3 – Компоненти Spring Boot [5]

2.1.2.3. Spring MVC

Фреймворк Spring MVC забезпечує архітектуру шаблону Model - View - Controller (Модель - Відображення - Контролер) за допомогою слабо пов'язаних готових компонентів. Шаблон MVC розділяє аспекти додатку (логіку введення, бізнес-логіку і логіку UI), забезпечуючи при цьому вільний зв'язок між ними [6].

Логіка роботи Spring MVC побудована навколо DispatcherServlet, що приймає і обробляє з UI всі HTTP-запити і відповіді на них.

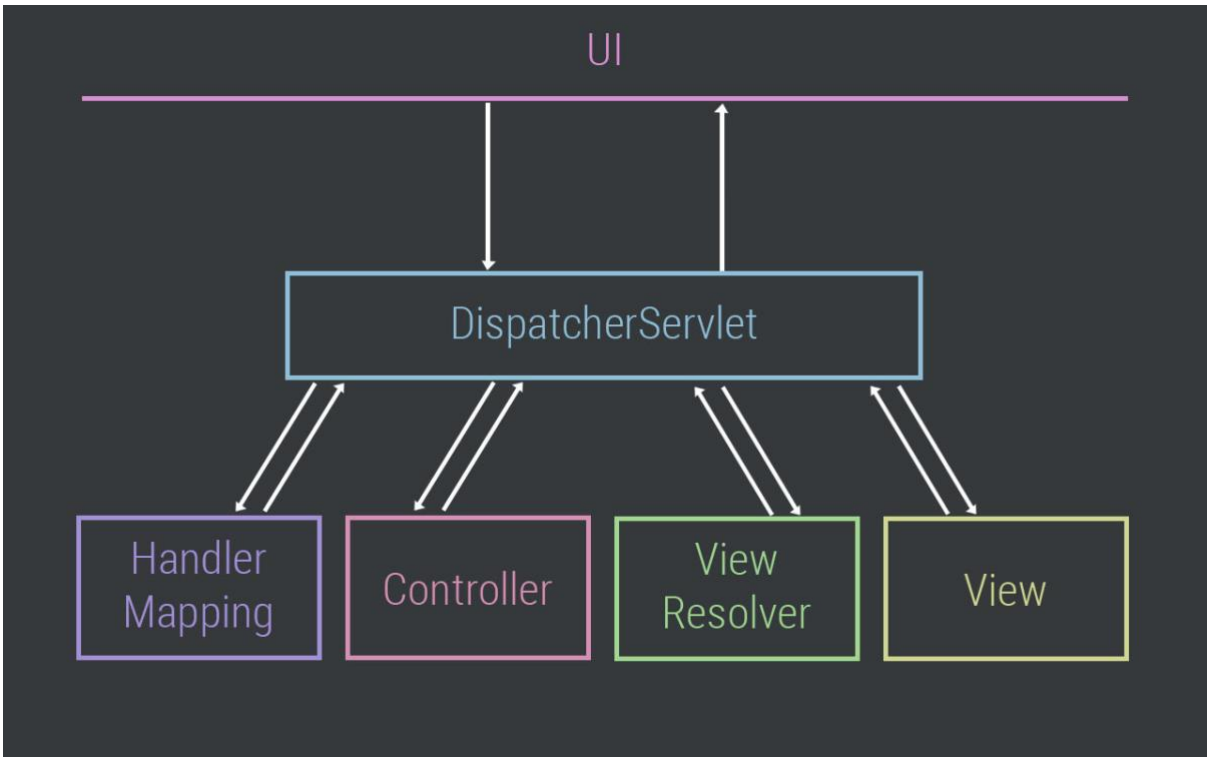


Рисунок 2.4 – Опис роботи DispatcherServlet [6]

При отримані HTTP-запиту виконується така послідовність дій:

- DispatcherServlet звертається до інтерфейсу HandlerMapping, який відповідає за визначення відповідного контролеру та наступне відправлення запиту до нього.

- Контролер приймає запит і запрошує відповідний метод (GET чи POST). Відповідно до бізнес-логіки метод повертає об’єкт ім’я вигляду в DispatcherServlet.

- За допомогою ViewResolver DispatcherServlet на підставі отриманого імені визначає, який вигляд потрібно використовувати.

- Після того, як вигляд створений, DispatcherServlet відправляє дані моделі у вигляді атрибутів в вигляд, який буде відображатися в браузері.

Всі компоненти, що були розглянуті вище, а саме, HandlerMapping, Controller і ViewResolver, є частинами інтерфейсу WebApplicationContext що наслідує ApplicationContext додаючи необхідні функції для створення веб-додатків.

2.1.2.4. Spring Security

Spring Security - це система безпеки, яка захищає корпоративні додатки на основі J2EE, забезпечуючи потужні, настроювані функції безпеки, такі як автентифікація та авторизація. Це фактичний стандарт захисту програм на основі Spring [7].

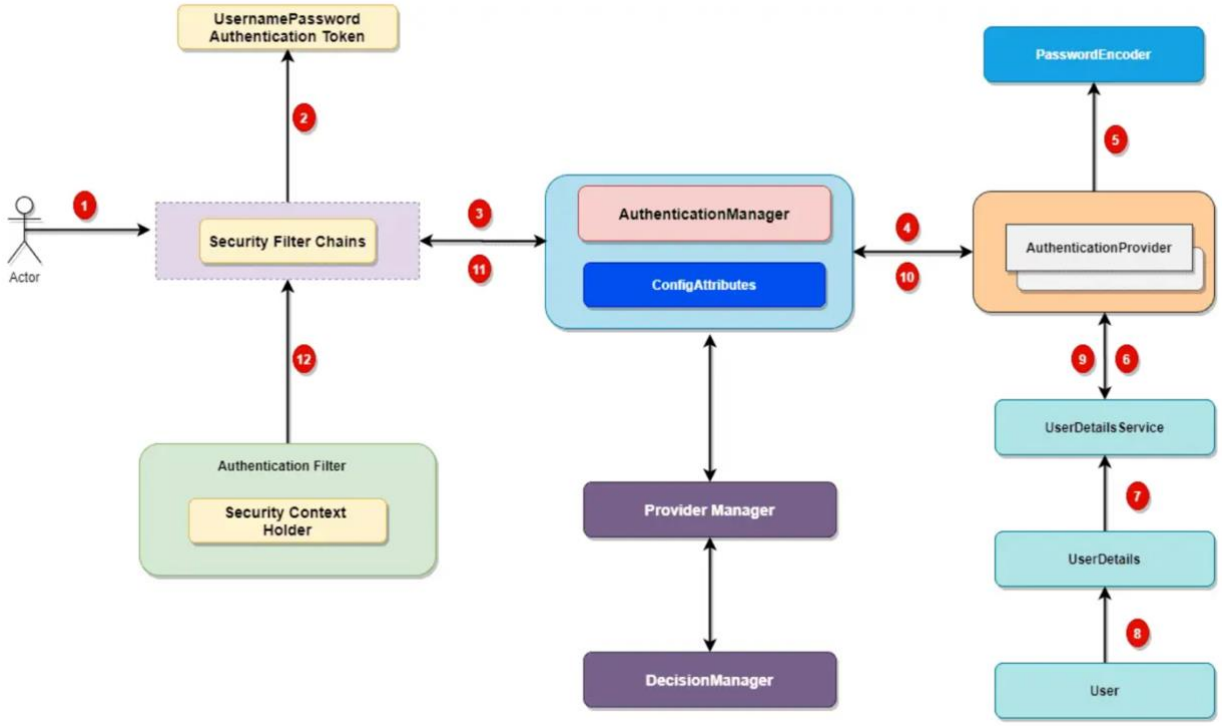


Рисунок 2.5 – Принцип роботи Spring Security [7]

Ключові об'єкти контексту Spring Security:

- в SecurityContextHolder міститься інформація про поточний контекст безпеки, з детальною інформацією про поточного користувача. Для зберігання такої інформації SecurityContextHolder використовує стратегію ThreadLocal, тобто контекст безпеки завжди буде доступний для методів, що виконуються в тому ж потоці.
- SecurityContext, містить об’єкт Authentication.
- Authentication відображає користувача (Principal) зі сторони Spring Security.
- GrantedAuthority представляє ролі, що наявні у користувача.

- UserDetails(містить ім'я користувача, пароль, ролі) надає необхідну інформацію для побудови об'єкта Authentication з додатків DAO.

- UserDetailsService, використовується для створення UserDetails об'єкту, реалізуючи loadUserByUsername(…) методу цього інтерфейсу. Далі об’єкт UserDetails може використовуватись контекстом Spring Security.

Стандартний сценарій аутентифікації:

- Користувачу буде запропоновано увійти в систему з власним ім'ям і паролем.

- Система звіряє пароль і підтверджує що він був введений правильно для даного користувача.

- Отримує контекстну інформацію для даного користувача (ролі та інші відповідні дані).

- Відбувається встановлення контексту безпеки для користувача.

- Користувач перенаправляється на виконання запрошеної операції, яка може бути захищена механізмом контролю доступу, що на підставі інформації поточного контексту безпеки перевіряє необхідні дозволи.

Сценарій аутентифікації в Spring Security:

- Ім'я користувача і пароль і формують UsernamePasswordAuthenticationToken (що наслідує Authentication).

- Токен передається об’єкту AuthenticationManager для перевірки на коректність.

- В разі успішної аутентифікації AuthenticationManager заповнює все необхідні дані для Authentication, та повертає екземпляр.

- Викликається SecurityContextHolder.getContext().setAuthentication(...), куди передається заповнений екземпляр Authentication, внаслідок чого встановлюється контекст безпеки.

Переваги Spring Security:

- можливо використовувати аутентифікацію з паролем, токеном, OAuth, LDAP;

- реалізовані шифрування PBKDF2, Bcrypt, Scrypt;
- надає захист від поширених атак;
- інтеграція з шаблонізаторами;

2.1.2.5. Spring Data JPA

JPA - це специфікація для зоб’єктно-реаляційного відображення об'єктів в додатку та збереження, читання та управління такими об’єктами.

Spring Data JPA не є JPA постачальником, а є специфікацією. Це фреймворк, який додає додатковий рівень абстракції зверху постачальника JPA. Він просто “приховує” Java Persistence API (і постачальника JPA) за своєю абстракцією репозиторію [8].

Реалізація специфікації JPA надається постачальниками JPA (Hibernate, Toplink, iBatis і т. д.)

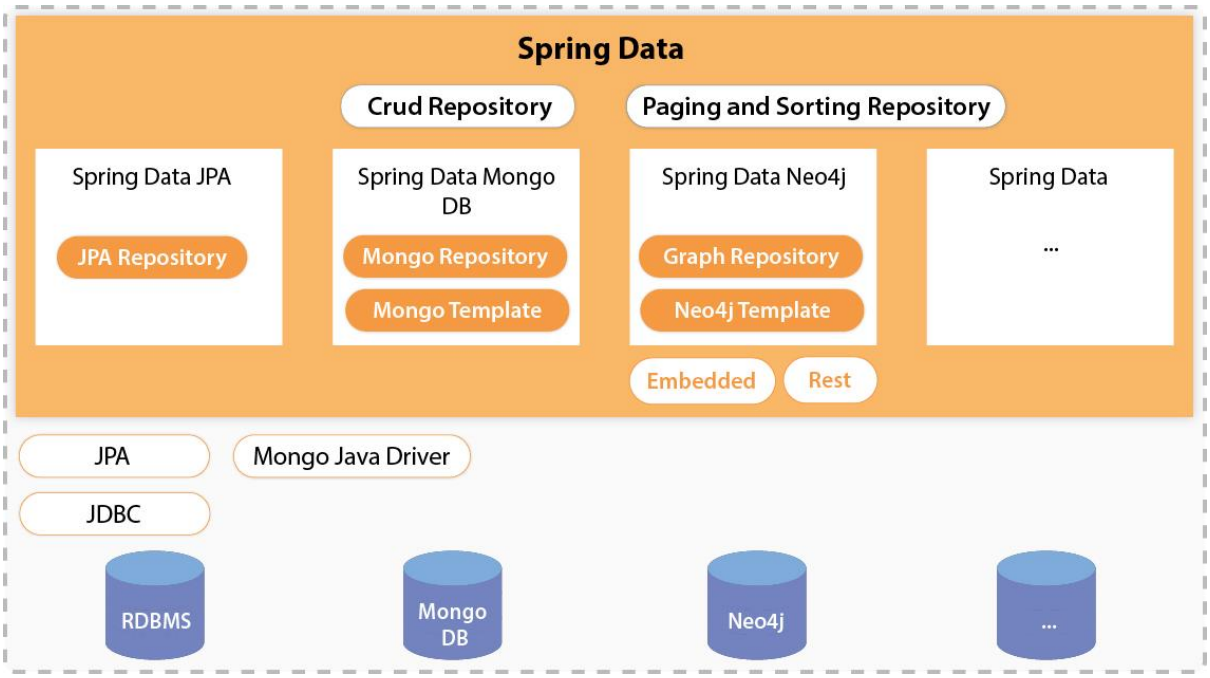


Рисунок 2.6 – Репозиторії в Spring Data JPA [8]

Можливості Spring Data JPA:

- Створювати та підтримувати репозиторії, створені за допомогою Spring і JPA;
- Підтримує запити JPA;
- Підтримує пакетне завантаження, сортування, динамічні запити;

- Підтримує XML відображення для сутностей та за допомогою анотацій;

- Зменшує розмір коду для загальних CRUD операцій за допомогою CrudRepository;

Необхідні компоненти:

- Драйвер JDBC, який дозволяє Java-додатку взаємодіяти з базою даних.

- Datasource надає всю технічну інформацію, необхідну для доступу до даних.

- Провайдер JPA (найпоширенішим є Hibernate)

- Spring Data JPA приховує використовуваний постачальник JPA за абстракцією сховища.

Java Persistence API використовується для управління, збереження і доступу до даних між об'єктами і базою даних.

2.1.3. Hibernate

Hibernate – засіб відображення між об'єктами та реляційними структурами (object-relational mapping, ORM) для платформи Java. Hibernate є вільним програмним забезпеченням, яке поширюється на умовах GNU Lesser General Public License. Hibernate надає легкий для використання каркас (фреймворк) для відображення між об'єктно-орієнтованою моделлю даних і традиційною реляційною базою даних [9].

Hibernate - це інструмент ORM, який реалізує специфікацію JPA. Він створює зв'язок між таблицями в базі даних і Java-класами і навпаки. Це позбавляє розробників від великої кількості рутинної роботи, в якій вкрай легко припустити помилку.

Переваги:

- Забезпечує простий API для отримання та запису Java-об'єктів з та в базу даних.

- Використовує стратегію fetching, що дозволяє мінімізувати доступ до БД.

- Дозволяє працювати з типами даних Java, замість SQL.

- Виконує створення зв'язків між Java-класами і таблицями БД за допомогою XML-файлів або анотацій.

- Якщо необхідно змінити БД, то достатньо лише внести зміни в XML-файли або анотації.

- Можливість писати універсальні SQL запити для всіх БД, варто лише замінити діалект БД.

- Механізм кешування, що значно роботу додатку, завдяки зменшенню кількості операцій в базу даних.

- Дає змогу додатку підтримувати декілька транзакцій одночасно, завдяки механізму блокування.

2.1.4. MySQL

MySQL — вільна система керування реляційними базами даних, яка була розроблена компанією “TcX” для підвищення швидкодії обробки великих баз даних. Ця система керування базами даних (СКБД) з відкритим кодом була створена як альтернатива комерційним системам. MySQL з самого початку була дуже схожою на mSQL, проте з часом вона все розширювалася і зараз MySQL — одна з найпоширеніших систем керування базами даних. Вона використовується, в першу чергу, для створення динамічних веб-сторінок, оскільки має чудову підтримку з боку різноманітних мов програмування [10]. Переваги:

- Забезпечує масштабованість, володіючи здатністю обробляти вбудовані додатки використовуючи менше місця навіть для величезних сховищ даних.

- MySQL дозволяє виконати повну настройку та додати унікальні вимоги, завдяки відкритому вихідному коду.

- Архітектура механізму зберігання дозволяє налаштовувати сервер бази даних спеціально для конкретних додатків, в результаті чого досягаються приголомшливі показники продуктивності. Завдяки швидкісним утилітам

завантаження, кешах пам'яті, повнотекстових індексах і іншим механізмам підвищення продуктивності MySQL пропонує всі необхідні засоби для сьогоднішніх систем.

- Пропонує безліч варіантів високої доступності від конфігурацій високошвидкісний реплікації master/slave, до спеціалізованих серверів кластерів, що забезпечують миттєве перемикання при відмові.

- Включають повну підтримку транзакцій ACID (атомарні, узгоджені, ізольовані, надійні), необмежене блокування на рівні рядків, можливість розподілених транзакцій і підтримку транзакцій з декількома версіями, при яких зчитувачі ніколи не блокують записи і навпаки.

- Підходить для веб-сайтів з високою відвідуваністю завдяки продуктивному механізму запитів, швидкій можливості вставки даних і підтримці спеціалізованих веб-функцій, таких як швидкий повнотекстовий пошук.

- Надає механізми, що гарантують, що тільки авторизовані користувачі мають доступ до сервера бази даних, з можливістю блокування користувачів до рівня клієнтської машини. Передбачена детальна структура привілеїв об'єктів, так що користувачі бачать лише відповідні дані, а функції шифрування гарантують, що конфіденційні дані захищені від несанкціонованого перегляду. Утиліти резервного копіювання та відновлення дозволяють виконувати повне логічне і фізичне резервне копіювання, а також повне відновлення і відновлення на певний момент часу.

- Забезпечує підтримку збережених процедур, тригерів, функцій, уявлень, курсорів, ANSI-стандартного SQL і багато чого іншого. MySQL також надає з'єднувачі і драйвери (ODBC, JDBC і т. д.).

- Пропонує можливість швидкого запуску.

2.1.5. Maven

Maven - це потужний інструмент управління проектами, заснований на POM (об'єктній моделі проекту). Він використовується для збірки проектів, залежностей і документації. Він спрощує процес збірки.

Це інструмент, який можна використовувати для створення і управління будь-яким Java-проєктом.

За принципами роботи кардинально відрізняється від Apache Ant, та має простіший вигляд щодо build-налаштувань, яке надається в форматі XML. XML-файл описує проєкт, його зв'язки з зовнішніми модулями і компонентами, порядок будування (build), папки та необхідні плагіни [11].

Maven складається з трьох компонентів:

- POM: файл, що описує проект Maven і його залежності. Кожен проект, який використовує Maven, має файл POM (об'єктної моделі проекту) в своєму кореневому каталозі. Pom.xml описує залежності проекту і вказує, як його збирати.

- Каталог: стандартизований формат для опису проекту Maven в POM.

Він реалізує те угоду про конфігурацію. Замість того, щоб вимагати від розробників визначення макета і ручної настройки компонентів для кожного нового проекту (як у випадку з make-файлом і Ant), Maven встановлює загальну структуру проекту і пропонує стандартний формат файлу для опису того, як він працює. Можна просто описати свої вимоги, а Maven викличе залежності і налаштовує проект.

- Сховища: де зберігається і виявляється додаткове програмне забезпечення. Maven використовує централізовані репозиторії як для виявлення, так і для публікації пакетів проекту як залежностей. Коли розробник посилаєтеся на залежність в своєму проекті, Maven виявить її в централізованому сховищі, завантажить в локальний репозиторій і встановить в проект.

Фази збірки:

- validate: перевірити правильність проекту і доступність всієї необхідної інформації;

- compile: скомпілювати вихідний код проекту;

- test: протестувати скомпільований вихідний код за допомогою відповідного середовища модульного тестування.;

- package: упакувати скомпільований код в розповсюджуваний формат, такий як JAR;

- integration-test: обрати і розгорнути пакет, в середовищі, де можна запускати інтеграційні тести;

- verify: запустити перевірки, щоб переконатися, що пакет валідний і відповідає критеріям якості;

- install: встановити пакет в локальний репозиторій для використання в якості залежності в інших проектах локально;

- deploy: виконується в середовищі інтеграції або випуску, копіює фінальний пакет в віддалений репозиторій для спільного використання з іншими розробниками і проектами.

2.1.6. Thymeleaf

Thymeleaf - це механізм шаблонів Java XML / XHTML / HTML5, який може працювати як у веб (на основі сервлетів), так і в офлайн середовищі. Він більше підходить для обслуговування XHTML / HTML5 на рівні перегляду веб-додатків на основі MVC, але він може обробляти будь-який XML-файл навіть в автономному середовищі. Він забезпечує повну інтеграцію Spring Framework. [12].

Він застосовує набір перетворень до файлів шаблонів для відображення даних або тексту, створених додатком. Він підходить для обслуговування XHTML / HTML5 в веб-додатках.

Thymeleaf надає стильний і добре сформований спосіб створення шаблонів. Він заснований на тегах і атрибутах XML. Ці теги XML визначають

виконання визначеної логіки в DOM (об'єктній моделі документа) замість явного написання цієї логіки у вигляді коду всередині шаблону. Це заміна JSP.

Архітектура Thymeleaf дозволяє швидко обробляти шаблони, завдяки кешуванню проаналізованих файлів. Він використовує мінімально можливу кількість операцій введення-виведення під час виконання.

JSP трохи схожий на HTML. Але він не повністю сумісний з HTML, як Thymeleaf. Можемо відкривати і відображати файл шаблону Thymeleaf в браузері, а файл JSP - немає.

Thymeleaf підтримує вираження змінних ($ {...}), такі як Spring EL, і виконується для атрибутів моделі, вираження зірочки (* {...}) виконуються в компоненті підтримки форми, хеш-вирази (# {...}) призначені для інтернаціоналізація і вирази посилань (@ {...}) перезаписують URL-адреси.

Як і JSP, Thymeleaf гарний вибір для електронних листів в форматі Rich HTML.

Може обробляти шість типів шаблонів (також відомих як режим шаблону), а саме:

- XML;
- Valid XML;
- XHTML;
- Valid XHTML;
- HTML5;
- Legacy HTML5.

За винятком застарілого режиму HTML5, всі перераховані вище режими відносяться до чітко певних файлів XML. Це дозволяє обробляти файли HTML5 з такими функціями, як автономні теги, атрибути тегів без значення або без лапок. Для обробки файлів в цьому конкретному режимі Thymeleaf виконує перетворення, яке перетворює файли в правильно сформований файл XML (дійсний файл HTML5).

Також дозволяє визначати власний режим, вказуючи обидва способи синтаксичного аналізу шаблонів в цьому режимі. Таким чином, все, що можна змоделювати як дерево DOM, може ефективно оброблятися Thymeleaf як шаблон.

2.1.7. Bootstrap

Bootstrap – це безкоштовний набір інструментів з відкритим кодом, призначений для створення веб-сайтів та веб-додатків, який містить шаблони CSS та HTML для типографіки, форм, кнопок, навігації та інших компонентів інтерфейсу, а також додаткові розширення JavaScript. Він спрощує розробку динамічних веб-сайтів і веб-додатків [13].

Після додавання в проект Bootstrap надає базові визначення стилів для всіх елементів HTML. Результатом є однаковий зовнішній вигляд тексту, таблиць і елементів форм у всіх веб-браузерах. Крім того, розробники можуть скористатися перевагами класів CSS, визначених у Bootstrap, для подальшої настройки зовнішнього вигляду їх вмісту. Наприклад, в Bootstrap передбачені заголовки сторінок та таблиці світлого і темного кольорів.

Bootstrap також поставляється з декількома компонентами JavaScript у вигляді плагінів jQuery. Вони надають додаткові елементи призначеного для користувача інтерфейсу, такі як спливаючі підказки, діалогові вікна і каруселі.

Найбільш помітними компонентами Bootstrap є його компоненти макета. Базовий компонент макета називається контейнер, так як в нього поміщається будь-який інший елемент на сторінці. Він використовує одну з п'яти фіксованих значень ширини, залежно від розміру екрана, на якому відображається сторінка:

- < 576 px;
- 576-768 px;
- 768-992 px;
- 992-1200 px;
- > 1200 px.

Після розміщення контейнера інші компоненти макета можуть визначати рядки і стовпці.

Попередньо скомпільована версія Bootstrap доступна у вигляді одного файлу CSS і трьох файлів JavaScript, які можна легко додати в будь-який проект. Також у розробників є можливість використовувати необроблену форму, що дозволяє реалізувати подальшу настройку і оптимізацію розміру.

2.2. Вибір архітектури

2.2.1. MVC: Model View Controller

Model-View-Controller – схема розділення даних додатків, користувацького інтерфейсу та керуючої логіки на три окремих компонени: модель, представлення та контролер - таким чином, що модифікація кожного компонента може здійснюватись незалежно [14].

Кконтролер і відображення залежать від моделі, але модель ніяк не залежить від цих двох компонент.

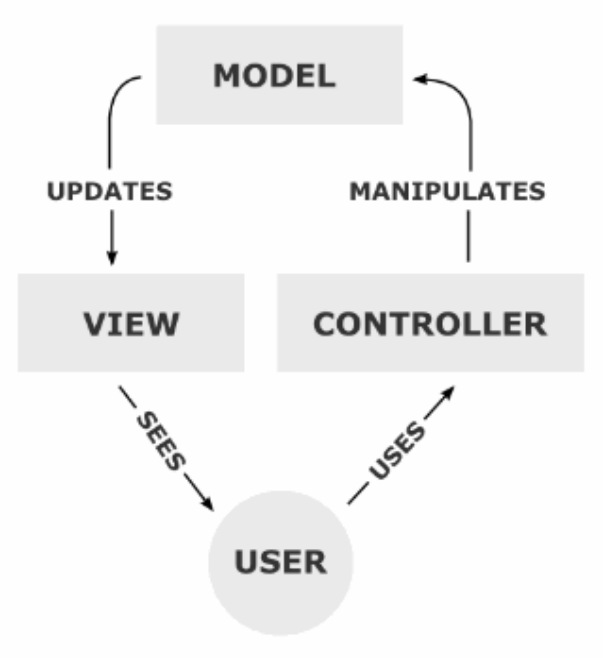


Рисунок 2.7 – Схема архітектури MVC [14]

Model:

- не залежить від відображення і контролера;
- містить в собі функціональну бізнес-логіку;

- може бути як просто шаром даних (DAO, XML-файл), так і менеджером бази даних, чи просто логікою програми.

View:

- відображає дані моделі (HTML-сторінка, Windows Form);
- не може впливати на модель;
- іноді може мати код, що реалізує деяку бізнес-логіку.

Controller:

- контролер визначає яке відображення має бути відображено в даний момент;
- події відображення можуть вплинути на контролер, а контролер може вплинути на модель і визначити інше відображення;
- у контролера може бути багато відображень.

ВИСНОВКИ ДО РОЗДІЛУ 2

У даному розділі були проаналізовані технології та архітектури, що підходять для розробки системи для управління курсами образотворчого мистецтва. Були обрані наступні технології та сфери їх призначення:

- Java – мова програмування;
- Spring Boot – для створення веб-додатків;
- Spring Security – для забезпечення безпеки;
- Spring Data JPA – для доступу до бази даних;
- Hibernate – для об'єктно-реляційного відображення;
- Thymeleaf – шаблонізатор;
- MySQL – база даних;
- Maven – система збірки.

РОЗДІЛ 3 РЕАЛІЗАЦІЯ ДОДАТКУ

3.1. Структура бази даних

- course – таблиця для зберігання курсів образотворчого мистецтва.
- instructor – таблиця для зберігання персональних даних вчителів.
- student – таблиця для зберігання персональних даних учнів.
- course_student – таблиця для зберігання співвідношень курсів та студентів.
- course_day – таблиця для зберігання співвідношень курсів та можливих днів відвідування.
- day – таблиця для зберігання можливих днів відвідування.
- course_discipline – таблиця для зберігання співвідношень курсів та дисциплін.
- discipline – таблиця для зберігання можливих дисциплін.
- photo – таблиця для зберігання даних про фотографії робіт студентів.
- payment – таблиця для зберігання даних про покупки курсів.

3.2. Java-класи проекту

Таблиця 3.1 – Java-класи проекту

№ з/п	Позначення	Призначення
1	LoginController	Контролер, що відповідає за авторизацію студентів та викладачів та реєстрацію нових студентів. Також відповідає за відновлення паролю шляхом відправлення на електронну пошту посилання та наступне створення нового паролю.

Продовження таблиці 3.1

2	StudentController	Контролер для пошуку та отримання детальної інформації про студентів. Також забезпечує роботу з персональним кабінетом студентів, перегляд та редагування власних даних, загрузку фотографії профілю.
3	InstuctorController	Контролер для пошуку та отримання детальної інформації про викладачів. Також забезпечує роботу з персональним кабінетом викладачів, перегляд та редагування власних даних, загрузку фотографії профілю.
4	CourseController	Контролер, який відповідає за пошук та отримання детальної інформації по курсам.
5	InstructorCourseController	Контролер, що забезпечує роботу викладачів з курсами. Для перегляду, створення, редагування та видалення власних курсів.
6	StudentCourseController	Контролер, що забезпечує роботу студентів з курсами. Для перегляду власних курсів, історії платіжок та запису на нові курси.
7	GalleryController	Контролер для пошуку, перегляду та загрузки фотографій виконаних робіт в онлайн-галерею.
8	SecurityServiceImpl	Сервіс, який відповідає за авторизацію та отримання даних поточного користувача.
9	UserDetailsServiceImpl	Сервіс, який забезпечує отримання даних користувача по унікальному ідентифікатору.

Продовження таблиці 3.1

10	PasswordResetServiceImpl	Сервіс, який відповідає за відновлення паролю користувача.
11	UserServiceImpl	Сервіс для отримання інформації, створення, редагування та видалення учнів та вчителів.
12	StudentSearchServiceImpl	Сервіс для пошуку студентів по курсам.
13	CourseSearchServiceImpl	Сервіс для пошуку курсів.
14	CourseServiceImpl	Сервіс що забезпечує створення, редагування та видалення курсів викладачами. Також відповідає за можливість запису студентів на курс.
15	DateSercviceImpl	Сервіс для роботи з датою.
16	DayServiceImpl	Сервіс для роботи з днями тижнів.
17	DisciplineServiceImpl	Сервіс для роботи з дисциплінами курсів.
18	PaymentServiceImpl	Сервіс для роботи з платіжками по курсах.
19	PhotoServiceImpl	Сервіс, що відповідає за створення та пошук фотографій робіт в онлайн-галереї.
20	EmailServiceImpl	Сервіс, що забезпечує відправку емейл-повідомлень.
21	LoadPhotoServiceImpl	Сервіс для зчитування та запису фалів фотографій студентів, вчителів, курсі та онлайн-галереї.
22	PageableServiceImpl	Сервіс, що відповідає за пагінацію результатів.
23	CourseRepository	Репозиторій для отримання даних курсів.
24	StudentRepository	Репозиторій для отримання даних студентів.
25	InstructorRepository	Репозиторій для отримання даних інструкторів.

Продовження таблиці 3.1

26	PhotoRepository	Репозиторій для отримання даних онлайн-галереї.
27	DateRepository	Репозиторій для отримання даних дат.
28	DayRepository	Репозиторій для отримання даних днів тижнів.
29	DisciplineRepository	Репозиторій для отримання даних дисциплін.
30	CustomUserRepository	Репозиторій для отримання даних авторизованих користувачів та токену для відновлення паролю.
31	PaymentRepository	Репозиторій для отримання даних платіжок.
32	InitialDataLoader	Сервіс, що дозволяє загрузити початкові дані в базу даних.
33	ArtSchoolApplication	Клас, що відповідає за запуск веб-додатку.
34	Course	Модель курсу.
35	Date	Модель дати.
36	Day	Модель дня тижня.
37	Discipline	Модель дисципліни.
38	Payment	Модель платіжки.
39	CustomUser	Загальна модель користувача.
40	Student	Модель студента.
41	Instructor	Модель викладача.
42	PasswordResetToken	Модель для відновлення паролю.
43	Photo	Модель фотографії.
44	CourseForm	Форма для створення курсу.
45	ProfileForm	Форма для створення профілю користувача.
46	SearchCourseForm	Форма для пошуку курсу.
47	SearchPhotoForm	Форма для пошуку фотографії.

Продовження таблиці 3.1

48	SignUpInstructorForm	Форма для реєстрації вчителя.
49	SignUpStudentForm	Форма для реєстрації студента.
50	Audience	Енумерація аудиторії.
51	Gender	Енумерація статі.
52	UserRole	Енумерація ролі користувача.

3.3. HTML-сторінки проекту

Таблиця 3.2 – HTML-сторінки проекту

№ з/п	Позначення	Призначення
1	index	Головна сторінка додатку.
2	error	Сторінка що відображає помилку в системі, за наявності.
3	login	Сторінка для авторизації користувача.
4	forgot_password	Сторінка для вказання емейл-адреси для надсилання токену для відновленню паролю.
5	reset_password	Сторінка для введення нового паролю.
6	profile	Сторінка профілю користувача.
7	profile_by_uuid	Сторінка профілю користувача по унікальному ідентифікатору.
8	edit_profile	Сторінка для редагування власного профілю.
9	create_instructors	Сторінка для сворення аккаунту нового вчителя.
10	users	Сторінка всіх користувачів (студентів чи викладачів).
11	all_courses	Сторінка всіх курсів.
12	course	Сторінка детальної інформації про курс.

Продовження таблиці 3.2

13	create_course	Сторінка для створення курсу.
14	edit_course	Сторінка для редагування курсу.
15	user_courses	Сторінка для відображення курсів користувача (курси на які записаний студент чи курси що створив інструктор).
16	user_payments	Сторінка з історією платіжок по курсах.
17	gallery	Сторінка онлайн-галереї.
18	user_gallery	Сторінка онлайн-галереї з фотографіями загруженими користувачем.
19	add_photo	Сторінка для додавання нової фотографії в онлайн-галерею.
20	styles	Фрагмент, що містить стилі.
21	header	Фрагмент з навігаційним меню.
22	user_menu	Фрагмент додаткового меню авторизованого користувача.
23	users	Фрагмент з основною інформацією про користувача.
24	profile	Фрагмент з детальною інформацією про користувача.
25	courses	Фрагмент з детальною інформацією про курс.
26	photos	Фрагмент з детальною інформацією про фотографію.
27	payment	Фрагмент з історією платіжок користувача.
28	search_course	Фрагмент для пошуку курсів.
29	search_student	Фрагмент для пошуку студентів.
30	search_instructor	Фрагмент для пошуку вчителів.
31	search_photo	Фрагмент для пошуку фотографій.

3.4. JS-файли проекту

Таблиця 3.3 – JS-файли проекту

№ з/п	Позначення	Призначення
1	bootstrap-select	Скрипт для настройки плагіну для множинного вибору.
2	date_range_picker	Скрипт для настройки плагіну для вибору дати.
3	clock_picker	Скрипт для настройки плагіну для вибору часу.
4	x_editable	Скрипт для настройки плагіну для заміни статусу коритсувачів.
5	form_validation	Скрипт для активації валідації форм.
6	change_gender_images	Скрипт для заміни фотографії профілю користувачів за замовчуванням відповідно до статі.

3.5. Варіанти використання системи

Можливості неавторизованого користувача:

- переглянути головну сторінку;
- зареєструватися в якості студента;
- авторизуватися використовуючи свій емейл та пароль;
- відновити забутий пароль, якщо обліковий запис вже був створений для вказаного емейлу.
- переглянути та виконати пошук курсів, переглянути детальну інформацію про кожен;
- переглянути та виконати пошук учнів, переглянути детальну інформацію про кожного;
- переглянути та виконати пошук викладачів, переглянути детальну інформацію про кожного;

- переглянути роботи онлайн-галереї.

Можливості студента:

- переглядати та редагувати особисті дані;
- записатись на курс;
- переглянути курси на які вже записався та детальну інформацію про кожен;
- скасувати запис на курс;
- поновити запис на курс;
- переглянути історію платіжок по курсах;
- додати фотографії власних робіт в онлайн-галерею;
- переглянути свої фотографії в онлайн-галереї;
- вийти з облікового запису.

Можливості викладача:

- переглядати та редагувати особисті дані;
- зареєструвати іншого викладача;
- створювати та редагувати власні курси;
- переглядати власні курси та детальну інформацію про кожен;
- додати фотографії власних робіт чи робіт студентів в онлайн-галерею;
- переглянути свої фотографії в онлайн-галереї;
- вийти з облікового запису.

ВИСНОВОК ДО РОЗДІЛУ 3

У даному розділі було проведено опис структури сервера, його модулі та класи, які використовуються для системи управління курсами образотворчого мистецтва, що показує зрозумілу структуру проекту та його легке розширення. Структура серверу розроблена на основі шаблону MVC, який описаний в попередньому розділі.

Також, показана схема бази даних, де можна переглянути всі таблиці, їх зв’язки та дані, які там зберігаються. Кожна таблиця коротко описана.

РОЗДІЛ 4 ВПРОВАДЖЕННЯ ТА ОГЛЯД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1. Розгортання програмного забезпечення

Для розгорнення веб-додатку для управління курсами образотворчого мистецтва необхідна наявність JRE 11+ версії, та налаштована база даних MySQL. Інтерфейс користувача вбудований в серверну частину і не потребує ніяких налаштувань.

4.2. Огляд інтерфейсу системи

Розглянемо сторінки веб-додатку для управліннями курсами образотворчого мистецтва у порядку, в якому їх бачитиме користувач.

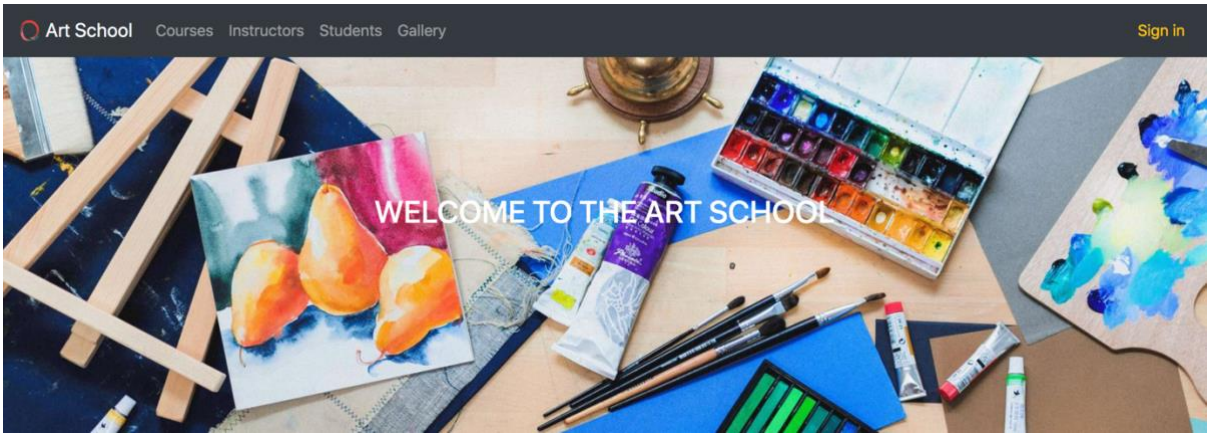


Рисунок 4.1 – Інтерфейс екрана привітання

При вході на сайт відображається привітання користувача та є можливість авторизуватись та зареєструватись.

Користувач має можливість ввести емейл та пароль для авторизації.

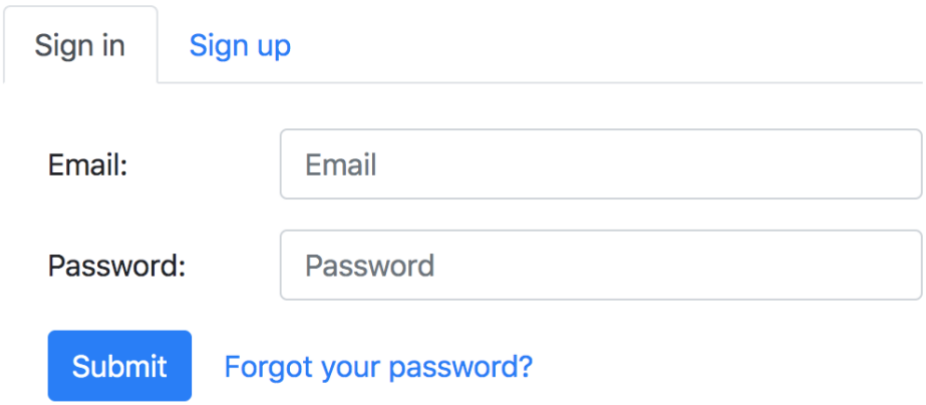


Рисунок 4.2 – Інтерфейс авторизації

Далі буде виконана валідація вхідних даних. Якщо дані не відповідають запрошеним, поля для вводу будуть обведені червоним, а якщо введено правильно – зеленим. Після чого користувач може виправити помилку та спробувати ще раз.

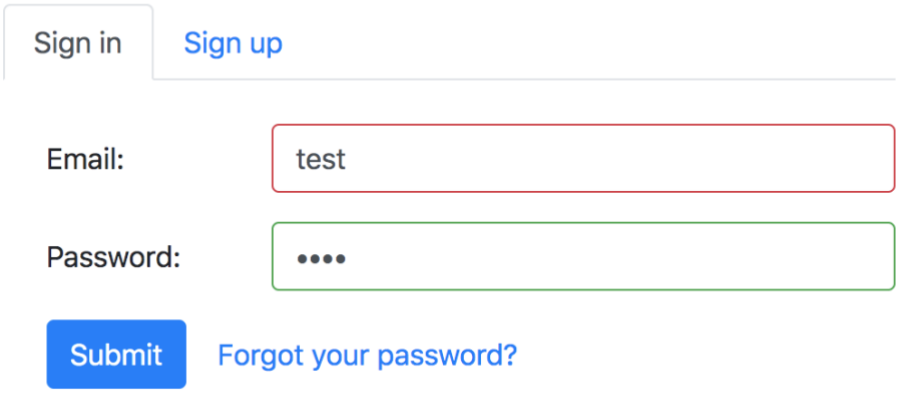


Рисунок 4.3 – Користувач ввів некоректний емейл

Після успішної валідації, дані будуть відправлені на сервер, де відбудеться перевірка паролю. Якщо дані не відповідають, користувачу буде показано помилку та надана можливіть заново ввести дані.

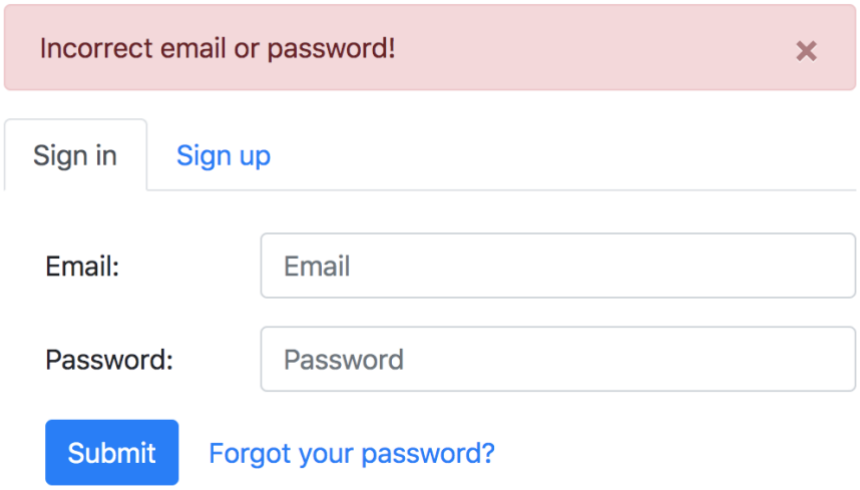


Рисунок 4.4 – Користувач ввід неправильний емейл чи пароль

Якщо користувач забув свій пароль, є можливість його відновити. Для цього потрібно вказати власний емейл з яким пов’язаний створений раніше обліковий запис.

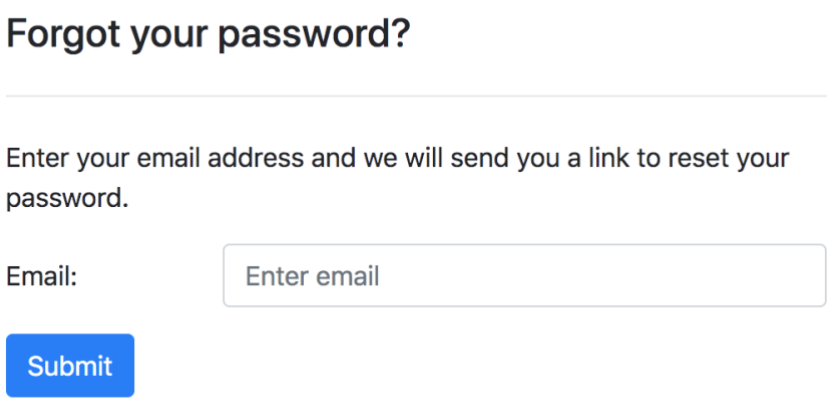


Рисунок 4.5 – Інтерфейс відновлення паролю

За умови успішної валідації емейлу, користувачу буде відправлено повідомлення з посиланням на відновлення паролю.

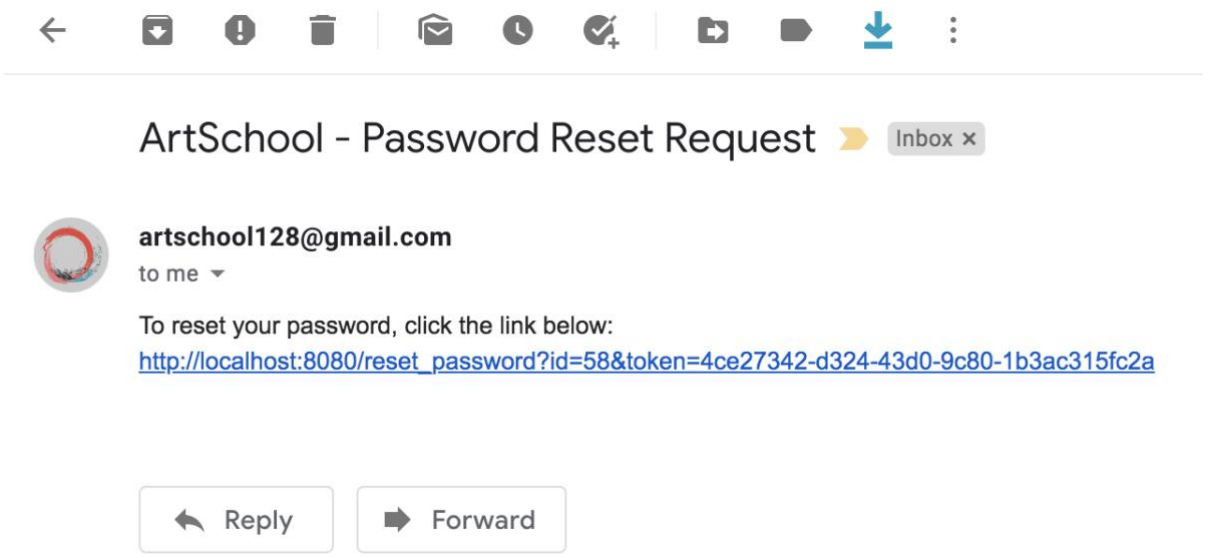


Рисунок 4.6 – Емейл з посиланням на відновлення паролю

Після переходження по посиланню користувачу буде запропоновано створити новий пароль, що може бути використаний для подальшої авторизації.

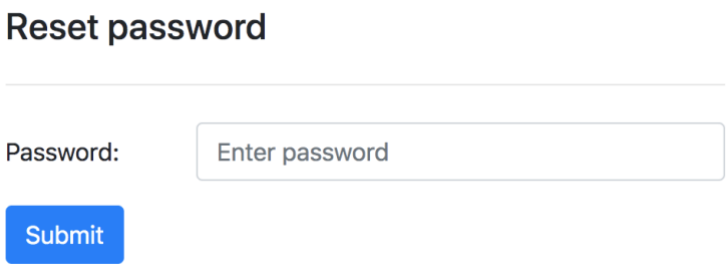


Рисунок 4.7 – Інтерфейс створення нового паролю

Користувач також може зареєструватись в системі в якості студента, якщо в нього немає облікового запису.

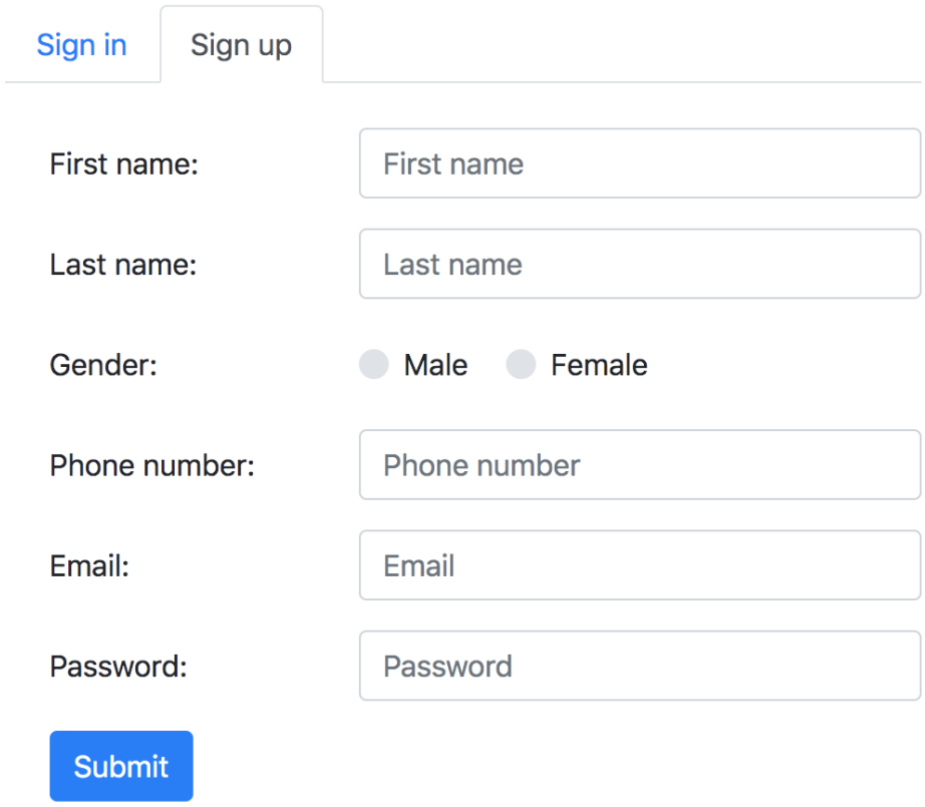


Рисунок 4.8 – Сторінка реєстрації студента

В разі успішної валідації введених даний, буде сворений обліковий запис студента. Далі користувача буде перенаправлено на сторінку особистого кабінету, де він може змінити фото профілю та інші особисті дані.

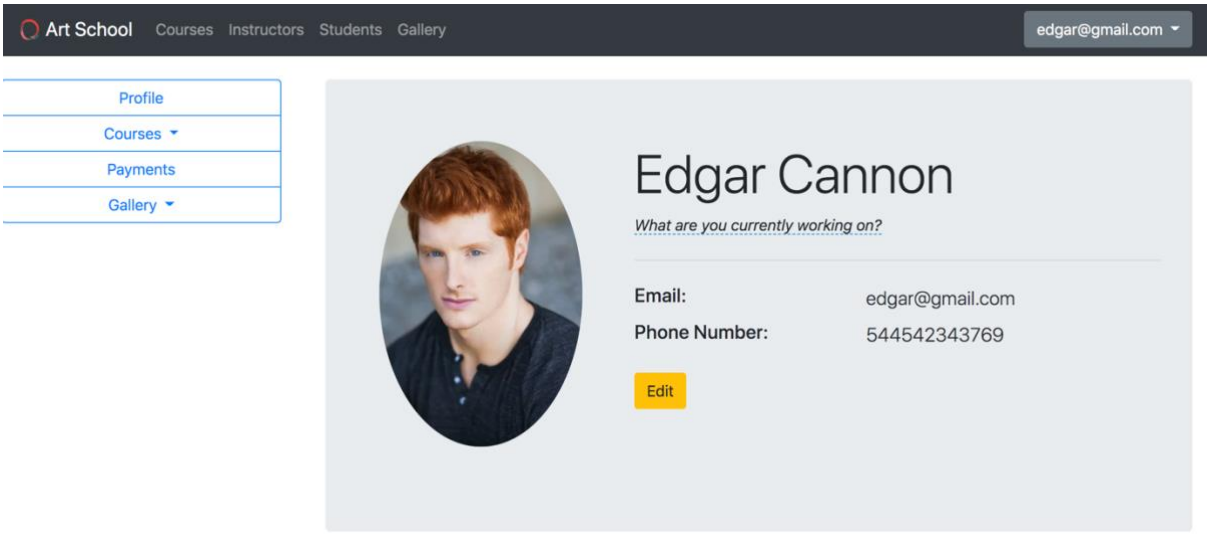


Рисунок 4.9 – Особистий кабінет студента

Користувач може виконати пошук курсів з використанням

різноманітних фільтрів, таких як назва, дисципліна, аудиторія, викладач, ціна та день проведення.

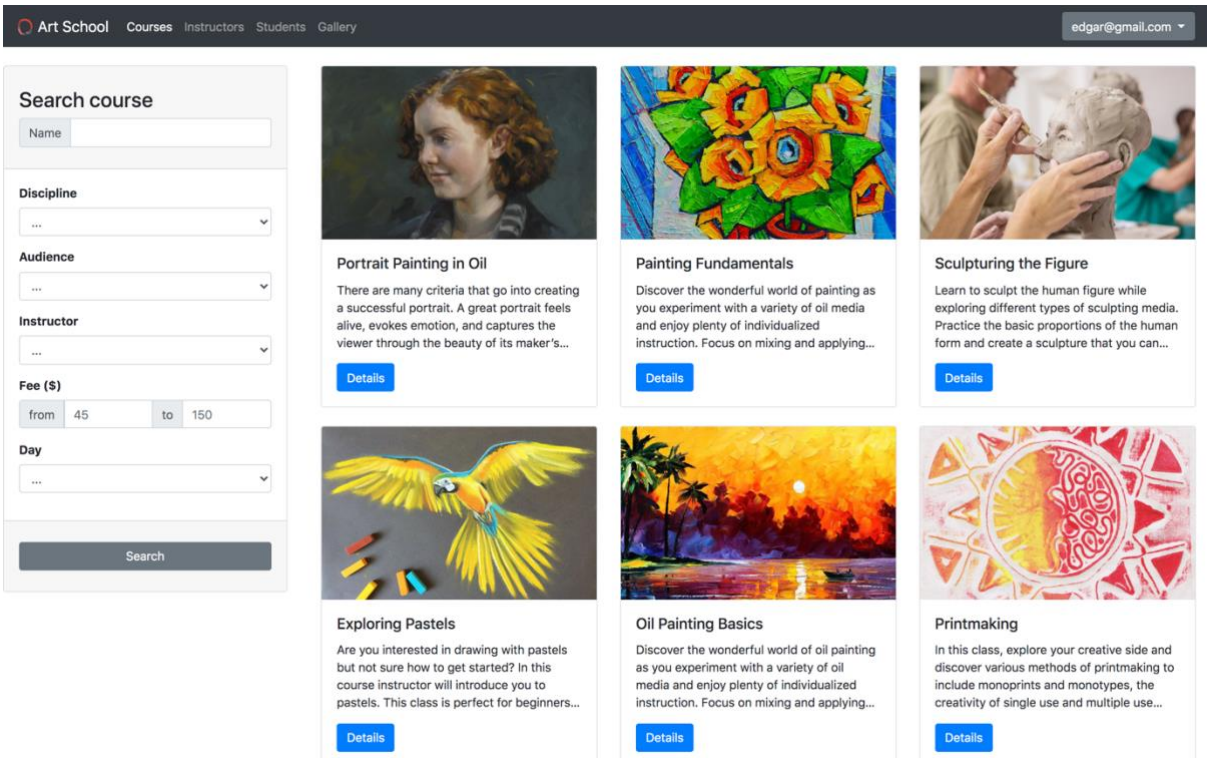


Рисунок 4.10 – Курси образотворчого митсецтва

Та переглянути інформацію про кожен курс, детальний опис, викладача та інших студентів.

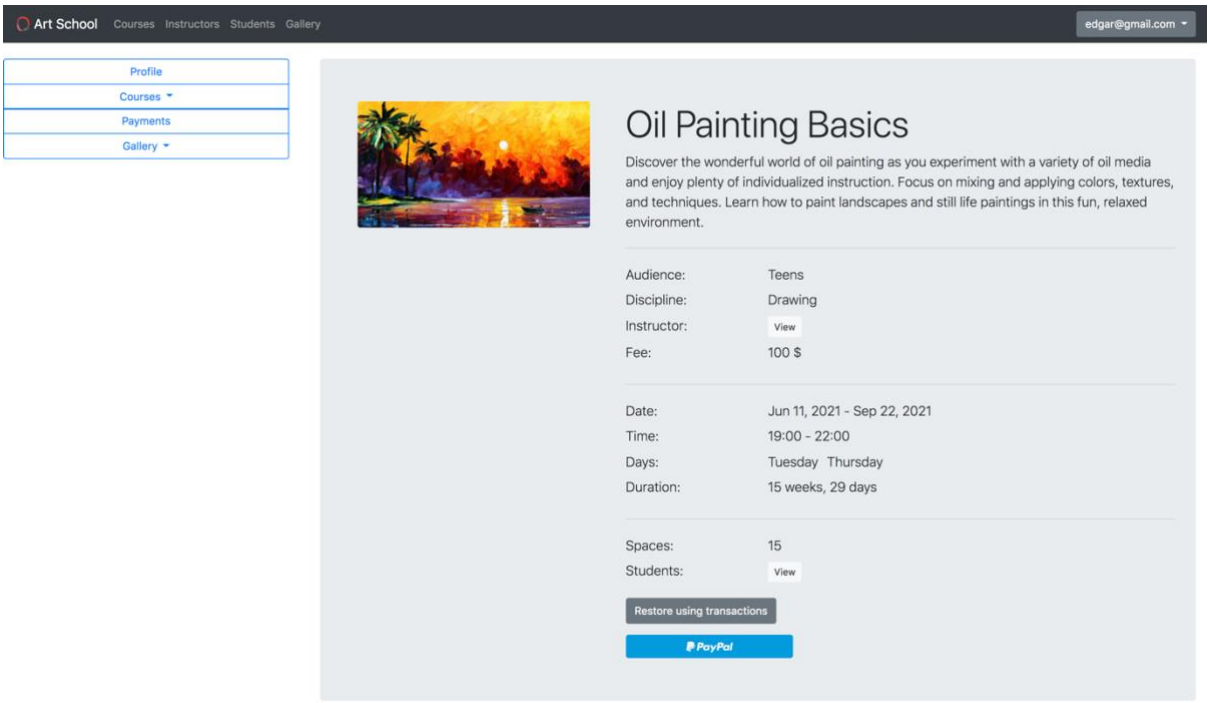


Рисунок 4.11 – Детальний опис курсу

Після вибору курсу студент може записатись та сплатити онлайн використовуючи PayPal. Буде відкрито модальне вікно для проведення платіжної операції, де користувачу буде потрібно авторизуватися в системі PayPal.

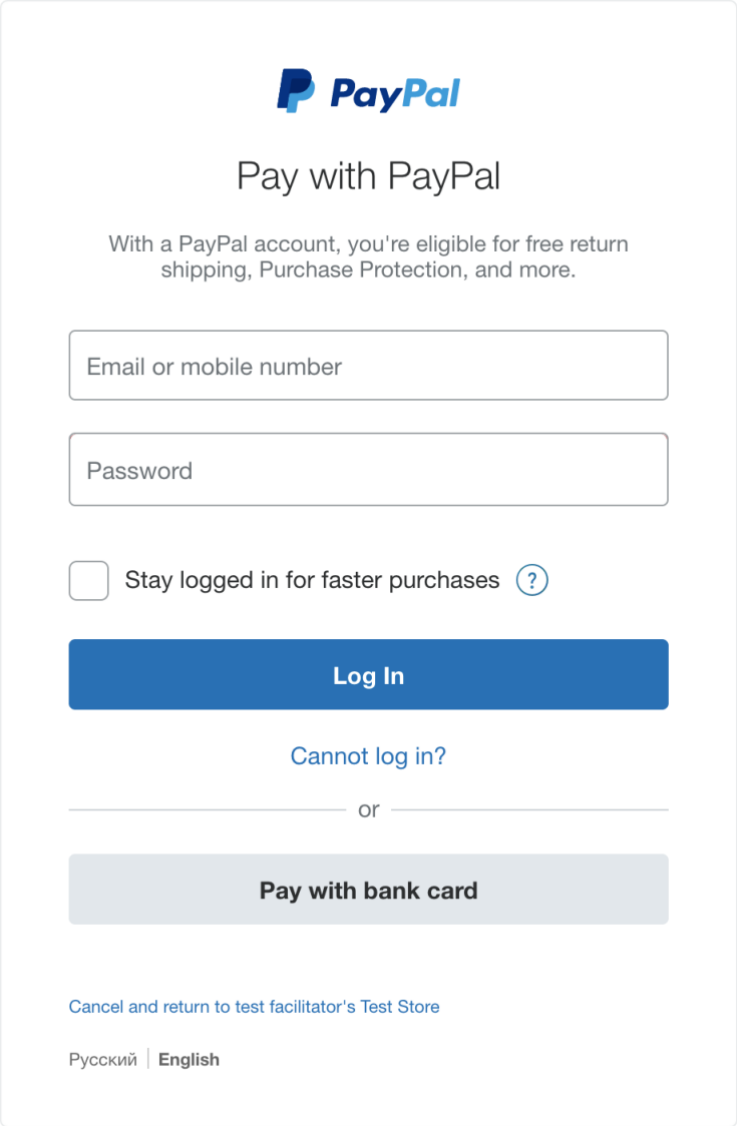


Рисунок 4.12 – Інтерфейс авторизації PayPal

Після успішної авторизації потрібно вибрати спосіб здійснення покупки.

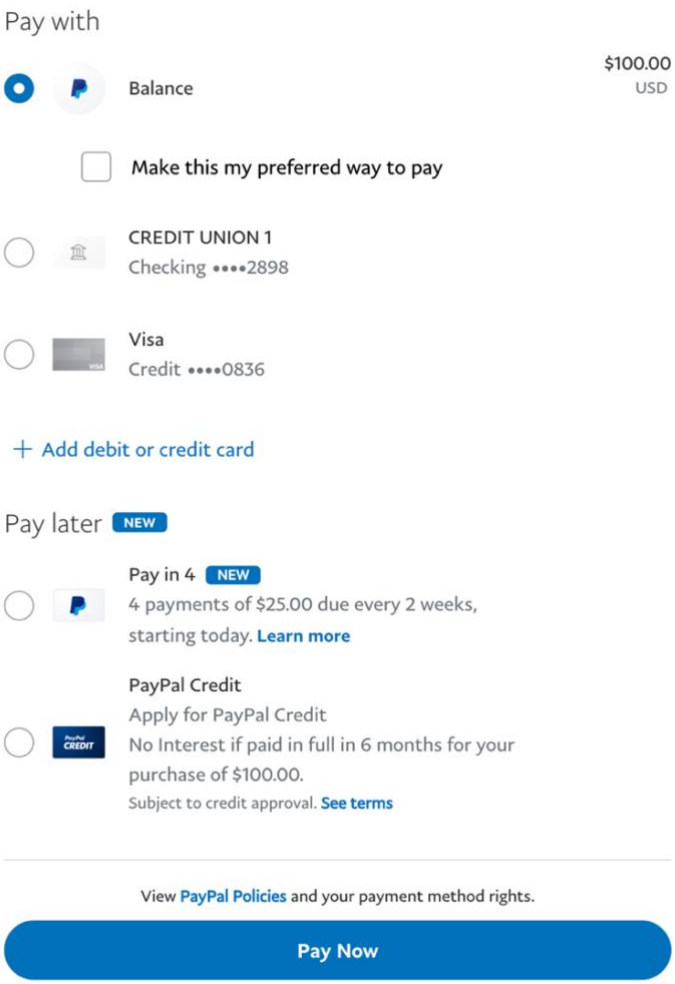


Рисунок 4.13 – Інтерфейс вибору карти для проведення платіжної операції

Після успішного проведення оплати користувача буде записано на курс на перенаправлено на сторінку особистого кабінету зі списком курсів, на які він вже записаний. Студент може відмінити запис на курс, проте повернення коштів не передбачено. Натомість користувач може записатись знову, завдяки історії платіжних операцій.

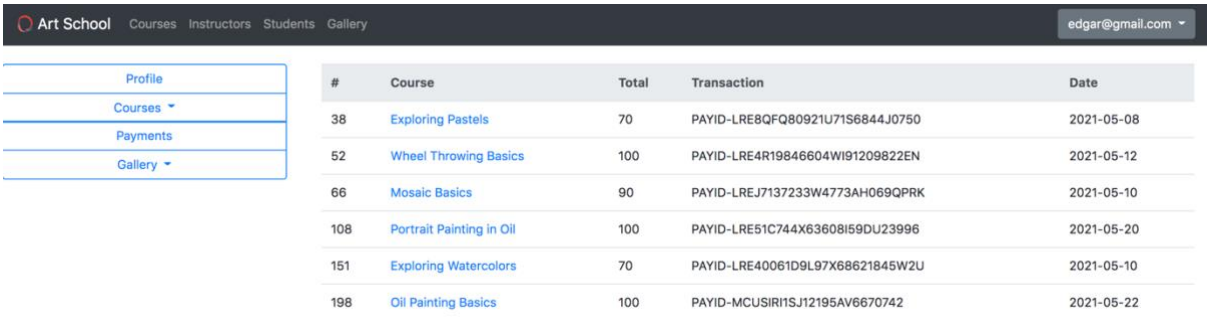


Рисунок 4.14 – Історія платіжних операцій

Розглянемо особистий кабінет викладача. Викладач, як і студент, може змінювати персональні дані. Також додатково присутня власна біографія.

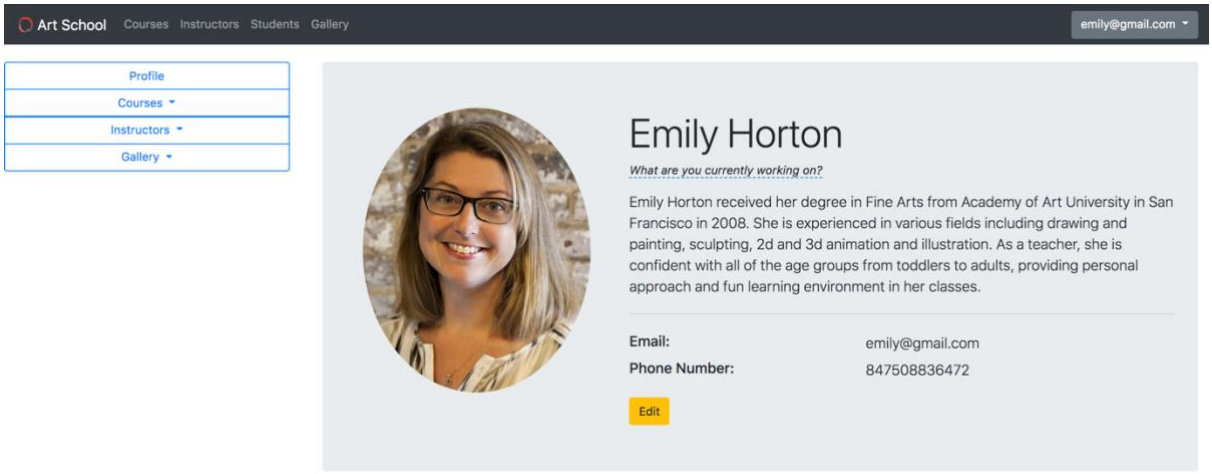


Рисунок 4.15 – Особистий кабінет викладача

Викладач має змогу створювати, редагувати, та видаляти власні курси. У разі успішної валідації введених даних, буде створений курс, що можна буде переглядати на загальній сторінці. Додатково у персональному кабінеті можна працювати з курсами, що буди створені конкретним викладачем.

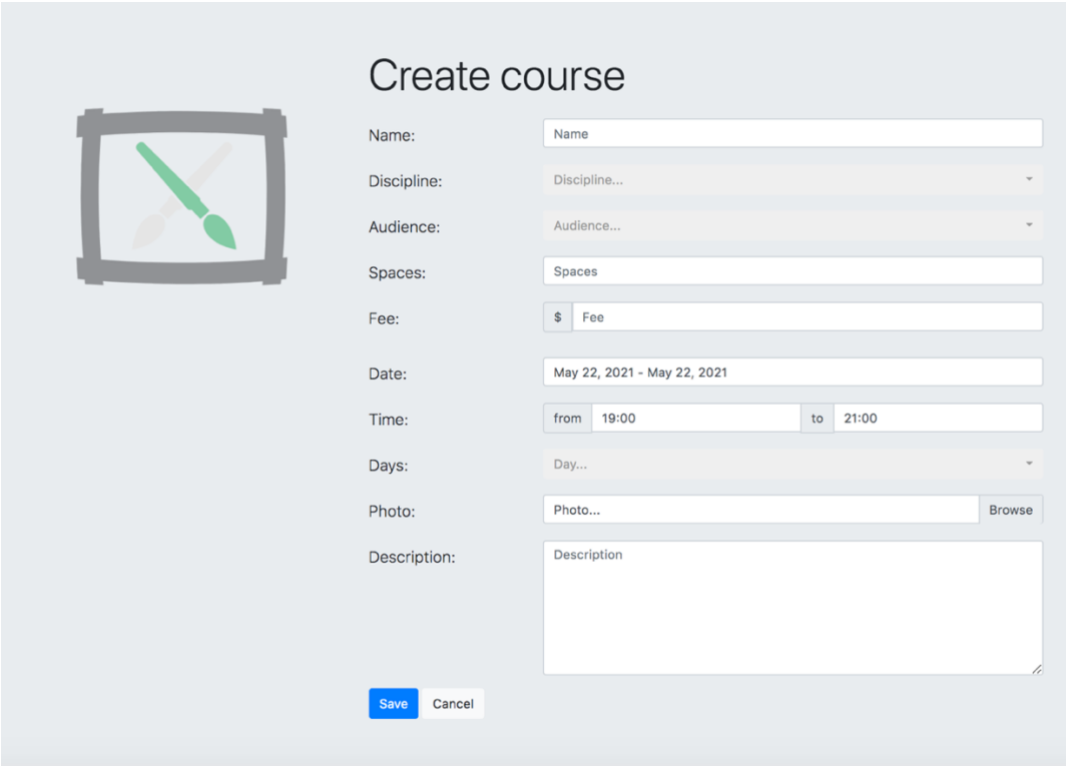


Рисунок 4.16 – Інтерфейс створення курсу

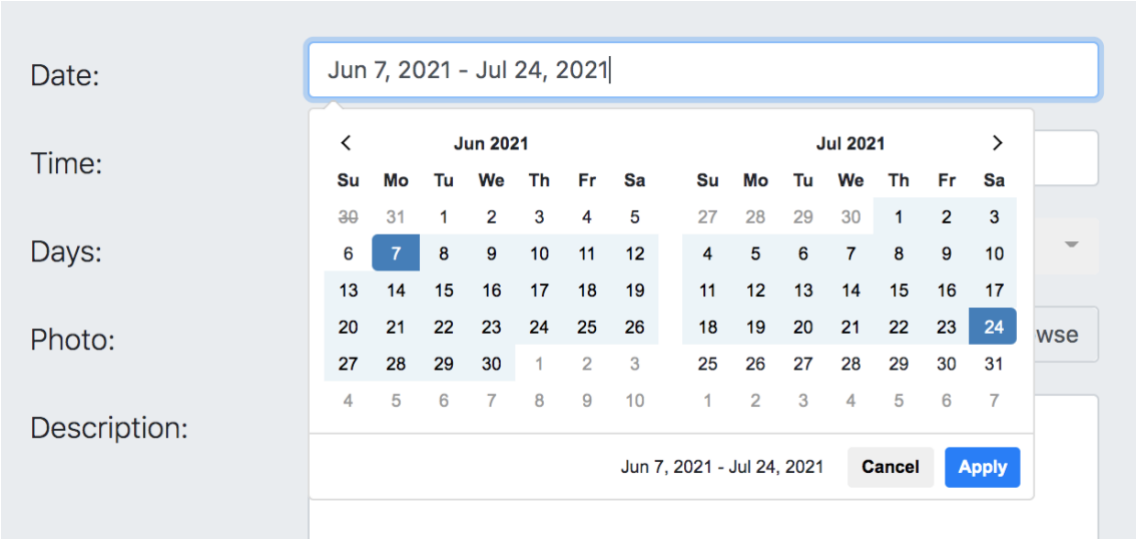


Рисунок 4.17 – Інтерфейс вибору дати проведення курсу

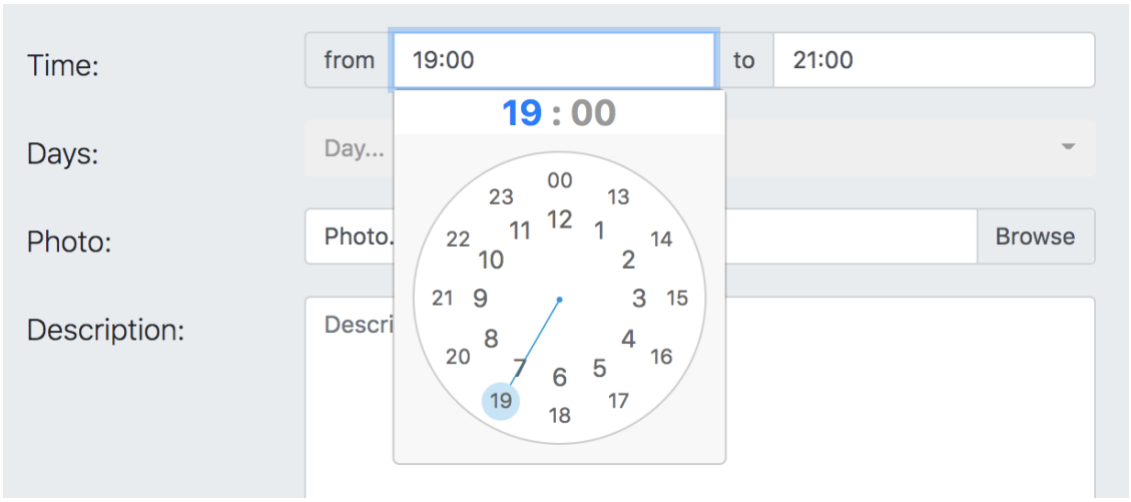


Рисунок 4.18 – Інтерфейс вибору часу проведення курсу

Викладач може запросити колегу для спільної роботи, створивши для нього власний обліковий запис. Після цього новий викладач зможе авторизуватися в системі використавши інформацію, передану колегою, та, за необхідності, змінити персональні дані.

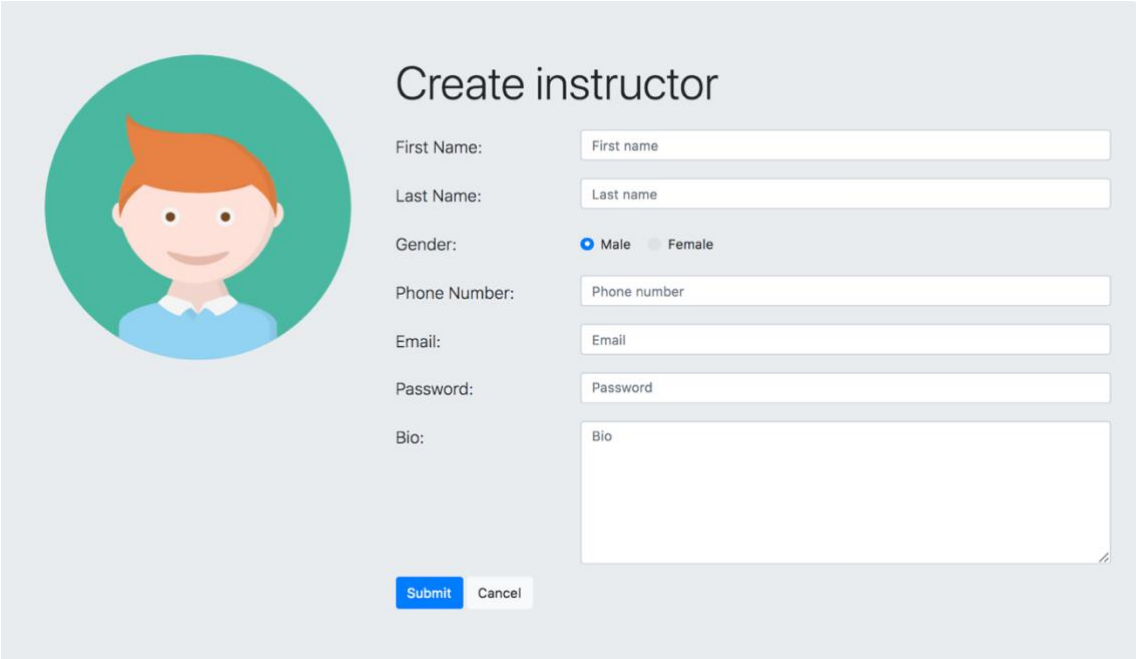


Рисунок 4.19 – Інтерфейс реєстрації викладача

Користувач може переглянути викладачів, виконати пошук по імені, та переглянути детальну інформацію про кожного.

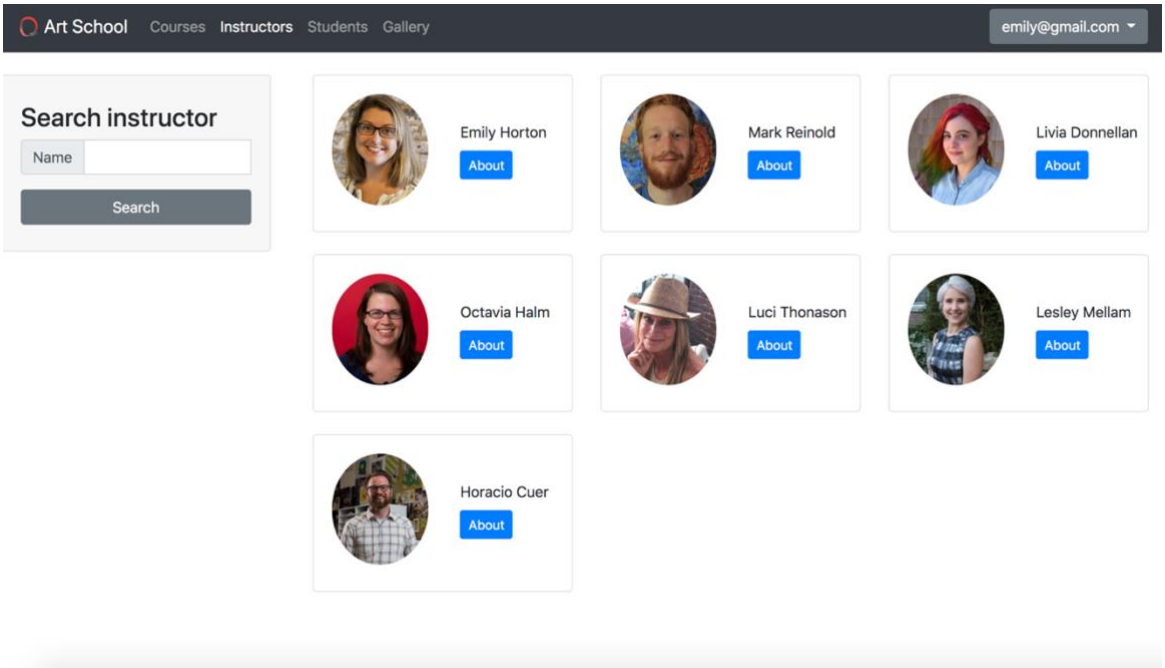


Рисунок 4.20 – Сторінка викладачів

Користувач може переглянути студентів, виконати пошук по імені та курсу, та переглянути детальну інформацію про кожного. Якщо користувачі не змінювали фото профілю, то буде використовуватись зображення за замовчуванням залежно від статі.

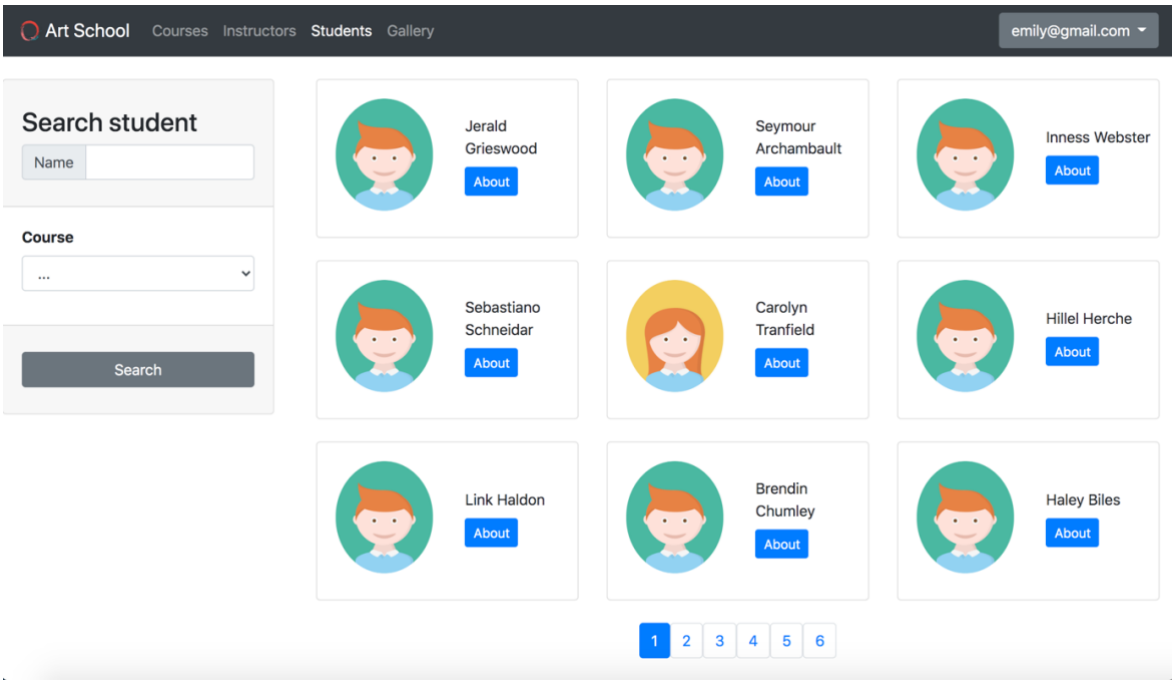


Рисунок 4.21 – Сторінка студентів

Користувач може переглянути роботи інших студентів в онлайн-галереї ти виконати пошук по назві, курсу, та автору.

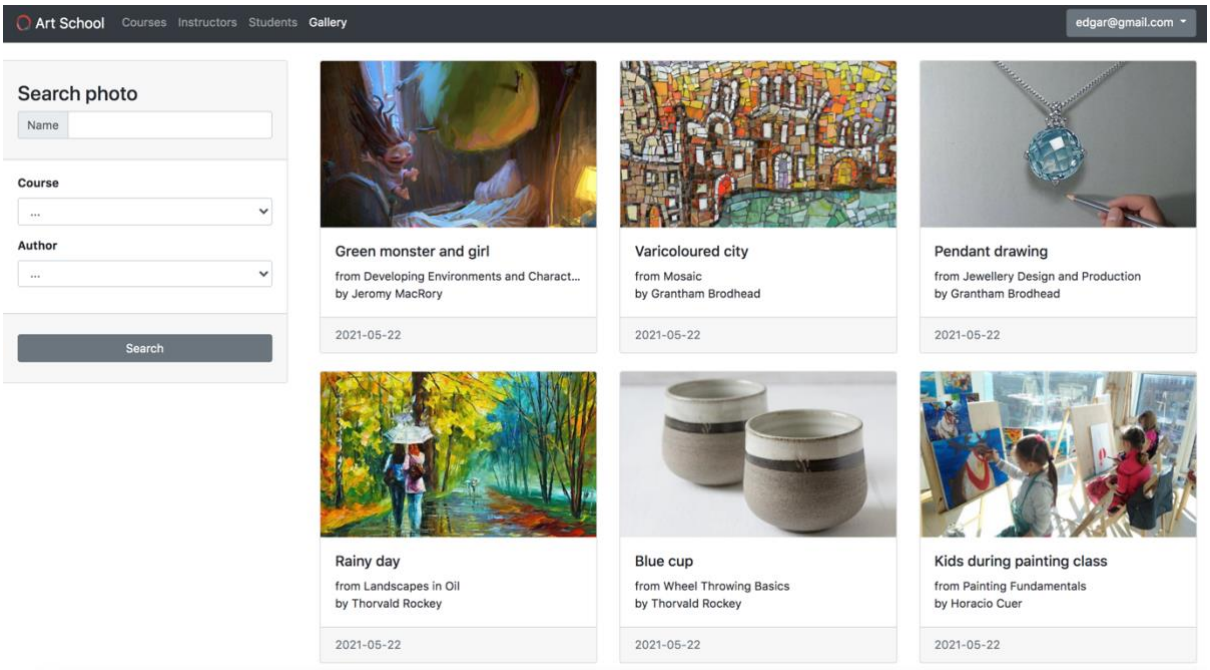


Рисунок 4.22 – Онлайн-галерея

Та за бажанням може загрузити фотографії своїх власних робіт.

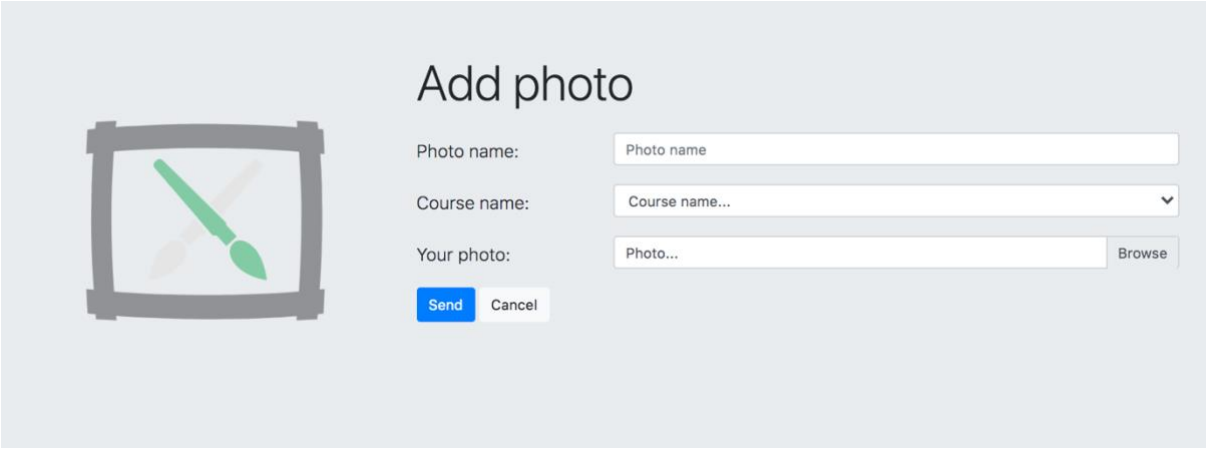


Рисунок 4.23 – Інтерфейс для загрузки фотографії в онлайн-галерею

Після вказання всіх необхідних даних, фотографію буде загружено в загальну галерею, та її зможуть переглядати всі користувачі. Також в особистому кабінеті можна переглянути всі власні роботи окремо.

ВИСНОВОК ДО РОЗДІЛУ 4

Отже, в результаті роботи була створена система для управління курсами образотворчого мистецтва, показано процес розгортання сервера та проведено демонстрацію інтерфейсу та можливостей студентів та викладачів, показано зручність веб-додатку у використанні, а також, що всі вимоги до системи були реалізовані.

ВИСНОВКИ

У ході дипломної роботи був спроектований та розроблений веб-додаток для організації курсів образотворчого мистецтва, що дає змогу студентам шукати курси використовуючи різноманітні фільтри, переглядати детальну інформацію про кожен, та записуватись і проводити оплату онлайн на курс, що зацікавив. А викладачу дає можливість створювати та редагувати власні курси, запрошувати нових викладачів. Також доступна онлайн-галерея де можна розглядати роботи талановитих студентів. Реалізовані вимоги до ролей неавторизованого користувача, студента та викладача. Дотримано всі правила доступу до функцій веб-додатку для різних типів користувачів. Перевірено всі можливі варіанти сценарію роботи з додатком.

Розробка проводилась в середовищі розробки Intellij IDEA з використанням мови Java та фреймворку для веб-розробки Spring. Готовим продуктом є WAR-файл, що може бути запущеним на будь-якому сервері.

Система має простий та зрозумілий інтерфейс, дані зберігаються в реляційній базі даних MySQL. Виконане дослідження існуючих систем показує, що робота може бути застосована на практиці та є актуальним рішенням.

СПИСОК ВИКОРИСТАННОЇ ЛІТЕРАТУРИ

1. Образотворче мистецтво [Електронний ресурс]. Режим доступу: https://uk.wikipedia.org/wiki/Образотворче_мистецтво
2. Java [Електронний ресурс]. Режим доступу: https://uk.wikipedia.org/wiki/Java
3. Introduction to Spring Framework [Електронний ресурс]. Режим доступу: https://docs.spring.io/spring-framework/docs/4.0.x/spring-framework-reference/html/overview.html
4. Spring IoC container internal working explained with diagram [Електронний ресурс]. Режим доступу: https://tutorialsinhand.com/Articles/explain-spring-ioc-container-internal-working.aspx
5. Learn Spring Boot Tutorial [Електронний ресурс]. Режим доступу: https://www.javatpoint.com/spring-boot-tutorial
6. Spring MVC — основні принципи [Електронний ресурс]. Режим доступу: https://habr.com/ru/post/336816/
7. What is Spring Security [Електронний ресурс]. Режим доступу: https://www.javadevjournal.com/spring/what-is-spring-security/
8. Spring Data JPA [Електронний ресурс]. Режим доступу: https://innovationm.co/spring-data-jpa/
9. Hibernate [Електронний ресурс]. Режим доступу: https://uk.wikipedia.org/wiki/Hibernate
10. MySQL [Електронний ресурс]. Режим доступу: https://uk.wikipedia.org/wiki/MySQL
11. Apache Maven [Електронний ресурс]. Режим доступу: https://uk.wikipedia.org/wiki/Apache_Maven
12. Thymeleaf [Електронний ресурс]. Режим доступу: https://en.wikipedia.org/wiki/Thymeleaf
13. Bootstrap https://uk.wikipedia.org/wiki/Bootstrap

14. Model–view–controller [Електронний ресурс]. Режим доступу: https://ru.wikipedia.org/wiki/Model-View-Controller

ГРАФІЧНІ МАТЕРІАЛИ

до дипломного проєкту

на тему: «Веб-додаток для управління курсами образотворчого мистецтва»

Київ – 2021 р.

ДОДАТОК 1

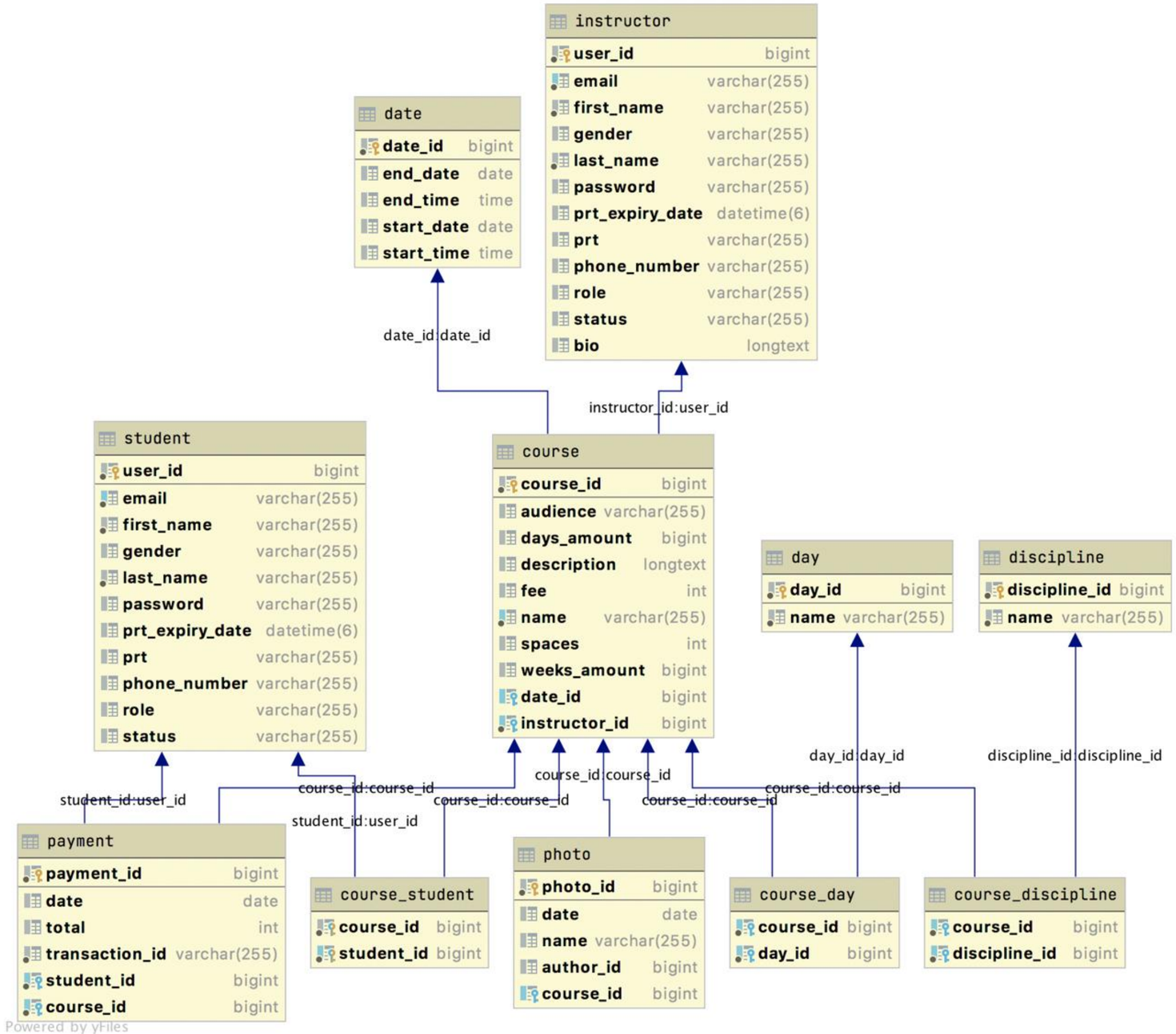
Веб-додаток для управління курсами образотворчого мистецтва

Схема бази даних

ІАЛЦ.467800.004 Д1

Аркушів 1

Київ – 2021 р.



					ІАЛЦ.467800.004 Д1	
Зм.	Арк.	№ докум.	Підпис	Дата		
Розроб.		Гладка Т. А.			Схема бази даних	Літ. / Маса / Масштаб
Перевір.		Алещенко О. В.				Арк. / Аркушів
Н. контр.		Сімоненко В. П.			Дипломна робота	НТУУ “КПІ ім. Ігоря Сікорського” ФІОТ група ІВ-72
Затверд.						

ДОДАТОК 2

Веб-додаток для управління курсами образотворчого мистецтва

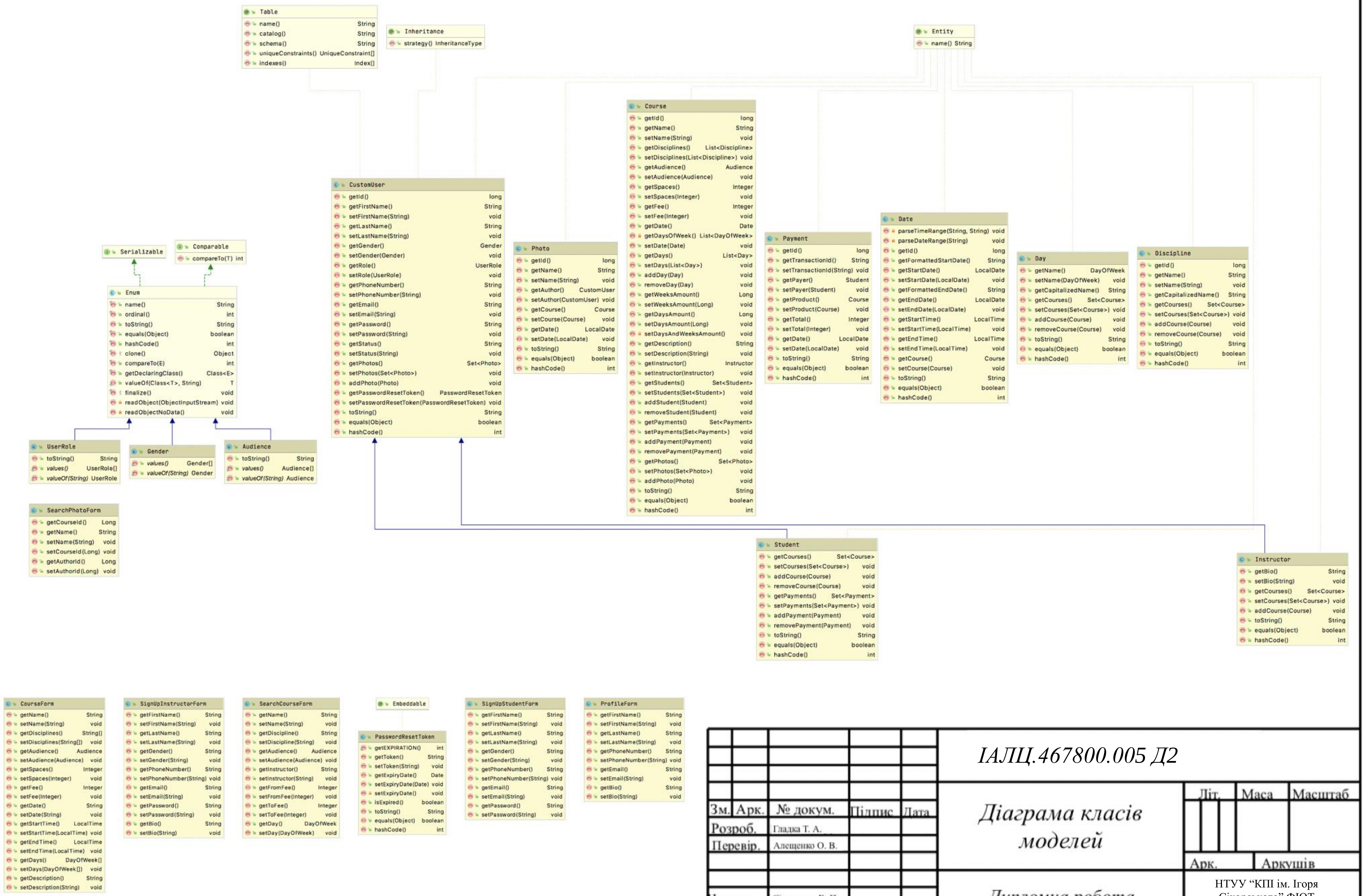
Діаграма класів моделей

ІАЛЦ.467800.005 Д2

Аркушів 1

Київ – 2021 р.

ІАЛЦ.467800.005 Д2



Зм.	Арк.	№ докум.	Підпис	Дата
Розроб.		Гладка Т. А.		
Перевір.		Алещенко О. В.		
Н. контр.		Сімоненко В. П.		
Затверд.				

ІАЛЦ.467800.005 Д2

Діаграма класів моделей

Літ.	Маса	Масштаб

Арк.	Аркушів

Дипломна робота

НТУУ "КПІ ім. Ігоря Сікорського" ФІОТ група ІВ-72

ДОДАТОК 3

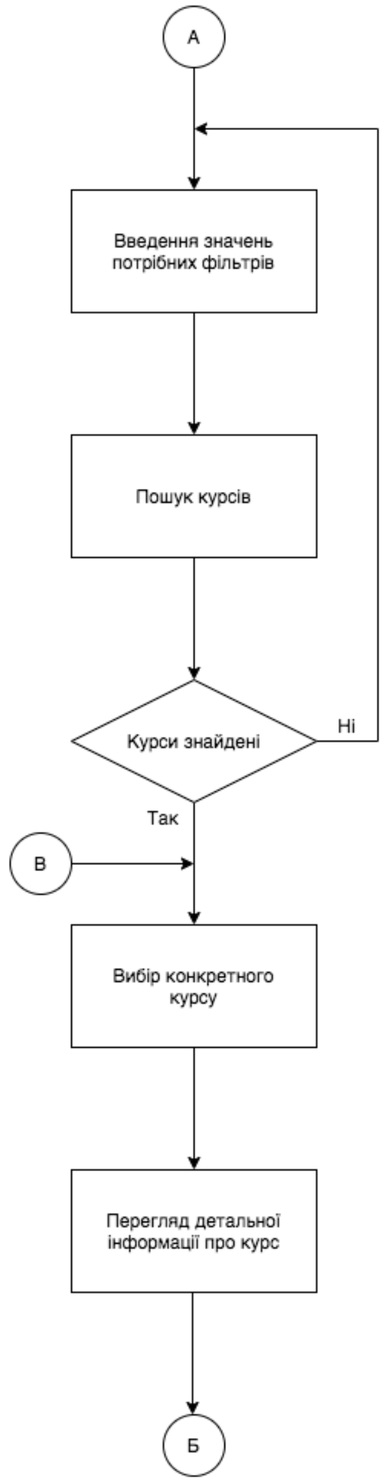
Веб-додаток для управління курсами образотворчого мистецтва

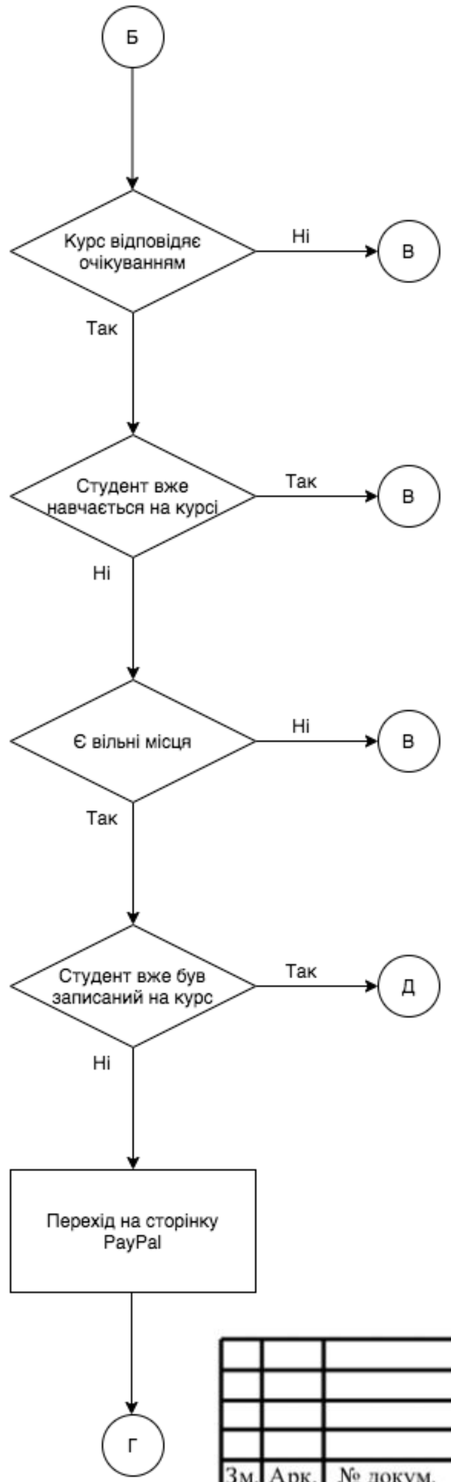
Алгоритм запису студента на курс

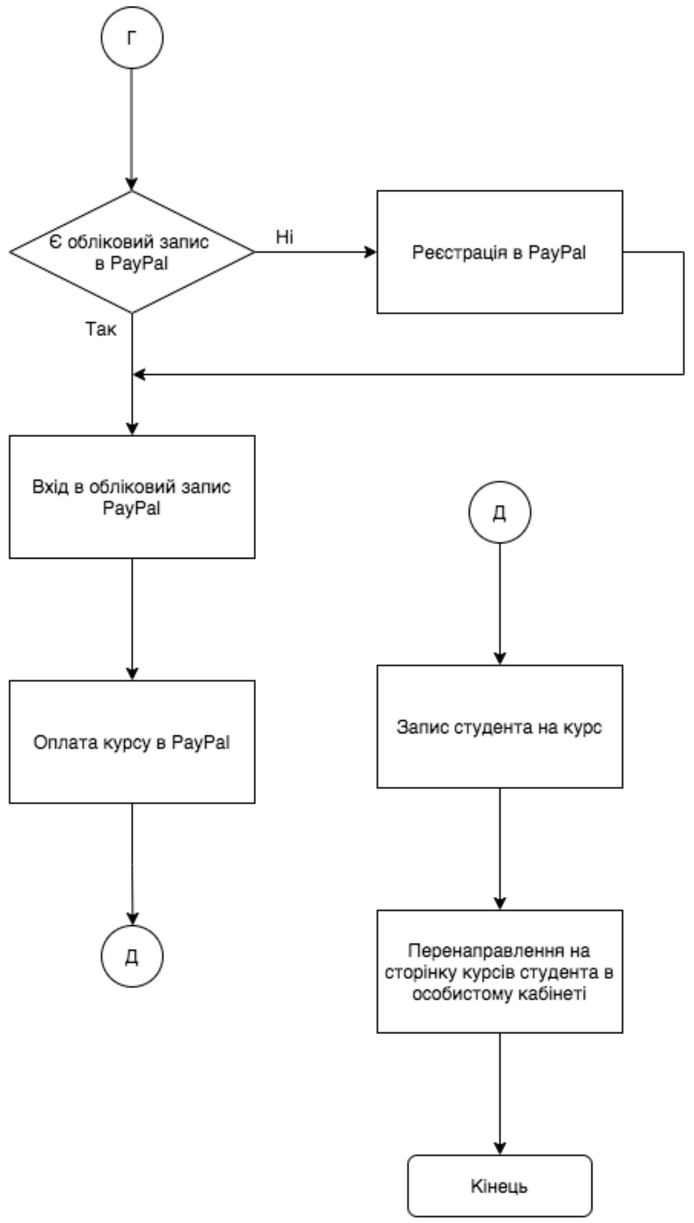
ІАЛЦ.467800.006 Д3

Аркушів 1

Київ – 2021 р.







					ІАЛЦ.467800.006 ДЗ	
Зм.	Арк.	№ докум.	Підпис	Дата	Алгоритм запису студента на курс	Літ. / Маса / Масштаб
Розроб.		Гладка Т. А.				
Перевір.		Алещенко О. В.				Арк. / Аркушів
Н. контр.		Сімоненко В. П.			Дипломна робота	НТУУ “КПІ ім. Ігоря Сікорського” ФІОТ група ІВ-72
Затверд.						

ДОДАТОК 4

Веб-додаток для управління курсами образотворчого мистецтва

Лістинг частини коду програми

ІАЛЦ.467800.007 Д4

Аркушів 13

Київ – 2021 р.

```java
package com.artschool.controller.course;

import com.artschool.model.entity.*;
import com.artschool.model.form.SearchCourseForm;
import com.artschool.service.course.CourseService;
import com.artschool.service.course.DayService;
import com.artschool.service.course.DisciplineService;
import com.artschool.service.course.CourseSearchService;
import com.artschool.service.util.LoadPhotoService;
import com.artschool.service.user.UserService;
import com.artschool.service.util.PageableService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Page;
import org.springframework.http.ResponseEntity;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.multipart.MultipartFile;
import org.springframework.web.servlet.ModelAndView;

import java.io.IOException;
import java.security.Principal;

@Controller
@RequestMapping("/course")
public class CourseController {

    private final CourseService courseService;

    private final DisciplineService disciplineService;

    private final UserService userService;

    private final CourseSearchService courseSearchService;

    private final DayService dayService;

    private final LoadPhotoService loadPhotoService;

    private final PageableService<Course> pageableService;

    @Autowired
    public CourseController(CourseService courseService, DisciplineService
disciplineService, UserService userService,
                            CourseSearchService courseSearchService, DayService
dayService,
                            LoadPhotoService loadPhotoService,
PageableService<Course> pageableService) {
        this.courseService = courseService;
        this.disciplineService = disciplineService;
        this.userService = userService;
        this.courseSearchService = courseSearchService;
        this.dayService = dayService;
        this.loadPhotoService = loadPhotoService;
        this.pageableService = pageableService;
    }

    @GetMapping("/search")
    public ModelAndView search(@ModelAttribute SearchCourseForm form,
                               @RequestParam(required = false) Integer page,
                               @RequestParam(required = false) Integer size) {
        ModelAndView modelAndView = new ModelAndView("/course/all_courses");

        Page<Course> courses =
pageableService.paginate(courseSearchService.findCourses(form), page, size);
        modelAndView.addObject("courses", courses);
```

```
        modelAndView.addObject("pages", courses.getTotalPages());
        initSearchParameters(modelAndView);
        return modelAndView;
    }

    @GetMapping("/{id}")
    public ModelAndView get(@PathVariable long id, Principal principal) {
        Course course = courseService.findCourseByIdAndInit(id);
        ModelAndView modelAndView = new ModelAndView("/course/course", "course",
course);
        initSearchParameters(modelAndView);
        if (principal == null) return modelAndView;

        CustomUser customUser = userService.findByEmail(principal.getName());
        if (customUser instanceof Student) {
            modelAndView.addObject("enrolled", courseService.isEnrolled((Student)
customUser, course));
        } else if (customUser instanceof Instructor){
            modelAndView.addObject("author", courseService.isAuthor((Instructor)
customUser, course));
        }
        return modelAndView;
    }

    @PostMapping("/upload_photo/{id}")
    public String uploadCoursePhoto(@PathVariable long id, @RequestParam
MultipartFile photo) throws IOException {
        loadPhotoService.writeCoursePhoto(id, photo);
        return "redirect:/course/" + id;
    }

    @GetMapping("/get_photo/{id}")
    public ResponseEntity<byte[]> getCoursePhoto(@PathVariable long id) throws
IOException {
        return loadPhotoService.readCoursePhoto(id);
    }

    private void initSearchParameters(ModelAndView modelAndView) {
        modelAndView.addObject("disciplines",
disciplineService.findDisciplines());
        modelAndView.addObject("instructors", userService.findInstructors());
        modelAndView.addObject("days", dayService.findDays());
        modelAndView.addObject("fromFee", courseService.findMinCourseFee());
        modelAndView.addObject("toFee", courseService.findMaxCourseFee());
    }
}
package com.artschool.controller.course;

import com.artschool.model.entity.Course;
import com.artschool.model.form.CourseForm;
import com.artschool.service.course.CourseService;
import com.artschool.service.course.DayService;
import com.artschool.service.course.DisciplineService;
import com.artschool.service.util.LoadPhotoService;
import com.artschool.service.user.UserService;
import com.artschool.service.util.PageableService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Page;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.multipart.MultipartFile;
import org.springframework.web.servlet.ModelAndView;
import org.springframework.web.servlet.mvc.support.RedirectAttributes;

import java.io.IOException;
```

```
import java.security.Principal;

@Controller
@RequestMapping("/instructor/course")
public class InstructorCourseController {

    private final CourseService courseService;

    private final UserService userService;

    private final DayService dayService;

    private final DisciplineService disciplineService;

    private final LoadPhotoService loadPhotoService;

    private final PageableService<Course> pageableService;

    @Autowired
    public InstructorCourseController(CourseService courseService, UserService
userService, DayService dayService,
                                      DisciplineService disciplineService,
LoadPhotoService loadPhotoService,
                                      PageableService<Course> pageableService) {
        this.courseService = courseService;
        this.userService = userService;
        this.dayService = dayService;
        this.disciplineService = disciplineService;
        this.loadPhotoService = loadPhotoService;
        this.pageableService = pageableService;
    }

    @GetMapping("/list")
    public ModelAndView userCourses(Principal principal,
                                    @RequestParam(required = false) Integer page,
                                    @RequestParam(required = false) Integer size)
{
        ModelAndView modelAndView = new ModelAndView("/course/user_courses");
        Page<Course> courses =
pageableService.paginate(userService.getInstructorCourses(principal.getName()),
                page, size);
        modelAndView.addObject("courses", courses);
        modelAndView.addObject("pages", courses.getTotalPages());
        return modelAndView;
    }

    @GetMapping("/edit/{id}")
    public ModelAndView edit(@PathVariable long id) {
        ModelAndView modelAndView = new ModelAndView("/course/edit_course");
        modelAndView.addObject("course", courseService.findCourseByIdAndInit(id));
        modelAndView.addObject("days", dayService.findDays());
        modelAndView.addObject("disciplines",
disciplineService.findDisciplines());
        return modelAndView;
    }

    @PostMapping("/update/{id}")
    public String update(@PathVariable long id,
                         @ModelAttribute CourseForm form,
                         @RequestParam(required = false) MultipartFile photo,
                         RedirectAttributes redirectAttributes) throws IOException
{
        Course course = courseService.updateCourse(id, form);
        if (course == null) {
            redirectAttributes.addAttribute("error", "Course with this name
```

```
already exists!");
            return "redirect:/instructor/course/edit/" + id;
        }
        if (photo != null && !photo.isEmpty())
loadPhotoService.writeCoursePhoto(id, photo);
        return "redirect:/instructor/course/list";
    }

    @GetMapping("/create")
    public ModelAndView create() {
        ModelAndView modelAndView = new ModelAndView("/course/create_course");
        modelAndView.addObject("days", dayService.findDays());
        modelAndView.addObject("disciplines",
disciplineService.findDisciplines());
        return modelAndView;
    }

    @PostMapping("/save")
    public String save(@ModelAttribute CourseForm form,
                       @RequestParam(required = false) MultipartFile photo,
                       Principal principal,
                       RedirectAttributes redirectAttributes) throws IOException {
        Course course = courseService.createCourse(form, principal.getName());
        if (course == null) {
            redirectAttributes.addAttribute("error", "Course with this name
already exists!");
            return "redirect:/instructor/course/create";
        }
        if (photo != null && !photo.isEmpty())
loadPhotoService.writeCoursePhoto(course.getId(), photo);
        return "redirect:/instructor/course/list";
    }

    @GetMapping("/delete/{id}")
    public String delete(@PathVariable long id) throws IOException {
        courseService.deleteCourse(id);
        loadPhotoService.deleteCoursePhoto(id);
        return "redirect:/instructor/course/list";
    }
}
package com.artschool.controller.course;

import com.artschool.model.entity.*;
import com.artschool.service.course.CourseService;
import com.artschool.service.course.PaymentService;
import com.artschool.service.user.UserService;
import com.artschool.service.util.PageableService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Page;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.servlet.ModelAndView;
import org.springframework.web.servlet.mvc.support.RedirectAttributes;

import java.security.Principal;
import java.time.LocalDate;
import java.util.List;

@Controller
@RequestMapping("/student/course")
public class StudentCourseController {

    private final CourseService courseService;

    private final UserService userService;
```

```
    private final PaymentService paymentService;

    private final PageableService<Course> coursePageableService;

    private final PageableService<Payment> paymentPageableService;

    @Autowired
    public StudentCourseController(CourseService courseService, UserService
userService,
                                   PaymentService paymentService,
PageableService<Course> coursePageableService,
                                   PageableService<Payment>
paymentPageableService) {
        this.courseService = courseService;
        this.userService = userService;
        this.paymentService = paymentService;
        this.coursePageableService = coursePageableService;
        this.paymentPageableService = paymentPageableService;
    }

    @GetMapping("/list")
    public ModelAndView userCourses(Principal principal,
                                    @RequestParam(required = false) Integer page,
                                    @RequestParam(required = false) Integer size)
{
        ModelAndView modelAndView = new ModelAndView("/course/user_courses");
        Page<Course> courses =
coursePageableService.paginate(userService.getStudentCourses(principal.getName()),
page, size);
        modelAndView.addObject("courses", courses);
        modelAndView.addObject("pages", courses.getTotalPages());
        return modelAndView;
    }

    @PostMapping("/enroll/{id}")
    @ResponseBody
    public void enroll(@PathVariable long id, @RequestParam String transactionId,
Principal principal) {
        Course course = courseService.findCourseById(id);
        Student student = userService.findStudentByEmail(principal.getName());
        paymentService.createPayment(new Payment(transactionId, student, course,
course.getFee(), LocalDate.now()));
        courseService.enrollInCourse(principal.getName(), id);
    }

    @GetMapping("/unenroll/{id}")
    public String unenroll(@PathVariable long id, Principal principal) {
        courseService.unenrollFromCourse(principal.getName(), id);
        return "redirect:/student/course/list";
    }

    @GetMapping("/restore/{id}")
    public String restore(@PathVariable long id,
                          Principal principal,
                          RedirectAttributes redirectAttributes) {
        List<Payment> payments = paymentService.findPayments(principal.getName(),
id);
        if (payments != null && !payments.isEmpty()) {
            courseService.enrollInCourse(principal.getName(), id);
            return "redirect:/student/course/list";
        }
        redirectAttributes.addAttribute("error", "You haven't made a payment for
this course! " +
                "If you want to enroll, please pay for it by PayPal.");
```

```
        return "redirect:/course/" + id;
    }

    @GetMapping("/payments")
    public ModelAndView payments(Principal principal,
                                 @RequestParam(required = false) Integer page,
                                 @RequestParam(required = false) Integer size) {
        ModelAndView modelAndView = new ModelAndView("/course/user_payments");
        Page<Payment> payments =
paymentPageableService.paginate(paymentService.findPayments(principal.getName()),
                page, size);
        modelAndView.addObject("payments", payments);
        modelAndView.addObject("pages", payments.getTotalPages());
        return modelAndView;
    }
}
package com.artschool.controller.gallery;

import com.artschool.model.entity.Photo;
import com.artschool.model.form.SearchPhotoForm;
import com.artschool.service.course.CourseService;
import com.artschool.service.util.LoadPhotoService;
import com.artschool.service.gallery.PhotoService;
import com.artschool.service.user.UserService;
import com.artschool.service.util.PageableService;
import org.springframework.data.domain.Page;
import org.springframework.http.ResponseEntity;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.multipart.MultipartFile;

import java.io.IOException;
import java.security.Principal;

@Controller
@RequestMapping("/gallery")
public class GalleryController {

    private final PhotoService photoService;

    private final UserService userService;

    private final CourseService courseService;

    private final LoadPhotoService loadPhotoService;

    private final PageableService<Photo> pageableService;

    public GalleryController(PhotoService photoService, UserService userService,
CourseService courseService,
                             LoadPhotoService loadPhotoService,
PageableService<Photo> pageableService) {
        this.photoService = photoService;
        this.userService = userService;
        this.courseService = courseService;
        this.loadPhotoService = loadPhotoService;
        this.pageableService = pageableService;
    }

    @GetMapping
    public String searchPhoto(@ModelAttribute SearchPhotoForm form,
                              @RequestParam(required = false) Integer page,
                              @RequestParam(required = false) Integer size,
                              Model model) {
```

```
        Page<Photo> photos =
pageableService.paginate(photoService.findPhotos(form), page, size);
        model.addAttribute("photos", photos);
        model.addAttribute("pages", photos.getTotalPages());

        model.addAttribute("authors", userService.findUsers());
        model.addAttribute("courses", courseService.findCourses());
        return "/gallery/gallery";
    }

    @GetMapping("/user")
    public String userPhoto(Principal principal,
                            @RequestParam(required = false) Integer page,
                            @RequestParam(required = false) Integer size,
                            Model model) {
        Page<Photo> photos =
pageableService.paginate(photoService.findByAuthorEmail(principal.getName()),
page, size);
        model.addAttribute("photos", photos);
        model.addAttribute("pages", photos.getTotalPages());

        model.addAttribute("authors", userService.findUsers());
        model.addAttribute("courses", courseService.findCourses());
        return "/gallery/user_gallery";
    }

    @GetMapping("/add_photo")
    public String showAddPhotoPage(Model model) {
        model.addAttribute("courses", courseService.findCourses());
        return "/gallery/add_photo";
    }

    @PostMapping("/upload_photo")
    public String addPhoto(@RequestParam String name,
                           @RequestParam String course,
                           @RequestParam MultipartFile photo,
                           Principal principal) throws IOException {
        Photo p = photoService.createPhoto(name, principal.getName(), course);
        loadPhotoService.writeGalleryPhoto(p.getId(), photo);
        return "redirect:/gallery/user";
    }

    @GetMapping("/get_photo/{id}")
    public ResponseEntity<byte[]> getPhoto(@PathVariable long id) throws
IOException {
        return loadPhotoService.readGalleryPhoto(id);
    }
}
package com.artschool.controller.user;

import com.artschool.model.entity.Instructor;
import com.artschool.model.form.ProfileForm;
import com.artschool.service.course.CourseService;
import com.artschool.service.util.LoadPhotoService;
import com.artschool.service.user.UserService;
import com.artschool.service.util.PageableService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Page;
import org.springframework.http.ResponseEntity;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.multipart.MultipartFile;
import org.springframework.web.servlet.ModelAndView;
import org.springframework.web.servlet.mvc.support.RedirectAttributes;
```

```
import java.io.IOException;
import java.security.Principal;

@Controller
public class InstructorController {

    private final UserService userService;

    private final CourseService courseService;

    private final LoadPhotoService loadPhotoService;

    private final PageableService<Instructor> instructorPageableService;

    @Autowired
    public InstructorController(UserService userService, CourseService
courseService, LoadPhotoService loadPhotoService,
                                PageableService<Instructor>
instructorPageableService) {
        this.userService = userService;
        this.courseService = courseService;
        this.loadPhotoService = loadPhotoService;
        this.instructorPageableService = instructorPageableService;
    }

    @GetMapping("/search/instructor")
    public ModelAndView searchInstructor(@RequestParam(required = false) String
name,
                                         @RequestParam(required = false) Integer
page,
                                         @RequestParam(required = false) Integer
size) {
        ModelAndView modelAndView = new ModelAndView("/user/users");
        Page<Instructor> instructors;
        if (name == null || name.equals(""))
            instructors =
instructorPageableService.paginate(userService.findInstructors(), page, size);
        else
            instructors =
instructorPageableService.paginate(userService.findInstructorsByName(name), page,
size);

        modelAndView.addObject("instructors", instructors);
        modelAndView.addObject("pages", instructors.getTotalPages());
        return modelAndView;
    }

    @GetMapping("/search/instructor/{id}")
    public ModelAndView instructorById(@PathVariable long id, Principal principal)
{
        ModelAndView modelAndView = new ModelAndView("/user/profile_by_id");
        if (principal != null) {
            Instructor owner =
userService.findInstructorByEmail(principal.getName());
            if (owner != null && owner.getId() == id)
modelAndView.addObject("owner", true);
        }
        modelAndView.addObject("member", userService.findInstructorById(id));
        modelAndView.addObject("courses", courseService.findCourses());
        return modelAndView;
    }

    @GetMapping("/instructor/profile/edit")
    public ModelAndView editProfile(Principal principal) {
        return new ModelAndView("/user/edit_profile", "user",
```

```
                userService.findInstructorByEmail(principal.getName()));
    }

    @PostMapping("/instructor/profile/save")
    public String saveProfile(@ModelAttribute ProfileForm form,
                              @RequestParam(required = false) MultipartFile photo,
                              Principal principal,
                              RedirectAttributes redirectAttributes) throws
IOException {
        Instructor instructor = userService.editInstructor(principal.getName(),
form);
        if (instructor == null) {
            redirectAttributes.addAttribute("error", "User with this email already
exists!");
            return "redirect:/profile/edit";
        }
        if (photo != null && !photo.isEmpty())
loadPhotoService.writeInstructorPhoto(instructor.getId(), photo);

        if (instructor.getEmail().equals(principal.getName())) return
"redirect:/profile";
        else return "redirect:/logout";
    }

    @PostMapping("/instructor/edit_status")
    public @ResponseBody String editStatus(@RequestParam("value") String status,
Principal principal) {
        userService.editInstructorStatus(principal.getName(), status);
        return status;
    }

    @PostMapping("/upload_photo/instructor/{id}")
    public String uploadInstructorPhoto(@PathVariable long id, @RequestParam
MultipartFile photo) throws IOException {
        loadPhotoService.writeInstructorPhoto(id, photo);
        return "redirect:/instructor/profile";
    }

    @GetMapping("/get_photo/instructor/{id}")
    public ResponseEntity<byte[]> getInstructorPhoto(@PathVariable long id) throws
IOException {
        return loadPhotoService.readInstructorPhoto(id);
    }

}
package com.artschool.controller.user;

import com.artschool.model.entity.Student;
import com.artschool.model.form.ProfileForm;
import com.artschool.service.course.CourseService;
import com.artschool.service.util.LoadPhotoService;
import com.artschool.service.user.StudentSearchService;
import com.artschool.service.user.UserService;
import com.artschool.service.util.PageableService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Page;
import org.springframework.http.ResponseEntity;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.multipart.MultipartFile;
import org.springframework.web.servlet.ModelAndView;
import org.springframework.web.servlet.mvc.support.RedirectAttributes;

import java.io.IOException;
import java.security.Principal;
```

```java
@Controller
public class StudentController {

    private final UserService userService;

    private final CourseService courseService;

    private final StudentSearchService studentSearchService;

    private final LoadPhotoService loadPhotoService;

    private final PageableService<Student> studentPageableService;

    @Autowired
    public StudentController(UserService userService, CourseService courseService,
                             StudentSearchService studentSearchService,
LoadPhotoService loadPhotoService,
                             PageableService<Student> studentPageableService) {
        this.userService = userService;
        this.courseService = courseService;
        this.studentSearchService = studentSearchService;
        this.loadPhotoService = loadPhotoService;
        this.studentPageableService = studentPageableService;
    }

    @GetMapping("/search/student")
    public ModelAndView searchStudent(@RequestParam(required = false) String name,
                                      @RequestParam(required = false) Integer
course,
                                      @RequestParam(required = false) Integer
page,
                                      @RequestParam(required = false) Integer
size){
        ModelAndView modelAndView = new ModelAndView("/user/users");
        modelAndView.addObject("courses", courseService.findCourses());

        Page<Student> students =
studentPageableService.paginate(studentSearchService.findStudents(name, course),
                page, size);

        modelAndView.addObject("students", students);
        modelAndView.addObject("pages", students.getTotalPages());
        return modelAndView;
    }

    @GetMapping("/search/student/{id}")
    public ModelAndView studentById(@PathVariable long id, Principal principal) {
        ModelAndView modelAndView = new ModelAndView("/user/profile_by_id");
        if (principal != null) {
            Student owner = userService.findStudentByEmail(principal.getName());
            if (owner != null && owner.getId() == id)
modelAndView.addObject("owner", true);
        }
        modelAndView.addObject("member", userService.findStudentById(id));
        modelAndView.addObject("courses", courseService.findCourses());
        return modelAndView;
    }

    @GetMapping("/student/profile/edit")
    public ModelAndView editProfile(Principal principal) {
        return new ModelAndView("/user/edit_profile", "user",
                userService.findStudentByEmail(principal.getName()));
    }
```

```
    @PostMapping("/student/profile/save")
    public String saveProfile(@ModelAttribute ProfileForm form,
                              @RequestParam(required = false) MultipartFile photo,
                              Principal principal,
                              RedirectAttributes redirectAttributes) throws
IOException {
        Student student = userService.editStudent(principal.getName(), form);
        if (student == null) {
            redirectAttributes.addAttribute("error", "User with this email already
exists!");
            return "redirect:/student/profile/edit";
        }
        if (photo != null && !photo.isEmpty())
loadPhotoService.writeStudentPhoto(student.getId(), photo);

        if (student.getEmail().equals(principal.getName())) return
"redirect:/profile";
        else return "redirect:/logout";
    }

    @PostMapping("/student/edit_status")
    public @ResponseBody String editStatus(@RequestParam("value") String status,
Principal principal) {
        userService.editStudentStatus(principal.getName(), status);
        return status;
    }

    @PostMapping("/upload_photo/student/{id}")
    public String uploadStudentPhoto(@PathVariable long id, @RequestParam
MultipartFile photo) throws IOException {
        loadPhotoService.writeStudentPhoto(id, photo);
        return "redirect:/student/profile";
    }

    @GetMapping("/get_photo/student/{id}")
    public ResponseEntity<byte[]> getStudentPhoto(@PathVariable long id) throws
IOException {
        return loadPhotoService.readStudentPhoto(id);
    }

}
package com.artschool.controller.user;

import com.artschool.model.entity.CustomUser;
import com.artschool.model.entity.Instructor;
import com.artschool.model.entity.PasswordResetToken;
import com.artschool.model.entity.Student;
import com.artschool.model.form.SignUpStudentForm;
import com.artschool.model.form.SignUpInstructorForm;
import com.artschool.service.security.PasswordResetService;
import com.artschool.service.security.SecurityService;
import com.artschool.service.user.UserService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.servlet.ModelAndView;
import org.springframework.web.servlet.mvc.support.RedirectAttributes;

import javax.servlet.http.HttpServletRequest;
import java.security.Principal;

@Controller
public class LoginController {
```

```
    private final UserService userService;

    private final SecurityService securityService;

    private final PasswordEncoder passwordEncoder;

    private final PasswordResetService passwordResetService;

    @Autowired
    public LoginController(UserService userService, SecurityService
securityService, PasswordEncoder passwordEncoder,
                           PasswordResetService passwordResetService) {
        this.userService = userService;
        this.securityService = securityService;
        this.passwordEncoder = passwordEncoder;
        this.passwordResetService = passwordResetService;
    }

    @GetMapping("/")
    public String index() {
        return "index";
    }

    @GetMapping("/profile")
    public ModelAndView profile(Principal principal) {
        ModelAndView modelAndView = new ModelAndView("/user/profile");
        modelAndView.addObject("user",
userService.findByEmail(principal.getName()));
        modelAndView.addObject("owner", true);
        return modelAndView;
    }

    @GetMapping("/login")
    public String login() {
        return "/user/login";
    }

    @PostMapping("/new_user")
    public String createUser(@ModelAttribute SignUpStudentForm form,
RedirectAttributes redirectAttributes) {
        Student student = userService.createStudent(form, passwordEncoder);
        if (student != null) {
            securityService.login(form.getEmail(), form.getPassword());
            return "redirect:/profile";
        }
        redirectAttributes.addAttribute("error", "User with this email already
exists!");
        return "redirect:/login";
    }

    @GetMapping("/instructor/create")
    public String createInstructor() {
        return "/user/create_instructor";
    }

    @PostMapping("/instructor/create")
    public String showCreateInstructorPage(@ModelAttribute SignUpInstructorForm
form,
                                           RedirectAttributes redirectAttributes)
{
        Instructor instructor = userService.createInstructor(form,
passwordEncoder);
        if (instructor == null) {
            redirectAttributes.addAttribute("error", "User with this email already
exists!");
```

```
            return "redirect:/instructor/create";
        }
        return "redirect:/profile";
    }

    @GetMapping("/forgot_password")
    public String showForgotPasswordPage() {
        return "/user/forgot_password";
    }

    @PostMapping("/forgot_password")
    public String sendEmail(@RequestParam String email,
                            RedirectAttributes redirectAttributes,
                            HttpServletRequest req) {
        CustomUser user = userService.findByEmail(email);
        if (user == null) {
            redirectAttributes.addAttribute("error", "There are not account
associated with this email!");
            return "redirect:/forgot_password";
        }
        String token = passwordResetService.generateToken();
        user.setPasswordResetToken(new PasswordResetToken(token));
        userService.saveOrUpdate(user);

        passwordResetService.sendEmail(user, token, req);
        redirectAttributes.addAttribute("success", "Check your email for a link to
reset your password.");
        return "redirect:/login";
    }

    @GetMapping("/reset_password")
    public String showResetPasswordPage() {
        return "/user/reset_password";
    }

    @PostMapping("/reset_password")
    public String resetPassword(@RequestParam long id,
                                @RequestParam String token,
                                @RequestParam String password,
                                RedirectAttributes redirectAttributes) {
        CustomUser user = userService.findByResetToken(token);
        PasswordResetToken resetToken = user.getPasswordResetToken();
        if (resetToken == null || resetToken.isExpired() || user.getId() != id) {
            redirectAttributes.addAttribute("error", "Wrong password reset
token!");
            return "redirect:/login";
        }
        user.setPassword(passwordEncoder.encode(password));
        userService.saveOrUpdate(user);

        redirectAttributes.addAttribute("success", "Password was successfully
restored.");
        return "redirect:/login";
    }
}
```