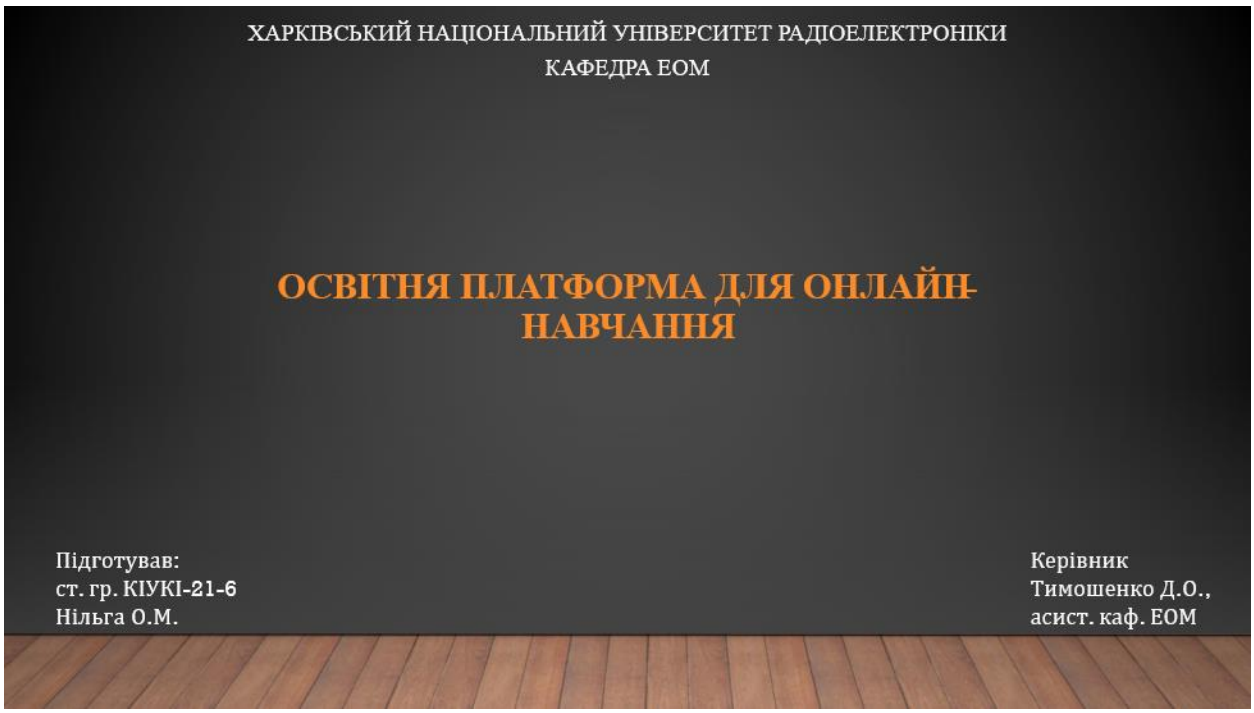
ДОДАТОК А

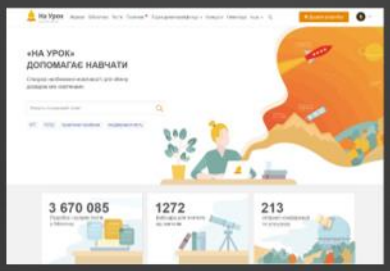
Графічний матеріал кваліфікаційної роботи


ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ
КАФЕДРА ЕОМ
ОСВІТНЯ ПЛАТФОРМА ДЛЯ ОНЛАЙН-НАВЧАННЯ
Підготував:
ст. гр. КІУКІ-21-6
Нільга О.М.
Керівник
Тимошенко Д.О.,
асист. каф. ЕОМ


2
АКТУАЛЬНІСТЬ ТЕМИ
Онлайн-платформи сприяють формуванню культури безперервного навчання.
Зручна організація навчального процесу.
Доступ до великої кількості матеріалів.
Зростання попиту на освітні онлайн-платформи.

3

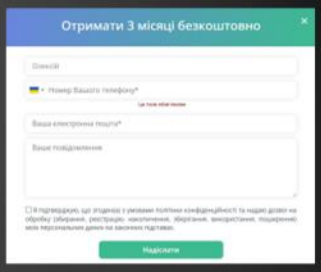
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

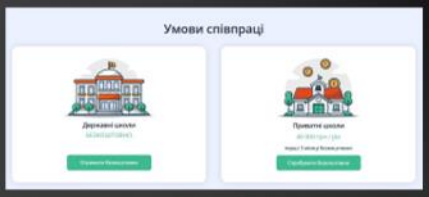


4

АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

- Одним із недоліків сайту є обов’язкова приналежність до державної, або приватної школи для того щоб створити повноцінний аккаунт





5 ПІДСУМКИ АНАЛІЗУ ІСНУЮЧИХ РІШЕНЬ

Проведений порівняльний аналіз таких поширених веб-сайтів:

- освітній проект «Prosvita»;
- освітній проект «На Урок»;
- освітній проект «Google classroom»;
- освітній проект «Moodle».

- Результати порівняльного аналізу свідчать про необхідність створення універсального рішення з гнучкою структурою курсів, простим та зрозумілим дизайном, різними категоріями користувачів та локалізацією українською мовою.

6 ПОСТАНОВКА ЗАДАЧІ

При реалізації веб-застосунку першочерговим завданням є визначення ролей користувачів. У веб-застосунку буде роль адміністратора, користувача та гостя.

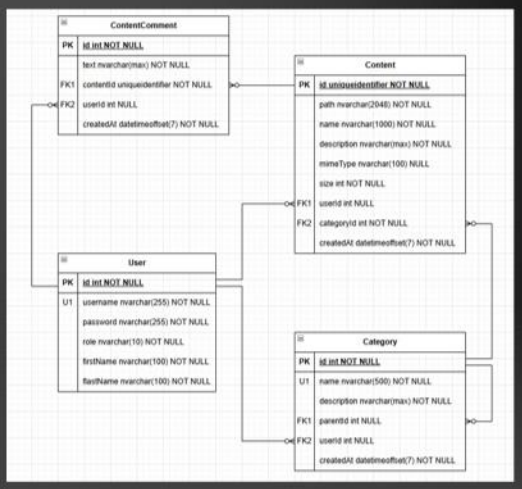
Гість матиме можливість зареєструватись або увійти в профіль.

Зареєстрований користувач матиме особистий кабінет, в якому зможе переглядати особисті дані та доступні йому курси, переглядати інформацію пов'язану з цими курсами.

Сайт повинен мати навчальні матеріали до яких користувачі зможуть доступитись по ієрархічній системі курсів.

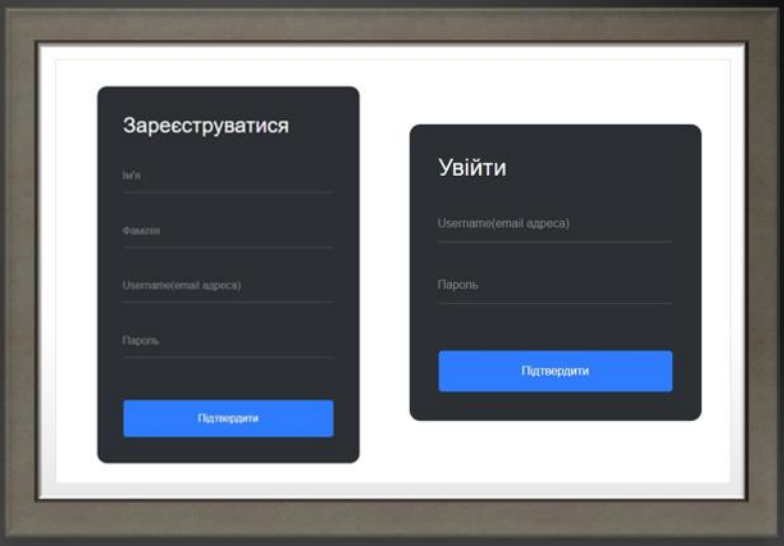
7

СХЕМА БАЗИ ДАНИХ

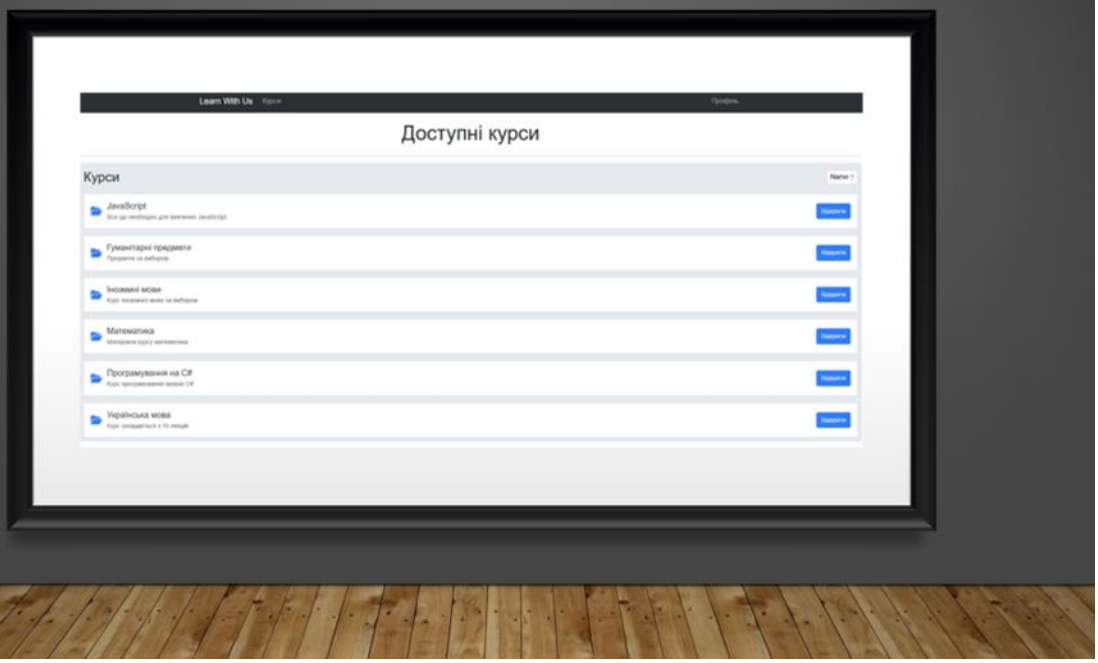


8

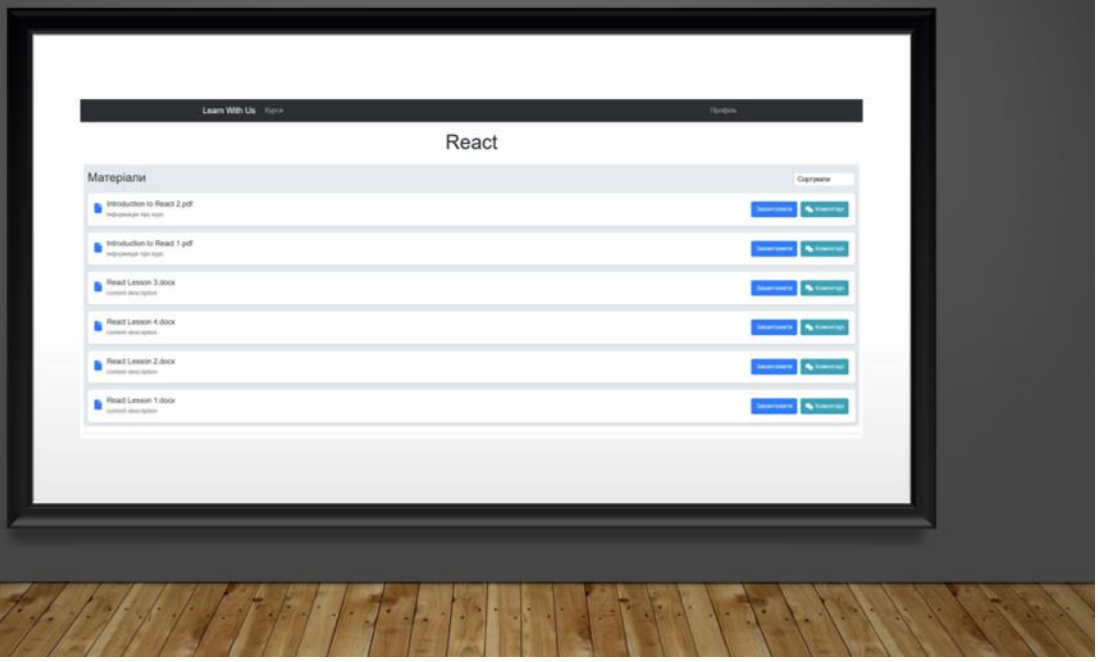
ФОРМИ ДЛЯ АВТОРИЗАЦІЇ



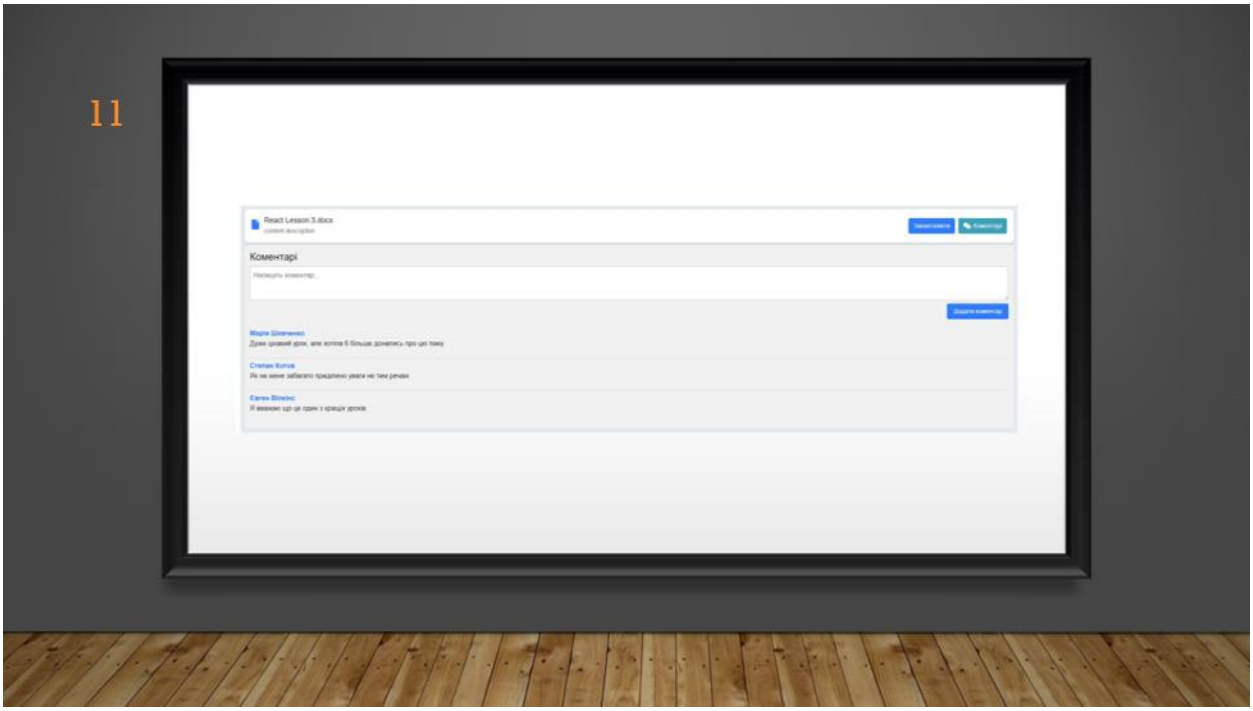
9


Learn With Us
Доступні курси
Курси
JavaScript
Гуманітарні предмети
Іноземні мови
Математика
Програмування на C#
Українська мова

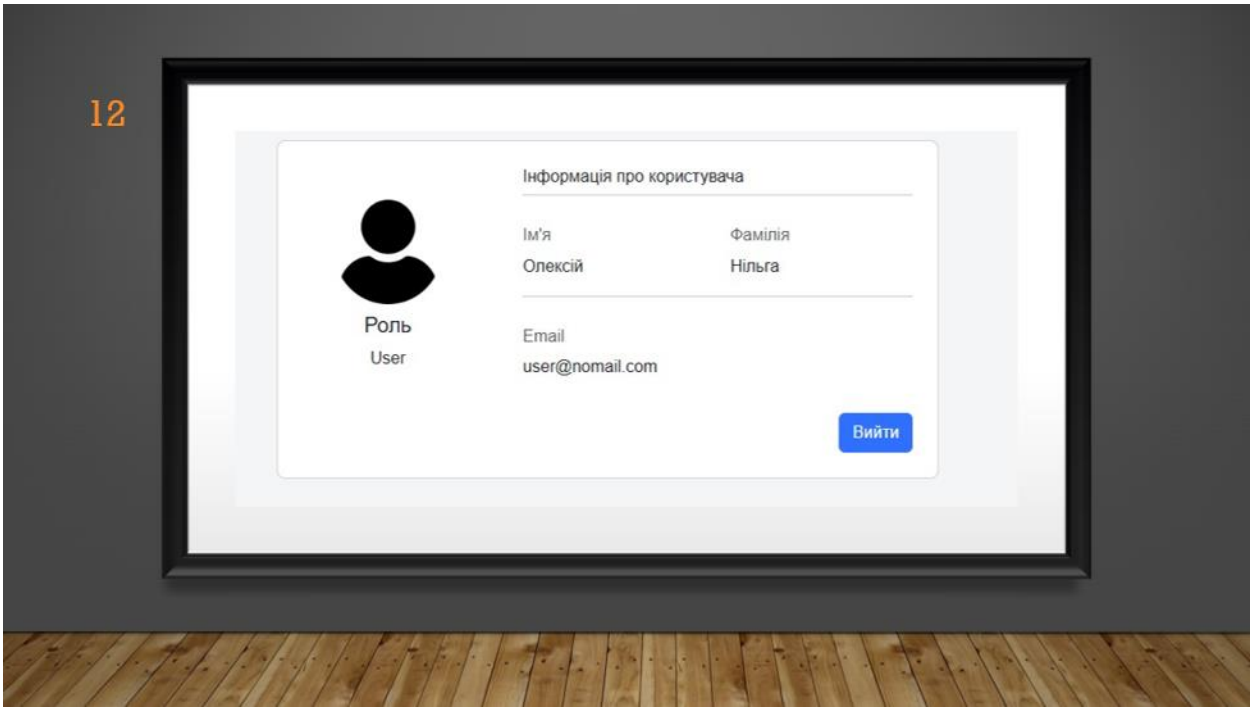
10


Learn With Us
React
Матеріали
Introduction to React 2.pdf
Introduction to React 1.pdf
React Lesson 3.docx
React Lesson 4.docx
React Lesson 2.docx
React Lesson 1.docx

11

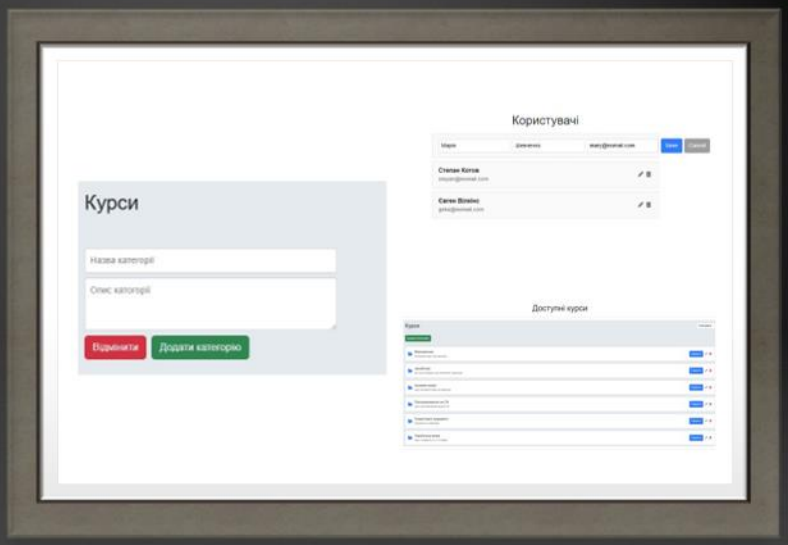

Коментарі

12


Інформація про користувача
Ім'я
Олексій
Фамілія
Нільга
Роль
User
Email
user@nomail.com
Вийти

13

МЕНЮ АДМІНА



14

ВИСНОВКИ

- Кваліфікаційна робота присвячена розробці веб -сервісу для онлайн навчання. Цей додаток має мету покращити або вдосконалити якість навчання для людей різного віку та матеріального статусу.
- У процесі дослідження були вивчені можливості популярних освітніх платформ, таких як Prosvita, На Урок, Google Classroom та Moodle, визначено їхні переваги й недоліки.
- Було спроєктовано архітектуру веб-застосунку, реалізовано систему користувачів з розмежуванням ролей, створено інтерфейс для управління навчальними курсами, завданнями, тестами та оцінюванням.
- Застосунок демонструє приклад сучасного React додатку. Чітку архітектуру, використання безпечної автентифікації та захисту облікових даних користувачів, грамотну маршрутизацію та інтеграцію з популярними UI-бібліотеками.
- Проєкт є повністю робочим та виконує всі поставлені завдання.

ДОДАТОК Б

Вихідний код

Б.1 Модель Категорії

```
module.exports = (sequelize, DataTypes) => {
  const Category = sequelize.define(
    'Category',
    {
      // Define columns
      id: {
        type: DataTypes.INTEGER,
        primaryKey: true,
        autoIncrement: true,
      },
      name: {
        type: DataTypes.STRING(500),
        unique: { msg: 'Category name has already been taken' },
// Enforce uniqueness
        validate: {
          notEmpty: { msg: "name cannot be empty" },
          len: {
            args: [3, 500],
            msg: "Catgory name must be from 3 to 500 characters"
          }
        }
      },
      description: {
        type: DataTypes.TEXT,
        validate: {
          notEmpty: { msg: "description cannot be empty" }
        }
      },
      parentId: {
        type: DataTypes.INTEGER,
        allowNull: true
      },
      userId: {
        type: DataTypes.INTEGER,
        allowNull: true
      },
      createdAt: {
        type: DataTypes.DATE,
        allowNull: false,
        defaultValue: DataTypes.NOW
      },
    },
```

```
    {
      tableName: 'Category', // Match your actual table name
      timestamps: false   // Disable if your table doesn't use
timestamps
    }
  );

  Category.associate = (models) => {
    Category.belongsTo(models.User, {
      foreignKey: 'userId',
      as: 'creator',
    });

    Category.hasMany(models.Content, {
      foreignKey: 'categoryId',
      as: 'contents',
    });
  };

  Category.hasMany(Category, {
    as: 'children',
    foreignKey: 'parentId'
  });

  Category.belongsTo(Category, {
    as: 'parent',
    foreignKey: 'parentId'
  });


  return Category;
};
```

Б.2 Модель Користувачів

```
module.exports = (sequelize, DataTypes) => {

  const User = sequelize.define(
    'User',
    {
      // Define columns
      id: {
        type: DataTypes.INTEGER,
        primaryKey: true,
        autoIncrement: true,
      },
      username: {
        type: DataTypes.STRING(255), // 255 characters max
        allowNull: false, // ensures it's not null
        unique: { msg: 'Username has already been taken' },
        validate: {
```

```
          isEmail: { msg: "Username must be a valid email
address" }, // ensures it's a valid email format
          notEmpty: { msg: "Username is required" } // ensures
it's not an empty string
        }
      },
      password: {
        type: DataTypes.STRING(255), // 255 characters max
        validate: {
          notEmpty: { msg: "Password is required" }
        }
      },
      role: {
        type: DataTypes.STRING(10),
        validate: {
          isIn: {
            args: [['Admin', 'User']],
            msg: 'Role must be one of Admin, User'
          }
        },
        defaultValue: "User"
      },
      firstName: {
        type: DataTypes.STRING(100),
        allowNull: false, // ensures it's not null
        validate: {
          notEmpty: { msg: "First name is required" }, //
ensures it's not an empty string
          len: {
            args: [2, 100],
            msg: "First name must be from 2 to 100 characters"
          }
        },
      },
      lastName: {
        type: DataTypes.STRING(100),
        allowNull: false, // ensures it's not null
        validate: {
          notEmpty: { msg: "Last name is required" }, // ensures
it's not an empty string
          len: {
            args: [2, 100],
            msg: "Last name must be from 2 to 100 characters"
          }
        }
      },
    },
    {
      tableName: 'User', // Match your actual table name
      timestamps: false   // Disable if your table doesn't use
timestamps
    },
  );
```

```
  // Override toJSON on the prototype, to hide the password hash
  User.prototype.toJSON = function () {
    const values = { ...this.get() };
    delete values.password;
    return values;
  };

  User.associate = (models) => {

    User.hasMany(models.Category, {
      foreignKey: 'userId',
      as: 'categories',
    });

    User.hasMany(models.Content, {
      foreignKey: 'userId',
      as: 'contents',
    });

    User.hasMany(models.ContentComment, {
      foreignKey: 'userId',
      as: 'contentComments',
    });

  };

  return User;
};
```

Б.3 Аутентифікація користувачів

```
export async function registerUser(formData) {
    try {
        const response = await axios.post(
            `${process.env.REACT_APP_API_BASE_URL}/user`,
            formData,
            {headers: {'Content-Type': 'application/json'}}
        );

        return {
            status: response.status,
            success: true,
            message: "Success"
        }
    } catch (error) {
        return {
            status: error.response.status,
            success: false,
            message: error.response.data.error
        }}}
```

```
export async function loginUser(formData) {

    try {
        const response = await axios.post(
            `${process.env.REACT_APP_API_BASE_URL}/user/login`,
            formData,
            {headers: {'Content-Type': 'application/json'}}
        );
        return {
            status: response.status,
            success: true,
            message: "Success",
            token: response.data.token
        }
    } catch (error) {
        return {
            status: error.response.status,
            success: false,
            message: error.response.data.error
        }
    }
}

export const getUserData = async (userId) => {
    try {
        const token = localStorage.getItem("token");
        const response = await axios.get(

`${process.env.REACT_APP_API_BASE_URL}/user/${userId}`,
            {
                headers: {
                    'Content-Type': 'application/json',
                    'Authorization': `Bearer ${token}`
                }
            }
        );
        return {
            status: response.status,
            success: true,
            message: "Success",
            data: response.data
        }
    } catch (error) {
        return {
            status: error.response?.status,
            success: false,
            message: error.response?.data?.error || "An error
occurred",
            data: null
        }}}
```