МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»

Кафедра інженерії програмного забезпечення

**КУРСОВА РОБОТА**

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

з дисципліни: «Веб-технології. Частина 2»

на тему:

**«Розробка порталу для завантаження та прослуховування музики»**

студента ІІ курсу групи ІПЗ-20-4

спеціальності 121 «Інженерія програмного забезпечення»

\_Довгалюка Владислава Олександровича\_

(прізвище, ім’я та по-батькові)

Керівник Морозов А.В.

Дата захисту: " \_\_\_ " \_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_ р.

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

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

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

Члени комісії \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Граф М.С. \_

(підпис) (прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Кузьменко О.В. .

(підпис) (прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Чижмотря О.Г. \_

(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»

В.о.зав. кафедри ІПЗ

\_\_\_\_\_\_\_\_\_\_\_\_ А. В. Морозов.

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

ЗАВДАННЯ

НА КУРСОВИЙ ПРОЕКТ СТУДЕНТУ

Довгалюку Владиславу Олександровичу

1. Тема роботи: Розробка порталу для завантаження та прослуховування музики,

керівник роботи: Морозов А.В.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Строк подання студентом: “ 12 ” січня 2022р.
2. Вихідні дані до роботи: Розробити портал для завантаження та прослуховування музики, розробити та реалізувати систему адміністрування для сайту.
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. Презентація до КП

2. Посилання на репозиторій: https://gitlab.com/2020-2024/ipz-20-4/dovgalyuk-vladislav/cursova\_2\_curse

1. Консультанти розділів проекту (роботи)

|  |  |  |  |
| --- | --- | --- | --- |
| Розділ | Прізвище, ініціали та посади консультанта | Підпис, дата | |
| завдання  видав | завдання прийняв |
| 1,2 | Чижмотря О.Г., ст. викладач каф. ІПЗ |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

1. Дата видачі завдання “\_15\_”\_\_ вересня\_\_2021 р.

|  |  |  |  |
| --- | --- | --- | --- |
| № з/п | Назва етапів курсової роботи | Строк виконання етапів роботи | Примітки |
| 1 | Постановка задачі | 20.12.2021-21.12.2021 | Виконав |
| 2 | Пошук, огляд та аналіз аналогічних розробок | 25.12.2021-27.12.2021 | Виконав |
| 3 | Формулювання технічного завдання | 27.12.2021-27.12.2021 | Виконав |
| 4 | Опрацювання літературних джерел | 03.01.2022-03.01.2022 | Виконав |
| 5 | Проектування структури | 03.01.2022-04.01.2022 | Виконав |
| 6 | Написання програмного коду | 04.01.2022-13.01.2022 | Виконав |
| 7 | Налагодження | 13.01.2022-17.01.2022 | Виконав |
| 8 | Написання пояснювальної записки | 17.01.2022-18.01.2022 | Виконав |
| 9 | Захист | 19.01.2022 |  |

**КАЛЕНДАРНИЙ ПЛАН**

**Студент** \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Довгалюк В.О

(підпис) (прізвище та ініціали)

**Керівник роботи** \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Морозов А.В.

(підпис) (прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до курсового проекту на тему «Розробка порталу для завантаження та прослуховування музики » складається з переліку умовних скорочень, вступу, трьох розділів, висновків, списку використаної літератури та додатку.

Текстова частина викладена на 36 сторінках друкованого тексту.

Пояснювальна записка має 5 сторінок додатків. Список використаних джерел містить 5 найменувань і займає 1 сторінку. В роботі наведено 18 рисунків. Загальний обсяг роботи – 40 сторінок.

У першому розділі було обґрунтовано створення сайту на тему «Портал для прослуховування та завантаження музики»

У другому розділі проведено проектування і розробка програмного продукту.

У третьому розділі проведено тестування програмного продукту.

Висновок містить в собі результати виконаної роботи при створенні веб-додатку на тему «Портал для прослуховування та завантаження музики»

У додатку представлений лістинг розробленого програмного продукту.

Ключові слова: MVC, JAVASCRIPT, PHP, MYSQL, INNODB, веб-сайт, веб-додаток, HTML, CSS, МУЗИКА, МУЗИЧНИЙ ПОРТАЛ, БАЗИ ДАНИХ, PLAYERJS

ЗМІСТ

[ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ 6](#_Toc93484286)

[**Вступ** 7](#_Toc93484287)

[**РОЗДІЛ 1 АНАЛІЗ ПРОБЛЕМАТИКИ, МЕТОДІВ ТА ЗАСОБІВ ВИРІШЕННЯ ЗАДАЧІ** 8](#_Toc93484288)

[**1.1 Аналіз задачі, засобів та методів її вирішення** 8](#_Toc93484289)

[**1.2 Аналіз існуючого програмного забезпечення за тематикою курсової роботи.** 9](#_Toc93484290)

[**1.3 Технічне завдання на курсову роботу** 11](#_Toc93484291)

[**1. Загальне положення** 11](#_Toc93484292)

[**2. Підстава для розробки** 11](#_Toc93484293)

[**3. Вимоги до програми** 12](#_Toc93484294)

[**Висновки до першого розділу :** 14](#_Toc93484295)

[**РОЗДІЛ 2 ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ** 15](#_Toc93484296)

[**2.1 Проектування загального алгоритму роботи веб-додатку** 15](#_Toc93484297)

[**2.2 Розробка функціональних алгоритмів роботи програми** 17](#_Toc93484298)

[**2.3 Розробка програмного забезпечення** 19](#_Toc93484299)

[**Висновки до 2 розділу :** 23](#_Toc93484300)

[**РОЗДІЛ 3 ОПИС РОБОТИ З ДОДАТКОМ** 24](#_Toc93484301)

[**3.1 Опис роботи з додатком (Опис інтерфейсу)** 24](#_Toc93484302)

[**Висновки до 3 розділу :** 29](#_Toc93484303)

[**ВИСНОВКИ** 30](#_Toc93484304)

[**СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ** 31](#_Toc93484305)

[**ДОДАТКИ** 32](#_Toc93484306)

# ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

БД – База даних

КР – курсова робота

MVC – Модель–вигляд–контролер (або Модель–представлення–контролер, Model-view-controller, MVC) — архітектурний шаблон, який використовується під час проектування та розробки програмного забезпечення.

ПЗ – Програмне забезпечення

# **Вступ**

У даній курсовій роботі буде наведено процес розробки порталу для

прослуховування та завантаження музики.

Актуальність даної роботи полягає в тому, щоб пересічна людина змогла опублікувати свою пісню, пісню іншої людини або групи, прослухати її на сайті, і за потреби завантажити на свій носій.

Метою курсової роботи є дослідження особливостей проектування та реалізації cистеми адміністрування порталу для прослуховування та завантаження музики.

Завданням на курсову роботу є:

* аналіз теоретичних засад проектування та реалізації CMS;
* аналіз існуючих веб-сайтів на обрану вами тематику, визначення унікальності;
* визначення інформаційних потреб предметної області дослідження;
* розробка адаптивного інтерфейсу веб-сайту за допомогою HTML5, CSS3, JavaScript;
* розробка математичної та алгоритмічної моделі функціонування системи на основі БД;
* проектування бази даних за визначеною предметною областю;
* написання серверної частини;
* реалізація CMS.

Об’єктом дослідження є методи та засоби розробки CMS для порталу для прослуховування та завантаження музики.

Предметом дослідження є використання веб-технологій для забезпечення культурних потреб в сферах життя та розваг.

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

## **1.1 Аналіз задачі, засобів та методів її вирішення**

Задача полягає в тому, що потрібно:

1. Визначити потреби звичайного користувача
2. Зрозуміти основні принципи та алгоритми роботи порталу для прослуховування та завантаження музики
3. Продумати можливість додавання та редагування пісень, груп та користувачів
4. Реалізувати можливість прослуховування пісень на сайті
5. Реалізувати можливість пошуку пісень та груп
6. Продумати можливість створення нових адміністраторів та редакторів
7. Створити БД
8. Продумати дизайн сайту
9. Реалізувати приємний дизайн інтерфейсу для користувачів

Для розробки сайту буде використана мова програмування PHP для серверної частини та PHP + JS для клієнтської частини.

Для КР використано текстовий редактор Visual Studio Code (для створення клієнтської частини сайту) та середовище розробки PHPStorm (для створення северної частини сайту).

У якості БД використано MySQL.

## **1.2 Аналіз існуючого програмного забезпечення за тематикою курсової роботи.**

При аналізі вже існуючих веб-сайтів за тематикою КР було виявлено багато подібних проєктів, але вибрано два перших що відобразились при пошуку. Всі вони мають подібну структуру та реалізацією.

Аналоги порталів для прослуховування та завантаження музики:

Приклад 1:

z2.fm, посилання на сайт - <https://z2.fm/>

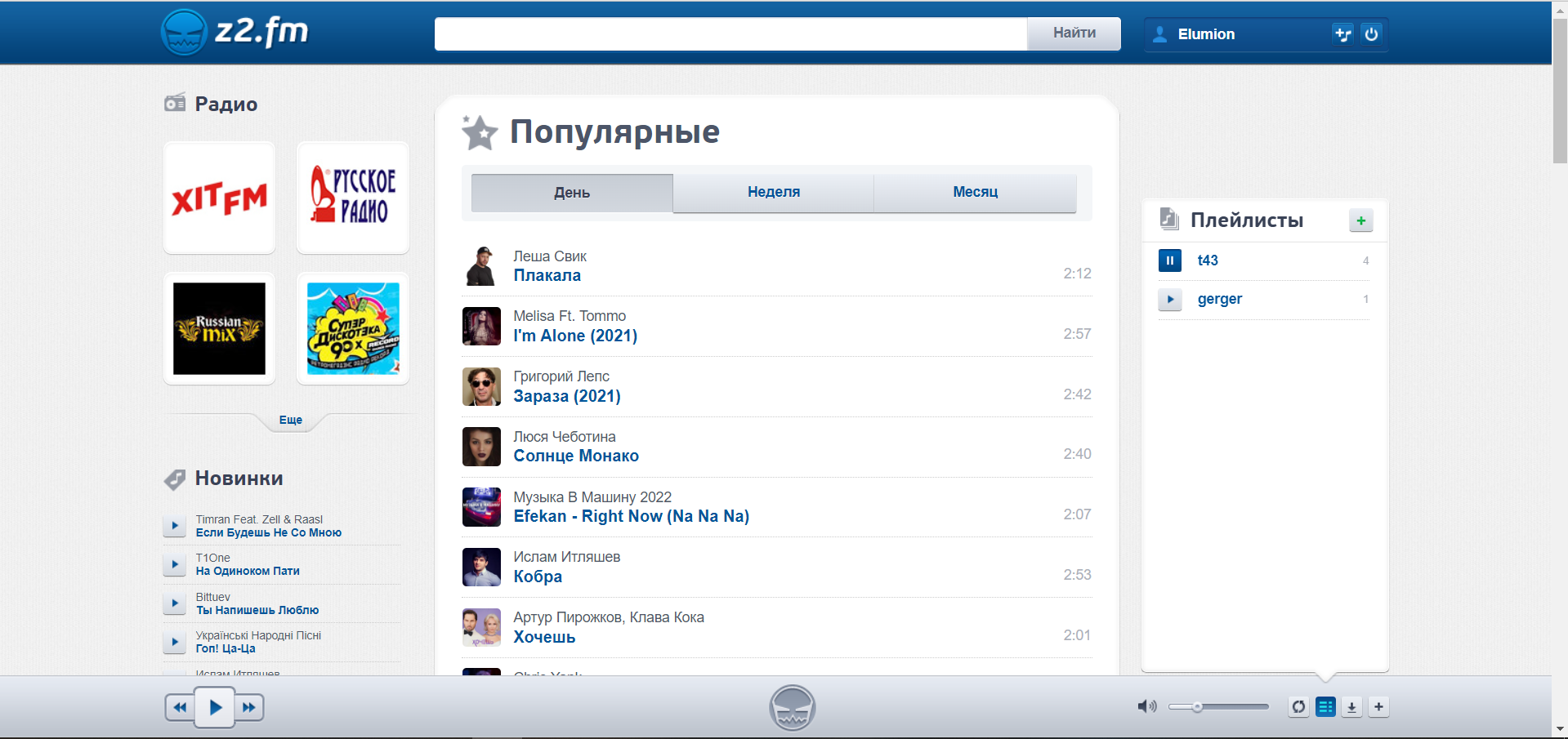


Рис 1.1 z2.fm

Портал для музики z2.fm. Сайт пропонує безкоштовне завантажування музики з сайту і на сайт.

Основні плюси:

1. Прослуховування музики
2. Створення плейлиста
3. Завантаження музики
4. Зручний плеєр
5. Зрозумілий дизайн

Приклад 2:

Sefon, посилання на ресурс - <https://sefon.pro/>

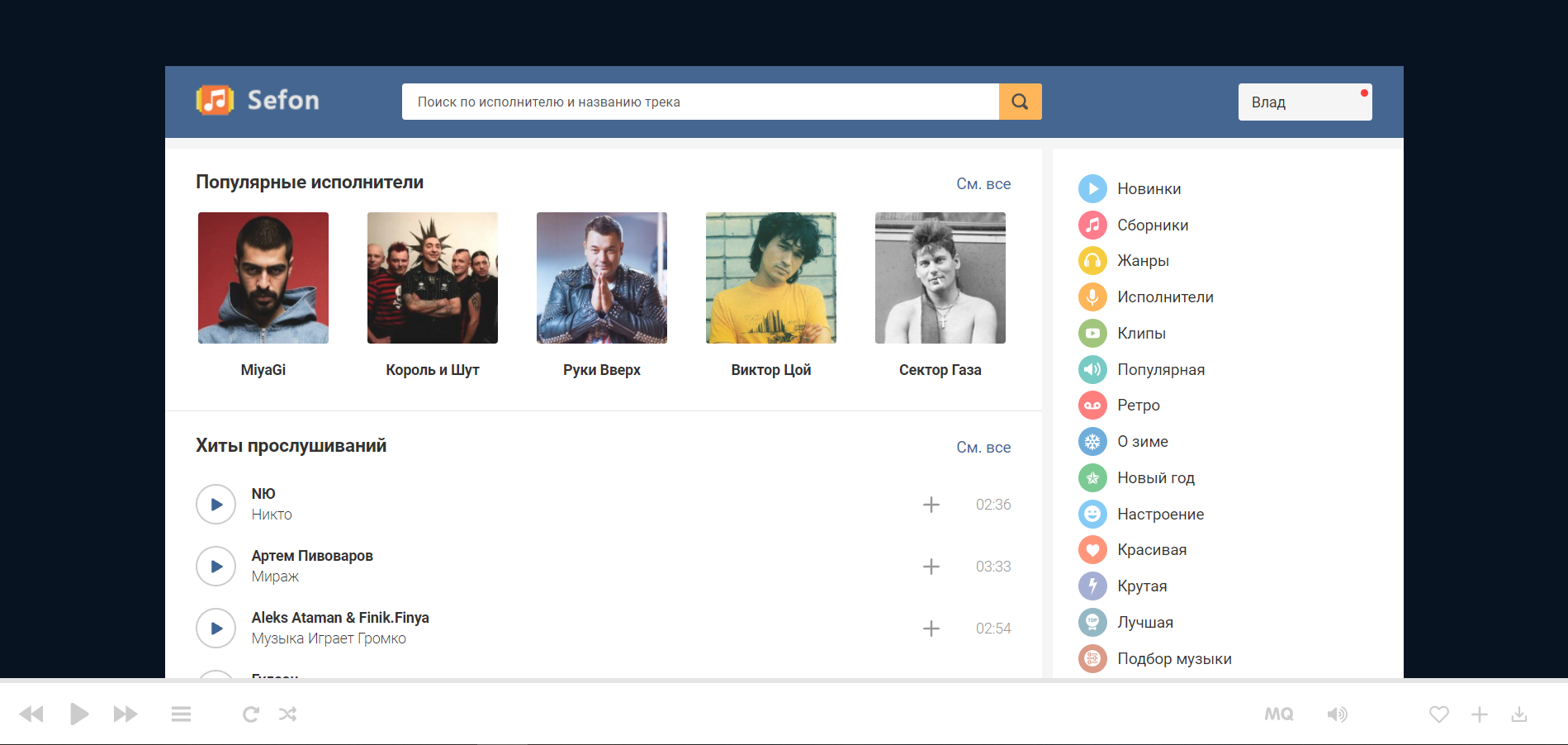


Рис 1.2 Sefon

Музичний портал що дозволяє завантажувати музику на носій. З недоліків неможливість звичайному користувачу завантажувати музику на сайт.

Основні плюси:

1. Наявність різних категорії музики
2. Зручний плеєр
3. Можливість створювати свій плейлист

Актуальним напрямком реалізації власного порталу для прослуховування та завантаження музики є веб-сайт з подібно реалізованою логікою та інтерфейсом веб-сайту. Також реалізувати сучасний дизайн сайту.

## **1.3 Технічне завдання на курсову роботу**

### **1. Загальне положення**

#### **1.1 Найменування програмного засобу**

Повне найменування програмної системи: "Розробка порталу для прослуховування та завантаження музики" (надалі "веб-додаток"). Коротка назва програмної системи - "OnMuzik"

#### **1.2. Призначення розробки та область застосування**

Веб-сайт «OnMuzik» призначений для публікації музичних творів, завантаження її, пошуку та прослуховування.

Даний веб-додаток дозволить прослуховувати музику, завантажувати пісні на сайт, шукати інформацію про виконавця пісні, долзволить редакторам змінювати інформацію про виконавців і адміністраторам здійснювати контроль над сайтом.

#### **1.3. Найменування розробника та замовника.**

Розробник даного продукту - студент групи ІПЗ-20-4 Довгалюк Владислав Олександрович (надалі "розробник"). Замовник програмного продукту – кафедра інженерія програмного забезпечення Житомирського державного технологічного університету в межах виконання курсової з дисципліни «Web-тееології» Чижмотря Олена Геннадіївна, Морозов Андрій Васильович.

### **2. Підстава для розробки**

#### **2.1. Документ на підставі якого ведеться розробка**

Робота ведеться на підставі навчального плану за напрямом 121 «Інженерія програмного забезпечення».

### **3. Вимоги до програми**

#### **3.1. Вимоги до функціональних характеристик.**

#### **3.1.1. Загальні вимоги**

Веб-додаток має забезпечувати:

* Постійний доступ користувачів до веб-додатку
* Завантаження пісні
* Прослуховування пісні
* Організацію управління сайтом
* Можливість доступу до бази даних
* Можливість редагування даних про виконавців
* Можливість редагування користувача
* Можливість редагування пісні

#### **3.1.2. Склад виконуваних функцій**

Розробити портал для прослуховування та завантаження музики, що підтримує виконання наступних операцій:

1. Прослуховування пісні
2. Завантаження пісні на сайт з вказанням наступних даних:

* Назва пісні
* Картинка
* Виконавець
* Файл пісні

1. Завантаження пісні з сайту на носій
2. Редагування пісні
3. Редагування виконавців
4. Пошук пісень та груп за назвою
5. Перегляд виконавців
6. Редагування акаунта користувача
7. Створення акаунта та авторизація

#### **3.1.3. Організація вхідних і вихідних даних**

Вхідними даними є інформація про пісню (Назва, виконавець, картинка, файл пісні), інформація про виконавця (назва, опис, картинка), інформація про користувача (Логін, пошта, пароль, картинка).

Вихідними даними може бути файл музики що завантажений з сайту

#### **3.1.4. Часові характеристики і розмір пам'яті, необхідної для роботи програми.**

* Час реакції програми на дії користувача (маніпуляції з пристроями введення даних) не повинен перевищувати 0,25 с.
* Час виконання команд меню не більше 1 с.
* Відображення масивів даних за запитами не більше 3 хвилин.
* Доступність БД – 90% цілодобово.
* Операції з’єднання з БД не більше 1 хвилини.
* Обсяг оперативної пам'яті, необхідний для роботи програми не менше 1Гб.
* Дисковій простір, необхідний для збереження програми і файлів даних не більше 300 Мбат для робочої станції та 20 ГБайт.
* Інсталяційний пакет програми, що містить у складі БД не повинні перевищувати 100 Мбайт.

#### **3.2. Вимоги до методів рішення і мов програмування**

Вибір методів рішення здійснюється розробникам без узгодження з замовником.

#### **3.2.1. Вимоги до системи програмних засобів.**

OpenServer, PHP 5, MySQL, HTML 5, CSS 3, JavaScript:

Вимоги до програмного забезпечення робочої станції:

PHP Storm

## **Висновки до першого розділу :**

У ході виконання першого розділу було поставлено завдання КР, а саме який має бути функціонал і визначено актуальність теми. Також визначено мови і фреймворки для розробки та актуальний напрямок розробки власного продукту. Розглянуто існуючі Web-сайти зі схожою тематикою та більш детально проаналізовано.

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

## **2.1 Проектування загального алгоритму роботи веб-додатку**

На даному етапі розробки розглянемо загальний алгоритм роботи інтернет-магазину.

Загальна схема роботи веб-додатку наведена на рисунку 2.1.

Для початку роботи потрібно продумати базу даних і таблиці та поля в ній.

Після цього потрібно продумати ядро системи, реалізувати зручні методи взаємодії з ним та звертання до бази даних.

Для авторизації та реєстрації створимо форму і через контроллер будемо будемо передавати дані в моделі, яка в свою чергу буде перевіряти дані на валідацію, а потім передавати результат валідацію в базу даних і якщо потрібно передавати результат на вихід.

Всі сторінки зберігатимуться у папці views і для кожної сторінки буде своя папка у views із відповідною назвою і назвою методу.

При запуску сайту користувач буде перенаправлений на сторінку зі списком пісень, також буде реалізована пагінація, щоб користувач отримував не весь список пісень з бази даних, адже це потребуватиме значних ресурсів для завантаження.

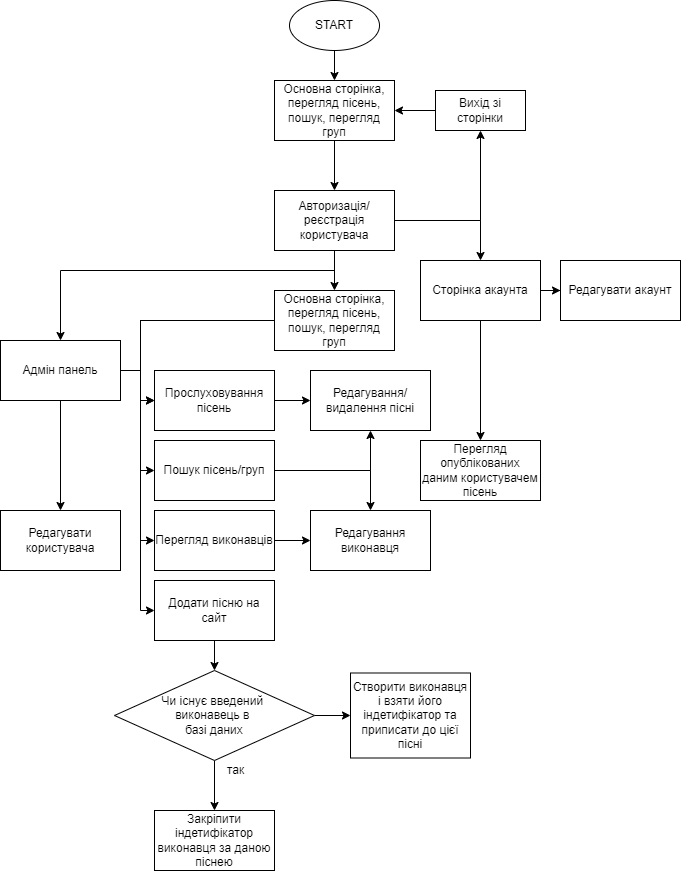


Рис 2.1 Загальна схема роботи сайту

## **2.2 Розробка функціональних алгоритмів роботи програми**

Маючи загальний алгоритм веб-додатку, розглянемо роботу основних алгоритмів.

Основна задача порталу для прослуховування та завантаження музики – це реалізація завантаження та прослуховування музики, тому розглянемо даний алгоритм.

Для прослуховування пісні користувачу, навіть неавторизованому, потрібно перейти на сторінку з піснями і вибрати пісню зі списку. Для завантаження пісні на сайт, користувач повинен авторизуватись, перейти на сторінку добавлення пісні, вказати назву пісні, та завантажити файл. Також потрібно, але необов’язково, вказати виконавця та картинку для пісні.

Після відправки форми, в контроллер методу передається супер-глобальний асоціативний масив з полями що заповнив користувач. Потім контроллер передає даний масив в модель. В моделі відбувається процес валідації, та перевірки даних. Якщо введена група існує в базі даних, тоді вона береться з неї, якщо ж групи не було виявлено в базі даних, вона створюється. Таким чином, щоб створити групу нам потрібно завантажити пісню з вказаним виконавцем. Після проходження всіх валідацій, користувачу виводиться або повідомлення з помилками, або в разі успішного завантаження пісні на сайт, виводиться повідомлення про успішну операцію.

Потім пісня відображатиметься на сайті, і редагувати її зможуть лише користувач що завантажив її на сайт, або ж користувач з рівнем доступу адміністратор.

Редагування виконавців доступні лише користувачам з рівнем доступу адміністратор, або користувачам з рівнем доступу редактор.

В роботі веб-додатку основним алгоритмом є:

Створення запиту – отримання даних на сервері – обробка отриманих даних та взаємодія з БД – генерування HTML сторінки - відображення сторінки користувачу.

## **2.3 Розробка програмного забезпечення**

Структура рішення курсової роботи:

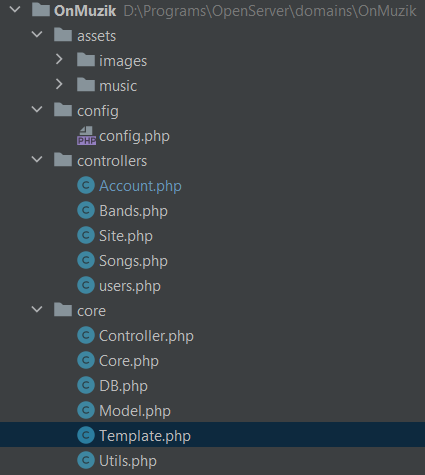


Рис 2.2 Структура веб-додатку

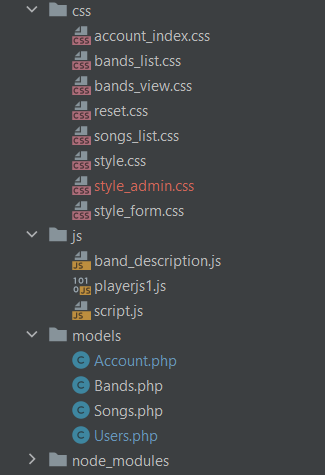


Рис 2.3 Структура веб-додатку

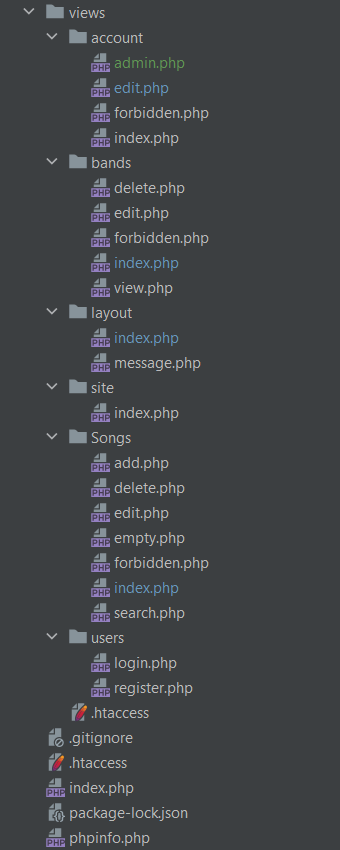


Рис 2.4 Структура веб-додатку

Для прикладу розглянемо контролер додавання пісні на сайт.

*/\*\*  
 \* add song  
 \*/*public function actionAdd()  
{  
 $titleForbidden = 'Доступ заборонено';  
  
 if (empty($this->user))  
 return $this->render('forbidden', null, [  
 'MainTitle' => $titleForbidden,  
 'PageTitle' => $titleForbidden  
 ]);

На початку йде перевірка на те чи авторизований користувач. Якщо користувач неавторизований тоді генерується сторінка з повідомленням про помилку, і метод закінчує роботу. Якщо користувач авторизований тоді відбувається наступна частина коду.

$title = 'Додавання пісні';  
  
if ($this->isPost()) {  
 $allowedTypesImg = ['image/png', 'image/jpeg'];  
 $allowedTypesMedia = ['audio/mpeg'];  
  
  
 $result = $this->songModel->AddSongs($\_POST);

Йде перевірка на те, яким методом відправлена форма, якщо це метод пост, тоді згенерований супер-глобальний асоціативний масив передається у метод моделі добавлення пісні.

*/\*\*  
 \* add song to db  
 \** ***@param*** *$row  
 \** ***@return*** *array|bool  
 \*/*public function AddSongs($row)  
{  
 $userModel = new Users();  
 $user = $userModel->GetCurrentUser();  
 if ($user == null) {  
 $result = [  
 'error' => true,  
 'messages' => ['Користувач не аунтефікований']  
 ];  
 return $result;  
 }  
 $validateResult = $this->Validate($row);  
 if (is\_array($validateResult)) {  
 $result = [  
 'error' => true,  
 'messages' => $validateResult  
 ];  
 return $result;  
 }  
 $fields = ['title', 'media', 'image'];  
 $rowFiltered = Utils::*ArrayFilter*($row, $fields);  
 $rowFiltered['date'] = date('Y-m-d H:i:s');  
 $rowFiltered['user\_id'] = $user['id'];  
  
 $rowFiltered["image"] = 'default\_song.png';  
 $rowFiltered["media"] = '...media...';  
  
 $rowFiltered['band\_id'] = $this->GetBandIDByBandName($row['band\_name']);  
  
 $id = Core::*getInstance*()->getDB()->insert('song', $rowFiltered);  
  
 return [  
 'error' => false,  
 'id' => $id  
 ];  
}

В даній частині код перевіряється на помилки, і при успішній валідації пісня добавляється в базу даних.

return $this->renderMessage('ok', 'Пісню успішно додано', null, [  
 "MainTitle" => $title,  
 "PageTitle" => $title  
  
]);

І при успішному добавлені пісні, користувачу відображається повідомлення про успішний процес добавлення пісні.

$message = implode('<br/>', $result['messages']);  
 return $this->render("add", null, [  
 "MainTitle" => $title,  
 "PageTitle" => $title,  
 'style' => 'style\_form',  
 "MessageClass" => "error",  
 "MessageText" => $message  
 ]);  
}

Якщо ж з’явились якісь помилки при валідації даних, тоді користувачу відображається повідомлення з наявними помилками.

## **Висновки до 2 розділу :**

У ході виконання другого розділу було спроектовано загальну схему програми. Показано структуру КР. Визначено та детально описано основні можливості програми. Пояснено алгоритми найважливіших методів веб-додатку та приведено код їх основних частин. Приведено приклади коду деяких методів. Роз’яснено деякі нюанси роботи програми.

# **РОЗДІЛ 3 ОПИС РОБОТИ З ДОДАТКОМ**

## **3.1 Опис роботи з додатком (Опис інтерфейсу)**

Після запуску додатку появляється головна сторінка з піснями (рис 3.1). Відразу у користувача з’являється можливість прослуховувати завантажені іншими користувачами пісні, а також переглядати сторінку з виконавцями і здійснювати пошук пісень.

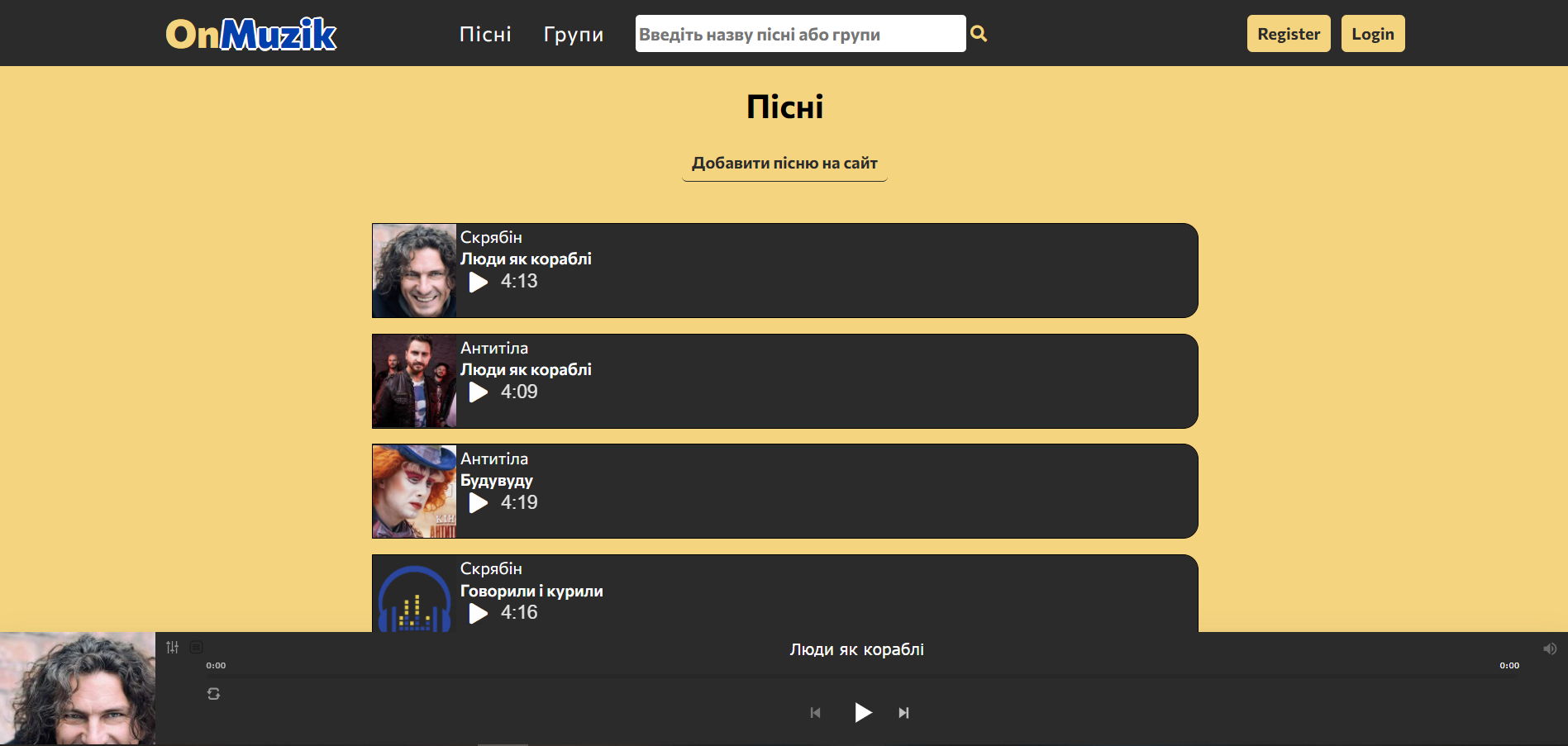


Рис 3.1 головна сторінка

Якщо ми натиснемо на добавлення пісні, то згенерується сторінка з повідомленням про те що користувач неавторизований. Тому для добавлення пісень користувачу потрібно натиснути кнопку реєстрації, або логіну щоб зайти на сайт

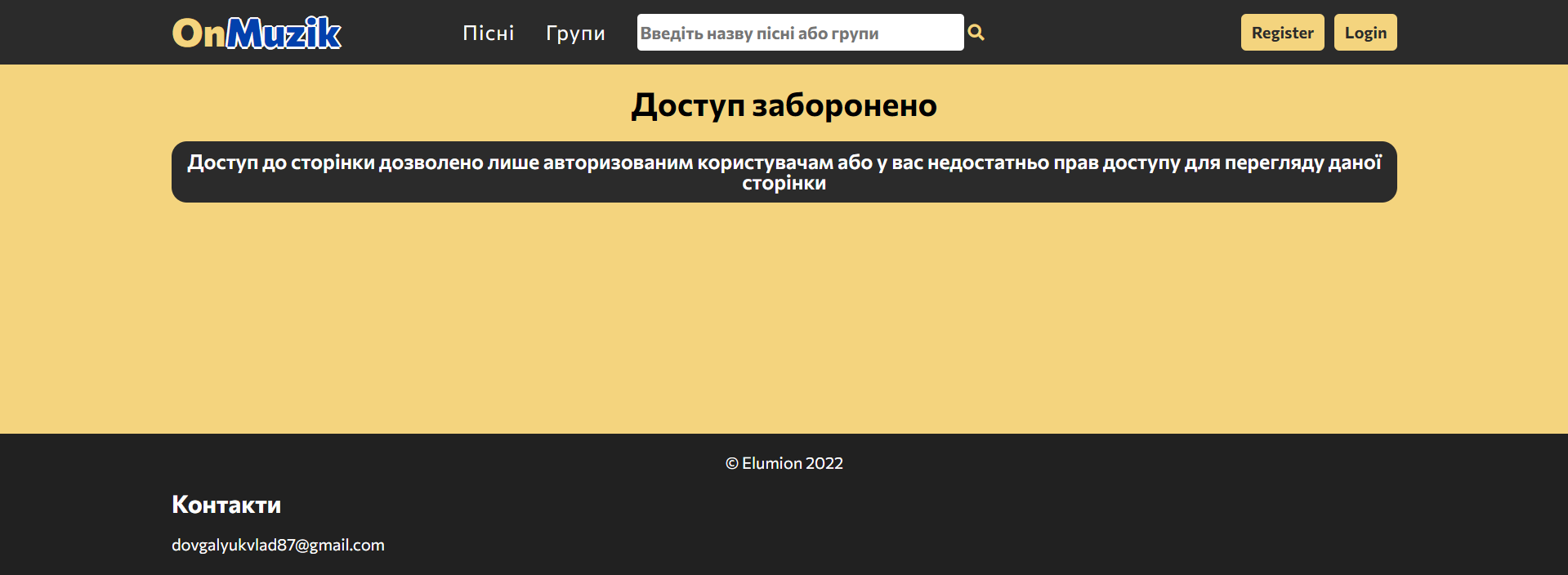


Рис 3.2 доступ заборонено

Для реєстрації користувача (Рис 3.3) потрібно заповнити дану форму. Після відправки, всі дані будуть занесені до бази даних, а пароль спочатку захешується, а потім теж занесеться до БД, таким чином, навіть отримавши доступ до БД не буде змоги отримати пароль користувачів.

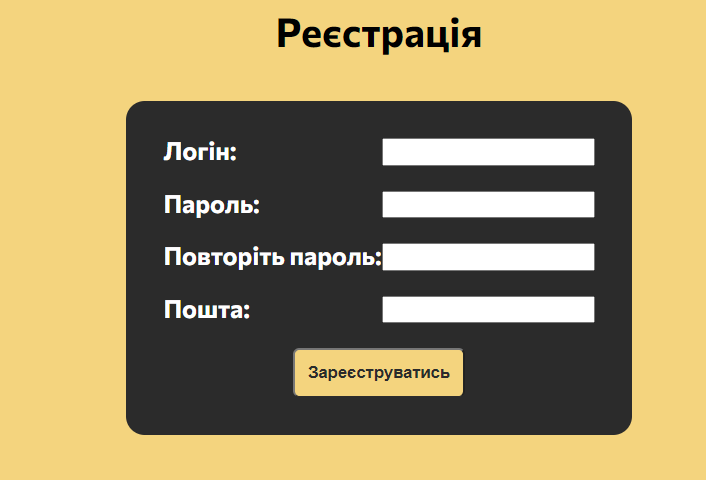


Рис 3.3 Форма реєстрації

Після успішної реєстрації, користувачу необхідно авторизуватись(Рис 3.4)

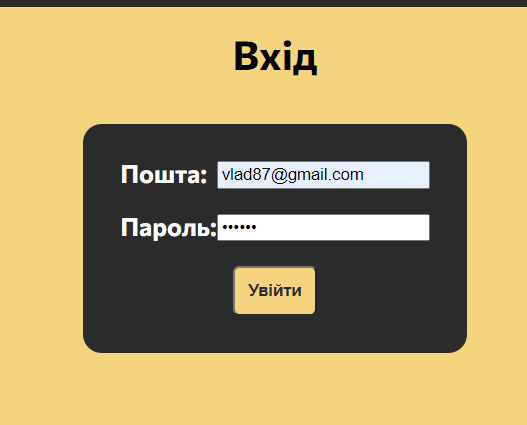


Рис 3.4 Форма авторизації

Після успішної авторизації, користувачу згенерується повідомлення(Рис 3.5) про успішну авторизацію

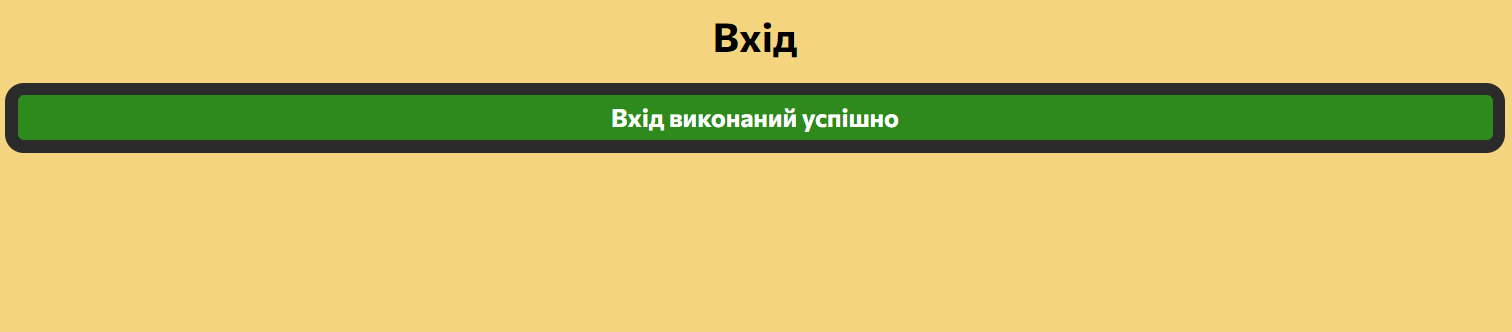


Рис 3.5 Успішна авторизація

Після авторизації, користувач може добавляти пісні та групи на сайт, заповнивши форму (Рис 3.6). Для того щоб створити виконавця, потрібно добавити пісню на сайт і при цьому вказати назву виконавця, якщо виконавець раніше був занесений у БД тоді він і приписується до пісні, якщо ж виконавець в БД не знайдений, то він створюється із заповненим іменем що вказане в полі виконавець, при цьому картинка та опис виконавця будуть порожні, і редактувати виконавця зможе лише адміністратор, або користувач з рівнем доступу редактор.

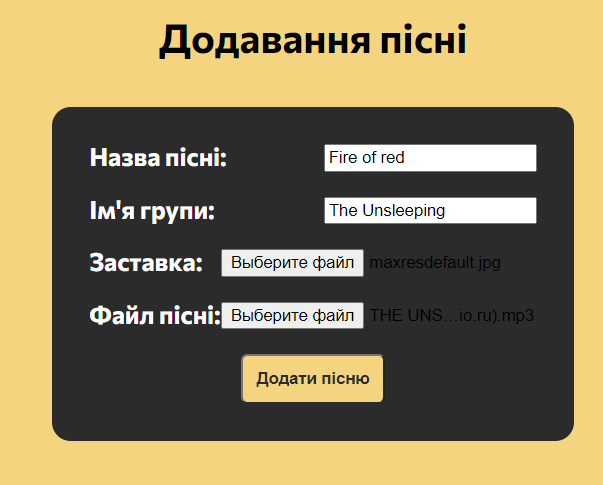


Рис 3.6 Форма добавлення пісні

Після добавлення пісні на сайт, користувач зможе редагувати або видалити пісню що він добавив, як і адміністратор.

Для пошуку пісні або групи, достатньо скористатись пошуком що знаходиться в шапці сторінки (Рис 3.7).



Рис 3.7 Пошук пісень та груп

Для перегляду даних своїх даних, користувач повинен перейти на сторінку акаунта(Рис 3.8). На сторінці буде відображено картинка профілю, логін, пошта та завантажені пісні даним користувачем. Він також в змозі змінити власну інформацію, перейшовши за посиланням змінити акаунт (дана функція доступна лише власнику акаунта і адміністраторам).

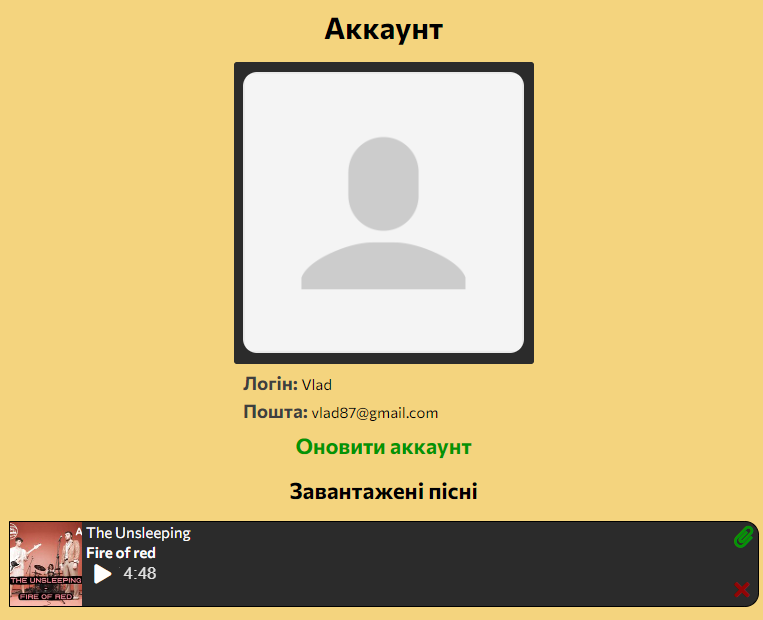


Рис 3.8 Сторінка користувача

Веб-додаток “OnMuzik” доступний не лише для власників персональних комп’ютерів, в додатку реалізований адаптивний режим з повною функціональністю до розширення ширини в 300 пікселів, тобто на сучасних телефонах користуватись даним сайтом буде зручно.

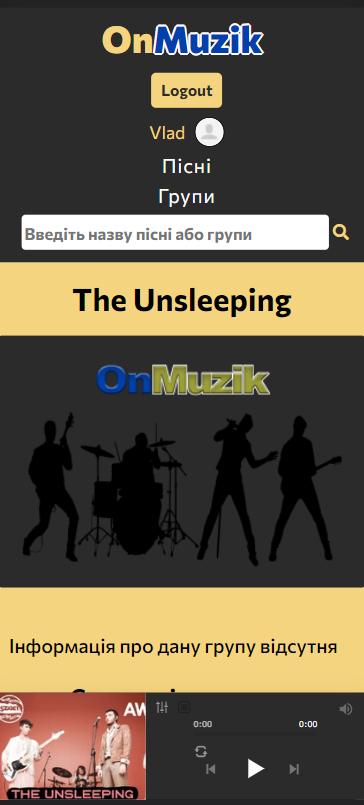


Рис 3.9 Мобільний вигляд сайту

Для користувачів з рівнем доступу адміністратор зроблено окреме посилання (рис 3.10 - 3.11) на список всіх користувачів сайту, для перегляду та зміни інформації про них.

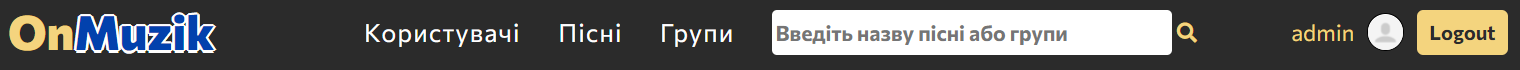


Рис 3.10 панель адміністратора

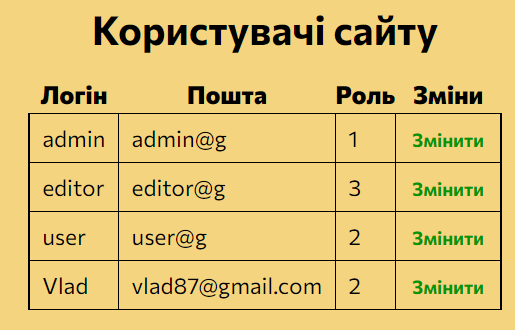


Рис 3.11 Сторінка адміністратора

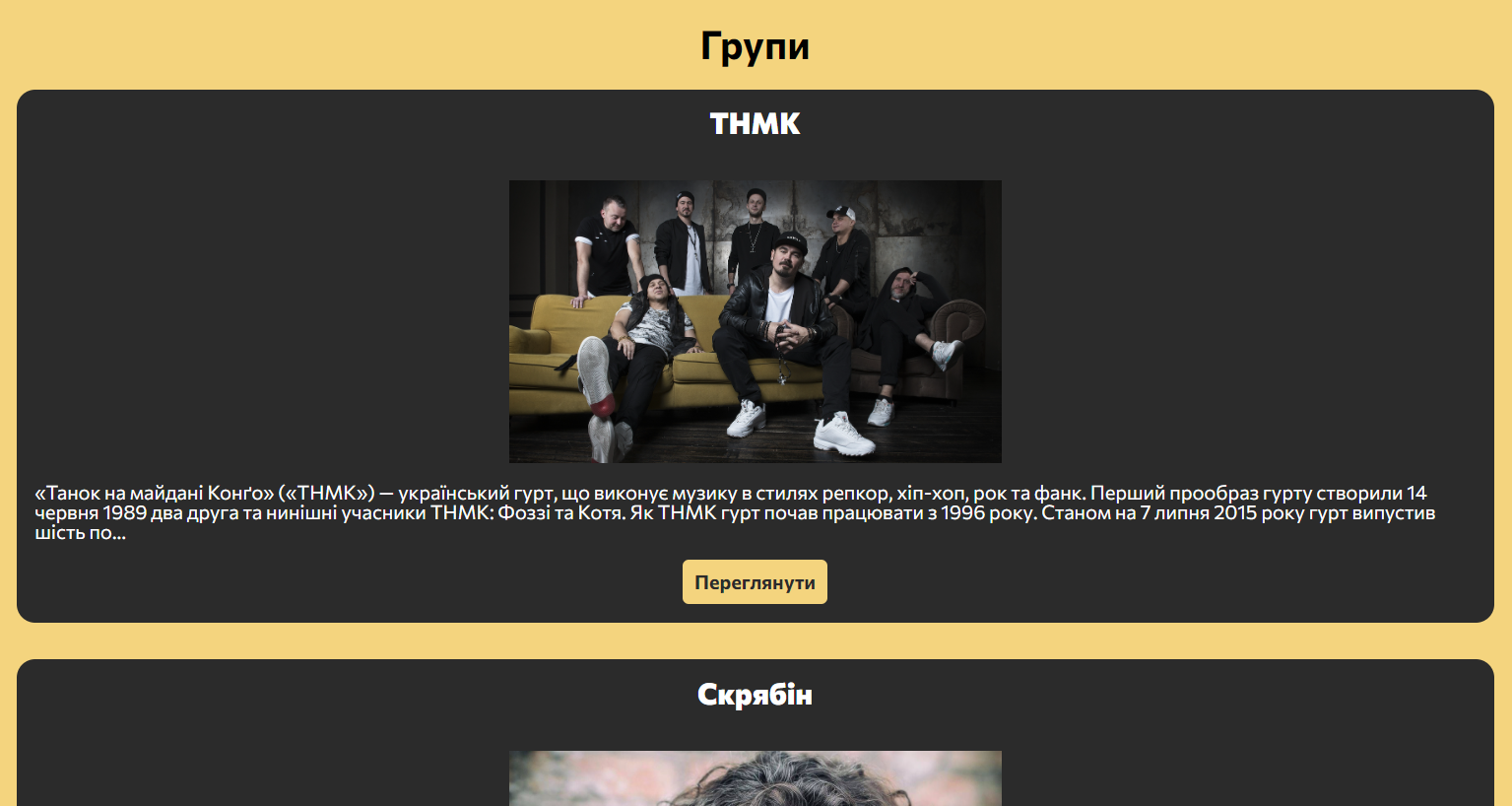


Рис 3.12 Сторінка виконавців

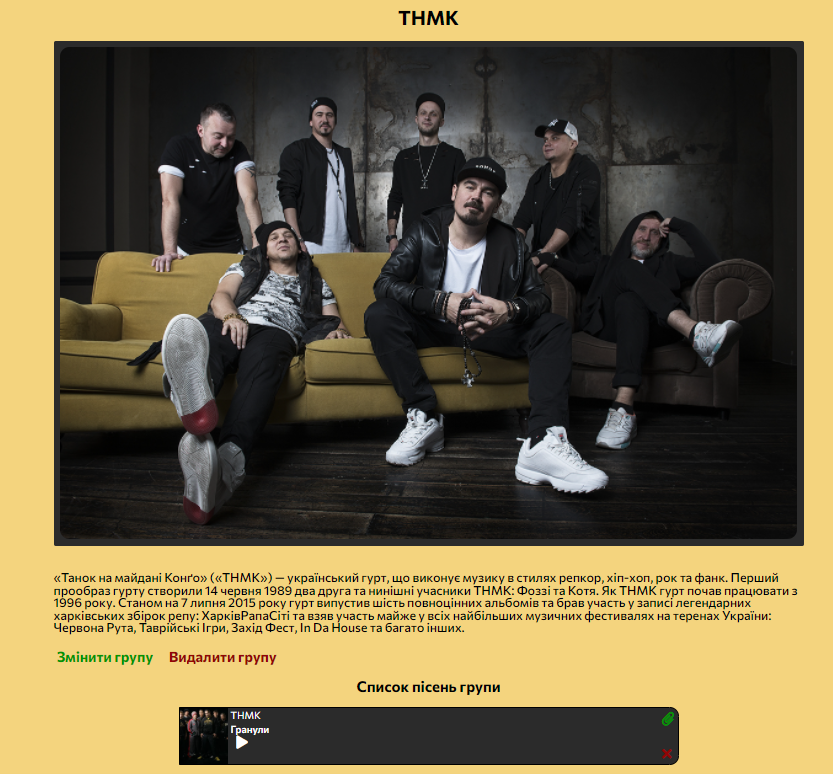


Рис 3.13 Сторінка перегляду виконавця

## **Висновки до 3 розділу :**

У ході виконання третього розділу було детально описано роботу Web-сайту, користувацький інтерфейс, очікувані дії користувача та їх результати. Також було розглянуто основні можливості веб-додатку.

# **ВИСНОВКИ**

Під час написання даної курсової роботи було отримано навички роботи з розробки веб-додатку.

В першому розділі курсової роботи, були проаналізовані знайдені портали для прослуховування та завантаження музики зі схожим функціоналом та інтерфейсом, та було сформовано уявлення про портал для прослуховування та завантаження музики в цілому, що розроблюється. Було виявлено основні потреби для написання даного веб-додатку.

В другому розділі курсової роботи, було проведено розробку функціональних і загальних частин веб-додатку. При розробці функціонального алгоритму, розробили уявлення про методи розробки програмного коду. На заключному етапі, розроблено його. В результаті, отримано готовий програмний продукт, який виконує всі заплановані функції і задачі.

В третьому розділі курсової роботи було описано методи користування створеним веб-додатком, були пояснені і продемонстровані всі створені елементи інтерфейсу. Також було проведено тестування. В ході тестування проблем не виникало.

В результаті виконання курсової роботи отримано портал для прослуховування та завантаження музики «OnMuzik», який повністю відповідає запланованому функціоналу, умовам та вимогам, поставленим на початку проектування. Веб-додаток є зрозумілим та зручним в користуванні.

# **СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ**

1. Розробка CMS системи на PHP, автор Андрій морозов -  
   <https://www.youtube.com/playlist?list=PLqKuDFs5Nd3L5hVkFH05vLYIgWeDTimDb>
2. Документація по настройці плеєра Playerjs - <https://playerjs.com/>
3. LIKE (Transct-SQL) -  
   <https://docs.microsoft.com/ru-ru/sql/t-sql/language-elements/like-transact-sql?view=sql-server-ver15>
4. Документація для PHP 7.2 - <https://www.php.net/>
5. Open Server - https://ospanel.io/

# **ДОДАТКИ**

Додаток А

Controller.php:

<?php

namespace core;

/\*\*

 \*Basic class for controllers

 \*/

class Controller

{

    public function isPost(){

        if ($\_SERVER['REQUEST\_METHOD']=="POST")

            return true;

        else

            return false;

    }

    public function isGet()

    {

        if ($\_SERVER['REQUEST\_METHOD']=="GET")

            return true;

        else

            return false;

    }

    public function postFilter($fields)

    {

        return Utils::ArrayFilter($\_POST,$fields);

    }

    public function render($viewName, $localParams = null, $globalParams = null){

        $tmp = new Template();

        if (is\_array($localParams))

            $tmp->setParams($localParams);

        if (!is\_array($globalParams))

            $globalParams = [];

        $moduleName = strtolower((new \ReflectionClass($this))->getShortName());

        $globalParams['PageContent'] = $tmp->render("views/{$moduleName}/{$viewName}.php");

        return $globalParams;

    }

    public function renderMessage($type,$message, $localParams = null, $globalParams = null )

    {

        $tmp = new Template();

        if (is\_array($localParams))

            $tmp->setParams($localParams);

        $tmp->setParam('MessageText', $message);

        switch ($type){

            case 'ok':

                $tmp->setParam('MessageClass','success');

                break;

            case 'error':

                $tmp->setParam('MessageClass','error');

                break;

            case 'info':

                $tmp->setParam('MessageClass','info');

                break;

        }

        if (!is\_array($globalParams))

            $globalParams = [];

        $globalParams['PageContent'] = $tmp->render("views/layout/message.php");

        return $globalParams;

    }

}

Core.php:

<?php

namespace core;

use mysql\_xdevapi\Exception;

/\*\*

 \* Class Core

 \* @package core

 \*/

class Core

{

    private static $instance;

    private static $mainTemplate;

    private static $db;

    private function \_\_construct()

    {

        global $Config;

        spl\_autoload\_register( '\core\Core::\_\_autoload');

        self::$db = new \core\DB($Config['Database']['Server'],

            $Config['Database']['Username'],

            $Config['Database']['Password'],

            $Config['Database']['Database']);

    }

    /\*\*

     \* get Instance of Core

     \* @return Core

     \*/

    public static function getInstance(){

        if (empty(self::$instance)){

            self::$instance = new Core();

            return self::getInstance();

        }

        else

            return self::$instance;

    }

    /\*\*

     \* The main process of the site

     \*/

    public function run(){

        $path = $\_GET['path'];

        $pathParts = explode('/',$path);

        $className = ucfirst( $pathParts[0]);

        if(empty($className))

            $fullClassName = 'controllers\\Site';

        else

            $fullClassName = 'controllers\\'.ucfirst( $className);

        $methodName = ucfirst($pathParts[1]);

        if(empty($methodName))

            $fullMethodName = 'actionIndex';

        else

            $fullMethodName = 'action'.ucfirst($methodName);

        if(class\_exists($fullClassName)){

            $controller = new $fullClassName();

            if(method\_exists($controller, $fullMethodName)){

                $method = new \ReflectionMethod($fullClassName,$fullMethodName);

                $paramsArray = [];

                foreach ( $method->getParameters() as $parameter){

                    array\_push($paramsArray,isset($\_GET[$parameter->name]) ? $\_GET[$parameter->name] : null);

                }

                $result = $method->invokeArgs($controller, $paramsArray);

                if (is\_array($result)){

                    self::$mainTemplate->setParams($result);

                }

            }

            else

                throw new \Exception('404 Not Found');

        }

        else

            throw new \Exception("404 Not Found");

    }

    /\*\*

     \* Return db for future work with

     \* @return DB

     \*/

    public function getDB(){

        return self::$db;

    }

    public function init(){

        session\_start();

        self::$mainTemplate = new Template();

    }

    /\*\*

     \* Shoutdown the cms system and show the results

     \*/

    public function done(){

        self::$mainTemplate->display('views/layout/index.php');

    }

    public static function \_\_autoload($className)

    {

        $filename = $className.'.php';

        if(is\_file($filename))

            include ($filename);

    }

}

DB.php:

<?php

namespace core;

class DB

{

    protected $pdo;

    public function \_\_construct($server, $login, $password, $database)

    {

        $this->pdo = new \PDO("mysql:host={$server};dbname={$database};charset=UTF8", $login, $password);

    }

    /\*\*

     \* select from database

     \* @param $table

     \* @param string $fields

     \* @param null $where

     \* @param null $orderBy

     \* @param null $limit

     \* @param null $offset

     \* @return array

     \*/

    public function select($table, $fields = "\*", $where = null, $orderBy = null, $limit = null, $offset = null, $like = null)

    {

        $fieldsSTR = "\*";

        if (is\_string($fields)) {

            $fieldsSTR = $fields;

        }

        if (is\_array($fields)) {

            $fieldsSTR = implode(", ", $fields);

        }

        $sql = "SELECT {$fieldsSTR} FROM {$table}";

        if (is\_array($where) && count(($where)) > 0) {

            $whereParts = [];

            foreach ($where as $key => $value) {

                if (!empty($like)) {

                    $where[$key] .= "%";

                    $whereParts[] = "{$key} LIKE ?";

                } else {

                    $whereParts[] = "{$key} = ?";

                }

            }

            $whereSTR = implode(" AND ", $whereParts);

            $sql .= ' WHERE ' . $whereSTR;

        }

        if (is\_string($where)) {

            $sql .= " WHERE " . $where;

        }

        if (is\_array($orderBy)) {

            $orderByParts = [];

            foreach ($orderBy as $key => $value) {

                $orderByParts[] = " {$key} {$value}";

            }

            $sql .= ' ORDER BY ' . implode(", ", $orderByParts);

        }

        if (!empty($limit)) {

            if (!empty($offset))

                $sql .= " LIMIT {$offset}, {$limit}";

            else

                $sql .= " LIMIT {$limit}";

        }

        $sth = $this->pdo->prepare($sql);

        if (is\_array($where) && count(($where)) > 0)

            $sth->execute(array\_values($where));

        else

            $sth->execute();

        return $sth->fetchAll();

    }

    /\*\*

     \* insert into database

     \* @param $table

     \* @param $row

     \*/

    public function insert($table, $row)

    {

        $fieldsSTR = implode(", ", array\_keys($row));

        $valuesParts = [];

        foreach ($row as $key => $value) {

            $valuesParts[] = '?';

        }

        $valuesSTR = implode(', ', $valuesParts);

        $sql = "INSERT INTO {$table} ($fieldsSTR) VALUES ($valuesSTR)";

        $sth = $this->pdo->prepare($sql);

        $sth->execute(array\_values($row));

        return $this->pdo->lastInsertId();

    }

    /\*\*

     \* delete from database

     \* @param $table

     \* @param null $where

     \*/

    public function delete($table, $where = null)

    {

        $sql = "DELETE FROM {$table}";

        if (is\_array($where) && count(($where)) > 0) {

            $whereParts = [];

            foreach ($where as $key => $value)

                $whereParts[] = "{$key} = ?";

            $whereSTR = implode(" AND ", $whereParts);

            $sql .= ' WHERE ' . $whereSTR;

        }

        if (is\_string($where))

            $sql .= " WHERE " . $where;

        $sthDel = $this->pdo->prepare('SET FOREIGN\_KEY\_CHECKS=0;');

        $sthDel->execute();

        $sth = $this->pdo->prepare($sql);

        if (is\_array($where) && count(($where)) > 0)

            $sth->execute(array\_values($where));

        else

            $sth->execute();

        $sthDel = $this->pdo->prepare('SET FOREIGN\_KEY\_CHECKS=1;');

        $sthDel->execute();

    }

    /\*\*

     \* update table in database

     \* @param $table

     \* @param $newRow

     \* @param $where

     \*/

    public function update($table, $newRow, $where)

    {

        $sql = "UPDATE {$table} SET ";

        $setParts = [];

        $paramsArr = [];

        foreach ($newRow as $key => $value) {

            $setParts[] = "{$key} = ?";

            $paramsArr[] = $value;

        }

        $sql .= implode(', ', $setParts);

        if (is\_array($where) && count(($where)) > 0) {

            $whereParts = [];

            foreach ($where as $key => $value) {

                $whereParts[] = "{$key} = ?";

                $paramsArr[] = $value;

            }

            $whereSTR = implode(" AND ", $whereParts);

            $sql .= ' WHERE ' . $whereSTR;

        }

        if (is\_string($where))

            $sql .= " WHERE " . $where;

        $sth = $this->pdo->prepare($sql);

        $sth->execute($paramsArr);

    }

}