МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проєктами

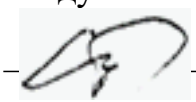
Кафедра програмного забезпечення автоматизованих систем

Спеціальність 121 «Інженерія програмного забезпечення»

Освітня програма «Інженерія програмного забезпечення»

«Допущений до захисту»

Завідувач кафедри

_  __ проф. Приходько С.Б.

«___» _____ 2025 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти «бакалавр»

на тему: **Розробка вебзастосунку для автоматизації обліку прокату туристичного спорядження у магазині «Хан-Тенгрі»**

Виконала: студентка групи 4151

 Філіна А. С.

(підпис, ПІБ)

Керівник роботи:

доцент кафедри ПЗАС, к.т.н., доцент

(посада, науковий ступень вчене звання)

 Макарова Л. М

Миколаїв – 2025 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проектами
Кафедра програмного забезпечення автоматизованих систем
Спеціальність 121 «Інженерія програмного забезпечення»
Освітня програма «Інженерія програмного забезпечення»

«ЗАТВЕРДЖУЮ»
Гарант освітньої програми
_____ доц. Макарова Л.М.
«_08_» ___04___ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ
на здобуття ступеня вищої освіти «бакалавр»

Студенту Філіної Анастасії Сергіївни
(Прізвище, ім'я, по батькові)

1. Тема роботи: Розробка вебзастосунку для автоматизації обліку прокату туристичного спорядження у магазині «Хан-Тенгрі»
Керівник роботи доцент кафедри ПЗАС, к.т.н., доцент Макарова Л. М
Затверджені наказом ректора №153 від 08.04.2025 р.
2. Термін подання роботи: 02.06.2025 р.
3. Вихідні дані по роботі:
4. Перелік питань, що належать до розробки (найменування розділів)
Титульний аркуш; Завдання на кваліфікаційну роботу; Анотація (українською та англійською мовами); Зміст; Перелік умовних позначень, символів, одиниць та термінів (при необхідності); Вступ; Аналіз практичної задачі інженерії програмного забезпечення; Проєкт програмного забезпечення; Результати розробки програмного забезпечення; Розділ з охорони праці; Висновки; Список використаних джерел; Додатки (технічне завдання, текст програми, опис програми, інструкція користувача, програма та методика випробувань програмного забезпечення).
5. Перелік презентаційних матеріалів
Титульний слайд, Мета та завдання кваліфікаційної роботи, Аналіз вимог до ПЗ, Діаграма варіантів використання, Архітектура програмного забезпечення, Концептуальна модель даних, Діаграма компонентів та їх взаємодії на клієнтському рівні, Діаграми взаємодії, Логічна модель бази даних, Технічний проект ПЗ, Результати розробки, Висновки, Заключний слайд.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
4	Гурець Н.В., ст. викладач		

7. Дата видачі завдання 08.04.2025 р.

КАЛЕНДАРНИЙ ПЛАН

Номер	Назва етапів роботи	Терміни виконання	Примітка
1.	Підготовка розділу ВСТУП	31.03.2025	ПП*
2.	Аналіз практичної задачі інженерії ПЗ	07.04.2025	ПП*
3.	Аналіз вимог до ПЗ	14.04.2025	ПП*
4.	Розробка архітектури ПЗ	21.04.2025	
5.	Розробка проекту ПЗ	05.05.2025	
6.	Реалізація та тестування ПЗ	12.05.2025	
7.	Підготовка розділу Результати розробки ПЗ	16.05.2025	
8.	Підготовка розділу з охорони праці	19.05.2025	ПП*
9.	Підготовка розділу ВИСНОВКИ	23.05.2025	
10.	Оформлення списку використаних джерел та додатків	26.05.2025	
11.	Подання на кафедру ПЗАС тексту остаточного варіанту роботи, підписаного її керівником, у роздрукованому та електронному форматі разом із заявами щодо самостійності виконання роботи та ідентичності друкованої та електронної версій роботи (Додатки 1 і 2 «Порядку здійснення заходів з перевірки робіт на наявність текстових збігів/ідентичності/схожості із використанням програмно-технічних засобів», який введений в дію наказом ректора НУК за №20 від 20.01.2020 р.)	02.06.2025	не пізніше, ніж за 14 діб до захисту (згідно п.4.1 зазначеного Порядку
12.	Підготовка презентації та доповіді	06.06.2025	
13.	Попередній захист роботи на засіданні кафедри ПЗАС	09.06.2025	
14.	Подання на кафедру ПЗАС електронних версій наступних документів у форматі pdf: кваліфікаційної роботи; файлу-опису кваліфікаційної роботи (згідно Додатку до наказу ректора НУК за №287-уч від 19.05.2020 р.); презентації доповіді, а також Авторського договору з додатком	16.06.2025	

* - за результатами переддипломної практики (ПП), яка була з 03.02.2025 до 23.03.2025 р.

Студент _____ Філіна А.С,
(підпис) (ПІБ)

Керівник роботи _____ Макарова Л.М.
(підпис) (ПІБ)

АНОТАЦІЯ

Кваліфікаційна робота присвячена розробці вебзастосунку для автоматизації обліку прокату туристичного спорядження у магазині «Хан-Тенгрі», що відноситься до практичної задачі інженерії програмного забезпечення з комплексністю та невизначеністю умов.

У процесі виконання кваліфікаційної роботи було проведено аналіз практичної задачі інженерії програмного забезпечення та розглянуто існуючі програмні продукти та методи розробки, що стосуються відповідної проблематики. Викладені результати аналізу спрямовані на підтримку розробки вебзастосунку для автоматизації обліку прокату туристичного спорядження у магазині «Хан-Тенгрі». Об’єктом даної роботи є процес розробки вебзастосунку для автоматизації обліку прокату туристичного спорядження у магазині «Хан-Тенгрі».

Кваліфікаційна робота викладена на 143 сторінках друкованого тексту, містить 31 рисунок, список використаних джерел з 22 найменувань та 6 додатків.

ABSTRACT

The qualification paper is dedicated to the development of a web application for automating the rental accounting of tourist equipment in the "Han-Tengry" store, which relates to the practical problem of software engineering under conditions of complexity and uncertainty.

In the course of this qualification paper, an analysis of this practical software engineering problem was conducted, and existing software products and development methods relevant to the issue were examined. The analysis results presented are aimed at supporting the development of a web application for automating the rental accounting of tourist equipment in the "Han-Tengry" store. The object of this work is the process of developing a web application for automating the rental accounting of tourist equipment in the "Han-Tengry" store.

The qualification paper spans 143 pages of printed text, includes 31 figures, a list of references with 22 entries, and 6 appendixes.

ЗМІСТ

ВСТУП

У сучасних умовах швидкого розвитку інформаційних технологій все складніше уявити підприємства, які не застосовують сучасні засоби автоматизації. Це стосується і сфери торгівлі, де програмне забезпечення допомагає знизити ймовірність помилок, підвищити продуктивність праці й оптимізувати роботу з клієнтами та товаром. Зокрема, магазини, що займаються прокатом туристичного спорядження, постійно стикаються з потребою швидко обробляти замовлення, вести облік клієнтів і товарів, а також контролювати фінансові операції.

Туристична галузь в Україні нині зіштовхується зі значними викликами, пов’язаними з воєнним станом, однак для окремих підприємців вона досі лишається напрямом, у якому продовжують здійснювати діяльність, зокрема й у сфері прокату туристичного спорядження [1]. У таких умовах ефективне управління процесом прокату стає особливо важливим: потрібно швидко здійснювати пошук та резервування наявного обладнання, фіксувати всі операції та контролювати повернення товарів у встановлені терміни. Для малого та середнього бізнесу, який працює в цій сфері, наявність сучасного інструменту для автоматизації обліку прокату може допомогти оптимізувати робочі процеси й утримуватися на ринку попри складну ситуацію.

Саме тому розробка вебзастосунку для автоматизації обліку прокату туристичного спорядження в магазині «Хан-Тенгрі» є актуальною задачею. Такий вебзастосунок повинен значно спростити управління процесом прокату: від реєстрації нових клієнтів і формування замовлень до аналізу даних про стан складу та оборот товарів. Крім того, його використання дасть змогу уникнути трудомісткої ручної роботи, зменшити кількість помилок під час оформлення прокату й пришвидшити обслуговування клієнтів. У результаті підвищується

ефективність ведення бізнесу та конкурентоспроможність магазину, що нині є одним із ключових чинників успіху.

Дана кваліфікаційна робота присвячена розв'язанню практичної задачі інженерії програмного забезпечення, що характеризується комплексністю та невизначеністю умов, а саме розробці вебзастосунку для туристичного магазину «Хан-Тенгрі». Метою роботи є створення вебзастосунку для автоматизації обліку прокату туристичного спорядження, що дозволить підвищити зручність і ефективність роботи менеджерів, відповідальних за його організацію та управління, а також зменшити кількість помилок у процесі обліку та видачі спорядження. Для досягнення цієї мети необхідно провести аналіз практичної задачі інженерії програмного забезпечення, а саме дослідити роботу магазину туристичного спорядження «Хан-Тенгрі» з урахуванням уже наявних систем автоматизації в магазині та проаналізувати сучасні тенденції автоматизації прокату туристичного обладнання. Окремої уваги потребує обґрунтування доцільності розробки власного програмного забезпечення для автоматизації повсякденних операцій адміністратора та менеджера магазину.

Програмне забезпечення, що розробляється, призначене для спрощення роботи менеджера магазину «Хан-Тенгрі» та полегшення ведення обліку прокату туристичного спорядження.

Для досягнення поставленої мети необхідно:

– проаналізувати роботу магазину «Хан-Тенгрі», зокрема сучасні тенденції автоматизації процесів прокату туристичного спорядження;

– зробити висновки щодо доцільності розробки власного програмного забезпечення для автоматизації обліку прокату туристичного спорядження магазину «Хан-Тенгрі»;

– проаналізувати існуючі аналоги з метою визначення їх основного функціоналу, виявлення переваг і недоліків, а також порівняння з потенційними можливостями власної розробки;

– сформулювати постановку задачі на розробку програмного забезпечення для автоматизації обліку прокату туристичного спорядження;

– провести аналіз вимог до програмного забезпечення для автоматизації обліку прокату туристичного спорядження магазину «Хан-Тенгрі»;

– розробити проєкт програмного забезпечення, який включає всі технічні та графічні аспекти майбутнього рішення;

– здійснити реалізацію та тестування програмного забезпечення для автоматизації обліку прокату туристичного спорядження магазину «Хан-Тенгрі»;

– розробити всю необхідну програмну документацію.

Об’єктом кваліфікаційної роботи є процес розробки вебзастосунку для автоматизації обліку прокату туристичного спорядження у магазині «Хан-Тенгрі».

1 АНАЛІЗ ПРАКТИЧНОЇ ЗАДАЧІ ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТА ПОСТАНОВКА ЗАДАЧІ НА РОЗРОБКУ ВЕБЗАСТОСУГКУ ДЛЯ АВТОМАТИЗАЦІЇ ОБЛІКУ ПРОКАТУ ТУРИСТИЧНОГО СПОРЯДЖЕННЯ МАГАЗИНУ «ХАН-ТЕНГРІ»

1.1 Аналіз практичної задачі інженерії вебзастосунку для автоматизації обліку прокату туристичного спорядження магазину «Хан-тенгрі»

Магазин «Хан-Тенгрі» функціонує на ринку туристичного спорядження з 2005 року, пропонуючи широкий спектр товарів та послуг для активного відпочинку, включно з гірським і пішохідним туризмом, альпінізмом, промисловим альпінізмом (промальп) тощо. Основна діяльність магазину охоплює:

– Продаж туристичного спорядження. У «Хан-Тенгрі» представлено продукцію як вітчизняних, так і міжнародних брендів, що забезпечує широкий вибір і високу якість товарів. Асортимент включає як основне спорядження (намети, рюкзаки, спальні мішки), так і додаткові аксесуари (ліхтарики, карабіни, газові пальники тощо).

– Консультації та підтримка клієнтів. Кваліфіковані працівники надають рекомендації щодо вибору оптимального спорядження, зважаючи на тип подорожі, умови використання та досвід клієнта. Такий підхід допомагає мінімізувати ризики під час активного відпочинку й підвищує рівень задоволеності покупців.

– Оренда туристичного спорядження. Для клієнтів, які подорожують нерегулярно або бажають випробувати обладнання перед покупкою, магазин надає послуги прокату. Це дозволяє уникнути зайвих витрат на придбання

спорядження, яке може використовуватися нечасто. У прокаті доступні намети, рюкзаки, спальні мішки та інші позиції.

– Інтернет-магазин. Для зручності покупців «Хан-Тенгрі» має власну онлайн-платформу (рис. 1.1), через яку можна переглянути асортимент, отримати консультації й оформити замовлення незалежно від місцезнаходження клієнта [2].

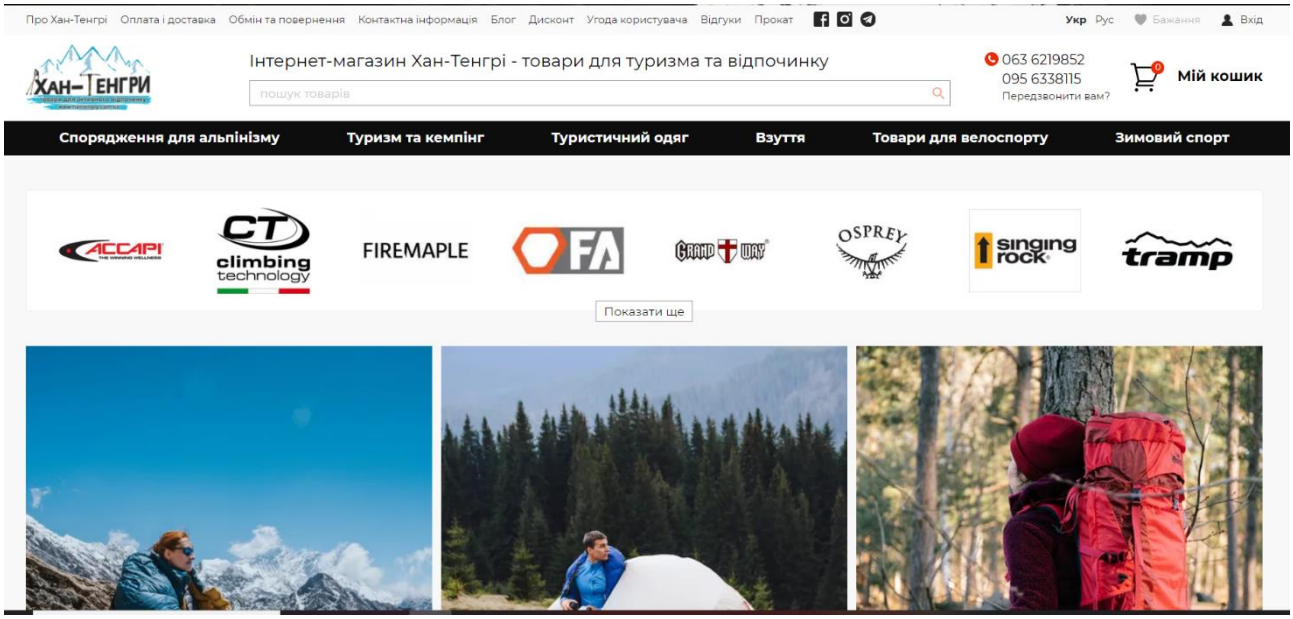


Рисунок 1.1 – Головна сторінка онлайн магазину «Хан-Тенгрі»

Крім здійснення продажу й оренди товарів, магазин «Хан-Тенгрі» активно співпрацює з дитячими туристичними гуртками та клубами, що підкреслює його соціальну орієнтованість і сприяє розширенню кола потенційних клієнтів. Широкий асортимент туристичного спорядження, що включає одяг, взуття, альпіністські приладдя, кемпінгові товари та обладнання для висотних робіт, формує значний обсяг інформації, яку необхідно постійно оновлювати та систематизувати.

Без спеціалізованого програмного забезпечення менеджери змушені вручну вести облік орендних операцій, реєструвати нові замовлення, контролювати строки прокату та фіксувати фінансові розрахунки, такі як вартість

оренди, оплата. Також здійснюється ручне оновлення даних про клієнтів, що ускладнює відстеження історії оренди та своєчасну реакцію на їх запити, а постійна перевірка стану товарів ускладнює планування та обробку замовлень, що в результаті підвищує ризик помилок і затримок.

Внаслідок таких недоліків якість обслуговування клієнтів страждає, а загальна ефективність бізнес-процесів знижується, адже відсутність автоматизації ускладнює контроль термінів повернення спорядження, стану обладнання та своєчасне фіксування оплат чи заборгованості. З огляду на ці фактори, виникає потреба у розробленні спеціалізованого програмного забезпечення для автоматизації обліку прокату туристичного спорядження. Запропоноване рішення дозволить зменшити обсяг ручних операцій, мінімізувати ймовірність помилок, скоротити час обробки замовлень та підвищити швидкість обслуговування клієнтів, а також забезпечить централізоване зберігання і оперативне оновлення даних про товари, клієнтів і фінансові операції.

Отже, врахувавши всі вимоги, рішенням практичної задачі інженерії програмного забезпечення буде побудова односторінкового вебзастосунку (SPA) з чітким розділенням на front-end і back-end частини. SPA забезпечує високу швидкість роботи та кращий користувацький досвід завдяки кешуванню та динамічному підвантаженню даних. Для вирішення практичної задачі було прийняте рішення використовувати такі технології:

- Front-end (React.js): відповідає за інтерфейс користувача, відображення даних та взаємодію з ними. Реалізований на React.js із використанням компонентного підходу й управлінням станом через Context API.
- Back-end (Node.js + Express): обробляє HTTP-запити від клієнта, реалізує RESTful API для управління даними (оренда, товари, клієнти, користувачі тощо), проводить валідацію та автентифікацію.

- База даних (MySQL): зберігає інформацію про товари, клієнтів, замовлення, користувачів та історію прокатів. Для доступу до БД використовується ORM Sequelize або безпосередні SQL-запити через mysql2.

Для розгортання вебзастосунку буде використовуватися хостинг із встановленою панеллю керування FastPanel.

Впровадження такої архітектури й стеку технологій забезпечить масштабованість, безпеку й швидку розробку.

Таким чином, аналіз практичної задачі інженерії програмного забезпечення для магазину «Хан-Тенгрі» свідчить про доцільність розробки та впровадження вебзастосунку, який автоматизує ключові процеси прокату туристичного спорядження, що відповідає сучасним вимогам ринку та сприяє підвищенню конкурентоспроможності магазину.

1.2 Аналіз існуючих програмних рішень

Розглянемо вже існуючі програмні рішення.

Одне з таких – програма Remonline. Це комплексне рішення для автоматизації процесів оренди, яке дозволяє вести облік обладнання, управляти базою клієнтів, контролювати строки оренди, проводити фінансові розрахунки та формувати звітність [3].

Головна сторінка програми Remonline зображена на рисунку 1.2

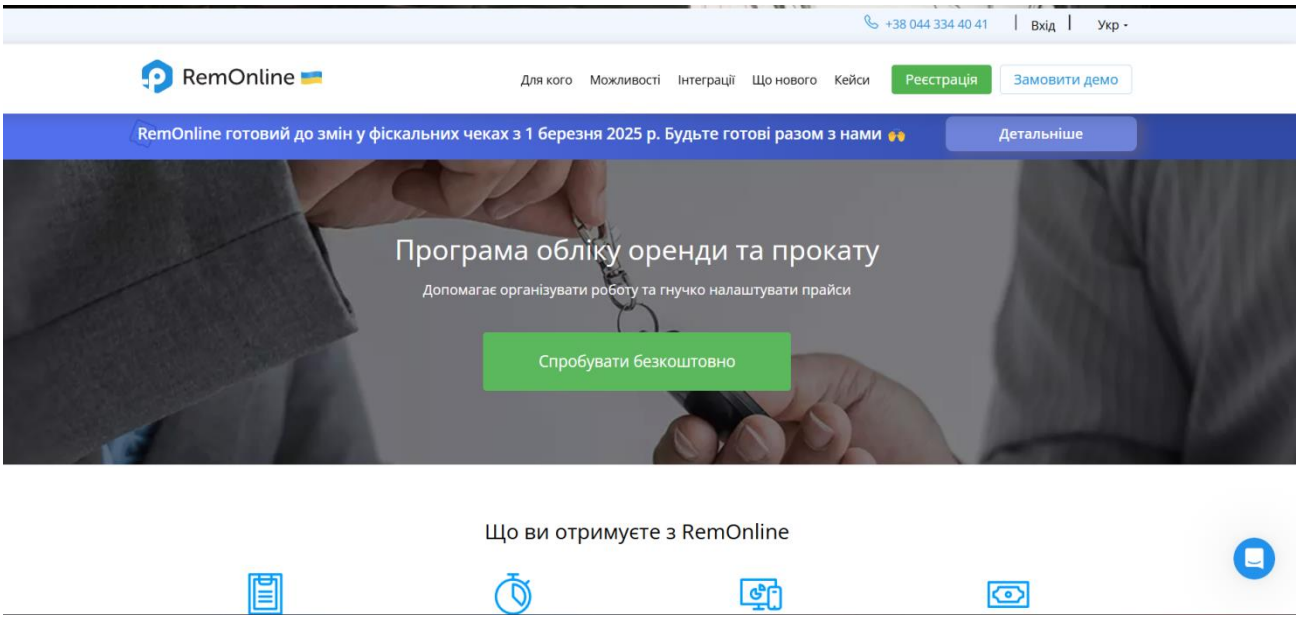


Рисунок 1.2 – Головна сторінка програми Remonline

Переваги:

– комплексне рішення для автоматизації процесів оренди;

– модульність вебдодатку, що дозволяє адаптувати окремі функції під потреби бізнесу;

– зручний та інтуїтивно зрозумілий інтерфейс для користувачів;

– можливість інтеграції з іншими програмними продуктами;

– автоматизація основних процесів оренди, таких як облік замовлень, розрахунок вартості та формування звітності.

Недоліки:

– універсальний характер програми, що не враховує специфіку роботи з туристичним спорядженням;

– обмежені можливості для гнучкого налаштування тарифікації залежно від категорій товарів;

– недостатня адаптація для роботи з особливими сегментами клієнтів, зокрема дитячими туристичними гуртками та клубами;

– наявність лише тестового безкоштовного періоду, що може обмежувати можливості опробування продукту перед покупкою.

Таким чином, хоча Remonline демонструє високий рівень автоматизації загальних процесів оренди, її універсальний характер і відсутність спеціалізованих функцій для роботи з туристичним спорядженням роблять програму менш придатною для магазина «Хан-Тенгрі». Для цього бізнесу потрібне рішення, яке зможе врахувати специфічні особливості асортименту товарів, гнучко налаштовувати тарифікацію оренди і адаптуватися під потреби роботи з різними категоріями клієнтів.

Ще один аналог програмного забезпечення для обліку оренди – «Універсальна Система Обліку» від AS-Service. Вона є програмним продуктом, орієнтованим на автоматизацію та оптимізацію роботи пунктів оренди й прокату. Завдяки широкому функціоналу, система дозволяє контролювати наявність та рух товарів, вести облік клієнтів і фінансових операцій, а також формувати аналітичну звітність для прийняття управлінських рішень [4].

Головна сторінка програми «Універсальна Система Обліку» від AS-Service зображена на рисунку 1.3

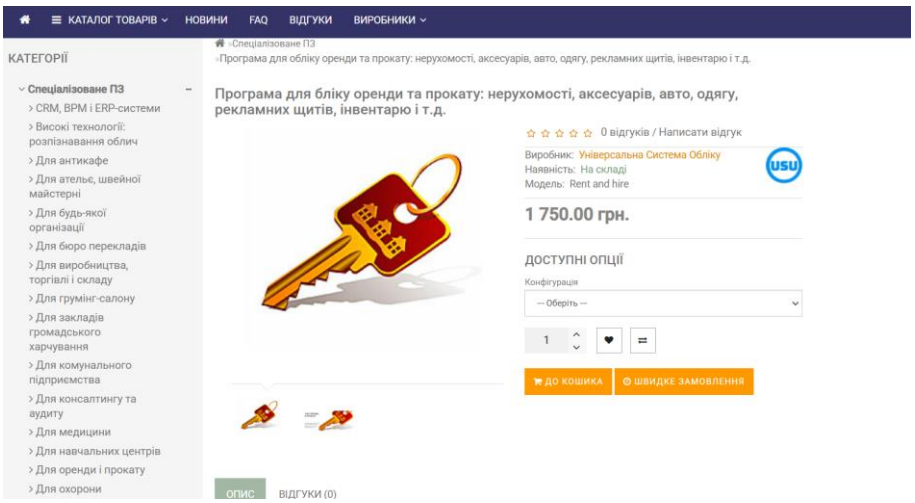


Рисунок 1.3 – Головна сторінка програми «Універсальна Система Обліку» від AS-Service

Переваги:

– забезпечує комплексну автоматизацію обліку оренди та прокату, що полегшує контроль за фінансовими операціями й строками оренди;

– інтуїтивно зрозумілий інтерфейс, який допомагає швидко навчити персонал роботі в системі;

– модульна архітектура дає змогу розширювати функціонал або відключати непотрібні елементи, що є важливим для масштабування;

– наявність інструментів для формування звітності, зокрема щодо доходів, залишків товарів і строків прокату.

Недоліки:

– програма орієнтована на широкий спектр орендних бізнесів, що може призводити до надмірної універсальності та появи непотрібних для магазину туристичного спорядження функцій;

– відсутність наочної часової стрічки, яка б відображала всі періоди прокату кожного товару, ускладнює контроль завантаженості спорядження та планування його використання;

– наявність тестового безкоштовного періоду, після закінчення якого користувач зобов’язаний придбати ліцензію, може стати перепоною для повноцінного оцінювання функціоналу в довгостроковій перспективі.

«Універсальна Система Обліку» від AS-Service є гнучким і масштабованим рішенням, однак її загальний підхід, орієнтований на різноманітні типи орендного бізнесу, не враховує низку специфічних аспектів прокату туристичного спорядження. Це може стати суттєвим обмеженням для магазину «Хан-Тенгрі», який потребує спеціалізованих функцій для ефективного контролю та планування прокату.

Отже, необхідність розробки власного програмного забезпечення для автоматизації обліку прокату туристичного спорядження в магазині «Хан-

Тенгрі», яке повністю відповідатиме специфічним потребам цього бізнесу, є цілком актуальною. Створення такого програмного продукту дасть змогу підвищити ефективність роботи менеджерів та адміністратора, зокрема: оперативно отримувати дані про наявність спорядження, скоротити час обслуговування клієнтів та автоматизувати формування звітності щодо орендних операцій.

1.3 Постановка задачі на розробку вебзастосунку для автоматизації обліку прокату туристичного спорядження у магазині «Хан-Тенгрі»

Зважаючи на результати проведеного аналізу практичної задачі інженерії програмного забезпечення, з метою підвищення рівня автоматизації обліку прокату туристичного спорядження в магазині «Хан-Тенгрі» було прийняте рішення розробити власне програмне забезпечення. Його призначення полягає в оптимізації роботи менеджера та адміністратора магазину, а також у спрощенні процесу ведення обліку прокату туристичного спорядження.

Вимоги до складу виконуваних функцій:

Додатком будуть користуватися два види користувачів: адміністратор (admin) та менеджер (user). Менеджер може виконувати усі функції адміністратора крім:

– додавання та редагування інформації про товари на складі (тільки адміністратор);

– перегляд та видалення зареєстрованих менеджерів (тільки адміністратор).

Нижче всі функції які доступні адміністратору

– реєстрація користувачів;

Вхідні дані: логін, пароль, роль

Результат: Створення нового облікового запису в системі.

– авторизація користувачів;

Вхідні дані: Логін, пароль, роль (підтягується автоматично згідно з даними вже зареєстрованого користувача)

Результат: Вхід у систему з правами доступу ролі, яка була визначена під час реєстрації (admin/user)

– перегляд та обробка інформації про товари на складі (додавання, видалення, редагування);

Вхідні дані: Код товару, фото, назва, кількість, ціна.

Результат: Додавання, видалення або редагування товарів на складі. (додавання/редагування доступне тільки admin)

– перегляд та обробка інформації про клієнтів (додавання, видалення, редагування);

Вхідні дані: Прізвище, ім'я, по батькові, категорія, телефон, електронна пошта.

Результат: Додавання, видалення або редагування даних клієнтів.

– перегляд та обробка інформації про ціни прокату (додавання, видалення, редагування);

Вхідні дані: Назва товару, ціни за 1-3, 4-6, 7+ днів.

Результат: Додавання, видалення або редагування цін прокату для різних термінів.

– створення замовлення на прокат (додавання усієї інформації щодо прокату, пошук товару та клієнта, внесення нового клієнта у разі його відсутності);

Вхідні дані: номер замовлення, прізвище клієнта, дата замовлення, статус прокату, прізвище співробітника що оформлював прокат, дата початку прокату, дата кінця прокату, ціна прокату, оплачена сума, борг, форма залогу, сума залогу, кількість товару в прокат

Результат: Створення нового запису прокату (додавання клієнта та товарів).

– редагування та видалення замовлення на прокат;

Вхідні дані: номер замовлення, прізвище клієнта, дата замовлення, статус прокату, прізвище співробітника що оформлював прокат, дата початку прокату, дата кінця прокату, ціна прокату, оплачена сума, борг, форма залогу, сума залогу, кількість товару в прокат

Результат: редагування та видалення прокату.

– перегляд часової стрічки (відображення проміжку від початкової до кінцевої дати прокату кожного товару що був взятий в прокат з урахуванням кількості цього товару);

Вхідні дані: Немає.

Результат: Відображення зайнятості товару у часі з маркуванням кількості конкретного товару взятого в прокати.

– оновлення статусу замовлення на прокат на «повернено» (зникнення маркування з часової стрічки, збереження запису в базі);

Вхідні дані: Статус ("повернено").

Результат: Оновлення статусу прокату, зникнення маркування на часовій стрічці.

– перегляд списку усіх товарів, записаних у конкретний прокат;

Вхідні дані: id прокату.

Результат: Перегляд списку товарів, що були взяті в прокат за певним замовленням.

– перегляд інформації про ціну прокату, пов’язаної з кожним товаром;

Вхідні дані: id товару.

Результат: Відображення пов’язаної ціни прокату.

– перегляд та видалення зареєстрованих менеджерів (доступне тільки admin)

Вхідні дані для видалення: id user.

Результат для видалення: Відображення списку менеджерів без обраного (видаленого) user.

– вихід з облікового запису.

Вхідні дані: Немає.

Результат: Завершення сесії адміністратора/менеджера.

Вимоги до організації вхідних та вихідних даних:

Дані зберігаються у базі даних MySQL.

Вхідні дані: введення даних здійснюється за допомогою пристроїв введення (клавіатура та миша). Також дані можуть вводитися вручну або обиратися із наявних випадаючих списків.

Вихідні дані: виведення даних відбувається на моніторі в режимі реального часу.

Вимоги до складу та параметрів технічних засобів:

Для роботи з ПЗ необхідна система наступної мінімальної конфігурації:

– процесор: двоядерний із тактовою частотою не менше 2,0 ГГц;

– оперативна пам'ять – 4GB;

– системний диск: 100 ГБ вільного дискового простору;

– необхідно мережеве з'єднання з виходом в інтернет.

Вимоги до інформаційної та програмної сумісності:

– сумісність з операційними системами: Операційна система не нижче Windows 10;

– сумісність з базами даних: Наявність системи управління базами даних (MySQL), що використовується для зберігання даних;

– сумісність з веббраузерами: Оскільки програма має веб-інтерфейс, то вона повинна бути сумісною з веб-браузерами, такими як Google Chrome, Mozilla Firefox, Opera.

Для розгортання вебзастосунку використовується хостинг із встановленою панеллю керування FastPanel або аналогічним програмним забезпеченням, що забезпечує зручне адміністрування серверних ресурсів [5]. Основними вимогами до середовища виконання є:

– наявність стабільного інтернет-з’єднання, оскільки вся взаємодія із системою відбувається онлайн;

– сервер із параметрами, достатніми для коректної роботи веб-додатку (рекомендовано принаймні 1 ядро процесора, 2 GB оперативної пам’яті та 10 GB вільного дискового простору);

– операційна система на сервері, сумісна з обраними технологіями (наприклад, Linux-дистрибутив з підтримкою веб-серверів Apache або Nginx);

– наявність підтримуваної версії вебсервера (Apache або Nginx), що забезпечує обробку HTTP-запитів;

– підтримка необхідної версії середовища виконання (Node.js).

Крім того, для повноцінної роботи з вебзастосунком на стороні користувача потрібно мати сучасний веб-браузер (Google Chrome, Mozilla Firefox, Opera) з актуальними оновленнями та доступ до мережі Інтернет.

Детальна постановка задачі представлена у технічному завданні, яке наведено у додатку А.

2 РОЗРОБКА АРХІТЕКТУРИ ВЕБЗАСТОСУНКУ ДЛЯ АВТОМАТИЗАЦІЇ ОБЛІКУ ПРОКАТУ ТУРИСТИЧНОГО СПОРЯДЖЕННЯ МАГАЗИНУ «ХАН-ТЕНГРІ»

2.1 Обгрунтування вибору архітектурного стилю для вебзастосунку для автоматизації обліку прокату туристичного спорядження магазину «Хан-Тенгрі»

Для розробки вебзастосунку для автоматизації обліку прокату туристичного спорядження магазину «Хан-Тенгрі» було обрано архітектурний стиль – багаторівнева архітектура (Layered Architecture).

Багаторівнева архітектура складається з кількох шарів (рівнів), де кожен рівень відповідає за виконання певних функцій і взаємодіє з іншими рівнями в упорядкованій структурі. В програмному забезпеченні для автоматизації обліку прокату туристичного спорядження магазину «Хан-Тенгрі» кожен шар виконує визначені завдання, починаючи з обробки запитів, перевірки прав доступу, бізнес-логіки та закінчуючи доступом до бази даних. Ця архітектура дозволяє розділити відповідальності, спрощуючи розробку, тестування та масштабування застосунку [6].

Обґрунтування вибору багаторівневої архітектури:

– чітке розділення обов’язків: Кожен рівень відповідає за конкретні функції, що робить систему зрозумілою і легкою в підтримці;

– гнучкість: Багаторівнева архітектура дозволяє легко змінювати чи оновлювати окремі шари;

– простота реалізації: Логіка взаємодії з базою даних реалізована безпосередньо в контролерах, що зберігає простоту коду і спрощує його розробку;

– масштабованість: Багаторівнева архітектура забезпечує можливість масштабування системи, оскільки нові функції можуть бути додані як окремі контролери або middleware, не порушуючи структуру інших рівнів.

2.2 Розробка діаграми розгортання та схеми архітектури вебзастосунку для автоматизації обліку прокату туристичного спорядження магазину «Хан-Тенгрі» на основі вибраного архітектурного стилю

На основі вибраного стилю було розроблено таку схему архітектури вебзастосунку для автоматизації обліку прокату туристичного спорядження магазину «Хан-Тенгрі». Загальна діаграма архітектури представлена на рисунку 2.1.

Структура :

1. Презентаційний рівень (Client на React.js): Цей шар відповідає за інтерфейс користувача, де відбувається взаємодія користувача із системою. Він відправляє HTTP-запити на сервер, які надалі обробляються backend-ом.
2. Рівень обробки запитів (Request handling): На цьому рівні server.js приймає та обробляє вхідні HTTP-запити від клієнта. Цей рівень забезпечує маршрутизацію запитів до відповідних контролерів і додає загальну структуру для роботи застосунку.
3. Проміжний рівень (Middleware): adminOnly.js виконує функції проміжного рівня, забезпечуючи перевірку прав доступу для адміністративних дій. Це дозволяє контролювати доступ до певних маршрутів, забезпечуючи захист даних.
4. Рівень бізнес-логіки (Controllers): Контролери, такі як clientController.js, rentController.js, goodsController.js, та інші, містять основну логіку обробки даних для різних сутностей. Тут виконуються CRUD-операції над

даними, що дозволяє структурувати бізнес-логіку і забезпечувати різноманітні операції з клієнтами, товарами, цінами та прокатами.

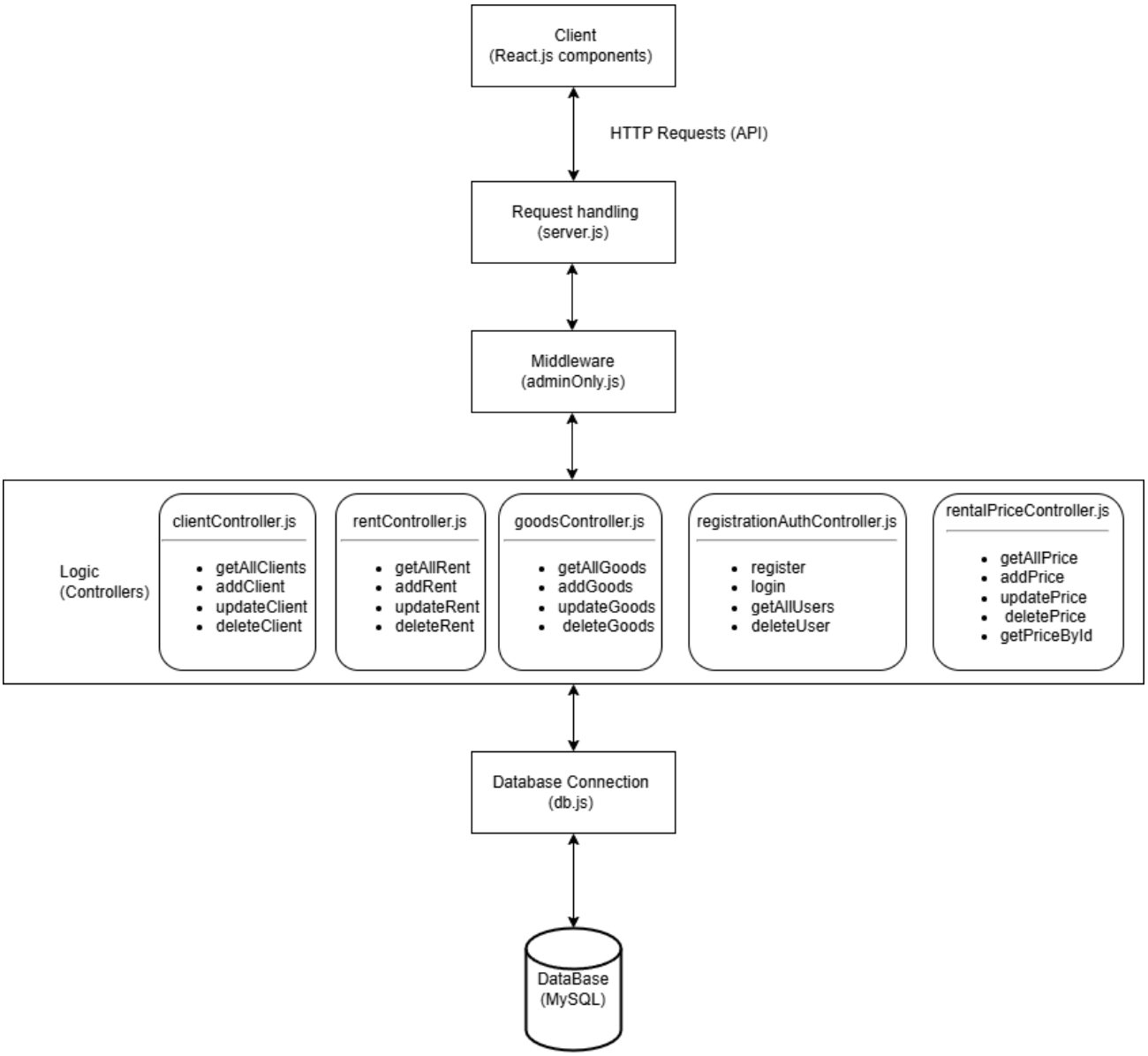


Рисунок 2.1 – Загальна діаграма архітектури програмного забезпечення для автоматизації обліку прокату туристичного спорядження магазину «Хан-Тенгрі»

5. Рівень доступу до даних (Database connection): db.js встановлює з’єднання з базою даних MySQL та надає можливість контролерам напряму

працювати з базою.

6. База даних (MySQL): База даних відповідає за зберігання всіх необхідних даних, таких як клієнти, товари, прокати, користувачі, ціни прокатів.

Діаграма розгортання (Deployment Diagram) у мові моделювання UML використовується для відображення фізичного розміщення (розгортання) програмних компонентів на апаратних або віртуальних вузлах (nodes). Вона наочно показує, як різні частини системи (так звані «артефакти») взаємодіють між собою та де вони «живуть» під час виконання програми. Зазвичай такі діаграми використовуються на етапі проєктування або документування архітектури системи, щоб краще зрозуміти конфігурацію середовища розробки та розгортання, а також взаємозв’язки між компонентами [7].

Діаграма розгортання модельованої системи програмного забезпечення для автоматизації обліку прокату туристичного спорядження магазину «Хан-Тенгрі» представлена на рисунку 2.2.

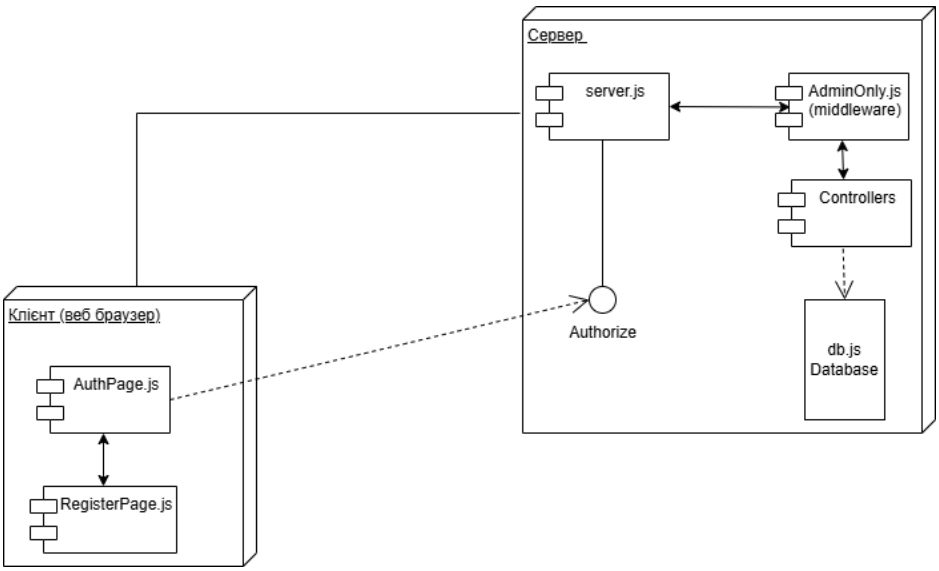


Рисунок 2.2 - Діаграма розгортання модельованої системи програмного забезпечення для автоматизації обліку прокату туристичного спорядження магазину «Хан-Тенгрі»

На наведеній діаграмі є два основних вузли:

1. Клієнт (веббраузер), де розміщено всі файли фронтенду (представлені тільки ті що відповідають за авторизацію та реєстрацію):

– AuthPage.js – компонент для авторизації користувачів.

– RegisterPage.js – компонент для реєстрації нових користувачів. Ці файли запускаються у середовищі веббраузера, і саме звідти відправляються запити до сервера.

2. Сервер, який включає кілька програмних компонентів:

– server.js – головний файл, що відповідає за запуск сервера (наприклад, на Node.js). Він приймає та обробляє HTTP-запити від клієнтів.

– AdminOnly.js (middleware) – проміжний шар (middleware), який перевіряє права доступу користувачів. Якщо користувач не має ролі «адміністратора», виклик може бути відхилено або перенаправлено.

– Controllers – модулі, що містять логіку обробки даних, взаємодіють із базою даних та формують відповіді клієнтам.

– db.js (Database) – базу даних та модуль взаємодії з нею.

3 РОЗРОБКА ПРОЄКТУ ВЕБЗАСТОСУНКУ ДЛЯ АВТОМАТИЗАЦІЇ ОБЛІКУ ПРОКАТУ ТУРИСТИЧНОГО СПОРЯДЖЕННЯ МАГАЗИНУ «ХАН-ТЕНГРІ»

3.1 Аналіз вимог до вебзастосунку для автоматизації обліку прокату туристичного спорядження магазину «Хан-Тенгрі»

3.1.1 Побудова моделі варіантів використання

Діаграма варіантів використання (Use Case Diagram) – це один із типів діаграм у мові моделювання UML, який дає змогу візуально відобразити основні вимоги до програмного забезпечення. Вона слугує відправною концептуальною моделлю для проєктованої системи, допомагаючи визначити, хто (актор) взаємодіє із системою та які послуги чи функції (прецеденти) ця система має надавати [8].

Зазвичай діаграма варіантів використання складається з трьох ключових елементів:

1. Актори – користувачі або зовнішні системи, що взаємодіють із розроблюваним ПЗ.
2. Прецеденти (варіанти використання) – конкретні сценарії або функціональні можливості, які система надає акторам.
3. Зв’язки – різноманітні типи відносин (асоціації, узагальнення тощо) між акторами та прецедентами, що вказують на спосіб і характер взаємодії.

Один актор може бути пов’язаний із кількома прецедентами, якщо він виконує різні ролі чи завдання в системі. У свою чергу, один прецедент може обслуговувати декількох акторів, якщо функціональність, яку він реалізує, є спільною для кількох груп користувачів.

У теоретичному аспекті діаграми варіантів використання часто застосовують на початкових етапах розробки ПЗ, оскільки вони дають змогу швидко узгодити функціональні вимоги між замовниками, аналітиками та командою розробників. Завдяки наочній формі представлення вони полегшують розуміння системи та допомагають уникнути непорозумінь у подальшому процесі розробки.

У програмному забезпеченні для обліку туристичного спорядження магазину «Хан-Тенгрі» будуть такі актори:

– адміністратор (admin);
– менеджер (user).

Короткий опис акторів представлено в таблиці 3.1.

Таблиця 3.1 – Короткий опис акторів

Актор	Короткий опис
Адміністратор (admin)	Користувач з розширеними правами доступу до системи, відповідальний за управління базою даних, контроль за процесами прокату, керування товарами, налаштуванням ціноутворення, керування обліковими записами співробітників.
Менеджер (user)	Користувач з обмеженими правами доступу, який може керувати прокатами, клієнтами, але не має доступу до адміністративних функцій (керування товарами та управління обліковими записами).

Виявлені варіанти використання представлені у таблиці 3.2.

Таблиця 3.2 – Виявлені варіанти використання

Актор	Назва	Короткий опис
1	2	3
Адміністратор, менеджер	Реєстрація	Дозволяє адміністратору та співробітнику створити новий обліковий запис.
Адміністратор, менеджер	Авторизація	Дозволяє адміністратору та співробітнику увійти в систему під відповідним обліковим записом.

Продовження таблиці 3.2

1	2	3
Адміністратор, менеджер	Перегляд товару на складі	Дозволяє адміністратору та співробітнику переглянути увесь список (таблицю) товарів які є на складі.
Адміністратор	Додавання товару на склад	Дозволяє адміністратору додати товар у базу даних (склад).
Адміністратор	Редагування товару на складі	Дозволяє адміністратору змінити інформацію про наявний товар у базі даних (на складі).
Адміністратор	Перегляд зареєстрованих користувачів	Дозволяє адміністратору переглядати всіх зареєстрованих у системі користувачів
Адміністратор	Видалення зареєстрованого користувача	Дозволяє адміністратору видалити обраного зареєстрованого користувача
Адміністратор, менеджер	Видалення товару зі складу	Дозволяє адміністратору та співробітнику видалити товар з бази даних (зі складу).
Адміністратор, менеджер	Перегляд бази даних клієнтів	Дозволяє адміністратору та співробітнику переглянути увесь список (таблицю) клієнтів.
Адміністратор, менеджер	Додавання клієнтів в базу даних	Дозволяє адміністратору та співробітнику додати клієнта у базу даних.
Адміністратор, менеджер	Редагування клієнтів в базу даних	Дозволяє адміністратору та співробітнику змінити інформацію про наявного клієнта у базі даних
Адміністратор, менеджер	Видалення клієнтів з бази даних	Дозволяє адміністратору та співробітнику видалити клієнта з бази даних.
Адміністратор, менеджер	Перегляд ціни прокату	Дозволяє адміністратору та співробітнику переглянути увесь список цін прокатів.
Адміністратор, менеджер	Додавання ціни прокату в базу даних	Дозволяє адміністратору та співробітнику додати ціну прокату у базу даних.
Адміністратор, менеджер	Редагування ціни прокату	Дозволяє адміністратору та співробітнику змінити інформацію про наявну ціну прокату у базі даних
Адміністратор, менеджер	Видалення ціни прокату	Дозволяє адміністратору та співробітнику видалити ціну прокату з бази даних.
Адміністратор, менеджер	Перегляд прокатів	Дозволяє адміністратору та співробітнику переглянути увесь список прокатів.
Адміністратор, менеджер	Створення прокату	Дозволяє адміністратору та співробітнику додавання всієї інформації щодо прокату, пошук товару за номером та запис його в прокат (кількість цього товару яку бажає взяти в прокат клієнт), пошук клієнта за номером телефона та запис клієнта в цей прокат. При відсутності даного клієнта в базі даних, можливість додавати клієнта під час створення прокату та робити запис створеного клієнта в цей прокат

Закінчення таблиці 3.2

1	2	3
Адміністратор, менеджер	Редагування прокату	Дозволяє адміністратору та співробітнику змінити інформацію про наявний прокат у базі даних
Адміністратор, менеджер	Видалення прокату	Дозволяє адміністратору та співробітнику видалити прокат з бази даних.
Адміністратор, менеджер	Перегляд часової стрічки	Дозволяє адміністратору та співробітнику перегляд часової стрічки в якій напроти кожного товару знаходиться маркування (від початкової дати прокату до кінцевої) з кількістю конкретного товару взятого в різні прокати
Адміністратор, менеджер	Зміна статусу прокату на «повернено»	Дозволяє адміністратору та співробітнику зміну статусу прокату на «повернено» при якому маркування зникають з часової стрічки при тому що запис про прокат залишається в базі даних
Адміністратор, менеджер	Перегляд усіх товарів які записані в конкретний прокат	Дозволяє адміністратору та співробітнику перегляд усіх товарів які записані в конкретний прокат
Адміністратор, менеджер	Перегляд ціни прокату що пов’язана з кожним товаром	Дозволяє адміністратору та співробітнику перегляд ціни прокату що пов’язана з кожним товаром
Адміністратор, менеджер	Вихід з облікового запису	Дозволяє адміністратору та співробітнику вийти з діючого облікового запису

Діаграма варіантів використання програмного забезпечення для обліку туристичного спорядження магазину «Хан-Тенгрі» представлена на рисунку 3.1

Рисунок 3.1 – Діаграма варіантів використання програмного забезпечення для обліку прокату туристичного спорядження магазину «Хан-Тенгрі»

Таким чином ми розробили діаграму варіантів використання програмного забезпечення для обліку прокату туристичного спорядження магазину «Хан-Тенгрі»

3.1.2 Специфікація основних варіантів використання вебзастосунку для обліку туристичного спорядження магазину «Хан-Тенгрі»

Розглянемо більш детально основні варіанти використання програмного забезпечення для обліку туристичного спорядження магазину «Хан-Тенгрі»

Специфікація основних варіантів використання

Специфікації додавання, редагування та видалення є однаковими у всіх сутностях (товари, ціни прокату, клієнти) окрім прокату. Його специфікації ми розглянемо окремо.

1. Додавання товару на склад

Призначення: додавання товару у базу даних (склад).

Учасники: адміністратор

Передумови: користувач авторизований як адміністратор

Стартова точка (що робиться для запуску): запуск користувачем процесу додавання товару

Процедура (сценарій):

1) Система виводить користувачу сторінку з таблицею зі всіма товарами

2) Користувач виконує дію для запуску процесу додавання товару

3) Система показує користувачу форму для додавання товару

4) Користувач заповнює форму необхідною інформацією та погоджується з додаванням товару

5) Система перевіряє поля форми на валідацію

5.1) Система зберігає товар у разі успіху у базу даних

5.2) Система показує повідомлення про помилку у разі провалу проходження валідації та чекає поки користувач виправить помилки.

5.3) Після виправлення помилок користувач повторно погоджується з додаванням товару та система знову перевіряє поля форми на валідацію (і так доки форма не пройде перевірку на валідацію)

Альтернативна процедура: немає

Аварійні умови (при яких умовах відбувається аварійне завершення): немає

Результати: доданий новий товар у базу даних (відображення нового товару разом з іншими в таблиці зі всіма товарами)

Умови після завершення використання: немає

Обмеження \ виключення: немає

Особливості: немає

Коментарі: немає

Відповідні умови: немає

Складність \ важливість: низька/базовий процес для роботи з товарами в БД

2. Редагування товару на складі

Призначення: редагування товару у базі даних (склад).

Учасники: адміністратор

Передумови: користувач авторизований як адміністратор

Стартова точка (що робиться для запуску): запуск користувачем процесу редагування товару

Процедура (сценарій):

1) Система виводить користувачу сторінку з таблицею зі всіма товарами

2) Користувач обирає товар

3) Користувач виконує дію для запуску процесу редагування обраного товару

4) Система показує користувачу форму для редагування товару

5) Користувач змінює обрані поля форми з інформацією про обраний товар

6) Користувач погоджується з редагуванням товару

7) Система перевіряє поля форми на валідацію

7.1) Система зберігає зміни у товарі у разі успіху у базу даних

7.2) Система показує повідомлення про помилку у разі провалу проходження валідації та чекає поки користувач виправить помилки.

7.3) Після виправлення помилок користувач повторно погоджується з редагуванням товару та система знову перевіряє поля форми на валідацію (і так доки форма не пройде перевірку на валідацію)

Альтернативна процедура: немає

Аварійні умови (при яких умовах відбувається аварійне завершення): немає

Результати: оновлення інформації про конкретний товар у базі даних

Умови після завершення використання: немає

Обмеження \ виключення: немає

Особливості: немає

Коментарі: немає

Відповідні умови: немає

Складність \ важливість: низька/базовий процес для роботи з товарами в БД

3. Видалення товару зі складу

Призначення: видалення товару з бази даних (склад).

Учасники: адміністратор, співробітник

Передумови: користувач авторизований у додатку

Стартова точка (що робиться для запуску): запуск користувачем процесу видалення товару

Процедура (сценарій):

1) Система виводить користувачу сторінку з таблицею зі всіма товарами

2) Користувач обирає товар

3) Користувач виконує дію для запуску процесу видалення обраного товару

4) Система показує попередження про видалення

5) Користувач підтверджує видалення товару

6) Система видаляє товар з бази даних

Альтернативна процедура: немає

Аварійні умови (при яких умовах відбувається аварійне завершення): немає

Результати: видалення обраного товару із бази даних

Умови після завершення використання: немає

Обмеження \ виключення: немає

Особливості: немає

Коментарі: немає

Відповідні умови: немає

Складність \ важливість: низька/базовий процес для роботи з товарами в БД

4. Створення прокату

Призначення: створення прокату (внесення інформації про прокат в базу даних, створення зв’язку між клієнтом, товаром та прокатом, створення маркування на часовій стрічці).

Учасники: адміністратор, співробітник

Передумови: користувач авторизований в додатку

Стартова точка (що робиться для запуску): запуск користувачем процесу створення прокату

Процедура (сценарій):

1) Система виводить користувачу сторінку з часовою стрічкою та товарами які пов’язані з нею.

2) Користувач виконує дію для запуску процесу створення прокату

3) Система показує користувачу форму для створення прокату

4) Користувач заповнює форму необхідною інформацією

5) Система виконує пошук клієнта за номером телефона

6) Система виконує запис цього клієнта в прокат

7) Користувач погоджується зі створенням прокату

8) Система робить перевірку на валідацію

8.1) Система зберігає прокату у разі успіху у базу даних (система створює маркування на часовій стрічці напроти товару який знаходиться в прокаті)

8.2) Система показує повідомлення про помилку у разі провалу проходження валідації та чекає поки користувач виправить помилки.

8.3) Після виправлення помилок користувач повторно погоджується зі створенням прокату та система знову перевіряє поля форми на валідацію (і так доки форма не пройде перевірку на валідацію)

Альтернативна процедура:

1) Система виводить користувачу сторінку з часовою стрічкою та товарами які пов’язані з нею.

2) Користувач виконує дію для запуску процесу створення прокату

3) Система показує користувачу форму для створення прокату

4) Користувач заповнює форму необхідною інформацією

5) Система виконує пошук клієнта за номером телефона

6) Система виконує запис цього клієнта в прокат

7) Система виконує пошук товару по його коду

8) Система виконує запис знайденого товару в прокат (кількість товару та підтягнення цього товару з бази даних)

9) Система виконує перевірку на валідацію

1.1) Система зберігає прокат у разі успіху у базу даних (система створює маркування на часовій стрічці напроти товару який знаходиться в прокаті)

1.2) Система показує повідомлення про помилку у разі провалу проходження валідації та чекає поки користувач виправить помилки.

1.3) Після виправлення помилок користувач повторно погоджується зі створенням прокату та система знову перевіряє поля форми на валідацію (і так доки форма не пройде перевірку на валідацію)

Альтернативна процедура:

1) Система виводить користувачу сторінку з часовою стрічкою та товарами які пов’язані з нею.

2) Користувач виконує дію для запуску процесу створення прокату

3) Система показує користувачу форму для створення прокату

4) Користувач заповнює форму необхідною інформацією

5) Система виконує пошук клієнта за номером телефона

6) Система не знаходить клієнта в базі даних (користувач запускає процедуру додавання нового клієнта)

7) Клієнт заповнює необхідні поля в формі додавання нового клієнта

8) Система виконує запис нового клієнта в базу даних та в цей прокат

9) Система виконує перевірку на валідацію

9.1) Система зберігає прокату у разі успіху у базу даних (система створює маркування на часовій стрічці напроти товару який знаходиться в прокаті)

9.2) Система показує повідомлення про помилку у разі провалу проходження валідації та чекає поки користувач виправить помилки.

9.3) Після виправлення помилок користувач повторно погоджується зі створенням прокату та система знову перевіряє поля форми на валідацію (і так доки форма не пройде перевірку на валідацію)

Аварійні умови (при яких умовах відбувається аварійне завершення): немає

Результати: створений новий прокат, створено маркування напроти того товару який взяли в прокат

Умови після завершення використання: немає

Обмеження \ виключення: немає

Особливості: немає

Коментарі: немає

Відповідні умови: немає

Складність \ важливість: середня/головний процес створення прокату та зв’язків

4 Редагування прокату

Призначення: редагування прокату у базі даних.

Учасники: адміністратор, співробітник

Передумови: користувач авторизований в додатку

Стартова точка (що робиться для запуску): запуск користувачем процесу редагування прокату

Процедура (сценарій):

1) Система виводить користувачу сторінку з таблицею зі всіма прокатами

2) Користувач обирає прокат

3) Користувач виконує дію для запуску процесу редагування обраного прокату

4) Система показує користувачу форму для редагування прокату

5) Користувач змінює обрані поля форми з інформацією про обраний прокат

6) Користувач погоджується з редагуванням прокату

7) Система перевіряє поля форми на валідацію

7.1) Система зберігає зміни прокату у разі успіху у базу даних

7.2) Система показує повідомлення про помилку у разі провалу проходження валідації та чекає поки користувач виправить помилки.

7.3) Після виправлення помилок користувач повторно погоджується з редагуванням прокату та система знову перевіряє поля форми на валідацію (і так доки форма не пройде перевірку на валідацію)

Альтернативна процедура:

1) Система виводить користувачу сторінку з таблицею зі всіма прокатами

2) Користувач обирає прокат

3) Користувач виконує дію для запуску процесу редагування обраного прокату

4) Система показує користувачу форму для редагування прокату

5) Користувач змінює поле статусу прокату на «повернено»

6) Користувач погоджується з редагуванням прокату

7) Система виконує перевірку на валідацію

7.1) Система зберігає оновлену інформацію про прокат у разі успіху у базу даних

7.2) Система видаляє маркування з часової стрічки напроти тих товарів які входили в цей прокат (інформація про цей прокат все ще залишається у базі даних)

7.3) Система показує повідомлення про помилку у разі провалу проходження валідації та чекає поки користувач не виправить помилку

7.4) Після виправлення помилок користувач повторно погоджується з редагуванням прокату та система знову перевіряє поля форми на валідацію (і так доки форма не пройде перевірку на валідацію)

Аварійні умови (при яких умовах відбувається аварійне завершення): немає

Результати: оновлення інформації про прокат у базі даних

Умови після завершення використання: немає

Обмеження \ виключення: немає

Особливості: немає

Коментарі: немає

Відповідні умови: немає

Складність \ важливість: низька/базовий процес для роботи з прокатами в БД

5 Видалення прокату

Призначення: видалення прокату з бази даних.

Учасники: адміністратор, співробітник

Передумови: користувач авторизований у додатку

Стартова точка (що робиться для запуску): запуск користувачем процесу видалення прокату

Процедура (сценарій):

1) Система виводить користувачу сторінку з таблицею зі всіма прокатами

2) Користувач обирає прокат

3) Користувач виконує дію для запуску процесу видалення обраного прокату

4) Система показує попередження про видалення

5) Користувач підтверджує видалення прокату

6) Система видаляє прокат з бази даних та маркування цього прокату з часової стрічки

Альтернативна процедура: немає

Аварійні умови (при яких умовах відбувається аварійне завершення): немає

Результати: видалення обраної ціни прокату із бази даних

Умови після завершення використання: немає

Обмеження \ виключення: немає

Особливості: немає

Коментарі: немає

Відповідні умови: немає

Складність \ важливість: низька/базовий процес для роботи з ціною прокату в БД

3.2 Проєкт вебзастосунку для автоматизації обліку прокату туристичного спорядження магазину «Хан-Тенгрі»

3.2.1 Ескізний проект вебзастосунку для автоматизації обліку прокату туристичного спорядження магазину «Хан-Тенгрі»

Сценарії основних варіантів використання

Кожен об’єкт програмного забезпечення може мати власну поведінку: він здатний перебувати в певних станах, переходити з одного стану в інший і виконувати конкретні дії під час реалізації свого сценарію поведінки. Ця поведінка відображається через сукупність станів об’єкта, а діаграми діяльності надають можливість візуально продемонструвати такі переходи й дії

Опис варіанту використання «Додавання товару на склад» представлений у пункті 3.1.2.

Діаграма діяльності для варіанту використання додавання товару на склад представлена на рисунку 3.2.

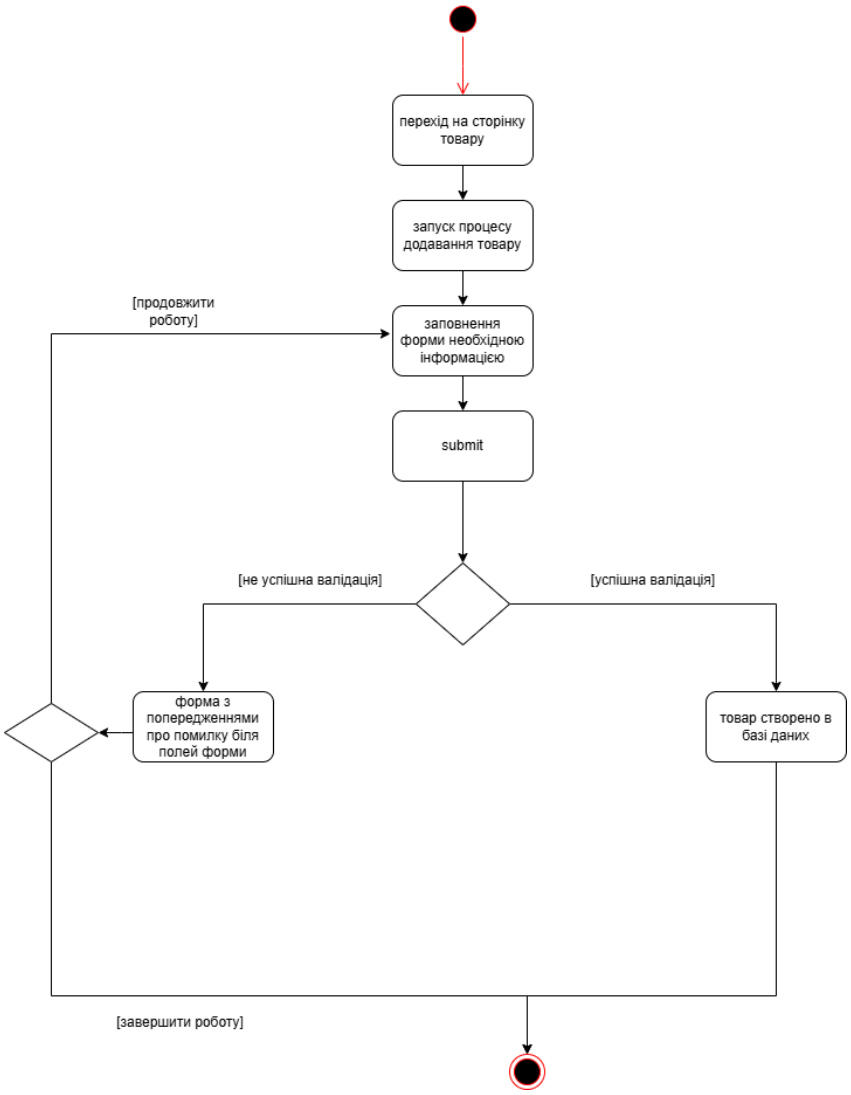


Рисунок 3.2 – Діаграма діяльності для варіанту використання додавання товару на склад

Опис варіанту використання «Видалення товару зі складу» представлений у пункті 3.1.2.

Діаграма діяльності для варіанту використання видалення товару зі складу представлена на рисунку 3.3

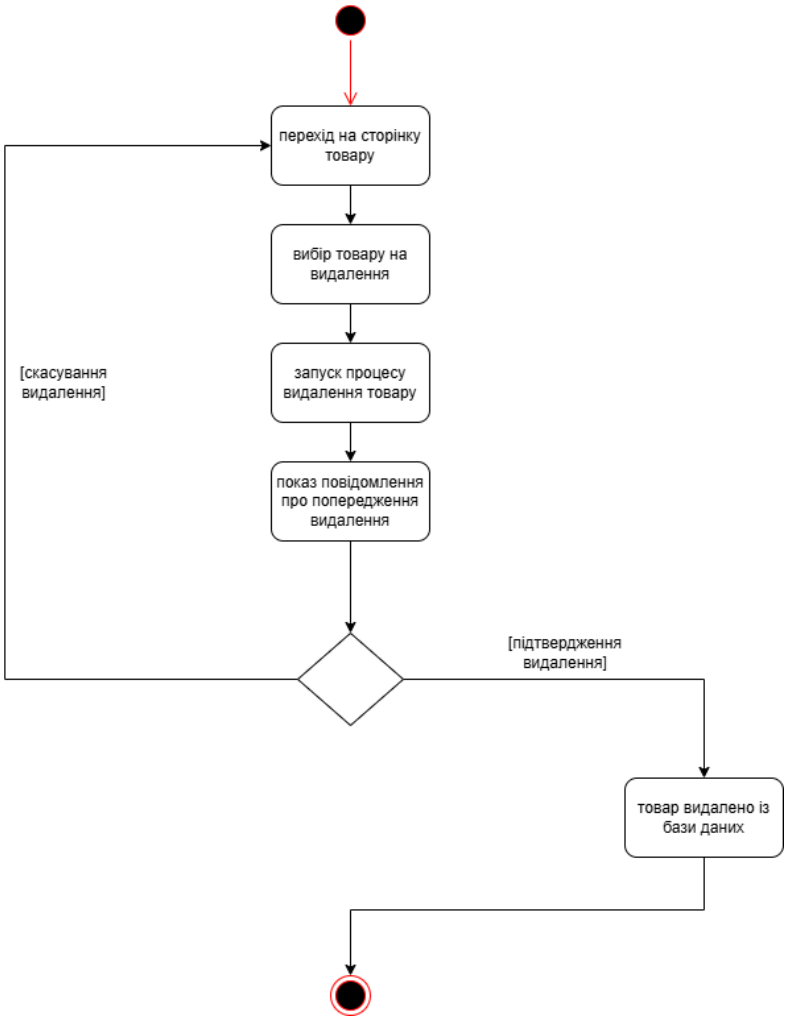


Рисунок 3.3 – Діаграма діяльності для варіанту використання видалення товару зі складу.

Розробка концептуальної моделі

Сутність – це реальний або уявний об'єкт, інформація про який повинна зберігатися в проектованій системі. Атрибут повинен мати ім'я, унікальне в межах сутності [9].

В роботі можна виділити наступні сутності та їх атрибути:

– Клієнт: прізвище, ім’я, по батькові, категорія, телефон, пошта.
– Товар: фото, код, назва, ціна товару, кількість товару на складі
– Ціна прокату: ціна за 1-3 дні, ціна за 4-6 днів, ціна за 7+ днів

– Прокат: номер замовлення, дата замовлення, статус прокату, прізвище співробітника, дата початку прокату, дата кінця прокату, ціна прокату, оплачена сума, борг, форма залогу, сума залогу

– User: логін, пароль, роль

Концептуальна модель для вебзастосунку для обліку прокату туристичного спорядження у магазину «Хан-Тенгрі» представлена на рисунку 3.4

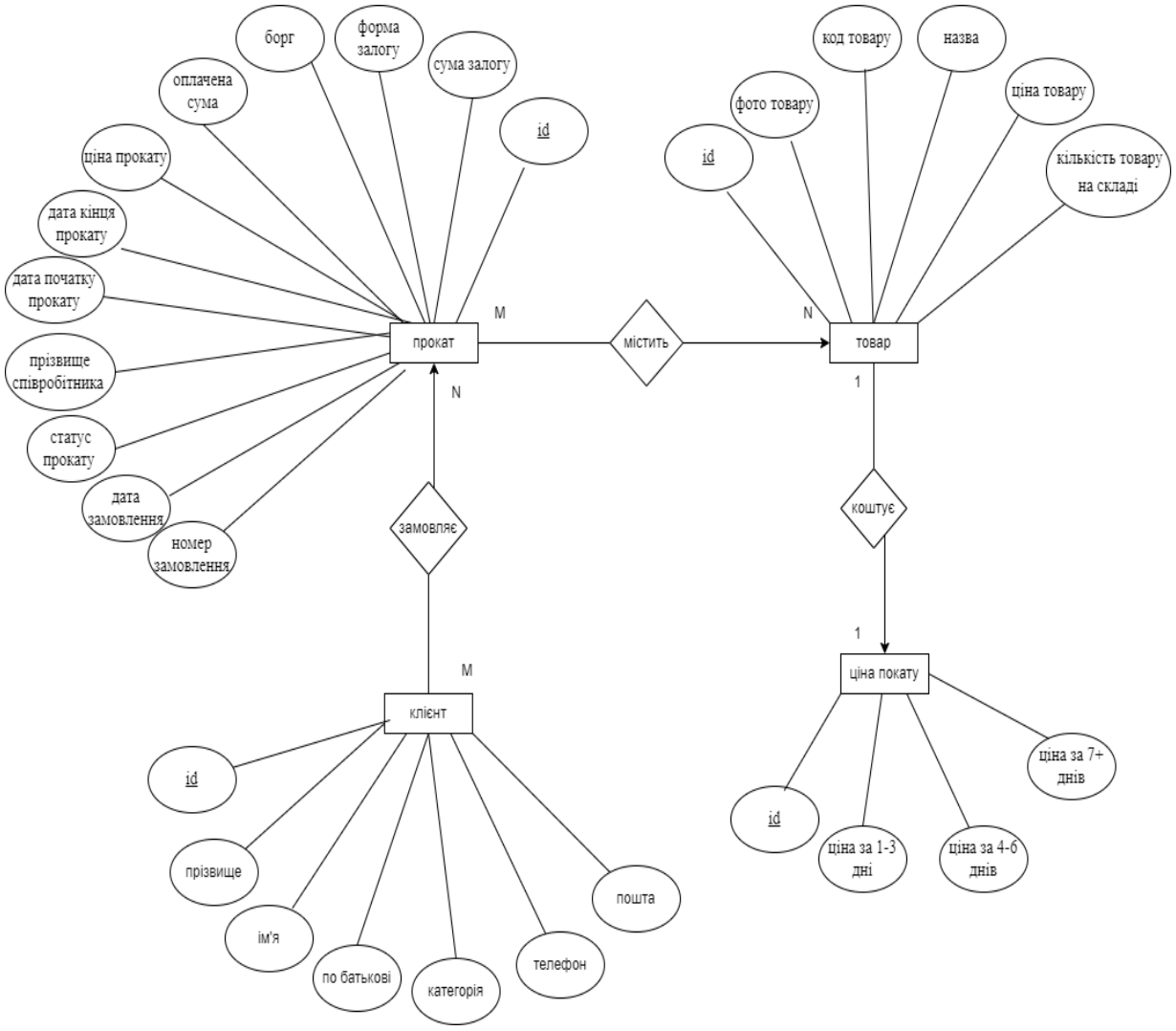


Рисунок 3.4 – Концептуальна модель для вебзастосунку для обліку прокату туристичного спорядження у магазину «Хан-Тенгрі»

Розробка прототипу інтерфейсу користувача

Початковим кроком у створенні інтерфейсу стала побудова діаграми варіантів використання, яка описує функціональні програмного забезпечення для автоматизації обліку прокату туристичного спорядження магазину «Хан-Тенгрі»

На її основі було спроектовано візуальну частину застосунку – набір компонентів та елементів вікон, що забезпечують взаємодію користувача із системою.

Зразки графічного інтерфейсу наведено на рисунках 3.5 – 3.7.

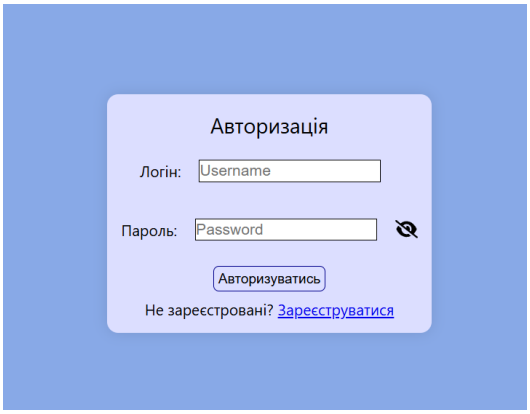


Рисунок 3.5 – Форма авторизації вебзастосунку для автоматизації обліку прокату туристичного спорядження магазину «Хан-Тенгрі»

Рисунок 3.6 – Головна сторінка прокатів вебзастосунку для автоматизації обліку прокату туристичного спорядження магазину «Хан-Тенгрі»

Рисунок 3.7 – Головна сторінка товарів вебзастосунку для автоматизації обліку прокату туристичного спорядження магазину «Хан-Тенгрі»

Після завершення роботи над ескізним проєктом вебзастосунку для автоматизації обліку прокату туристичного спорядження магазину «Хан-Тенгрі», можна переходити до чергового етапу проєктування – розробки технічного проєкту цього ПЗ.

3.2.2 Технічний проект вебзастосунку для автоматизації обліку прокату туристичного спорядження магазину «Хан-Тенгрі»

У своїй роботі я використовувала мову програмування JavaScript та бібліотеку React, яка має компонентну структуру. Оскільки в проекті застосовувалися функціональні компоненти, класична діаграма класів не є доцільною. Замість цього, було розроблено діаграму ієрархії компонентів, яка наочно відображає взаємозв'язки та структуру функціональних елементів програми.

Діаграма компонентів та їх взаємодії на клієнтському рівні представлена на рисунку 3.8.

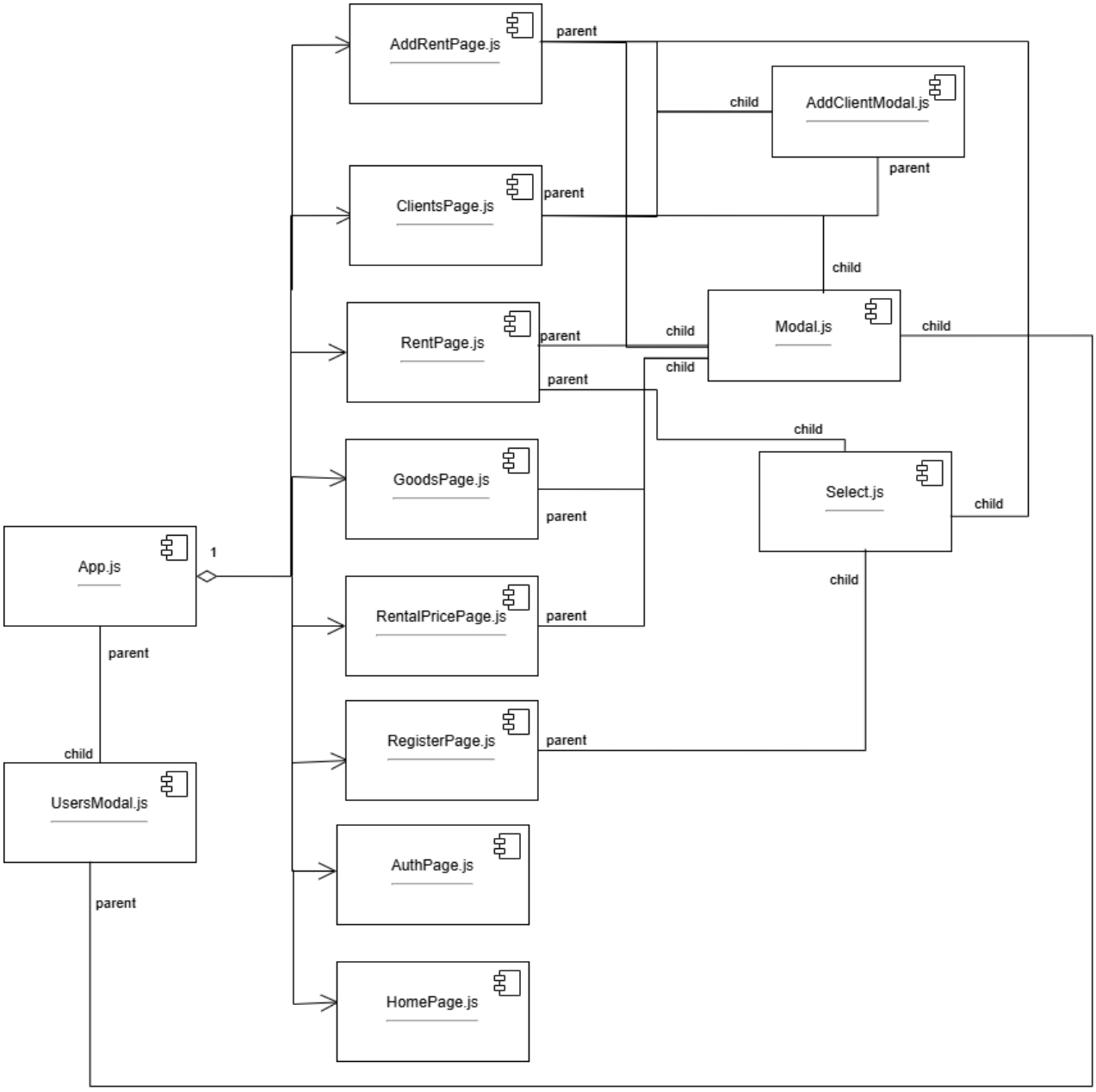


Рисунок 3.8. – Діаграма компонентів та їх взаємодії на клієнтському рівні для програмного забезпечення для автоматизації обліку прокату туристичного спорядження магазину «Хан-Тенгрі»

Специфікація компонентів:

1. Специфікація компонента App.js

Ім'я: App.js

Відповідальність: Головний компонент, що відповідає за маршрутизацію додатку, відображення навігаційних посилань та контроль авторизації користувача.

Атрибути (properties компонентів): Немає прямих properties.

Стан:

– isLoggedIn для збереження стану авторизації
– userRole для збереження ролі користувача

Методи:

– handleLogin: асинхронний метод для авторизації користувача через API (axios).
– handleLogout: метод для виходу користувача, очищення локального сховища.

Залежність між компонентами:

– Батьківський компонент для всіх сторінок (дочірні компоненти, завантажувані через Suspense і lazy).
– Передає properties дочірнім компонентам, таким як Home, AuthPage, GoodsPage тощо.

2. Специфікація компонента ClientsPage.js

Ім'я: ClientsPage.js

Відповідальність: Компонент відповідає за відображення та управління списком клієнтів, включаючи додавання, редагування та видалення клієнтів. Виконує взаємодію з API для отримання, створення, оновлення та видалення клієнтів.

Атрибути (properties компонентів): Немає прямих properties.

Стан (state):

– clients: масив клієнтів, завантажених із сервера.
– editingClient: об’єкт поточного клієнта, дані якого редагуються.

- error: повідомлення про помилку для відображення помилок у мережі.
- showAddClientModal: логічне значення для контролю відкриття модального вікна для додавання нового клієнта.
- showEditClientModal: логічне значення для контролю відкриття модального вікна для редагування клієнта.
- showDeleteConfirmationModal: логічне значення для контролю відкриття модального вікна підтвердження видалення.
- deleteItemId: ідентифікатор клієнта, який буде видалено.
- validationErrors: об’єкт помилок валідації для форми додавання або редагування клієнта.
- formData: об'єкт даних форми для додавання клієнта.

Методи:

- openAddClientModal: відкриває модальне вікно для додавання нового клієнта.
- handleCloseModal: закриває модальне вікно для додавання клієнта, очищає дані форми та помилки валідації.
- openEditClientModal: відкриває модальне вікно для редагування існуючого клієнта та встановлює дані для редагування.
- handleCloseEditModal: закриває модальне вікно для редагування клієнта.
- handleDeleteConfirmation: відкриває модальне вікно для підтвердження видалення клієнта та зберігає ID клієнта для видалення.
- handleAddClient: перевіряє дані форми, додає нового клієнта через API та оновлює список клієнтів після успішного додавання. Закриває модальне вікно після завершення.
- handleChange: обробляє зміни у полях форми для додавання або редагування клієнта, оновлює editingClient та formData.

- handleEditClient: надсилає оновлені дані клієнта через API для збереження змін, оновлює список клієнтів та закриває модальне вікно після успішного редагування.
- handleDeleteClient: видаляє клієнта з API та оновлює список клієнтів після підтвердження видалення.

Залежність між компонентами:

- Використовує дочірні компоненти Modal для відображення модальних вікон (додавання, редагування та підтвердження видалення).
- Використовує компонент AddClientModal для модального вікна додавання нового клієнта, передаючи дані форми, помилки валідації, методи handleChange та handleAddClient.

3. Специфікація компонента GoodsPage.js

Ім'я: GoodsPage.js

Відповідальність: Відповідає за відображення та управління списком товарів, а також взаємодію з товарами (додавання, редагування, видалення товарів, перегляд ціни на прокат).

Атрибути (properties компонентів): userRole: визначає роль користувача, використовується для перевірки прав доступу (чи є користувач адміністратором).

Стан:

- goods (масив товарів)
- editingGoods (поточний товар для редагування)
- editedPhotoFile (фото для редагування)
- error (для обробки помилок)
- showAddGoodsModal, showEditGoodsModal, showPriceModal, showDeleteConfirmationModal (для керування модальними вікнами)
- deleteItemId (для збереження id товару, який буде видалено)
- selectedRow (ідентифікатор товару для відображення його ціни)

– photoUpdateSuccess (для відображення стану оновлення фото)

– validationErrors (для обробки помилок валідації)

– formData (дані для форми)

Методи:

– openAddGoodsModal: відкриває модальне вікно для додавання нового товару.

– handleCloseModal: закриває модальне вікно для додавання товару та очищає форму.

– openEditGoodsModal: відкриває модальне вікно для редагування існуючого товару.

– handleCloseEditModal: закриває модальне вікно для редагування товару.

– handleDeleteConfirmation: відкриває модальне вікно для підтвердження видалення товару.

– handleSubmit: додає новий товар після перевірки даних та оновлює список товарів.

– handlePhotoChange: обробляє зміну фото у формах.

– handleChange: обробляє зміну полів у формах додавання та редагування.

– handleEditGoods: редагує товар і оновлює список після успішного редагування.

– handleDeleteGoods: видаляє товар зі списку після підтвердження.

– showRentPrice: відображає ціну прокату товару для вибраного товару.

Залежність між компонентами: Використовує дочірній компонент Modal для відображення модальних вікон (додавання, редагування та підтвердження видалення товарів, перегляд ціни на прокат).

4. Специфікація компонента RentalPricePage.js

Ім'я: RentalPricePage.js

Відповідальність: Відповідає за відображення та управління списком цін на прокат товарів. Компонент надає можливість додавання нових цін, редагування існуючих, а також видалення цін.

Атрибути (properties компонентів): Немає

Стан:

- prices (масив цін)
- editingPrice (поточна ціна для редагування)
- error (для обробки помилок)
- showAddPriceModal, showEditPriceModal, showDeleteConfirmationModal (для керування модальними вікнами)
- deleteItemId (для збереження id ціни, яка буде видалена)
- formData (дані для форми додавання)
- validationErrors (для збереження помилок валідації).

Методи:

- openAddPriceModal: відкриває модальне вікно для додавання нової ціни.
- handleCloseModal: закриває модальне вікно для додавання ціни та очищає форму.
- openEditPriceModal: відкриває модальне вікно для редагування існуючої ціни.
- handleCloseEditModal: закриває модальне вікно для редагування ціни.
- handleDeleteConfirmation: відкриває модальне вікно для підтвердження видалення ціни.
- handleSubmit: додає нову ціну після перевірки даних та оновлює список цін.

– handleChange: обробляє зміну полів у формах додавання та редагування.

– handleEditPrice: редагує ціну та оновлює список після успішного редагування.

– handleDeletePrice: видаляє ціну зі списку після підтвердження.

Залежність між компонентами: Використовує дочірній компонент Modal для відображення модальних вікон (додавання, редагування та підтвердження видалення).

5. Специфікація компонента RentPage.js

Ім'я: RentPage.js

Відповідальність: Відповідає за відображення та управління списком прокатів, включаючи редагування та видалення прокатів, а також перегляд товарів, взятих в оренду.

Атрибути (properties компонентів): Немає

Стан:

– rents (масив прокатів)

– editingRent (прокат для редагування)

– error (для обробки помилок), showEditRentModal, showDeleteConfirmationModal, showGoodsModal (для керування модальними вікнами)

– deleteItemId (id прокату для видалення)

– selectedVoucherForm (форма залогу)

– selectedStatusForm (статус прокату)

– selectedRow (ідентифікатор вибраного прокату)

– editingRent.rentedGoodsData (товари, взяті в оренду).

Методи:

- openEditRentModal: відкриває модальне вікно для редагування прокату.
- handleCloseEditModal: закриває модальне вікно для редагування прокату.
- handleCloseShowGoodsModal: закриває модальне вікно для перегляду товарів у прокаті.
- handleDeleteConfirmation: відкриває модальне вікно для підтвердження видалення прокату.
- handleChange: обробляє зміну полів у формі редагування прокату.
- handleEditRent: зберігає зміни у прокаті після редагування та оновлює список.
- handleDeleteRent: видаляє прокат зі списку після підтвердження.
- showGoods: показує товари, взяті у прокат, для вибраного прокату.
- formatDateForInput: форматує дати у вигляді, прийнятному для полів типу date у формі редагування.

Залежність між компонентами: Використовує дочірні компоненти Modal та Select для відображення модальних вікон (редагування прокату, перегляд товарів, підтвердження видалення) та вибору значень для форми залогу або статусу прокату.

6. Специфікація компонента AddRentPage.js

Ім'я: AddRentPage.js

Відповідальність: Компонент відповідає за створення нового прокату з можливістю додавання клієнтів, товарів, вибору форми залогу, оплати та дат прокату. Він також візуалізує на часовій шкалі (timeline) періоди оренди для різних товарів і дозволяє користувачу додавати нові прокати, взаємодіючи з товарами.

Атрибути (properties компонентів): Немає.

Стан (state):

- rents: масив прокатів, отриманих із сервера.
- goods: масив товарів для прокату.
- clients: масив клієнтів.
- selectedGoodsIds: масив ідентифікаторів обраних товарів для поточного прокату.
- selectedClientIds: масив ідентифікаторів обраних клієнтів для поточного прокату.
- selectedVoucherForm: обрана форма залогу (гроші, документи тощо).
- selectedStatusForm: обраний статус прокату (в процесі, повернено).
- formdata: об'єкт даних форми для прокату та додавання клієнтів.
- timelineGroups: групи для відображення товарів у часовій шкалі.
- timelineItems: елементи для відображення прокатів у часовій шкалі.
- validationErrors: об’єкт валідаційних помилок для форми прокату.
- validationClientErrors: об’єкт валідаційних помилок для форми клієнта.
- rentedGoodsQuantities: масив об’єктів з ідентифікатором товару та кількістю взятого у прокат товару.
- showAddRentModal: логічне значення для відображення модального вікна додавання прокату.
- showAddClientModal: логічне значення для відображення модального вікна додавання клієнта.
- showClientNotFound: логічне значення для відображення повідомлення, якщо клієнта не знайдено.
- showProductNotFound: логічне значення для відображення повідомлення, якщо товар не знайдено.
- addClientSuccess: індикатор успішного додавання клієнта.
- addProductSuccess: індикатор успішного додавання товару.
- highlightedGroup: ідентифікатор групи, яка підсвічується у timeline.

Методи:

- openAddRentModal: відкриває модальне вікно для додавання прокату та підсвічує вибраний товар у timeline.
- handleCloseModal: закриває модальне вікно для додавання прокату, очищає дані форми та скидає індикатори.
- handleRentedProductQuantityTwoChange: обробляє зміну кількості товарів для прокату, оновлюючи стан rentedGoodsQuantities.
- handleSubmit: обробляє подання форми прокату, перевіряє дані, надсилає їх на сервер для збереження та оновлює список прокатів.
- handleChange: обробляє зміну полів у формі, оновлюючи formdata.
- addGoodsToRent: додає вибраний товар до списку товарів для прокату, оновлюючи selectedGoodsIds і rentedGoodsQuantities.
- addClientToRent: додає обраного клієнта до списку клієнтів для прокату, оновлюючи selectedClientIds.
- validateForm: перевіряє правильність заповнення форми прокату, зберігаючи помилки у validationErrors.
- handleAddNewClient: відкриває модальне вікно для додавання нового клієнта.
- handleCloseClientModal: закриває модальне вікно для додавання клієнта та очищає помилки валідації клієнта.
- handleAddClientToRent: обробляє форму додавання нового клієнта, додає клієнта до бази даних і до списку клієнтів для поточного прокату.
- validateFormClient: перевіряє правильність заповнення форми для клієнта, зберігаючи помилки у validationClientErrors.
- generateColorForRent: генерує кольори для кожного прокату у timeline для візуальної ідентифікації прокатів.
- formatDate: форматує дату для відображення у формі та timeline.

Залежність між компонентами:

- Використовує компонент Modal для відображення модальних вікон (додавання прокату та додавання клієнта).
- Використовує компонент AddClientModal для додавання нового клієнта до прокату.
- Використовує компонент Select для вибору форми залогу та статусу прокату.

7. Специфікація компонента RegisterPage.js

Ім'я: RegisterPage.js

Відповідальність: Компонент відповідає за реєстрацію нового користувача з логіном, паролем та роллю (користувач або адміністратор).

Атрибути (properties компонентів): Немає

Стан:

- username (логін користувача).
- password (пароль користувача).
- role (роль користувача: user або admin).
- success (повідомлення про успішну реєстрацію).
- showPassword (стан відображення пароля).
- validationErrors (помилки валідації).
- error (повідомлення про помилку).

Методи:

- handleSubmit: обробляє відправку форми, перевіряє валідацію, надсилає запит на сервер, обробляє успіх або помилки.
- validateForm: перевіряє правильність введених даних (логін повинен бути більше ніж 1 символ і менше ніж 20, пароль повинен містити великі, малі літери та цифри, і бути не менше 8 символів).
- setShowPassword: перемикає режим відображення пароля (показувати або приховувати).

Залежність між компонентами: Використовує дочірній компонент Select для вибору ролі користувача (user або admin).

8. Специфікація компонента AuthPage.js

Ім'я: AuthPage.js

Відповідальність: Відповідає за обробку форми авторизації користувача. Відправляє дані на сервер для авторизації та обробляє відповідь, якщо авторизація успішна – перенаправляє користувача на головну сторінку.

Атрибути (properties компонентів): onLogin: функція, передана через пропси, яка обробляє процес авторизації користувача.

Стан:

- username (логін користувача).
- password (пароль користувача).
- showPassword (стан для відображення або приховування пароля).
- error (повідомлення про помилку, якщо авторизація неуспішна).

Методи:

- handleSubmit: обробляє відправку форми, передає введені дані до функції onLogin, яка виконує авторизацію. Якщо авторизація успішна, користувача перенаправляють на головну сторінку за допомогою navigate.
- setShowPassword: перемикає стан відображення пароля.

9. Специфікація компонента HomePage.js

Ім'я: HomePage.js

Відповідальність: Відповідає за відображення вітального повідомлення на головній сторінці додатку. Надає користувачу інструкції щодо використання додатку та навігації через верхнє меню.

Атрибути (properties компонентів): Немає

Методи: Немає методів, оскільки компонент є статичним і не вимагає логіки обробки стану або подій.

Залежність між компонентами: Незалежний компонент, який відображає лише текст. Може бути використаний як дочірній компонент у маршрутизації App, але сам по собі не залежить від інших компонентів.

10. Специфікація компонента Modal.js

Ім'я: Modal.js

Відповідальність: Відповідає за відображення модального вікна з можливістю його переміщення, згортання та закриття.

Атрибути (properties компонентів):

- onClose: функція для закриття модального вікна, викликається після натискання на кнопку закриття або коли модальне вікно приховане.
- children: дочірні елементи (контент, який буде відображатися всередині модального вікна).
- formdata, setformdata: properties, які передаються у дочірні елементи для управління даними форми.

Стан:

2 visible: керує відображенням модального вікна (чи вікно активне або приховане).

3 dragging: визначає, чи знаходиться модальне вікно у стані переміщення (перетягування).

4 position: координати позиції миші для обробки переміщення модального вікна.

5 collapsed: визначає, чи модальне вікно згорнуте (показує лише заголовок) або повністю розгорнуте.

Методи:

- handleMouseDown: починає процес перетягування модального вікна, зберігаючи поточні координати миші.
- handleMouseMove: обробляє рух миші, переміщує модальне вікно по екрану, оновлюючи позицію.

- handleMouseUp: зупиняє процес перетягування модального вікна.
- handleClose: закриває модальне вікно, робить його невидимим і викликає onClose.
- handleCollapse: перемикає стан згортання/розгортання модального вікна.

Залежність між компонентами: Використовує дочірні елементи (через properties children) для відображення контенту всередині модального вікна.

11. Специфікація компонента Select.js

Ім'я: Select.js

Відповідальність: Відповідає за відображення випадаючого списку (select), який дозволяє користувачу вибрати одне зі значень. Передає вибране значення назад у батьківський компонент через функцію зворотного виклику (onChange).

Атрибути (properties компонентів):

- options: масив опцій для випадаючого списку. Кожна опція містить об'єкт з полями id та value.
- value: поточне вибране значення, яке буде відображене у випадаючому списку.
- onChange: функція зворотного виклику, яка передає вибране значення назад у батьківський компонент.

Методи: handleSelectChange: обробляє зміну вибору у випадаючому списку, зчитує нове значення та викликає функцію onChange, щоб передати це значення батьківському компоненту.

Залежність між компонентами: Батьківський компонент передає в Select масив опцій (options), вибране значення (value) і функцію onChange, яка буде викликана після зміни вибору.

12. Специфікація компонента AddClientModal.js

Ім'я: AddClientModal.js

Відповідальність: Компонент AddClientModal відповідає за відображення модального вікна для додавання нового клієнта, що містить форму введення даних клієнта та валідаційні повідомлення для кожного поля.

Атрибути (properties компонентів):

- show: логічне значення, що визначає, чи показувати модальне вікно. Якщо false, компонент не рендериться.
- onClose: функція зворотного виклику, яка викликається для закриття модального вікна.
- formdata: об’єкт, що містить дані форми клієнта (прізвище, ім'я, побатькові, категорія, телефон та пошта).
- validationClientErrors: об’єкт, що містить помилки валідації для полів форми клієнта.
- handleChange: функція зворотного виклику для обробки змін у полях форми.
- handleAddClientToRent: функція, що обробляє подання форми для додавання клієнта.

Стан (state): У компонента немає власного стану (state). Всі дані та помилки отримуються через properties.

Методи:

- handleChange: функція, що передається як property, обробляє зміни у полях форми і дозволяє оновлювати formdata у батьківському компоненті.
- handleAddClientToRent: функція, що обробляє подання форми для додавання нового клієнта. Вона передається батьківським компонентом і обробляє додавання клієнта після перевірки.
- validationClientErrors: містить повідомлення про помилки валідації для кожного поля форми (прізвище, ім'я, побатькові, категорія, телефон, пошта). Помилки відображаються як error-message під відповідними полями.

Залежність між компонентами:

Використовує компонент Modal для відображення модального вікна.

13. Специфікація компонента UsersModal.js

Відповідальність: Компонент UsersModal.js відповідає за відображення модального вікна для перегляду усіх зареєстрованих користувачів та за можливість видалення обраного юзера (тільки адміністратор)

Атрибути (properties компонентів):

– onClose: функція зворотного виклику, яка викликається для закриття модального вікна.

– baseURL: посилання на сервер на якому розміщена серверна частина (Node.js)

– loggedInUserId: id юзера який авторизований на даний момент

– userRole: роль юзера який зараз авторизований

Стан (state):

– usersData: масив об’єктів-юзерів які зареєстровані в системі.

– showDeleteConfirmationModal: булеве значення яке визначає чи показувати повідомлення для підвердження видалення обраного юзера юзера

– deleteItemId: id юзера якого було вирішено видалити

– error: змінна для збереження помилок

Методи:

handleDeleteUser: функція що обробляє видалення юзера з конкретним переданим їй id

handleDeleteConfirmation: функція що відповідає за показ запиту на підтвердження видалення обраного юзера

Залежність між компонентами:

Використовує компонент Modal для відображення модального вікна.

Динамічна модель програмного забезпечення

Для автоматизації роботи менеджера та адміністратора вебзастосунку для автоматизації обліку прокату туристичного спорядження магазину «Хан-Тенгрі» динамічна модель може бути подана у вигляді діаграм послідовностей. Цей тип діаграм належить до категорії діаграм взаємодії, оскільки відображає поведінку системи з погляду взаємодії об’єктів у часовій послідовності. Таким чином, можна побачити, як і коли об’єкти викликають методи або обмінюються повідомленнями, що розкриває часові аспекти роботи системи.

Діаграми послідовностей допомагають деталізувати й уточнювати діаграми прецедентів, а також детально описувати логіку реалізації сценаріїв використання [10]. Зразок динамічної моделі ПЗ у вигляді основних діаграм послідовностей наведено на рисунках 3.9 – 3.10.

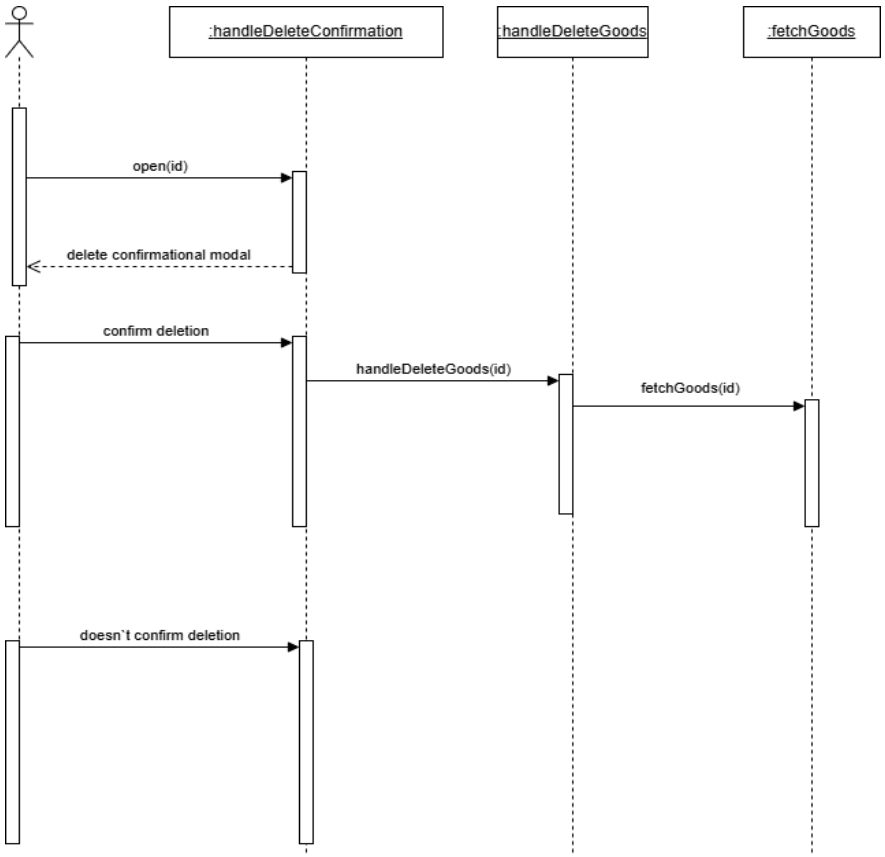


Рисунок 3.9 – Діаграма послідовності для варіанта використання «Видалення товару зі складу»

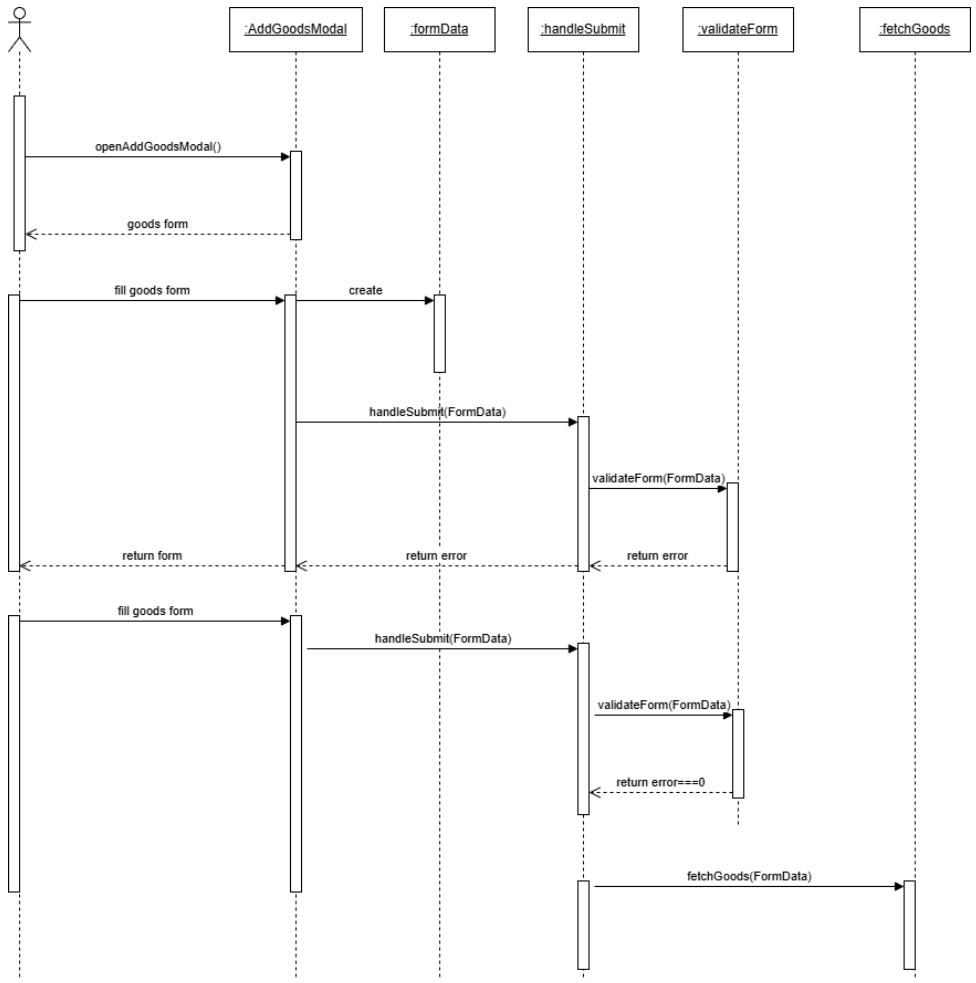


Рисунок 3.10 – Діаграма послідовності для варіанта використання «Додавання товару»

Перш ніж безпосередньо переходити до створення таблиць та інших об’єктів бази даних, слід визначити її логічну структуру. Це вкрай важливий та доволі складний етап, адже саме вдало спроєктована модель даних дає змогу розробити ефективну, надійну й масштабовану базу даних.

Для вебзастосунку для автоматизації обліку прокату туристичного спорядження магазину «Хан-Тенгрі» початок логічного проєктування бази даних базується на створеній концептуальній моделі системи. Логічна модель даних є абстрактним та концептуальним представленням інформації, яке не залежить від

конкретних технічних особливостей бази даних або системи управління базами даних (СУБД). На цьому рівні моделювання основна увага зосереджується на структурі даних, їхніх взаємозв’язках, а також на загальних принципах організації та отримання інформації [11].

Логічна модель даних вебзастосунку для автоматизації обліку прокату туристичного спорядження магазину «Хан-Тенгрі» зображена на рисунку 3.11.

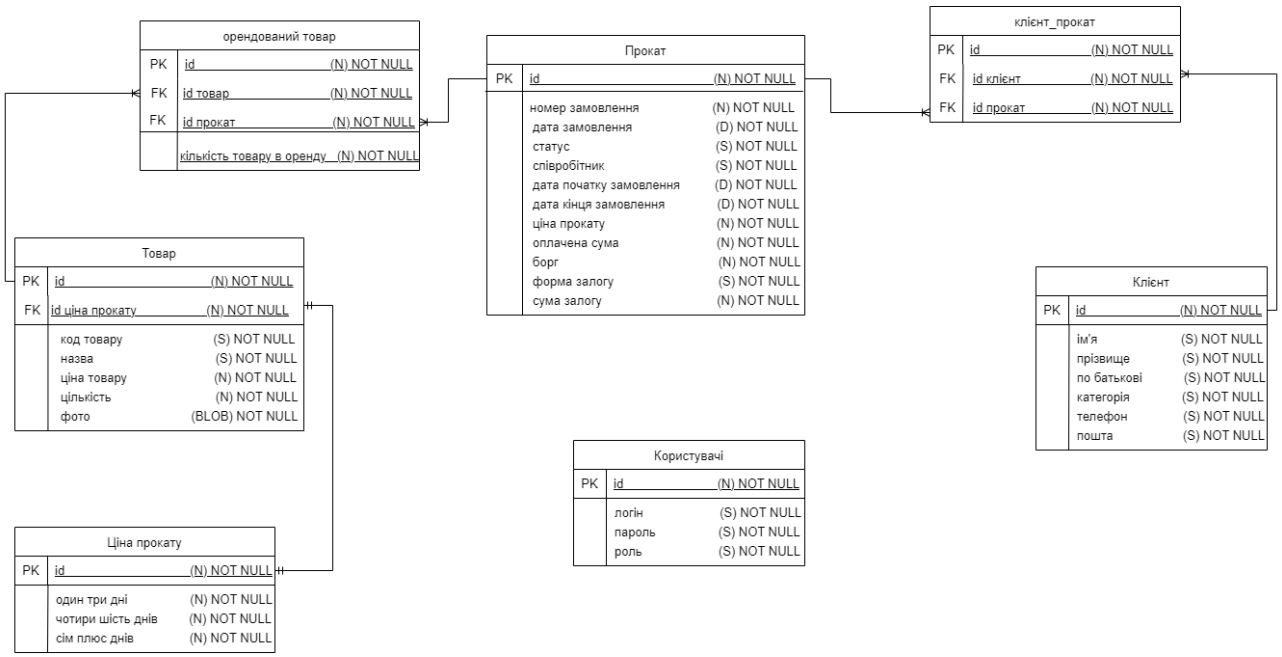


Рисунок 3.11 – Логічна модель даних вебзастосунку для автоматизації обліку прокату туристичного спорядження магазину «Хан-Тенгрі»

Після завершення роботи над технічним проєктом вебзастосунку для автоматизації обліку прокату туристичного спорядження магазину «Хан-Тенгрі», можна переходити до чергового етапу проєктування – розробки робочого проєкту цього ПЗ.

3.2.3 Робочий проект вебзастосунку для автоматизації обліку прокату туристичного спорядження магазину «Хан-Тенгрі»

Вибір мови програмування та СКБД

Для розробки проекту серед різних розробничих середовищ, таких як: IntelliJ IDEA, Eclipse, NetBeans, було обрано програмне забезпечення Microsoft Visual Studio Code (VS Code), провідне середовище для швидкої та ефективної розробки програмного забезпечення на різних мовах програмування. VS Code – це легкий, але потужний редактор вихідного коду з розвиненими засобами автоматизації, потужними інструментами рефакторингу коду, високотехнологічний набір тісно інтегрованих інструментів програмування, включно з вбудованою підтримкою Git, засобами налагодження, розширеннями для різних мов програмування, такими як Python, JavaScript, TypeScript, Java, C++ та багатьма іншими [12].

Із СУБД таких як: Percona Server for MySQL, MySQL, та Postgres, було обрано MySQL через її високу стабільність та широке застосування в промисловості. MySQL демонструє чудову продуктивність і ефективність обробки запитів, що важливо для автоматизації бізнес-процесів. Ще однією перевагою є велика спільнота розробників і доступ до численних ресурсів, що спрощує пошук рішень у разі виникнення проблем. Крім того, вона відзначається простотою інтеграції та налаштування, що значно полегшує розробку та подальше масштабування системи. Завдяки цьому MySQL вдало поєднує високі технічні характеристики з легкістю використання, що зробило її оптимальним вибором для даного проєкту [13].

Для розробки проєкту серед різних технологій для створення інтерфейсу користувача таких як: Angular, Vue.js, Ember.js, було обрано React.js, провідну бібліотеку для побудови користувацьких інтерфейсів. React.js – це потужний

інструмент для створення компонентних та динамічних вебзастосунків, що дозволяє ефективно управляти станом додатка та забезпечувати високу продуктивність завдяки використанню віртуального DOM. React.js пропонує декларативний підхід до розробки інтерфейсів, що значно спрощує створення складних UI, а також забезпечує повторне використання компонентів, що сприяє кращій організації коду та його підтримці. Використовуючи JSX, розробники можуть писати структури UI в зрозумілий спосіб, що поєднує JavaScript та HTML у одному файлі. Крім того, екосистема React включає в себе такі інструменти як React Router для управління маршрутизацією та Redux для управління станом додатка, що дозволяє створювати масштабовані та гнучкі рішення [14].

Для бекенду серед різних технологій таких як: Ruby on Rails, Django, Spring, було обрано Node.js, потужну платформу для серверної розробки на JavaScript. Node.js – це середовище виконання JavaScript, яке дозволяє створювати високопродуктивні та масштабовані серверні застосунки завдяки подієво-орієнтованій, неблокуючій моделі вводу/виводу. Node.js забезпечує високу продуктивність завдяки своїй однопоточній природі та асинхронному виконанню коду, що дозволяє обробляти багато запитів одночасно без потреби у створенні великої кількості потоків. Крім того, велика екосистема модулів та бібліотек, доступних через npm, дозволяє легко розширювати функціональність застосунків та інтегрувати різні сервіси і бази даних. Express.js, популярний веб-фреймворк для Node.js, забезпечує простоту у створенні маршрутизації та обробці HTTP-запитів, що робить розробку серверних застосунків більш зручною та ефективною [15].

Також в роботі були використані наступні засоби розробки:

– CSS – це потужний інструмент, який дозволяє визначати зовнішній вигляд елементів HTML, включаючи кольори, шрифти, розміри, відступи, вирівнювання та багато інших властивостей. CSS забезпечує можливість створення адаптивних та інтерактивних інтерфейсів користувача за допомогою

медіа-запитів, що дозволяє вебсторінкам коректно відображатися на різних пристроях та екранах [16].

– Для розробки інтерактивної логіки та функціональності веб-застосунків серед різних мов програмування таких як: Python, Ruby, PHP, я обрала JavaScript, основну мову для клієнтської розробки веб-застосунків. JavaScript – це динамічна, прототипно-орієнтована мова програмування, яка дозволяє створювати інтерактивні та динамічні веб-сторінки, додаючи до них поведінку та логіку. JavaScript забезпечує можливість маніпуляції з DOM (Document Object Model), що дозволяє динамічно змінювати вміст та структуру веб-сторінок у відповідь на дії користувача. Завдяки своїй універсальності та потужності, JavaScript широко використовується для створення як простих інтерактивних елементів, так і складних односторінкових застосунків (SPA) з використанням таких фреймворків як React.js, Angular, або Vue.js [17].

Фізична модель бази даних

Фізична модель даних – це структуроване представлення даних на рівні, який визначає, як дані зберігаються та організовані на рівні конкретної бази даних або системи управління базами даних (СУБД). Цей рівень моделювання даних концентрується на технічних аспектах зберігання і доступу до інформації [18].

Фізична модель бази даних програмного забезпечення для автоматизації обліку прокату туристичного спорядження магазину «Хан-Тенгрі» представлена на рисунку 3.12.

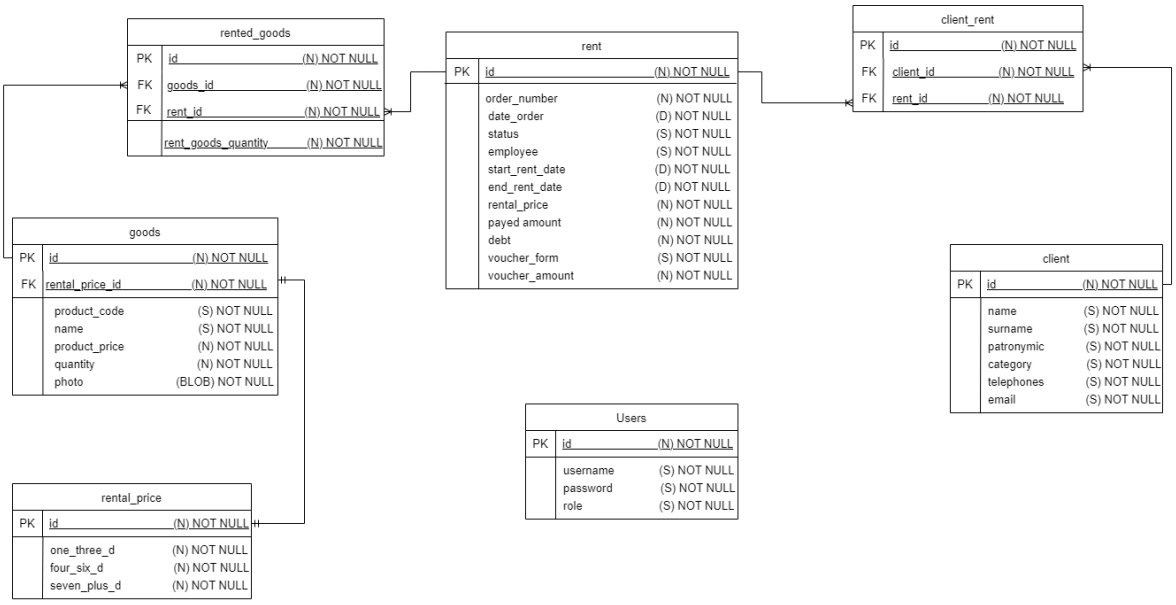


Рисунок 3.12 – Фізична модель бази даних програмного забезпечення для автоматизації обліку прокату туристичного спорядження магазину «Хан-Тенгрі»

Назви атрибутів та сутностей в фізичній та логічній моделі наведені в таблиці 3.5

Таблиця 3.5 – Таблиця назв атрибутів та сутностей в фізичній та логічній моделі

Назва таблиці в логічній моделі	Назва таблиці в фізичній моделі	Назва атрибутів в логічній моделі	Назва атрибутів в фізичній моделі
1	2	3	4
товар	goods	id ціна прокату	rental_price_id
		код товару	product_code
		назва	name
		ціна товару	product_price
		кількість	quantity
		фото	photo
ціна прокату	rental_price	один три дні	one_three_d
		чотири шість днів	four_six_d
		сім плюс днів	seven_plus_d
орендований товар	rented_goods	id товар	goods_id
		id прокат	rent_id
		кількість товару в оренду	rent_goods_quantity

Продовження таблиці 3.5

1	2	3	4
прокат	rent	номер замовлення	order_number
		дата замовлення	date_order
		статус	status
		співробітник	Employee
		дата початку замовлення	start_rent_date
		дата кінця замовлення	end_rent_date
		ціна прокату	rental_price
		оплачена сума	payed_amount
		борг	Debt
		форма залогу	voucher_form
		сума залогу	voucher_amount
клієнт_прокат	client_rent	id клієнт	client_id
		id прокат	rent_id
клієнт	client	ім’я	Name
		прізвище	Surname
		по батькові	Patronymic
		категорія	Category
		телефон	Telephones
		пошта	Email
користувачі	users	логін	Username
		пароль	Password
		роль	Role

Для написання вебзастосунку було використано такі засоби розробки:

– Мова програмування: JavaScript (ES6+).

– Середовище виконання: Node.js (версія 16+).

– IDE / редактор коду: Visual Studio Code.

– Менеджер пакетів: npm.

– Фреймворк для роботи з HTTP: Express.js (як базовий web-сервер).

– Middleware для завантаження файлів: Multer.

Розглянемо серверну частину функції «Додавання інформації про товар». Представимо код контролеру клієнта який буде тестуватися.

```
goodsController.js
const path = require('path');
```

```
const multer = require('multer');
const fs = require('fs');

module.exports = function(db, upload) {
    return {
        addGoods: (req, res) => {
            upload.single('photo')(req, res, function (err) {
                if (err instanceof multer.MulterError) {
                    console.error(err.message);
                    return res.status(400).send('Error uploading
file');
                } else if (err) {
                    console.error(err.message);
                    return res.status(500).send('Server error');
                }
                if (!req.file) {
                    return res.status(400).send('No file
uploaded');
                }

                const newGoods = req.body;
                const { product_code, name, product_price,
quantity, rental_price_id} = newGoods;
                const photo = req.file.filename;

                db.query('INSERT INTO goods (product_code, name,
product_price, quantity, photo, rental_price_id) VALUES (?, ?, ?,
?, ?, ?)',
                    [product_code, name, product_price, quantity,
photo, rental_price_id], function (err) {
                        if (err) {
                            console.error(err.message);
```

```
                            return res.status(500).send('Server
error');
                        }
                        return res.status(201).json({ id:
this.lastID });
                    });
            });
        },
    };
};
```

Наведений контролер відповідає за обробку HTTP-запиту на додавання нового товару із завантаженням фотографії і збереженням запису в базі даних. Пояснення структури і логіки роботи:

Імпорт залежностей

– Підключення модуля для роботи з шляхами файлової системи.

– Модуль для обробки multipart/form-data (завантаження файлів).

– Вбудований модуль для роботи з файловою системою.

Експорт функції-будівника контролера

Функція приймає два параметри:

– db - об’єкт для виконання SQL-запитів;

– upload - налаштований інстанс middleware для завантаження файлів.

Вона повертає об’єкт із методом addGoods.

Метод addGoods

– Ініціалізація завантаження файлу: викликає middleware, що очікує одне поле «photo».

– Обробка помилок завантаження:

Якщо це помилка самого мідлвару (обмеження розміру, некоректний тип файлу), повертається статус 400.

Якщо будь-яка інша помилка на етапі обробки, повертається статус 500.

– Перевірка наявності файлу: якщо поле із зображенням не передано, звернення також завершується з кодом 400.

– Збір даних: із тіла запиту дістаються текстові поля (код, назва, ціна, кількість, ідентифікатор тарифу), а з об’єкта запиту – ім’я збереженого файлу.

– Виконання SQL-запиту: формування параметризованої вставки запису в таблицю товарів.

У разі помилки бази даних – відповідь зі статусом 500.

При успішній вставці – відповідь зі статусом 201 та ідентифікатором нового запису.

Ключові принципи дизайну

– Використання middleware для розділення завантаження файлів і бізнес-логіки.

– Чітка обробка всіх можливих помилок на рівнях: мережевому (400) та серверному (500).

– Параметризовані SQL-запити, які запобігають SQL-ін’єкціям.

– Повернення REST-коректних статусів і мінімальної інформації в тілі відповіді.

Основна частина коду представлена в додатку Б.

Реалізація та тестування основних класів еквівалентності вебзастосунку для автоматизації обліку прокату туристичного спорядження магазину «Хан-Тенгрі»

Тестування програмного забезпечення (ПЗ) - це процес перевірки та перевірки відповідності програмного продукту вимогам та очікуванням. Існує кілька способів тестування ПЗ, включаючи:

– Модульне тестування: Тестування окремих модулів або компонентів програми для перевірки їхньої коректності та взаємодії.

– Інтеграційне тестування: Перевірка правильності взаємодії між різними модулями або компонентами програми під час їх об'єднання.

– Системне тестування: Тестування програмного продукту в цілому, перевірка його відповідності специфікаціям та вимогам.

Приймальне тестування: Верифікація програмного продукту замовником для підтвердження його готовності до використання.

Тестування функції методом розбиття на еквівалентності:

Класи еквівалентності функції «Додавання інформації про товар» представлені в таблиці 3.6.

Таблиця 3.6 – Класи еквівалентності функції «Додавання інформації про товар»

Вхідні дані	Правильні класи еквівалентності	Неправильні класи еквівалентності
код товару	1. числа, символи	
назва	2. символ	
ціна товару	3. числа	4. символ (крім чисел)
кількість на складі	5. числа	6. символ (окрім чисел)
фото	7. jpg (png)	
id ціна прокату	8. числа	9. символ (крім чисел)

Почнемо з тестування правильних класів еквівалентності які наведені в таблиці 3.7. За результатами тестування вебзастосунок працює правильно. Результат тестування представлено на рисунку 3.13.

Таблиця 3.7 – Тести для правильних класів еквівалентності

№ теста	№ покритого класу	Вхідні умови	Результат
1	1	7777.888	результат представлений на рисунку 3.9
	2	tent	
	3	500	
	5	13	
	7	tent.png	
	8	1	

Рисунок 3.13 – Результат перевірки правильних класів еквівалентності

Тепер протестуємо неправильні класи еквівалентності які представлені у таблиці 3.8. За результатами тестування вебзастосунок коректно оброблює не правильні значення та показує правильні повідомлення про помилки. Результат наведено на рисунку 3.14.

Таблиця 3.8 – Тести для неправильних класів еквівалентності

№ теста	№ покритого класу	Вхідні дані	Результат
1	4	ннн	Введіть число (десяткове значення дозволено)
2	6	нн	Введіть ціле число
3	9	н	Введіть ціле число

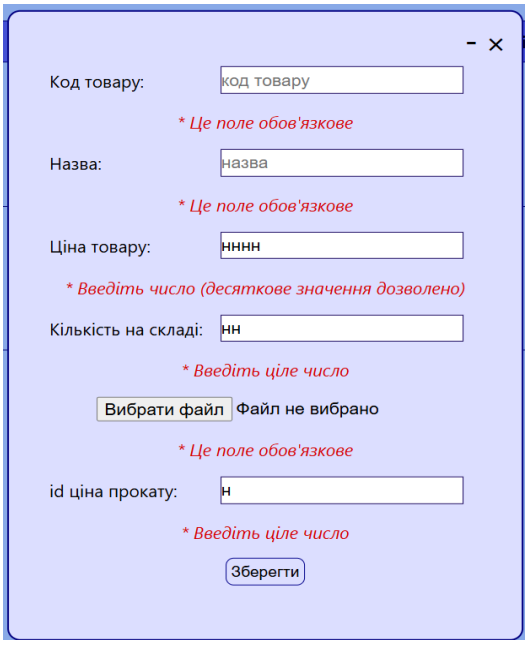


Рисунок 3.14 – Результат перевірки неправильних класів еквівалентності

Тестування вебзастосунку для автоматизації обліку прокату туристичного спорядження магазину «Хан-Тенгрі» виконувалося методом «чорного ящика». Тестування виконувалося за рахунок введення тестових наборів даних та аналізу отриманих результатів.

4 РЕЗУЛЬТАТИ РОЗРОБКИ ВЕБЗАСТОСУНКУ ДЛЯ АВТОМАТИЗАЦІЇ ОБЛІКУ ПРОКАТУ ТУРИСТИЧНОГО СПОРЯДЖЕННЯ МАГАЗИНУ «ХАН-ТЕНГРІ»

У межах цієї кваліфікаційної роботи була розв’язана практична задача інженерії програмного забезпечення, що характеризується комплексністю та невизначеністю умов, а саме розробка вебзастосунку для туристичного магазину «Хан-Тенгрі». Було створено вебзастосунок для автоматизації обліку прокату туристичного спорядження, що дозволить підвищити зручність і ефективність роботи менеджерів, відповідальних за його організацію та управління, а також зменшити кількість помилок у процесі обліку та видачі спорядження.

Зовнішній вигляд головної сторінки з прокатами показаний на рисунку 4.1.

Рисунок 4.1 – Зовнішній вигляд головної сторінки з прокатами вебзастосунку для автоматизації обліку прокату туристичного спорядження магазину «Хан-Тенгрі»

Було проведено аналіз практичної задачі інженерії програмного забезпечення, а саме досліджено роботу магазину туристичного спорядження «Хан-Тенгрі» з урахуванням уже наявних систем автоматизації в магазині та проаналізовано сучасні тенденції автоматизації обліку прокату туристичного спорядження. Було обґрунтовано доцільність розробки власного програмного забезпечення для автоматизації повсякденних операцій адміністратора та менеджера магазину.

В рамках ескізного проєкту засобами мови моделювання UML були побудовані концептуальна модель програмного забезпечення та діаграми діяльності. Також розроблений ескіз інтерфейсу.

В рамках технічного проєкту побудована логічна модель бази даних, діаграма компонентів та динамічна модель програмного забезпечення у вигляді діаграм послідовностей.

В рамках робочого проєкту виконано вибір і обґрунтування інструментів розробки та системи управління базою даних, побудована фізична модель бази даних, виконано кодування та тестування програмного забезпечення.

Основною мовою програмування було обрано JavaScript. В якості СКБД для роботи з базою даних було обрано MySQL. Середовищем розробки було обрано Microsoft Visual Studio Code (VS Code).

Було виконано тестування та випробування вебзастосунку, яке показало його повну працездатність та відповідність вимогам, наведеним у технічному завданні.

Також була розроблена наступна програмна документація: методика приймання, опис програми, інструкція користувача і методика випробувань. Вебзастосунок реалізує такі функції:

– реєстрація користувачів;
– авторизація користувачів;

– перегляд та обробка інформації про товари на складі (додавання, видалення, редагування);

– перегляд та обробка інформації про клієнтів (додавання, видалення, редагування);

– перегляд та обробка інформації про ціни прокату (додавання, видалення, редагування);

– створення замовлення на прокат (додавання усієї інформації щодо прокату, пошук товару та клієнта, внесення нового клієнта у разі його відсутності);

– видалення та редагування замовлення на прокат;

– перегляд часової стрічки (відображення проміжку від початкової до кінцевої дати прокату кожного товару що був взятий в прокат з урахуванням кількості цього товару);

– оновлення статусу замовлення на прокат на «повернено» (зникнення маркування з часової стрічки, збереження запису в базі);

– перегляд списку усіх товарів, записаних у конкретний прокат;

– перегляд інформації про ціну прокату, пов’язаної з кожним товаром;

– перегляд та видалення зареєстрованих користувачів (менеджерів);

– вихід з облікового запису.

Вебзастосунок було розроблено для магазину туристичного спорядження «Хан-Тенгрі»

Програмне забезпечення впроваджено, акт впровадження наведено в додатку Е. Програму і методику випробувань наведено у Додатку Д.

Протокол випробувань

Під час перевірки було здійснено повний цикл функціонального тестування, а також проведено навантажувальні випробування та перевірку на відмову всього програмно-апаратного комплексу. Усі виявлені недоліки програмного

забезпечення детально занесені до протоколів і виправлені ще до моменту впровадження вебзастосунку в експлуатацію.

Методи випробувань

Тестування виконували за стратегією «чорної скриньки». З огляду на широку функціональність системи, наведено приклади результатів перевірки кількох ключових функцій.

1. Перевірка відповідності розробленого програмного забезпечення технічному завданню.

– перевірка роботи «Додавання товару»

На сторінці товарів натиснути кнопку «Додати товар». З’являється форма з необхідними полями для додавання товару. Після введення необхідної інформації натискається кнопка «Зберегти».

– перевірка роботи «Редагування товару»

На сторінці товарів натиснути кнопку «Редагувати» напроти товару, який потрібно редагувати. З’являється форма з необхідними полями для редагування товару. Після введення необхідної інформації натискається кнопка «Зберегти».

2. Здійснено перевірку програмно-апаратної сумісності: проведено тестування на різних апаратно-програмних платформах, які задовольняють щонайменше мінімальні вимоги, викладені у технічному завданні. Виявлені недоліки, згадані у пункті 1, перевірено та виправлено.

3. Проведено кінцевий візуальний аналіз програми, під час якого виконано остаточне налагодження з метою виявлення та усунення дрібних помилок.

5 ОХОРОНА ПРАЦІ

5.1 Основні законодавчі і нормативні документи, які регламентують охорону праці в Україні

Охорона праці в Україні є важливою складовою державної політики, спрямованої на забезпечення безпечних та здорових умов праці для всіх працівників. Законодавча та нормативна база, що регулює охорону праці, включає низку документів, які визначають права та обов'язки роботодавців і працівників, а також стандарти безпеки на робочих місцях. Основними законодавчими та нормативними документами, які регламентують охорону праці в Україні, є:

– Конституція України: Стаття 43 гарантує кожному право на належні, безпечні та здорові умови праці, а також на соціальний захист у разі втрати працездатності [19].

– Кодекс законів про працю України (КЗпП): Головний законодавчий акт, що регулює трудові відносини в Україні. Розділ XI КЗпП присвячений охороні праці, визначає обов'язки роботодавців щодо забезпечення безпеки праці та права працівників на охорону праці [20].

– Закон України "Про охорону праці": Основний закон, що визначає правові, економічні та соціальні засади охорони праці в Україні. Закон зобов'язує роботодавців створювати безпечні умови праці, проводити навчання та інструктажі з питань охорони праці, забезпечувати засобами індивідуального захисту тощо [21].

– Закон України "Про загальнообов'язкове державне соціальне страхування": Визначає порядок страхування працівників від нещасних випадків на виробництві та професійних захворювань, порядок надання страхових виплат і соціальних послуг [22].

– Державні нормативно-правові акти з охорони праці (ДНАОП): Нормативні акти, що містять конкретні вимоги та правила щодо безпеки праці в різних галузях. ДНАОП розробляються Міністерством соціальної політики та іншими органами виконавчої влади.

– Гігієнічні нормативи та санітарні правила: Встановлюють вимоги до санітарно-гігієнічних умов на робочих місцях, включаючи рівні шуму, освітленості, мікроклімату, хімічних та біологічних факторів.

– Стандарти безпеки праці (ДСТУ): Національні стандарти, що регулюють безпеку праці, розробляються Державним комітетом України з питань технічного регулювання та споживчої політики.

– Інструкції з охорони праці: Локальні нормативні документи, які розробляються безпосередньо на підприємствах і містять конкретні вимоги щодо безпечного виконання робіт для різних професій та видів робіт.

– Положення про систему управління охороною праці (СУОП): Визначає організаційні основи системного управління охороною праці на підприємствах, установах та організаціях.

Дотримання цих законодавчих та нормативних документів забезпечує належний рівень безпеки на робочих місцях, зменшує ризики виробничого травматизму та професійних захворювань, а також сприяє створенню здорових та безпечних умов праці для всіх працівників в Україні.

5.2 Структура управління питаннями охорони праці на підприємстві "Хан-Тенгрі"

Управління питаннями охорони праці в магазині туристичного спорядження "Хан-Тенгрі" є важливою складовою загальної системи управління підприємством. Основною метою системи управління охороною праці є

забезпечення безпечних та здорових умов праці для всіх працівників магазину, запобігання виробничому травматизму та професійним захворюванням. Структура управління питаннями охорони праці на підприємстві включає кілька ключових елементів та рівнів відповідальності, які діють узгоджено для досягнення комплексного результату.

Однією з визначальних рис цієї структури є чіткий розподіл функцій між керівництвом та персоналом, що дозволяє ефективно взаємодіяти в процесі планування, організації та здійснення заходів із охорони праці. Крім того, завдяки постійному моніторингу за дотриманням встановлених вимог безпеки вдається оперативно реагувати на будь-які порушення і запобігати можливим ризикам. Нижче наведено основні посади та обов’язки працівників, які відповідають за реалізацію системи управління охороною праці в магазині «Хан-Тенгрі».

Директор магазину:

– несе загальну відповідальність за стан охорони праці на підприємстві;

– забезпечує фінансування заходів з охорони праці;

– приймає рішення про призначення відповідальних осіб за охорону праці;

– контролює виконання законодавчих і нормативних вимог з охорони праці.

Інженер з охорони праці:

– відповідає за розробку, впровадження та контроль виконання заходів з охорони праці;

– проводить регулярні перевірки умов праці та виявлення потенційних ризиків;

– організовує навчання та інструктажі з охорони праці для працівників;

– забезпечує ведення документації з охорони праці.

Діяльність інженера з охорони праці значною мірою визначає, наскільки ефективно у магазині дотримуються встановлених правил безпеки. За допомогою регулярних перевірок та аудиту робочих місць він виявляє можливі недоліки в організації праці й оперативно розробляє рекомендації для їх усунення. Організовані ним інструктажі та навчання формують у персонала розуміння важливості дотримання вимог охорони праці і сприяють загальному підвищенню культури безпеки в колективі.

Співробітники:

– зобов'язані дотримуватися вимог охорони праці, встановлених на підприємстві;

– виконують інструкції та правила безпеки під час роботи;

– повідомляють керівництво про будь-які небезпечні умови праці або нещасні випадки.

Навчання та інструктажі:

– всі працівники проходять первинний, повторний, позаплановий та цільовий інструктажі з охорони праці;

– проводяться регулярні навчання та тренування з питань охорони праці, евакуації при пожежі та надання першої медичної допомоги.

Організація навчальних заходів має систематичний характер і спрямована на підтримку належного рівня обізнаності персоналу щодо вимог безпеки. Проведення тренувань із евакуації та надання першої медичної допомоги забезпечує готовність працівників до оперативних дій у разі надзвичайних ситуацій. Такий підхід допомагає мінімізувати ризики для життя та здоров’я, а також гарантує чітку координацію дій у стресових обставинах.

Документація з охорони праці:

– ведеться облік всіх заходів з охорони праці, включаючи журнали інструктажів, акти перевірок, плани заходів з покращення умов праці;

– розробляються локальні нормативні акти, положення та інструкції з охорони праці, які відповідають законодавчим вимогам.

Структура управління охороною праці в магазині "Хан-Тенгрі" забезпечує комплексний підхід до створення безпечних умов праці, з чітко визначеними відповідальностями на всіх рівнях управління, що сприяє підвищенню рівня безпеки та захисту здоров'я працівників.

5.3 Розробка заходів по зменшенню дії небезпечних і шкідливих факторів на підприємстві "Хан-Тенгрі"

У відділі розробки програмного забезпечення магазину туристичного спорядження «Хан-Тенгрі» найбільш характерними шкідливими та небезпечними виробничими факторами є тривала робота за комп’ютером, підвищені вимоги до зору, електричні ризики при роботі з офісною технікою, а також необхідність дотримання правил безпеки при поводженні з туристичним спорядженням, яке може знадобитися для демонстрації чи тестування в процесі роботи.

Для створення безпечних і комфортних умов праці слід упроваджувати як організаційні, так і технічні заходи.

Забезпечення ергономічних умов праці:

– Монітори мають відповідати стандартам із вертикальною частотою розгортки не менше ніж 75 Гц, щоб зменшити навантаження на зір.

– Оптимальні мікрокліматичні показники: Рекомендовано підтримувати температуру в робочому приміщенні на рівні 23 °C у теплу пору року та близько 18 °C у холодну, з обов’язковим провітрюванням приміщень.

– Ергономіка робочого місця: Регульовані столи й крісла (з підтримкою попереку та можливістю налаштування висоти). Верхній край монітора слід розташовувати на рівні очей або трохи нижче.

Дотримання санітарно-гігієнічних норм:

– Вологе прибирання: Рекомендується проводити щодня, щоб зменшити кількість пилу та підтримувати чистоту приміщення.

– Кондиціонування й вентиляція: Встановлення кондиціонера або системи вентиляції для підтримання оптимальної температури й вологості повітря.

– Освітлення: У темний час доби варто використовувати люмінесцентні або LED-лампи з високим коефіцієнтом світловіддачі (75+ лм/Вт) і тривалим строком служби (понад 10 000 год).

Дотримання правил поводження з туристичним спорядженням:

– Безпечне зберігання: Туристичне обладнання (намети, рюкзаки, спальні мішки тощо), яке може перебувати у відділі розробки ПЗ для тестування або демонстрацій, повинне зберігатися в окремому приміщенні чи шафах із забезпеченням належних умов (сухість, належна вентиляція).

– Правила користування: Співробітники, які працюють із спорядженням, мають проходити короткий інструктаж щодо правил складання/розкладання, уникнення травмонебезпечних ситуацій (наприклад, порізи від гострих деталей, підйом важких предметів), аби запобігти травмам.

– Контроль стану та справності: Рекомендується періодично перевіряти спорядження на наявність пошкоджень (дірки, розірвані шви, зламані застібки тощо), аби уникнути виникнення аварійних ситуацій.

Електробезпека та протипожежні заходи:

– Пристрої заземлення: Необхідно забезпечити якісне заземлення комп’ютерів та периферійних пристроїв.

– Захисна автоматика: Використання пристроїв захисного відключення (ПЗВ) і мережевих фільтрів знижує ризик короткого замикання й пошкодження техніки.

– Протипожежні засоби: Рекомендується встановити порошкові вогнегасники, димові датчики та підлогу з негорючих матеріалів, а також розмістити план евакуації на видному місці.

Організація робочого часу та профілактика перевтоми:

– Регулярні перерви: Слід робити короткі перерви кожні 1–2 години, аби зменшити навантаження на очі та м’язи (гімнастика для рук і шиї).

– Планування завдань: Використання систем управління проєктами (трекерів завдань, календарів) допомагає рівномірно розподіляти навантаження та мінімізувати стрес.

– Психологічний клімат: Підтримання дружньої атмосфери в колективі, проведення мотиваційних заходів та тренінгів із боротьби зі стресом.

Навчання й контроль з питань охорони праці:

– Інструктажі: Регулярні вступні та повторні інструктажі з охорони праці, включно з правилами роботи за комп’ютером та безпечного поводження зі спорядженням.

– Планові перевірки: Періодичний огляд робочих місць, перевірка стану офісної та туристичної техніки, аналіз зауважень працівників.

– Документація: Розробка й оновлення локальних інструкцій, актів і журналів реєстрації інцидентів, що забезпечує системний підхід до охорони праці.

Упровадження наведених заходів у відділі розробки програмного забезпечення «Хан-Тенгрі» сприяє зменшенню впливу небезпечних і шкідливих факторів, а також підвищує ефективність і безпеку праці співробітників, створюючи комфортне робоче середовище навіть за умови періодичного використання туристичного спорядження.

ВИСНОВКИ

В ході виконання кваліфікаційної роботи було вирішено практичну задачу інженерії програмного забезпечення, що характеризується комплексністю та невизначеністю умов, а саме виконано розробку вебзастосунку для автоматизації обліку прокату туристичного спорядження в магазині «Хан-Тенгрі». Зокрема:

– проаналізовано роботу магазину «Хан-Тенгрі», зокрема сучасні тенденції автоматизації процесів прокату туристичного спорядження;

– зроблено висновки щодо доцільності розробки власного програмного забезпечення для автоматизації обліку прокату туристичного спорядження магазину «Хан-Тенгрі»;

– проаналізовано існуючі аналоги з метою визначення їх основного функціоналу, виявлення переваг і недоліків, а також порівняння з потенційними можливостями власної розробки;

– сформульовано постановку задачі на розробку програмного забезпечення для автоматизації обліку прокату туристичного спорядження;

– проведено аналіз вимог до програмного забезпечення для автоматизації обліку прокату туристичного спорядження магазину «Хан-Тенгрі»;

– розроблено проєкт програмного забезпечення, який включає всі технічні та графічні аспекти майбутнього рішення;

– здійснено реалізацію та тестування програмного забезпечення для автоматизації обліку прокату туристичного спорядження магазину «Хан-Тенгрі»;

– розроблено всю необхідну програмну документацію.

Окремо були розглянуті загальні питання охорони праці.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Туристична галузь України під час війни: виклики, загрози, шляхи виходу з кризи. URL: https://fakultet.site/data/monograph2024.pdf (дата звернення: 10.03.2025)

2. Інтернет-магазин Хан-Тенгрі - товари для туризма та відпочинку. URL:]https://hantengry.com.ua/?srsltid=AfmBOopDwTjYVSWykTJe2N5a3ugh6T2WkedYW_V1xW_Rc4fqAiMr2MfO (дата звернення: 10.03.2025)

3. Програма обліку оренди та прокату. URL: https://remonline.ua/hire-and-rental/ (дата звернення: 10.02.2025)

4. Програма для бліку оренди та прокату: нерухомості, аксесуарів, авто, одягу, рекламних щитів, інвентарю. URL: https://shop.as-service.com.ua/specialized_software/rent_and_hire (дата звернення: 10.02.2025)

5. FASTPANEL. URL: https://fastpanel.direct/ (дата звернення:10.02.2025)

6. N-рівнева архітектура. URL: https://www.vpnunlimited.com/ua/help/cybersecurity/n-tier-architecture? (дата звернення: 23.04.2025 р.)

7. Діаграма розгортання. URL: https://www.guru99.com/uk/deployment-diagram-uml-example.html (дата звернення: 23.04.2025 р.)

8. Діаграма варіантів використання. URL: https://www.maxzosim.com/use-cases-and-scenarios/ (дата звернення: 23.04.2025 р.)

9. Концептуальна модель. URL: https://studfile.net/preview/9336767/page:7/ (дата звернення: 23.04.2025 р.)

10. Діаграма послідовностей. URL: https://www.maxzosim.com/sequence-diagrams/ (дата звернення: 23.04.2025 р.)

11. Логічна модель даних. URL: https://data-life-ua.com/db/lohichna-model-danykh/ (дата звернення: 23.04.2025 р.)

12. Visual Studio Code. URL: https://code.visualstudio.com/ (дата звернення: 23.04.2025 р.)

13. MySQL. URL: https://www.mysql.com/ (дата звернення: 23.04.2025. р.)

14. React.js. URL: https://react.dev/ (дата звернення: 23.04.2025. р.)

15. Node.js. URL: https://nodejs.org/en (дата звернення: 23.04.2025. р.)

16. CSS. URL: https://w3schoolsua.github.io/css/index.html#gsc.tab= (дата звернення: 23.04.2025. р.)

17. JavaScript. URL: https://uk.javascript.info/ (дата звернення: 23.04.2025. р.)

18. Концептуальні, логічні та фізичні моделі даних. URL: https://data-life-ua.com/db/kontseptualni-lohichni-ta-fizychni-modeli-danykh/ (дата звернення: 23.04.2025. р.)

19. Конституція України. URL: https://zakon.rada.gov.ua/laws/show/254%D0%BA/96%D0%B2%D1%80#Text (дата звернення: 10.03.2025 р.)

20. Кодекс законів про працю України. URL: https://zakon.rada.gov.ua/laws/show/322-08#Text (дата звернення: 10.03.2025 р.)

21. Закон України "Про охорону праці". URL: http://profspilkaosvity.org.ua/okhorona-praci/zakon-pro-okhoronu-praci/ (дата звернення: 10.03.2025 р.)

22. Закон України "Про загальнообов’язкове державне соціальне страхування". URL: https://zakon.rada.gov.ua/laws/show/1105-14#Text (дата звернення: 10.03.2025 р.)

Додаток А Технічне завдання на розробку веб-застосунку для автоматизації обліку прокату туристичного спорядження у магазині «Хан-Тенгрі».

Вступ

Повна назва продукту: веб-застосунок для автоматизації обліку прокату туристичного спорядження у магазині «Хан-Тенгрі». Область застосування – магазин туристичного спорядження «Хан-Тенгрі».

1 Підстава для розробки

Підставою для розробки даного програмного забезпечення є наказ на затвердження тем кваліфікаційних робіт бакалаврів зі спеціальності 121 Інженерія програмного забезпечення в Національному університеті кораблебудування ім. адмірала Макарова.

2 Призначення розробки

– Експлуатаційне призначення

Експлуатаційне призначення програмного забезпечення для автоматизації обліку прокату туристичного спорядження у магазині «Хан-Тенгрі» полягає в автоматизації та оптимізації процесів обліку товарів, оформлення прокату, фінансових розрахунків та взаємодії з клієнтами. Це забезпечує ефективне управління ресурсами прокатного пункту, знижує можливість помилок, підвищує швидкість обслуговування та полегшує роботу співробітників, роблячи процес прокату зручнішим для всіх учасників.

– Функціональне призначення

Функціональне призначення полягає в можливості користувачів виконувати такі функції:

– реєстрація користувачів;

– авторизація користувачів;

– перегляд та обробка інформації про товари на складі (додавання, видалення, редагування);

– перегляд та обробка інформації про клієнтів (додавання, видалення, редагування);

– перегляд та обробка інформації про ціни прокату (додавання, видалення, редагування);

– створення замовлення на прокат (додавання усієї інформації щодо прокату, пошук товару та клієнта, внесення нового клієнта у разі його відсутності);

– видалення та редагування замовлення на прокат;

– перегляд часової стрічки (відображення проміжку від початкової до кінцевої дати прокату кожного товару що був взятий в прокат з урахуванням кількості цього товару);

– оновлення статусу замовлення на прокат на «повернено» (зникнення маркування з часової стрічки, збереження запису в базі);

– перегляд списку усіх товарів, записаних у конкретний прокат;

– перегляд інформації про ціну прокату, пов’язаної з кожним товаром;

– перегляд та видалення зареєстрованих користувачів (менеджерів);

вихід з облікового запису.

3 Вимоги до програмного забезпечення

3.1 Вимоги до функціональних характеристик

3.1.1 Вимоги до складу виконуваних функцій

Додатком будуть користуватися два види користувачів: адміністратор (admin) та менеджер (user). Менеджер може виконувати усі функції адміністратора крім:

– додавання та редагування інформації про товари на складі (тільки адміністратор);

– перегляд та видалення зареєстрованих менеджерів (тільки адміністратор).

Нижче функції які доступні адміністратору

Перегляд та видалення

– реєстрація користувачів;

Вхідні дані: логін, пароль, роль

Результат: Створення нового облікового запису в системі.

– авторизація користувачів;

Вхідні дані: Логін, пароль, роль (підтягується автоматично згідно з даними вже зареєстрованого користувача)

Результат: Вхід у систему з правами доступу ролі яка була визначена під час реєстрації (admin/user)

– перегляд та обробка інформації про товари на складі (додавання, видалення, редагування);

Вхідні дані: Код товару, фото, назва, кількість, ціна.

Результат: Додавання, видалення або редагування товарів на складі. (додавання/редагування доступне тільки admin)

– перегляд та обробка інформації про клієнтів (додавання, видалення, редагування);

Вхідні дані: Прізвище, ім'я, по батькові, категорія, телефон, електронна пошта.

Результат: Додавання, видалення або редагування даних клієнтів.

– перегляд та обробка інформації про ціни прокату (додавання, видалення, редагування);

Вхідні дані: Назва товару, ціни за 1-3, 4-6, 7+ днів.

Результат: Додавання, видалення або редагування цін прокату для різних термінів.

– створення замовлення на прокат (додавання усієї інформації щодо прокату, пошук товару та клієнта, внесення нового клієнта у разі його відсутності);

Вхідні дані: номер замовлення, прізвище клієнта, дата замовлення, статус прокату, прізвище співробітника що оформлював прокат, дата початку прокату, дата кінця прокату, ціна прокату, оплачена сума, борг, форма залогу, сума залогу, кількість товару в прокат

Результат: Створення нового запису прокату (додавання клієнта та товарів).

– видалення та редагування замовлення на прокат;

Вхідні дані: номер замовлення, прізвище клієнта, статус прокату, прізвище співробітника що оформлював прокат, дата початку прокату, дата кінця прокату, ціна прокату, оплачена сума, форма залогу, сума залогу, кількість товару в прокат

Результат: видалення або редагування прокату.

– перегляд часової стрічки (відображення проміжку від початкової до кінцевої дати прокату кожного товару що був взятий в прокат з урахуванням кількості цього товару);

Вхідні дані: Немає.

Результат: Відображення зайнятості товару у часі з маркуванням кількості конкретного товару взятого в прокати.

– оновлення статусу замовлення на прокат на «повернено» (зникнення маркування з часової стрічки, збереження запису в базі);

Вхідні дані: Статус ("повернено").

Результат: Оновлення статусу прокату, зникнення маркування на часовій стрічці.

– перегляд списку усіх товарів, записаних у конкретний прокат;

Вхідні дані: id прокату.

Результат: Перегляд списку товарів, що були взяті в прокат за певним замовленням.

– перегляд інформації про ціну прокату, пов’язаної з кожним товаром;

Вхідні дані: id товару.

Результат: Відображення пов’язаної ціни прокату.

– перегляд та видалення зареєстрованих менеджерів (доступне тільки admin)

Вхідні дані для видалення: id user.

Результат для видалення: Відображення списку менеджерів без обраного (видаленого) user.

– вихід з облікового запису.

Вхідні дані: Немає.

Результат: Завершення сесії адміністратора/менеджера.

3.1.2 Вимоги до організації вхідних та вихідних даних

Дані зберігаються у базі даних MySQL.

Вхідні дані: введення даних здійснюється за допомогою пристроїв введення (клавіатура та миша). Також дані можуть вводитися вручну або обиратися із наявних випадаючих списків.

Вихідні дані: виведення даних відбувається на моніторі в режимі реального часу.

3.2 Вимоги до надійності

Надійне (стійке) функціонування програми має бути забезпечене виконанням сукупності організаційно-технічних заходів, перелік якого подано нижче:

– резервне копіювання даних: регулярне створення резервних копій бази даних і конфігураційних файлів, а також забезпечення їх зберігання на віддалених серверах чи в безпечних об'єктах для запобігання втраті даних у разі аварії;

– моніторинг та логування: встановлення системи моніторингу, яка відстежує стан програми та обладнання серверів, а також системи логування, які реєструють інформацію про події та помилки для швидкого виявлення проблем;

– резервне живлення та джерела живлення: забезпечення надійного живлення серверів та інших інфраструктурних компонентів, включаючи використання джерел живлення з резервними джерелами, які можуть переключитися у разі відмови основного джерела.

3.3 Вимоги до експлуатації

Кліматичні умови експлуатації, при яких повинні забезпечуватися задані характеристики, повинні відповідати вимогам, що пред’являються до технічних засобів в частині умов їх експлуатації.

Користувач повинен мати базові навички роботи з ноутбуком або комп’ютером.

3.4 Вимоги до складу та параметрів технічних засобів

Для роботи з ПЗ необхідна система наступної мінімальної конфігурації:

– процесор: двоядерний із тактовою частотою не менше 2,0 ГГц;

– оперативна пам'ять – 4GB;

– системний диск: 100 ГБ вільного дискового простору;

– необхідно мережеве з'єднання з виходом в інтернет.

3.5 Вимоги до інформаційної та програмної сумісності

– сумісність з операційними системами: Операційна система не нижче Windows 10;

– сумісність з базами даних: Наявність системи управління базами даних (MySQL), що використовується для зберігання даних;

– сумісність з веб-браузерами: Оскільки програма має веб-інтерфейс, то вона повинна бути сумісною з веб-браузерами, такими як Google Chrome, Mozilla Firefox, Opera.

Для розгортання веб-застосунку використовується хостинг із встановленою панеллю керування FastPanel або аналогічним програмним забезпеченням, що забезпечує зручне адміністрування серверних ресурсів [5]. Основними вимогами до середовища виконання є:

– наявність стабільного інтернет-з’єднання, оскільки вся взаємодія із системою відбувається онлайн;

– сервер із параметрами, достатніми для коректної роботи веб-додатку (рекомендовано принаймні 1 ядро процесора, 2 GB оперативної пам’яті та 10 GB вільного дискового простору);

– операційна система на сервері, сумісна з обраними технологіями (наприклад, Linux-дистрибутив з підтримкою веб-серверів Apache або Nginx);

– наявність підтримуваної версії веб-сервера (Apache або Nginx), що забезпечує обробку HTTP-запитів;

– підтримка необхідної версії середовища виконання (Node.js).

Крім того, для повноцінної роботи з веб-застосунком на стороні користувача потрібно мати сучасний веб-браузер (Google Chrome, Mozilla Firefox, Opera) з актуальними оновленнями та доступ до мережі Інтернет.

3.6 Вимоги до маркування та пакування

Продукт необхідно підготувати для розгортання (деплою) на хостингу з панеллю FastPanel. Для цього проект повинен бути збудований (build) та залитий на хостинг разом з серверною частиною.

Файл із описом ReadMe.txt (містить інструкції з розгортання, налаштування та використання програми).

Файл rent.sql, у якому наведено структуру бази даних (для імпорту на сервері баз даних).

4 Вимоги до програмної документації

Склад програмної документації повинен включати в себе:

- технічне завдання;
- програму та методику випробувань;
- керівництво користувача;
- опис програми;
- код програми.

5 Техніко-економічні показники

У даній роботі техніко-економічні показники не розраховуються.

6 Етапи роботи

Етапи роботи представлені у таблиці А.1

Таблиця А.1 – Етапи роботи

Номер	Назва етапів роботи	Термін виконання
1.	Аналіз практичної задачі інженерії ПЗ	17.02.2025
2.	Аналіз вимог до ПЗ	28.03.2025
3.	Розробка архітектури ПЗ	08.04.2025
4.	Розробка проекту ПЗ	15.04.2025
5.	Реалізація та тестування ПЗ	28.04.2025

7 Порядок контролю та приймання

На кожному етапі створення програмного забезпечення контролюють процеси аналізу та проєктування його складових з урахуванням вимог, викладених у технічному завданні. Кожна фаза розробки має бути представлена у визначені терміни та погоджена із замовником.

Приймання виконується за затвердженою методикою і програмою випробувань, а результати фіксують у протоколі проведення перевірок. Якщо під час приймання програмного продукту виявлено помилки, складають відповідний акт, у якому їх описують. Цей документ підписують представники замовника й розробника, а затверджують керівники обох сторін. Розробник зобов’язаний упродовж не більше двох тижнів виправити всі виявлені недоліки та повідомити замовника про готовність до повторної перевірки.

Додаток Б Текст програми

clientController.js

```
module.exports = function(db) {
    return {
        getAllClients: (req, res) => {
            db.query('SELECT * FROM client', (err, rows) => {
                if (err) {
                    console.error(err.message);
                    return res.status(500).send('Server error');
                }
                return res.status(200).json(rows);
            });
        },
        addClient: (req, res) => {
            const newClient = req.body;
            const { name, surname, patronymic, category,
telephones, email } = newClient;

            db.query(
                'INSERT INTO client (name, surname, patronymic,
category, telephones, email) VALUES (?, ?, ?, ?, ?, ?)',
                [name, surname, patronymic, category, telephones,
email],
                function (err, result) {
                    if (err) {
                        console.error(err.message);
                        return res.status(500).send('Server
error');
```

```
                    }

                    return res.status(201).json({ id:
result.insertId });

                }

            );

        },

        updateClient: (req, res) => {

            const { name, surname, patronymic, category,
telephones, email } = req.body;

            const clientId = parseInt(req.params.id);

            db.query('UPDATE client SET name = ?, surname = ?,
patronymic = ?, category = ?, telephones = ?, email = ? WHERE id =
?',

                [name, surname, patronymic, category, telephones,
email, clientId], (err) => {

                    if (err) {

                        console.error(err.message);

                        return res.status(500).send('Server
error');

                    }

                    return res.status(200).json({ id: clientId,
name, surname, patronymic, category, telephones, email });

                });

        },

        deleteClient: (req, res) => {

            const clientId = parseInt(req.params.id);

            db.query('DELETE FROM client WHERE id = ?', clientId,
(err) => {

                if (err) {

                    console.error(err.message);

                    return res.status(500).send('Server error');
```

```
                }
                res.status(204).send();
            });
        }
    };
};
goodsController.js
const path = require('path');
const multer = require('multer');
const fs = require('fs');

module.exports = function(db, upload) {
    return {
        getAllGoods: (req, res) => {
            db.query('SELECT *, CAST(CONCAT("images-goods/", photo)
AS CHAR) AS photo FROM goods', (err, rows) => {
                if (err) {
                    console.error(err.message);
                    return res.status(500).send('Server error');
                }
                return res.status(200).json(rows);
            });
        },
        addGoods: (req, res) => {
            upload.single('photo')(req, res, function (err) {
                if (err instanceof multer.MulterError) {
                    console.error(err.message);
                    return res.status(400).send('Error uploading
file');
```

```
                } else if (err) {
                    console.error(err.message);
                    return res.status(500).send('Server error');
                }
                if (!req.file) {
                    return res.status(400).send('No file
uploaded');
                }

                const newGoods = req.body;
                const { product_code, name, product_price,
quantity, rental_price_id} = newGoods;
                const photo = req.file.filename;

                db.query('INSERT INTO goods (product_code, name,
product_price, quantity, photo, rental_price_id) VALUES (?, ?, ?,
?, ?, ?)',
                    [product_code, name, product_price, quantity,
photo, rental_price_id], function (err) {
                        if (err) {
                            console.error(err.message);
                            return res.status(500).send('Server
error');
                        }
                        return res.status(201).json({ id:
this.lastID });
                    });
            });
        },
        updateGoods: (req, res) => {
            const goodsId = parseInt(req.params.id);
```

```
            upload.single('photo')(req, res, function (err) {
                if (err instanceof multer.MulterError) {
                    console.error(err.message);
                    return res.status(400).send('Error uploading
file');
                } else if (err) {
                    console.error(err.message);
                    return res.status(500).send('Server error');
                }

                const newGoods = req.body;
                const { product_code, name, product_price,
quantity, rental_price_id } = newGoods;

                let photo = null;

                if (req.file) {
                    photo = req.file.filename;
                } else if (!newGoods.photo) {
                    db.query('SELECT photo FROM goods WHERE id =
?', [goodsId], (err, rows) => {
                        if (err) {
                            console.error(err.message);
                            return res.status(500).send('Server
error');
                        }
                        if (rows.length > 0) {
                            photo = rows[0].photo;
                        }
```

```
                        updateGoodsData();
                    });
                    return;
                } else {
                    photo = newGoods.photo;
                }

                updateGoodsData();

                function updateGoodsData() {
                    db.query('UPDATE goods SET product_code = ?,
name = ?, product_price = ?, quantity = ?, photo = ?,
rental_price_id = ? WHERE id = ?',
                        [product_code, name, product_price,
quantity, photo, rental_price_id, goodsId], (err) => {
                            if (err) {
                                console.error(err.message);
                                return res.status(500).send('Server
error');
                            }
                            return res.status(200).json({ id:
goodsId, product_code, name, product_price, quantity, photo,
rental_price_id });
                        });
                }
            });
        },
        deleteGoods: (req, res) => {
            const goodsId = parseInt(req.params.id);
            db.query('SELECT photo FROM goods WHERE id = ?',
goodsId, (err, row) => {
```

```
                if (err) {
                    console.error(err.message);
                    return res.status(500).send('Server error');
                }
                if (row && row.photo) {
                    const photoPath = path.join(__dirname,
'../images-goods', row.photo);
                    fs.unlink(photoPath, (err) => {
                        if (err) {
                            console.error(err.message);
                            return res.status(500).send('Server
error');
                        }
                        console.log('Photo deleted successfully');
                        db.query('DELETE FROM goods WHERE id = ?',
goodsId, (err) => {
                            if (err) {
                                console.error(err.message);
                                return res.status(500).send('Server
error');
                            }
                            res.status(204).send();
                        });
                    });
                } else {
                    db.query('DELETE FROM goods WHERE id = ?',
goodsId, (err) => {
                        if (err) {
                            console.error(err.message);
```

```
                            return res.status(500).send('Server
error');
                        }
                        res.status(204).send();
                    });
                }
            });
        }
    };
};
registrationAuthController.js
const path = require('path');
const jwt = require('jsonwebtoken');
const secret = process.env.JWT_SECRET;
const bcrypt = require('bcryptjs');

module.exports = function(db, upload) {
    return {
        register: (req, res) => {
            const { username, password, role = 'user' } = req.body;

            bcrypt.hash(password, 10, (err, hash) => {
                if (err) throw err;
                const sql = 'INSERT INTO users (username, password,
role) VALUES (?, ?, ?)';
                db.query(sql, [username, hash, role], (err, result)
=> {
                    if (err) throw err;
                    res.json({ status: 'User registered' });
```

```
            });
        });
    },
    login: (req, res) => {
        const { username, password } = req.body;
        const sql = 'SELECT * FROM users WHERE username = ?';
        db.query(sql, [username], (err, results) => {
          if (err) throw err;
          if (results.length > 0) {
            const user = results[0];
            bcrypt.compare(password, user.password, (err,
isMatch) => {
              if (err) throw err;
              if (isMatch) {
                const token = jwt.sign({ id: user.id, role:
user.role }, secret, { expiresIn: '1h' });
                res.json({ token, id: user.id, role: user.role
});
              } else {
                res.status(401).json({ message: 'Invalid
credentials' });
              }
            });
          } else {
            res.status(401).json({ message: 'Invalid
credentials' });
          }
        });
      },
    getAllUsers: (req, res) => {
```

```
            const sql = 'SELECT id, username, role FROM users';
            db.query(sql, (err, results) => {
              if (err) {
                return res.status(500).json({ error: 'Database
query error' });
              }
              res.json(results);
            });
          },
          deleteUser: (req, res) => {
            const { id } = req.params;
            const sql = 'DELETE FROM users WHERE id = ?';
            db.query(sql, [id], (err, result) => {
              if (err) {
                return res.status(500).json({ error: 'Failed to
delete user' });
              }
              if (result.affectedRows === 0) {
                return res.status(404).json({ message: 'User not
found' });
              }
              res.json({ status: 'User deleted' });
            });
          }
    };
};
```

Додаток В Опис програми

1. Загальні відомості

Розроблений вебзастосунок для автоматизації обліку прокату туристичного спорядження магазину «Хан-Тенгрі» представляє собою програму для адміністратора та менеджера магазину туристичного спорядження, які займаються створенням прокатів, роботою з клієнтами, товарами, цінами на прокат.

Вебзатосунок призначений для автоматизації обліку прокату туристичного спорядження, що дозволить підвищити зручність і ефективність роботи менеджерів, відповідальних за його організацію та управління, а також зменшити кількість помилок у процесі обліку та видачі спорядження.

Програмне забезпечення, необхідне для функціонування вебзастосунку:

- Серверне середовище виконання: Node.js (v16+)
- СУБД: MySQL (8.0+)
- Інструменти розгортання та адміністрування: FastPanel
- Клієнтська сторона: Сучасний веббраузер (Google Chrome, Mozilla Firefox, Opera)
- Додаткові утиліти:
- Git (контроль версій)
- Postman (тестування API)
- VS Code (редактор коду)

Мови програмування, на яких написана програма

- JavaScript (ES6+)
- Серверна логіка (Node.js + Express.js)
- Клієнтський інтерфейс (HTML/CSS у поєднанні з JS для динаміки)

2. Функціональне призначення

Функціональне призначення полягає в можливості користувачів виконувати такі функції:

– реєстрація користувачів;

– авторизація користувачів;

– перегляд та обробка інформації про товари на складі (додавання, видалення, редагування);

– перегляд та обробка інформації про клієнтів (додавання, видалення, редагування);

– перегляд та обробка інформації про ціни прокату (додавання, видалення, редагування);

– створення замовлення на прокат (додавання усієї інформації щодо прокату, пошук товару та клієнта, внесення нового клієнта у разі його відсутності);

– видалення та редагування замовлення на прокат;

– перегляд часової стрічки (відображення проміжку від початкової до кінцевої дати прокату кожного товару що був взятий в прокат з урахуванням кількості цього товару);

– оновлення статусу замовлення на прокат на «повернено» (зникнення маркування з часової стрічки, збереження запису в базі);

– перегляд списку усіх товарів, записаних у конкретний прокат;

– перегляд інформації про ціну прокату, пов’язаної з кожним товаром;

– перегляд та видалення зареєстрованих користувачів (менеджерів);

вихід з облікового запису.

3. Опис логічної структури

Розроблений вебзастосунок для автоматизації обліку прокату туристичного спорядження магазину «Хан-Тенгрі» базується на React з використанням функціональних компонентів, що дозволяє чітко розділити логіку окремих функцій між модулями, такими як робота з клієнтами, управління товарами та встановлення цін.

Для зберігання та обробки даних застосунок використовує базу даних MySQL із нормалізованою структурою, що гарантує цілісність інформації і швидкий доступ до неї під час операцій обліку та видачі спорядження. Такий підхід дозволяє оптимізувати робочі процеси, підвищити зручність управління і забезпечити надійність роботи системи в цілому.

4. Технічні засоби

Для роботи з вебзастосунком необхідна система наступної мінімальної конфігурації:

- процесор: двоядерний із тактовою частотою не менше 2,0 ГГц;
- оперативна пам'ять – 4GB;
- системний диск: 100 ГБ вільного дискового простору;
- необхідно мережеве з'єднання з виходом в інтернет.

5. Виклик та завантаження

URL вебзастосунку прив'язаний до іконки, при натисканні на неї миттєво відкривається браузер і завантажує систему. Після цього користувачу відкривається екран авторизації, де він може ввести свої облікові дані для входу до застосунку, або, якщо сесія вже активна, одразу потрапити на головний інтерфейс. Такий підхід спрощує доступ до функціоналу системи, забезпечуючи швидкий перехід до роботи з обліком прокату туристичного спорядження.

Автоматизація виклику і завантаження застосунку сприяє підвищенню зручності користування та оптимізації робочих процесів.

6. Вхідні та вихідні дані

Дані зберігаються у базі даних MySQL.

Вхідні дані: введення даних здійснюється за допомогою пристроїв введення (клавіатура та миша). Також дані можуть вводитися вручну або обиратися із наявних випадаючих списків.

Вихідні дані: виведення даних відбувається на моніторі в режимі реального часу.

Додаток Г Інструкція користувача

1. Призначення програми

Програма призначена для централізованого управління всіма етапами обліку та обробки прокатних операцій у магазині «Хан-Тенгрі». Після швидкого запуску через іконку на робочому столі користувач потрапляє на захищений екран авторизації, де може відновити активну сесію або ввести власні облікові дані. Інтерфейс забезпечує інтуїтивну навігацію між розділами товарів, клієнтів, тарифів і прокатів, дозволяючи в один клік створювати, редагувати або видаляти записи та миттєво відстежувати історію змін. Для менеджерів доступні всі необхідні інструменти – від налаштування цін до формування нових замовлень із відображенням часової стрічки зайнятості обладнання – а адміністратори додатково можуть керувати обліковими записами співробітників. Реалізована валідація даних та повідомлення про помилки гарантують коректність операцій, а збереження інформації в єдиній базі даних забезпечує їхню цілісність і доступність у режимі реального часу.

2. Умови використання програми

2.1. Апаратні засоби

- Персональний комп’ютер або ноутбук із ОС Windows 10 (або вище)
- Процесор: не менше двох ядер із тактовою частотою не менше 2,0 ГГц
- ОЗП: не менше 4 ГБ
- Вільне місце на диску: не менше 100 ГБ

2.2. Програмні засоби

- Node.js (версія 16 або вище)
- Менеджер пакетів npm (або yarn)

- СУБД MySQL (версія 8.0 або вище)
- Веб-сервер / середовище розгортання (FastPanel, Docker або аналог)
- Сучасний веб-браузер: Google Chrome, Mozilla Firefox, Opera (актуальні версії)

2.3. Додаткові умови

- Постійний доступ до Інтернету (для завантаження залежностей і оновлень)
- Наявність резервного живлення (акумулятор ноутбука або UPS)
- Налаштовані права доступу користувача до файлової системи (для збереження фотографій товарів)

Ці умови гарантують коректне завантаження, роботу інтерфейсу та збереження даних без втрат і збоїв.

3. Виконання програми

URL вебзастосунку для автоматизації обліку прокату туристичного спорядження магазину «Хан-Тенгрі» прив'язаний до іконки, при натисканні на яку, миттєво відкривається браузер і завантажує систему. Після цього користувачу відкривається екран авторизації, де він може ввести свої облікові дані для входу до застосунку, або, якщо сесія вже активна, одразу потрапити на головний інтерфейс. Головний екран авторизації показаний на рисунку Г.1

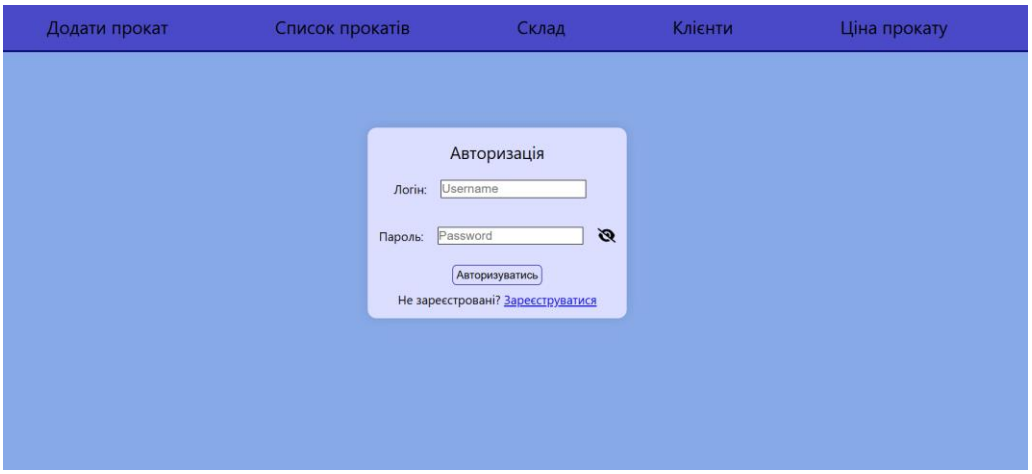


Рисунок Г.1 – Головний екран авторизації вебзастосунку для автоматизації обліку прокату туристичного спорядження магазину «Хан-Тенгрі»

У разі введення некоректних даних користувача відображається вікно з повідомленням про помилку та пропозицією повторити введення. Якщо дані вказано правильно, запускається головна сторінка програми показана на рисунку Г.2

Рисунок Г.2. – Головна сторінка вебзастосунку для автоматизації обліку прокату туристичного спорядження магазину «Хан-Тенгрі»

Для роботи з різними списками треба обрати потрібну вам вкладку у верхньому меню. Після цього у користувача з’явиться список із інформацією яка йому потрібна. На рисунку Г.3 зображений список з клієнтами.

Рисунок Г.3. – Сторінка з інформацією про клієнтів вебзастосунку для автоматизації обліку прокату туристичного спорядження магазину «Хан-Тенгрі»

На кожній сторінці з інформацією (окрім сторінки «додати прокат») користувач може побачити кнопку «Додати товар/клієнта/ціну товару» (в залежності на якій сторінці знаходиться користувач. При натисканні на неї з’являється форма з полями для запису нового товару/клієнта/ціни прокату. Форма для додавання клієнта зображена на рисунку Г.4.

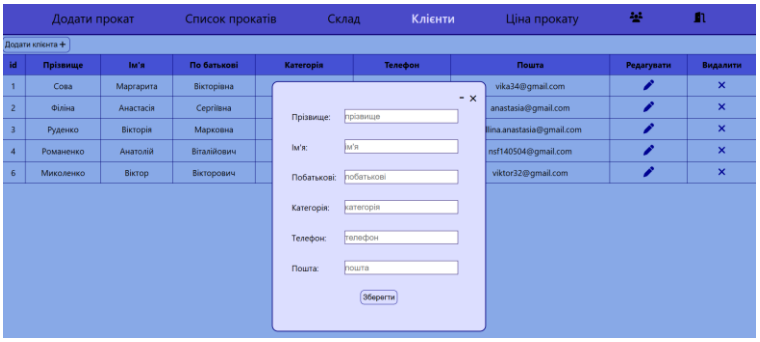


Рисунок Г.4. – Форма для додавання клієнта вебзастосунку для автоматизації обліку прокату туристичного спорядження магазину «Хан-Тенгрі»

При вводі некоректних даних користувачу будуть показані попереджальні повідомлення. При вводі коректних даних введена інформація буде збережена та показана у списку разом з іншими. Ця процедура працює однаково на кожній сторінці крім сторінки «Додати прокат».

Для редагування інформації в конкретному клієнті/товарі/ціні товару користувачу треба натиснути кнопку з іконкою олівця біля обраного клієнта/товару/ціні товару. Після цього відкриється форма для редагування інформації. Далі процедура виконання така сама як і при додаванні. Форма для редагування зображена на рисунку Г.5.

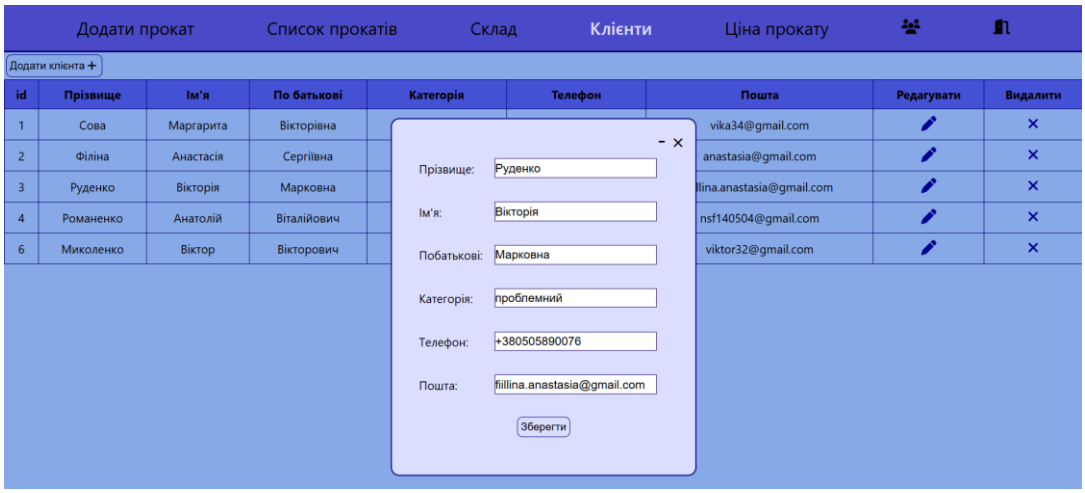


Рисунок Г.5. – Форма для редагування клієнта вебзастосунку для автоматизації обліку прокату туристичного спорядження магазину «Хан-Тенгрі»

Для видалення інформації конкретного клієнта/товару/ціни товару користувачу треба натиснути кнопку з іконкою хрестика біля обраного клієнта/товару/ціни товару. Після цього з’явиться попередження про видалення. Для видалення користувачу потрібно натиснути кнопку «видалити» або «закрити» для відміни операції. Попередження про видалення показане на рисунку Г.6.

Рисунок Г.6. – Попередження про видалення клієнта вебзастосунку для автоматизації обліку прокату туристичного спорядження магазину «Хан-Тенгрі»

На сторінці з товарами користувач може переглянути ціну прокату кожного товару до якого вона прив’язана. Для цього треба натиснути на кнопку «ціна прокату» біля товару інформацію про якого користувач хоче дізнатися. Після цього обраний товар виділиться кольором та з’явиться табличка з даними про ціну прокату цього товару. Результат такої дії зображений на рисунку Г.7

Рисунок Г.7 – Перегляд ціни прокату обраного товару вебзастосунку для автоматизації обліку прокату туристичного спорядження магазину «Хан-Тенгрі»

На сторінці з прокатами для перегляду інформації про товари, які були записані в обраний прокат користувачу потрібно натиснути кнопку «Товари» біля обраного прокату. Після цього користувачу буде показана таблиця з інформацією про товари які були записані в конкретний прокат. Результат цієї дії зображений на рисунку Г. 8.

Рисунок Г.8 – Перегляд записаних товарів обраного прокату вебзастосунку для автоматизації обліку прокату туристичного спорядження магазину «Хан-Тенгрі»

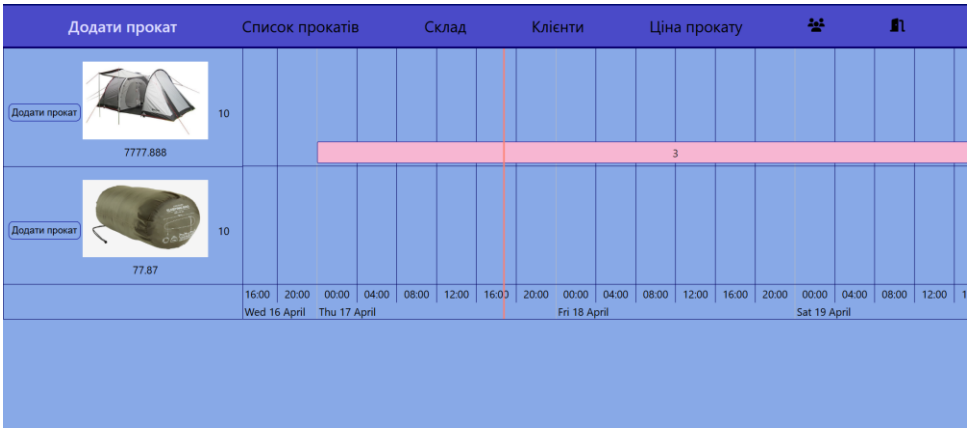
На сторінці «Додати прокат» користувачу показані наявні товари з часовою стрічкою на них. Для запису нового прокату користувачу треба натиснути на кнопку «Додати прокат» біля того товару, який він хоче записати в прокат. Після цього користувачу буде показана форма з полями для заповнення. Форма для додавання прокату показана на рисунку Г.9.

Рисунок Г. 9 – Форма для додавання прокату вебзастосунку для автоматизації обліку прокату туристичного спорядження магазину «Хан-Тенгрі»

Для додавання додаткового товару у прокат користувач повинен ввести код товару та його кількість та натиснути кнопку «Додати товар в прокат». При успішному додаванні користувачу буде показано повідомлення про успішний запис товару. При помилці користувачу буде показано повідомлення що такий товар не знайдено.

Для запису клієнта у прокат треба ввести його номер телефону в відповідне поле та натиснути кнопку «Записати клієнта». При успішному записі клієнта в прокат користувачу буде показане повідомлення про успішний запис клієнта. При помилці користувачу буде показане повідомлення що такого клієнта не знайдено та буде запропоноване створити нового клієнта. Для цього користувачу треба натиснути на кнопку «створити клієнта». Далі буде відбуватися така сама процедура, яка описувалася вище при створенні нового клієнта на сторінці клієнта. Після натискання кнопки «зберегти» клієнта буде додано в загальну базу та записано в конкретний прокат. Після цього при натисканні головної кнопки «зберегти прокат» буде створено та показано в загальному списку прокатів на сторінці з прокатами та біля товару який був доданий в прокат з’явиться

маркування на часовій стрічці з кількістю цього товару взятого в цей прокат. Результат створення прокату показаний на рисунку Г.10.



Рисунку Г.10 – Результат створення прокату вебзастосунку для автоматизації обліку прокату туристичного спорядження магазину «Хан-Тенгрі»

Для адміністратора доступна функція переглянути всіх користувачів. Для цього треба натиснути на іконку з користувачам в правому верхньому куті. Тоді адміністратору буде показана таблиця зі всіма користувачами зареєстрованими в системі. При потребі адміністратор може видалити користувача із системи натиснувши напроти нього на іконку з хрестиком. Процедура видалення така сама як і для інших списків. Таблиця з користувачами представлена на рисунку Г.11.

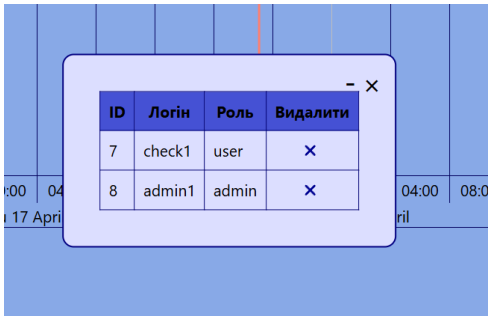


Рисунок Г. 11 – Таблиця з користувачами вебзастосунку для автоматизації обліку прокату туристичного спорядження магазину «Хан-Тенгрі»

Для виходу з облікового запису користувач повинен натиснути на іконку з дверима в правому верхньому куті. Тоді користувача буде повернено на сторінку з авторизацією.

Додаток Д Програма та методика випробування вебзастосунку для автоматизації обліку прокату туристичного спорядження у магазині «Хан-Тенгрі»

1 Об’єкт випробувань

1.1 Назва об’єкту

Повна назва об’єкта випробувань: вебзастосунок для автоматизації обліку прокату туристичного спорядження магазину «Хан-Тенгрі».

Коротка назва об’єкта випробувань: вебзастосунок

1.2 Область застосування:

Вебзастосунок призначений для автоматизації та оптимізації процесів обліку товарів, оформлення прокату, фінансових розрахунків та взаємодії з клієнтами у магазині туристичного спорядження «Хан-Тенгрі». Це забезпечує ефективне управління ресурсами прокатного пункту, знижує можливість помилок, підвищує швидкість обслуговування та полегшує роботу співробітників, роблячи процес прокату зручнішим для всіх учасників.

Функціональне призначення полягає в можливості користувачів виконувати такі функції:

– реєстрація користувачів;

– авторизація користувачів;

– перегляд та обробка інформації про товари на складі (додавання, видалення, редагування);

– перегляд та обробка інформації про клієнтів (додавання, видалення, редагування);

– перегляд та обробка інформації про ціни прокату (додавання, видалення, редагування);

– створення замовлення на прокат (додавання усієї інформації щодо прокату, пошук товару та клієнта, внесення нового клієнта у разі його відсутності);

– видалення та редагування замовлення на прокат;

– перегляд часової стрічки (відображення проміжку від початкової до кінцевої дати прокату кожного товару що був взятий в прокат з урахуванням кількості цього товару);

– оновлення статусу замовлення на прокат на «повернено» (зникнення маркування з часової стрічки, збереження запису в базі);

– перегляд списку усіх товарів, записаних у конкретний прокат;

– перегляд інформації про ціну прокату, пов’язаної з кожним товаром;

– перегляд та видалення зареєстрованих користувачів (менеджерів);

– вихід з облікового запису.

2 Мета випробувань

Метою проведення випробувань є необхідність перевірки відповіднсоті вебзастосунку вимогам, які викладені у технічному завданні, що наводиться у додатку А.

3 Вимоги до програми

Вебзастосунок повинен реалізовувати функції, описані в технічному завданні, наведеному в додатку А.

4 Вимоги до програмної документації

Для проведення випробування повинна бути надана наступна документація:

– технічне завдання на розробку програмного забезпечення;

– текст програми;

– опис програми;

– інструкція користувача;

– програма і методика випробувань програмного забезпечення.

5 Засоби і порядок випробувань

Технічні засоби, що використовуються під час випробувань

– Розробницький комп’ютер: процесор Intel Core i5, 8 ГБ ОЗП, SSD-накопичувач

– Стабільне підключення до Інтернету

Програмні засоби, що використовуються під час випробувань

– Середовище виконання: Node.js (v16+) + npm

– ОС Windows 10

– СУБД: MySQL (Workbench або CLI)

– Редактор коду: Visual Studio Code

– Web-сервер та middleware: Express.js, Multer

– Інструменти тестування API та БД:

Postman (ручні запити)

MySQL Workbench (імпорт/експорт дампів)

– Контроль версій: Git

Порядок проведення випробувань:

– Підготовка тестового середовища (інсталяція залежностей через npm install).

– Розробка тестових випадків на основі функціональних вимог.

– Написання тестів за допомогою Jest і запуск командою npm test.

– Аналіз результатів, виправлення дефектів і документування звіту за кожним тестом.

6 Методи випробувань

Розроблені тестові приклади для тестування методом «Чорної скриньки» за варіантами використання та функціональними вимогами. Показані приклади для основник варіантів використання:

Тестовий випадок 1

Ідентифікатор: TC-REG-01

Предмети тестування (Test items): Функція реєстрації нового користувача в системі.

Специфікація вхідних даних (Input specifications):

- Валідний логін (admin1)
- Валідний пароль (testAdmin12)
- Валідна роль (обирається з випадаючого списку) (admin)

Специфікація результатів (Output specifications):

- Створюється новий обліковий запис у системі.
- Система підтверджує успішну реєстрацію (через повідомлення).

Тестове оточення (Environmental needs):

- Вебдодаток або інтерфейс системи, що дозволяє реєструвати користувача
- Підключення до бази даних, у якій зберігаються користувачі
- Середовище виконання (браузер)

Спеціальні процедурні вимоги (Special procedural requirements):

- Необхідно мати доступ до бази даних для перевірки факту створення нового запису

Залежність з іншими тестами (Intercase dependencies):

- Тест виконується першим, щоб перевірити базову функціональність реєстрації
- Результат цього тесту може впливати на тести аутентифікації, оскільки створений обліковий запис може бути використано в подальшому

Кроки виконання (Typical test steps):

1. Відкрити форму реєстрації.

2. Ввести унікальний логін (admin1), валідний пароль (testAdmin12) та коректну роль (admin).

3. Натиснути кнопку «Зареєструватися».

Очікуваний результат (Expected result):

- Система підтверджує реєстрацію (з'являється повідомлення про успіх).
- У базі даних з'являється новий обліковий запис із введеними даними.

Тестовий випадок 2

Ідентифікатор: TC-REG-02

Предмети тестування (Test items): Функція реєстрації нового користувача за умови, що логін уже існує в базі.

Специфікація вхідних даних (Input specifications):

- Логін, який уже існує в базі (admin1)
- Валідний пароль (testAdmin14)
- Валідна роль (admin)

Специфікація результатів (Output specifications):

- Відмова в реєстрації
- Система має відобразити повідомлення, що логін уже зареєстровано

Тестове оточення (Environmental needs):

- Вебдодаток або інтерфейс системи для реєстрації
- База даних із попередньо створеним записом, що має логін admin1

Спеціальні процедурні вимоги (Special procedural requirements):

- Доступ до бази даних для попереднього створення запису з логіном existinguser
- Стан бази даних має бути таким, щоб existinguser точно існував перед виконанням тесту

Залежність з іншими тестами (Intercase dependencies):

– Виконується незалежно від попереднього тесту (TC-REG-01), оскільки треба заздалегідь підготувати базу даних

– Результат може впливати на подальші тести, які перевіряють обробку помилок реєстрації

Кроки виконання (Typical test steps):

1. Переконатися, що у базі існує користувач з логіном admin1.
2. Відкрити форму реєстрації.
3. Ввести логін admin1, testAdmin14 та роль (admin).
4. Натиснути «Зареєструватися».

Очікуваний результат (Expected result):

– Система відмовляє в реєстрації та показує повідомлення «Логін уже існує».

– Новий обліковий запис не створюється у базі даних.

Тестовий випадок 3

Ідентифікатор: TC-REG-03

Предмети тестування (Test items): Функція реєстрації з неповними даними (відсутність одного чи декількох обов’язкових полів).

Специфікація вхідних даних (Input specifications):

Логін, пароль, роль у різних комбінаціях, коли щонайменше одне поле пропущено:

– Порожній логін, але задані (testAdmin11) і роль (admin)

– Порожній пароль, але задані логін (admin2) і роль (admin)

– Порожня роль, але задані логін (admin2) і пароль (testAdmin11)

Специфікація результатів (Output specifications):

– Реєстрація має не відбутися

– Система має показати повідомлення про помилку(и), в яких вказано відсутнє(і) поле(я)

Тестове оточення (Environmental needs):

- Форма реєстрації у вебдодатку
- Підключення до бази даних (не очікується зміна стану БД, оскільки облікові записи не створюватимуться)

Спеціальні процедурні вимоги (Special procedural requirements):

- Необхідно переконатися, що всі обов’язкові поля дійсно є обов’язковими за логікою системи

Залежність з іншими тестами (Intercase dependencies):

- Немає прямої залежності від інших тестів, оскільки тут перевіряється поведінка з відсутніми даними
- Проте бажано виконувати після того, як переконалися, що базова реєстрація (TC-REG-01) працює

Кроки виконання (Typical test steps):

Для кожної комбінації відсутніх полів:

1. Відкрити форму реєстрації.
2. Залишити порожніми:

2.1 Порожній логін, але задані (testAdmin11) і роль (admin)

2.2 Порожній пароль, але задані логін (admin2) і роль (admin)

2.3 Порожня роль, але задані логін (admin2) і пароль (testAdmin11)

3 Натиснути «Зареєструватися».

Очікуваний результат (Expected result):

- Система показує повідомлення(я) про помилку(и), де вказано, що поле(-я) є обов’язковим(-ими).
- Новий обліковий запис не створюється у базі даних.

1. Авторизація

Специфікація:

Вхідні дані: логін (admin1), пароль (testAdmin12).

Результат: Вхід в систему з правами ролі.

Тестові вимоги:

– Перевірити, що користувач успішно входить до системи при введенні коректних даних.

– Перевірити, що при введенні некоректного логіну або паролю вхід до системи не здійснюється і видається повідомлення про помилку.

– Перевірити обробку порожніх даних.

Тестові випадки:

Тестовий випадок 1

Ідентифікатор: TC-AUTH-01

Предмети тестування (Test items): Функція авторизації (вхід до системи) з коректними даними.

Специфікація вхідних даних (Input specifications):

– Існуючий у системі логін (admin1)

– Коректний пароль (testAdmin12)

Специфікація результатів (Output specifications):

– Система надає доступ користувачу з відповідними правами (відповідно до ролі)

– Перенаправлення на головну сторінку користувача

Тестове оточення (Environmental needs):

– Вебінтерфейс, що дозволяє вводити логін та пароль

– Підготовлена база даних, у якій створений обліковий запис testuser з певним паролем та роллю

Спеціальні процедурні вимоги (Special procedural requirements):

– Потрібно забезпечити валідність облікового запису (логін/пароль) перед тестуванням

Залежність з іншими тестами (Intercase dependencies):

– Цей тест логічно може слідувати після тестів реєстрації (щоб мати дійсні облікові дані), але може бути виконаний і незалежно, якщо дані в базі є заздалегідь

Кроки виконання (Typical test steps):

1. Відкрити форму авторизації.
2. Ввести існуючий логін (admin1) та коректний пароль (testAdmin12).
3. Натиснути кнопку «Авторизуватися» (Login).

Очікуваний результат (Expected result):

- Система авторизує користувача та надає доступ до особистого кабінету або іншої сторінки, призначеної для авторизованих користувачів.
- Роль користувача відображається коректно, і система надає відповідні права доступу.

Тестовий випадок 2

Ідентифікатор: TC-AUTH-02

Предмети тестування (Test items): Функція авторизації з некоректними даними (логін або пароль).

Специфікація вхідних даних (Input specifications):

- Логін, який не існує в базі даних (admin88)

АБО

- Існуючий логін (admin1), але некоректний пароль (testadmin123)

Специфікація результатів (Output specifications):

- Система відмовляє у вході до системи
- Відображається повідомлення про помилку («Невірний логін або пароль»)

Тестове оточення (Environmental needs):

- Веб-інтерфейс для введення даних авторизації
- База даних із зареєстрованим обліковим записом для другої частини тесту (щоб перевірити негативний сценарій некоректного пароля)

Спеціальні процедурні вимоги (Special procedural requirements):

- Необхідно перевірити, що в повідомленні про помилку не розкриваються зайві деталі (наприклад, уточнена причина: «логін не існує»), якщо це суперечить вимогам безпеки.

Залежність з іншими тестами (Intercase dependencies):

- Виконується незалежно від позитивного тесту (TC-AUTH-01), але часто його запускають слідом, щоб перевірити негативний сценарій

Кроки виконання (Typical test steps):

1. Відкрити форму авторизації.
2. Ввести некоректні дані:

– Зовсім неіснуючий логін (admin88), або

– Існуючий логін (admin1), але неправильний пароль. (testadmin123)

3. Натиснути кнопку «Увійти».

Очікуваний результат (Expected result):

- Система не авторизує користувача.
- Відображається повідомлення про невірні облікові дані («Невірний логін або пароль»).

Тестовий випадок 3

Ідентифікатор: TC-AUTH-03

Предмети тестування (Test items): Функція авторизації з порожніми даними.

Специфікація вхідних даних (Input specifications):

- Порожній логін та пароль

Специфікація результатів (Output specifications):

- Система має відмовити у вході та вивести повідомлення про помилку
- Ніякий доступ до системи не надається

Тестове оточення (Environmental needs):

- Форма авторизації

- База даних (теоретично не змінюється, бо користувач не входить до системи)

Спеціальні процедурні вимоги (Special procedural requirements):

- Переконатися, що система однаково обробляє порожні дані незалежно від локалізації та налаштувань

Залежність з іншими тестами (Intercase dependencies):

- Може виконуватися після TC-AUTH-01 і TC-AUTH-02, оскільки перевіряє окремий набір негативних сценаріїв
- Результат не впливає на інші тести.

Кроки виконання (Typical test steps):

1. Відкрити форму авторизації.
2. Спробувати виконати вхід, залишивши обидва поля порожніми
3. Натиснути «Атворизуватися».

Очікуваний результат (Expected result):

- Система не здійснює авторизацію.
- З’являється повідомлення про те, що логін та пароль є обов’язковими полями.

2. Створення прокату (додавання всієї інформації щодо прокату, пошук товару за номером та запис його в прокат (кількість цього товару яку бажає взяти в прокат клієнт), пошук клієнта за номером телефона та запис клієнта в цей прокат. При відсутності даного клієнта в базі даних, можливість додавати клієнта під час створення прокату та робити запис створеного клієнта в цей прокат);

Специфікація:

Вхідні дані:

Номер замовлення: 012345

Товар (пошук за номером): 123.345

Прізвище клієнта (пошук за телефоном): Іваненко Петро (тел. +380671234567)

Дата замовлення: 10.02.2024

Статус прокату: Активний

Прізвище співробітника, що оформлював прокат: Ковальчук Марина

Дата початку прокату: 12.02.2024

Дата кінця прокату: 20.02.2024

Ціна прокату: 1500 грн

Оплачена сума: 1000 грн

Борг: 500 грн

Форма залогу: гроші

Сума залогу: 2000 грн

Кількість товару в прокат: 2

Тестові вимоги:

– Перевірити, що при введенні валідних даних створюється новий запис прокату.

– Перевірити, що пошук товару за номером відбувається коректно і обраний товар правильно записується у прокат.

– Перевірити, що пошук клієнта за телефоном здійснюється коректно, і при відсутності клієнта система пропонує можливість його додавання.

– Перевірити, що при неповних або некоректних даних система видає повідомлення про помилку.

Тестові випадки:

Тестовий випадок 1

Ідентифікатор: TC-RENT-ORDER-01

Предмети тестування (Test items): Функція створення нового запису прокату з повним набором коректних даних.

Специфікація вхідних даних (Input specifications):

Номер замовлення: 012345

Товар (пошук за номером): 123.345

Прізвище клієнта (пошук за телефоном): Іваненко Петро (тел. +380671234567)

Дата замовлення: 10.02.2024

Статус прокату: Активний

Прізвище співробітника, що оформлював прокат: Ковальчук Марина

Дата початку прокату: 12.02.2024

Дата кінця прокату: 20.02.2024

Ціна прокату: 1500 грн

Оплачена сума: 1000 грн

Борг: 500 грн

Форма залогу: гроші

Сума залогу: 2000 грн

Кількість товару в прокат: 2

Специфікація результатів (Output specifications):

- У системі створюється новий запис прокату із зазначеними даними.
- Система повідомляє про успішне створення прокату.

Тестове оточення (Environmental needs):

- Інтерфейс керування прокатом (вебдодаток)
- Підключення до бази даних, де зберігаються товари та клієнти.

Спеціальні процедурні вимоги (Special procedural requirements):

- Перевірити, що всі дати та числові поля введені у правильному форматі.

Залежність з іншими тестами (Intercase dependencies): -

Кроки виконання (Typical test steps):

1. Увійти в систему під користувачем із правами створення прокату.
2. Перейти до розділу «Створити прокат» (або аналогічного).
3. Заповнити форму нової операції прокату всіма переліченими валідними даними.

Номер замовлення: 012345

Товар (пошук за номером): 123.345

Прізвище клієнта (пошук за телефоном): Іваненко Петро (тел. +380671234567)

Дата замовлення: 10.02.2024

Статус прокату: Активний

Прізвище співробітника, що оформлював прокат: Ковальчук Марина

Дата початку прокату: 12.02.2024

Дата кінця прокату: 20.02.2024

Ціна прокату: 1500 грн

Оплачена сума: 1000 грн

Борг: 500 грн

Форма залогу: гроші

Сума залогу: 2000 грн

Кількість товару в прокат: 2

4. Натиснути «Зберегти» або «Створити прокат».

Очікуваний результат (Expected result):

- Система успішно створює нове замовлення

Тестовий випадок 2

Ідентифікатор: TC-RENT-ORDER-02

Предмети тестування (Test items): Правильність пошуку товару за унікальним номером та коректне додавання вибраного товару до замовлення.

Специфікація вхідних даних (Input specifications):

- Номер товару (123.345), що вже є в базі даних.
- Кількість товару (2)

Специфікація результатів (Output specifications):

- Товар з номером (123.345) успішно знаходиться.
- Додавання товару (2) до прокату відбувається без помилок.

- Система підтверджує, що товар додано до замовлення.

Тестове оточення (Environmental needs):

- Система з наявним довідником товарів (товар 123.345 повинен існувати).

Спеціальні процедурні вимоги (Special procedural requirements):

- У разі відсутності товару в базі система має видати помилку

Залежність з іншими тестами (Intercase dependencies):

- Може бути частиною загального процесу створення прокату (TC-RENT-ORDER-01).
- Результат важливий для подальшого завершення оформлення прокату.

Кроки виконання (Typical test steps):

1. Увійти до інтерфейсу створення/редагування прокату.
2. У полі пошуку товару ввести 123.345
3. Переконатися, що система знайшла потрібний товар
4. Вказати кількість «2» та додати товар до списку прокату.

Очікуваний результат (Expected result):

- Після натискання «Додати» товар з’являється у запису на прокат
- Система не відображає жодних помилок.

Тестовий випадок 3

Ідентифікатор: TC-RENT-ORDER-03

Предмети тестування (Test items): Функція пошуку клієнта за номером телефону під час створення прокату і можливість додати нового клієнта, якщо пошук не дав результату.

Специфікація вхідних даних (Input specifications):

- Номер телефону клієнта, який існує в базі (+380501234567)
- Номер телефону, якого немає в базі (+380501234999)

Специфікація результатів (Output specifications):

- Якщо номер телефону є в базі — система успішно додає клієнта в прокат (+380501234567)
- Якщо номер телефону не знайдено — система пропонує створити нового клієнта, з відповідними полями (прізвище, ім’я, по батькові, інша контактна інформація).

Прізвище (Іваненко)

Ім'я (Петро)

По батькові (Петрович)

Категорія (постійний)

Телефон (+380501234999)

Після створення нового клієнта його автоматично додає до прокату.

Тестове оточення (Environmental needs):

- База даних з принаймні одним клієнтом, зареєстрованим на «+380501234567».
- Відсутній запис про «+380501234999»

Спеціальні процедурні вимоги (Special procedural requirements):

- Перевірити форму додавання клієнта (чи вимагає система всіх обов’язкових полів).
- У разі успішного додавання нового клієнта переконатися, що він з’являється у базі й прив’язується до поточного замовлення.

Залежність з іншими тестами (Intercase dependencies):

- Виконується після авторизації
- Дані створеного клієнта можуть бути використані в інших тестах, наприклад, редагування чи подальший пошук.

Кроки виконання (Typical test steps):

1. У формі створення прокату в полі «Пошук клієнта за телефоном» ввести «+380501234567».

– Переконатися, що система знаходить існуючого клієнта

2. Для перевірки негативного сценарію ввести «+380501234999» (номер, якого немає в базі).

– Система має повідомити, що клієнта не знайдено, й запропонувати додати нового.

3. Ввести дані для нового клієнта (ПІБ, категорія тощо) і зберегти.

Прізвище (Іваненко)

Ім'я (Петро)

По батькові (Петрович)

Категорія (постійний)

Телефон (+380501234999)

Очікуваний результат (Expected result):

– При введенні відомого номера телефону клієнт знаходиться, і його дані додаються до замовлення.

– При введенні невідомого номера телефону система дозволяє створити нового клієнта; після збереження клієнт з’являється в базі й автоматично додається до замовлення.

Тестовий випадок 4

Ідентифікатор: TC-RENT-ORDER-04

Предмети тестування (Test items): Валідація обов’язкових полів і коректність формату введених даних під час оформлення прокату.

Специфікація вхідних даних (Input specifications):

– Відсутність обов’язкових полів (прізвище співробітника пусте поле).

Специфікація результатів (Output specifications):

– Система не дає зберегти прокат із помилковими даними.

– Користувач бачить повідомлення(я) (прізвище співробітника є обов’язковим полем)

Тестове оточення (Environmental needs):

– Форма/інтерфейс створення прокату.

– Налаштовані механізми валідації (бізнес-логіка) для усіх обов’язкових полів.

Спеціальні процедурні вимоги (Special procedural requirements):

– Перевірити, чи система вказує конкретно, які поля є помилковими.

– У разі кількох помилок — користувач бачить повний перелік помилок.

Залежність з іншими тестами (Intercase dependencies):

– Може виконуватися до або після позитивних сценаріїв, щоб перевірити негативну логіку.

– Не впливає напряму на інші тести, бо запис прокату не буде збережений у разі помилок.

Кроки виконання (Typical test steps):

1. Перейти у розділ «Створити прокат» (або аналогічний).
2. Залишити поле прізвище співробітника порожнім
3. Натиснути «Зберегти».

Очікуваний результат (Expected result):

– Система відмовляється зберігати прокат з некоректними/неповними даними.

– Відображається повідомлення(я) про помилку (поле прізвище співробітника є обов’язковим)

3. Перегляд часової стрічки в якій напроти кожного товару знаходиться

Додаток Е Акт впровадження

А К Т

впровадження результатів кваліфікаційної роботи

на здобуття ступеня вищої освіти «бакалавр»

на тему: Розробка вебзастосунку для автоматизації обліку прокату туристичного спорядження у магазині «Хан-Тенгрі»

Філіної Анастасії Сергіївни

Я, що підписався нижче, керівник магазину «Хан-Тенгрі» Пішеходько Наталя Олександрівна, склав цей акт про те, що результати бакалаврської роботи Філіної Анастасіяї Сергіївни, представленої на здобуття освітнього ступеня бакалавра зі спеціальності 121 «Інженерія програмного забезпечення», впроваджені в поточну діяльність магазину «Хан-Тенгрі», а саме:

- Односторінковий веб-застосунок (SPA) для автоматизації обліку прокату туристичного спорядження з чітким розділенням на front-end і back-end частини.
- Front-end (React.js) — реалізований із компонентним підходом і керуванням станом через Context API; забезпечує динамічне підвантаження та кешування даних.
- Back-end (Node.js + Express) – реалізує RESTful API для керування даними про оренду, товари, клієнтів і користувачів з валідацією та автентифікацією.
- База даних (MySQL) – зберігає інформацію про товари, клієнтів, замовлення та історію прокатів.
- Деплоймент на хостингу з FastPanel – забезпечує стабільність роботи, швидке розгортання та централізоване адміністрування.

Запропоноване програмне забезпечення дозволяє:

- централізовано зберігати та оперативно оновлювати дані про товари, клієнтів і фінансові операції;
- мінімізувати обсяг ручних операцій і ймовірність помилок при обробці замовлень;

– скоротити час обробки замовлень та прискорити обслуговування клієнтів;

– здійснювати контроль термінів повернення спорядження та своєчасне фіксування оплат і заборгованості.

Впровадження зазначених результатів сприяє підвищенню якості обслуговування клієнтів магазину «Хан-Тенгрі», оптимізації бізнес-процесів та підвищенню загальної ефективності роботи персоналу.

Директор
ФОП «Пішеходько
Н.О.»