МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК
СЕКЦІЯ ІКТ

ВИПУСКНА РОБОТА

на тему:

«Верстка книжного інтернет-магазину»

Завідувач
випускаючої кафедри **Довбиш А.С.**

Керівник роботи **Шаповалов С. П.**

Студент гр. ІН-61 **Дремлюга О. О.**

Суми 2020

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК
СЕКЦІЯ ІКТ

Затверджую_____

Зав. кафедрою Довбиш А.С.

“_____” _____2020 р.

ЗАВДАННЯ

до випускної роботи

Студента четвертого курсу, групи ІН-61 спеціальності “Інформатика” денної форми навчання Дремлюги Олега Олександровича.

Тема: “ Верстка книжного інтернет-магазину ”

Затверджена наказом по СумДУ
№_____ від _____2020 р.

Зміст пояснювальної записки:1)огляд існуючих рішень; 2)постановка завдання й формування завдань дослідження;3)проектування та розробка інтернет-магазину;4) програмна реалізація та її опис;5)висновки.

Дата видачі завдання “_____”_____2020 р.
Керівник випускної роботи_____ Шаповалов С.П.
Завдання прийняв до виконання _____ Дремлюга О. О.

РЕФЕРАТ

Записка: 41 стор., 12рис., 1 табл., 2 додаток, 8 джерела.

Об’єкт дослідження — веб-ресурс для продажу книжок.

Мета роботи — придбання теоретичних знань і практичних умінь з розробки веб додатку , а також з використання фреймворків та веб-технологій таких як “express” “Bootstrap” і тд…

Методи дослідження — технології створення веб-додатків.

Результати — розроблено книжковий інтернет-магазин на основі фреймворку node js “express”. Всі вимоги були враховані. Після тестування платформи не було виявлено багів.

WEB , NODE JS, EXPRESS, MONGODB, HTML, CSS, JAVASCRIPT, JQUERY

ЗМІСТ

ВСТУП

Інтернет безперечно є найбільший винахід людства. За півстоліття існування робота мережі інтернет кардинально змінила парадигму світу. Мережа проникла в найвіддаленіші куточки планети. Свобода і інформативність надали простому користувачеві необмежені можливості по самоосвіті і спілкуванню.

Інтернет дозволяє обмінюватися різними файлами на великі відстані. Інший користувач всесвітньої павутини може задати будь-яке питання, яке його цікавить, і отримати відповідь від інших користувачів[1, с. 64].

Сьогодні інтернет - це глобальне явище, яке охопило всі сфери людського життя. Напевно, немає користувача, який би не задавався питанням, що таке інтернет і для чого він потрібен.

Мережу використовують для:

- Пошуку і обміну інформацією.
- Комунікації.
- Розваги.
- Комерції.

У кожній сфері життя суспільства перебуває застосування інтернету.

Всесвітня павутина зростає. Нам все частіше пропонують роботу в Інтернеті. Нам пропонують реалізувати свій бізнес та здійснювати бізнес- процеси через Інтернет. Робота через Інтернет має свої переваги - це вільний графік роботи, відсутність транспортних витрат, економія часу на роботу і т. Д. Також розвивається бездротовий Інтернет, кілька років тому мобільні телефони не могли самостійно завантажувати сторінки. Більше вдосконалюйте моделі телефонів, екрани для зручності користування Інтернетом. Також Інтернет-мережа зручна для інтернет-магазинів, в ній ми можемо купити буквально все. Замовте улюблену дрібничку або придбайте подарунок другові. Ми можемо

оплачувати покупки за допомогою електронних грошей, що, до речі, дуже зручно. Ми також отримуємо додаткову освіту в Інтернеті, виконуючи завдання, які надсилаються нам поштою, не потрібно відвідувати університет, який цікавить вас [2].

Купувати в Мережі стало звичною справою для багатьох користувачів. Кожна поважаюча себе компанія має не тільки сайт, але і власний інтернет-магазин, через який продає товари.

В останні роки популярним стає таке поняття, як «інтернет речі», - ідея позбавити людину від необхідності здійснювати рутинні покупки. Розробники пропонують забезпечити побутові речі засобами доступу в інтернет, щоб вони здійснювали покупки за заданим алгоритмом. Наприклад, холодильник, підключений до Мережі, буде закуповувати ті продукти, які закінчуються або відсутні на його полицях

1 ОГЛЯД ІСНУЮЧИХ РІШЕНЬ

1.1 Дослідження предметної області

Веб-сайтом називають якусь сутність, сукупність веб-документів, об'єднаних в єдину структуру, яка розташована за певною доменною адресою. Існують різні класифікації веб-сайтів.

Web-ресурс - це сторінка або набір сторінок, розміщених в мережі Інтернет, які можуть включати як текстову і графічну інформацію, так і мультимедіа-компоненти (відео, музику і т.д.).[3, с. 52]

Сторінки веб-сайту можуть бути як статичними (plain-HTML), так і динамічними. Наприклад, вони можуть генеруватися на сервері гіпертекстовим препроцесором з використанням інформації з таблиць бази даних або включати в себе інформацію, яку видає скрипт. [4]

З поняттям веб-сайту тісно пов'язано поняття доменного імені, що визначає унікальний адреса, який використовується для того, щоб знайти веб-ресурс в мережі Інтернет.

1.2 Огляд існуючих аналогів

Тема роботи створення книжкового інтернет-магазину, і це не нова ідея, в наш час існує безліч ресурсів , що мають схожу мету та функціонал. Тому для того щоб створити конкурентноспроможний продукт потрібно оглянути існуючі аналоги та провести детальний аналіз. Для обстеження схожих рішень обрано такі інтернет-магазини:

- ‘Book24’ (https://book24.ua/) (рис 1.1)
- ‘Odissey’(https://odissey.kiev.ua/category_3754.html) (рис 1.2)
- ‘Bookzone’ (https://bookzone.com.ua/) (рис 1.3)
- ‘Papirus’ (https://papirus.kh.ua/?gclid=Cj0KCQjwlN32BRCCARIsADZ-J4uWqf5dPzJq1p3VGNUEI7QDKEILWUFZG9vLRKeFyC6ezpkYoJ_2qkUaAj5ZEALw_wcB) (рис 1.4)

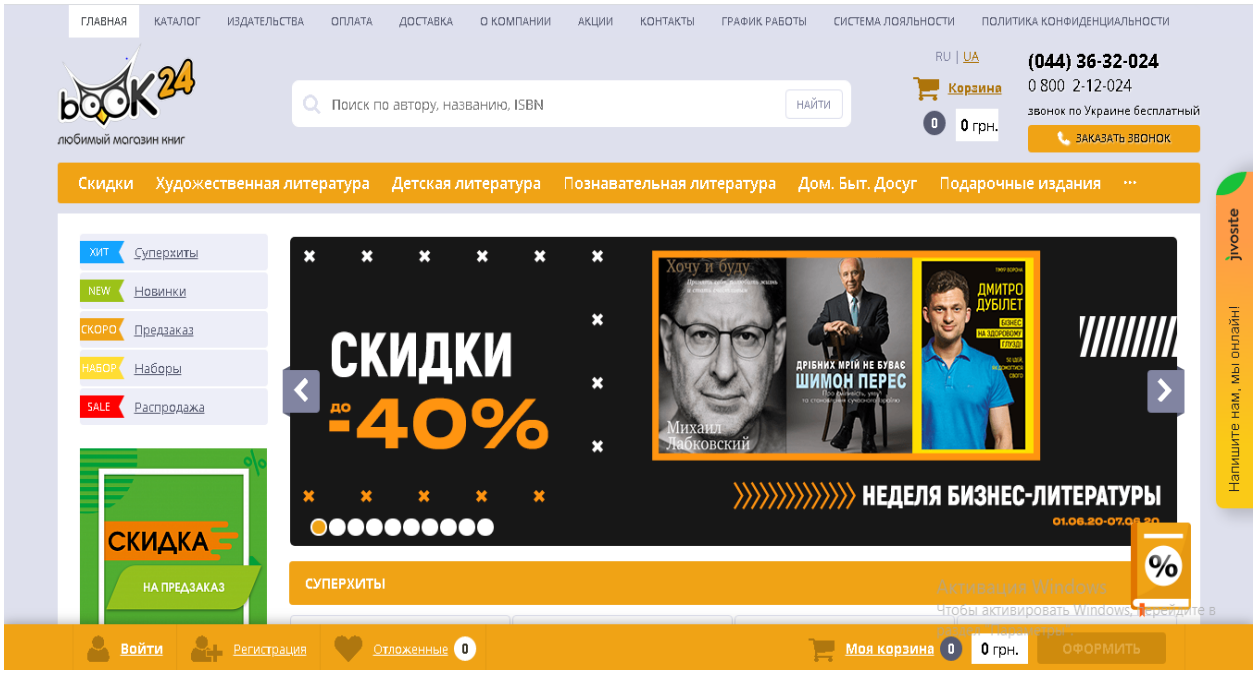


Рисунок 1.1 – Інтернет-магазин ‘Book24’

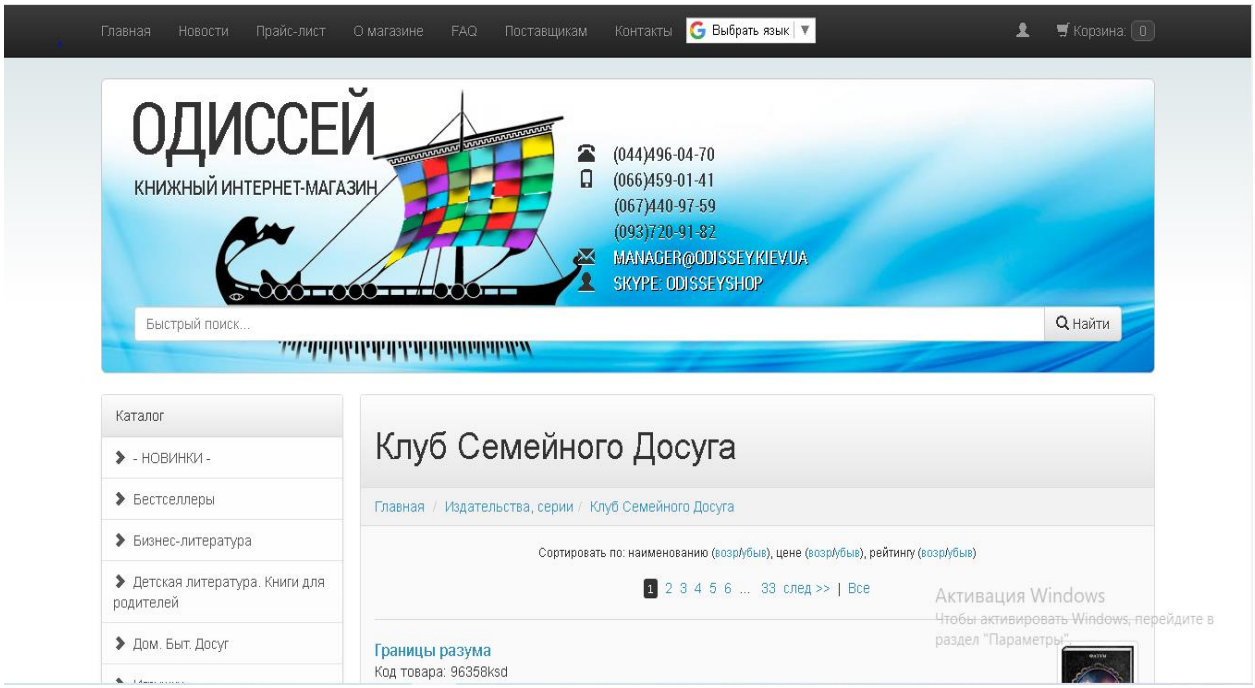


Рисунок 1.2 – Інтернет магазин ‘Одіссей’

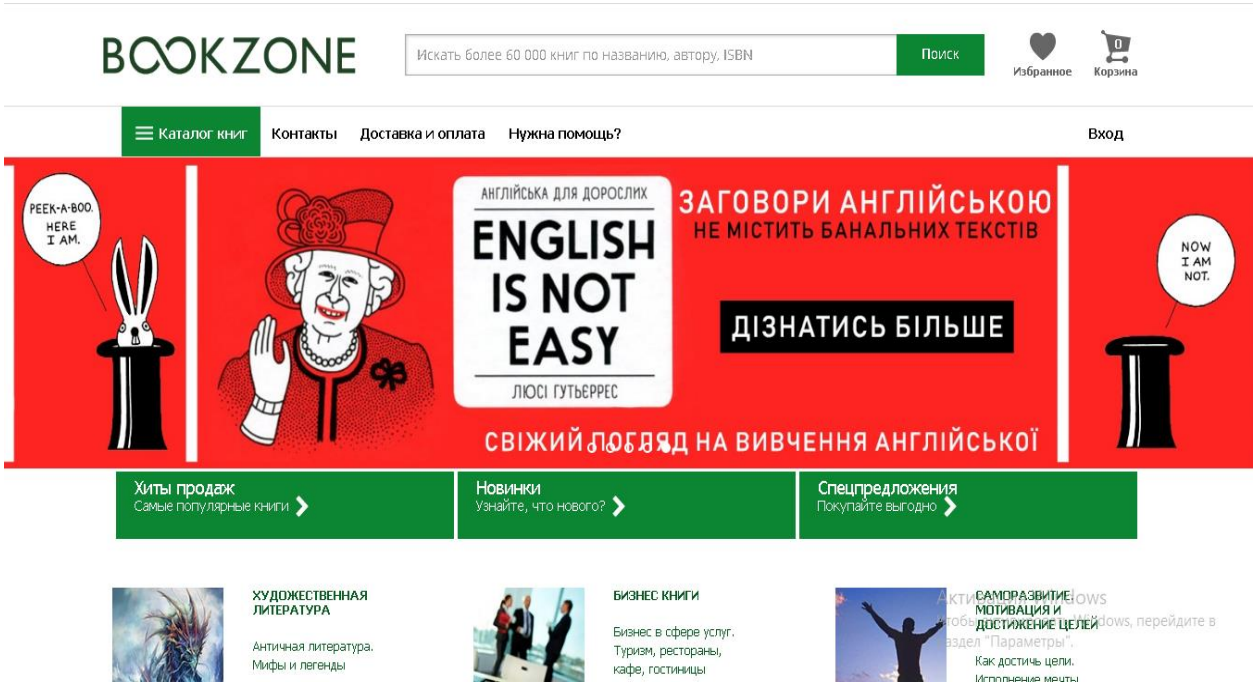


Рисунок 1.3 – Інтернет-магазин ‘Bookzone’

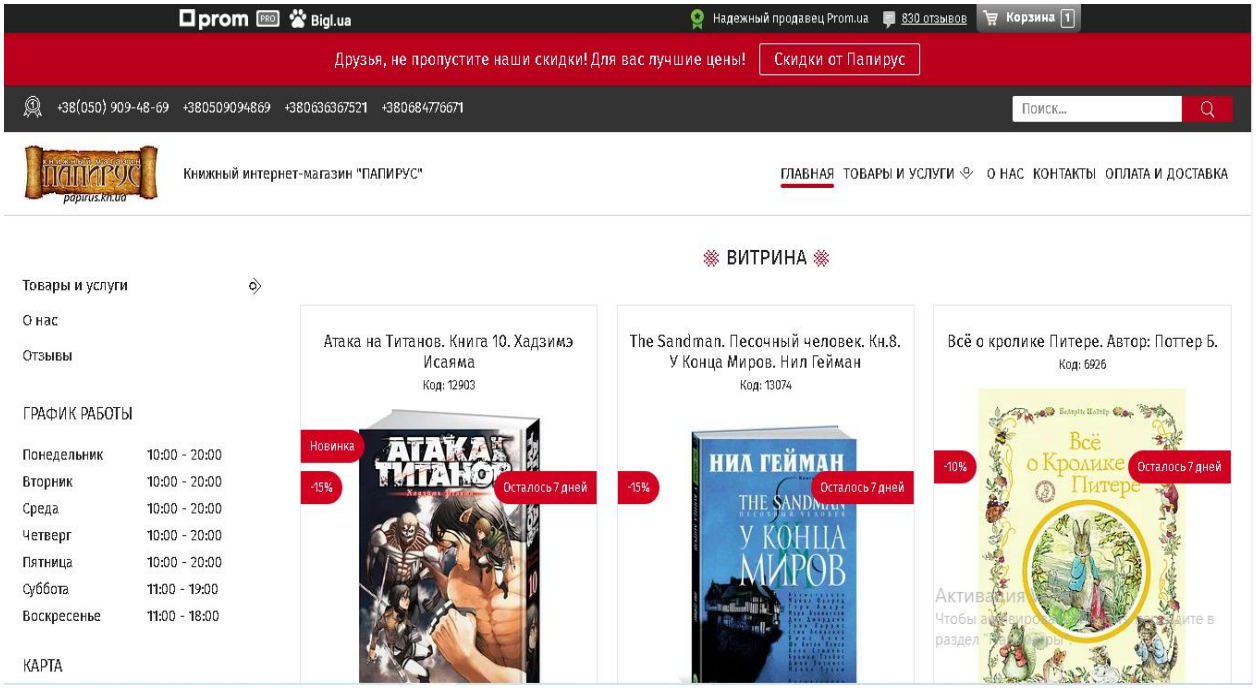


Рисунок 1.4 - Інтернет магазин ‘Папірус’

В ході дослідження подібних веб-ресурсів було побудовано порівняльну таблицю відомих рішень(табл 1.1)

Таблиця 1.1 – Порівняльна таблиця існуючих аналогів

Критерії	Назва сайту			
	‘Book24’	‘Одіссей’	‘Bookzone’	‘Папірус’
Зручний інтерфейс	+	-	+	-
Сучасний дизайн	+	-	+	-
Адаптивність	+	+	+	-
Блок з посиланнями на соціальні мережі	+	-	+	-
Відгуки клієнтів	+	+	+	+
Наявність особистого кабінету	+	+	+	-
Підписка для отримання спеціальних пропозицій	-	-	+	-
Контактна інформація	+	+	+	+
Кроссбраузерність	+	-	+	+

На основі огляду сайтів-аналогів та порівняльної таблиці див. табл. 1.1 було сформульовано переваги та недоліки існуючих веб-ресурсів.

Переваги:

- Зрозумілий для користувача інтерфейс
- Адаптивна верстка
- Кросбраузерність
- Можливість залишити відгук

Недоліки:

- Відсутність блоку з посиланнями на соціальні мережі
- Відсутність підписки на почтову скриньку
- Не вдалий дизайн

Всі ці критерії будуть враховані в наступному етапі - постановка задачі

1.3 Постановка задачі

Темою дипломного проекту є верстка книжного інтернет-магазину. Програмний продукт повинен відповідати таким вимогам:

- Зрозумілий інтерфейс з сучасним дизайном
- Можливість реєстрації та авторизації
- Перегляд існуючих товарів (книг)
- Перегляд книг за категорією
- Перегляд додаткової інформації потрібного товару
- Пошук потрібної книги за назвою або автором
- Пагінація для зручного перегляду товарів
- Можливість додати книгу до корзини покупця та видалити з неї
- Можливість залишити відгук
- Панель адміністратора (Створити новий товар або категорію)
- Адаптивна верстка для відображення сайту на різних пристроях

Результатом роботи повинен бути web-додаток для продажу книжок з панелю адміністратора, адаптивною верстkою та сервером що буде забезпечувати виконання вище зазначених функцій та критерій.

2 ВИБІР МЕТОДІВ ВИРІШЕННЯ

2.1 Вибір методів розроблення

Для написання серверу вирішено використати мову програмування node js та фреймворк “Express”.

Node.js - це програмна платформа, яка надає середовище для виконання коду JavaScript на стороні сервера.

На відміну від PHP, який працює під управлінням "стороннього" веб-сервера, наприклад, Apache або IIS, Node.js сама є веб-сервером. Але часто вона працює в зв'язці з Nginx в якості основного сервера. Використання Nginx зовсім не обов'язково, але виправдано для кешування даних запиту, віддачі статичних файлів або поділу доменів в межах одного сервера (reverse proxy).

Node.js відноситься до подієво-орієнтованих систем і використовує модель виконання операцій введення-виведення (I / O) таким чином, що основний процес ніколи не блокується, за винятком рідкісних випадків. А відсутність єдиного блокується потоку виконання дозволяє створювати ефективні, високопродуктивні і легко масштабовані серверні додатки. [5, с. 12]

Express - це фреймворк для Node.js, в якому реалізований шар функцій, необхідних для створення ефективних програм і API. Його використання значно скорочує час написання коду.

Node.js Express має готові функції обробки HTTP запитів, причому для кожного HTTP методу є своя функція, що особливо зручно при створенні REST API. І це далеко не єдина причина використання Express.

Фреймворк - програмна платформа, яка визначає структуру програмної системи; програмне забезпечення, що полегшує розробку і об'єднання різних компонентів великого програмного проекту. Це каркас програмної системи (або підсистеми). Може включати: допоміжні програми, бібліотеки коду, мова сценаріїв і інше програмне забезпечення, що полегшує розробку і об'єднання

різних компонентів великого програмного проекту. Зазвичай об'єднання відбувається за рахунок використання єдиного API.

2.2 Вибір засобів програмування

Для верстки сторінок сайту буде використано:

- **HTML** - мова розмітки гіпертексту, який призначений для створення веб-сторінок. Коли така сторінка відкривається в браузері, він переглядає код HTML, знаходить спеціальні символи, звані тегами, і використовує їх для створення елементів, таких як: малюнки, таблиці, посилання та ін.

- **MaterializeCSS** - сучасний, адаптивний фреймфорк, побудований на принципі матеріального дизайну. У MaterializeCSS включений набір компонентів, а також стилі для них. Принцип роботи схожий на Bootstrap

CSS - Каскадні таблиці стилів (Cascading Style Sheets, CSS) дозволяють зберігати колір, розміри тексту і інші параметри в стилях. Стилем називається набір правил форматування, який застосовується до елементу документа, щоб швидко змінити його зовнішній вигляд.

- **JQuery** – набір функцій JavaScript, що фокусується на взаємодії JavaScript і HTML. Бібліотека jQuery допомагає легко отримувати доступ до будь-якого елементу DOM, звертатися до атрибутів і вмісту елементів DOM, маніпулювати ними. Також бібліотека jQuery надає зручний API для роботи з AJAX.

- **Handlebars** - це шаблонний процесор, який динамічно генерує вашу HTML-сторінку, що економить ваш час на ручному оновленні.

Handlebars генерує ваш HTML, використовуючи структуру JSON і запускаючи її через шаблон. Ці шаблони написані в основному в звичайному HTML і набиті наповнювачами, які дозволяють вам при необхідності вводити дані.

Існують дві основні причини, за якими ви хочете створити шаблон для свого сайту. Перш за все, створення шаблону спонукає вас відокремлювати логічний код від фактичного подання, допомагаючи вам дотримуватися шаблону View / Controller. По-друге, шаблони зберігають ваш код чистим і підтримуваним, що, в свою чергу, робить процес оновлення вашого сайту швидким. Ви не створюєте сайт з Handlebars. Замість цього ви створюєте рекомендації і структури, які визначають, як має виглядати сайт, що не фокусуючись на сервері сторінку.[7]

В якості бази данних використаємо MongoDB та пакет mongoose для звязку серверу та БД

MongoDB - документоорієнтованих система управління базами даних з відкритим вихідним кодом, яка не потребує опису схеми таблиць. Класифікована як NoSQL, використовує JSON-подібні документи і схему бази даних. Написана на мові C ++. Використовується в веб-розробці, зокрема, в рамках JavaScript-орієнтованого стека MEAN.

NoSQL - термін, що позначає ряд підходів, спрямованих на реалізацію систем управління базами даних, що мають суттєві відмінності від моделей, що використовуються в традиційних реляційних СУБД з доступом до даних засобами мови SQL. Застосовується до баз даних, в яких робиться спроба вирішити проблеми масштабованості та доступності за рахунок атомарности і узгодженості даних. [8 с. 70].

2.3 Проектування структури сайту

Перш ніж розпочати програмну реалізацію слід чітко розуміти структуру майбутнього продукту для того щоб мінімізувати майбутні баги та уникнути аномалій в базі даних.

Для того щоб явно визначити межі системи та функціональні аспекти було розроблено uml діаграму варіантів використання рис2.1

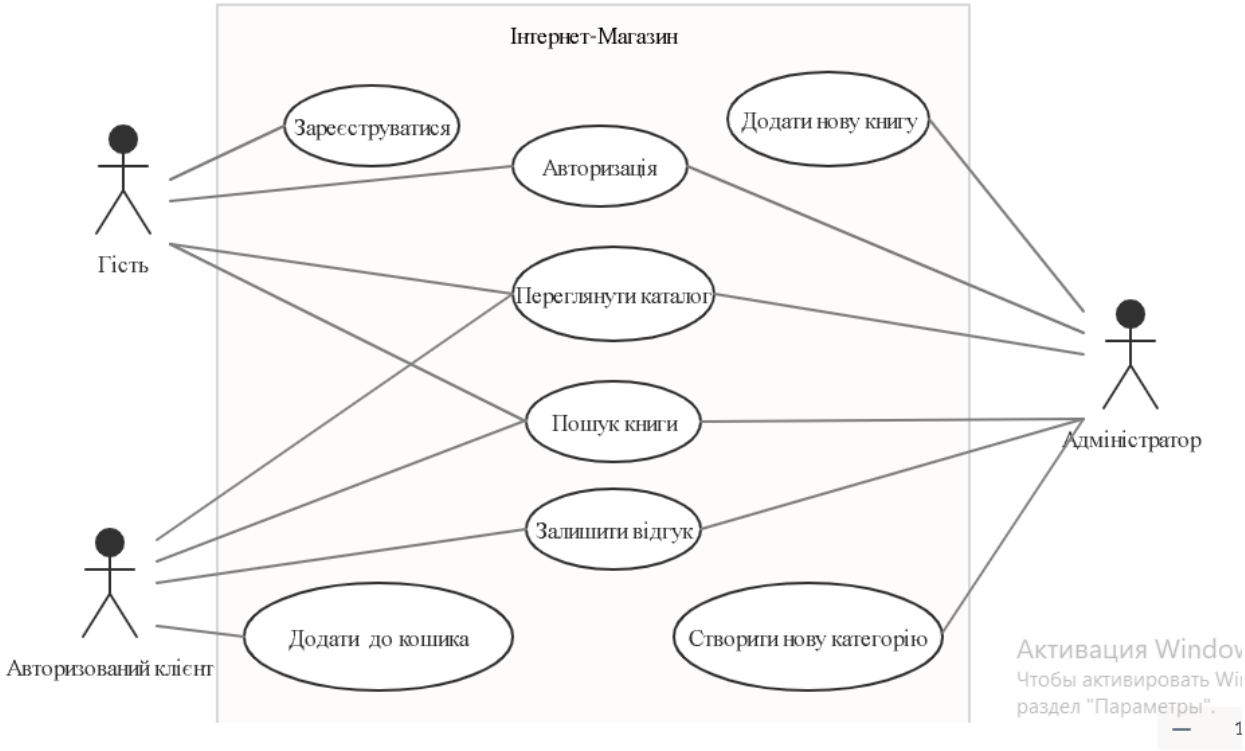


Рисунок 2.1 – Діаграма варіантів використання

Наступним етапом проектування стане опис внутрішньої структури сайту рис2.2

Рисунок 2.2 – Внутрішня структура сайту

Після того як визначено структуру та функціональні аспекти ресурсу можна перейти до проектування бази даних. Для створення ER діаграми треба визначити основні сутності що будуть присутні на сайті. Було виділено 5 основних сутностей:

- User – зберігає інформацію про користувачів сайту
- Product – містить інформацію про товари в каталозі сайту
- Category – перелік категорій
- Comment – відгуки до товару
- Basket – модель кошика покупок

Визначивши сутності можна створити ER-діаграму рис 2.3

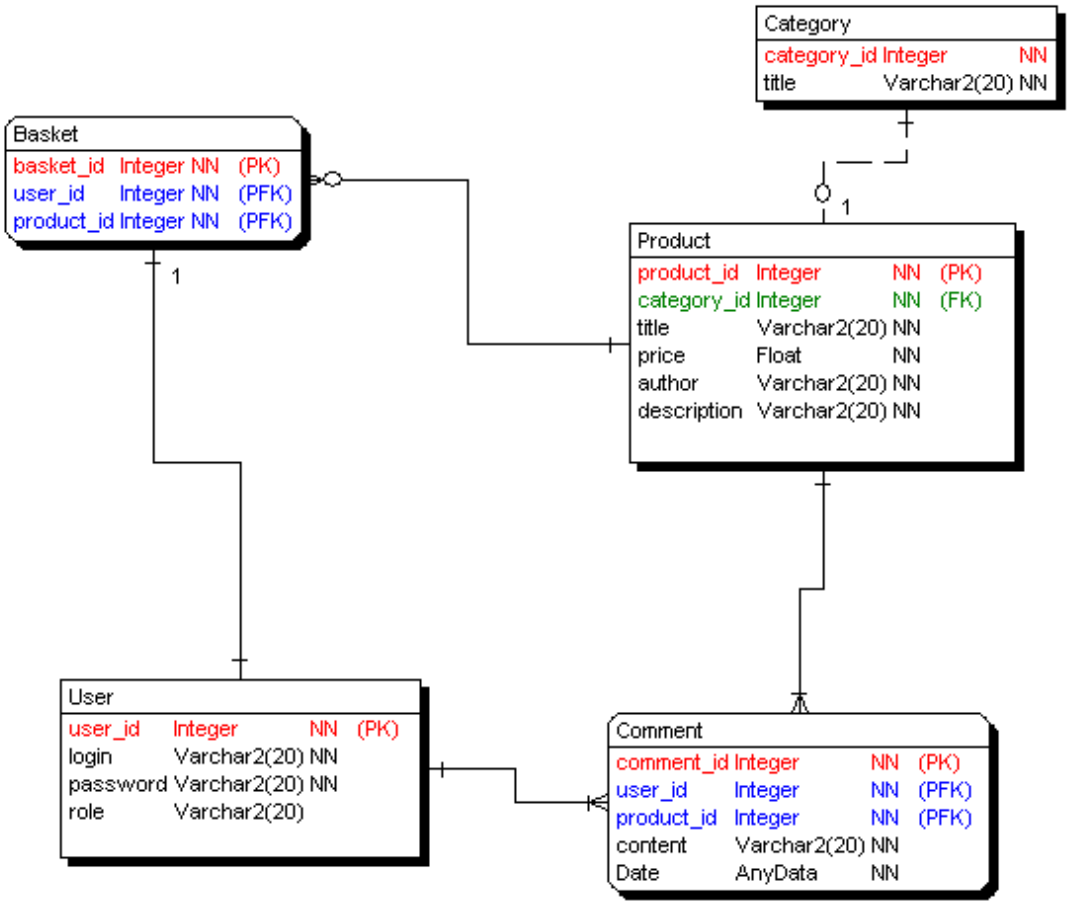


Рисунок 2.3 – IER- діаграма

Наступним кроком стане створення моделей для кожної таблиці бази даних.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Моделювання бази даних

На попередньому етапі було виділено 5 сутностей, кожна сутність буде мати окрему таблицю в базі даних. Оскільки було обрано MongoDB як систему управління даними то сутності мають бути представлені в вигляді моделей , що за синтаксисом нагадує JSON.

В результаті маємо 5 відповідних моделей:

Модель User.js

```
const User = new Schema({
    username: {
        type: String,
        required: true
    },
    mail: {
        type: String,
        required: true
    }
});
```

Модель Product.js

```
const Product = new Schema({
    img: {
        type: Object,
        required: true
    },
    title: {
        type: String,
        required: true
    },
    category: {
        type: Schema.ObjectId, ref: 'Category'
    },
    description: {
        type: String,
        required: true
    },
    price: {
        type: Number,
        required: true
    },
    author:{
        type: String,
        default: null
    },
    rate:{
        type: Number,
```

```
        default: 0
    }
});
```

Модель Category.js

```
const Category = new Schema({
    title: {
        type: String,
        required: true
    }
});
```

Модель Busket.js

```
const userBusket = new Schema({
    user: { type: Schema.ObjectId, ref: 'User', index: {unique: true, dropDups:
true}},
    products: [{ type: Schema.ObjectId, ref: 'Product', unique: true}]
});
```

Модель Comments.js

```
const Comment = new Schema({
    user: {
        type: Schema.ObjectId,
        ref: 'User',
        required: true
    },
    username: {
        type: String
    },
    product: {
        type: Schema.ObjectId,
        ref: 'Products',
        required: true
    },
    text: {
        type: String,
        required: true
    },
    data: {
        type: Date,
        default: Date.now()
    }
});
```

3.2 Написання серверної частини

Файл index.js – головний файл серверу відповідає за запуск серверу , підключення до MongoDB, підключення потрібних пакетів, бібліотек та функцій:

```
const express = require('express');
const mongoose = require('mongoose');
const path = require('path');
const exphbs = require('express-handlebars');
const categoryRoutes = require('./routes/categoryes');
const userRoutes = require('./routes/users');
const productRoutes = require('./routes/products');
var cookieParser = require('cookie-parser');
const bodyParser = require("body-parser");
const passport = require('passport');
const User = require('./models/User');
const passportLocalMongoose = require('passport-local-mongoose');
const expressSession = require('express-session')({
    secret: 'secret',
    resave: false,
    saveUninitialized: false
});

const PORT = process.env.PORT || 8000;

const app = express();
const hbs = exphbs.create({
    defaultLayout: 'main',
    extname: 'hbs'
});

app.engine('hbs', hbs.engine);
app.set('view engine', 'hbs');
app.set('views', 'views');

app.use(bodyParser.urlencoded({ extended: true }));
app.use(express.static(path.join(__dirname, 'public')));
app.use(cookieParser());
app.use(expressSession);

app.use(passport.initialize());
app.use(passport.session());

passport.use(User.createStrategy());
passport.serializeUser(User.serializeUser());
passport.deserializeUser(User.deserializeUser());

app.use(productRoutes);
app.use(userRoutes);
app.use(categoryRoutes);


async function start(){
    try{
        await mongoose.connect('mongodb+srv://oleh:oleh@cluster0-
napt5.mongodb.net/todos', {
```

```
            useNewUrlParser: true,
            useFindAndModify: false,
            useUnifiedTopology: true,
            useCreateIndex: true
        });
        app.listen(PORT, () => {
            console.log('Server started...');
        });
    }catch (e) {
        console.log(e);
    }
}



start();
```

Для підключення до бази даних використано бібліотеку Mongoose:

```
async function start(){
    try{
        await mongoose.connect('mongodb+srv://oleh:oleh@cluster0-
napt5.mongodb.net/todos', {
            useNewUrlParser: true,
            useFindAndModify: false,
            useUnifiedTopology: true,
            useCreateIndex: true
        });
        app.listen(PORT, () => {
            console.log('Server started...');
        });
    }catch (e) {
        console.log(e);
    }
}
```

Для авторизації користувача використаю бібліотеку express-passport:

```
router.post('/registration', async (req, res) => {
    try {
        const user = new User({username: req.body.username,
                               mail: req.body.mail});
        User.register(user, req.body.password, function(err, user){
            if(err){
                console.log('error registering user');
                console.log(err);
                return res.send('oops');
            }
```

```
            passport.authenticate('local')(req, res, function(){
                req.logIn(user, function(err) {
                    if (err) {
                        console.log(err);
                    }
                    res.cookie('userID', user._id);
                    return res.redirect('/');
                });
            });
        })
    }catch (e) {
        console.log(e);
    }
});
```

Додавання нової книги до бази даних:

```
outer.post('/create', upload.single('img'), async (req, res) => {
    try {
        const product = new Product({
            img: req.file,
            title: req.body.title,
            description: req.body.description,
            price: req.body.price,
            author: req.body.author,
            category: req.body.category
        });

        await product.save();
        res.redirect('/')
    }catch (e) {
        console.log(e)
    }
});
```

Отримання існуючих товарі з бд:

```
router.get('/shop', async (req, res)=>{
    let products = await Product.find({});
    let pages = Math.ceil(products.length/12);
    let currPage = req.query.page || 1;
    let currUrl = '/shop?page=';
    if(pages > 1) {
        products = await Product.find({}).skip((req.query.page - 1) *
12).limit(12).lean();
    }
    let category = await Category.find({}).lean();
    res.render('shop', {
        title: 'shop',
        isShop: true,
        allPages: pages,
        recentPage: currPage,
        currUrl: currUrl,
        products,
        category
```

```
    })
});
```

Написання Коментарю:

```
router.post('/createComment', async (req, res) => {
    try {
        let user = await User.findById(req.cookies['userID']);
        const comment = new Comment({
            text: req.body.text,
            user: user._id,
            username: user.username,
            product: req.body.product
        });

        await comment.save();
        res.redirect('/item/'+ req.body.product);
    }catch (e) {
        console.log(e)
    }
});
```

Всі інші функції та файли серверу знаходяться у ‘Додатку А’

3.3 Верстка клієнтської частини

Перед тим як розпочати верстати сайт , треба продумати дизайн та структуру. Інтернет-магазин має містити такі розділи:

- Головна сторінка - має панель навігації , слайдер, інформацію про магазин та підвал
- Сторінки Авторизації та реєстрації - мають відповідні форми вводу інформації
- Сторінка магазину - повинна мати панель навігації з категоріями , поле пошуку , секцію нові книги, секцію топ книг , панель пагінації
- Сторінка Створення нового товару - має форму для додавання нової інформації про книгу до бази данних
- Корзина покупок - повинна містити додані до неї продукти
- Сторінка Книги містить додаткову інформацію та відгуки

Окремо виділимо шаблони які будуть присутні на всіх сторінках :

- Head – містить всю іформацію про сайт, мета теги, та посилання на потрібні бібліотеки.

  Файл head.hbs

```
<!doctype html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
          content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-
scale=1.0, minimum-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link href="https://fonts.googleapis.com/icon?family=Material+Icons"
rel="stylesheet">
    <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/materialize/1.0.0/css/materialize.min.cs
s">
    <link rel="stylesheet" href="/index.css">
    <title>{{title}}</title>
    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/materialize/1.0.0/js/materialize.min.js">
</script>
    <script
src="https://cdn.jsdelivr.net/npm/handlebars@latest/dist/handlebars.js"></script>
    <script>
        function getCookie(name) {
            var matches = document.cookie.match(new RegExp(
                    "(?:^|; )" + name.replace(/([\.$?*|{}\(\)\[\]\\\/\+^])/g, '\\$1')
+ "=([^;]*)"
            ));
            return matches ? decodeURIComponent(matches[1]) : undefined;
        }

        $(document).ready(function(){
            $('.carousel.carousel-slider').carousel({
                fullWidth: true,
                indicators: true
            });
        });

        $(document).ready(function(){
            $('select').formSelect();
        });

        $(document).ready(function(){
            $('.sidenav').sidenav();
        });
    </script>
</head>
```

- Nav – Панель навігації ресурсу

  Файл nav.hbs

```
<nav class="grey darken-3 nav-extended">
    <div class="container">
    <div class="nav-wrapper">
        <a href="/" class="brand-logo">Library</a>
        <a href="#" data-target="mobile-demo" class="sidenav-trigger"><i
class="material-icons">menu</i></a>
        <ul id="nav-mobile" class="right hide-on-med-and-down">

            {{#if isIndex}}
                <li class="active"><a href="/">Main</a></li>
            {{else}}
                <li><a href="/">Main</a></li>
            {{/if}}

            {{#if isShop}}
                <li class="active"><a href="/shop">Shop</a></li>
            {{else}}
                <li><a href="/shop">Shop</a></li>
            {{/if}}
            <script>
                var menu = document.getElementById('nav-mobile');
                if(!getCookie('userID')) {
                    menu.innerHTML += `
             {{#if isLogin}}
                <li class="active"><a href="/login">Login</a></li>
            {{else}}
                <li><a href="/login">Login</a></li>
            {{/if}}
                        {{#if isRegistration}}
                <li class="active"><a href="/registration">Registration</a></li>
            {{else}}
                <li><a href="/registration">Registration</a></li>
            {{/if}}`;
                }else {
                    menu.innerHTML += `{{#if isCreate}}
                                <li class="active"><a href="/create">Create</a></li>
                                        {{else}}
                                <li><a href="/create">Create</a></li>
                                        {{/if}}
                        <li><a href="/logout">Logout</a></li>
                        {{#if isBucket}}
                        <li class="active"><a href="/bucket"><i class="material-
icons">shopping_cart</i></a></li>
                        {{else}}
                        <li><a href="/bucket"><i class="material-
icons">shopping_cart</i></a></li>
                        {{/if}}
                    `

                }
            </script>
        </ul>
    </div>
    </div>
    {{#if isShop}}
        <div class="nav-content">
```

```
            <ul class="tabs tabs-transparent">
                {{#each category}}
                <li class="tab"><a
href="/shop/getItemsByCategory?id={{_id}}">{{title}}</a></li>
                {{/each}}
            </ul>
        </div>
    {{/if}}
</nav>

<ul class="sidenav grey darken-3" id="mobile-demo">
    <li><a href="/" class="white-text">Main</a></li>
    <li><a href="/shop" class="white-text">Shop</a></li>
    <li><a href="/login" class="white-text">Login</a></li>
    <li><a href="/registration" class="white-text">Registration</a></li>
</ul>
```

- Footer – Підвал сторінки , що містить другорядну інформацію та посилання

  Файл footer.hbs

```
<footer class="page-footer grey darken-4">
       <div class="container">
           <div class="row">
               <div class="col l6 s12">
                   <h5 class="white-text">Footer Content</h5>
                   <p class="grey-text text-lighten-4">You can use rows and columns
here to organize your footer content.</p>
               </div>
               <div class="col l4 offset-l2 s12">
                   <h5 class="white-text">Links</h5>
                   <ul>
                       <li><a class="grey-text text-lighten-3" href="#!">Link
1</a></li>
                       <li><a class="grey-text text-lighten-3" href="#!">Link
2</a></li>
                       <li><a class="grey-text text-lighten-3" href="#!">Link
3</a></li>
                       <li><a class="grey-text text-lighten-3" href="#!">Link
4</a></li>
                   </ul>
               </div>
           </div>
       </div>
       <div class="footer-copyright black darken-2">
           <div class="container">
               © 2014 Copyright Text
               <a class="grey-text text-lighten-4 right" href="#!">More Links</a>
           </div>
       </div>
   </footer>
```

Ці файли будуть знаходитись у папці views/partials, і будуть додані до кожної сторінки за допомогою heandlebars.

Головна сторінка повинна мати слайдер , інформацію про сайт та декілька секцій з товарами.

Файл index.hbs

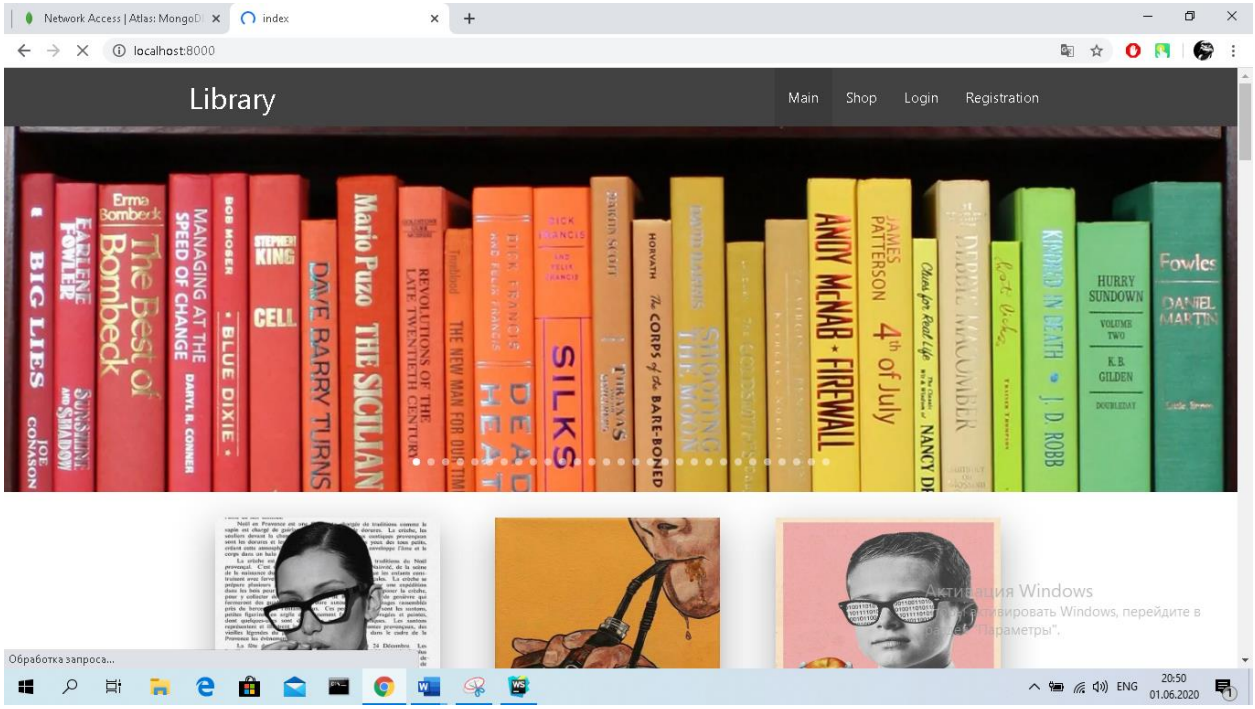
```
<div class="carousel carousel-slider fullsize-slider">
    {{#each products}}
        <a class="carousel-item" href="/item/{{_id}}"><img
src="https://www.washingtonpost.com/rf/image_1484w/2010-
2019/WashingtonPost/2015/08/04/LocalLiving/Images/WEBrainbow.jpg?t=20170517"></a>
    {{/each}}
 </div>
<div class="container">
{{#each products}}
    <div class="hoverable itemWrap z-depth-5">
        <img src="/upload/{{img.filename}}" class="img">
        <span class="">{{title}}</span>
        <form action="/item/{{_id}}" method="get">
            <input type="submit" value="more" class="btn">
            <input type="hidden" value="{{_id}}" name="id">
        </form>
    </div>
{{/each}}
</div>
```

Завдяки використанню шаблонів handlebars ми суттєво зменшуємо обсяги програмного коду, що дає змогу легко орієнтуватися та вносити зміни до вже існуючих сторінок. Код розмітки інших сторінок представлений у ‘Додатку А’

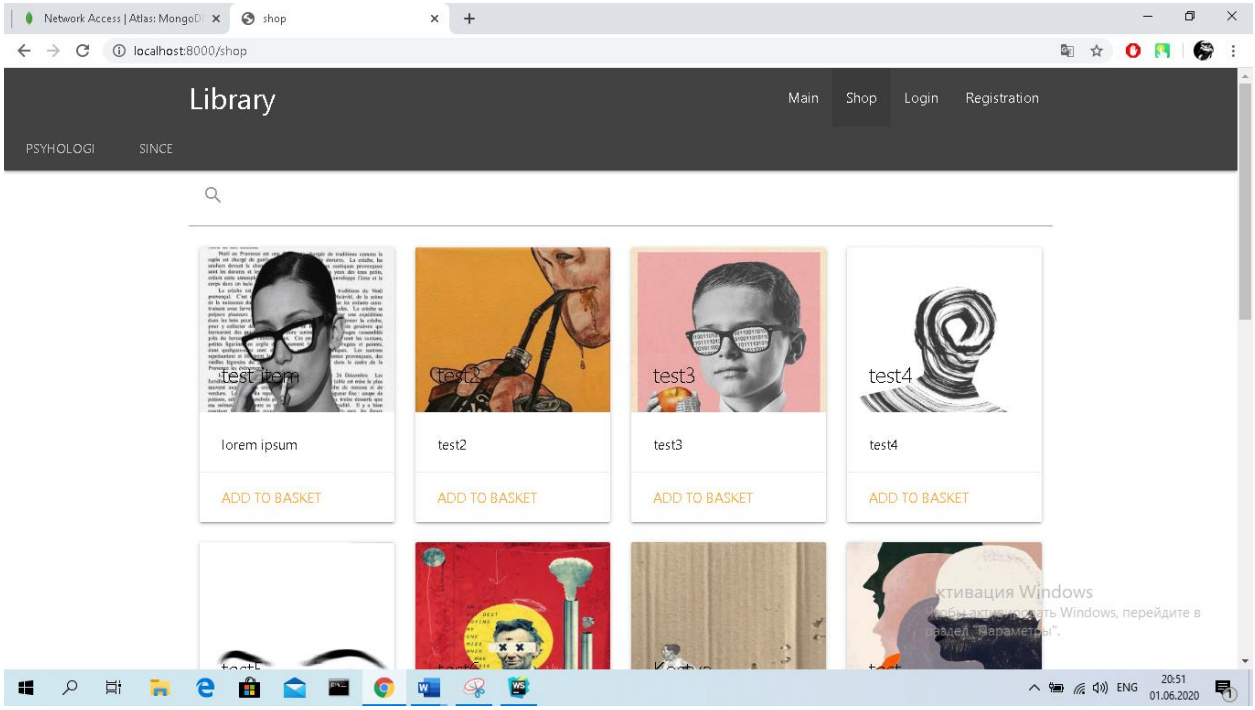
3.4 Тестування створеного ресурсу

Після закінчення реалізації дипломного проекту сайт було протестовано на наявність багів. В ході перевірки програмний продукт функціонував як це й було заплановано. Багів не виявлено , робота бази даних стабільна.

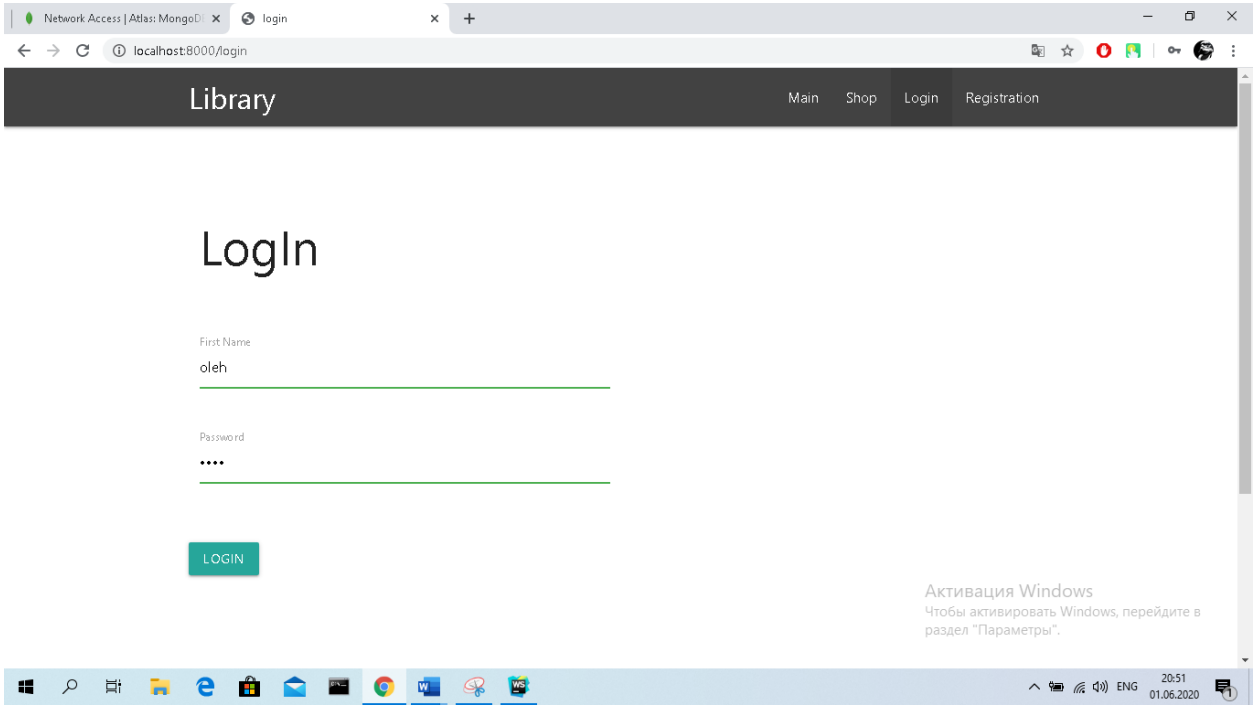
Далі на рис3.1, 3.2, 3.3, 3.4, 3.5 показано функціонування створеного інтернет магазину.



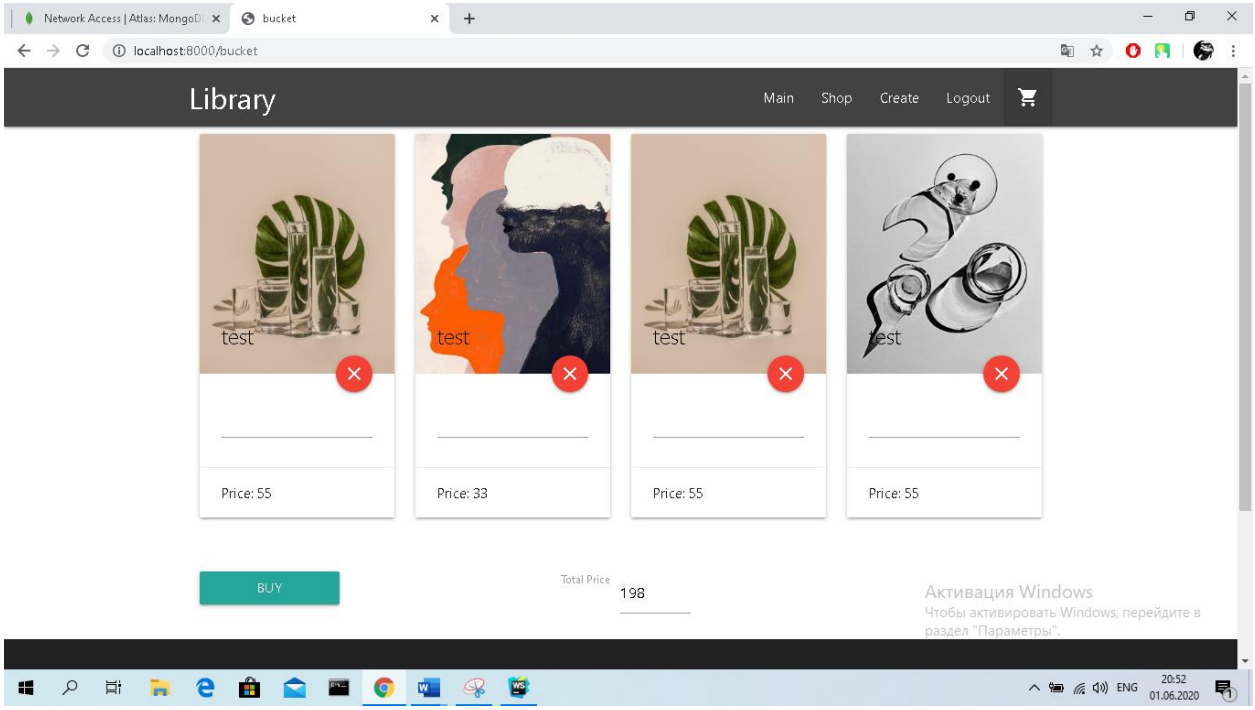
Малюнок 3.1 – головна сторінка



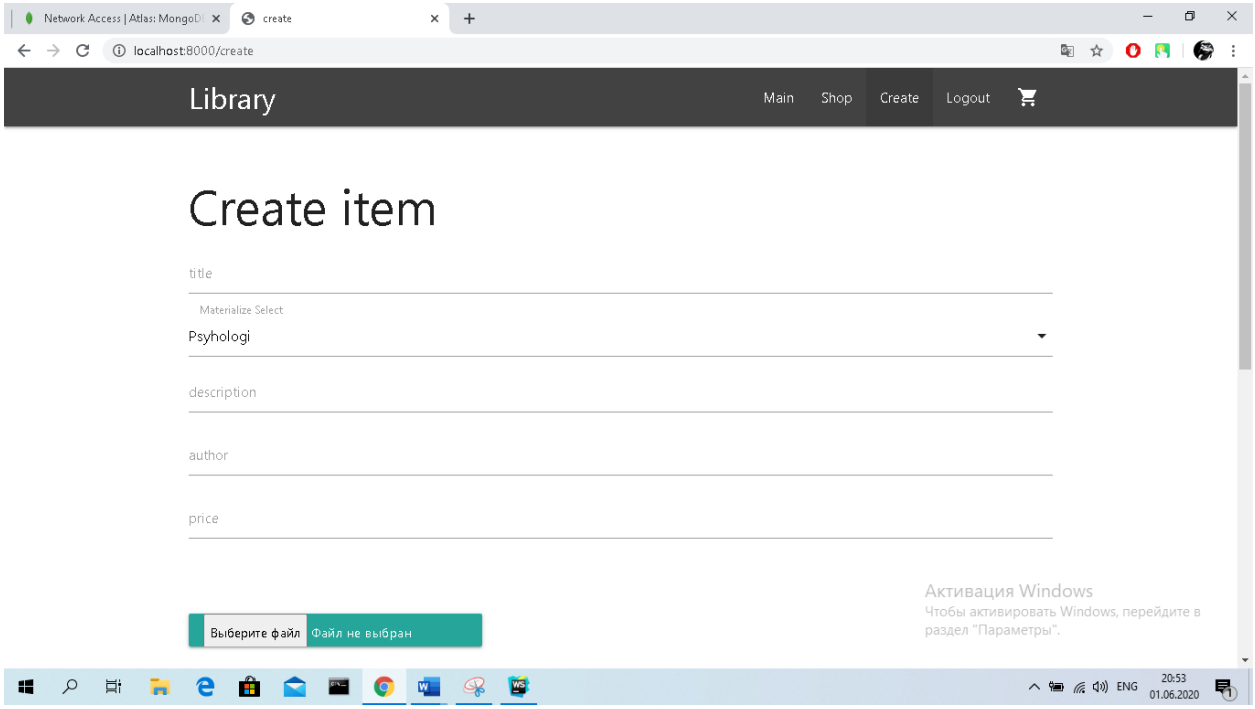
Малюнок 3.2 – сторінка магазину



Малюнок 3.3 – сторінка авторизації



Малюнок 3.4 – сторінка корзини користувача



Малюнок 3.5 – сторінка створення ного товару

ВИСНОВОК

В ході дипломної роботи було реалізовано книжковий інтернет-магазин з використанням фреймворку node js “express”. Інтернет ресурс відповідає поставленим цілям та критеріям , має адаптивну верстку, зрозумілий інтерфейс, сучасний дизайн, та всі вище зазначені функції.

Даний ресурс , можна розширювати та додавати нові функції, оскільки його архітектура є масштабованою.

СПИСОК ЛІТЕРАТУРИ

1. Булгакова, М.В. Информационные технологии в экономике: учебное пособие / М. В. Булгакова. Челябинск: Изд -во Челяб. гос. ун-та, 2013. 106 с. (Классическое университетское образование).
2. Крамер Э. HTML: наглядный курс web-дизайна Диалектика. — 2001. — 304 с.
3. JavaScript: The Definitive Guide S.Spainhour, R.Eckstein Webmaster in a Nutshell. 2nd Edition O'Reilly. — 1999. — 572 с.
4. Муссиано Ч., Кеннеди Б. HTML и XHTML. Подробное руководство. Символ-Плюс. — 2011. — 752 с.
5. Коржинский С.Н. Настольная книга Web-мастера: эффективное применение HTML, CSS и JavaScript. КноРус. — 2000. — 320 с.
6. David Flanagan. JavaScript: The Definitive Guide 3rd Edition. O'Reilly & Associates. — 1998. — 792 с.
7. Ташков П.А.: Веб-мастеринг: HTML, CSS, JavaScript, PHP, CMS, графика, раскрутка. СПб.: Питер. — 2009. — 512 с.
8. Круг С. Как сделать сайт удобным. Целесообразность по методу Стива Круга — СПб.: Питер. 2010. — 170 с.

ДОДАТКИ

Додаток А

Сторінка Магазину

```
<div class="container">
<form action="/search" method="get">
    <div class="input-field">
        <input id="search" type="search" name="search" required>
        <label class="label-icon" for="search"><i class="material-
icons">search</i></label>
        <i class="material-icons">close</i>
    </div>
</form>
</div>
<div class="container">
<div class="row">
    {{#if products.length}}
    {{#each products}}
        <div class="col s12 m6 l3">
        <div class="card small z-depth-1">
            <a href="/item/{{_id}}">
            <div class="card-image hoverable">
                <img src="/upload/{{img.filename}}" class="img">
                <span class="card-title black-text flow-text">{{title}}</span>
            </div>
            </a>
            <div class="card-content">
                <p>{{description}}</p>
            </div>
            <div class="card-action"><a href="/addToBucket/{{_id}}"
onclick="M.toast({html: '<span>{{title}} add to bucket<span> <a
href=\'/deleteFromBucket/{{_id}} \' class=\'btn\'>Undo</a>'})">Add to
basket</a></div>
        </div>
        </div>
    {{/each}}
{{else}}
    <p>No products</p>
{{/if}}
</div>
</div>
<div class="container pagWrapper">
    <ul class="pagination" id="pagination">
        <script>
            function getPaginator() {
                const paginator = document.getElementById('pagination');
                if({{allPages}} > 1){
                    if ({{recentPage}} === 1)
                    {
                        paginator.innerHTML += `<li class="disabled"><a href=""><i
class="material-icons">chevron_left</i></a></li>`;
                    }else{
                        paginator.innerHTML += `<li class="waves-effect"><a
href="{{currUrl}}` + ({{recentPage}} -1) + `"><i class="material-
icons">chevron_left</i></a></li>`;
                    }
```

```
                    for (let i = 1; i <= {{allPages}};i++){
                        if ({{recentPage}} === i){
                            paginator.innerHTML += `<li class="active"><a
href="{{currUrl}}` + i + `">` + i + `</a></li>`;
                        }else{
                            paginator.innerHTML += `<li class="waves-effect"><a
href="{{currUrl}}` + i + `">` + i + `</a></li>`;
                        }
                    }
                    if ({{recentPage}} === {{allPages}}) {
                        paginator.innerHTML += `<li class="disabled"><a href=""><i
class="material-icons">chevron_right</i></a></li>`;
                    }else {
                        paginator.innerHTML += `<li class="waves-effect"><a
href="{{currUrl}}` + ({{recentPage}} + 1) + `"><i class="material-
icons">chevron_right</i></a></li>`;
                    }
                }
            }

            getPaginator();
        </script>
    </ul>
</div>
```

Сторінка Кошика користувача

```
<div class="container">
    <div class="row">
        {{#if items.length}}
            <script>
                let sum = 0;
            </script>
            {{#each items}}
                <div class="col s12 m6 l3">
                    <div class="card">
                        <div class="card-image">
                            <img src="/upload/{{img.filename}}" class="img">
                            <span class="card-title black-text">{{title}}</span>
                            <a href="/deleteFromBucket/{{_id}}" class="btn-floating
halfway-fab waves-effect waves-light red hoverable"><i class="material-icons
small">close</i></a>
                        </div>
                        <div class="card-content"><input type="number"></div>
                        <div class="card-action">
                            <span>Price: {{price}}</span>
                        </div>
                    </div>
                </div>
                <script>
                    sum += {{price}};
                </script>
            {{/each}}
        <form class="col s12">
            <br><br>
            <input type="submit" class="btn col s2" value="BUY">
            <label for="totalPrice" class="col s3 l1 offset-s3 offset-l3">Total
```

```
Price</label>
            <input id="totalPrice" type="number" class="col s1">
            <script>
                console.log(sum);
                const totalPrice = document.getElementById('totalPrice');
                totalPrice.value = sum;
            </script>
        </form>
        <br><br><br>
    {{else}}
        <p>No items in bucket</p>
    {{/if}}
    </div>
</div>
```

‘Додаток Б’

Маршрути для користувачів

```
const { Router } = require('express');
const User = require('../models/User');
const Bucket = require('../models/Buket');
const Product = require('../models/Product');
const connectEnsureLogin = require('connect-ensure-login');
const passport = require('passport');
const router = Router();

router.get('/registration', (req, res)=>{
   res.render('registration', {
        title: 'registration',
        isRegistration: true,
    });
});

router.get('/login', (req, res) => {
    res.render('login', {
        title: 'login',
        isLogin: true
    })
});

//logout router
router.get('/logout', (req, res) => {
    req.logout();
    res.clearCookie('userID');
    return res.redirect('/');
});

router.get('/bucket' , async (req, res) => {
    try {
        const user = await User.findById(req.cookies['userID']);
        const bucket = await Bucket.find({user: user._id});
        let items = [];
        if(bucket[0]) {
            for (let i = 0; i <= bucket[0].products.length - 1; i++) {
                items.push(await Product.findById(bucket[0].products[i]).lean());
            }
        }
            res.render('bucket',{
            title: 'bucket',
            isBucket: true,
            items
        })
    }catch (e) {
        console.log(e);
    }
});

router.get('/addToBucket/:id', async (req, res) => {
    try{
        const user = await User.findById(req.cookies['userID']);
        const products = await Product.findById(req.params.id);
        try{
```

```
            await Bucket.updateOne({user: user}, {$push:{products: products}},{
upsert: true });
        }catch (e){
            console.log(e);
        }
    }catch(e){
        console.log(e);
    }
});

router.get('/deleteFromBucket/:id', async (req,res) => {
    try{
        await Bucket.updateOne({user: req.cookies['userID']}, {$pull: {products:
req.params.id}});
    }catch (e) {
        console.log(e);
    }
    res.redirect('/bucket')
});

//registration router
router.post('/registration', async (req, res) => {
    try {
        const user = new User({username: req.body.username,
                               mail: req.body.mail});
        User.register(user, req.body.password, function(err, user){
            if(err){
                console.log('error registering user');
                console.log(err);
                return res.send('oops');
            }
            passport.authenticate('local')(req, res, function(){
                req.logIn(user, function(err) {
                    if (err) {
                        console.log(err);
                    }
                    res.cookie('userID', user._id);
                    return res.redirect('/');
                });
            });
        })
    }catch (e) {
        console.log(e);
    }
});

//login router
router.post('/login', (req, res, next) => {
    passport.authenticate('local',
        (err, user, info) => {
            if (err) {
                return next(err);
            }

            if (!user) {
                return res.redirect('/login?info=' + info);
            }

            req.logIn(user, function(err) {
                if (err) {
```

```
                    return next(err);
                }
                res.cookie('userID', user._id);
                return res.redirect('/');
            });

        }, {})(req, res, next)
});



module.exports = router;
```

Routes/products.js

```
const { Router } = require('express');
const Product = require('../models/Product');
const Comment = require('../models/Comment');
const User = require('../models/User');
const Category = require('../models/Category');
const  multer   =  require ( 'multer' );
const upload  =  multer ( {  dest : 'public/upload/'  } );
const router = Router();

//take all products
router.get('/', async (req, res)=>{
    const products = await Product.find({}).lean();
    res.render('index', {
        title: 'index',
        isIndex: true,
        products
    })
});

router.get('/shop', async (req, res)=>{
    let products = await Product.find({});
    let pages = Math.ceil(products.length/12);
    let currPage = req.query.page || 1;
    let currUrl = '/shop?page=';
    if(pages > 1) {
        products = await Product.find({}).skip((req.query.page - 1) *
12).limit(12).lean();
    }
    let category = await Category.find({}).lean();
    res.render('shop', {
        title: 'shop',
        isShop: true,
        allPages: pages,
        recentPage: currPage,
        currUrl: currUrl,
        products,
        category
    })
});

router.get('/shop/getItemsByCategory', async (req, res)=> {
    try {
```

```
        let products = await Product.find({category: req.query.id}).lean();
        let category = await Category.find({}).lean();
        let pages = Math.ceil(products.length / 12);
        let currPage = req.query.page || 1;
        let currUrl = '/shop/getItemsByCategory?id=' + req.query.id + '&page=';
        if (pages > 1) {
            products = await Product.find({category:
req.query.id}).skip((req.query.page - 1) * 12).limit(12).lean()
        }

        res.render('shop',{
            isShop: true,
            allPages: pages,
            currUrl: currUrl,
            recentPage: currPage,
            products,
            category
        });
    }catch (e) {
      console.log(e)
    }
});

router.get('/create', async (req, res) => {
    let allCategories = await Category.find({}).lean();
    res.render('create', {
        title: 'create',
        isCreate: true,
        allCategories
    })
});

router.get('/item/:id', upload.none(), async (req, res) => {
    try {
        const item = await Product.findById(req.params.id);
        const comments = await Comment.find({product: req.params.id}).lean();
        res.render('item', {
            itemTitle: item.title,
            itemAuthor: item.author,
            itemPrice: item.price,
            itemDescription: item.description,
            itemImg: item.img.filename,
            itemID: req.params.id,
            comments
        })
    }catch (e) {
        console.log(e)
    }
});

router.post('/createComment', async (req, res) => {
    try {
        let user = await User.findById(req.cookies['userID']);
        const comment = new Comment({
            text: req.body.text,
            user: user._id,
            username: user.username,
            product: req.body.product
        });
```

```
        await comment.save();
        res.redirect('/item/'+ req.body.product);
    }catch (e) {
        console.log(e)
    }
});

router.post('/create', upload.single('img'), async (req, res) => {
    try {
        const product = new Product({
            img: req.file,
            title: req.body.title,
            description: req.body.description,
            price: req.body.price,
            author: req.body.author,
            category: req.body.category
        });

        await product.save();
        res.redirect('/')
    }catch (e) {
        console.log(e)
    }
});

router.get('/search', async (req,res)=>{
    try {
        let products = await Product.find({title: {$regex: req.query.search,
$options:"i"}}).lean();
        let category = await Category.find({}).lean();
        let pages = Math.ceil(products.length / 12);
        let currPage = req.query.page || 1;
        let currUrl = '/search?search=' + req.query.search + '&page=';
        if (pages > 1) {
            try {
                products = await Product.find({
                    title: {
                        $regex: req.query.search,
                        $options: "i"
                    }
                }).skip((req.query.page - 1) * 12).limit(12).lean()
            }catch (e) {
                console.log(e);
            }
        }

        res.render('shop',{
            isShop: true,
            allPages: pages,
            recentPage: currPage,
            currUrl: currUrl,
            products,
            category
        });
    }catch (e) {
        console.log(e)
    }
});

module.exports = router;
```