МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»

Кафедра інженерії програмного забезпечення

**КУРСОВА РОБОТА**

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

з дисципліни: «Бази даних»

на тему:

**«Розробка бази даних для інструменту керування завданнями»**

студента II курсу групи ВТ-21-1

спеціальності 121 «Інженерія програмного забезпечення.

Бабушка Андрія Сергійовича

(прізвище, ім’я та по-батькові)

Керівник: ст. викл. Кравченко О.Г.

Дата захисту: "8" травня 2023 р.

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

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

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

Члени комісії \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ І.І. Сугоняк

(підпис) (прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ С.М. Кравченко

(підпис) (прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ О.В. Коротун

(підпис) (прізвище та ініціали)

Житомир – 2022

ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»

Факультет інформаційно-комп’ютерних технологій

Кафедра інженерії програмного забезпечення

Освітній рівень: бакалавр

Спеціальність 121 «Інженерія програмного забезпечення»

«ЗАТВЕРДЖУЮ»

К.т.н., доцент

\_\_\_\_\_\_\_\_\_\_\_\_ А.В.Морозов

“\_\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_ р.

ЗАВДАННЯ

НА КУРСОВИЙ ПРОЕКТ СТУДЕНТУ

Бабушку Андрію Сергійовичу

1. Тема роботи: Розробка бази даних для інструменту керування завданнями,

керівник роботи: старший викладач Кравченко С.М.

1. Строк подання студентом: “ 8 ” травня 2023р.
2. Вихідні дані до роботи: Розробити базу даних для інструменту керування завданнями.
3. Зміст розрахунково-пояснювальної записки(перелік питань. Які підлягають розробці)
   * + 1. Постановка завдання
       2. Аналіз аналогічних розробок
       3. Алгоритми роботи програми

4. Опис роботи програми

5. Програмне дослідження

1. Перелік графічного матеріалу(з точним зазначенням обов’язкових креслень)

1. Посилання на репозиторій:

<https://github.com/AndriiBabushko/task-management-tool>

1. Консультанти розділів проекту (роботи)

|  |  |  |  |
| --- | --- | --- | --- |
| Розділ | Прізвище, ініціали та посади консультанта | Підпис, дата | |
| завдання  видав | завдання прийняв |
| 1,2 | Сугоняк І.І., к.т.н., доц. |  |  |
| 1,2 | Коротун О.В., к.пед.н. |  |  |
| 1,2 | Кравченко С.М., ст. викладач |  |  |
| 1,2 | Чижмотря О.В., ст. викладач |  |  |
|  |  |  |  |
|  |  |  |  |

1. Дата видачі завдання “ 8 ” березня 2023 р.

|  |  |  |  |
| --- | --- | --- | --- |
| № з/п | Назва етапів курсового проекту | Строк виконання етапів проекту | Примітки |
| 1 | Постановка задачі | 01.12.2022 | Виконано |
| 2 | Пошук, огляд та аналіз аналогічних розробок | 13.12.2022 | Виконано |
| 3 | Формулювання технічного завдання | 14.12.2022 | Виконано |
| 4 | Опрацювання літературних джерел | 16.12.2022 | Виконано |
| 5 | Проектування структури | 18.12.2022 | Виконано |
| 6 | Написання програмного коду | 19.01.2022 | Виконано |
| 7 | Відлагодження | 08.01.2022 | Виконано |
| 8 | Написання пояснювальної записки | 11.01.2022 | Виконано |
| 9 | Захист | 13.01.2022 | Виконано |

**КАЛЕНДАРНИЙ ПЛАН**

**Студент** \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Бабушко А.С.

(підпис) (прізвище та ініціали)

**Керівник проекту** \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Кравченко С.М.

(підпис) (прізвище та ініціали)

РЕФЕРАТ

Завданням на курсову роботу було розробка бази даних для інструменту керування завданнями.

Пояснювальна записка до курсової роботи на тему «Розробка бази даних для інструменту керування завданнями» складається з вступу, переліку умовних скорочень, чотирьох розділів, висновків, списку використаної літератури та додатку.

Текстова частина викладена на 00 сторінках друкованого тексту.

Пояснювальна записка має 00 сторінок додатків. Список використаних джерел містить 00 найменувань і займає 00 сторінку. В роботі наведено 00 рисунків. Загальний обсяг роботи – 00 сторінки.

У першому розділі було обґрунтовано актуальність та причини створення програмного продукту.

У другому розділі проведено проектування і розробка програмного продукту.

У третьому розділі проведено тестування програмного продукту.

Висновок містить в собі результати виконаної роботи зі створення програмного продукту.

У додатку наведений лістинг розробленого програмного продукту.

Ключові слова: ООП, ДОСТУП, ЗБЕРЕЖЕННЯ ІНФОРМАЦІЇ, БАЗА ДАНИХ, ЗАВДАННЯ, NODE JS, REACT JS, MONGO DB, ІНСТРУМЕНТ, МОДЕЛЬ, КОНТРОЛЛЕР, РОУТИ.

ЗМІСТ

[**ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ** 6](#_Toc124945510)

[**ВСТУП** 7](#_Toc124945511)

[**РОЗДІЛ 1. АНАЛІЗ ЗАДАЧІ** 8](#_Toc124945512)

[1.1 Аналіз задачі, засобів та методів її вирішення 8](#_Toc124945513)

[1.2 Аналіз існуючого програмного забезпечення за тематикою курсової роботи. 9](#_Toc124945514)

[Висновки з першого розділу 11](#_Toc124945515)

[**РОЗДІЛ 2. ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ** 12](#_Toc124945516)

[2.1 Проектування загального алгоритму роботи програми 12](#_Toc124945517)

[2.2 Розробка функціональних алгоритмів роботи програми 15](#_Toc124945518)

[2.3 Розробка програмного забезпечення 17](#_Toc124945519)

[Висновки з другого розділу 20](#_Toc124945520)

[**РОЗДІЛ 3. ОПИС РОБОТИ З ПРОГРАМНИМ ДОДАТКОМ ТА ЙОГО ТЕСТУВАННЯ** 21](#_Toc124945521)

[3.1 Опис роботи з програмним додатком 21](#_Toc124945522)

[3.2 Тестування роботи програмного забезпечення 35](#_Toc124945523)

[Висновки до третього розділу 36](#_Toc124945524)

[**ВИСНОВКИ** 37](#_Toc124945525)

[**СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ** 38](#_Toc124945526)

[**ДОДАТКИ** 39](#_Toc124945527)

**ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ**

WS – WebStorm

JS – мова програмування «JavaScript»

TS – мова програмування «Typescript»

HTML – Hyper Text Markup Language

CSS – Cascade Style Sheet

ООП – об’єктно орієнтовний підхід

ПЗ – Програмне забезпечення

БД – База даних

MongoDB – База даних «Mongo»

REST – Representational State Transfer, дослівно: передача представницького стану.

API – Application programming interface, дослівно: Інтерфейс прикладного програмування.

# **ВСТУП**

У курсовій роботі буде наведено процес створення бази даних з використанням Mongo DB для інструменту керування завданнями.

Інструмент керування завданнями - це програмний продукт або інструмент, який допомагає управляти та організовувати завдання та проекти, встановлювати терміни та пріоритети, стежити за виконанням завдань та оцінювати продуктивність роботи.

Такі інструменти часто мають функції, такі як створення списку завдань, розподіл завдань між членами команди, спільна робота над проектом, моніторинг часу, призначення термінів, повідомлення про статус завдань та багато іншого. Вони дозволяють керівникам та командам більш ефективно організовувати свою роботу та досягати поставлених цілей.

***Актуальність теми:*** обумовлена бурхливим розвитком IT сфери та потребою в організації завдань, термінів та пріоритетів для продуктивної роботи як звичайної людини, так і співробітників в компанії.

***Предмет дослідження:*** робота та створення складної системи бази даних для можливості створення, перегляду, оновлення та видалення завдань та всіх пов’язаних з ними баз для зручного та приємного використання програми для покращення та спрощення життя людини.

***Об’єкт дослідження:*** процес та реалізація бази даних Mongo DB з використанням мови програмування Node JS.

***Мета роботи:*** розробка бази даних для інструменту керування завданнями, дослідження особливостей бази даних, проектування та налагодження.

# **РОЗДІЛ 1. АНАЛІЗ ПРОБЛЕМАТИКИ, МЕТОДІВ ТА ЗАСОБІВ ВИРІШЕННЯ ЗАДАЧІ**

* 1. Аналіз задачі, засобів та методів її вирішення

Перед тим, як почати розробляти та писати програмний код, потрібно зробити аналіз та визначити порядок виконання завдання.

Задача на курсову роботу полягає у створенні бази даних. У процесі аналізу роботи було виділено наступні умови та функції, які має містити даний програмний продукт:

* Змоделювати загальну схему додатку, схему відношень таблиць БД, продумати логіку роботи додатку та інше.
* Розробити REST API для майбутніх запитів на клієнтській частині додатку.
* Визначити, як саме користувач буде взаємодіяти з програмою.
* Розділити ролі користувачів сайту.
* Розробити можливість зміни важливих даних сайту за допомогою інтерактивного інтерфейсу для адміністраторів сайту та користувацьких даних для звичайних користувачів.
* Розробити можливість перегляду та оцінки виконаних чи поставлених задач інших користувачів системи.
* Розробити сторінки: головна, стрічка задач, категорії задач, налаштування сайту, авторизація, реєстрація, створення груп користувачів для роботи над задачами, сторінки повідомлень про помилки або успіх виконаних операцій.

Під час створення сайту пріоритети було віддано розробці основних можливостей перегляду відео курсу, покупку, перегляд карток курсів, категорій, сортування та покупку цих курсів.

Для розробки додатку було обрано таку середу, як PS. Проект розроблявся за допомогою таких веб-технологій, як HTML, CSS, PHP, JS та деяких додаткових бібліотек, наприклад Bootstrap, для полегшення процесу написання програмного коду продукту.

Для початку було вирішено розробити основу MVC системи для подальшого розвитку та розширення програмного продукту. Це стосується структури проекту, його логіки, набору класів, локацій певних файлів.

Далі було вирішено розробити основну сторінку сайту, хедер та футер, сторінку налаштувань користувача, авторизацію та реєстрацію для використання цієї інформації у подальшому. Всі дані, які надійшли чи надійдуть з клієнтської частини програми будуть перевірятися та потрапляти на сервер, який буде зберігати їх в БД у відповідну таблицю та діставати ці дані при потребі клієнта чи сервера.

Також, буде створено сторінки для адміністраторів сайту, які зможуть вносити нові актуальні дані, змінювати застарілі та видаляти непотрібні.

* 1. Аналіз існуючого програмного забезпечення за тематикою курсової роботи.

При аналізі все існуючого ПЗ за тематикою курсової роботи було виявлено безліч проектів. Кожен з них має індивідуальні принципи роботи та логіку.

Кожен сайт, який пропонує послуги продажу та перегляду курсів з програмування виконані за допомогою різних мов програмування та мають різних зовнішній вигляд. Наприклад рисунок 1.1 та рисунок 1.2 демонструють різницю у дизайні та реалізації сайту курсів з програмування.

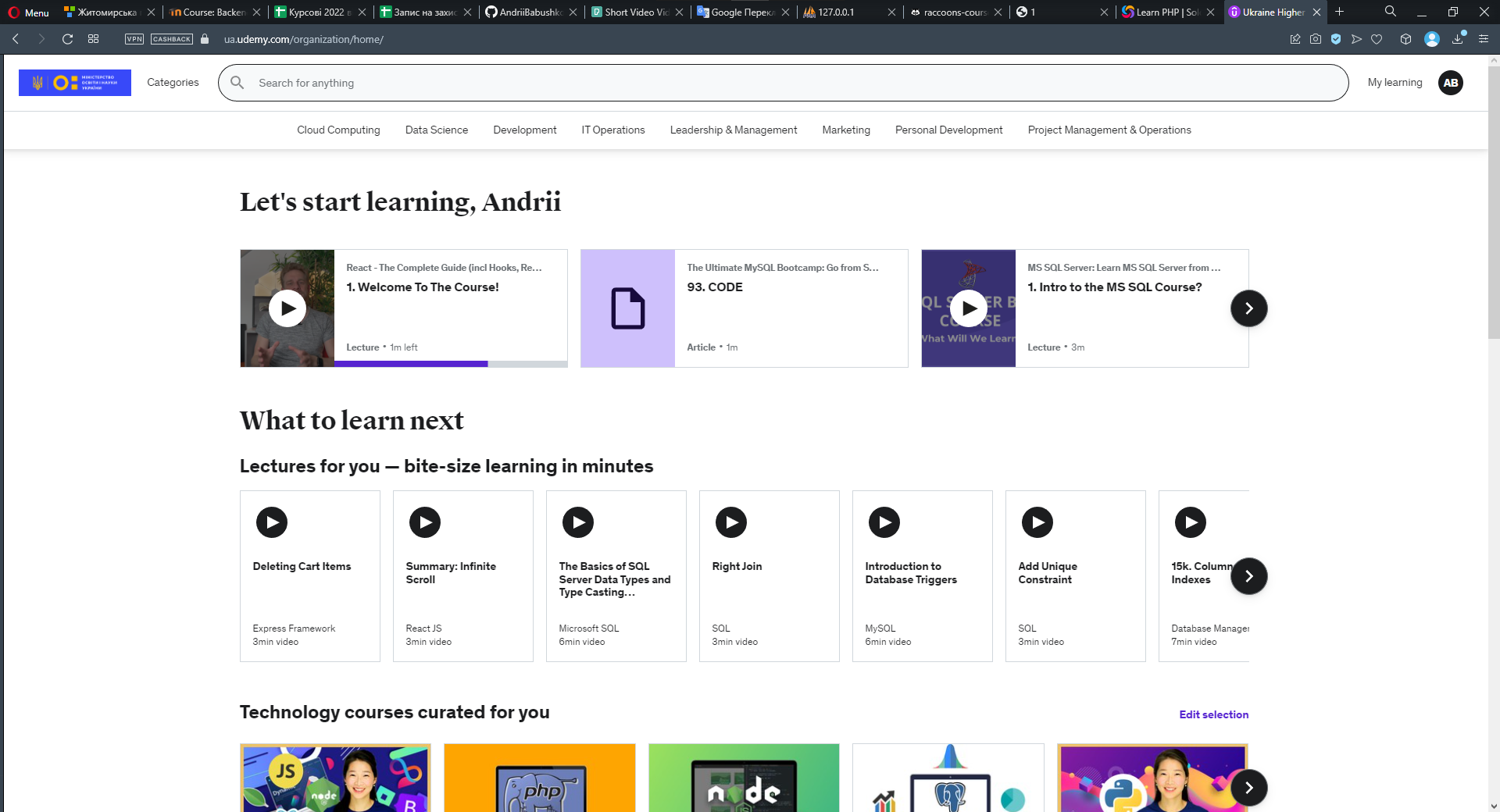


Рис. 1.1. – перший приклад реалізації сайту курсів з програмування.

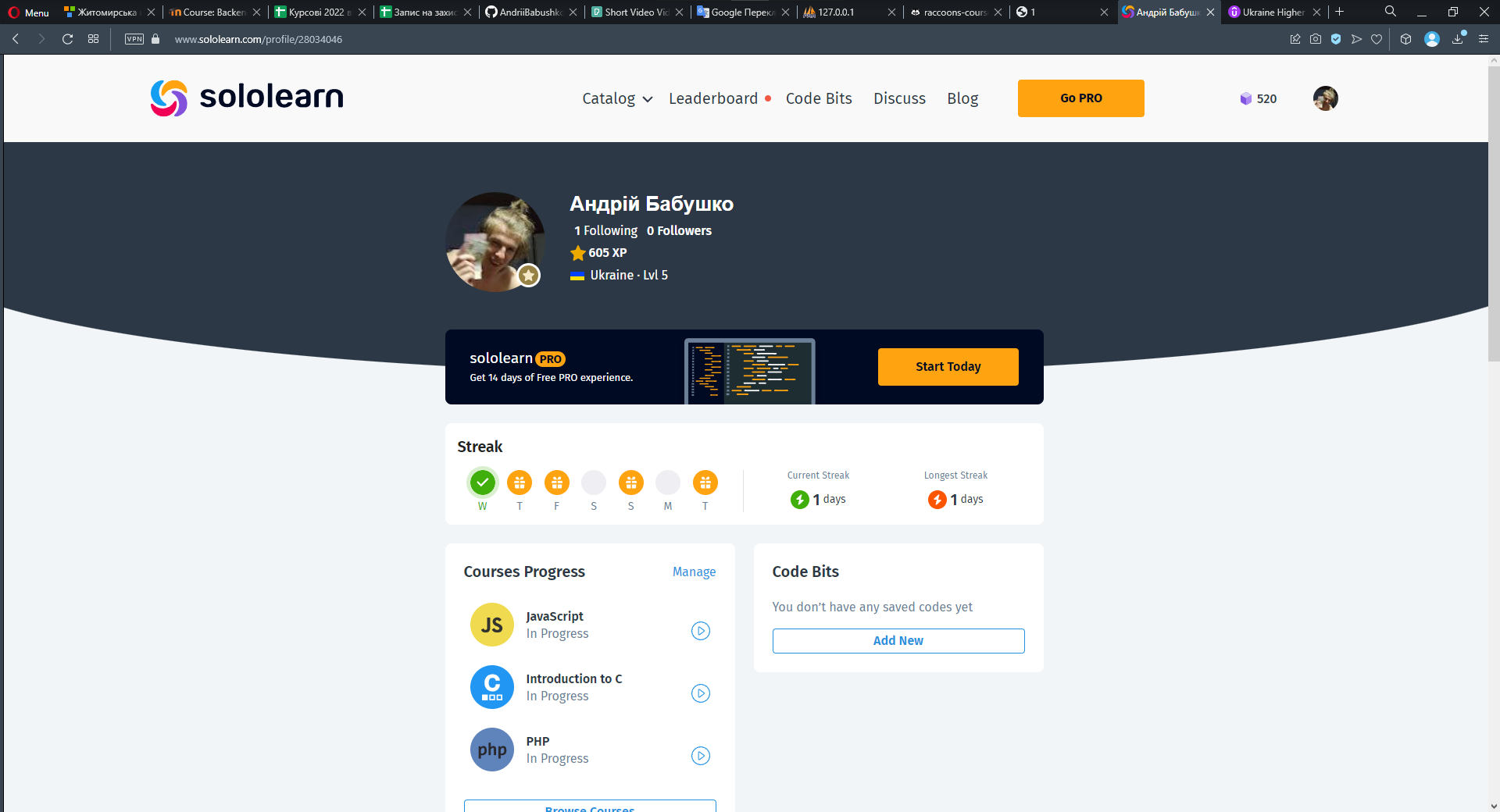


Рис. 1.2. – другий приклад реалізації сайту курсів з програмування.

На цих рисунках ми можемо побачити, що кожен сайт відрізняється користувацьким інтерфейсом та має свої особливості та дизайн. На рисунку 1 інтерфейс виглядає повноцінним та інформаційним, але дещо громіздким, в свою чергу на 2 рисунку він виглядає простіше, яскравіше та привабливіше. Але, все ж таки, у першому варіанті додаткового функціоналу більше, що дає користувачеві більше інтерактивності та можливостей вибору.

Таким чином, при розробці програми слід враховувати, що чим простіший та яскравіший інтерфейс має сайт, тим він стає більш привабливішим, але все ж повинен зберігати в собі основний функціонал, яким буде керуватися сайт. Тому це є ідеальним варіантом та вибором для розробки програмного продукту.

## Висновки з першого розділу

У ході виконання першого розділу було проаналізовано схожі варіанти реалізації інтерфейсу сайту курсів з програмування та вирішено питання способу реалізації.

Було визначено основні інструменти для написання програми, її функціонал та варіативність. Також було визначено, що основне у складі повинен мати продукт, а що другорядне. Було визначено переваги та недоліки деяких схожих програмних продуктів та обрано найкращий.

# **РОЗДІЛ 2. ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ**

## 2.1 Проектування загального алгоритму роботи програми

За результатами попередніх досліджень було визначено необхідну структуру та архітектуру програми.

Програма реалізує:

* CRUD функціонал БД(читання, створення, оновлення, видалення даних)
* Рівні доступу до змін даних сайту
* Можливість перегляду відео курсу та коментарів до відео
* Сторінки налаштувань, кошику, курсів та категорій до курсів
* Сортування курсів та категорій
* Можливість зміни мови та теми сайту

Також було проведено моделювання структури програми для візуалізації програмних можливостей та функціональних умов системи. На схемі 2.1 зображено загальну схему роботи програми.

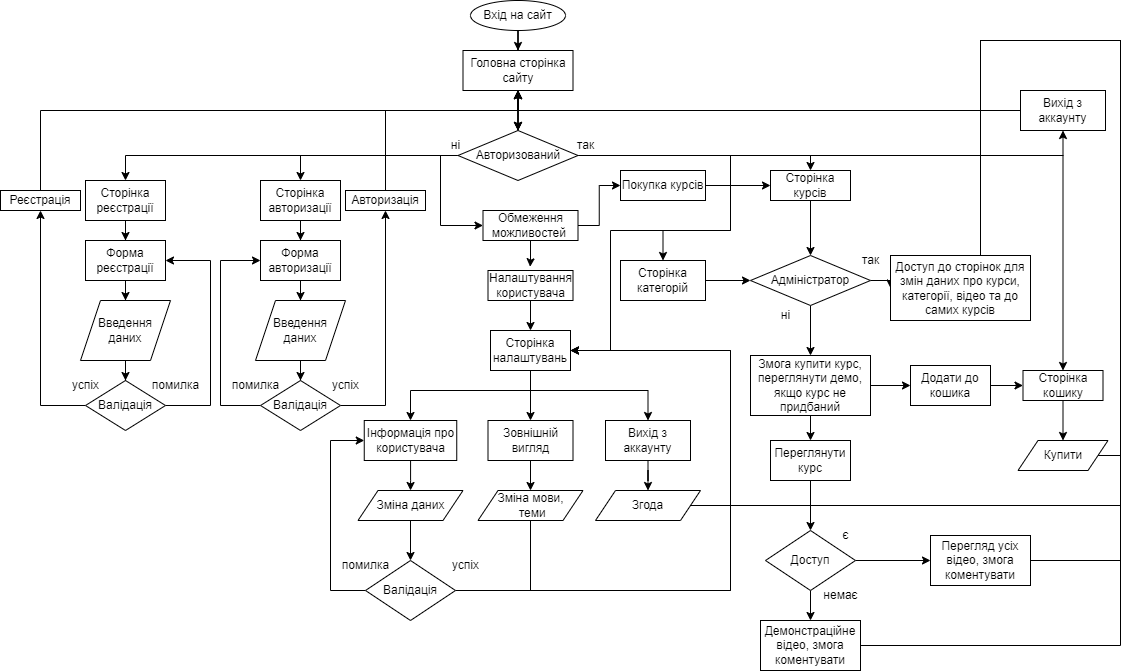


Рис. 2.1. Загальна логічна схема роботи сайту

При першому заході на сайт користувач бачить хедер сайту, де розміщені посилання на сторінки сайту, головну сторінку сайту та футер, де зазначена додаткова інформація про сайт та його розробника. Сайт завантажується на англійській мові та зі світлою темою. Також не авторизованим користувачам не надається можливість перегляду кошику та додавання курсів до цього кошику. Тому, розглянемо усі основні сторінки та їх функціонал по черзі.

Головна сторінка представляє собою невеличку сторінку з основною інформацією про сайт. Там знаходиться логотип, опис та блок з курсами, які даний сервіс пропонує своїм користувачам. Саме на цій сторінці починається знайомство користувача з сайтом.

Далі у користувача є можливість натиснути на кнопку «Почати навчання», яка переводить його на сторінку реєстрації, якщо користувач не авторизований, інакше на сторінку з курсами. Користувачеві необхідно зареєструватися, заповнивши такі поля форми, як ім’я, прізвище, номер телефону, електронна пошта, пароль та підтвердження паролю і після цього натиснути на відповідну кнопку. Після цього на клієнтська частина перевіряє дані на правильність введення та виводить помилки, якщо вони є, інакше відправляється запит на сервер, де до БД додається користувач та створюється для нього пустий кошик для подальшої роботи. Користувач не зможе отримати повного доступу до функціоналу сайту не провівши авторизацію при цьому. Тому далі користувачу необхідно перейти на форму авторизації та заповнити 2 поля: електронна пошта та пароль і натиснути відповідну кнопку. Сервер перевіряє введені дані, виводить помилки, якщо вони наявні та авторизує користувача записуючи ці дані у сесію.

Саме після цих дій відкриваються усі можливості сайту. Насправді, не всі які є, бо для цього на сайт необхідно зайти з правами адміністратора, які можна отримати лише зайшовши у редактор БД phpMyAdmin і змінивши потрібному користувачеві поле. Було проаналізовано, що потреби динамічно змінювати права користувачів не було, тому даний функціонал не було реалізовано.

Отже, після успішної авторизації користувач отримує безліч можливостей сайту. Деякі з них це придбати курс та мати доступ його повністю переглядати, залишати коментарі до курсів та змінювати свій особистий профіль на сайті. Для адміністраторів реалізовано додатково можливість змінювати дані про категорії, курси та відео до цих курсів, для яких створено додаткові сторінки з формами. Всі дані, які були введені адміністраторами перевіряються на правильність за заносяться лише у випадку їх коректності, інакше адміністратору буде повідомлено про помилки у відповідних полях.

2.2 Розробка функціональних алгоритмів роботи програми

Для представлення статичної структури моделі системи в ООП використовується «Діаграма класів». На діаграмі відображуються основні зв’язки між класами, об’єктами та полями класів.

Діаграма класів складається з великою кількістю елементів, які відображають інформацію про конкретну область. Ці дані переводяться за допомогою мови UML, таких як класи, інтерфейси та відносини між їх компонентами.

У даному проекті було створено багато класів для правильної та коректної роботи програми, які містять в собі спеціальні поля та методи. Одними з них є класи у папці «core» у корені проекту. Тут містяться класи для роботи з ядром системи, таблицями бази даних, класами які керують завантаженням представлень сторінок, генерацією помилок та універсальним класом з корисними методами. Усі вони зображені на діаграмі класів (Рис. 2.2.).

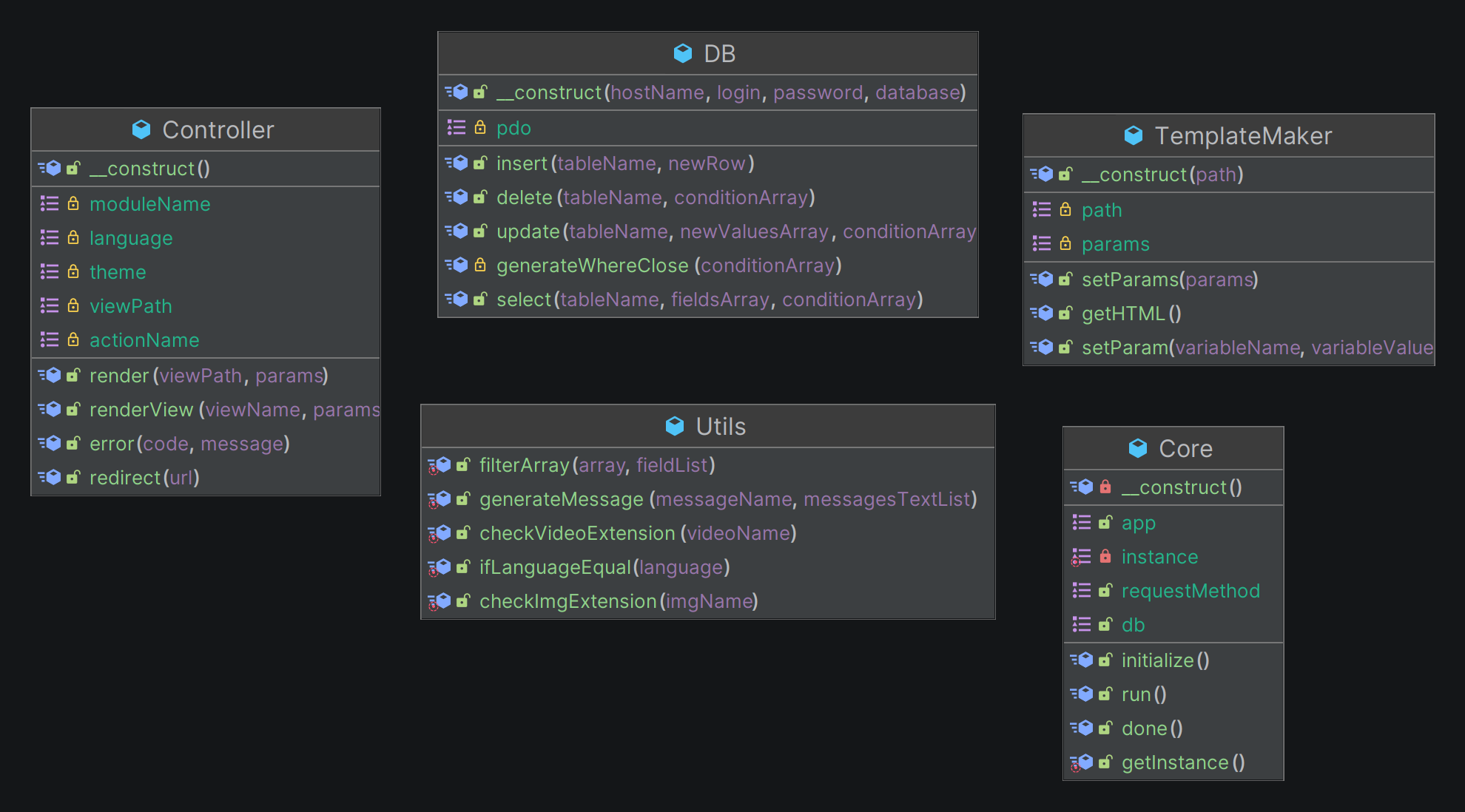


Рис. 2.2. Діаграма класів папки «core»

Іншими корисними класами у проекті є ті, які розміщені в папці «controllers». Вони керують роботою кожної сторінки сайту, генерують HTML код, виконують логічні операції над даними, заносять дані до БД та інше. Усі ці класи зображені на наступній діаграмі класів (Рис. 2.3.).

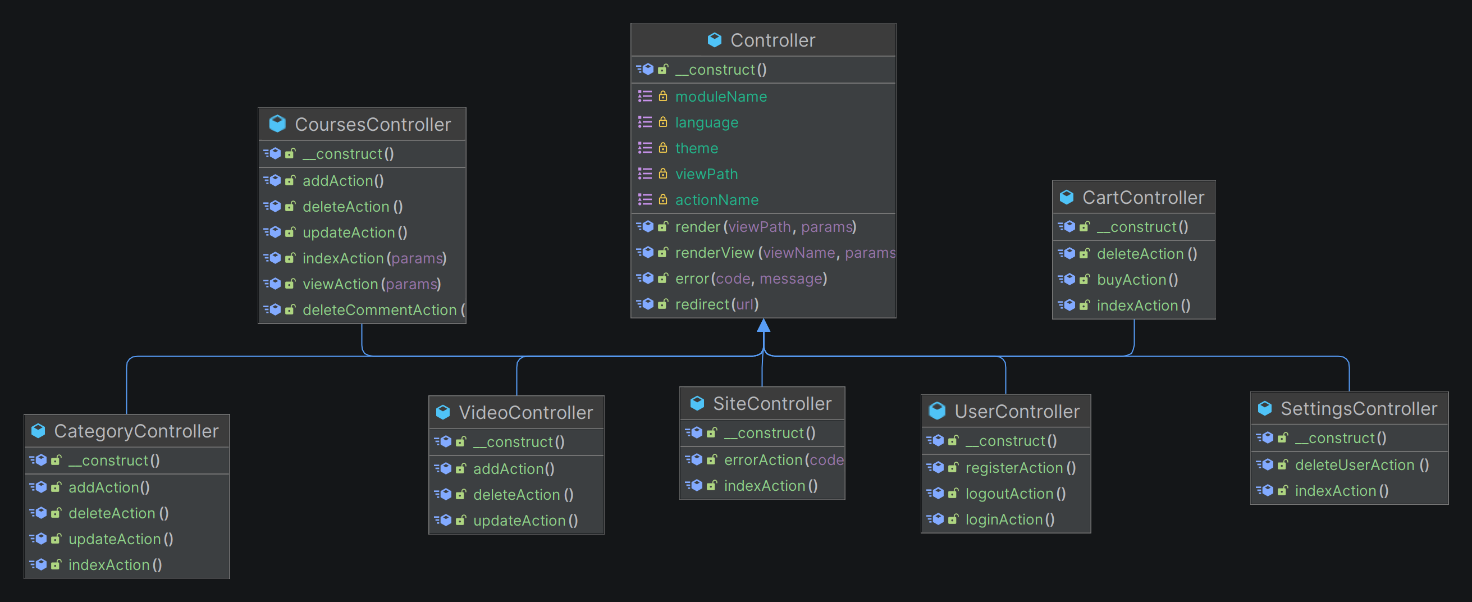


Рис. 2.3. Діаграма класів папки «controllers»

Також варто зазначити, що БД теж має свою певну структуру та відносини до кожної з таблиць. Відносини будуються за рахунок ключів 2-ух видів: первинного та зовнішнього ключів. Відносини, які побудовані за рахунок ключів між таблицями зображено на рисунку 2.4.

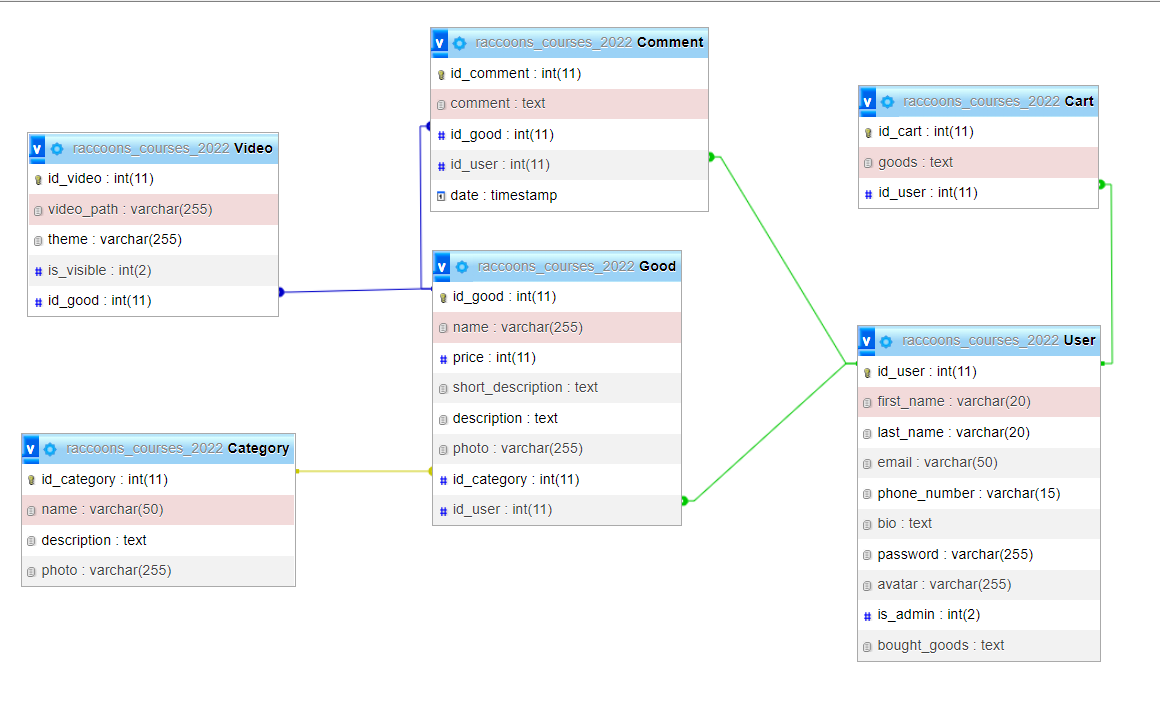


Рис. 2.4. Діаграма відносин таблиць у БД всього проекту

У даній курсовій роботі реалізуються малі за об’ємом та швидкі для виконання методи зчитування, оновлення, видалення та запису даних з БД в окремому класі. Це дає змогу програмі швидко реагувати на зміни даних, їх оновлювати та використовувати для відображення на клієнтській стороні програми.

## 2.3 Розробка програмного забезпечення

Для розробки програмного забезпечення було створено 7 різних папок, кожна з яких виконує свої певні функції та методи. Основною папкою, яка є ядром системи має назву «core». Вона має в собі 5 класів, які керують класами в інших папках. Програма починає запуск саме з цієї папки і саме з класу «Core». Опишемо цей клас в табличці нижче.

Клас «Core»

|  |  |
| --- | --- |
| Метод | Опис |
| private function \_\_construct() | метод для ініціалізації даних класу та старту програми |
| public static function getInstance() | метод який повертає екземпляр класу Core та дає змогу користуватися його можливостями |
| public function initialize() | метод розпочинає сесію та налаштовує з'єднання з БД |
| public function run() | метод який зчитує рядок адреси браузера та формує з нього рядок для виклику певного методу та певного класу для відображення сторінки у браузері |
| public function done() | метод передає дані зформовані в класі на сторінку та відображує її у браузері |

У даному класі проводиться ініціалізація усього проекту, його налаштувань та виклику інших конструкторів класів та методів. Ось один з методів, який надає змогу звертатися до екземпляру класу «Core» ззовні та отримувати дані з нього:

public static function getInstance(): ?Core  
{  
 if (empty(self::$instance)) {  
 self::$instance = new self();  
 }  
  
 return self::$instance;  
}

Також один з 3 основних методів з якого саме починається використання інших методів класу, наприклад «run» чи «done». Якщо їх викликати раніше методу «initialize», програма може зламатися, бо не буде з’єднання з БД та не буде доступу до супер глобального асоціативного масиву сесії.

public function initialize(): void  
{  
 session\_start();  
 $this->db = new DB(*DATABASE\_HOST*, *DATABASE\_LOGIN*, *DATABASE\_PASSWORD*, *DATABASE\_BASENAME*);  
 $this->requestMethod = $\_SERVER['REQUEST\_METHOD'];  
}

Метод «run» є головним у пошуку класу для виклику та генерації сторінки, яку запросив користувач сайту, а також у генерації помилки 404, якщо вона відсутня.

Метод «done» завершує генерацію HTML коду сторінки та віддає його відповідному викликаному класу у методі «run». Далі наведено код цього методу:

public function done(): void  
{  
 $pathToLayout = "themes/{$this->app['theme']}/layout.php";  
  
 $templateMaker = new TemplateMaker($pathToLayout);  
 $templateMaker->setParam('content', $this->app['actionResult']);  
 $templateMaker->setParam('language', $this->app['language']);  
 $templateMaker->setParam('theme', $this->app['theme']);  
 $templateMaker->setParam('title', $this->app['pageTitle']);  
 $html = $templateMaker->getHTML();  
  
 echo $html;  
}

Існує це в цій папці такий клас, як Controller, який є батьком усіх контролерів у папці «controllers». Він містить у собі корисні методи, один з яких повертає HTML код сторінки, в якій він був викликаний. Всі ці та схожі методи описані в наступній табличці.

Клас «Controller»

|  |  |
| --- | --- |
| Метод | Опис |
| public function \_\_construct() | метод для ініціалізації полів класу певними даними |
| public function render(string $viewPath = null, array $params = null) | метод який генерує сторінку за певним шляхом та передає до неї параметри(змінні) |
| public function renderView(string $viewName, array $params = null) | метод генерує певний вигляд сторінки за назвою та передає до неї параметри |
| public function redirect(string $url) | метод який змінює рядок адреси браузера |
| public function error(int $code, string $message = null) | метод який генерує помилку за певним кодом та передає до неї повідомлення |

У цьому класі всі методи є корисними та використовуються постійно, але одним з основних є метод «render», який отримує HTML код та передає його до певної сторінки разом з параметрами, які до нього передаються у вигляді асоціативного масиву. Код даного методу наведено нижче:

public function render(string $viewPath = null, array $params = null): bool|string  
{  
 if (empty($viewPath))  
 $viewPath = $this->viewPath;  
  
 $templateMaker = new TemplateMaker($viewPath);  
  
 if (!empty($params)) {  
 $templateMaker->setParams($params);  
 }  
  
 return $templateMaker->getHTML();  
}

Також варто звернути увагу на теж досить корисний метод в цьому класі з назвою «renderView», який генерує певне представлення в залежності від назви цього представлення та де це представлення знаходиться. Ось код цього методу:

public function renderView(string $viewName, array $params = null): bool|string  
{  
 $path = "views/{$this->theme}/{$this->moduleName}/{$this->language}/{$viewName}.php";  
  
 $templateMaker = new TemplateMaker($path);  
  
 if (!empty($params)) {  
 $templateMaker->setParams($params);  
 }  
  
 return $templateMaker->getHTML();  
}

## Висновки з другого розділу

У другому розділі було розглянуто структуру програми у вигляді діаграм класів, БД та структурної діаграми, її основні можливості, описано основні методи та класи, які необхідні для коректної роботи даної програми.

# **РОЗДІЛ 3. ОПИС РОБОТИ З ПРОГРАМНИМ ДОДАТКОМ ТА ЙОГО ТЕСТУВАННЯ**

## 3.1 Опис роботи з програмним додатком

Після запуску сайту перед користувачем з’являється головна сторінка сайту, яка має усю інформацію про те, що це за сайт та що він пропонує. Також кожна сторінка має хедер та футер, звідки можна перейти на інші сторінки сайту та ознайомитися з ним ще більше. Вигляд який має дана сторінка з темною темою зображено на рисунку 3.1.



Рис. 3.1. Вигляд головної сторінки сайту з темною темою

Користувач може перейти на будь-яку зі сторінок, які зазначені в хедері сайту, але він повинен знати, що усіх можливостей він не зможе мати доти, доки не пройде реєстрацію та авторизацію на сайті, тому щоб отримати доступ він натискає на кнопку «Register» якщо стоїть англійська мова, а якщо українська, то, відповідно, «Реєстрація». Вигляд сторінки описаний на рисунку 3.2.

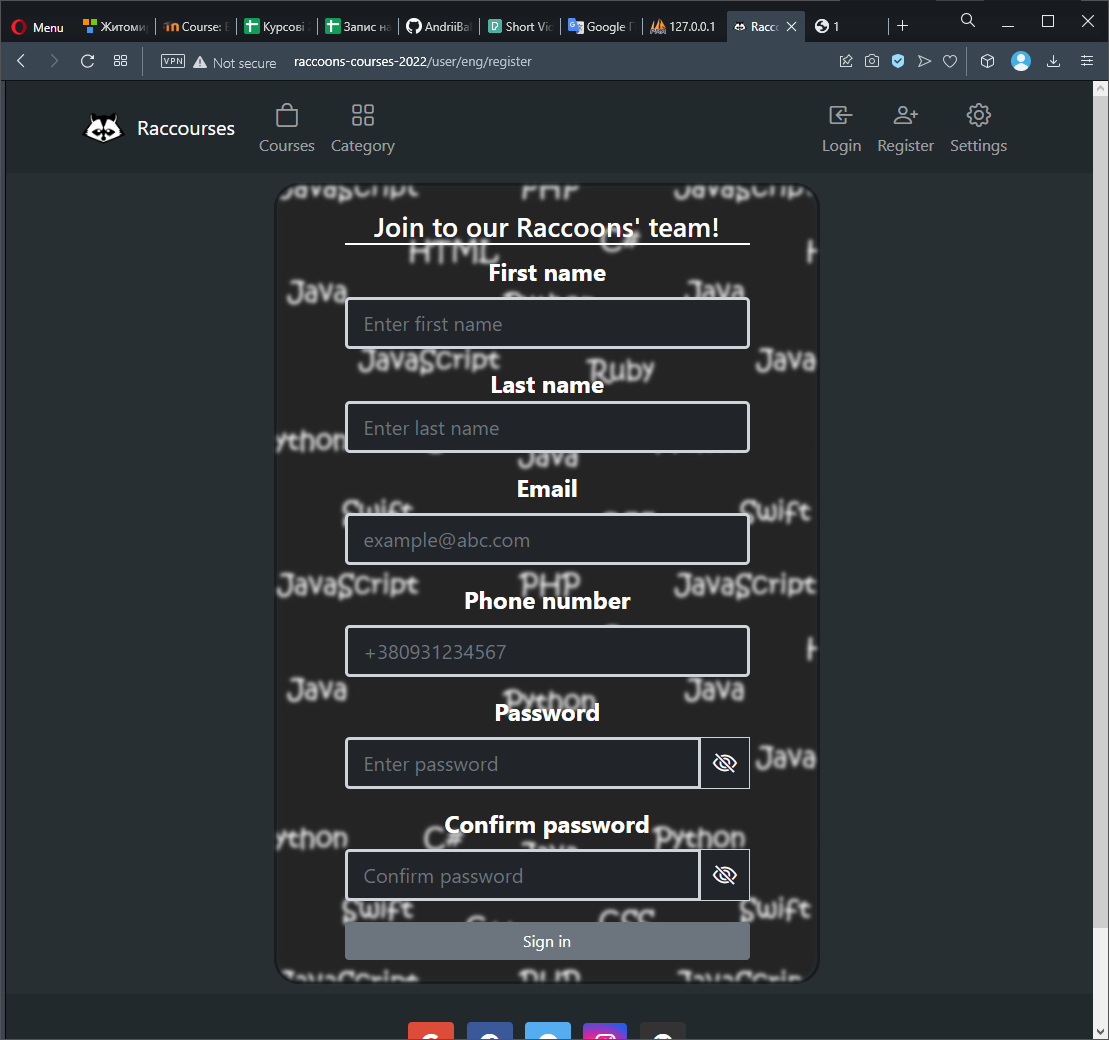


Рис. 3.2. Вигляд сторінки реєстрації користувача

Отже, після заповнення усіх полів форми і натиску на кнопку «Зареєструватися» або «Sign in» поля перевіряються на коректність введення та відправляються на сервер для запису у БД. Після реєстрації з’являється ось таке повідомлення, яке описано на рисунку 3.3.

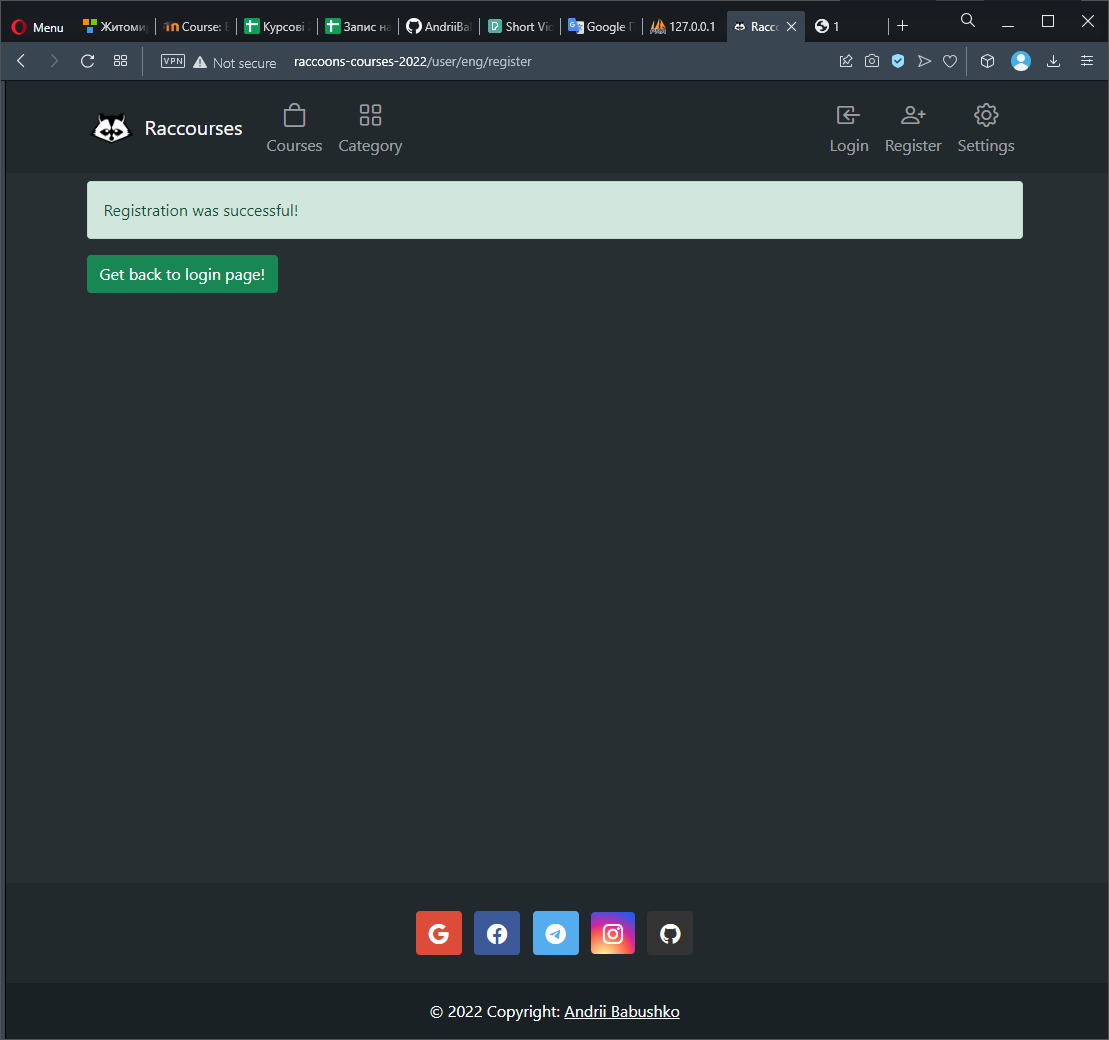


Рис. 3.3. Вигляд сторінки повідомлення після реєстрації користувача

Таких повідомлень є досить багато на різних сторінках сайту, тому далі їх демонстрації не буде.

Після реєстрації необхідно провести обов’язково авторизацію, щоб отримати доступ до можливостей сайту. Сторінка авторизації користувача зображена на рисунку 3.4.

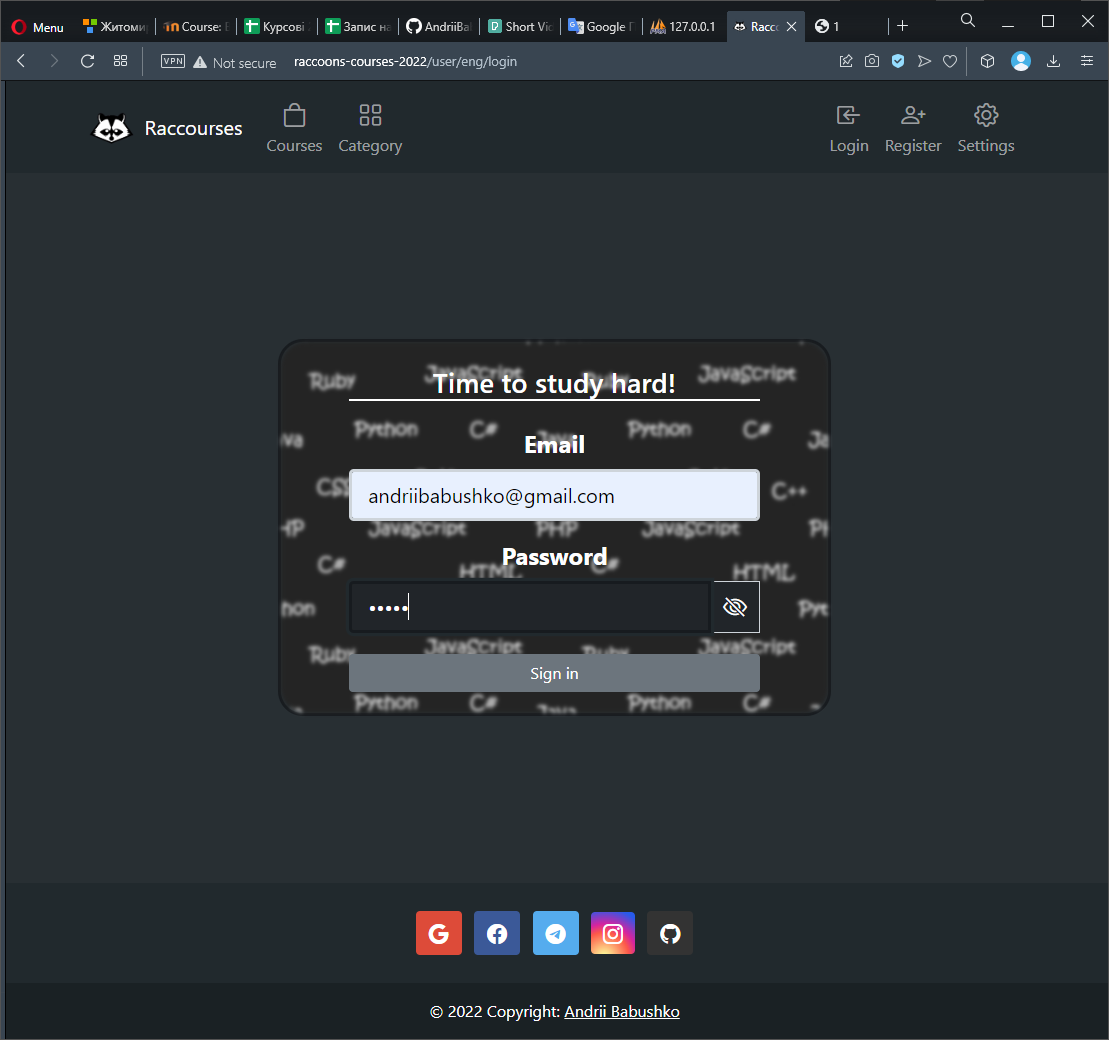


Рис. 3.4. Вигляд сторінки авторизації користувача

Одразу після успішної авторизації користувача хедер сайту змінюється. Його зміни зображено на наступному рисунку 3.5.



Рис. 3.5. Вигляд хедеру сайту

Далі користувач може повністю використовувати весь функціонал сайту, який надано саме вільним користувачам. У адміністраторів сайду дещо інший вигляд та є додаткові можливості для зміни та редагуванню контенту сайту.

Сторінка налаштувань сайту. Тут користувач може змінити свої дані, які були внесені при реєстрації на сайті, змінити аватар, написати про себе біографію, змінити мову та тему сайту. Вигляд цієї сторінки описано на рисунках 3.6, 3.7 та 3.8.



Рис. 3.6. Вигляд сторінки налаштувань даних про користувача

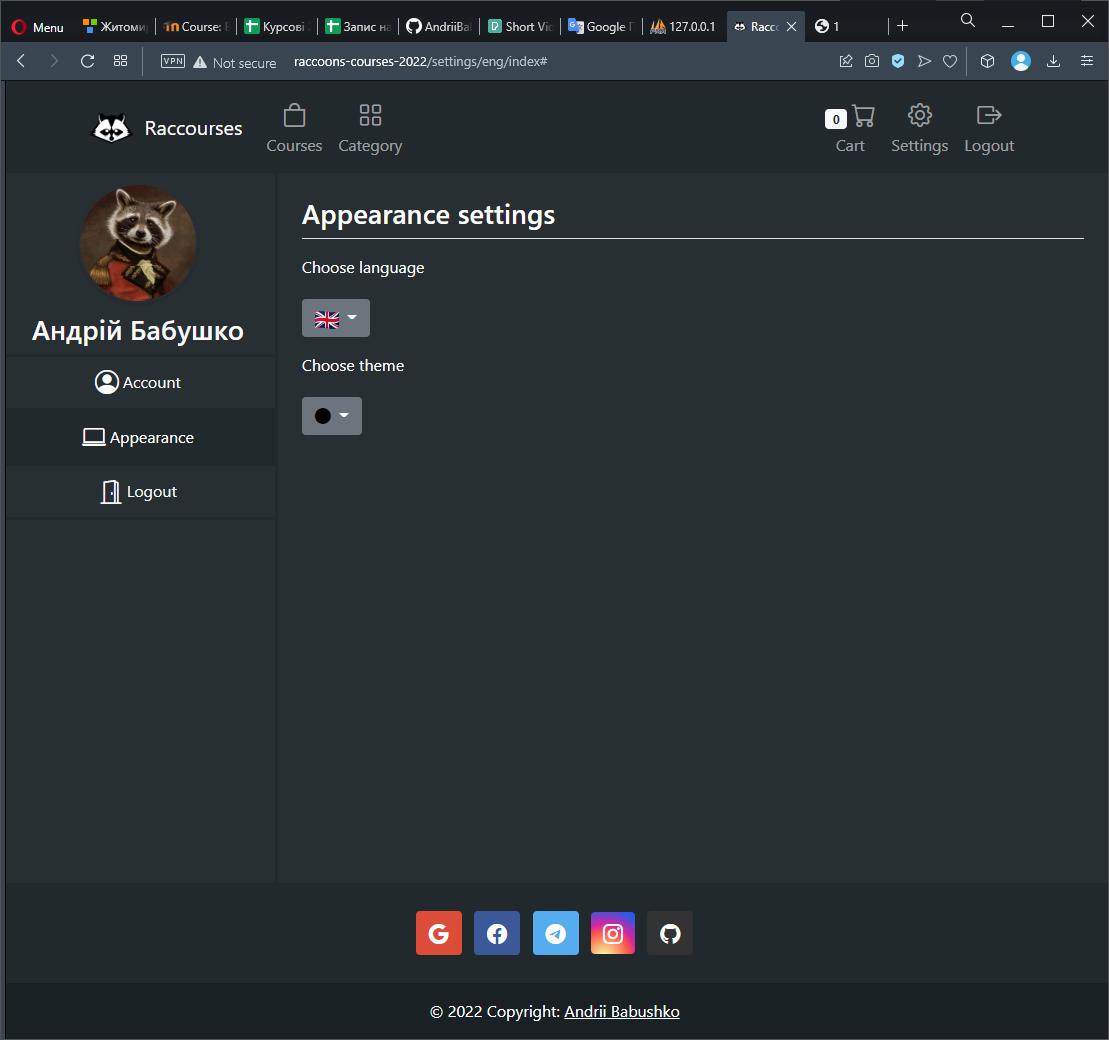


Рис. 3.7. Вигляд сторінки налаштувань сайту(мови та теми)

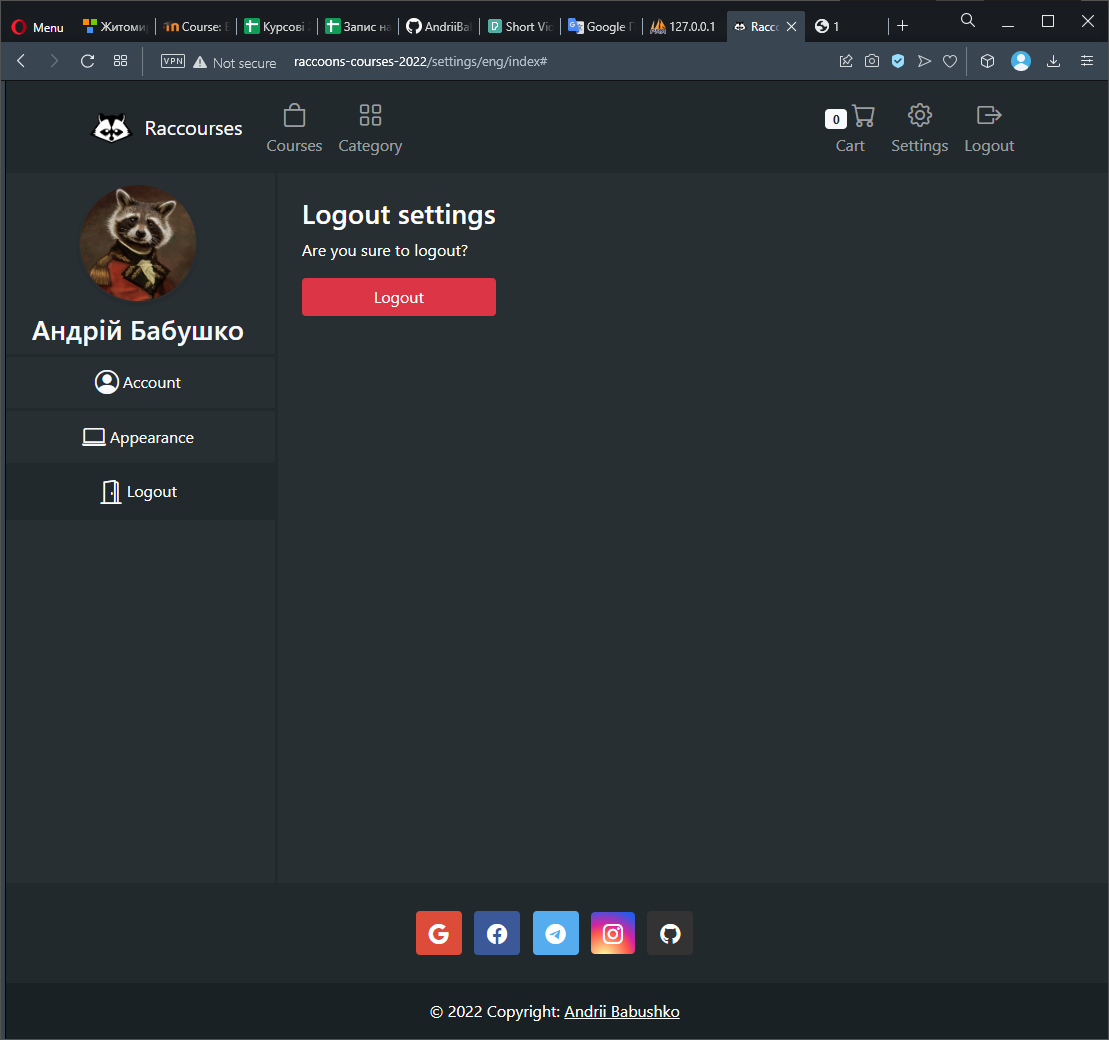


Рис. 3.8. Вигляд сторінки налаштувань для виходу користувача з облікового запису

Після зміни даних про користувача їх буде оновлено в БД та на сайті. Вигляд сторінки налаштувань користувача після оновлення даних описано на рисунку 3.9.

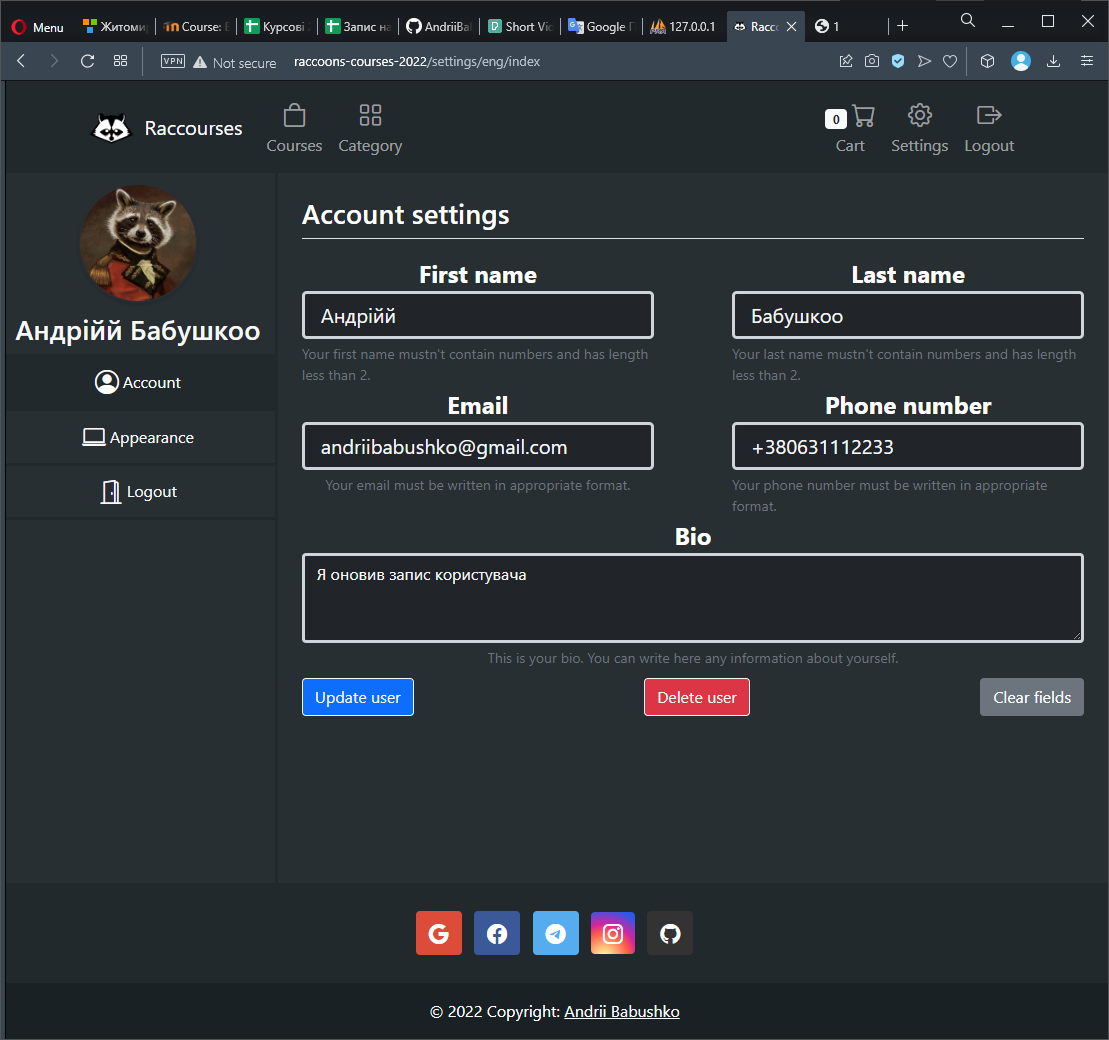


Рис. 3.9. Вигляд сторінки налаштувань після зміни інформації облікового запису користувача

Також на рисунку 3.10. показано результат зміни мови та теми у пункті налаштувань зовнішнього вигляду сайту.

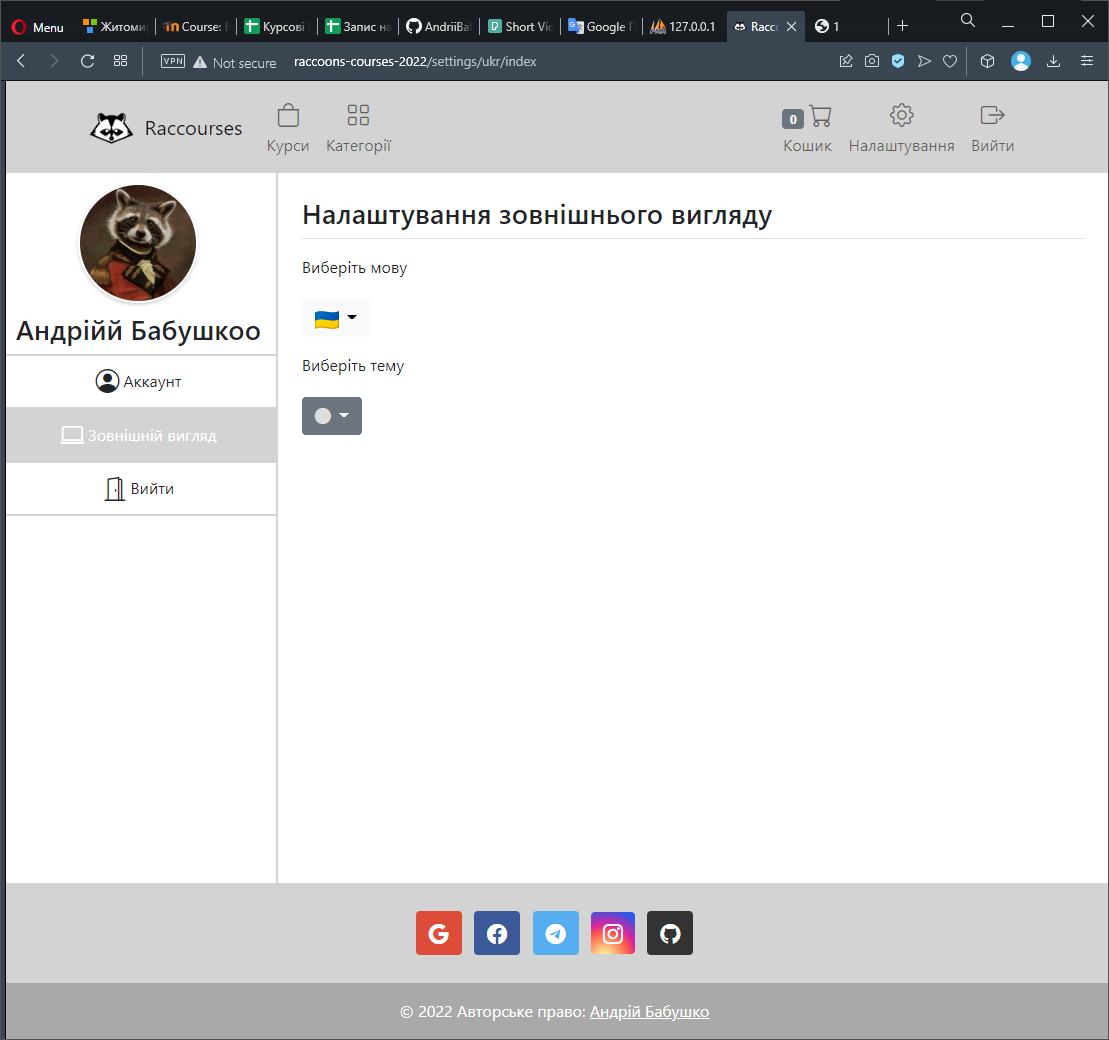


Рис. 3.10. Вигляд сайту після змін зовнішнього вигляду

Отже, після реєстрації користувач хоче одразу розпочати навчання, але для цього йому необхідно вибрати певний курс для навчання. На рисунку 3.11 зображено вигляд сторінки з курсами.

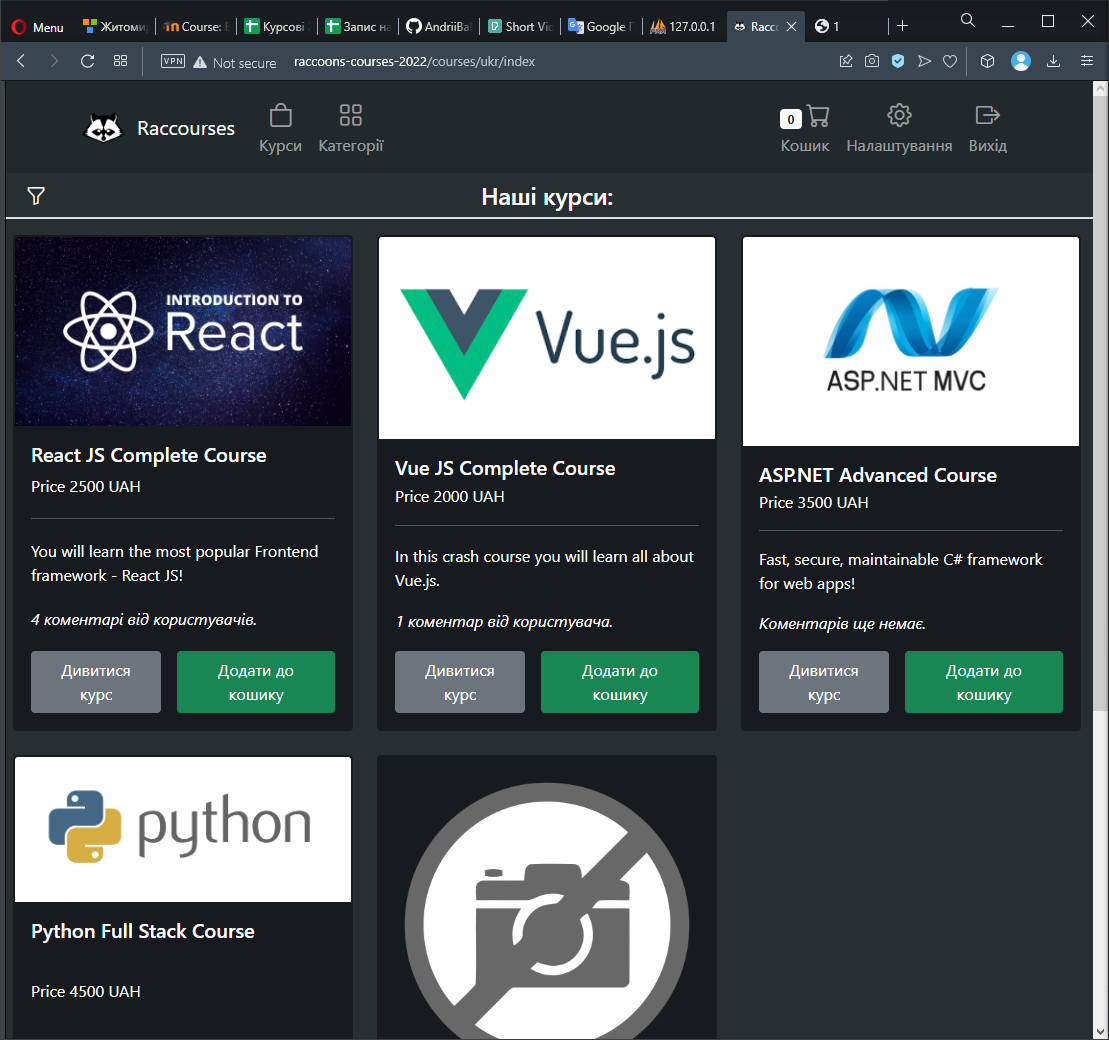


Рис. 3.11. Вигляд сторінки курсів

Також є можливість перегляду курсів по категоріям. Це можна здійснити перейшовши на сторінку з категоріями та натиснути на кнопку «Переглянути категорію». Вигляд сторінки описано на рисунку 3.12.

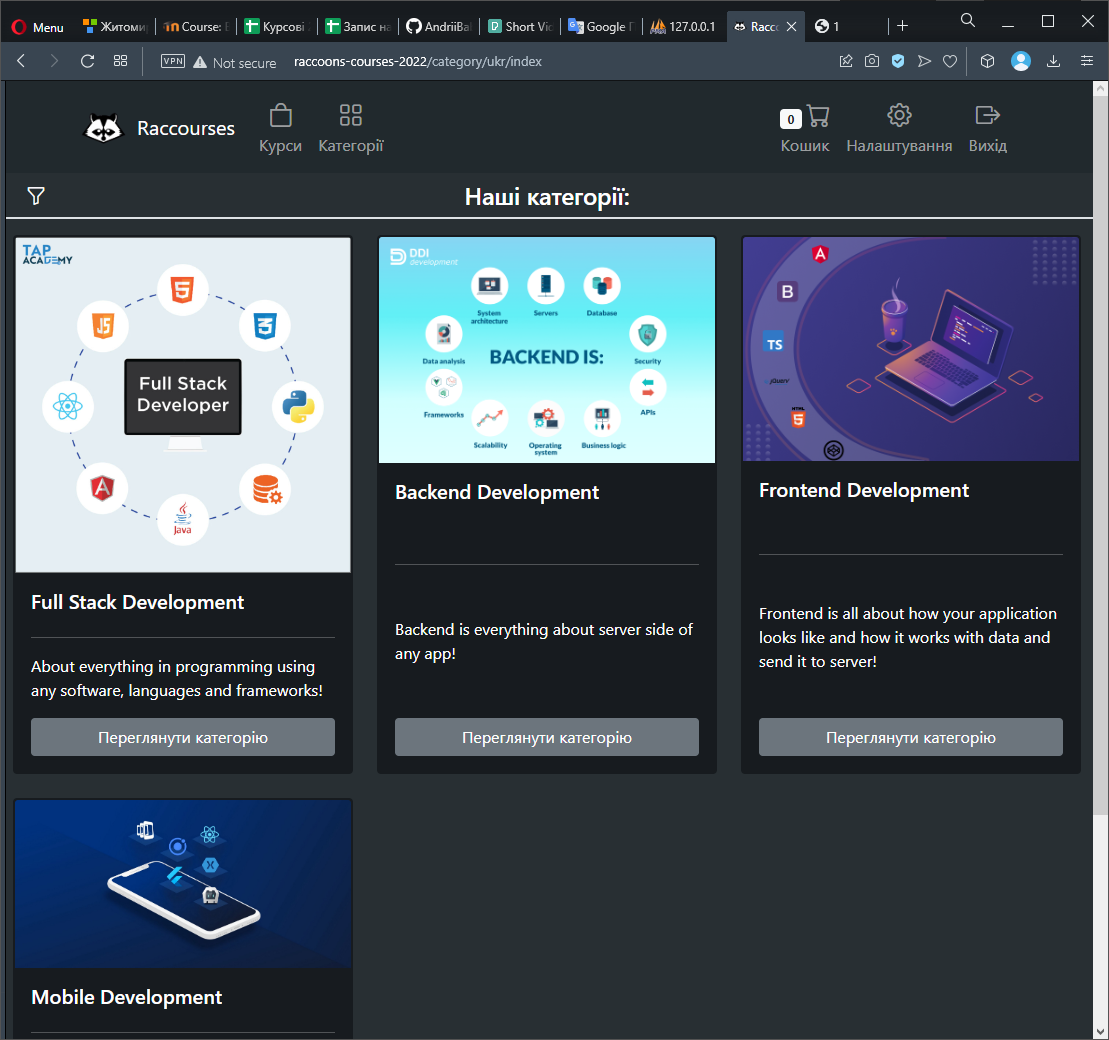


Рис. 3.12. Вигляд сторінки категорій

Також на сторінках «Курси» та «Категорії» можна здійснити фільтрацію або пошук. Для цього треба натиснути на кнопку, яка знаходиться зліва від заголовку сторінки. Після натискання відкриється бокове меню для сортування. Воно зображено на рисунках 3.13 та 3.14.

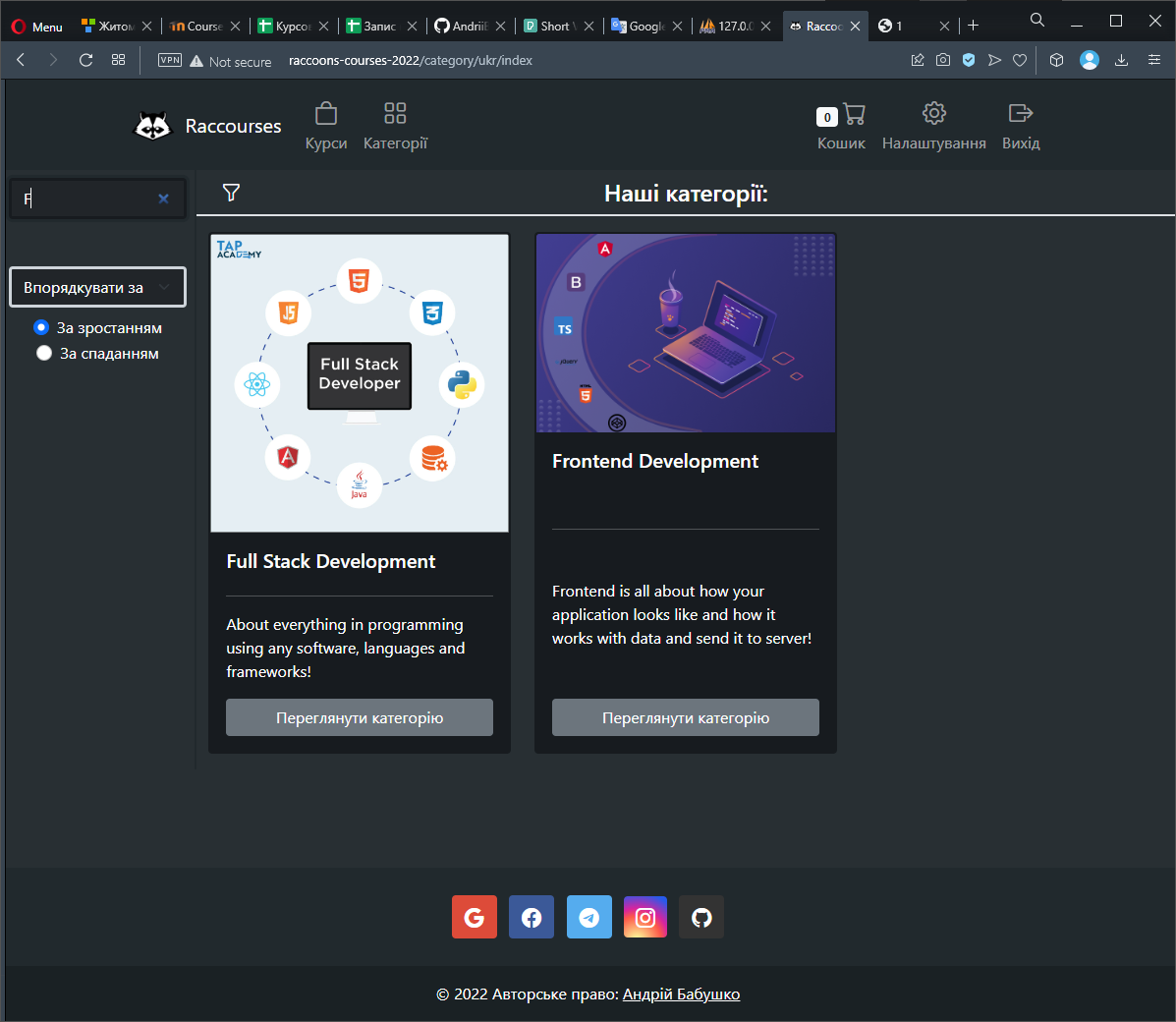


Рис. 3.13. Вигляд сторінки категорій після пошуку

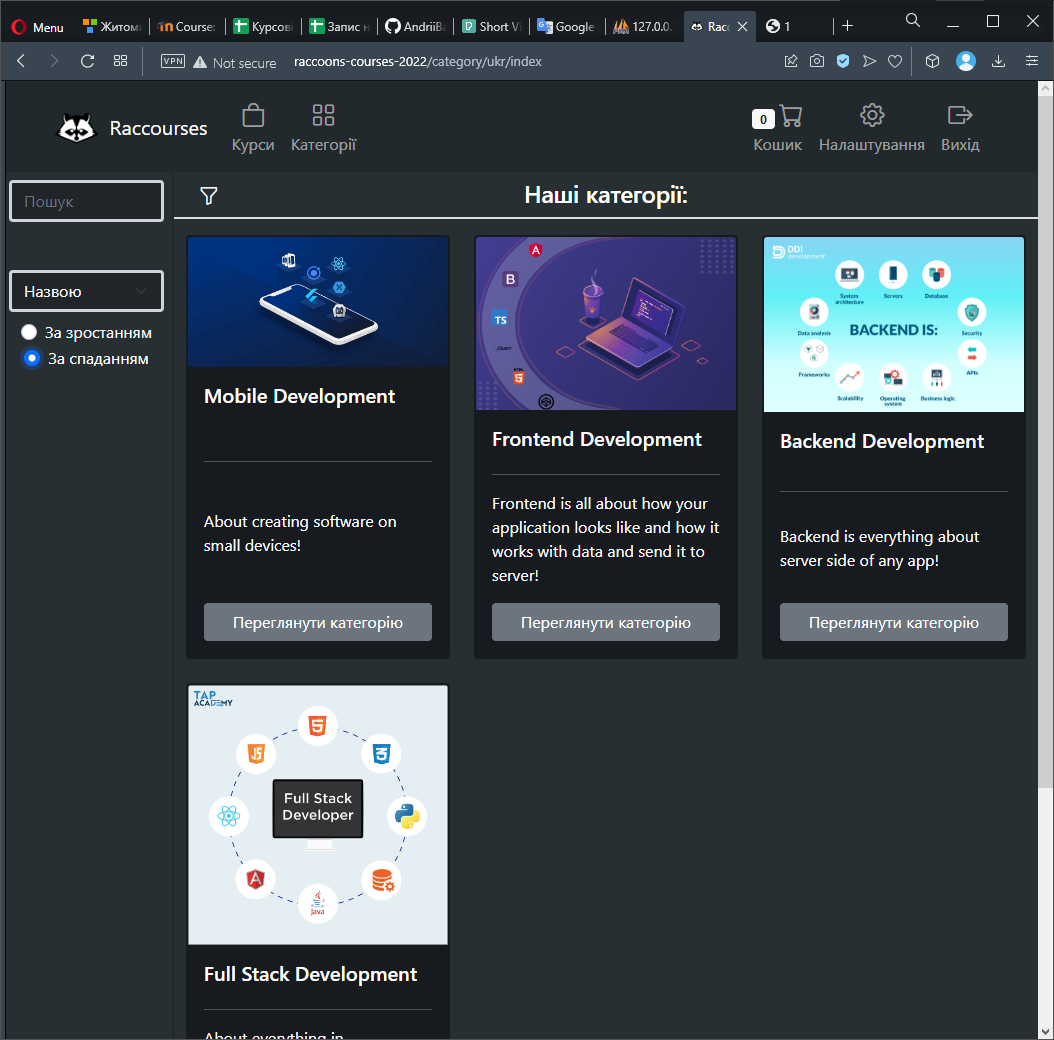


Рис. 3.14. Вигляд сторінки категорій після впорядкування

Щоб переглянути користувачеві було зрозуміло з чого складається курс, що він має при собі, хто його створив, скільки коментарів у курсу треба зайти на сторінку курсу та переглянути демонстраційний ролик, який є безкоштовним для усіх користувачів, які ще не купили даний курс. Для перегляду наступних, курс необхідно придбати. Вигляд сторінки перегляду курсу зображено на рисунку 3.15.

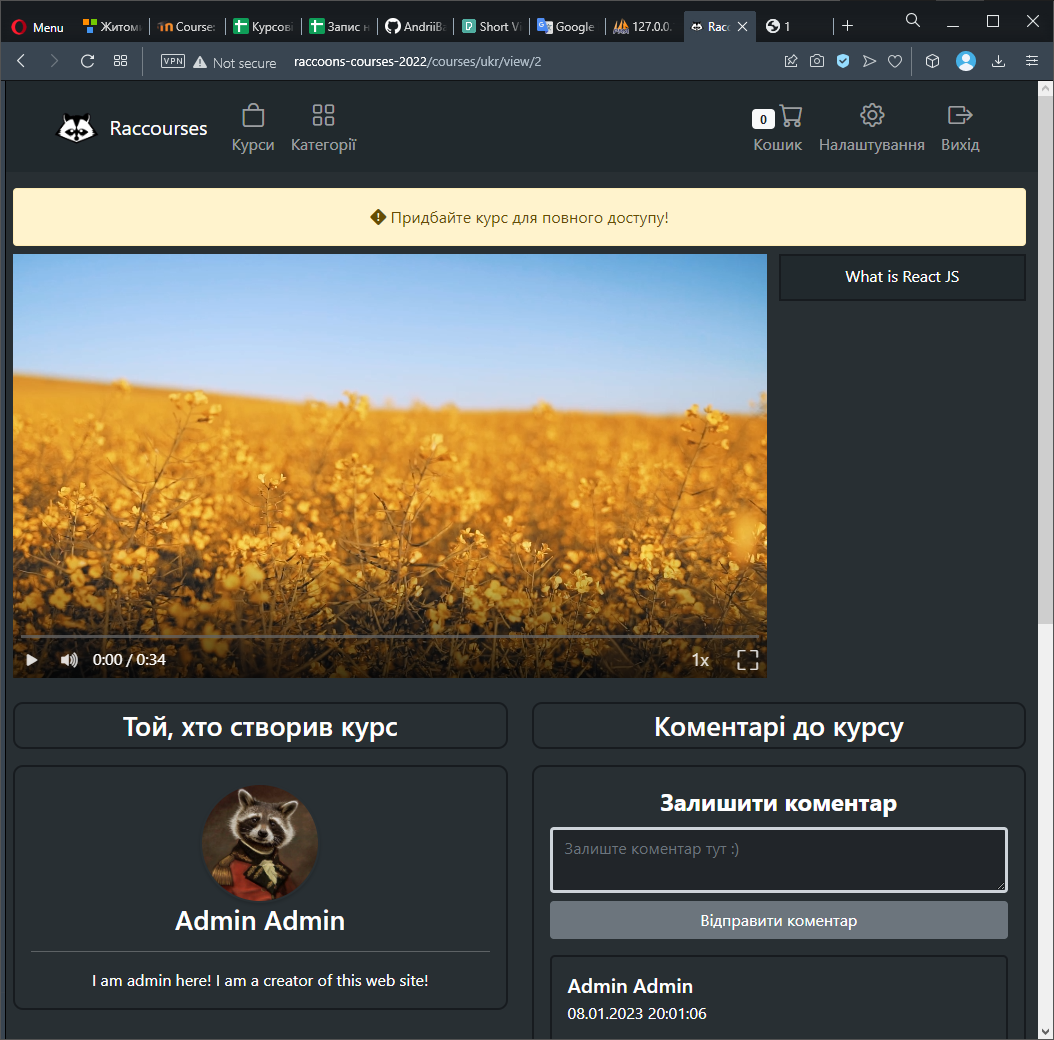


Рис. 3.15. Вигляд сторінки перегляду продукту

Користувач може написати коментар до курсу у будь-який момент, навіть якщо у курсу ще немає відео, а його тільки створили.

Якщо демонстраційний ролик користувачеві сподобався, він може додати товар у кошик та придбати його. Наступний рисунок 3.16 описує вигляд кошику після обирання курсу для покупки.

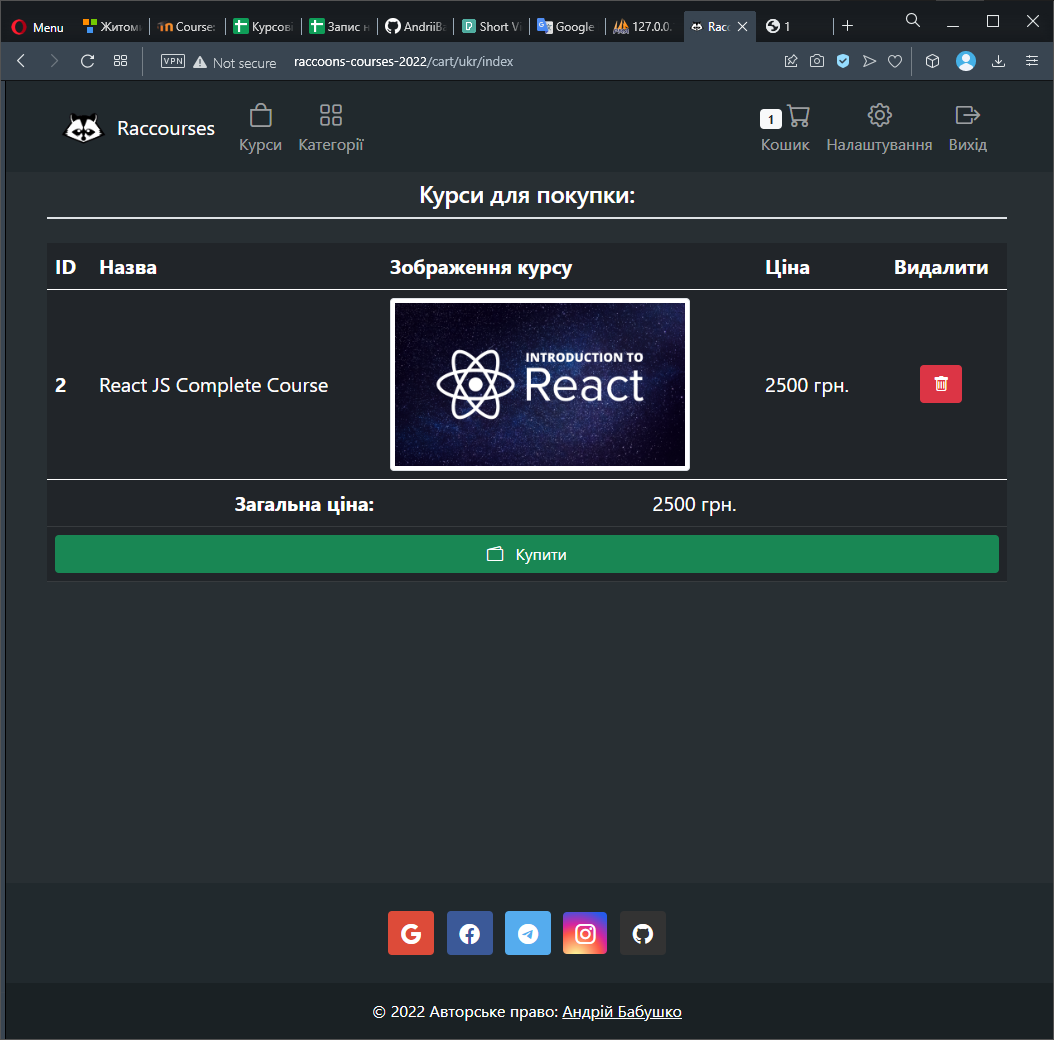


Рис. 3.16. Сторінка «Кошик» після обирання продукту для покупки

Також для зручності користування кошиком було вирішено розробити окремий вигляд кошику для менших екранів. На рисунку 3.17 зображено мобільну версію кошику.

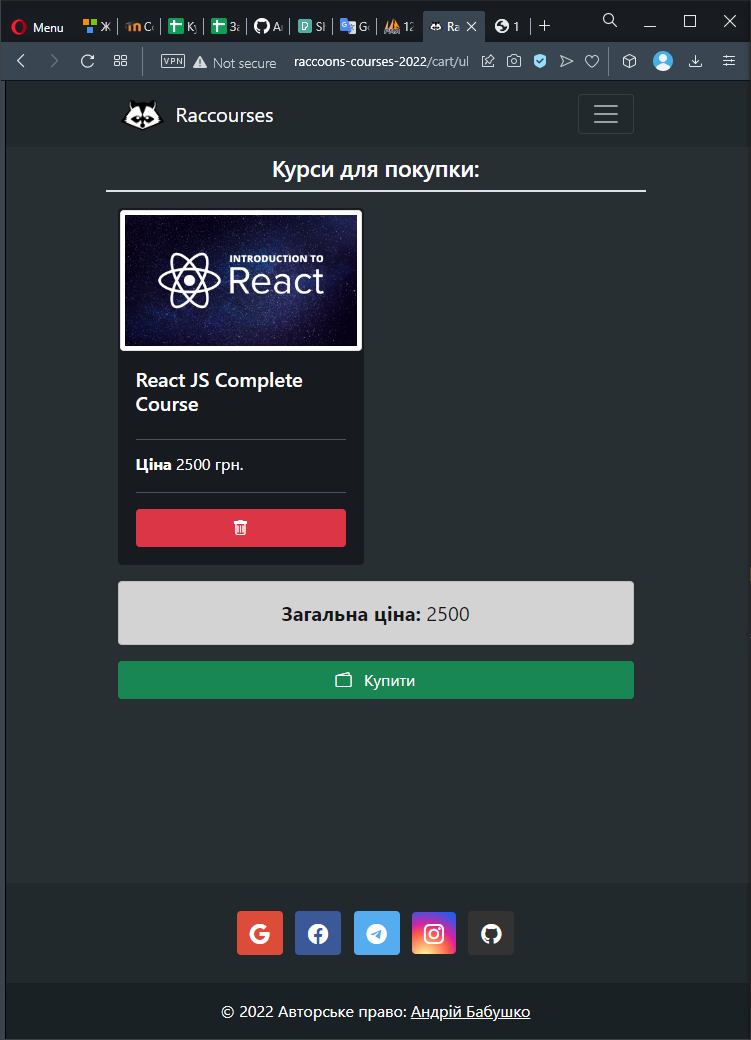


Рис. 3.17. Сторінка «Кошик» для мобільних телефонів

Після покупки з’являється повідомлення про покупку і користувач може перейти до перегляду курсу у повній мірі з повним доступом до всіх відео. Сторінку курсу після повного доступу зображено на рисунку 3.18.

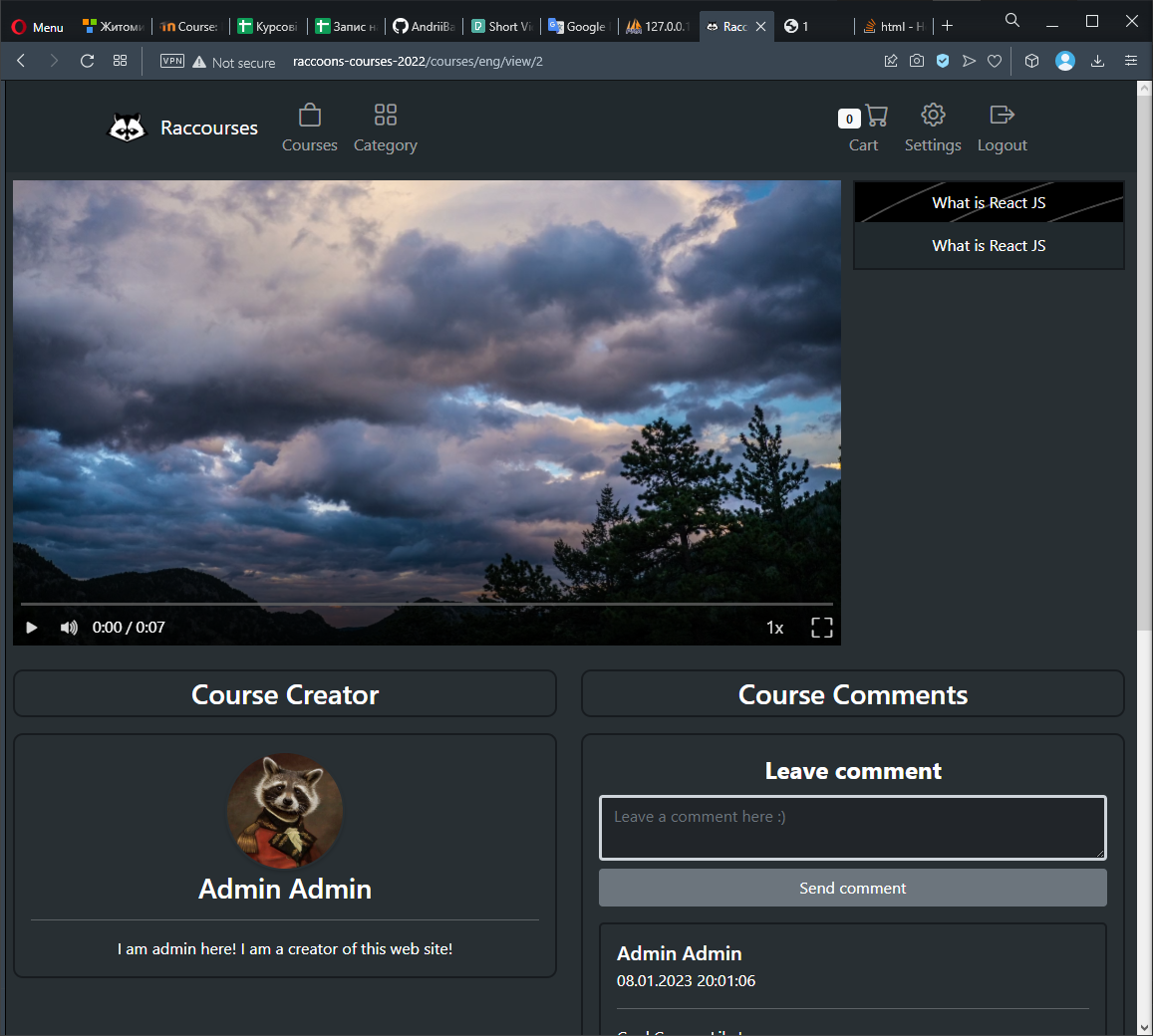


Рис. 3.18. Вигляд сторінки курсу після покупки

Такий вигляд мали сторінки для звичайного користувача. Для адміністратора вони виглядають загалом однаково, але є деякі відмінності. Наприклад на сторінках курси та категорії з’являються додаткові кнопки для додавання, видалення та редагування курсів чи категорій. Це зображено на рисунку 3.19.

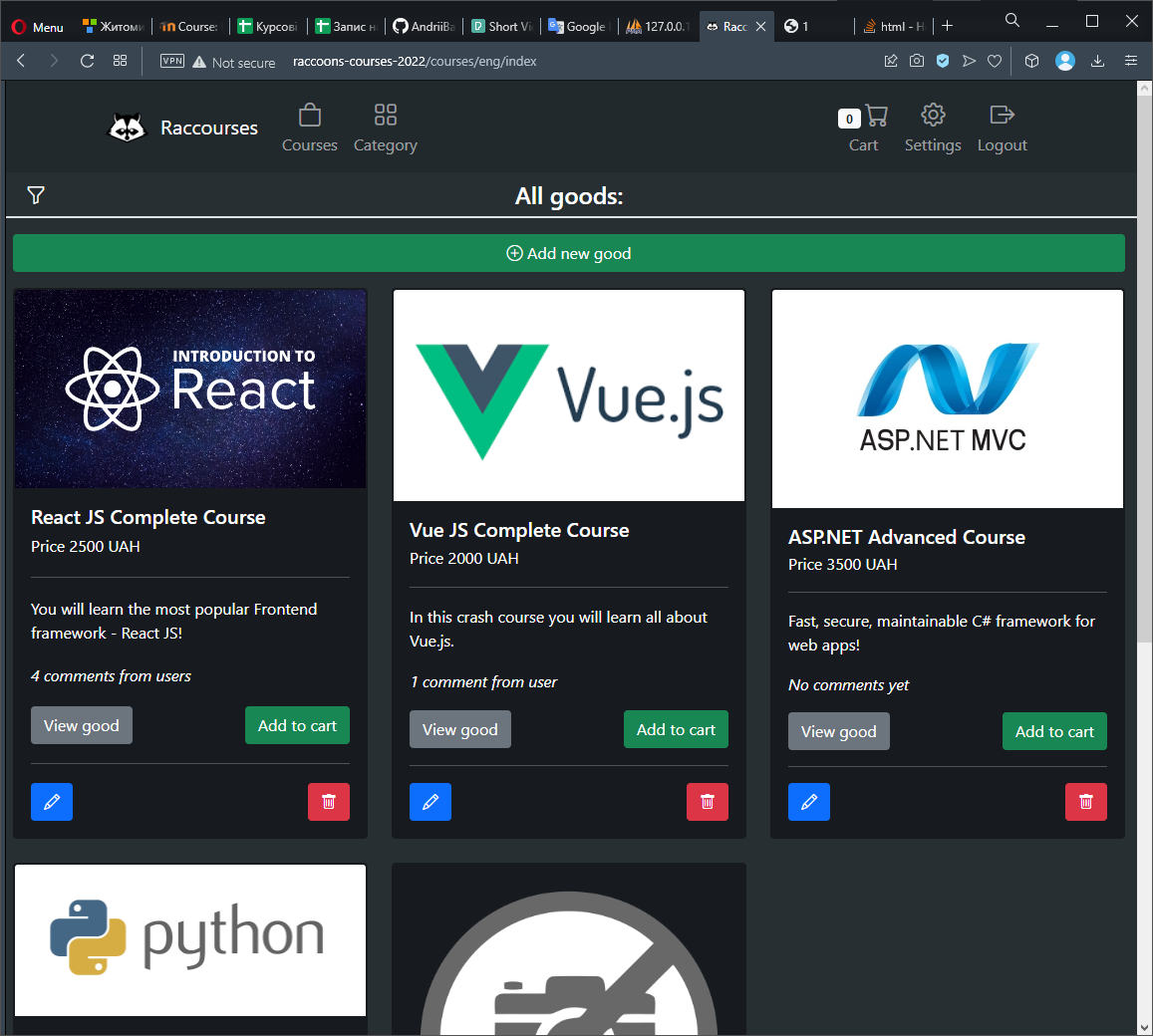


Рис. 3.19. Вигляд сторінки з курсами для адміністратора

Як зазначалося раніше, адміністратор має змогу редагувати будь-який товар, змінювати його дані чи видаляти. Кожна CRUD операція має свою сторінку для відправки даних для додавання чи зміни і підтвердження видалення адміністратором даного товару. Одну з таких сторінок зображено на рисунку 3.20.

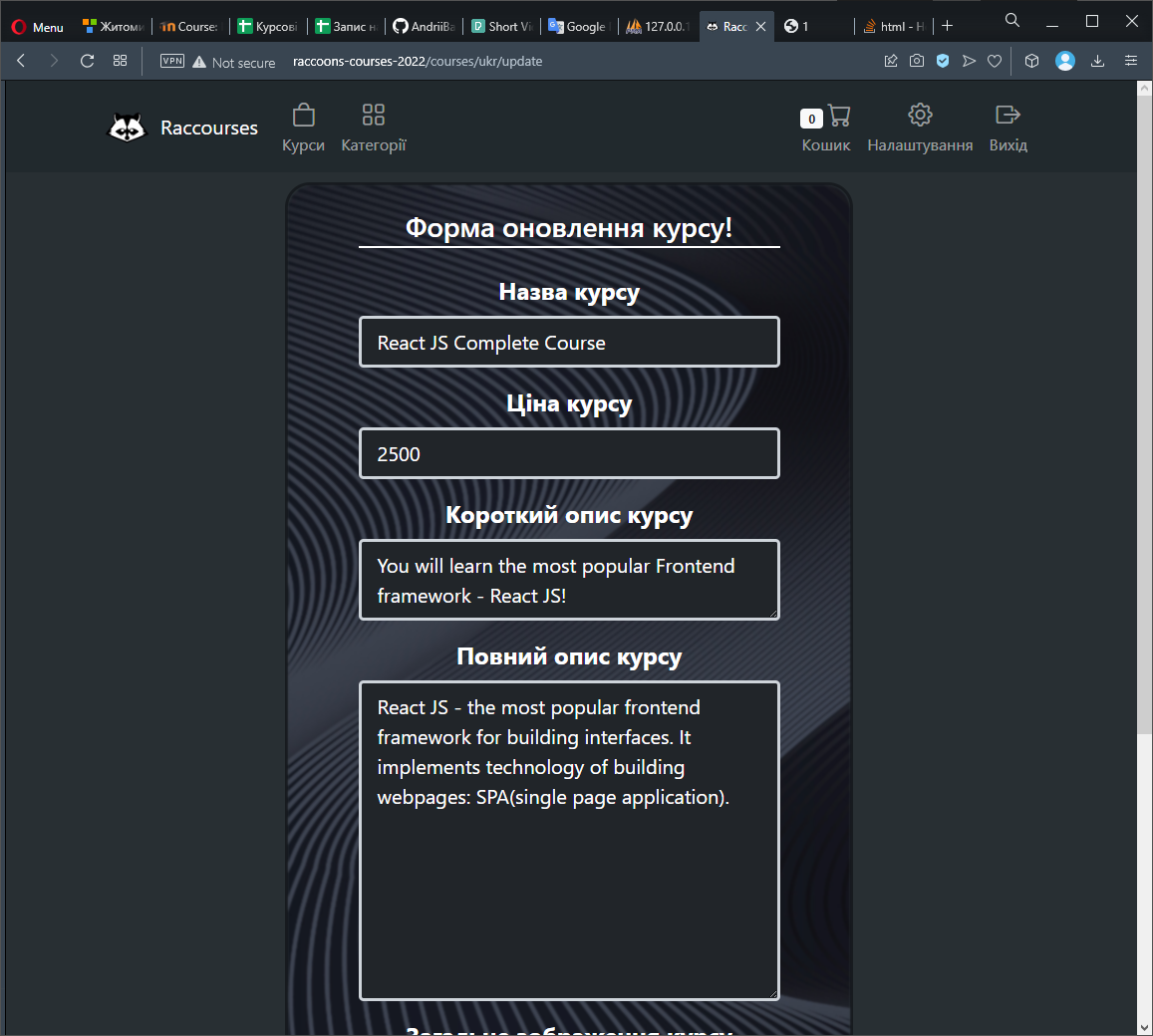


Рис. 3.20. Вигляд сторінки адміністратора для зміни даних

## 3.2 Тестування роботи програмного забезпечення

Тестування даного ПЗ проводилось неодноразово та виконувалось майже кожен день після додавання деякої функції або методу для того, щоб виявити помилки та дефекти, з метою оцінки та удосконалення програми.

Тестування ПЗ проводилось в декілька етапів:

1. Зайти на сайт та переконатися, що кожен елемент завантажується відносно швидко та відображається так, як було задумано.
2. Перевірка чи усі дані було збережено у БД, відправлено та отримано у потрібному вигляді.
3. Тестування роботи усіх посилань, переходів між сторінками.
4. Збереження даних після закриття та виходу з сайту.

На першому етапі тестування було перевірено зручність використання та інтуїтивну зрозумілість програми, перевірено основний функціонал сайту та правильну передачу відповідних даних на клієнт та назад на сервер.

На другому етапі було перевірено наявність багів та помилок, які може видати програма під час роботи, некоректність введення, передачу, перевірку, коректне відображення даних. Також було усунено усі можливі виключення та перебої у програмі.

На останньому етапі було перевірено відповідність усім вимогам до поставленої задачі та удосконалено інтерфейс програми.

## Висновки до третього розділу

В цьому розділі було розглянуто роботу програми, зроблено повне її тестування, перевірено коректність взаємодії усіх введених даних та даних класів, розглянуто основні можливості ПЗ, ознайомлено з основними виключеннями та похибками. Помилки у роботі програми можуть виникнути у разі неправильного використання програми, або за непередбачуваних обставин.

# **ВИСНОВКИ**

Під час виконання поставленого завдання на курсову роботу були отримані практичні навички з написання власної CMS системи за допомогою шаблону MVC на мові програмування PHP і продемонстровано ці навички при написанні реального сайту, який працює без нарікань та проблем.

В першому розділі було проаналізовано поставлену задачу саме які дії потрібно виконати для повної розробки програми, а також конкурентів зі схожим функціоналом та інтерфейсом. Стало зрозуміло основні потреби при розробці програми та поставлено конкретні цілі на проект.

В другому розділі було описано логічну роботу програми, її потреби та проілюстровано за допомогою блок-схеми, діаграм класів та діаграми відносин БД. Цей етап роботи та розробки програмного забезпечення є одним з найважливіших, оскільки саме від проектування та логічної реалізації всіх функцій програмного додатку залежить надійність його роботи.

В третьому розділі було описано повну роботу готової програми, зроблено повне тестування та виправлено всі її недоліки.

Підсумовуючи вище сказане, можем сказати, що ця робота дала нам змогу закріпити раніше вивчений матеріал, написавши для цього реальний веб-сайт. Тобто, задача на курсову роботу була успішно виконана та досягнута!

# **СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ**

ЛІТЕРАТУРА

1. Редактор блок-схем - …
2. Редактор для діаграм - <https://excalidraw.com>
3. А.Зандстра, Ф.Матт. PHP Objects, Patterns, and Practice, 2013. Режим доступу: <https://www.apress.com/la/book/9781430260325>
4. admin. MS Access и C# [Електронний ресурс] / admin. – 2018. – Режим доступу до ресурсу: <https://vscode.ru/prog-lessons/ms-access-i-c-sharp-rabotaem-sbd.html>
5. admin. Все операции с БД в графическом приложении [Електронний ресурс] / admin. – 2015. – Режим доступу до ресурсу: <https://metanit.com/sharp/adonet/3.5.php>
6. Дмитрий Кирсанов. “Веб-дизайн” [електронний ресурс] - 2006. Режим доступу: <http://web-diz.com.ua/skachat/veb-dizayn--dmitriy-kirsanovskachatbesplatno/>
7. Тим Кедлек. “Адаптивный дизайн. Делаем сайты для любых устройств” [електронний ресурс] - 2013. Режим доступу: <https://www.ozon.ru/context/detail/id/21098804/>
8. Документація бібліотеки JQuery [Електронний ресурс] // JQuery incorp. – 2016. – Режим доступ: <https://jquery-docs.ru>
9. Выразительный JavaScript [електронний ресурс] - 2017. Режим доступу: <https://legacy.gitbook.com/book/karmazzin/eloquentjavascript_ru/details>
10. Основы наследования [електронний ресурс] - 2015. Режим доступу: <https://professorweb.ru/my/csharp/charp_theory/level7/7_1.php>
11. Плюси і мінуси об’єктно-орієнтованого програмування [електронний ресурс] - 2015. Режим доступу: <http://damp.biz/plyusi-i-minusiobyektnooriyentovanogo-programuvannya/>
12. SQL [електронний ресурс] - 2016. Режим доступу: https://uk.wikibooks.org/wiki/SQL 11. Создание SQL запросов [електронний ресурс] - 2016. Режим доступу: <https://www.politerm.com/zuludoc/spatial_query_sql.htm#_sql_overview>

# **ДОДАТКИ**

1. **core/Core.php:**

<?php  
  
namespace core;  
  
use controllers\SiteController;  
use models\Error;  
  
class Core  
{  
 private static $instance = null;  
 public array $app;  
 public DB $db;  
 public string $requestMethod;  
  
 private function \_\_construct()  
 {  
 $this->app = [];  
 }  
  
 public static function getInstance(): ?Core  
 {  
 if (empty(self::$instance)) {  
 self::$instance = new self();  
 }  
  
 return self::$instance;  
 }  
  
 public function initialize(): void  
 {  
 session\_start();  
 $this->db = new DB(*DATABASE\_HOST*, *DATABASE\_LOGIN*, *DATABASE\_PASSWORD*, *DATABASE\_BASENAME*);  
 $this->requestMethod = $\_SERVER['REQUEST\_METHOD'];  
 }  
  
 public function run(): void  
 {  
 $routeParts = [];  
  
 if (isset($\_GET['route'])) {  
 $route = $\_GET['route'];  
 $routeParts = explode('/', $route);  
 if (count($routeParts) >= 3) {  
 $moduleName = strtolower(array\_shift($routeParts));  
 $language = strtolower(array\_shift($routeParts));  
 $actionName = strtolower(array\_shift($routeParts));  
 }  
 }  
  
 if (empty($moduleName) || empty($actionName) || empty($language)) {  
 $moduleName = 'site';  
 $language = 'eng';  
 $actionName = 'index';  
 }  
  
 if (!empty($\_GET['language']))  
 if ($\_GET['language'] == 'eng' || $\_GET['language'] == 'ukr')  
 $language = $\_GET['language'];  
  
  
 $this->app['moduleName'] = $moduleName;  
 $this->app['actionName'] = $actionName;  
 $this->app['language'] = $language;  
 if (empty($\_SESSION['theme']))  
 $this->app['theme'] = 'light';  
 else  
 $this->app['theme'] = $\_SESSION['theme'];  
  
 if (!empty($\_GET['theme']))  
 if ($\_GET['theme'] == 'light' || $\_GET['theme'] == 'dark') {  
 $\_SESSION['theme'] = $\_GET['theme'];  
 $this->app['theme'] = $\_GET['theme'];  
 }  
  
 $controllerName = '\\controllers\\' . ucfirst($moduleName) . 'Controller';  
 $controllerActionName = $actionName . 'Action';  
  
 $statusCode = 200;  
 if (class\_exists($controllerName)) {  
 $controller = new $controllerName();  
 if (method\_exists($controller, $controllerActionName)) {  
 $actionResult = $controller->$controllerActionName($routeParts);  
  
 if ($actionResult instanceof Error)  
 $statusCode = $actionResult->code;  
  
 $this->app['actionResult'] = $actionResult;  
 $this->app['pageTitle'] = 'Raccoons Courses - ' . ucfirst($moduleName);  
 } else {  
 $statusCode = 404;  
 }  
 } else {  
 $statusCode = 404;  
 }  
  
 $statusCodeType = intval($statusCode / 100);  
 if ($statusCodeType == 4 || $statusCodeType == 5) {  
 $this->app['moduleName'] = 'site';  
 $this->app['actionName'] = 'error';  
 $siteController = new SiteController();  
 $this->app['actionResult'] = $siteController->errorAction($statusCode);  
 $this->app['pageTitle'] = 'Raccoons Courses - Error';  
 }  
 }  
  
 public function done(): void  
 {  
 $pathToLayout = "themes/{$this->app['theme']}/layout.php";  
  
 $templateMaker = new TemplateMaker($pathToLayout);  
 $templateMaker->setParam('content', $this->app['actionResult']);  
 $templateMaker->setParam('language', $this->app['language']);  
 $templateMaker->setParam('theme', $this->app['theme']);  
 $templateMaker->setParam('title', $this->app['pageTitle']);  
 $html = $templateMaker->getHTML();  
  
 echo $html;  
 }  
}

1. **core/Controller.php:**

<?php  
  
namespace core;  
  
use JetBrains\PhpStorm\NoReturn;  
use models\Error;  
  
class Controller  
{  
 protected string $viewPath;  
 protected string $moduleName;  
 protected string $actionName;  
 protected string $language;  
 protected string $theme;  
  
 public function \_\_construct()  
 {  
 $this->moduleName = \core\Core::getInstance()->app['moduleName'];  
 $this->actionName = \core\Core::getInstance()->app['actionName'];  
 $this->language = \core\Core::getInstance()->app['language'];  
 $this->theme = \core\Core::getInstance()->app['theme'];  
 $this->viewPath = "views/{$this->theme}/{$this->moduleName}/{$this->language}/{$this->actionName}.php";  
 }  
  
 public function render(string $viewPath = null, array $params = null): bool|string  
 {  
 if (empty($viewPath))  
 $viewPath = $this->viewPath;  
  
 $templateMaker = new TemplateMaker($viewPath);  
  
 if (!empty($params)) {  
 $templateMaker->setParams($params);  
 }  
  
 return $templateMaker->getHTML();  
 }  
  
 public function renderView(string $viewName, array $params = null): bool|string  
 {  
 $path = "views/{$this->theme}/{$this->moduleName}/{$this->language}/{$viewName}.php";  
  
 $templateMaker = new TemplateMaker($path);  
  
 if (!empty($params)) {  
 $templateMaker->setParams($params);  
 }  
  
 return $templateMaker->getHTML();  
 }  
  
 #[NoReturn] public function redirect(string $url): void  
 {  
 header("Location: $url");  
 die();  
 }  
  
 public function error(int $code, string $message = null): Error  
 {  
 return new Error($code, $message);  
 }  
}

1. **core/DB.php:**

<?php  
  
namespace core;  
  
class DB  
{  
 protected \PDO $pdo;  
  
 public function \_\_construct(string $hostName, string $login, string $password, string $database)  
 {  
 $this->pdo = new \PDO("mysql: host={$hostName}; dbname={$database}", $login, $password);  
 }  
  
 /\*\*  
 \* Method to INSERT one row data into DB  
 \* ***@param*** string $tableName The name of the table  
 \* ***@param*** array $newRow Association array with keys and values to INSERT  
 \* ***@return*** bool Return TRUE if INSERT is successful, otherwise FALSE  
 \*/  
 public function insert(string $tableName, array $newRow): bool  
 {  
 $fieldsParamsArray = array\_keys($newRow);  
 $fieldsParamsString = implode(', ', $fieldsParamsArray);  
  
 $newValuesArray = [];  
 foreach ($newRow as $key => $value) {  
 $newValuesArray [] = ":" . $key;  
 }  
 $newValuesString = implode(', ', $newValuesArray);  
  
 $res = $this->pdo->prepare("INSERT INTO {$tableName} ({$fieldsParamsString}) VALUES ({$newValuesString})");  
  
 return $res->execute($newRow);  
 }  
  
 /\*\*  
 \* Method to READ and GET data from DB  
 \* ***@param*** string $tableName The name of the table  
 \* ***@param*** string|array $fieldsArray The association array of named fields of table to SELECT  
 \* ***@param*** array|null $conditionArray The association array with keys and values to build WHERE close  
 \* ***@return*** bool|array Return all found rows in the table if successful, otherwise FALSE  
 \*/  
 public function select(string $tableName, string|array $fieldsArray = "\*", array|null $conditionArray = null): bool|array  
 {  
 if (is\_string($fieldsArray))  
 $fieldsListString = $fieldsArray;  
  
 if (is\_array($fieldsArray))  
 $fieldsListString = implode(', ', $fieldsArray);  
  
 $whereCloseString = "";  
 if (is\_array($conditionArray)) {  
 $whereCloseString = self::generateWhereClose($conditionArray);  
 }  
  
 $result = $this->pdo->prepare("SELECT {$fieldsListString} FROM {$tableName} {$whereCloseString}");  
 $result->execute($conditionArray);  
  
 return $result->fetchAll(\PDO::*FETCH\_ASSOC*);  
 }  
  
 /\*\*  
 \* Method to UPDATE data in certain table  
 \* ***@param*** string $tableName The name of the table  
 \* ***@param*** array $newValuesArray The association array with keys and values to UPDATE  
 \* ***@param*** array $conditionArray The association array with keys and values to build WHERE close  
 \* ***@return*** bool Return TRUE if successful, otherwise FALSE  
 \*/  
 public function update(string $tableName, array $newValuesArray, array $conditionArray): bool  
 {  
 $paramsArray = [];  
 $setCloses = [];  
 foreach ($newValuesArray as $key => $value) {  
 $setCloses[] = "{$key} = :set{$key}";  
 $paramsArray ["set" . $key] = $value;  
 }  
  
 $setCloseString = "SET " . implode(', ', $setCloses);  
  
 foreach ($conditionArray as $key => $value)  
 $paramsArray [$key] = $value;  
  
 $whereCloseString = self::generateWhereClose($conditionArray);  
  
 $result = $this->pdo->prepare("UPDATE {$tableName} {$setCloseString} {$whereCloseString}");  
 return $result->execute($paramsArray);  
 }  
  
 /\*\*  
 \* Method to DELETE data from certain table by some values  
 \* ***@param*** string $tableName The name of the table  
 \* ***@param*** array $conditionArray The association array with keys and values to build WHERE close  
 \* ***@return*** bool Return TRUE if successful, otherwise FALSE  
 \*/  
 public function delete(string $tableName, array $conditionArray): bool  
 {  
 $whereCloseString = self::generateWhereClose($conditionArray);  
  
 $result = $this->pdo->prepare("DELETE FROM {$tableName} {$whereCloseString}");  
 return $result->execute($conditionArray);  
 }  
  
 /\*\*  
 \* ***@param*** array $conditionArray The association array with keys and values to build WHERE close  
 \* ***@return*** string Return built WHERE close for DB queries  
 \*/  
 protected function generateWhereClose(array $conditionArray): string  
 {  
 $whereCloses = [];  
  
 foreach ($conditionArray as $key => $value) {  
 $whereCloses [] = "{$key} = :{$key}";  
 }  
  
 return "WHERE " . implode(' AND ', $whereCloses);  
 }  
}

1. **core/TemplateMaker.php:**

<?php  
  
namespace core;  
  
class TemplateMaker  
{  
 protected string $path;  
 protected array $params;  
  
 public function \_\_construct($path)  
 {  
 $this->path = $path;  
 $this->params = [];  
 }  
  
 public function setParam($variableName, $variableValue): void  
 {  
 $this->params[$variableName] = $variableValue;  
 }  
  
 public function setParams($params): void  
 {  
 foreach ($params as $name => $value){  
 $this->setParam($name, $value);  
 }  
 }  
  
 public function getHTML(): bool|string  
 {  
 ob\_start();  
  
 extract($this->params);  
 include($this->path);  
 $html = ob\_get\_contents();  
  
 ob\_end\_clean();  
  
 return $html;  
 }  
}

1. **core/Utils.php:**

<?php  
  
namespace core;  
  
class Utils  
{  
 public static function filterArray(array $array, array $fieldList): array  
 {  
 $newArray = [];  
 foreach ($array as $key => $value) {  
 if (in\_array($key, $fieldList))  
 $newArray [$key] = $value;  
 }  
  
 return $newArray;  
 }  
  
 public static function ifLanguageEqual(string $language): bool  
 {  
 return \core\Core::getInstance()->app['language'] === $language;  
 }  
  
 public static function generateMessage(string $messageName, array $messagesTextList): array  
 {  
 $errors = [];  
  
 foreach ($messagesTextList as $key => $value)  
 if (Utils::ifLanguageEqual($key))  
 $errors[$messageName] = $value;  
  
 return $errors;  
 }  
  
 public static function checkImgExtension(string $imgName): bool  
 {  
 $explodeArray = explode('.', $imgName);  
 $extension = end($explodeArray);  
 if ($extension === 'png' || $extension === 'jpg')  
 return true;  
  
 return false;  
 }  
  
 public static function checkVideoExtension(string $videoName): bool  
 {  
 $explodeArray = explode('.', $videoName);  
 $extension = end($explodeArray);  
 if ($extension === 'mp4')  
 return true;  
  
 return false;  
 }  
}