БАКАЛАВРСЬКА РОБОТА

БР.КІ-01.00.00.000 ПЗ

Група КІ-22-1

Агафонкін Богдан

2026

Міністерство освіти і науки України
Івано-Франківський національний технічний університет нафти і газу
Факультет інформаційних технологій
Кафедра комп'ютерних систем і мереж

Агафонкін Богдан Владиславович

УДК 004.42

БАКАЛАВРСЬКА РОБОТА

Розробка web-сайту більярдного клубу з функцією пріоритетного бронювання засобами HTML, PHP та SQL

Комп'ютерна інженерія
(назва освітньої програми)

123 - Комп'ютерна інженерія
(шифр і назва спеціальності)

Робота містить результати власних досліджень, використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело:

Здобувач освітнього ступеня _____ *Агафонкін Б. В.* _____
(підпис, ініціали та прізвище здобувача)

Науковий керівник _____ *Гарасимів Т. Г., асист.* _____
(підпис, прізвище, ім'я, по батькові, науковий ступінь, вчене звання керівника)

Допущено до захисту
Завідувач кафедри КСМ

д.т.н.,професор _____ *С.І. Мельничук*
(посада) (підпис) (дата) (ініціали та прізвище)

Івано-Франківськ – 2026 рік

Івано-Франківський національний технічний університет нафти і газу

(повне найменування вищого навчального закладу)

Факультет *інформаційних технологій*

Кафедра *комп'ютерних систем і мереж*

Освітній ступінь *бакалавр*

Спеціальність *123 – Комп'ютерна інженерія*

(шифр і назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри КСМ

(С.І. Мельничук)

"*21*" *квітня* *2026* року

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТОВІ

Агафонкіну Богдану Владиславовичу

(прізвище, ім'я, по батькові)

1. Тема роботи *"Розробка web-сайту більярдного клубу з функцією пріоритетного бронювання засобами HTML, PHP та SQL"*

керівник роботи *Гарасимів Т. Г., асист.*,

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від "21" квітня 2026 року № 180/7

2. Термін подання студентом роботи *11 червня 2026 р*

3. Вихідні дані до роботи *Матеріали і результати отримані під час проходження переддипломної практики, методичні вказівки, технічна література.*

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) *1. Аналіз предметної області та постановка завдання. 2. Розробка алгоритму програмного забезпечення та структури бази даних. 3. Практична реалізація коду web-програми, створення та опис запитів бази даних.*

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання *02 лютого 2026 р.*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз предметної області та постановка завдання	Лютий, 2026	
2	Розробка алгоритму програмного забезпечення	Березень, 2026	
3	Розробка структури бази даних	Березень, 2026	
4	Створення бази даних та запитів до неї	Травень, 2026	
5	Реалізація коду web-програми	Травень, 2026	
6	Оформлення пояснювальної записки	Червень, 2026	

Студент _____ *Агафонкін Б. В.*
(підпис) (прізвище та ініціали)

Керівник роботи _____ *Гарасимів Т. Г.*
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

У дипломній роботі було виконано розробку web-сайту більярдного клубу з функціями пріорітетного бронювання засобами HTML, PHP та MySQL. На етапі теоретичного дослідження проведено аналіз сучасних підходів до створення веб-застосунків для сфери обслуговування, розглянуто принципи організації клієнт-серверної взаємодії, систем керування базами даних та методів обробки користувацьких запитів. Також було проаналізовано сучасні технології веброзробки, засоби забезпечення безпеки даних та особливості реалізації систем бронювання.

На етапі проектування розроблено алгоритм функціонування програмного забезпечення, який забезпечує взаємодію користувача із системою, обробку запитів, реєстрацію та авторизацію користувачів, а також створення та керування бронюваннями. Визначено основні сценарії використання системи та логіку роботи окремих модулів web-програми.

Практична частина роботи охоплює: реалізації таблиць для зберігання інформації про користувачів, більярдні столи та бронювання; розробку SQL-запитів для додавання, редагування, видалення й отримання необхідної інформації; створення користувацького інтерфейсу web-сайту, реалізацію взаємодії між клієнтською та серверною частинами, підключення бази даних MySQL та тестування працездатності всіх функцій системи. Особливу увагу приділено зручності користування сайтом, швидкості обробки запитів та коректності відображення інформації.

Результати виконаної роботи повністю відповідають поставленим вимогам та демонструють ефективність використання HTML, PHP і MySQL для створення сучасного web-сайту більярдного клубу з функціями пріорітетного онлайн-бронювання.

Ключові слова: web-сайт, більярдний клуб, онлайн-бронювання, HTML, PHP, MySQL, база даних.

SUMMARY

This diploma project involved the development of a billiards club website with priority booking features using HTML, PHP, and MySQL. During the theoretical research phase, an analysis was conducted of modern approaches to creating web applications for the service sector, and the principles of client-server interaction, database management systems, and methods for processing user requests were examined. Modern web development technologies, data security measures, and the specifics of implementing reservation systems were also analyzed.

During the design phase, an algorithm was developed to govern the software's operation, enabling user interaction with the system, request processing, user registration and authentication, as well as the creation and management of reservations. The main usage scenarios for the system and the operational logic of the web application's individual modules were defined.

The practical part of the work covers: the implementation of tables for storing information about users, billiard tables, and reservations; the development of SQL queries for adding, editing, deleting, and retrieving necessary information; the creation of the website's user interface, the implementation of interaction between the client-side and server-side components, the connection to a MySQL database, and the testing of the functionality of all system features. Particular attention was paid to the website's user-friendliness, query processing speed, and the correct display of information.

The results of the work fully meet the set requirements and demonstrate the effective use of HTML, PHP, and MySQL to create a modern billiards club website with priority online booking features.

Keywords: website, billiards club, online booking, HTML, PHP, MySQL, database.

ЗМІСТ

ВСТУП

Сучасний етап розвитку інформаційного суспільства характеризується тотальною цифровізацією всіх аспектів людської діяльності, де швидкість та зручність отримання послуг стають визначальними чинниками конкурентоспроможності будь-якого бізнесу. Веб-технології трансформували традиційні методи взаємодії між надавачами послуг та споживачами, перетворивши статичні інформаційні сторінки на потужні інтерактивні платформи. Впровадження автоматизованих систем дозволяє підприємствам не лише оптимізувати внутрішні адміністративні процеси, а й забезпечити клієнтам цілодобовий доступ до сервісів, що є критично важливим в умовах високої динаміки сучасного ринку.

Сфера відпочинку та розваг, зокрема робота більярдних клубів, стикається з викликами складного розподілу ресурсів та управління часовими інтервалами. Традиційні способи бронювання за допомогою телефонних дзвінків або паперових журналів обліку часто призводять до виникнення помилок “людського фактору”, накладання графіків та втрати потенційних відвідувачів через відсутність візуальної наочності вільних місць. Саме тому розробка спеціалізованої веб-орієнтованої системи, що поєднує в собі надійну логіку серверної обробки даних та інтуїтивно зрозумілий інтерфейс, є необхідним кроком для виведення якості обслуговування на сучасний рівень.

Актуальність теми полягає в тому, що процеси цифровізації сучасної сфери послуг вимагають від бізнесу впровадження ефективних інструментів взаємодії з клієнтами. Для закладів сфери відпочинку, зокрема більярдних клубів, критично важливим є перехід від застарілих методів обліку (паперових журналів або електронних таблиць) до автоматизованих інформаційних систем. Проблема існуючих підходів полягає у високій імовірності виникнення конфліктів часових інтервалів (овербукінгу), відсутності візуальної наочності для клієнта та неефективності зворотного зв’язку.

Питання проектування систем автоматизації та архітектури веб-сервісів висвітлені у працях таких зарубіжних фахівців, як М. Фаулер та Е. Фрімен, а також у роботах вітчизняних дослідників, які займаються питаннями впровадження інформаційних технологій в економічний сектор. Проте, незважаючи на значну кількість готових SaaS-рішень, існує потреба в розробці легких, кастомізованих систем, що враховують специфіку інвентарного обліку (кількість та типи столів) та забезпечують високий рівень безпеки даних за низьких витрат на впровадження. Оцінка сучасного стану завдання показує, що реалізація веб-орієнтованої системи з інтерактивною сіткою бронювання є найбільш раціональним інженерним рішенням для малого та середнього бізнесу.

О б ’ є к т д о с л і д ж е н н я – процес надання та обліку послуг у сфері відпочинку та розваг за допомогою автоматизованих систем.

П р е д м е т д о с л і д ж е н н я – методи, алгоритми та програмні засоби розробки інформаційної системи онлайн-бронювання більярдних столів з використанням технологій PHP, MySQL та AJAX.

М е т а р о б о т и – розробити інформаційну систему онлайн-бронювання “PoolKing”, яка забезпечує автоматизований вибір часових інтервалів, візуальний контроль завантаженості закладу та розмежування прав доступу для різних категорій користувачів.

Для досягнення поставленої мети необхідно виконати такі завдання:

− провести аналіз бізнес-процесів більярдного клубу та визначити функціональні вимоги до інформаційної системи;

− спроєктувати реляційну базу даних для зберігання відомостей про клієнтів, більярдні столи та бронювання;

− розробити механізм перевірки доступності ресурсів з метою уникнення конфліктів часових інтервалів;

− створити користувацький інтерфейс із застосуванням сучасних підходів до вебдизайну, зокрема стилю Glassmorphism, та інтерактивних компонентів;

− реалізувати технологію асинхронного оновлення даних без пере-завантаження сторінок за допомогою AJAX та Fetch API;

− розробити функціональні модулі для клієнтів (створення, редагування і видалення бронювання) та менеджерів (панель керування).

Методи дослідження – у роботі використано: метод системного аналізу для формування структури проекту; методи моделювання баз даних (ER-діаграми) для побудови логічної схеми збереження даних; алгоритмічне моделювання для реалізації логіки бронювання; методи веброзробки для програмної реалізації серверної та клієнтської частин системи.

Практичне значення полягає у створенні готового до експлуатації програмного продукту, який може бути впроваджений у діючі більярдні клуби. Використання розробленої системи дозволяє мінімізувати помилки персоналу, підвищити лояльність клієнтів через зручність онлайн-запису та оптимізувати роботу менеджера закладу. Окремі модулі системи (наприклад, інтерактивна сітка часу) можуть бути адаптовані для інших типів орендного бізнесу.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Огляд нормативної та технічної документації у сфері надання послуг

Функціонування сучасних інформаційних систем у галузі послуг та відпочинку регулюється низкою правових актів та технічних регламентів. Це забезпечує юридичну легітимність взаємодії між закладом та клієнтом, а також визначає архітектурні вимоги до програмного продукту. Проектування системи онлайн-бронювання “PoolKing” базується на нормах чинного законодавства України та міжнародних стандартах веброзробки.

Основою правового регулювання у сфері надання послуг є Закон України “Про захист прав споживачів” [1]. Відповідно до ст. 15 цього нормативного акта, інформаційна система повинна забезпечувати надання повної та достовірної інформації про послугу до моменту її придбання чи бронювання. У контексті більярдного клубу це зумовлює необхідність точного відображення в інтерфейсі вартості оренди, технічних характеристик столів (тип гри: пул, снукер, російський більярд) та чітко встановлених часових меж надання послуг.

Оскільки робота системи передбачає збір та обробку інформації про користувачів (ім’я, адреса електронної пошти), обов’язковим є дотримання вимог Закону України “Про захист персональних даних” [2]. Даний норматив визначає засади збереження конфіденційності та безпеки інформації. На технічному рівні це вимагає впровадження методів криптографічного захисту, зокрема незворотного хешування паролів, а також використання підготовлених SQL-запитів для запобігання несанкціонованому доступу до бази даних.

Якість розроблюваного програмного забезпечення оцінюється згідно зі стандартами ДСТУ ISO/IEC 25010 [3]. Пріоритетними характеристиками для системи бронювання визначено функціональну придатність (Functional Suitability) та зручність використання (Usability). Технічна реалізація клієнтської частини сайту здійснюється з урахуванням стандартів консорціуму W3C (World Wide Web Consortium), що гарантує кросбраузерність та коректне відображення

інтерактивних елементів інтерфейсу, таких як сітка годин та форми вибору ресурсів.

Окрім загальнодержавних стандартів, логіка функціонування системи опирається на внутрішню технічну документацію закладу: прейскурант цін та правила внутрішнього розпорядку. Програмне впровадження цих регламентів дозволяє автоматизувати контроль за дотриманням графіка роботи клубу (10:00-22:00) та забезпечити прозорість фінансових розрахунків залежно від категорії обраного обладнання. Таким чином, інформаційна система виступає цифровим засобом реалізації адміністративних правил та стандартів обслуговування більярдного клубу.

1.2 Техніко-експлуатаційна характеристика та аналіз локального ринку

Ефективне проектування інформаційної системи для більярдного клубу потребує глибокого аналізу об’єктів, що підлягають автоматизації. Більярдний стіл у даному контексті виступає не лише як спортивний інвентар, а як основний ресурс закладу, що має певні технічні характеристики, тривалість експлуатаційного циклу та специфіку попиту.

В сучасній індустрії відпочинку виділяють три основні типи ігрових столів, кожен з яких має бути відображений в базі даних системи “PoolKing” з урахуванням її особливостей:

− Американський пул (Pool). Найбільш поширений тип гри для масового сегмента. Використовуються столи розміром від 7 до 9 футів. Для професійних клубів стандартом є 9-футовий стіл (ігрова поверхня 2.54 х 1.27 м). Технічно пул характеризується широкими створами луз (до 135 мм) та синтетичним сукном, що забезпечує високу швидкість руху куль. З точки зору бізнес-логіки, пул є динамічною грою з високим коефіцієнтом ротації клієнтів, що вимагає від системи можливості бронювання коротких часових інтервалів (від 1 години).

– П і р а м і д а (Pyramid). Характеризується найвищим рівнем складності через мінімальний зазор між кулею та бортом лузи. Стандартний професійний стіл має розмір 12 футів (3.50 м х 1.75 м) та вагу до 1200 кг, оскільки в основі ігрового поля лежить плита з натурального сланцю (ардезії). Через специфіку гри, партія в піраміду може тривати значно довше, ніж в пул, що вимагає впровадження в системі “PoolKing” функціоналу для вибору тривалих часових слотів та відповідної цінової диференціації.

– С н у к е р (Snooker). Англійський різновид більярду, що потребує максимальної концентрації та точності. Столи для снукеру також мають розмір 12 футів, але відрізняються типом сукна (з ворсом) та геометрією луз, які мають закруглені краї. В інформаційній системі снукерні столи виділяються в окрему категорію через їхню обмежену кількість та специфічну аудиторію гравців.

Окремим об’єктом автоматизації виступають VIP-кімнати (VIP-zones). Це закриті локації, що, окрім більярдного столу преміум-класу, можуть бути обладнані м'якими зонами відпочинку, мультимедійними системами або ігровими консолями. Для бази даних VIP-кімната є складним об’єктом, оскільки її оренда передбачає не лише оплату за ігровий стіл, а й плату за ексклюзивність простору. Програмна реалізація бронювання таких зон потребує додаткових перевірок прав доступу та можливості відображення розширеного опису послуг.

Аналіз ринку більярдних послуг у Івано-Франківську демонструє великий попит на цей вид дозвілля, проте виявляє значний технологічний дефіцит у методах управління клієнтами. Розглянемо роботу провідних закладів міста:

– Більярдний клуб “Buffalo” (вул. Незалежності): Один із найбільших закладів, що орієнтований на професійний спорт. Тут представлена значна кількість столів для пулу і піраміди. Але, аналіз показує, що бронювання здійснюється переважно через адміністратора за телефоном. Клієнт не має змоги візуально оцінити завантаженість залу на вечірній час, що призводить до частих відмов через відсутність вільних місць.

– Клуб “Пасаж” (вул. Вічевий Майдан): Популярне місце відпочинку в центрі міста. Бронювання тут часто відбувається на місці або через прямі

повідомлення в соціальних мережах (Instagram). Такий метод не є ефективним, оскільки повідомлення можуть оброблятися із затримкою, а менеджер змушений вручну зіставляти запити з різних каналів зв'язку, що створює ризик помилок.

– Клуб "Піраміда": Спеціалізується на класичних видах більярду. Використовує традиційний паперовий журнал обліку. Це унеможливлює ведення аналітики щодо популярності окремих столів або годин пікового навантаження, а також виключає можливість віддаленого контролю з боку власника закладу.

Загальною проблемою для всіх вищезгаданих закладів є відсутність інтерактивного клієнтського сервісу. Клієнт змушений витрачати час на комунікацію замість того, щоб за декілька секунд обрати потрібний стіл на екрані смартфона. Відсутність автоматизованої перевірки конфліктів часових інтервалів (овербукінгу) призводить до конфліктних ситуацій, коли на один час претендують два різні відвідувачі.

Для глибшого аналізу методів управління клієнтськими потоками доцільно систематизувати переваги та недоліки існуючих підходів у порівнянні з розроблюваною системою "PoolKing". Це дозволить наочно продемонструвати доцільність впровадження веб-орієнтованого рішення. В таблиці 1.1 наведено порівняльний аналіз методів бронювання послуг:

Таблиця 1.1 – Порівняльний аналіз методів бронювання послуг

Критерій порівняння	**Паперовий журнал (офлайн)**	**Телефонний дзвінок**	**Соціальні мережі (Instagram/Viber)**	**Вебсайт клубу (PoolKing)**
1	**2**	**3**	**4**	**5**
Доступність для клієнта	Тільки в закладі	Тільки в робочі години	Цілодобово (із затримкою)	Цілодобово (миттєво)
Візуалізація вільних місць	Відсутня	Відсутня (тільки зі слів менеджера)	Відсутня	Інтерактивна сітка годин
Ризик "овербукінгу" (накладання)	Високий (людський фактор)	Високий (людський фактор)	Середній (затримка листування)	Нульовий (автоматична перевірка)
Швидкість підтвердження	Миттєво (на місці)	Миттєво (в розмові)	Від 5хв до кількох годин	Миттєво (автоматично)

Кінець таблиці 1.1

1	2	3	4	5
Витрати часу менеджера	Високі	Високі	Дуже високі	Мінімальні
Збереження даних (безпека)	Низьке (журнал можна загубити)	Низьке (якщо не фіксується документально)	Середнє (історія чату)	Високе (база даних)
Можливість аналітики	Тільки ручний підрахунок	Відсутня / ручний підрахунок	Обмежена	Повна (автоматичні звіти)

Аналіз даних таблиці 1.1 підтверджує, що традиційні методи бронювання, які наразі домінують на ринку Івано-Франківська, мають критичні недоліки. Найбільшою проблемою є високе навантаження на адміністратора, який змушений виконувати роль “комунікаційного вузла”, вручну обробляючи запити з різних каналів. Це не лише сповільнює процес, а й створює передумови для помилок, що негативно впливає на репутацію закладу.

Впровадження інформаційної системи “PoolKing” дозволяє перенести основну частину операційної роботи на програмний алгоритм. Клієнт отримує автономність у виборі послуги, а менеджер – чудовий інструмент моніторингу, де всі дані вже структуровані та перевірені на логічну цілісність. Такий підхід трансформує процес бронювання із “сервісу за запитом” у сучасний Self-Service, що відповідає стандартам цифрової економіки.

Отже, розробка системи “PoolKing” є відповіддю на реальні проблеми локального бізнесу. Впровадження веб-платформи з живою сіткою годин дозволить закладам Івано-Франківська вийти на новий рівень сервісу, забезпечивши прозорість, швидкість та 100% точність обліку ігрових ресурсів.

1.3 Аналіз наявних методів та засобів автоматизації бронювання

Автоматизація процесів бронювання у сфері послуг на сьогодні реалізується за допомогою декількох основних підходів: від використання готових хмарних платформ до розробки індивідуальних програмних рішень. Кожен із методів має

свої технічні та економічні особливості, які впливають на вибір інструментарію для конкретного бізнес-завдання.

Найпростішим методом автоматизації є використання SaaS-рішень (Software as a Service), таких як Bookly (рис. 1.1), SimplyBook.me або EasyWeek. Основною перевагою таких систем є швидкість впровадження та наявність готових модулів для оплати й сповіщень [4].

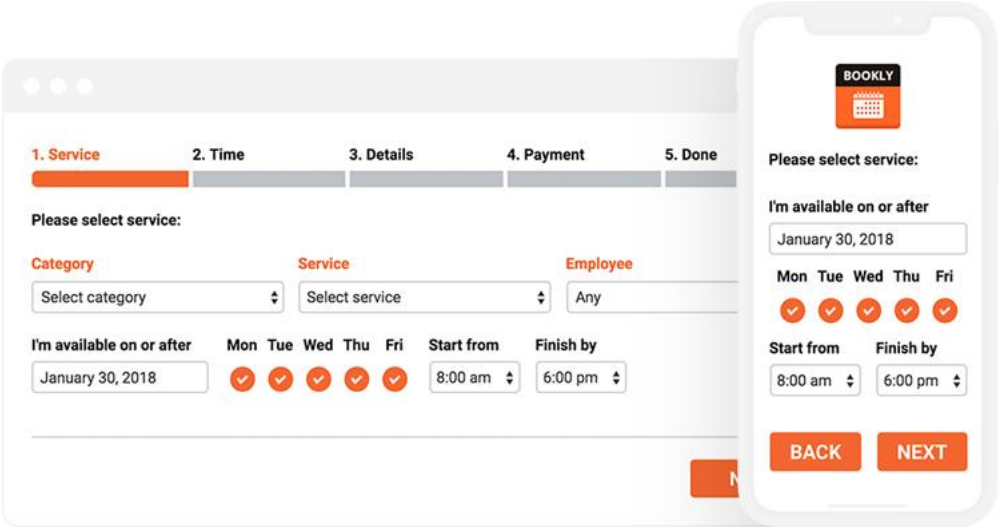

Рисунок 1.1 – Вигляд сервісу бронювання Bookly

Проте аналіз показує, що для спеціалізованих закладів, таких як більярдні клуби, готові платформи часто виявляються надто універсальними. Вони не завжди дозволяють реалізувати специфічну візуальну сітку завантаженості для різних типів столів (пул, снукер) та вимагають щомісячної абонентської плати, що збільшує фінансове навантаження на малий бізнес. Окрім цього, при використанні сторонніх сервісів дані клієнтів зберігаються на серверах провайдера, що обмежує контроль власника закладу над безпекою інформації.

Іншим підходом є впровадження модулів для систем управління контентом (CMS), наприклад, WordPress чи Joomla. Це дозволяє інтегрувати систему бронювання у вже існуючий сайт. Однак такі плагіни часто мають надлишкову функціональність, що негативно впливає на швидкість завантаження сторінок та ускладнює інтерфейс для кінцевого користувача. Крім того, архітектурна закритість багатьох готових плагінів перешкоджає реалізації унікальної логіки,

наприклад, динамічної зміни тривалості бронювання залежно від наявності вільних інтервалів у реальному часі.

Найбільш гнучким методом є індивідуальна розробка із використанням мов програмування серверної сторони (PHP, Python) та систем управління базами даних (MySQL, PostgreSQL). Такий підхід дозволяє створити максимально легкий та швидкий програмний продукт, позбавлений зайвих функцій. Саме використання стеку технологій PHP та MySQL дає змогу реалізувати унікальну архітектуру “PoolKing”, де логіка перевірки конфліктів часових інтервалів та інтерактивна візуалізація столів працюють як єдиний механізм. Нижче наведено таблицю 1.2 з порівняльним аналізом засобів автоматизації.

Таблиця 1.2 – Порівняльний аналіз систем бронювання

Критерій порівняння	**Готові SaaS-рішення (Easyweek, Bookly)**	**Плагіни для CMS (Wordpress Booking)**	**Індивідуальна розробка (проєкт “PoolKing”)**
Вартість експлуатації	Щомісячна абонентська плата	Одноразова купівля ліцензії / безкоштовно	Безкоштовно (після завершення розробки)
Контроль над даними	Дані зберігаються на серверах провайдера	Власна база даних у межах CMS	Повний контроль над базою даних
Гнучкість логіки	Обмежена стандартним функціоналом сервісу	Середня (залежить від налаштувань плагіна)	Абсолютна (реалізація будь-якої бізнес-логіки)
Швидкість роботи	Залежить від навантаження на сервіс	Можливе сповільнення через надлишковий код CMS	Максимальна (відсутність зайвих модулів)
Унікальність UI	Стандартні форми, обмежена стилізація	Шаблонний дизайн плагінів	Унікальний інтерфейс
Специфіка закладу	Складно адаптувати під різні типи столів	Шаблонні списки послуг	Інтерактивна візуальна сітка для 12 столів

На основі даних таблиці 1.2 можна зробити висновок, що індивідуальна розробка є найкращим рішенням для даного проєкту з точки зору унікальності та інтерактивності інтерфейсу. Попри те, що готові SaaS-платформи пропонують швидкий старт, вони створюють фінансову залежність від провайдера та не дозволяють реалізувати специфічну візуальну сітку годин, яка є критично важливою для зручності клієнтів більярдного клубу.

Використання власного програмного забезпечення гарантує високу продуктивність системи за рахунок відсутності зайвого коду, що зазвичай притаманний універсальним плагінам для CMS, та забезпечує повну безпеку конфіденційних даних користувачів на власному серверному обладнанні.

1.4 Постановка задачі на розробку системи “PoolKing”

На основі проведеного аналізу предметної області, нормативної документації та існуючих засобів автоматизації, було визначено необхідність розробки індивідуального програмного рішення для більярдного клубу. Основним інженерним завданням даної роботи є створення інформаційної системи пріорітетного бронювання “PoolKing”, яка дозволить оптимізувати процес розподілу ресурсів закладу та підвищити якість обслуговування клієнтів.

Інформаційна система повинна забезпечувати виконання таких ключових завдань:

− Автоматизація обліку ресурсів: переведення даних про стан більярдних столів у цифровий формат із можливістю відстеження їх завантаженості в режимі реального часу.

− Управління користувачами: реалізація надійного механізму реєстрації та автентифікації відвідувачів, а також розмежування прав доступу між клієнтами та адміністративним персоналом.

− Візуалізація процесу бронювання: створення інтерактивного графічного інтерфейсу у вигляді сітки годин, що дозволяє користувачеві наочно обирати доступні часові інтервали.

− Запобігання конфліктам даних: впровадження логічного алгоритму, який унеможливлює створення накладених у часі бронювань на один і той самий стіл.

− Оперативне управління: надання менеджеру інструментів для глобального моніторингу замовлень, їх редагування та видалення з обов’язковим автоматичним сповіщенням клієнта про зміни.

Система базуватиметься на клієнт-серверній архітектурі з використанням асинхронної передачі даних для швидкодії інтерфейсу. Результатом розробки стане програмний комплекс, що автоматизує бронювання, мінімізує помилки персоналу та забезпечує цілодобовий онлайн-доступ до послуг закладу.

Висновок до розділу

У підсумку, проведений аналіз підтвердив потребу в автоматизації бронювання для усунення помилок “людського фактора” та підвищення якості сервісу. Сформовані вимоги та обґрунтування індивідуальної розробки стали основою для подальшого проектування архітектури й бази даних системи.

2 ПРОЕКТУВАННЯ МОДЕЛЕЙ ТА АЛГОРИТМІВ СИСТЕМИ

2.1 Специфікація вимог до системи: ролі користувачів та безпека даних

Формування чіткої специфікації вимог є критично важливим етапом проектування інформаційної системи “PoolKing”, оскільки саме ці параметри визначають стабільність взаємодії між клієнтами та адміністрацією більярдного клубу. Системний підхід до опису вимог дозволяє ще на етапі розробки мінімізувати ризики виникнення логічних помилок та забезпечити відповідність програмного продукту бізнес-завданням закладу. Усі вимоги до системи традиційно поділяються на дві основні групи: функціональні, що описують перелік дій та можливостей сервісу, та нефункціональні, які стосуються надійності, продуктивності та безпеки обробки інформації [6].

Основним завданням специфікації для даного проекту є створення надійного механізму управління обмеженими ресурсами (більярдними столами). Це вимагає впровадження жорстких правил обробки вхідних запитів, де кожен елемент системи (від форми вибору дати до фінальної кнопки підтвердження) працює у межах встановлених технічних обмежень. Таке структурування не лише спрощує процес розробки та тестування коду, а й гарантує цілісність даних у базі, запобігаючи випадковим втратам або некоректному використанню ресурсів клубу.

Важливим аспектом проектування є визначення функціональних вимог, які описують перелік дій, доступних для різних груп користувачів, та встановлюють межі їхньої взаємодії з ресурсами сайту. У системі “PoolKing” реалізовано чітке розмежування прав доступу на основі трьох основних ролей: Гість, Клієнт та Менеджер. Це дозволяє не лише структурувати бізнес-процеси більярдного клубу, а й забезпечити належний рівень захисту інформації.

Для ролі Гостя, що представляє неавторизованих відвідувачів, встановлено найбільш обмежений набір можливостей. Гість може переглядати загальну інформацію на головній сторінці, ознайомлюватися з галереєю залів, типами столів та прейскурантом цін. Оскільки інтерактивна сітка годин та система

бронювання містять дані про завантаженість закладу, доступ до них для цієї категорії користувачів заблоковано. Таким чином, основною функціональною вимогою для Гостя є можливість ознайомлення з сервісом та проходження процедури реєстрації для отримання повного доступу.

Після успішної автентифікації користувач отримує роль К л і є н т а (User), що відкриває доступ до основного інструментарію системи. Функціональні вимоги для цієї ролі охоплюють повний цикл управління замовленням: від візуального вибору вільного часового проміжку на сітці до контролю статусу броні у власному профілі. Клієнт може самостійно планувати візит, обираючи категорію столу (пул, снукер чи російський більярд), а також має право на редагування параметрів свого запису або його повне скасування в разі зміни планів.

Роль М е н е д ж е р а (Manager) розроблена для адміністративного управління роботою клубу та передбачає розширені повноваження. Основною вимогою до цього рівня доступу є можливість глобального моніторингу завантаженості всіх 12 столів одночасно. Менеджер працює зі спеціалізованою панеллю керування, де дані структуровані за допомогою вкладок для кожного столу. Окрім стандартних функцій, адміністратор наділений правом втручання в будь-яке бронювання в системі – від зміни часу до видалення запису. Такий функціонал доповнюється вимогою автоматичного зворотного зв’язку: у разі примусового скасування броні менеджером, система ініціює відправку інформаційного повідомлення на електронну пошту клієнта [6].

Детальний розподіл прав доступу між Гостем, Клієнтом та Менеджером наведено у таблиці 2.1.

Таблиця 2.1 – Розподіл прав доступу

Функціональна можливість	Гість (неавторизований)	Клієнт (User)	Менеджер (Manager)
1	2	3	4
Перегляд головної сторінки та галереї залів	+	+	+
Реєстрація акаунту та авторизація	+	+	+
Перегляд інтерактивної сітки вільних годин	-	+	+

Кінець таблиці 2.1

1	2	3	4
Бронювання столика на обраний час	-	+	+
Керування власними бронюваннями (зміна/скасування)	-	+	+
Перегляд завантаженості всіх 12 столів одночасно	-	-	+
Видалення або редагування чужих записів	-	-	+
Отримання Email-сповіщень про статус замовлення	-	+	+

Окрім функціональних можливостей, важливе значення мають нефункціональні вимоги, які визначають якісні характеристики роботи системи “PoolKing”. Першочерговою серед них є вимога до зручності використання (Usability). Інтерфейс має бути інтуїтивно зрозумілим, щоб клієнт міг здійснити бронювання з мінімальною кількістю дій. Для цього реалізовано візуальну легенду кольорів та інтерактивний відгук: при виборі тривалості гри система автоматично підсвічує відповідну кількість часових слотів на сітці, що дозволяє користувачеві наочно бачити обраний інтервал перед підтвердженням.

Ще однією важливою нефункціональною вимогою є продуктивність та швидкість відгуку. Використання методів асинхронного обміну даними дозволяє оновлювати інформацію про стан столів у реальному часі без повного перезавантаження сторінок, що значно підвищує швидкість взаємодії користувача з інтерфейсом. Система повинна обробляти запити до бази даних у режимі реального часу, миттєво блокуючи доступ до щойно заброньованих годин для інших відвідувачів. Також висуваються вимоги до естетичного оформлення інтерфейсу: застосування стилю Glassmorphism (скляні панелі) та темної кольорової гами не лише створює атмосферу більярдного клубу, а й знижує втому очей менеджера при тривалій роботі з панеллю керування [7].

Завершальним, проте критично важливим блоком специфікації є вимоги до безпеки та захисту інформації. Оскільки система передбачає роботу з персональними даними відвідувачів та управлінням ресурсами закладу, вона

повинна забезпечувати надійний захист від несанкціонованого доступу та мережевих атак. Першочерговою вимогою є впровадження механізму автентифікації та авторизації, який базується на унікальних ідентифікаторах сесій. Це дозволяє серверу розпізнавати користувача під час переходу між сторінками та обмежувати доступ до адміністративних функцій на основі визначених ролей.

Особлива увага приділяється захисту конфіденційної інформації, зокрема паролів. Відповідно до стандартів безпеки, збереження паролів у відкритому вигляді є неприпустимим, тому система повинна використовувати алгоритми незворотного криптографічного хешування. Це гарантує, що навіть у разі отримання доступу до бази даних зловмисники не зможуть відновити реальні паролі користувачів.

Окрім цього, архітектура взаємодії з базою даних має передбачати використання підготовлених запитів із параметризацією. Такий підхід повністю нівелює загрозу SQL-ін’єкцій, запобігаючи можливості підміни або видалення даних через поля вводу.

Додатково висуваються вимоги до фільтрації всіх вхідних даних для захисту від міжсайтового скриптингу (XSS). Система повинна автоматично очищувати текстові поля від потенційно шкідливого коду. Цілісність інформації забезпечується на рівні логічної структури бази даних через використання механізмів зв’язку між таблицями, що дозволяє уникнути появи “залишкових” або некоректних записів при видаленні користувачів чи скасуванні бронювань. Дотримання цих вимог дозволяє побудувати захищене середовище, яке відповідає сучасним критеріям кібербезпеки та захисту приватності [8].

Нижче в таблиці 2.2 наведено специфікацію вимог до безпеки та захисту даних.

Таблиця 2.2 – Специфікація вимог до безпеки та захисту даних

Об’єкт захисту	Вимога до реалізації	Захист від загрози
Паролі користувачів	Використання алгоритмів незворотного криптографічного хешування	Несанкціоноване зчитування паролів адміністратором або зловмисником
Запити до бази даних	Застосування підготовлених запитів із використанням параметрів	Впровадження стороннього коду (SQL-ін’єкції)
Контроль доступу	Перевірка ролей через ідентифікатори сесій на рівні сервера	Прямий перехід на сторінки менеджеру неавторизованими особами
Текстові поля вводу	Фільтрація та очищення символів у вхідних даних	Міжсайтовий скриптинг (XSS-атаки)
Цілісність даних	Логічне обмеження зв’язків між об'єктами через механізм зовнішніх ключів	Поява помилкових записів або “сміттєвих” даних у таблицях

Впровадження цих вимог на етапі проектування гарантує високу надійність системи “PoolKing”.

2.2 Розробка структурної схеми та моделі бази даних

Робота над інформаційною системою “PoolKing” починається з планування того, як він буде працювати. На цьому етапі потрібно чітко визначити, хто буде користуватися сайтом і які основні дії вони будуть виконувати. Це допоможе побудувати логічну схему, де головними учасниками є Клієнт і Менеджер, а головним об’єктом – більярдний стіл.

Вся робота системи будується навколо процесу “Бронювання”. Клієнт заходить на сайт, вибирає вільний стіл і час, а система перевіряє, чи можна зробити цей запис. Якщо все добре, дані зберігаються, і менеджер відразу бачить нове замовлення у себе в панелі керування.

Щоб розібратися, як програма влаштована всередині, була розроблена структурна схема, яка подана на рисунку 2.1.

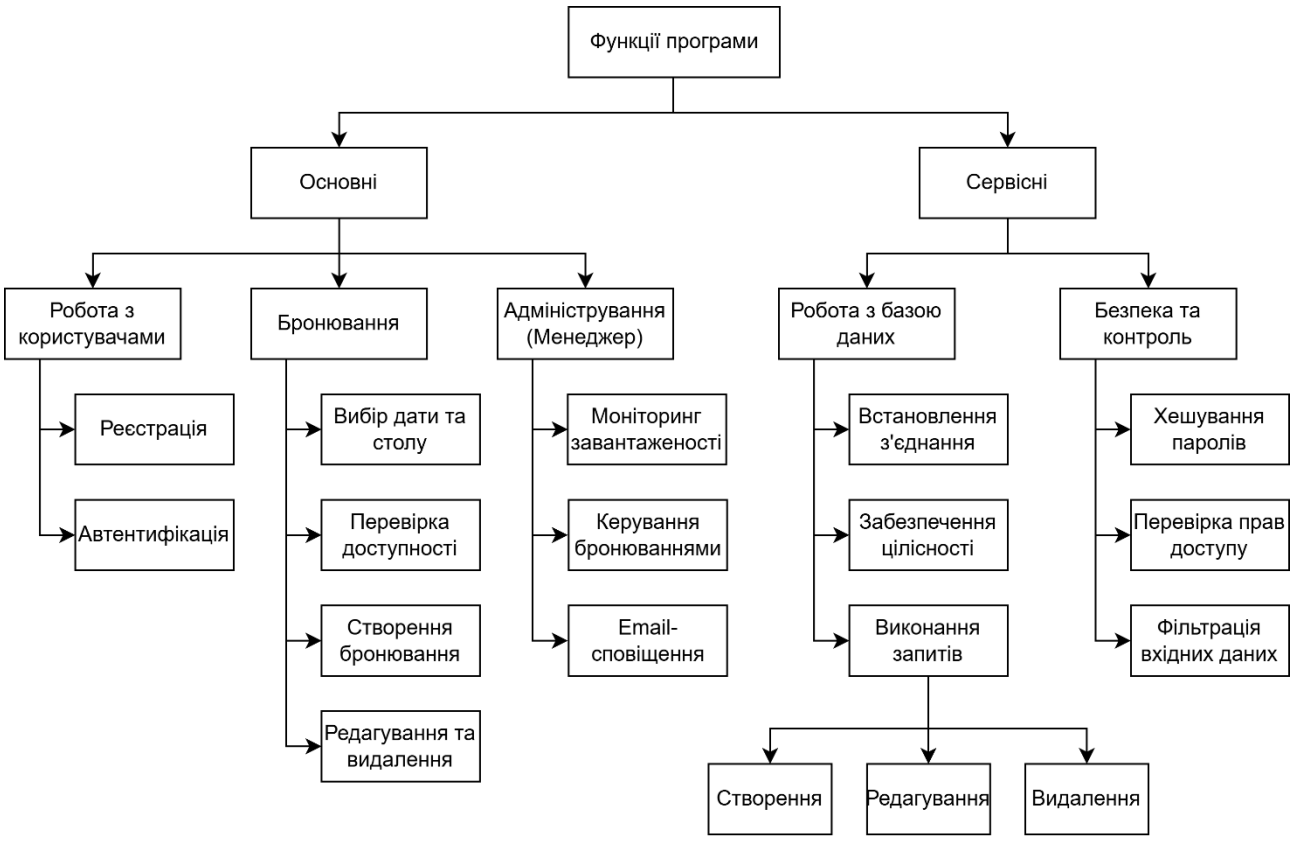

Рисунок 2.1 – Структурна схема програми

На рисунку 2.1 показано, з яких частин складається система. Усі можливості програми можна розділити на дві великі групи:

– О с н о в н і ф у н к ц і ї : те, з чим безпосередньо працюють люди. Сюди входить реєстрація нових користувачів, вхід у свій акаунт, зручна сітка для вибору часу та столів, а також особистий кабінет клієнта. Для менеджера це спеціальна панель, де він бачить усі замовлення за день.

– С е р в і с н і ф у н к ц і ї : це робота системи, яку користувач не бачить, але без якої вона не буде працювати. Це підключення до бази даних, перевірка паролів, захист від зламу та головне – перевірка того, щоб два різні клієнти не могли забронювати один стіл на той самий час.

Щоб усі ці функції працювали правильно і ніякі дані не губилися (наприклад, ім'я клієнта чи номер його стола), потрібна база даних. Вона є фундаментом системи. Тому було детально розроблено структуру таблиць, де буде зберігатися вся інформація про клієнтів, столи та кожну заброньовану годину.

Побудова надійної архітектури збереження даних є фундаментальним етапом проектування інформаційної системи “PoolKing”. Саме структура бази даних визначає, наскільки швидко та коректно система буде обробляти запити користувачів, контролювати завантаженість закладу та зберігати конфіденційну інформацію. На етапі розробки моделі необхідно трансформувати виявлені раніше інформаційні потоки у чітко визначені сутності та встановити логічні зв’язки між ними.

Для організації надійного зберігання та ефективного управління даними в інформаційній системі “PoolKing” обрано реляційну модель бази даних. Такий вибір зумовлений структурованістю інформаційних потоків закладу, де кожен об'єкт (клієнт, більярдний стіл чи запис про бронювання) має чітко визначений набір атрибутів та сталі зв’язки між ними.

Використання реляційного підходу забезпечує кілька критично важливих переваг для системи бронювання:

– Цілісність та несуперечливість даних: завдяки механізмам первинних та зовнішніх ключів система автоматично контролює логічний зв'язок між таблицями. Це унеможливлює ситуації, коли бронювання може бути створене для неіснуючого столу або анонімного користувача без облікового запису.

– Забезпечення транзакційності: процес бронювання ресурсу потребує суворого дотримання принципу атомарності операцій. Реляційна модель гарантує, що дані будуть збережені в базі лише у разі успішного завершення всіх перевірок на доступність часу, що є ключовим для запобігання помилкам при одночасному доступі кількох користувачів до одного ресурсу.

– Ефективність пошуку та фільтрації: структура реляційних таблиць дозволяє швидко виконувати складні запити з об’єднанням даних. Наприклад, для формування звіту менеджеру система має одночасно отримати інформацію про клієнта, технічну категорію столу та параметри часового інтервалу, що найефективніше реалізується саме в межах реляційної логіки.

– Нормалізація даних: застосування принципів нормалізації дозволяє мінімізувати надмірність інформації та усунути аномалії оновлення. Це значно зменшує обсяг збережених даних та спрощує подальше масштабування системи, наприклад, при додаванні нових залів або розширенні переліку додаткових послуг клубу.

Таким чином, реляційна модель виступає оптимальним фундаментом для розроблюваної системи, забезпечуючи високу швидкість обробки запитів та гарантуючи надійність збереження конфіденційної інформації відвідувачів більярдного клубу.

На основі аналізу бізнес-процесів більярдного клубу та вимог до функціоналу системи, було виділено три ключові сутності, які складають ядро бази даних. Кожна сутність має унікальний набір атрибутів, необхідних для коректного відображення інформації та виконання розрахунків.

– Користувачі (Users): ця сутність призначена для зберігання облікових даних усіх осіб, які мають доступ до системи. Основними атрибутами є унікальний ідентифікатор, ім'я, адреса електронної пошти, яка виступає унікальним логіном для входу, та пароль у зашифрованому вигляді. Критично важливим атрибутом є “роль”, що дозволяє системі розмежовувати права доступу на рівні програмного коду, відокремлюючи клієнтів від адміністративного персоналу.

– Більярдні столи (Billiard Tables): ця сутність описує матеріальні ресурси (столи для гри) закладу, що підлягають бронюванню. До її складу входять: порядковий номер столу, його категорія (американський пул, снукер або піраміда), вартість оренди за одну годину та тип залу (загальний чи VIP).

– Бронювання (Bookings): це центральна сутність, яка фіксує оренду конкретного столу конкретним користувачем. Вона містить часові та загальні параметри замовлення: користувача, стіл, дату, час початку та час завершення гри. Ця сутність виконує роль сполучної ланки, оскільки через систему зовнішніх ключів вона пов’язана з ідентифікаторами користувача та столу. Саме на основі

даних цієї сутності працює алгоритм перевірки доступності ресурсів та будується інтерактивна сітка завантаженості закладу.

Для візуалізації логічної структури бази даних та визначення механізмів взаємодії між об'єктами використовується ER-діаграма (Entity-Relationship). Вона дозволяє наочно представити архітектуру збереження даних, де центральне місце займає фіксація транзакцій бронювання.

На основі виділених раніше сутностей встановлюються такі типи логічних зв'язків:

– Зв'язок між “Користувачами” та “Бронюваннями”: між цими об'єктами реалізовано зв'язок типу “один-до-багатьох” (1:N). Це означає, що один зареєстрований користувач має можливість створювати необмежену кількість записів про бронювання протягом усього часу користування системою. Водночас кожне окреме бронювання може належати лише одному конкретному клієнту, що ідентифікується через його унікальний номер (ID).

– Зв'язок між “Більярдними столами” та “Бронюваннями”: тут також застосовується зв'язок типу “один-до-багатьох” (1:N). Один і той самий ігровий стіл може бути об'єктом оренди багато разів у різні дати та часові інтервали. При цьому в конкретний заброньований проміжок часу стіл залишається жорстко прив'язаним до одного запису в базі, що є основою для роботи алгоритму запобігання конфліктам.

Графічне представлення ER-діаграми показано на рисунку 2.2.

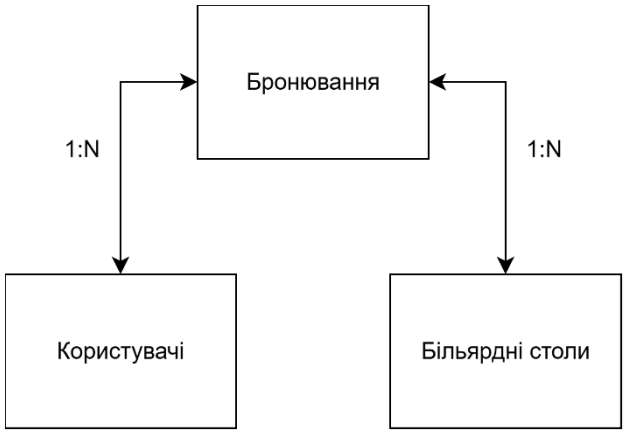

Рисунок 2.2 – ER-діаграма розроблюваної бази даних

Така модель є оптимальною для автоматизації більярдного клубу, оскільки вона виключає дублювання інформації та забезпечує високу швидкість виконання запитів при побудові графіків завантаженості.

Наступним етапом після розробки логічної схеми є фізичне проектування структури бази даних. Цей процес передбачає трансформацію теоретичних сутностей у конкретні таблиці з чітко визначеними типами даних, розмірністю полів та обмеженнями цілісності. Правильний вибір характеристик кожного поля дозволяє мінімізувати обсяг дискового простору для зберігання інформації та забезпечити максимальну швидкість виконання операцій читання та запису.

Повна архітектура бази даних із зазначенням усіх полів, типів зв'язків та ключів представлена на рисунку 2.3.

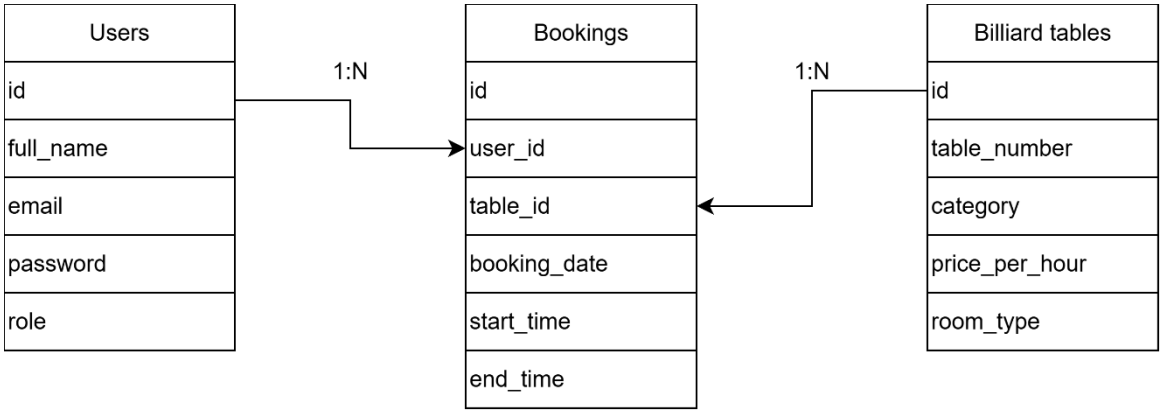

Рисунок 2.3 – Архітектура розроблюваної бази даних

Для детального аналізу параметрів збереження даних нижче наведено опис структури кожної таблиці інформаційної системи "PoolKing".

Таблиця "Users" призначена для ведення обліку всіх зареєстрованих осіб. Окрім ідентифікаційних даних, вона містить поле для збереження зашифрованого пароля, розмірність якого розрахована на зберігання довгих хеш-сум, що генеруються сучасними алгоритмами криптографічного захисту. Опис полів таблиці наведено в таблиці 2.3.

Таблиця 2.3 – Опис полів таблиці “Users”

Назва поля	Тип даних і розмір	Атрибути	Опис
id	INT	Первинний ключ Не нульове Автоінкремент	ID користувача
full_name	VARCHAR(100)	Не нульове	Ім’я користувача
email	VARCHAR(100)	Не нульове Унікальний	Електронна пошта користувача
password	VARCHAR(255)	Не нульове	Хеш пароля користувача
role	ENUM(user, manager)	Не нульове	Роль користувача

Таблиця “Billiard tables” містить технічну інформацію про ігрові столи закладу. Використання спеціальних числових типів для поля вартості дозволяє проводити точні фінансові розрахунки без похибок, що виникають при роботі з дробовими числами. Опис полів таблиці наведено в таблиці 2.4.

Таблиця 2.4 – Опис полів таблиці “Billiard tables”

Назва поля	Тип даних і розмір	Атрибути	Опис
id	INT	Первинний ключ Не нульове Автоінкремент	ID стола
table_number	INT	Не нульове Унікальний	Номер стола
category	ENUM(pool, snooker, piramid)	Не нульове	Категорія стола
price_per_hour	DECIMAL(10, 2)	Не нульове	Ціна оренди за годину
room_type	ENUM(default, vip)	Не нульове	Тип залу

Таблиця “Bookings” є центральним вузлом системи, де фіксується взаємодія клієнтів із ресурсами клубу. Особливістю цієї структури є використання форматів дати та часу, які дозволяють системі проводити математичні порівняння інтервалів під час перевірки завантаженості столів.

Опис полів таблиці наведено в таблиці 2.5.

Таблиця 2.5 – Опис полів таблиці “Bookings”

Назва поля	Тип даних і розмір	Атрибути	Опис
id	INT	Первинний ключ Не нульове Автоінкремент	ID бронювання
user_id	INT	Зовнішній ключ Не нульове	ID користувача, який здійснив бронювання. Зовнішній ключ до таблиці “Users”
table_id	INT	Зовнішній ключ Не нульове	ID стола, який заброньовано. Зовнішній ключ до таблиці “Tables”
booking_date	DATE	Не нульове	Дата бронювання
start_time	TIME	Не нульове	Час початку бронювання
end_time	TIME	Не нульове	Час кінця бронювання

При проектуванні таблиць до кожного поля було застосовано відповідні атрибути. Зокрема, ідентифікатори мають властивість автоматичного інкременту, що гарантує унікальність кожного запису без втручання користувача. Обмеження “Не нульове” встановлено для всіх полів, що запобігає появі порожніх або некоректних значень у системі.

Для забезпечення високої надійності та ефективності збереження даних, розроблена структура таблиць була приведена до третьої нормальної форми (3NF). Це дозволяє мінімізувати надмірність інформації, усунути дублювання та запобігти виникненню аномалій при вставці, оновленні чи видаленні записів.

Згідно з вимогами третьої нормальної форми, кожен неключовий атрибут таблиці повинен залежати лише від первинного ключа і не мати транзитивних залежностей від інших неключових полів. У контексті системи “PoolKing” це реалізовано шляхом винесення описових даних у окремі таблиці. Наприклад, у таблиці “Bookings” зберігаються лише ідентифікатори (ID) користувача та столу, а не їхні повні імена чи технічні характеристики. Це гарантує, що зміна вартості оренди столу або оновлення імені клієнта відбудеться

лише в одному місці, автоматично відобразившись у всіх пов'язаних записах системи.

Важливим аспектом підтримки цілісності бази даних є впровадження механізму каскадного видалення (“ON DELETE CASCADE”). В інформаційній системі більярдного клубу об'єкти жорстко пов'язані між собою, тому видалення головного запису без корекції залежних даних може призвести до появи “сміттєвої” інформації або логічних помилок.

Каскадне видалення у проекті працює за такими сценаріями:

– Видалення користувача: якщо обліковий запис клієнта видаляється із системи (наприклад, за запитом користувача або адміністратора), база даних автоматично видаляє всі пов'язані з ним записи про бронювання. Це дозволяє дотримуватися правил захисту персональних даних та звільняти дисковий простір.

– Видалення більярдного столу: у разі виведення ігрового столу з експлуатації та видалення його з таблиці “Більярдні столи”, всі майбутні або минулі записи про його оренду також будуть автоматично видалені. Це запобігає ситуаціям, коли менеджер бачить бронювання для столу, якого фізично не існує в закладі.

Використання обмежень зовнішніх ключів разом із каскадними операціями забезпечує повну автоматизацію контролю за станом даних. Це звільняє мову програмування від необхідності вручну перевіряти наявність пов'язаних записів перед видаленням, що підвищує загальну швидкість роботи та відмовостійкість системи.

2.3 Розробка Use-Case та UML діаграм

Для моделювання функціональних можливостей системи та опису того, як саме користувачі взаємодіють із системою “PoolKing”, було використано діаграми варіантів використання (Use-Case).

Перша діаграма відображає дії “Клієнта”. На рисунку 2.4 представлено повний цикл його взаємодії з інтерфейсом.

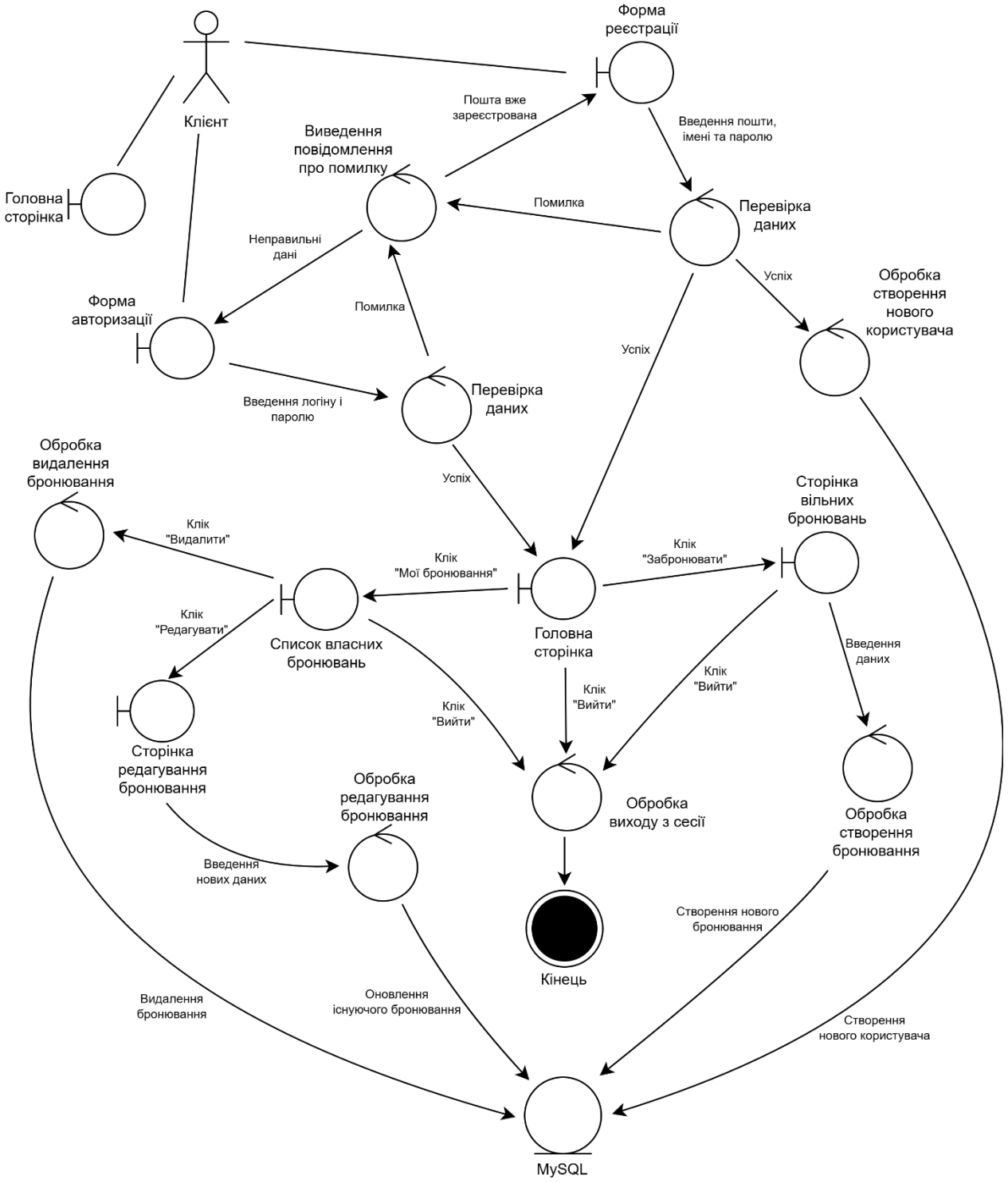

Рисунок 2.4 – Use-Case діаграма “Клієнта”

Згідно з цією моделлю, основний потік дій користувача включає такі варіанти використання, як реєстрація, авторизація та створення бронювання. Також передбачено функції для роботи у особистому кабінеті, зокрема перегляд історії та можливість самостійної зміни параметрів активної броні, якщо плани клієнта змінилися.

Друга діаграма описує можливості “Менеджера”. На рисунку 2.5 показано перелік адміністративних функцій.

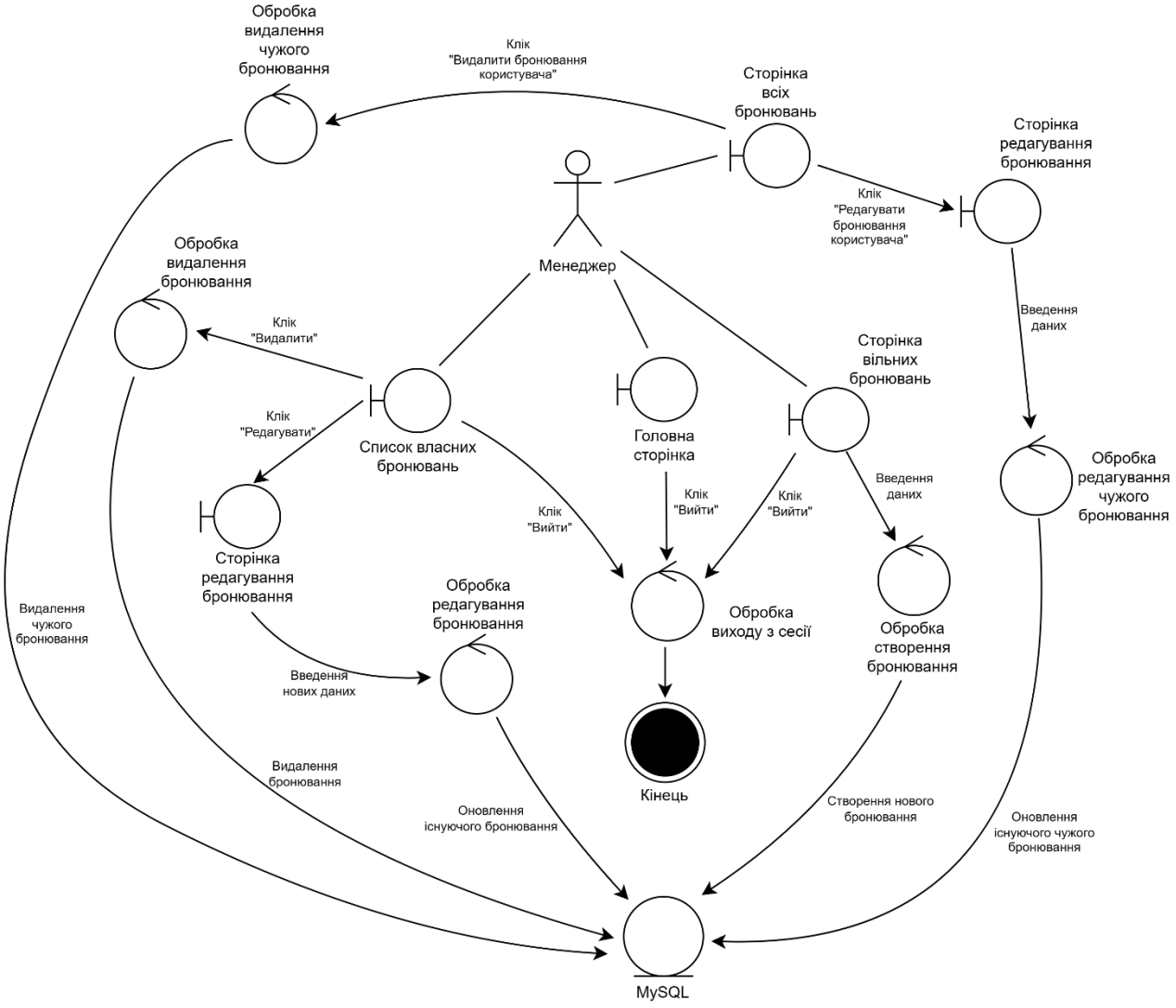

Рисунок 2.5 – Use-Case діаграма “Менеджера”

Основною функцією менеджера є моніторинг всіх бронювань. Він може редагувати або видаляти існуючі бронювання інших користувачів. Також менеджер може сам створити бронювання та побачити його на вже згаданій

сторінці кабінету на випадок якщо замовлення буде здійснено по телефону або на місці клієнтом.

Для опису динаміки роботи системи та розуміння того, як дані передаються між різними рівнями програмного комплексу, розроблено діаграму послідовності. Вона дозволяє відстежити часовий порядок взаємодії між користувачем, інтерфейсом браузера, серверними скриптами та базою даних.

На рисунку 2.6 представлено типовий процес отримання інформації (запит “SELECT”), який є базовим для більшості функцій системи, таких як перегляд вільної сітки годин або перевірка історії замовлень.

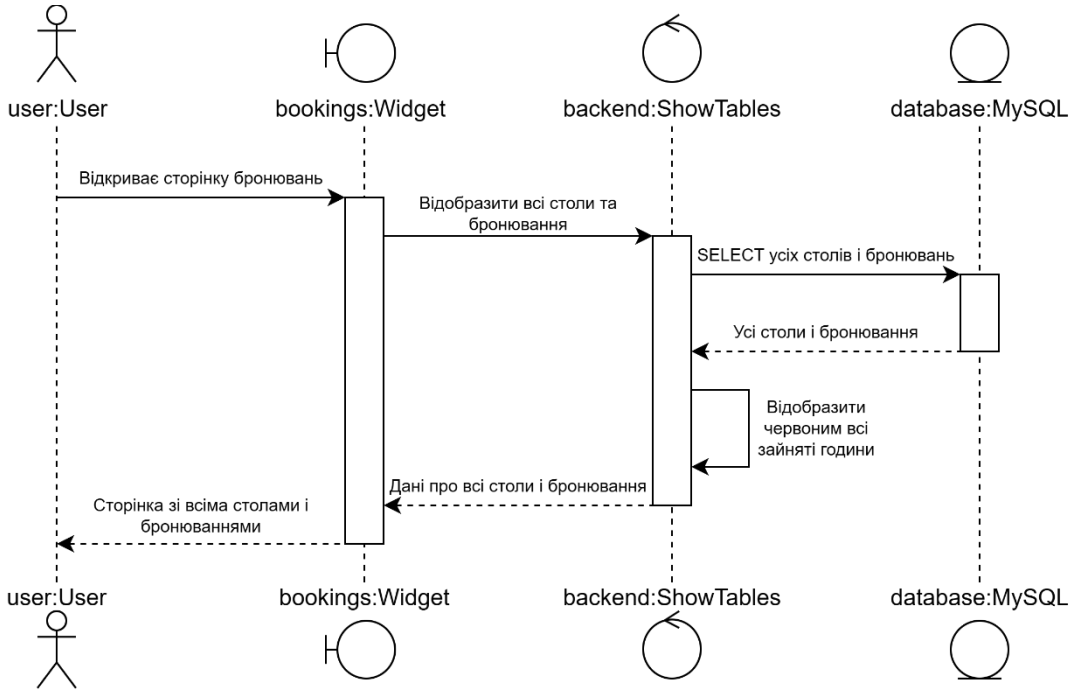

Рисунок 2.6 – Діаграма послідовності для отримання інформації з бази даних (запит “SELECT”)

Операції внесення нових даних (“INSERT”) та редагування існуючих записів (“UPDATE”) працюють за ідентичною логічною схемою. Основна відмінність полягає в тому, що на етапі обробки сервер додатково виконує алгоритми валідації та перевірки конфліктів інтервалів, які були описані в підрозділі 2.5. Після виконання запису або оновлення база даних повертає підтвердження успішної

операції, а система перенаправляє користувача на відповідну сторінку з інформаційним повідомленням.

Така уніфікована модель взаємодії компонентів забезпечує високу швидкість відгуку системи та дозволяє легко додавати нові функції, зберігаючи стабільність існуючих процесів обміну даними.

Взаємозв'язок компонентів інформаційної системи "PoolKing" побудований на принципі розділення відповідальності між клієнтською та серверною частинами. Це дозволяє створити динамічне середовище, де інтерфейс миттєво реагує на дії користувача, а вся складна логіка обробки даних виконується надійно захищеним серверним кодом.

Клієнтська частина, що працює у браузері, виступає в ролі активного інтерфейсу. Вона не лише відображає структуру сторінок, а й забезпечує збір параметрів для взаємодії з базою даних. Кожен вибір користувача – наприклад, зміна номера столу в списку – активує механізм асинхронного обміну даними. Замість того, щоб оновлювати всю сторінку повністю, система надсилає запит до конкретного програмного обробника, який відповідає за генерацію потрібного фрагмента коду (зокрема, сітки вільних годин).

Серверна частина системи виконує функцію центрального вузла обробки інформації. Вона приймає запити від інтерфейсу, проводить їхню перевірку на відповідність правилам безпеки та ролям користувачів, після чого взаємодіє з накопичувачем даних. Після отримання результату від бази даних сервер формує відповідь і передає її назад браузеру. Такий підхід дозволяє реалізувати логіку "живого" інтерфейсу: користувач бачить актуальну інформацію про завантаженість закладу в режимі реального часу.

2.4 Алгоритми автентифікації та розмежування прав доступу

Для забезпечення стабільної та безпечної роботи інформаційної системи "PoolKing" особлива увага приділяється захисту даних клієнтів. Оскільки система працює з персональною інформацією, першим і головним етапом проектування її

логіки є створення надійних алгоритмів ідентифікації користувачів та розмежування їхніх можливостей.

Одним із найважливіших елементів безпеки в системі є механізм криптографічного захисту та збереження паролів. У “PoolKing” паролі ніколи не зберігаються у чистому (читабельному) вигляді. Замість цього використовується алгоритм незворотного хешування BCrypt. Принцип його роботи полягає в тому, що пароль перетворюється на унікальний набір символів (хеш).

Цей метод обрано тому що він автоматично додає до кожного пароля випадкову “сіль”, що робить його стійким до підбору. Навіть якщо зловмисник отримає доступ до бази даних, він не зможе відновити реальні паролі. Під час входу користувача система не розшифровує пароль з бази, а просто заново хешує введені дані та порівнює отриманий результат із тим, що вже збережений.

Процес створення нового облікового запису розпочинається із заповнення форми, де вказується ім'я, адреса електронної пошти та пароль. Система спочатку перевіряє пошту на унікальність: якщо такий користувач уже є, реєстрація відхиляється. Тільки після успішної перевірки та описаного вище хешування пароля дані потрапляють у базу.

Вхід у систему (автентифікація) працює через порівняння введених даних зі списком зареєстрованих людей. Якщо логін і пароль вірні, система створює спеціальний сеанс (сесію). Це дозволяє програмі пам'ятати дані користувача, поки він переходить між сторінками, щоб йому не доводилося щоразу вводити свої дані заново. Процес реєстрації та автентифікації (блок-схему) детально показано на рисунку 2.6. Також на блок-схемі (рис. 2.6) наведено процес виходу з профілю, при якому система перевіряє чи браузер користувача зберігає дані про вхід (cookies), якщо так – видаляє, та завершує сесію. Після чого користувача перекидає на головну сторінку.

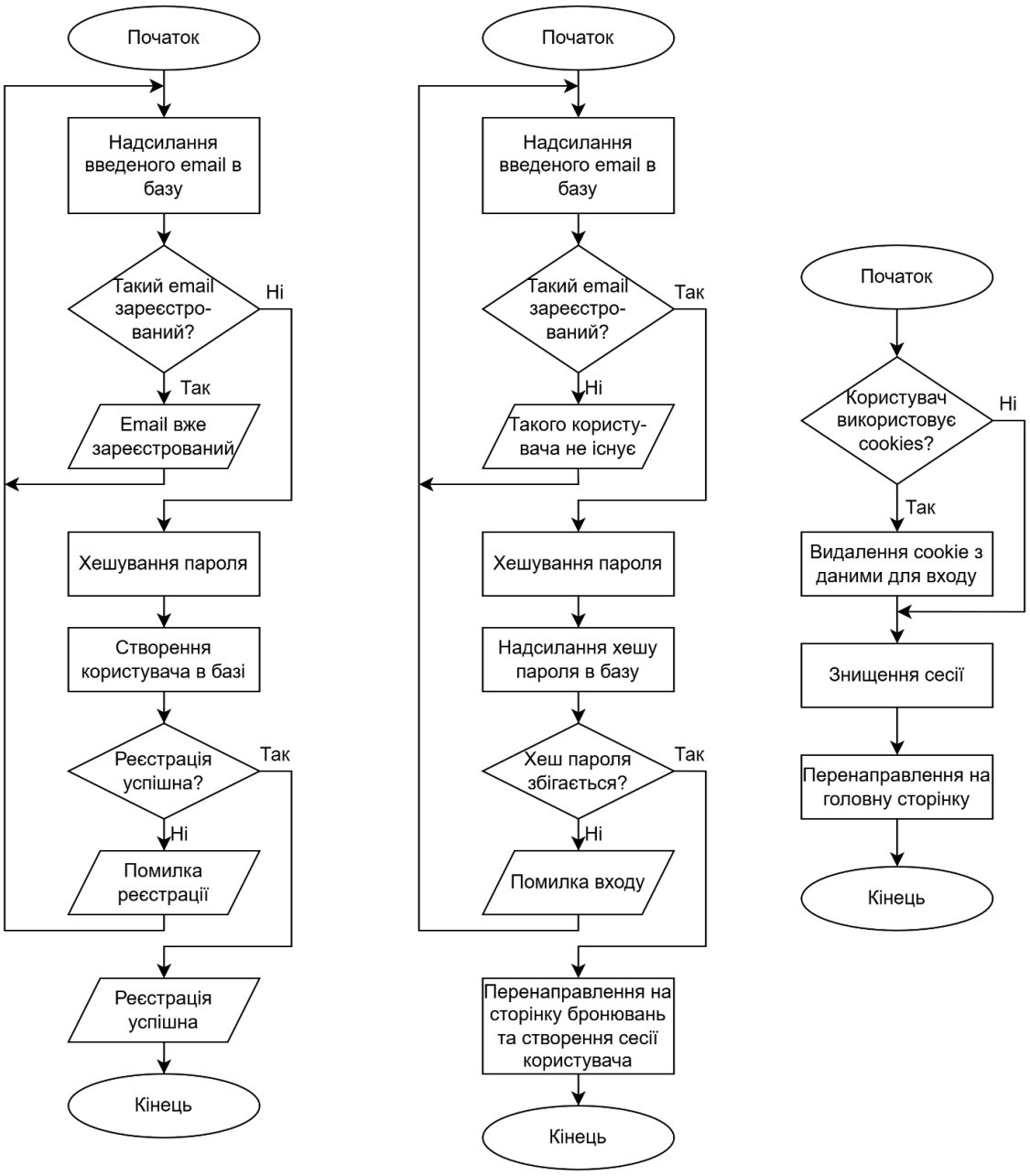

Рисунок 2.6 – Блок-схема процесу реєстрації а), автентифікації б) та виходу з профілю в)

Оскільки в системі передбачено різні типи користувачів (клієнти та менеджери), необхідно постійно контролювати, хто намагається отримати до-ступ до тих чи інших функцій. Для цього кожному профілю в базі даних при-своєна певна роль. Процес перевірки прав (блок-схему) показано на рисунку 2.7.

Рисунок 2.7 – Блок-схема процесу перевірки прав

Алгоритм перевірки прав (рис. 2.7) працює за таким принципом: щоразу, коли хтось хоче відкрити панель керування чи особистий кабінет, програма автоматично перевіряє роль у поточному сеансі (якщо такий є). Якщо звичайний клієнт спробує вручну ввести адресу сторінки для адміністрації, система миттєво відреагує і перенаправить його назад на головну сторінку. Це дозволяє надійно розділити функції управління закладом і функції бронювання столів.

Застосування цих алгоритмів робить інформаційну систему “PoolKing” стійкою до помилок та стороннього втручання. Клієнти можуть бути впевнені в безпеці своїх даних, а менеджер отримує закритий інструмент для керування клубом, недоступний для сторонніх осіб.

2.5 Алгоритм перевірки доступності столів та конфліктів інтервалів

Центральною частиною логіки системи “PoolKing” є алгоритм перевірки доступності столів. Його головне завдання – зробити так, щоб жоден стіл не був заброньований двома різними клієнтами на один і той самий час. У звичайних умовах, коли адміністратор записує клієнтів у журнал, завжди існує ризик помилитися, не помітити запис або неправильно порахувати години. Автоматизований алгоритм повністю вирішує цю проблему, перетворюючи процес перевірки на чітку математичну операцію.

Для того, щоб алгоритм міг почати роботу, інформаційна система збирає кілька важливих параметрів. Коли користувач хоче забронювати стіл, програма отримує такі дані:

– Номер столу – ідентифікатор конкретного ресурсу, який вибрав клієнт.

– Дата – конкретний день, на який планується візит.

– Час початку – точна година, коли клієнт хоче розпочати гру.

– Тривалість – кількість годин, яку користувач планує провести в закладі.

Отримавши ці дані, алгоритм має провести “моментальну ревізію” бази даних. Він перевіряє всі існуючі записи на цей стіл у вибраний день і порівнює їх із новим запитом. Тільки після того, як програма переконається, що жодна хвилина нового замовлення не накладається на вже існуючі, система дозволяє завершити процедуру бронювання. Такий підхід гарантує 100% точність обліку часу і дозволяє адміністрації закладу уникнути конфліктних ситуацій між відвідувачами.

Перш ніж перевіряти, чи вільний стіл, інформаційна система має точно визначити межі майбутнього бронювання. Оскільки користувач вказує лише час початку та кількість годин, програма повинна самостійно розрахувати точний час завершення гри. Це необхідно для того, щоб алгоритм міг чітко бачити весь часовий проміжок, який планує зайняти клієнт.

Логіка розрахунку працює за таким принципом: система бере “Час початку” (наприклад, 14:00) і додає до нього обрану “Тривалість” (наприклад, 2 години). У результаті програма отримує нове значення – “Час завершення” (16:00). Саме цей розрахований параметр стає ключовим для подальшого порівняння з іншими записами в базі даних.

Важливим моментом на цьому етапі є приведення всіх даних до єдиного стандарту. Щоб база даних могла коректно порівнювати час, програма автоматично форматує всі значення. Наприклад, якщо клієнт просто вибрав годину, система доповнює її хвилинами та секундами, перетворюючи на чіткий формат (наприклад, “14:00:00”). Такий підхід дозволяє уникнути помилок при розрахунках і гарантує, що система буде порівнювати ідентичні типи даних. Після того, як “коридор” часу (від початку до кінця) визначено, програма готова до найвідповідальнішого кроку – пошуку конфліктів із уже існуючими замовленнями.

Найскладнішим і найвідповідальнішим етапом роботи алгоритму є безпосереднє виявлення конфліктів між часовими інтервалами. Коли система вже знає планований час початку та кінця нової гри, вона повинна миттєво порівняти цей проміжок з усіма записами, які вже є в базі даних для конкретного столу.

Для цього використовується спеціальна математична модель перетину інтервалів. Головна складність полягає в тому, що нове бронювання може накладатися на старе різними способами: воно може починатися раніше і закінчуватися всередині існуючого проміжку, може починатися всередині і виходити за його межі, або взагалі повністю “поглинати” старий запис. Щоб врахувати абсолютно всі ці випадки однією дією, в інформаційній системі застосовується універсальна логічна умова.

Суть цієї моделі полягає у перевірці двох одночасних умов:

– “Новий початок” менший за “Існуючий кінець” – це означає, що замовлення клієнта починається до того, як звільниться стіл після попереднього гравця.

– “Новий кінець” більший за “Існуючий початок” – це підтверджує, що замовлення клієнта закінчується вже після того, як стіл був зайнятий кимось іншим.

Якщо обидві ці умови справджуються одночасно, система робить висновок: інтервали перетинаються. У такому разі алгоритм зупиняє процес і видає повідомлення про помилку, не дозволяючи створити дублюючий запис. Якщо ж хоча б одна з умов не виконується, це означає, що часові проміжки не мають спільних хвилин і стіл вільний.

Така логіка є максимально надійною, оскільки вона виключає можливість “пропустити” конфлікт навіть на одну хвилину. Використання цієї математичної моделі дозволяє системі “PoolKing” працювати безпомилково, забезпечуючи чесну черговість відвідувачів і стабільну роботу більярдного клубу.

Окрім технічної перевірки на перетин інтервалів, алгоритм інформаційної системи враховує адміністративні правила та обмеження, встановлені більярдним клубом. Це необхідно для того, щоб робота програми повністю відповідала реальному графіку закладу та внутрішнім розпорядкам адміністрації.

Першим важливим обмеженням є робочий час. Клуб працює за чітким розкладом – з 10:00 до 22:00. Алгоритм системи налаштований таким чином, що будь-яка спроба обрати час поза цими межами автоматично відхиляється. Перевірка відбувається у два етапи: спочатку система контролює час початку гри, а потім – розрахований час її завершення. Наприклад, якщо клієнт намагається забронювати стіл о 21:00 на три години, програма вирахує, що гра закінчиться о 00:00. Оскільки цей час виходить за межі робочого дня закладу, алгоритм видасть повідомлення про помилку і не дозволить створити запис.

Другим обмеженням є мінімальна дата бронювання. Для забезпечення стабільного планування роботи персоналу в системі реалізовано правило “бронювання мінімум на завтра”. Це означає, що алгоритм постійно порівнює поточну дату із тою, яку обирає клієнт. Якщо користувач намагається створити запис на “сьогодні”, система блокує таку можливість. Такий підхід дозволяє адміністрації клубу заздалегідь знати графік завантаженості на наступний

день і уникати хаосу з миттєвими онлайн-замовленнями, які можуть співпасти з візитами відвідувачів “з вулиці”.

Також алгоритм перевіряє логічність тривалості гри. Оскільки мінімальний час оренди зазвичай становить одну годину, програма не приймає некоректні або нульові значення. Усі ці перевірки виконуються ще до того, як система почне шукати конфлікти в базі даних. Такий багаторівневий контроль дозволяє відсіяти помилкові запити на ранньому етапі, що значно підвищує надійність системи та дисциплінує клієнтів закладу.

Щоб зібрати всі перевірки в єдиний процес, було розроблено блок-схему, зображену на рисунку 2.8.

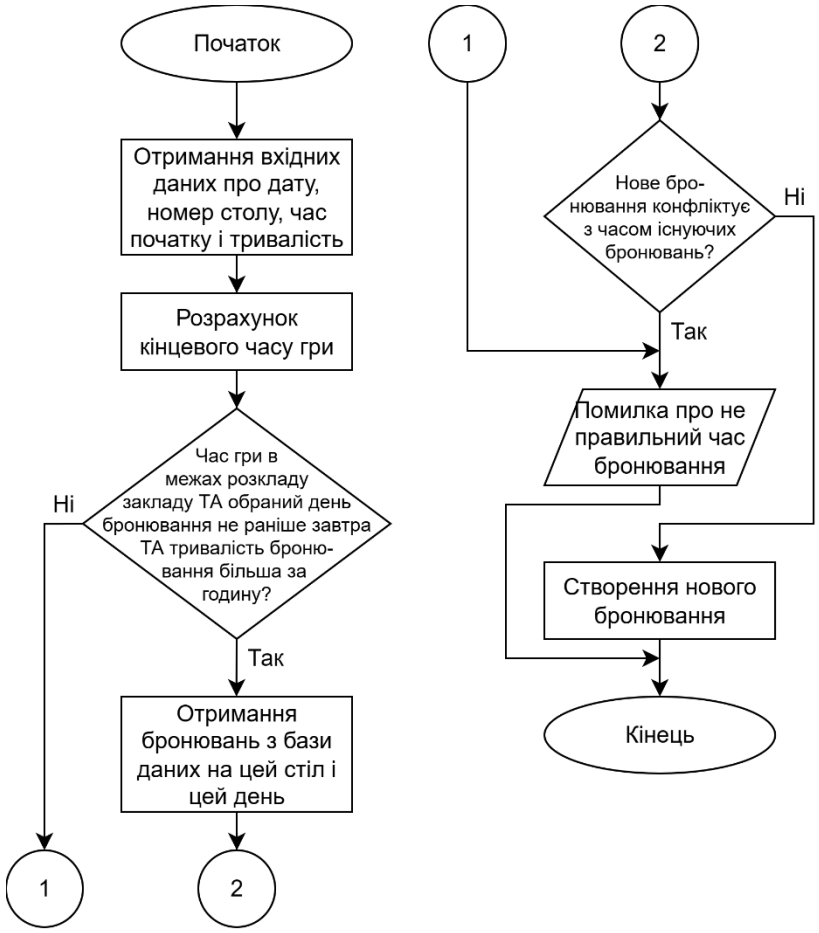

Рисунок 2.8 – Блок-схема алгоритму перевірки доступності столів та конфліктів часових інтервалів

Алгоритм працює як послідовний фільтр. Спочатку програма відсікає замовлення на минулі дати та неробочі години закладу. Після цього вона проводить головну перевірку – шукає в базі даних будь-які накладання за часом. Якщо шлях “чистий”, система дозволяє зберегти бронь.

Окремою особливістю алгоритму є його інтерактивність. Щоб користувач не помилився ще до натискання кнопки “Забронювати”, система в реальному часі аналізує сітку годин. Як тільки людина обирає тривалість (наприклад, 3 години), алгоритм миттєво підсвічує потрібну кількість клітинок жовтим кольором. Це дозволяє клієнту візуально оцінити свій вибір, а системі – заздалегідь приховати недоступні варіанти тривалості, якщо попереду є чуже бронювання.

2.6 Обґрунтування вибору технологій: HTML, PHP, MySQL та Apache

Вибір технологічного стеку для інформаційної системи “PoolKing” базується на необхідності створення доступного, швидкого та надійного сервісу. На етапі планування було вирішено розробляти саме веб-орієнтований додаток, а не звичайну програму для комп'ютера. Головна перевага такого підходу – кросплатформеність. Це означає, що клієнту не потрібно нічого завантажувати чи встановлювати: система однаково стабільно працює через будь-який сучасний браузер.

В основі функціонування системи лежить класична клієнт-серверна архітектура. Вона передбачає чіткий розподіл обов'язків: серверна частина відповідає за безпеку, збереження даних та виконання складних розрахунків, а клієнтська – за відображення інтерфейсу та взаємодію з користувачем. Такий поділ дозволяє централізовано зберігати всю інформацію про бронювання в одному місці, що виключає розсинхронізацію даних. Крім того, використання веб-технологій дозволяє власнику закладу легко масштабувати систему та вносити зміни в її роботу без необхідності оновлювати програмне забезпечення на пристроях кожного окремого клієнта.

Для побудови зовнішнього вигляду інформаційної системи обрано мову розмітки HTML5 та мову стилів CSS3.

H T M L 5 виступає як «каркас» проекту. Його використано для створення структури сторінок: логічних блоків, списків більярдних столів, кнопок та форм, через які користувач вводить дані. Сучасні стандарти HTML5 дозволяють системі коректно працювати з різними типами вводу (наприклад, вибір дати або часу) без залучення сторонніх інструментів, що робить код легшим і швидшим.

C S S 3 відповідає за оформлення та зручність інтерфейсу. Для системи “PoolKing” було обрано сучасний стиль “Glassmorphism” (ефект скла), який реалізується через напівпрозорі блоки та розмиття фону. Це дозволяє створити атмосферу затишного більярдного залу. Окрім естетики, CSS3 використовується для візуального відгуку: наприклад, зміна кольору клітинки при наведенні мишки або плавна поява панелі підтвердження. Це допомагає клієнту краще орієнтуватися на сторінці та уникати помилок при виборі часу. Головною перевагою такого вибору є те, що HTML та CSS є світовими стандартами, тому сайт буде виглядати однаково професійно у будь-якому браузері.

Для реалізації внутрішньої логіки системи, обробки запитів та зв'язку з базою даних було обрано мову програмування PHP.

Вибір PHP для проекту “PoolKing” обумовлений декількома практичними перевагами:

– С п е ц і а л і з а ц і я н а в е б р о з р о б ц і : PHP була створена саме для роботи в інтернеті. Вона легко обробляє дані з форм реєстрації та бронювання, працює з сесіями користувачів і миттєво генерує потрібний результат для відправки у браузер.

– Б е з п е к а р о б о т и з д а н и м и : завдяки вбудованому інтерфейсу PDO (PHP Data Objects), мова дозволяє створювати захищені запити до бази даних. Це критично важливо для нашої системи, адже саме на рівні PHP може бути реалізовано захист від SQL-ін’єкцій та надійне хешування паролів.

– В и с о к а ш в и д к і с т ь т а л е г к і с т ь : PHP-скрипти виконуються дуже швидко, що забезпечує моментальну перевірку вільного часу на столі. Крім

того, ця мова є повністю безкоштовною і відкритою, що дозволяє значно знизити витрати на розробку та підтримку системи.

– Сумісність: PHP стабільно працює з усіма сучасними базами даних та вебсерверами, що робить систему “PoolKing” універсальною – її можна легко перенести з локального комп'ютера на будь-який реальний хостинг у майбутньому.

Таким чином, PHP виступає “мозком” інформаційної системи, який відповідає за те, щоб усі математичні розрахунки та правила безпеки виконувалися чітко і без помилок.

Для надійного зберігання всієї інформації в системі “PoolKing” обрано систему керування базами даних MySQL. Оскільки наш проект базується на чітких зв'язках між клієнтами, столами та часом, MySQL є ідеальним рішенням завдяки своїй реляційній структурі.

Основні причини вибору саме цієї бази даних:

1. Надійність та цілісність даних: у системі бронювання критично важливо, щоб жоден запис не загубився. MySQL дозволяє налаштувати жорсткі зв'язки між таблицями.

2. Висока швидкість роботи: коли користувач відкриває сітку годин, база даних має миттєво опрацювати сотні записів, щоб знайти вільні вікна. MySQL оптимізована для швидкого виконання таких запитів, тому клієнт отримує відповідь від сайту без затримок.

3. Безпека та доступність: MySQL пропонує потужні інструменти для захисту інформації від несанкціонованого доступу. Крім того, вона є безкоштовною та відкритою, що дозволяє використовувати професійне рішення для збереження даних без додаткових витрат.

4. Ідеальна пара з PHP: ця база даних найчастіше використовується разом із мовою PHP. Це гарантує стабільність роботи системи, відсутність конфліктів у коді та легкість у налаштуванні сервера.

Отже, MySQL виконує роль цифрового “архіву” клубу, де вся інформація розкладена по поличках, захищена від сторонніх і доступна для миттєвого пошуку в будь-який момент.

Для роботи інформаційної системи використовується вебсервер Apache. Він виконує роль надійного посередника, який приймає запити від браузерів користувачів, передає їх серверному коду для обробки та повертає готовий результат назад на екран. Apache обрано через його високу стабільність, безпеку та ідеальну сумісність із мовою PHP, що гарантує швидку обробку кожного кліка користувача під час вибору столу чи часу.

Усі етапи розробки та тестування проекту проходили в середовищі XAMPP. Це спеціальний програмний пакет, який дозволяє запустити повноцінний сервер прямо на звичайному комп'ютері. Вибір XAMPP обумовлений тим, що він об’єднує всі необхідні інструменти (Apache, MySQL та PHP) в одну систему. Це дало змогу ретельно перевірити логіку бронювання та роботу бази даних у закритому середовищі, гарантуючи, що після переносу на реальний хостинг сайт працюватиме без помилок.

2.7 Проектування архітектури програмного комплексу

Для забезпечення зручного керування кодом та швидкої роботи системи “PoolKing” була розроблена чітка ієрархічна структура файлів. Весь проект розділений на логічні блоки, що дозволяє відокремити зовнішній вигляд сайту від його внутрішньої логіки та налаштувань безпеки. Такий підхід спрощує пошук потрібних компонентів та робить систему надійною у використанні.

Основними директоріями інформаційної системи є:

- “a s s e t s ” – містить усі статичні ресурси: файли стилів для дизайну, іконки та зображення, а також скрипти для забезпечення інтерактивності (підсвітка годин, AJAX-запити).

- “c o n f i g ” – технічний вузол, де зберігаються параметри підключення до бази даних.

- “i n c l u d e s ” – папка з шаблонами, які повторюються на всіх сторінках (шапка та підвал сайту), а також загальні допоміжні функції.

- “a u t h ” – закритий блок, відповідальний за реєстрацію, вхід та безпечне завершення сеансу користувача.

- “u s e r ” та “m a n a g e r ” – функціональні розділи, що містять інстру-менти для клієнтів (бронювання) та адміністрації (керування закладом).

Детальний опис кожного файлу системи, його розміщення та роль у загальній роботі програмного комплексу наведено в таблиці 2.5:

Таблиця 2.5 – Структура та опис файлів інформаційної системи “PoolKing”

Назва файлу	Розміщення	Опис та призначення
1	2	3
.htaccess	Корінь проекту	Конфігурація сервера для створення семантичних URL
index.php	Корінь проекту	Головний роутер: приймає всі запити та підключає потрібні модулі
home.php	Корінь проекту	Вміст головної сторінки з описом клубу та кнопками дій
404.php	Корінь проекту	Сторінка помилки, якщо користувач ввів неправильну адресу
db.php	config/	Налаштування зв'язку з базою даних через інтерфейс PDO
header.php	includes/	Верхня частина сайту: логотип, меню навігації та статус сесії
footer.php	includes/	Нижня частина сайту: контакти, соцмережі та підключення JS
functions.php	includes/	Загальні інструменти: перевірка ролей та очищення вхідних даних
handler.php	auth/	Головний файл автентифікації: обробляє дані реєстрації та входу
login.php	auth/	Форма входу в особистий кабінет
register.php	auth/	Форма створення нового акаунту клієнта
logout.php	auth/	Скрипт для безпечного виходу та знищення сесії
booking.php	user/	Інтерактивна сторінка вибору дати, столу та годин гри
profile.php	user/	Кабінет клієнта: список його бронювань та кнопки керування
edit-booking.php	user/	Сторінка зміни параметрів існуючого замовлення
process.php	user/	Головний обробник: перевіряє вільний час та надсилає дані в базу
ajax-edit-slots.php	user/	Технічний файл для підвантаження годин без оновлення сторінки
dashboard.php	manager/	Панель менеджера: вкладки всіх столів та моніторинг клієнтів
actions.php	manager/	Обробник дій адміністратора: видалення бронювань та відправка Email

Кінець таблиці 2.5

1	2	3
style.css	assets/	Всі візуальні налаштування сайту та ефекти "скляного" дизайну
main.js	assets/	Логіка інтерфейсу: підсвітка клітинок сітки та AJAX-запити

Така організація компонентів дозволяє системі працювати швидко і злагоджено. Кожен файл виконує свою вузьку задачу, що мінімізує ризик виникнення конфліктів у коді та полегшує подальше розширення функцій інформаційної системи більярдного клубу.

Висновок до розділу

Таким чином, розроблена архітектура бази даних та детально описані алгоритми перевірки доступності ресурсів створюють надійну основу для технічної реалізації системи. Спроектована рольова модель доступу та обґрунтований вибір технологічного стеку гарантують стабільність і безпеку функціонування програмного комплексу. Ці результати дозволяють перейти до етапу практичного написання коду та налаштування інтерфейсів інформаційної системи.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ МОДУЛІВ ТА ІНТЕРФЕЙСУ

3.1 Програмна реалізація серверної логіки та взаємодії з БД через PDO

Практична частина розробки інформаційної системи “PoolKing” розпочалася із перенесення спроектованої моделі даних у фізичне середовище. Першим етапом стало створення бази даних та опис структури її таблиць мовою SQL. Це дозволило підготувати середовище для збереження інформації про користувачів, більярдні столи та кожну заброньовану годину.

Нижче наведено фрагмент SQL-запиту, який використовувався для створення таблиці користувачів (повний запит створення бази даних наведено в додатку А):

```
CREATE TABLE `users` (
  `id` int NOT NULL AUTO_INCREMENT,
  `full_name` varchar(100) NOT NULL,
  `email` varchar(100) NOT NULL,
  `password` varchar(255) NOT NULL,
  `role` enum('user','manager') NOT NULL DEFAULT 'user',
  PRIMARY KEY (`id`),
  UNIQUE KEY `email` (`email`)
) ENGINE=InnoDB AUTO_INCREMENT=7 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;
```

У цьому коді визначено типи даних для кожного поля, встановлено первинний ключ та додано обмеження унікальності для електронної пошти, що запобігає повторній реєстрації.

Після створення структури бази даних необхідно було реалізувати механізм, за допомогою якого програма буде звертатися до цих даних. Для цього було створено окремий конфігураційний файл db.php. Його головне завдання – встановити стабільне з’єднання із сервером бази даних та налаштувати правила обміну інформацією. Використання окремого файлу для підключення дозволяє легко змінювати налаштування доступу в одному місці, що підвищує зручність підтримки всієї системи.

Для взаємодії з базою даних у проекті використано об'єктно-орієнтований інтерфейс PDO (PHP Data Objects). Головна перевага цього підходу полягає у використанні “підготовлених запитів”. Це означає, що замість прямої вставки даних у текст команди, програма спочатку надсилає шаблон запиту, а потім окремо передає дані як параметри. Така логіка повністю блокує можливість зламу бази через поля вводу (SQL-ін'єкції).

Нижче представлено код, який відповідає за ініціалізацію підключення (Повний код наведено в додатку Б). У ньому створюється об'єкт класу PDO, вказуються параметри сервера та задаються налаштування обробки помилок, щоб у разі збою система видавала технічне повідомлення, а не розкривала структуру коду.

```
$options = [
   PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION,
   PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC,
   PDO::ATTR_EMULATE_PREPARES => false
   ];
$pdo = new PDO($dsn, $user, $pass, $options);
```

Крім підключення, на рівні серверної логіки реалізовано безпечне виконання всіх операцій із даними: пошук вільних столів, додавання нових записів та оновлення інформації. Кожен такий запит проходить через етапи prepare (підготовка) та execute (виконання), що гарантує цілісність інформаційного простору системи.

Нижче наведено приклад коду (файл process.php), що демонструє, як система додає нове бронювання в базу даних (Повний код наведено в додатку В):

```
      $sql = "INSERT INTO bookings (user_id, table_id, booking_date, start_time,
end_time)VALUES (?, ?, ?, ?, ?)";
      $stmt = $pdo->prepare($sql);
      $result = $stmt->execute([$user_id, $table_id, $booking_date, $start_time,
$end_time]);
```

На цьому фрагменті видно, що система спочатку підготовує запит INSERT, а потім вставляє в нього необхідні дані і виконує. Результат цього запиту записується в змінну $result.

Така програмна реалізація взаємодії з базою даних забезпечує високу швидкість обробки замовлень у режимі реального часу та створює надійний фундамент для роботи всіх функціональних модулів інформаційної системи.

3.2 Розробка інтерфейсу користувача із застосуванням стилю Glassmorphism

Візуальна частина інформаційної системи “PoolKing” реалізована за концепцією “Modern Dark Luxury”, що відповідає естетиці більярдного закладу. Основний акцент зроблено на стилі Glassmorphism, який базується на поєднанні напівпрозорих елементів із розмиттям фону.

Для побудови такого інтерфейсу використано мову розмітки сторінки (HTML) та каскадні таблиці стилів (CSS). За допомогою мови HTML було створено всі основні елементи інтерфейсу, такі як кнопки, поля вводу, блоки та текст. Нижче розглянемо фрагмент коду головної сторінки (home.php), розробленої за допомогою HTML із застосуванням класів (повний код наведено в додатку Г):

```
<section id="about" class="features-grid">
    <div class="feature-card">
        <img src="./assets/img/home_pool.jpg" alt="Пул">
        <h3>Американський пул</h3>
        <p>Класичний пул — це динамічна гра, де важлива точність кожного удару і вміння прораховувати ситуацію на кілька ходів вперед.</p>
    </div>
```

У цьому фрагменті коду наведено створення інформаційної картки про американський пул. Також можна побачити використання класів – секція має клас “features-grid”, а її дочірній елемент – “feature-card”. Стилі для цих обох класів описані в файлі CSS. Розглянемо фрагмент коду зі стилем для інфомарційної картки (“feature-grid”):

```
.feature-card {
    padding: 10px;
    transition: 0.3s;
}
.feature-card:hover {
    background-color: var(--bg-dark);
    border-radius: 10px;
    transform: translateY(-3px);
}
```

Можемо також побачити застосування властивості “:hover”, яка дозволяє застосувати окремі стилі для блоків при наведенні курсором.

Ключовим технічним рішенням сайту є застосування властивості CSS “backdrop-filter: blur()”, що створює ефект матового скла для всіх основних інформаційних блоків: форм авторизації, карток столів та панелей керування. Фон сайту реалізований у вигляді триколірного лінійного градієнта, який створює глибину простору.

Усі кольорові параметри винесені у CSS-змінні (“:root”), що дозволяє підтримувати єдину стилістику у всьому проекті. Фірмовим кольором обрано “emerald green” (“#2ecc71”) – колір більярдного сукна, який використовується для кнопок, стрілок та іншого. Додатковий золотистий колір (“#f1c40f”) слугує для акцентування уваги на вибраних користувачем часових проміжках. Фрагмент коду з кольорами в CSS-змінних та стилями інформаційних блоків (Glassmorphism) наведено нижче:

```
:root {
    --primary-green: #2ecc71;
    --dark-green: #27ae60;
    --accent-gold: #f1c40f;
    --danger-red: #e74c3c;
    --bg-dark: #0f2027;
    --glass-bg: rgba(255, 255, 255, 0.07);
    --glass-border: rgba(255, 255, 255, 0.1);
    --text-main: #ffffff;
    --text-muted: #bdc3c7;}
.glass-card, .auth-form, .table-card, .date-selector, .manager-container,
.profile-container, .tab-content {
    background: var(--glass-bg) !important;
    backdrop-filter: blur(12px);
    -webkit-backdrop-filter: blur(12px);
    border: 1px solid var(--glass-border) !important;
    border-radius: 15px;
    box-shadow: 0 8px 32px 0 rgba(0, 0, 0, 0.3);
    color: var(--text-main);
    text-align: center;}
```

У коді продемонстровано використання CSS-змінних для кольорів та правила “!important”, яке використовується для надання більшої важливості властивості, ніж зазвичай.

Головна сторінка системи спроектована як лендинг, що містить секцію “Hero” з атмосферним фото та блоки з технічним описом типів гри (пул, снукер, піраміда). Зовнішній вигляд головної сторінки представлено на рисунку 3.1.

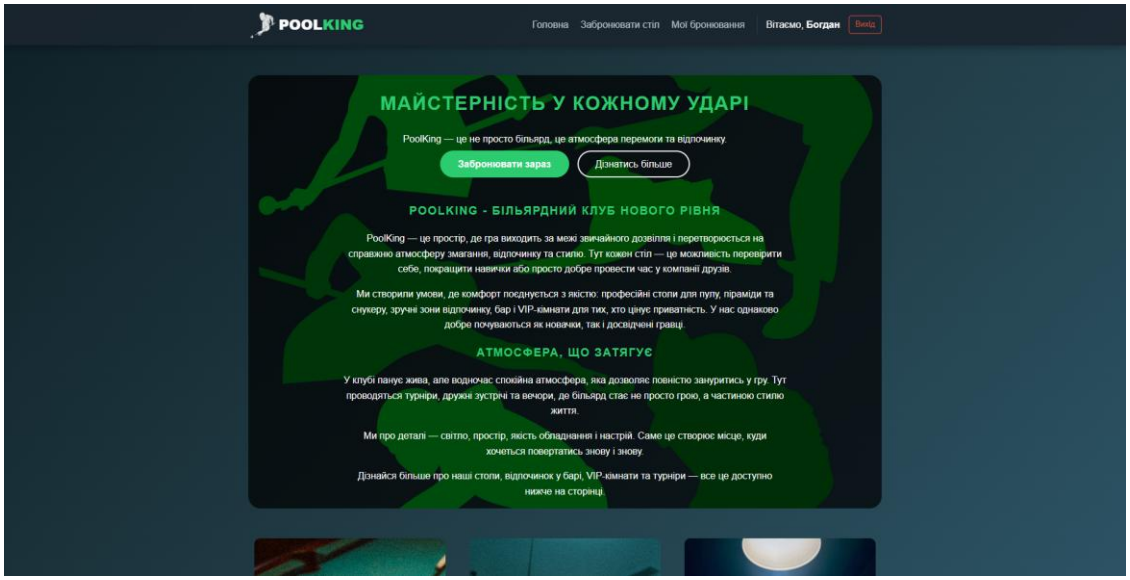

Рисунок 3.1 – Головна сторінка сайту “PoolKing”

Для сторінок реєстрації та входу реалізовано центровану макетну сітку. Поля вводу мають напівпрозорий фон та інтерактивні властивості: при наведенні або фокусуванні рамка змінює колір за допомогою плавного переходу (“transition”), що підвищує якість користувацького досвіду. Вигляд форми авторизації наведено на рисунку 3.2.

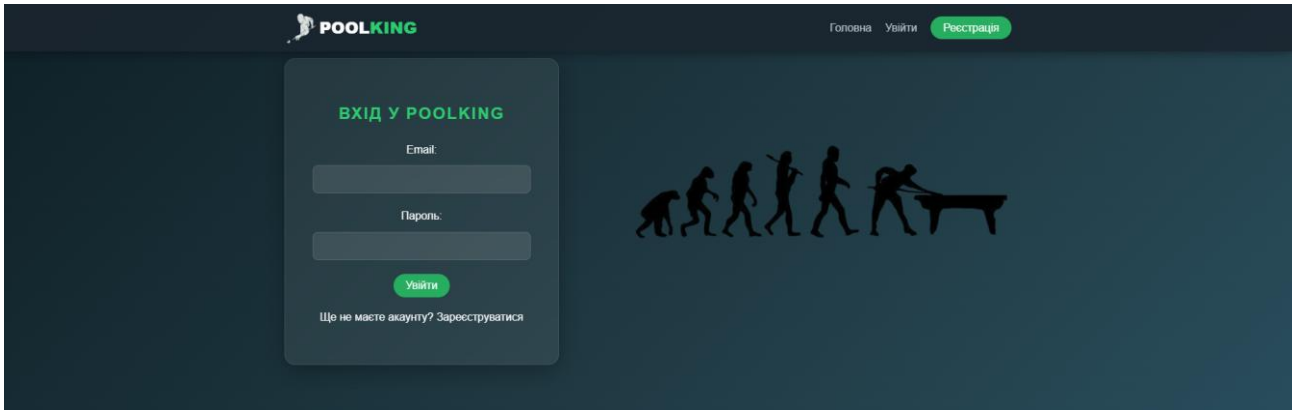

Рисунок 3.2 – Сторінка авторизації

Код сторінок авторизації, реєстрації файлу керування ролями та файлу завершення сесії, наведено в додатках Д.1, Д.2, Д.3 та Д.4 відповідно.

Верхня панель навігації (хедер) закріплена у верхній частині екрана за допомогою властивості “position: sticky”. Вона містить динамічне меню, яке змінюється залежно від ролі користувача: менеджер бачить посилання на адмін-панель, а клієнт – на профіль та бронювання. У нижній частині сторінок (футер) розміщено логотип клубу, контактні дані та іконки соціальних мереж. Повний код реалізації хедера та футера наведено в додатках Е.1 та Е.2 відповідно.

Така програмна реалізація інтерфейсу забезпечує не лише привабливий вигляд, а й чітку ієрархію інформації, дозволяючи користувачеві швидко орієнтуватися у функціях системи.

3.3 Реалізація інтерактивної сітки бронювання засобами AJAX та Fetch API

Ключовою особливістю системи “PoolKing” є відмова від стандартних випадаючих списків на користь інтерактивної графічної сітки. Це дозволяє користувачеві візуально оцінити завантаженість кожного столу та обрати потрібний час одним кліком. Технічно такий функціонал реалізовано за допомогою мови JavaScript та технології асинхронних запитів Fetch API.

Процес побудови інтерактивної сітки розділений на два рівні: серверний та клієнтський. На серверному рівні створено файл ajax_edit_slots.php, який отримує ідентифікатор столу та дату, після чого формує фрагмент HTML-коду з доступними годинами. Для кожного часового слота програма перевіряє його статус у базі даних: якщо година заброньована іншим клієнтом, клітинка отримує клас “busy” (червоний колір), якщо вільна – “free” (зелений колір) (Повний код наведено в додатку Ж).

Нижче наведено фрагмент коду серверної логіки, що генерує сітку годин для конкретного ресурсу:

```
<?php for ($h = 10; $h < 22; $h++):
        $slot = sprintf('%02d:00', $h);
        $is_busy = in_array($slot, $booked_hours);
    ?>
    <div class="hour-slot <?= $is_busy ? 'busy' : 'free' ?>"
        data-time="<?= $slot ?>"
        data-table-id="<?= $table_id ?>"
        onclick="<?= $is_busy ? '' : 'selectTime(this)' ?>">
        <?= $h ?>:00
    </div>
<?php endfor; ?>
```

На клієнтському рівні логіка зосереджена у файлі main.js. Коли користувач змінює номер столу у списку, спрацьовує функція, яка відправляє фоновий запит на сервер без оновлення сторінки. Отриманий результат (нова сітка годин) миттєво вставляється у відповідний контейнер на сторінці (Повний код наведено в додатку И).

Нижче представлено частину JavaScript-коду, що відповідає за асинхронне отримання даних про вільний час через Fetch API:

```
function loadEditSlots() {
    if (!editTableSelect || !editDateInput) return;
    const tableId = editTableSelect.value;
    const date = editDateInput.value;
    fetch(`/poolking/user/ajax_edit_slots.php?table_id=${tableId}&date=${dat
e}&booking_id=${currentBookingId}`)
    .then(res => res.text())
    .then(html => {
        editContainer.innerHTML = html;
        const startInput = document.getElementById('input-start-time');
        if (startInput && startInput.value) {
            const activeSlot = editContainer.querySelector(`[data-
time="${startInput.value}"]`);
        }
    });
}
```

Окрім підвантаження даних, JavaScript керує візуальною підсвіткою обраного інтервалу. При натисканні на вільну клітинку спрацьовує механізм, який автоматично розраховує діапазон годин залежно від обраної тривалості (1, 2 або 3 години) і фарбує відповідні блоки у жовтий колір. Це дозволяє клієнту наочно бачити часовий проміжок, який він збирається зайняти.

Вигляд інтерактивної сітки бронювання під час вибору часу представлено на рисунку 3.3.

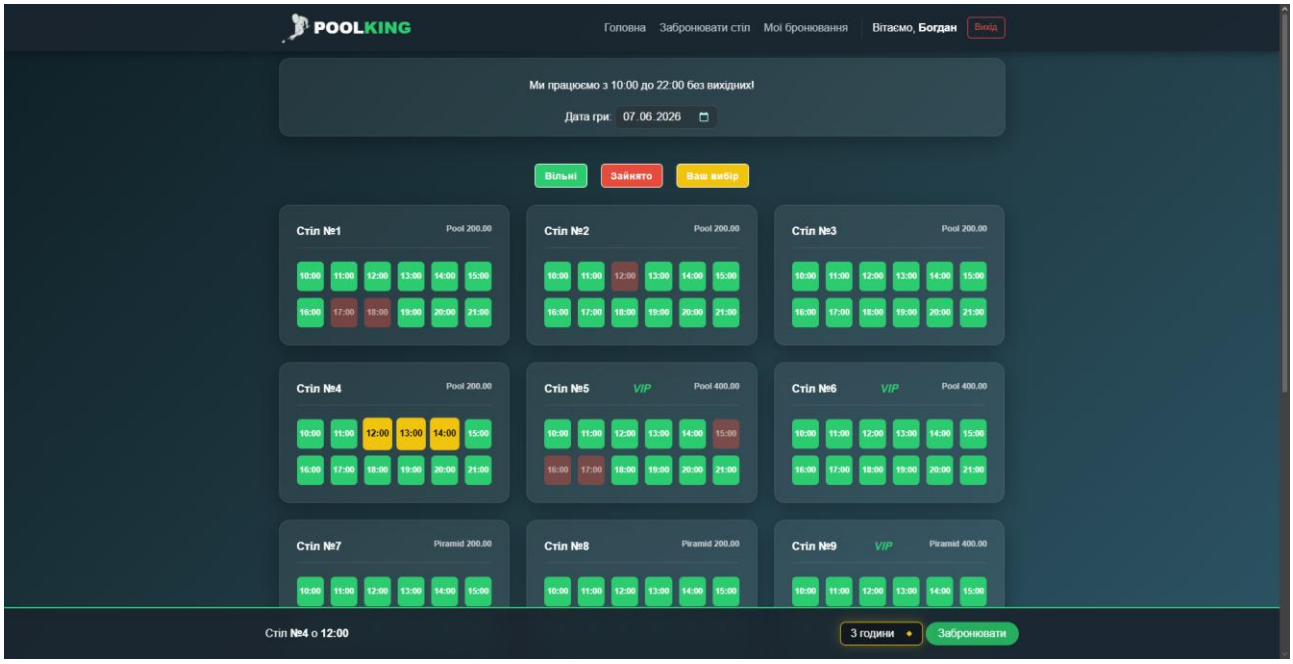

Рисунок 3.3 – Вигляд інтерактивної сітки бронювання

Код реалізації сторінки бронювання наведено в додатку К.

Важливою частиною логіки є динамічне керування списком тривалості. Якщо користувач натискає на годину, після якої через певний час стіл уже заброньований іншою людиною, система автоматично приховує варіанти "2 години" або "3 години", щоб запобігти накладанню замовлень ще на етапі вибору. Це завдання виконує функція hideOptionsFrom(), код якої можна переглянути в додатку И.

Така програмна реалізація робить процес взаємодії із системою максимально швидким та інтуїтивним, виключаючи необхідність постійних перезавантажень сторінок та мінімізуючи ймовірність помилок користувача.

3.4 Опис можливостей системи: функції клієнта та панель менеджера

Заключним етапом розробки стало створення функціональних модулів для різних категорій користувачів. Основна мета цих модулів – надати зручний інтерфейс для керування замовленнями та забезпечення оперативного моніторингу завантаженості закладу.

Перший модуль – особистий кабінет клієнта. Цей модуль реалізований у файлі profile.php (повний код наведено в додатку Л.1) та призначений для самоконтролю користувача. Після авторизації клієнт отримує доступ до таблиці зі своїми бронюваннями, де відображається номер столу, категорія більярду, дата та точний час гри. Для підтримання актуальності даних у системі передбачено логічне обмеження: кнопки редагування та скасування доступні лише для тих замовлень, час початку яких ще не настав. Зовнішній вигляд кабінету клієнта наведено на рисунку 3.4.

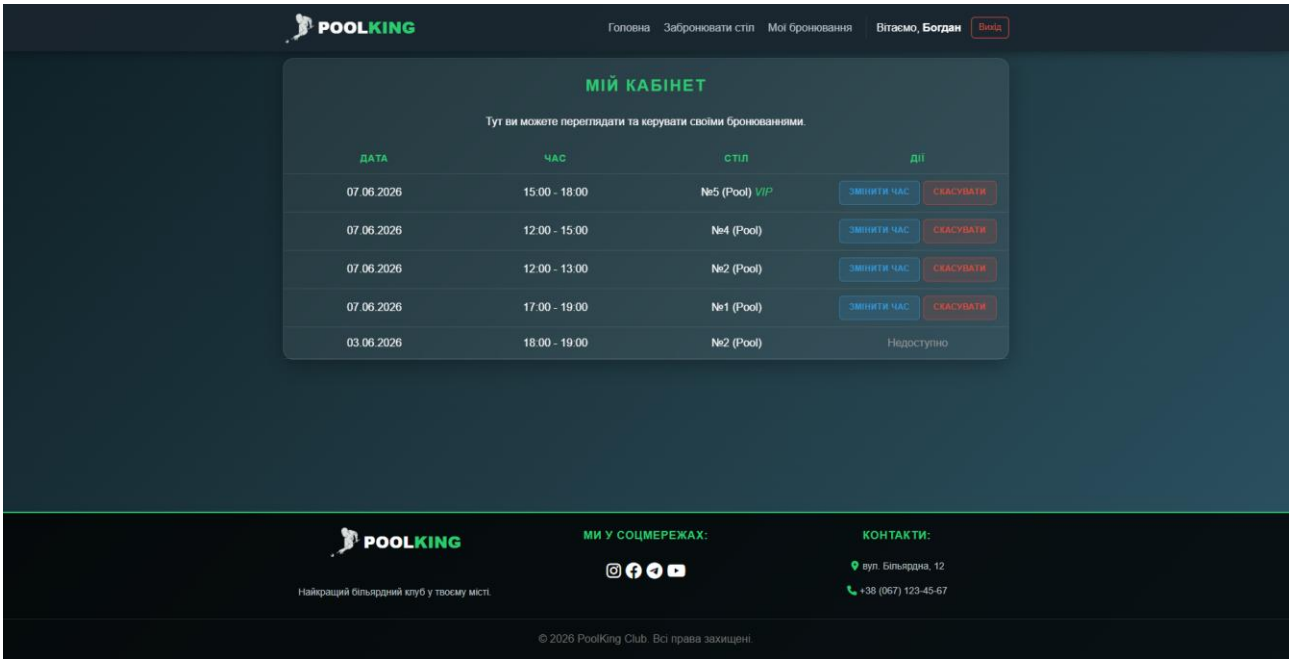

Рисунок 3.4 – Особистий кабінет клієнта

Процес скасування замовлення реалізований через обробник, який видаляє відповідний запис із бази даних. У разі потреби змінити параметри візиту, система перенаправляє користувача на сторінку редагування, де завантажуються поточні дані замовлення для їх подальшого коригування. За редагування та видалення записів відповідає файл edit_booking.php (код наведено в додатку Л.2). В цьому файлі реалізовано динамічну зміну відображення стола, вибраного в випадаючому списку. За це відповідає функція loadEditSlots(), код якої можна переглянути в додатку И. На рисунку 3.5 зображено вигляд сторінки редагування бронювання.

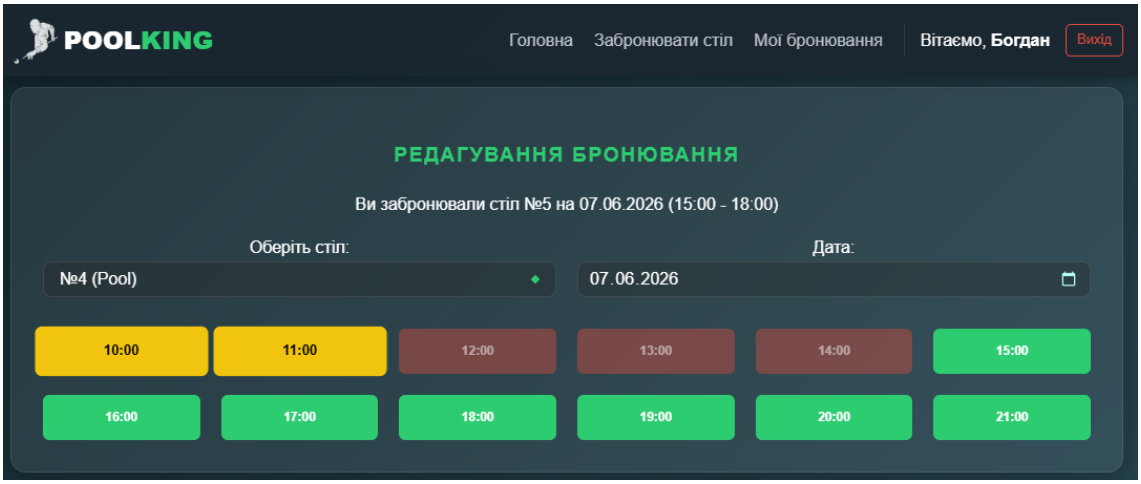

Рисунок 3.5 – Сторінка редагування бронювання

На цій сторінці також реалізовано інтерактивну сітку годин з відображенням вільних, вибраних та зайнятих годин.

Наступний модуль це панель керування менеджера. Для адміністративного персоналу розроблено розширений інтерфейс у файлі dashboard.php (код наведено в додатку М.1). Основною технічною особливістю панелі є використання вкладок для кожного з 12 столів, що дозволяє уникнути перевантаження екрана інформацією. Перемикання між столами реалізовано за допомогою JavaScript-функції openTable() (код наведено в додатку И), яка забезпечує миттєву зміну вмісту без оновлення сторінки. Візуалізацію панелі менеджера представлено на рисунку 3.6.

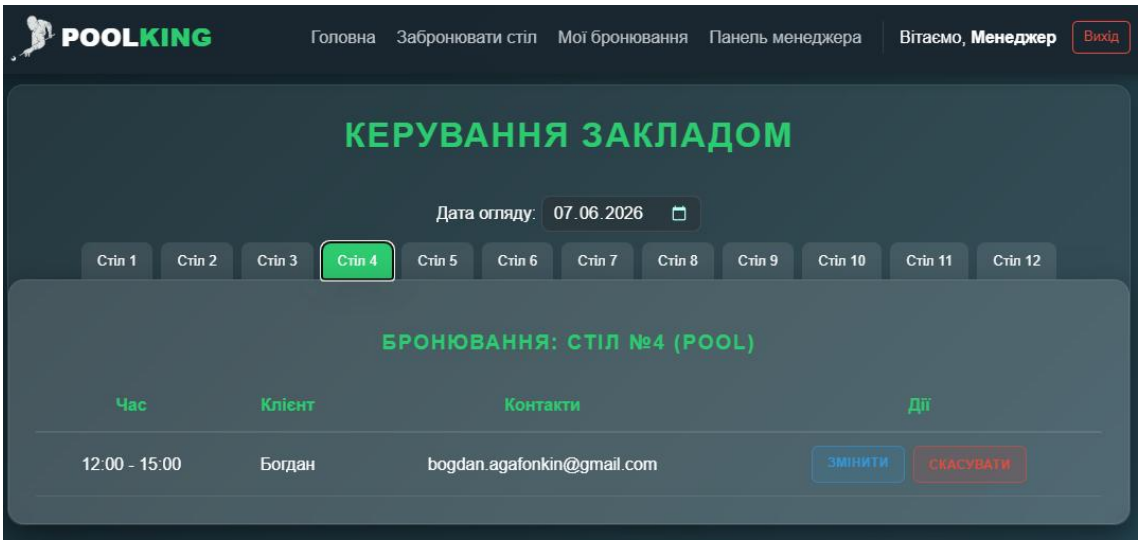

Рисунок 3.6 – Панель керування менеджера

Менеджер має можливість бачити повну інформацію про клієнтів (ім'я та контактний Email) для оперативного зв'язку. На відміну від клієнта, адміністратор може редагувати або видаляти будь-який запис у системі, що необхідно для оперативного вирішення конфліктних ситуацій у залі.

Наступний модуль – система автоматичних сповіщень. Важливою функцією модуля менеджера є зворотний зв’язок із клієнтом. У разі примусового скасування бронювання адміністратором через файл actions.php (код наведено в додатку М.2), система автоматично ініціює виконання функції sendStatusEmail(), код якої наведено нижче:

```
function sendStatusEmail($to, $subject, $message) {
    $headers = "MIME-Version: 1.0" . "\r\n";
    $headers .= "Content-type:text/html;charset=UTF-8" . "\r\n";
    $headers .= "From: PoolKing <poolking@gmail.com>" . "\r\n";
    return mail($to, $subject, $message, $headers);
}
```

Ця функція формує HTML-лист із логотипом клубу та деталями скасованого замовлення, після чого надсилає його на пошту клієнта. Ця функція знаходиться в файлі functions.php, код якого наведено в додатку Н. Приклад оформлення електронного листа наведено на рисунку 3.7.

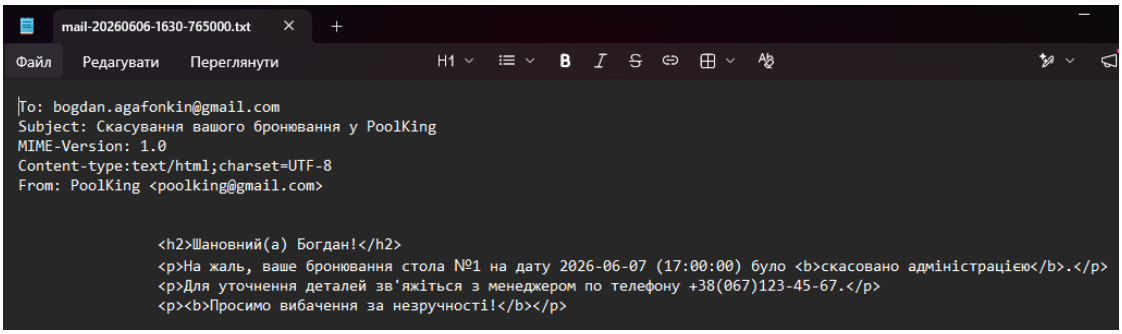

Рисунок 3.7 – Вигляд Email з повідомленням про видалення бронювання

Такий розподіл функціональних можливостей дозволяє системі “PoolKing” ефективно обслуговувати клієнтів у автоматичному режимі, одночасно надаючи адміністрації клубу повний інструментарій для контролю та управління ресурсами закладу.

3.5 Тестування можливостей системи

Перед впровадженням інформаційної системи “PoolKing” у реальну експлуатацію було проведено комплексне тестування її функціональних модулів. Процес перевірки був спрямований на виявлення можливих логічних помилок, перевірку стійкості системи до некоректних дій користувача та підтвердження надійності закладених алгоритмів безпеки. Для систематизації процесу перевірки було розроблено тест-план. Основні об'єкти та методи тестування наведено в таблиці 3.1.

Таблиця 3.1. План тестування функціональних можливостей системи “PoolKing”

ID	Об’єкт тестування	Тестовий сценарій	Очікуваний результат
1	Реєстрація користувача	Спроба створення акаунту на вже існуючу в базі пошту.	Блокування операції та виведення повідомлення про зайнятий Email.
2	Система бронювання	Спроба вибору тривалості гри, що накладається на існуючий запис.	Автоматичне приховування недоступних опцій тривалості в селекторі.
3	Валідація часу	Спроба забронювати стіл на час, що виходить за межі робочого дня (10:00–22:00).	Відмова системи та виведення помилки про неробочі години.
4	Цілісність даних	Спроба редагування чужого бронювання через зміну ID в адресному рядку.	Спроба ігнорується базою даних, оскільки проводиться перевірка власника за ID сесії.
5	Логіка дат	Ручне введення сьогоднішньої дати замість мінімально дозволеної (завтра).	Система ігнорує введення та автоматично переносить запис на найближчу доступну дату.
6	Розмежування ролей	Спроба прямого доступу звичайного клієнта до сторінок менеджера.	Спрацювання фільтра доступу та примусовий редірект на головну сторінку.
7	Маршрутизація (404)	Введення неіснуючої назви сторінки або шляху в адресному рядку браузера.	Виклик сторінки помилки 404 з пропозицією навігації на робочі розділи.
8	Контроль стану	Спроба взаємодії (редагування/видалення) із бронюванням, дата якого вже минула.	Кнопки дій неактивні, система блокує будь-які маніпуляції з історією.

Нижче детально розглянемо результати виконання зазначених сценаріїв, включаючи випадки, коли користувач ненавмисно або навмисно намагається порушити встановлену логіку роботи програми.

Розглянемо сценарій коли на сайт заходить новий користувач без облікового запису та бронювань. Спочатку він спробує зареєструватись, проте може ввести вже зайнятий email. Тоді система виведе повідомлення, зображене на рисунку 3.8:

Рисунок 3.8 – Повідомлення про зайнятий email

Якщо користувач введе інший email та зареєструється, ми зможемо побачити його дані в базі даних, таблиця “Users”. Запис нового користувача наведено на рисунку 3.9.

id	full_name	email	password	role
1	Менеджер	poolking@gmail.com	$2y$10$wL7lanxlv0yruyqJmxmMUeDSYvEg6yw…	manager
4	Богдан	bogdan.agafonkin@gmail.com	$2y$10$qlOq9uzFZr0QmHBT/pokHeyW6DKXMV…	user
6	Наталія	ngontar396@gmail.com	$2y$10$iTo01ub7Wf6Vqw8hBqDSjOEpxL7oZ3m…	user
7	Іван	ivan123456@gmail.com	$2y$10$.sWuTzW8myGc200OI8o/COzQYguPka…	user
NULL	NULL	NULL	NULL	NULL

Рисунок 3.9 – Запис щойно зареєстрованого користувача з ID = 7

Далі користувач спробує забронювати один з столів. Як було описано в розділі 3.3, зайняті години будуть підсвічені червоним та користувач не зможе їх вибрати. Якщо ж користувач спробує забронювати раніше існуючого та ввести більшу кількість годин, він побачить наступне (рис. 3.10).

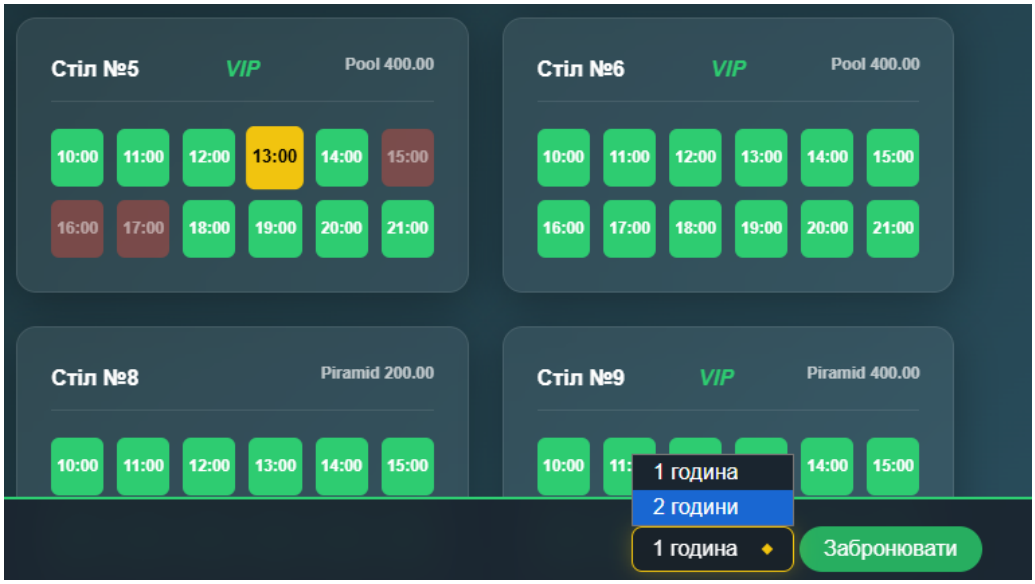

Рисунок 3.10 – Вигляд сітки бронювання

На рисунку 3.10 видно, що система не дозволяє користувачу забронювати стіл довше, ніж це можливо, оскільки на 15:00 вже існує бронювання. Тому користувач не бачить опції “3 години” – тільки 1 та 2. Також за цим самим принципом система не дозволяє забронювати стіл поза часом роботи закладу. Якщо ж користувач спробує ввести вручну тривалість гри (через інструменти розробника в браузері) система виведе помилку (рис. 3.11).

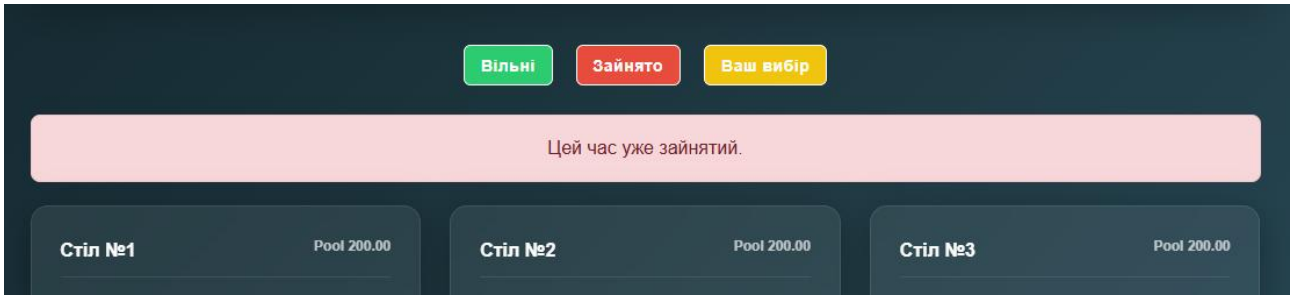

Рисунок 3.11 – Повідомлення про накладення часу

Далі користувач забронював стіл №2 з 18:00 до 21:00. Ми можемо переглянути його бронювання в базі даних, таблиця “Bookings” (рис. 3.12).

Result Grid | Filter Rows: | Edit:

id	user_id	table_id	booking_date	start_time	end_time
2	4	2	2026-06-03	18:00:00	19:00:00
3	4	2	2026-06-07	12:00:00	13:00:00
4	4	5	2026-06-07	15:00:00	18:00:00
6	4	4	2026-06-07	12:00:00	15:00:00
7	7	2	2026-06-07	18:00:00	21:00:00

Рисунок 3.12 – Бронювання користувача (user_id = 7) з ID = 7

Він може переглянути своє щойно створене бронювання в особистому кабінеті, відреагувати його або видалити. Особливістю системи є те, що ID користувача відображається в адресному рядку. І користувач, який хоче відредагувати замовлення з ID іншого користувача цього не зможе.

Припустимо, користувач змінив ID в адресному рядку з свого (7) на ID користувача “Богдан” (4). Після спроби змінити дані замовлення, користувач переміститься на сторінку кабінету і побачить що дані його та інших замовлень не змінились: база даних, при оновлені даних замовлення, перевіряє не тільки ID замовлення, а і ID власника бронювання.

Особливістю більярдного клубу “PoolKing” є те, що забронювати стіл можна тільки на завтра або пізніше. Розглянемо сценарій, коли користувач вручну змінює дату замовлення та спробує створити бронювання (рис. 3.13, 3.14).

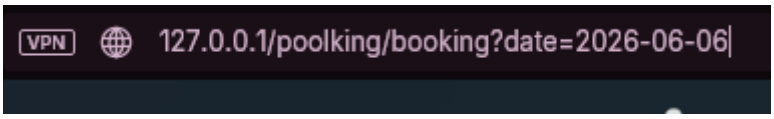

Рисунок 3.13 – Введення сьогоднішньої дати вручну

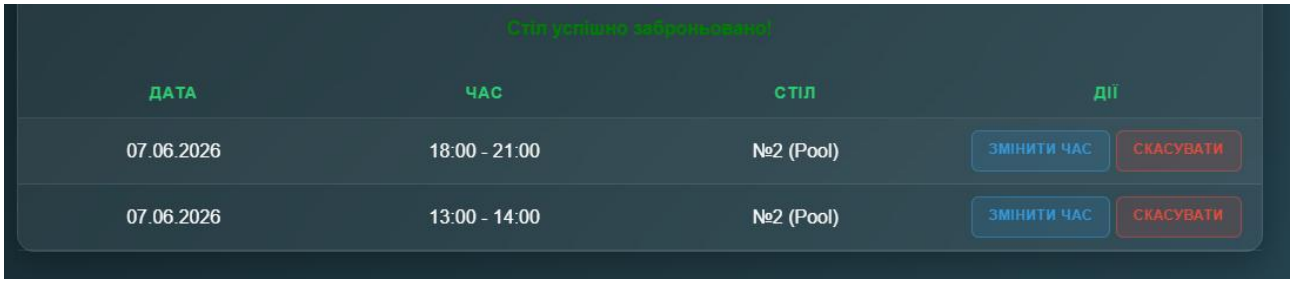

Рисунок 3.14 – Вигляд особистого кабінету

Як видно на рисунку 3.14, замовлення створилось але дата – завтра. Система не дозволить користувачу вибрати ранішу дату.

Далі розглянемо випадок, коли користувач спробує зайти на панель керування менеджера, не маючи таких прав доступу (рис. 3.15, 3.16).

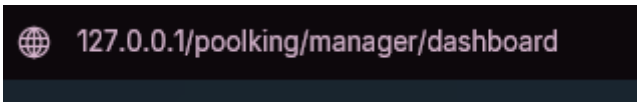

Рисунок 3.15 – Спроба входу на сторінку менеджера

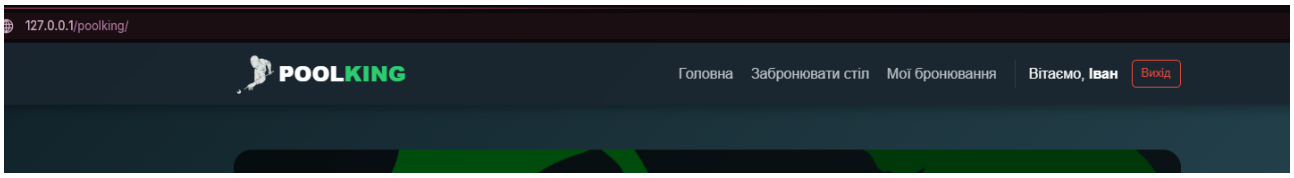

Рисунок 3.16 – Переадресація користувача на головну

Як видно на рисунку 3.16, система відмовила користувачу в доступі та повернула його на головну сторінку.

Якщо ж користувач помилиться при введені назви сторінки в адресному рядку, система виведе помилку 404 та запропонує повернутись на попередню сторінку або головну (рис. 3.17).

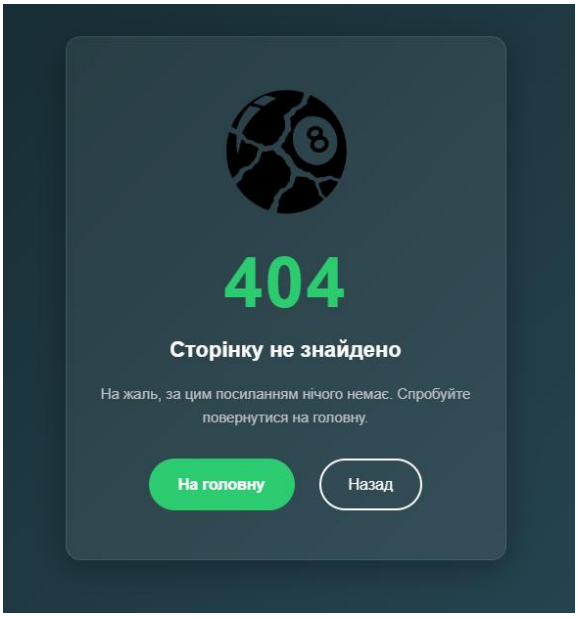

Рисунок 3.17 – Помилка 404 – сторінку не знайдено

Також в системі передбачена історія бронювань: якщо його час чи дата минув, воно не видаляється, а залишається на сторінці кабінету. Проте користувач ніяк не зможе з ним взаємодіяти. Таке бронювання (з минулою датою) зображено на рисунку 3.18.

Рисунок 3.18 – Вигляд актуального і неактуального бронювання

Розглянемо також сценарій, коли користувач неправильно ввів дані при авторизації (рис. 3.19, 3.20).

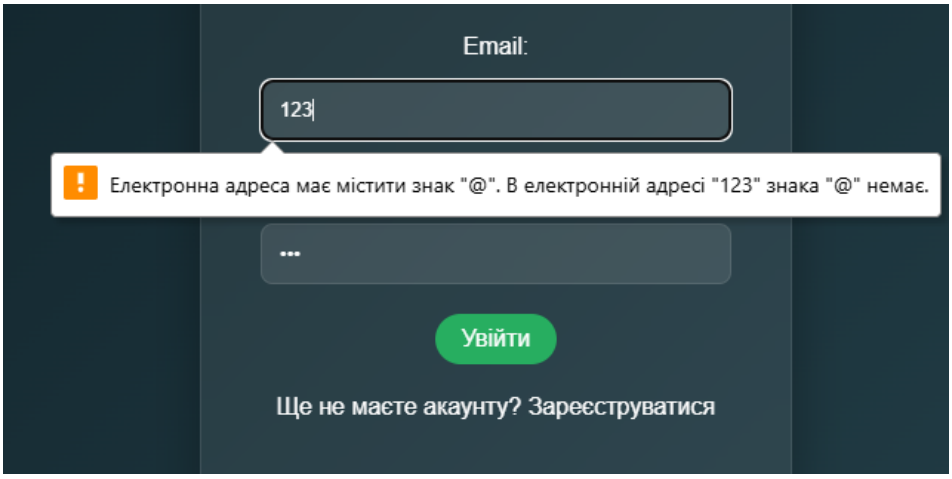

Рисунок 3.19 – Спроба введення недійсного email

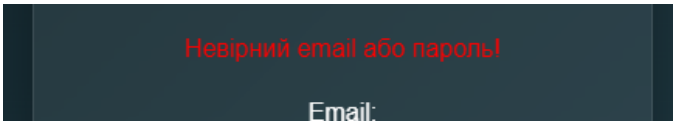

Рисунок 3.20 – Спроба введення невірних даних

Таким чином, практична реалізація всіх компонентів системи забезпечила створення функціонального та безпечного вебсервісу для автоматизації бронювання. Використання сучасних методів обробки даних та інтерактивного дизайну дозволило досягти високої швидкодії інтерфейсу та зручності управління ресурсами закладу.

ВИСНОВКИ

В умовах стрімкого розвитку цифрових технологій сфера послуг потребує впровадження ефективних інструментів для взаємодії з клієнтами. Основною метою виконання бакалаврської роботи було створення інформаційної системи, здатної автоматизувати процеси бронювання ресурсів більярдного клубу, позбувшись застарілих паперових методів обліку та мінімізувавши кількість помилок персоналу, спричинених “людським фактором”.

В першому розділі було проведено ґрунтовний аналіз предметної області, що дозволило визначити нормативно-правові та технічні вимоги до майбутнього продукту, зокрема у сфері захисту персональних даних та прав споживачів. Дослідження локального ринку послуг у місті Івано-Франківську підтвердило актуальність розробки, оскільки більшість наявних закладів досі не мають інтерактивних засобів онлайн-запису. Це дало змогу сформувати чітке технічне завдання та специфікацію функціональних ролей для клієнтів і менеджерів закладу.

Другий розділ став фундаментом для технічної реалізації системи, у ході якого було розроблено реляційну модель бази даних та ключові алгоритми перевірки доступності столів. Математична модель виявлення конфліктів часових інтервалів дозволила гарантувати цілісність даних та унеможливити ситуації подвійного бронювання одного ресурсу. Обґрунтування вибору технологічного стеку підтвердило раціональність використання мов HTML, CSS, PHP та бази даних MySQL для створення швидкого та відмовостійкого веб-сервісу з мінімальними витратами на підтримку.

Другий розділ став фундаментом для технічної реалізації системи, у ході якого було розроблено реляційну модель бази даних та ключові алгоритми перевірки доступності столів. Математична модель виявлення конфліктів часових інтервалів дозволила гарантувати цілісність даних та унеможливити ситуації подвійного бронювання одного ресурсу. Обґрунтування вибору технологічного стеку підтвердило раціональність використання мов HTML, CSS, PHP та бази

даних MySQL для створення швидкого та відмовостійкого веб-сервісу з мінімальними витратами на підтримку.

У третьому розділі було втілено всі закладені алгоритми та створено сучасний користувацький інтерфейс із застосуванням стилю Glassmorphism. Використання об'єктно-орієнтованого інтерфейсу PDO забезпечило високий рівень безпеки при роботі з базою даних, а впровадження технологій асинхронного обміну даними зробило процес вибору ігрових годин наочним та інтерактивним. Розроблені модулі особистого кабінету клієнта та панелі менеджера забезпечили повноцінний цикл управління замовленнями: від реєстрації відвідувача до автоматичного Email-сповіщення про зміни в розкладі.

Інформаційна система "PoolKing" є цілісним програмним продуктом, який успішно вирішує завдання автоматизації діяльності більярдного клубу. Система забезпечує цілодобовий доступ до послуг закладу, підвищує лояльність клієнтів завдяки зручності вибору часу на графічній сітці та надає адміністрації потужний інструмент для моніторингу завантаженості залів у реальному часі. Програма повністю відповідає вимогам безпеки, надійно зберігаючи конфіденційну інформацію та історію взаємодії з відвідувачами.

У процесі роботи було повністю досягнуто поставленої мети та реалізовано всі завдання згідно з вимогами. Отримані результати мають безпосередню практичну цінність і можуть бути впроваджені у роботу діючих закладів відпочинку. Розроблена архітектура системи залишається відкритою для подальшого масштабування, що дозволяє в майбутньому інтегрувати модулі онлайн-оплати або розширювати перелік додаткових послуг більярдного клубу.

ПЕРЕЛІК ПОСИЛАНЬ НА ДЖЕРЕЛА

1. Про захист прав споживачів : Закон України від 12.05.1991 № 1023-XII : станом на 24 груд. 2024 р. URL: https://zakon.rada.gov.ua/laws/show/1023-12#Text (дата звернення: 06.06.2026).

2. Про захист персональних даних : Закон України від 01.06.2010 № 2297-VI : станом на 14 черв. 2025 р. URL: https://zakon.rada.gov.ua/laws/show/2297-17#Text (дата звернення: 06.06.2026).

3. ДСТУ ISO/IEC 25010:2016 Системна та програмна інженерія. Вимоги до якості систем і програмного продукту та їх оцінювання.

4. China C. R. What is software as a service (SaaS)? | IBM. IBM. URL: https://www.ibm.com/think/topics/saas (дата звернення: 06.06.2026).

5. PHP tutorial. W3Schools Online Web Tutorials. URL: https://www.w3schools.com/php/default.asp (дата звернення: 06.06.2026).

6. Sommerville I. Software Engineering. – Pearson, 2015. – 816 с.

7. Norman D. The Design of Everyday Things. – Basic Books, 2013. – 368 с.

8. Anderson R. Security Engineering: A Guide to Building Dependable Distributed Systems. – Wiley, 2020. – 1232 с.

9. Шинкаренко В. Г. Економіка підприємства: сфера послуг : підручник. – Х. : ХНАДУ, 2017. – 312 с.

10. Ukrainian W. Уроки від W3Schools українською онлайн. W3Schools українською. Безплатні уроки онлайн для початківців, школярів та студентів. URL: https://w3schoolsua.github.io/css/index.html#gsc.tab=0 (дата звернення: 06.06.2026).

11. Васильєв О. Програмування мовою PHP. Ліра-К, 2024. 368 с.

12. Nichter D. Efficient MySQL Performance. O'Reilly Media, Incorporated, 2021.

13. MySQL Cookbook: Solutions for Database Developers and Administrators.

14. Botros S., Tinley J. High Performance MySQL. O'Reilly Media, Incorporated, 2021.

15. Learning PHP, MySQL & JavaScript: A Step-by-Step Guide to Creating Dynamic Websites. O'Reilly Media, 2021. 826 с.

16. PHP: PDO - Manual. PHP. URL: https://www.php.net/manual/en/class.pdo.php (дата звернення: 06.06.2026).

17. Що таке блок-схеми?. Experience - Discover How Dropbox Empowers Teams - Dropbox. URL: https://experience.dropbox.com/uk-ua/resources/flowcharts (дата звернення: 06.06.2026).

18. Більярд: історія, культура та різновиди гри. VIP Billiard. URL: https://vip-billiard.com.ua/bilyard-istoriya/ (дата звернення: 06.06.2026).

19. JavaScript | MDN. MDN Web Docs. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript (дата звернення: 06.06.2026).

20. JavaScript | W3Schools.com. W3Schools Online Web Tutorials. URL: https://www.w3schools.com/js/ (date of access: 06.06.2026).

21. CSS: Cascading Style Sheets | MDN. MDN Web Docs. URL: https://developer.mozilla.org/en-US/docs/Web/CSS (date of access: 06.06.2026).

22. CSS | Український веб-довідник. Український веб-довідник. URL: https://css.in.ua (дата звернення: 06.06.2026).

23. HTML5 - Glossary | MDN. MDN Web Docs. URL: https://developer.mozilla.org/en-US/docs/Glossary/HTML5 (date of access: 06.06.2026).

24. База даних MySQL. Promoter. URL: https://promoter.net.ua/articles/baza-danix-mysql.html (дата звернення: 06.06.2026).

25. MySQL | W3Schools.com. W3Schools Online Web Tutorials. URL: https://www.w3schools.com/MYSQL/default.asp (date of access: 06.06.2026).

ДОДАТКИ

ДОДАТОК А

Повний SQL-запит створення бази даних “PoolKing” та її таблиць

```
CREATE DATABASE `poolking` /*!40100 DEFAULT CHARACTER SET utf8mb4 COLLATE
utf8mb4_0900_ai_ci */ /*!80016 DEFAULT ENCRYPTION='N' */;

CREATE TABLE `users` (
  `id` int NOT NULL AUTO_INCREMENT,
  `full_name` varchar(100) NOT NULL,
  `email` varchar(100) NOT NULL,
  `password` varchar(255) NOT NULL,
  `role` enum('user','manager') NOT NULL DEFAULT 'user',
  PRIMARY KEY (`id`),
  UNIQUE KEY `email` (`email`)
) ENGINE=InnoDB AUTO_INCREMENT=7 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;

CREATE TABLE `billiard_tables` (
  `id` int NOT NULL AUTO_INCREMENT,
  `table_number` int NOT NULL,
  `category` enum('pool','snooker','piramid') NOT NULL,
  `price_per_hour` decimal(10,2) NOT NULL,
  `room_type` enum('default','vip') NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `table_number` (`table_number`)
) ENGINE=InnoDB AUTO_INCREMENT=17 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;

CREATE TABLE `bookings` (
  `id` int NOT NULL AUTO_INCREMENT,
  `user_id` int NOT NULL,
  `table_id` int NOT NULL,
  `booking_date` date NOT NULL,
  `start_time` time NOT NULL,
  `end_time` time NOT NULL,
  PRIMARY KEY (`id`),
  KEY `user_id` (`user_id`),
  KEY `table_id` (`table_id`),
  CONSTRAINT `bookings_ibfk_1` FOREIGN KEY (`user_id`) REFERENCES `users` (`id`) ON
DELETE CASCADE,
  CONSTRAINT `bookings_ibfk_2` FOREIGN KEY (`table_id`) REFERENCES `billiard_tables`
(`id`) ON DELETE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

ДОДАТОК Б

Повний код файлу db.php

```
<?php
$host = 'localhost:3306';
$db   = 'poolking';
$user = 'root';
$pass = '1111';
$charset = 'utf8mb4';

try {
    $dsn = "mysql:host=$host;dbname=$db;charset=$charset";
    $options = [
        PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION,
        PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC,
        PDO::ATTR_EMULATE_PREPARES => false
    ];
    $pdo = new PDO($dsn, $user, $pass, $options);
} catch (\PDOException $e) {
    die("Помилка підключення до бази даних: " . $e->getMessage());
}
```

ДОДАТОК В

Повний код файлу process.php

```
<?php
session_start();
require_once '../config/db.php';
require_once '../includes/functions.php';
if ($_SERVER['REQUEST_METHOD'] !== 'POST' || !isset($_SESSION['user_id'])) {
    header("Location: /poolking/booking");
    exit();
}
$current_role = $_SESSION['role'];
$user_id = $_SESSION['user_id'];
$action = $_POST['action'] ?? 'create';
$table_id = intval($_POST['table_id']);
$booking_date = $_POST['booking_date'];
$raw_start_time = $_POST['start_time'];
$duration = intval($_POST['duration']);
$booking_id = intval($_POST['booking_id']);
$start_time = date('H:i:s', strtotime($raw_start_time));
$end_timestamp = strtotime("$booking_date $start_time") + ($duration * 3600);
$end_time = date('H:i:s', $end_timestamp);
$club_open = "10:00:00";
$club_close = "22:00:00";
if ($start_time < $club_open || $end_time > $club_close || $end_time == "00:00:00") {
    $redirect = ($action === 'update') ? "/poolking/edit-
booking?id=".$booking_id."&error=outside_hours" :
"/poolking/booking?error=outside_hours";
    header("Location: $redirect");
    exit();
}
$tomorrow_date = date('Y-m-d', strtotime('+1 day'));
if ($booking_date < $tomorrow_date) {
    $redirect = ($action === 'update')
        ? "/poolking/edit-booking?id=".$booking_id."&error=invalid_date"
        : "/poolking/booking?error=invalid_date";
    header("Location: $redirect");
    exit();
}
$sql_check = "SELECT id, start_time, end_time FROM bookings
              WHERE table_id = :table_id
              AND booking_date = :b_date
              AND (start_time < :new_end
              AND end_time > :new_start)";
if ($action === 'update' && isset($_POST['booking_id'])) {
    $sql_check .= " AND id != :b_id";
}
$stmt_check = $pdo->prepare($sql_check);
$params = [
    ':table_id' => $table_id,
    ':b_date'   => $booking_date,
    ':new_end'  => $end_time,
    ':new_start'=> $start_time
];
if ($action === 'update') {
    $params[':b_id'] = $booking_id;
}
$stmt_check->execute($params);
```

```
$overlap = $stmt_check->fetch();
if ($overlap) {
    $error_type = "overlap";
    $redirect = ($action === 'update') ? "/poolking/edit-
booking?id=".$booking_id."&error=$error_type" :
"/poolking/booking?error=$error_type";
    header("Location: $redirect");
    exit();
}
if ($_SESSION['role'] === 'manager') {
    $stmt = $pdo->prepare("SELECT b.*, u.email, u.full_name, t.table_number
                           FROM bookings b
                           JOIN users u ON b.user_id = u.id
                           JOIN billiard_tables t ON b.table_id = t.id
                           WHERE b.id = ?");
    $stmt->execute([$booking_id]);
    $b_old = $stmt->fetch();
}
if ($action === 'update') {
    $sql = "UPDATE bookings SET table_id = ?, booking_date = ?, start_time = ?,
end_time = ?
            WHERE id = ? AND (user_id = ? OR ? = 'manager')";
    $stmt = $pdo->prepare($sql);
    $result = $stmt->execute([$table_id, $booking_date, $start_time, $end_time,
$booking_id, $user_id, $current_role]);
    $msg = "updated";
} else {
    $sql = "INSERT INTO bookings (user_id, table_id, booking_date, start_time,
end_time)
            VALUES (?, ?, ?, ?, ?)";
    $stmt = $pdo->prepare($sql);
    $result = $stmt->execute([$user_id, $table_id, $booking_date, $start_time,
$end_time]);
    $msg = "booked";
}
if ($result) {
    header("Location: /poolking/" . ($_SESSION['role'] === 'manager' ?
"manager/dashboard" : "profile") . "?success=$msg");
    if($_SESSION['role'] === 'manager') {
        $subject = "Зміна вашого бронювання №{$booking_id} у PoolKing";
        $message = "
            <h2>Шановний(а) {$b_old['full_name']}!</h2>
            <p>Ваше бронювання стола №{$b_old['table_number']} на дату
{$b_old['booking_date']} ({$b_old['start_time']}) було <b>перенесено
адміністрацією</b>.</p>
            <p>Нові дані вашого бронювання: Стіл №{$table_id} на дату {$booking_date}
({$start_time}).</p>
            <p>Для уточнення деталей зв'яжіться з менеджером по телефону +38(067)123-
45-67.</p>
            <p><b>Просимо вибачення за незручності!</b></p>
        ";
        sendStatusEmail($b_old['email'], $subject, $message);
    }
} else {
    echo "Помилка бази даних.";
}
```

ДОДАТОК Г

Повний код файлу home.php

```
<section class="hero-section">
    <div class="hero-content">
        <h1>Майстерність у кожному ударі</h1>
        <p>PoolKing — це не просто більярд, це атмосфера перемоги та відпочинку.</p>
        <div class="hero-btns">
            <a href="/poolking/booking" class="btn-main">Забронювати зараз</a>
            <a href="#about" class="btn-outline">Дізнатись більше</a>
        </div>
        <h3>PoolKing - Більярдний клуб нового рівня</h3>
        <p>PoolKing — це простір, де гра виходить за межі звичайного дозвілля і
перетворюється на справжню атмосферу змагання, відпочинку та стилю. Тут кожен стіл —
це можливість перевірити себе, покращити навички або просто добре провести час у
компанії друзів.</p>
        <p>Ми створили умови, де комфорт поєднується з якістю: професійні столи для
пулу, піраміди та снукеру, зручні зони відпочинку, бар і VIP-кімнати для тих, хто
цінує приватність. У нас однаково добре почуваються як новачки, так і досвідчені
гравці.</p>
        <h3>Атмосфера, що затягує</h3>
        <p>У клубі панує жива, але водночас спокійна атмосфера, яка дозволяє повністю
зануритись у гру. Тут проводяться турніри, дружні зустрічі та вечори, де більярд стає
не просто грою, а частиною стилю життя.</p>
        <p>Ми про деталі — світло, простір, якість обладнання і настрій. Саме це
створює місце, куди хочеться повертатись знову і знову.</p>
        <p>Дізнайся більше про наші столи, відпочинок у барі, VIP-кімнати та турніри
— все це доступно нижче на сторінці.<p></div></section>
<section id="about" class="features-grid">
    <div class="feature-card">
        <img src="./assets/img/home_pool.jpg" alt="Пул">
        <h3>Американський пул</h3>
        <p>Класичний пул — це динамічна гра, де важлива точність кожного удару і
вміння прораховувати ситуацію на кілька ходів вперед.</p></div>
    <div class="feature-card">
        <img src="./assets/img/home_piramid.jpg" alt="Піраміда">
        <h3>Піраміда</h3>
        <p>Українська піраміда — традиційний більярд, що поєднує точність, тактичне
мислення та контроль над кожним ударом.</p></div>
    <div class="feature-card">
        <img src="./assets/img/home_snooker.jpg" alt="Снукер">
        <h3>Снукер</h3>
        <p>Снукер — гра для тих, хто любить складні комбінації, максимальну
концентрацію і стратегічне планування кожного ходу.</p></div>
    <div class="feature-card">
        <img src="./assets/img/home_bar.jpg" alt="Бар">
        <h3>Бар</h3>
        <p>Барна зона — місце для відпочинку, спілкування та перерв між іграми, де
можна розслабитись і насолодитись атмосферою клубу.</p></div>
    <div class="feature-card">
        <img src="./assets/img/home_vip.jpg" alt="VIP кімнати">
        <h3>VIP-кімнати</h3>
        <p>VIP-кімнати створені для комфортної приватної гри, де можна
насолоджуватись більярдом у спокійній та преміальній атмосфері без зайвого
шуму.</p></div>
```

ДОДАТОК Д

Повний код авторизації, реєстрації, файлу керування ролями та завершення сесії

Д.1 Повний код файлу login.php

```
<section style="display: flex; justify-content: space-between;">
<div class="auth-form">
    <h2>Вхід у PoolKing</h2>
    <?php if(isset($_GET['error'])): ?>
        <p class="error" style="color: red;">Невірний email або пароль!</p>
    <?php endif; ?>
    <?php if(isset($_GET['success'])): ?>
        <p class="success" style="color: green;">Реєстрація успішна! Тепер
увійдіть.</p>
    <?php endif; ?>
    <form action="/poolking/auth/handler.php" method="POST">
        <input type="hidden" name="action" value="login">
        <div class="form-group">
            <label>Email:</label>
            <input type="email" name="email" required>
        </div>
        <div class="form-group">
            <label>Пароль:</label>
            <input type="password" name="password" required>
        </div>
        <button type="submit" class="btn-sub">Увійти</button>
    </form>
    <p>Ще не маєте акаунту? <a href="/poolking/register">Зареєструватися</a></p>
</div>
<div class="login-img">
    <img src="/poolking/assets/img/login.png" alt="Реєстрація">
</div>
</section>
```

Д.2 Повний код файлу register.php

```
<section style="display: flex; justify-content: space-between;">
<div class="auth-form">
    <h2>Реєстрація в PoolKing</h2>
    <?php if(isset($_GET['error'])): ?>
        <p class="error">Помилка: Такий Email вже зареєстровано!</p>
    <?php endif; ?>
    <form action="/poolking/auth/handler.php" method="POST">
        <input type="hidden" name="action" value="register">
        <div class="form-group">
            <label>Ваше ім'я:</label>
            <input type="text" name="full_name" required placeholder="Іван Іванов">
        </div>
        <div class="form-group">
            <label>Email:</label>
            <input type="email" name="email" required placeholder="example@mail.com">
        </div>
```

```
        <div class="form-group">
            <label>Пароль:</label>
            <input type="password" name="password" required minlength="6">
        </div>
        <button type="submit" class="btn-sub">Створити акаунт</button>
    </form>
    <p>Вже маєте акаунт? <a href="./login">Увійти</a></p>
</div>
<div class="login-img">
    <img src="/poolking/assets/img/login.png" alt="Реєстрація">
</div>
</section>
```

Д.3 Повний код файлу handler.php

```
<?php
session_start();
require_once '../config/db.php';
require_once '../includes/functions.php';
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $action = $_POST['action'] ?? '';
    if ($action === 'register') {
        $full_name = clean($_POST['full_name']);
        $email = clean($_POST['email']);
        $password = $_POST['password'];
        $checkEmail = $pdo->prepare("SELECT id FROM users WHERE email = ?");
        $checkEmail->execute([$email]);
        if ($checkEmail->rowCount() > 0) {
            header("Location: /poolking/register?error=email_exists");
            exit();}
        $hashedPassword = password_hash($password, PASSWORD_DEFAULT);
        $sql = "INSERT INTO users (full_name, email, password, role) VALUES (?, ?, ?,
'user')";
        $stmt = $pdo->prepare($sql);
        if ($stmt->execute([$full_name, $email, $hashedPassword])) {
            header("Location: /poolking/login?success=registered");
        } else {die("Помилка при реєстрації.");}}
    if ($action === 'login') {
        $email = clean($_POST['email']);
        $password = $_POST['password'];
        $stmt = $pdo->prepare("SELECT * FROM users WHERE email = ?");
        $stmt->execute([$email]);
        $user = $stmt->fetch();
        if ($user && password_verify($password, $user['password'])) {
            $_SESSION['user_id'] = $user['id'];
            $_SESSION['user_name'] = $user['full_name'];
            $_SESSION['role'] = $user['role'];
            if ($user['role'] === 'manager') {
header("Location: /poolking/manager/dashboard");
            } else {header("Location: /poolking/profile");}
        }
        else {
        header("Location: /poolking/login?error=wrong_auth");
      }
    }
}
else {
header("Location: /poolking/");}
```

Кінець додатку Д

Д.4 Повний код файлу logout.php

```
<?php
$_SESSION = [];
if (ini_get("session.use_cookies")) {
    $params = session_get_cookie_params();
    setcookie(session_name(), '', time() - 42000,
        $params["path"], $params["domain"],
        $params["secure"], $params["httponly"]
    );
}
session_destroy();
header("Location: /poolking/");
exit();
```

ДОДАТОК Е

Повний код шапки та підвалу сайту

Е.1 Файл header.php

```
<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title><?php echo isset($pageTitle) ? $pageTitle : 'PoolKing - Більярдний клуб';
?></title>
    <link rel="stylesheet" href="/poolking/assets/style.css">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/6.0.0/css/all.min.css">
    <link rel="icon" type="image/png" href="./assets/img/logo.png">
</head>
<body>
<header class="main-header">
    <div class="container">
        <div class="logo">
            <img src="/poolking/assets/img/logo.png" alt="Лого">
            <a href="/poolking/"><strong>Pool<span>King</span></strong></a>
        </div>
        <nav class="main-nav">
            <ul>
                <li><a href="/poolking/">Головна</a></li>
                <?php if (isset($_SESSION['user_id'])): ?>
                    <li><a href="/poolking/booking">Забронювати стіл</a></li>
                    <li><a href="/poolking/profile">Мої бронювання</a></li>
                    <?php if ($_SESSION['role'] === 'manager'): ?>
                        <li><a href="/poolking/manager/dashboard" class="manager-
link">Панель менеджера</a></li>
                    <?php endif; ?>
                    <li class="user-info">
                        <span>Вітаємо, <strong><?php echo
htmlspecialchars($_SESSION['user_name']); ?></strong></span>
                        <a href="/poolking/logout" class="logout-btn">Вихід</a>
                    </li>
                <?php else: ?>
                    <li><a href="/poolking/login">Увійти</a></li>
                    <li><a href="/poolking/register" class="btn-
reg">Реєстрація</a></li>
                <?php endif; ?>
            </ul>
        </nav>
    </div>
</header>
<main class="container content">
```

Е.2 Файл footer.php

```
</main>
<footer class="main-footer">
```

```
    <div class="container footer-content">
        <div class="footer-logo">
            <section>
                <img src="/poolking/assets/img/logo.png" alt="Лого">
                <a href="/poolking/">
                    <strong>Pool<span>King</span></strong>
                </a>
            </section>
            <p>Найкращий більярдний клуб у твоєму місті.</p>
        </div>
        <div class="footer-socials">
            <h4>Ми у соцмережах:</h4>
            <div class="social-icons">
                <a href="#"><i class="fab fa-instagram"></i></a>
                <a href="#"><i class="fab fa-facebook"></i></a>
                <a href="#"><i class="fab fa-telegram"></i></a>
                <a href="#"><i class="fab fa-youtube"></i></a>
            </div>
        </div>
        <div class="footer-contacts">
            <h4>Контакти:</h4>
            <p><i class="fas fa-map-marker-alt"></i> вул. Більярдна, 12</p>
            <p><i class="fas fa-phone"></i> +38 (067) 123-45-67</p>
        </div>
    </div>
    <div class="footer-bottom">
        <p>&copy; <?= date('Y') ?> PoolKing Club. Всі права захищені.</p>
    </div>
</footer>
<script src="/poolking/assets/main.js"></script>
</body>
</html>
```

ДОДАТОК Ж

Повний код файлу ajax_edit_slots.php

```
<?php
require_once '../config/db.php';

$table_id = intval($_GET['table_id']);
$date = $_GET['date'];
$current_b_id = intval($_GET['booking_id']);

$stmt = $pdo->prepare("SELECT start_time, end_time FROM bookings
                       WHERE table_id = ? AND booking_date = ? AND id != ?");
$stmt->execute([$table_id, $date, $current_b_id]);
$bookings = $stmt->fetchAll();

$booked_hours = [];
foreach ($bookings as $bk) {
    $start = intval(substr($bk['start_time'], 0, 2));
    $end = intval(substr($bk['end_time'], 0, 2));
    for ($i = $start; $i < $end; $i++) {
        $booked_hours[] = sprintf('%02d:00', $i);
    }
}
?>

<div class="timeline" style="gap: 20px !important;">
    <?php for ($h = 10; $h < 22; $h++):
        $slot = sprintf('%02d:00', $h);
        $is_busy = in_array($slot, $booked_hours);
    ?>
    <div class="hour-slot <?= $is_busy ? 'busy' : 'free' ?>"
        data-time="<?= $slot ?>"
        data-table-id="<?= $table_id ?>"
        onclick="<?= $is_busy ? '' : 'selectTime(this)' ?>">
        <?= $h ?>:00
    </div>
<?php endfor; ?>
</div>
```

ДОДАТОК И

Повний код файлу main.js

```
let currentSlot = null;
function resetSelection() {
    currentSlot = null;
    document.querySelectorAll('.hour-slot').forEach(el =>
el.classList.remove('selected'));
    const bar = document.getElementById('confirmation-bar');
    if (bar) {
        bar.classList.remove('active');
    }
}
function openTable(evt, tableId) {
    const tabContents = document.getElementsByClassName("tab-content");
    for (let i = 0; i < tabContents.length; i++) {
        tabContents[i].classList.remove("show");
    }
    const tabButtons = document.getElementsByClassName("tab-btn");
    for (let i = 0; i < tabButtons.length; i++) {
        tabButtons[i].classList.remove("active");
    }
    document.getElementById(tableId).classList.add("show");
    evt.currentTarget.classList.add("active");
}
function selectTime(element) {
    currentSlot = element;
    const tableId = element.getAttribute('data-table-id');
    const time = element.getAttribute('data-time');
    const hour = parseInt(time.split(':')[0]);
    document.getElementById('input-table-id').value = tableId;
    document.getElementById('input-start-time').value = time;
    document.getElementById('label-table').innerText = tableId;
    document.getElementById('label-time').innerText = time;
    const durationSelect = document.getElementById('input-duration');
    const timeline = element.parentElement;
    Array.from(durationSelect.options).forEach(opt => opt.hidden = false);
    for (let d = 2; d <= 3; d++) {
        let nextHour = hour + (d - 1);
        if (nextHour >= 22) {
            hideOptionsFrom(d, durationSelect);
            break;
        }
        let nextTimeStr = (nextHour < 10 ? '0' + nextHour : nextHour) + ':00';
        let nextSlot = timeline.querySelector(`[data-time="${nextTimeStr}"]`);
        if (!nextSlot || nextSlot.classList.contains('busy')) {
            hideOptionsFrom(d, durationSelect);
            break;
        }
    }
    if (durationSelect.options[durationSelect.selectedIndex].hidden) {
        durationSelect.value = "1";
    }
    const bar = document.getElementById('confirmation-bar');
    bar.classList.add('active');
    updateHighlights();
}
function hideOptionsFrom(start, select) {
```

```
    for (let i = 0; i < select.options.length; i++) {
        if (parseInt(select.options[i].value) >= start) {
            select.options[i].hidden = true;
        }
    }
}
function updateHighlights() {
    if (!currentSlot) return;
    document.querySelectorAll('.hour-slot').forEach(el =>
el.classList.remove('selected'));
    const startHour = parseInt(currentSlot.getAttribute('data-time').split(':')[0]);
    const duration = parseInt(document.getElementById('input-duration').value);
    const timeline = currentSlot.parentElement;
    for (let i = 0; i < duration; i++) {
        let h = startHour + i;
        let timeStr = (h < 10 ? '0' + h : h) + ':00';
        let slot = timeline.querySelector(`[data-time="${timeStr}"]`);
        if (slot) slot.classList.add('selected');
    }
}
document.addEventListener('click', function(event) {
    const bar = document.getElementById('confirmation-bar');
    if (!bar || !bar.classList.contains('active')) return;
    const isClickInsideBar = bar.contains(event.target);
    const isClickOnSlot = event.target.classList.contains('hour-slot');
    if (!isClickInsideBar && !isClickOnSlot) {
        resetSelection();
    }
});
const editTableSelect = document.getElementById('edit-table-select');
const editDateInput = document.getElementById('edit-date-input');
const editContainer = document.getElementById('edit-timeline-container');
function loadEditSlots() {
    if (!editTableSelect || !editDateInput) return;
    const tableId = editTableSelect.value;
    const date = editDateInput.value;
    fetch(`/poolking/user/ajax_edit_slots.php?table_id=${tableId}&date=${date}&bookin
g_id=${currentBookingId}`)
    .then(res => res.text())
    .then(html => {
        editContainer.innerHTML = html;
        const startInput = document.getElementById('input-start-time');
        if (startInput && startInput.value) {
            const activeSlot = editContainer.querySelector(`[data-
time="${startInput.value}"]`);
        }
    });
}
if (editTableSelect) {
    editTableSelect.addEventListener('change', loadEditSlots);
    editDateInput.addEventListener('change', loadEditSlots);
    document.getElementById('input-booking-date').value = editDateInput.value;
    loadEditSlots();
}
if (editDateInput) {
    editDateInput.addEventListener('change', (e) => {
        document.getElementById('input-booking-date').value = e.target.value;
    });
}
```

ДОДАТОК К

Повний код файлу booking.php

```
<?php
if (!isset($_SESSION['user_id'])) {
    header("Location: /poolking/login");
    exit();
}
$selected_date = $_GET['date'] ?? date('Y-m-d', strtotime('+1 day'));
$tomorrow = date('Y-m-d', strtotime('+1 day'));
if ($selected_date < $tomorrow) {
    $selected_date = $tomorrow;
}
$tables = $pdo->query("SELECT * FROM billiard_tables ORDER BY id ASC")->fetchAll();
$stmt = $pdo->prepare("SELECT table_id, start_time, end_time FROM bookings WHERE
booking_date = ?");
$stmt->execute([$selected_date]);
$bookings_raw = $stmt->fetchAll();
$booked_slots = [];
foreach ($bookings_raw as $b) {
    $booked_slots[$b['table_id']][] = [
        'start' => substr($b['start_time'], 0, 5),
        'end' => substr($b['end_time'], 0, 5)
    ];
}
?>
<div class="booking-container">
    <div class="date-selector">
        <p>Ми працюємо з 10:00 до 22:00 без вихідних!</p>
        <form method="GET" action="/poolking/booking">
            <label>Дата гри: </label>
            <input type="date" name="date" min="<?= date('Y-m-d', strtotime('+1
day')) ?>" value="<?= $selected_date ?>" onchange="this.form.submit()">
        </form>
    </div>
    <div class="legend">
        <div class="legend-item box-free">Вільні</div>
        <div class="legend-item box-busy">Зайнято</div>
        <div class="legend-item box-selected">Ваш вибір</div>
    </div>
    <?php if (isset($_GET['error'])): ?>
        <div style="background: #f8d7da; color: #721c24; padding: 15px; border-
radius: 8px; margin-bottom: 20px; text-align: center; border: 1px solid #f5c6cb;">
            <?php
                $errors = [
                    'overlap' => "Цей час уже зайнятий.",
                    'outside_hours' => "Ми працюємо з 10:00 до 22:00.",
                    'invalid_date' => "Неможливо забронювати на минулий час."
                ];
                echo $errors[$_GET['error']] ?? "Помилка бронювання.";
            ?>
        </div>
    <?php endif; ?>
    <div class="tables-grid">
        <?php foreach ($tables as $table): ?>
            <div class="table-card">
```

```
                <h4>Стіл №<?= $table['table_number']?><em><?=$table['room_type'] ===
'vip' ? 'VIP' : '' ?></em><span><?= ucfirst($table['category'])?> <?=
$table['price_per_hour'] ?></span></h4>
                <div class="timeline">
                    <?php
                    for ($h = 10; $h < 22; $h++):
                        $time_slot = sprintf('%02d:00', $h);
                        $is_busy = false;
                        if (isset($booked_slots[$table['id']])) {
                            foreach ($booked_slots[$table['id']] as $b) {
                                if ($time_slot >= $b['start'] && $time_slot <
$b['end']) {
                                    $is_busy = true; break;
                                }
                            }
                        }
                    ?>
                        <div class="hour-slot <?= $is_busy ? 'busy' : 'free' ?>"
                             data-time="<?= $time_slot ?>"
                             data-table-id="<?= $table['id'] ?>"
                             data-table-num="<?= $table['table_number'] ?>"
                             onclick="<?= $is_busy ? '' : 'selectTime(this)' ?>">
                            <?= $h ?>:00
                        </div>
                    <?php endfor; ?>
                </div>
            </div>
        <?php endforeach; ?>
    </div>
</div>
<div id="confirmation-bar">
    <div class="bar-content">
        <div class="bar-info">Стіл <strong>№<span id="label-table"></span></strong> о
<strong><span id="label-time"></span></strong></div>
        <form action="/poolking/user/process.php" method="POST" class="bar-form">
            <input type="hidden" name="booking_date" value="<?= $selected_date ?>">
            <input type="hidden" name="table_id" id="input-table-id">
            <input type="hidden" name="start_time" id="input-start-time">
            <select name="duration" class="modern-select" id="input-duration"
onchange="updateHighlights()">
                <option value="1">1 година</option>
                <option value="2">2 години</option>
                <option value="3">3 години</option>
            </select>
            <button type="submit" class="btn-sub">Забронювати</button>
        </form>
    </div>
</div>
```

ДОДАТОК Л

Код сторінок клієнта

Л.1 Повний код файлу profile.php

```
<?php
if (!isset($_SESSION['user_id'])) {
    header("Location: /poolking/login");
    exit();
}
$user_id = $_SESSION['user_id'];
$sql = "SELECT b.*, t.table_number, t.category, t.room_type
        FROM bookings b
        JOIN billiard_tables t ON b.table_id = t.id
        WHERE b.user_id = ?
        ORDER BY b.booking_date DESC, b.table_id DESC, b.start_time DESC";
$stmt = $pdo->prepare($sql);
$stmt->execute([$user_id]);
$bookings = $stmt->fetchAll();
?>
<div class="profile-container">
    <h2>Мій кабінет</h2>
    <p>Тут ви можете переглядати та керувати своїми бронюваннями.</p>
    <?php if (isset($_GET['success'])): ?>
        <p class="alert-success" style="color: green; font-weight: bold;">
            <?php
                if ($_GET['success'] == 'booked') echo "Стіл успішно заброньовано!";
                if ($_GET['success'] == 'cancelled') echo "Бронювання скасовано.";
                if ($_GET['success'] == 'updated') echo "Дані бронювання оновлено.";
            ?>
        </p>
    <?php endif; ?>
    <table class="bookings-table">
        <thead>
            <tr>
                <th style="padding: 10px;">Дата</th>
                <th style="padding: 10px;">Час</th>
                <th style="padding: 10px;">Стіл</th>
                <th style="padding: 10px;">Дії</th>
            </tr>
        </thead>
        <tbody>
            <?php if (count($bookings) > 0): ?>
                <?php foreach ($bookings as $b): ?>
                    <tr>
                        <td style="padding: 10px;"><?= date('d.m.Y',
strtotime($b['booking_date'])) ?></td>
                        <td style="padding: 10px;"><?= substr($b['start_time'], 0, 5)
?> - <?= substr($b['end_time'], 0, 5) ?></td>
                        <td style="padding: 10px;">№<?= $b['table_number'] ?> (<?=
ucfirst($b['category']) ?>) <em><?=$b['room_type'] === 'vip' ? 'VIP' : ''
?></em></td>
                        <td style="padding: 10px;">
                            <?php if (strtotime($b['booking_date'] . ' ' .
$b['start_time']) > time()): ?>
                                <a href="/poolking/edit-booking?id=<?= $b['id'] ?>"
class="btn-sm btn-edit">Змінити час</a>
```

```
                                    <a href="/poolking/user/cancel.php?id=<?= $b['id']
?>" class="btn-sm btn-cancel"
                                       onclick="return confirm('Ви впевнені, що хочете
скасувати це бронювання?')">Скасувати</a>
                                <?php else: ?>
                                    <span style="color: #999;">Недоступно</span>
                                <?php endif; ?>
                            </td>
                        </tr>
                    <?php endforeach; ?>
                <?php else: ?>
                    <tr>
                        <td colspan="4" >У вас ще немає бронювань.</td>
                    </tr>
                <?php endif; ?>
            </tbody>
        </table>
    </div>
```

Л.2 Повний код файлу edit_booking.php

```
<?php
if (!isset($_SESSION['user_id']) || !isset($_GET['id'])) {
    header("Location: /poolking/profile");
    exit();
}
$booking_id = intval($_GET['id']);
$user_id = $_SESSION['user_id'];
$stmt = $pdo->prepare("SELECT * FROM bookings WHERE id = ?");
$stmt->execute([$booking_id]);
$b = $stmt->fetch();
if (!$b) die("Бронювання не знайдено.");
$tables = $pdo->query("SELECT * FROM billiard_tables")->fetchAll();
?>
<div class="booking-container">
    <div class="glass-card" style="padding: 30px;">
        <h3>Редагування бронювання</h3>
        <p>Ви забронювали стіл №<?= $b['table_id'] ?> на <?= date('d.m.Y',
strtotime($b['booking_date'])) ?> (<?= substr($b['start_time'], 0, 5) ?> - <?=
substr($b['end_time'], 0, 5) ?>)</p>
        <div style="display: flex; gap: 20px; margin-bottom: 30px;">
            <div style="flex: 1;">
                <label>Оберіть стіл:</label>
                <select id="edit-table-select" name="table_id" class="modern-select"
style="width: 100%;">
                    <?php foreach ($tables as $t): ?>
                        <option value="<?= $t['id'] ?>" <?= ($t['id'] ==
$b['table_id']) ? 'selected' : '' ?>>
                            №<?= $t['table_number'] ?> (<?= ucfirst($t['category'])
?>) <em><?=$t['room_type'] === 'vip' ? 'VIP' : '' ?></em>
                        </option>
                    <?php endforeach; ?>
                </select>
            </div>
            <div style="flex: 1;">
                <label>Дата:</label>
                <input type="date" id="edit-date-input" name="booking_date"
                       value="<?= $b['booking_date'] ?>"
                       min="<?= date('Y-m-d', strtotime('+1 day')) ?>"
                       class="modern-input" style="width: 100%;">
            </div>
```

```
        </div>
        <div id="edit-timeline-container">
        </div>
    </div>
</div>
<div id="confirmation-bar">
    <div class="bar-content">
        <div class="bar-info">Оновлення: Стіл <strong>№<span id="label-
table"></span></strong> о <strong><span id="label-time"></span></strong></div>
        <form action="/poolking/user/process.php" method="POST" class="bar-form">
            <input type="hidden" name="action" value="update">
            <input type="hidden" name="booking_id" value="<?= $b['id'] ?>">
            <input type="hidden" name="table_id" id="input-table-id">
            <input type="hidden" name="booking_date" id="input-booking-date">
            <input type="hidden" name="start_time" id="input-start-time">
            <select name="duration" id="input-duration" class="modern-select"
onchange="updateHighlights()">
                <option value="1">1 година</option>
                <option value="2">2 години</option>
                <option value="3">3 години</option>
            </select>
            <button type="submit" class="btn-sub">Зберегти зміни</button>
        </form>
    </div>
</div>
<script>
    const currentBookingId = <?= $b['id'] ?>;
    const initialStartTime = "<?= substr($b['start_time'], 0, 5) ?>";
</script>
```

ДОДАТОК М

Панель керування та файл обробки дій менеджера

М.1 Повний код файлу dashboard.php

```
<?php
checkRole('manager');
$selected_date = $_GET['date'] ?? date('Y-m-d');
$tables = $pdo->query("SELECT * FROM billiard_tables ORDER BY id ASC")->fetchAll();
$sql = "SELECT b.table_id, b.id, b.start_time, b.end_time, u.full_name, u.email
        FROM bookings b
        JOIN users u ON b.user_id = u.id
        WHERE b.booking_date = ?
        ORDER BY b.start_time ASC";
$stmt = $pdo->prepare($sql);
$stmt->execute([$selected_date]);
$all_bookings = $stmt->fetchAll(PDO::FETCH_GROUP);
?>
<div class="manager-container">
    <h1>Керування закладом</h1>
    <?php if (isset($_GET['success'])): ?>
        <p class="alert-success" style="color: green; font-weight: bold;">
            <?php
                if ($_GET['success'] == 'cancelled') echo "Бронювання скасовано.";
                if ($_GET['success'] == 'updated') echo "Дані бронювання оновлено.";
            ?>
        </p>
    <?php endif; ?>
    <div class="date-filter">
        <form method="GET" action="/poolking/manager/dashboard">
            <label>Дата огляду:</label>
            <input type="date" name="date" value="<?= $selected_date ?>"
onchange="this.form.submit()">
        </form>
    </div>
    <div class="manager-tabs">
        <div class="tab-buttons">
            <?php foreach ($tables as $index => $table): ?>
                <button class="tab-btn <?= $index === 0 ? 'active' : '' ?>"
onclick="openTable(event, 'table-<?= $table['id'] ?>')">
                    Стіл <?= $table['table_number'] ?>
                </button>
            <?php endforeach; ?>
        </div>
        <?php foreach ($tables as $index => $table): ?>
            <div id="table-<?= $table['id'] ?>" class="tab-content <?= $index === 0 ?
'show' : '' ?>">
                <h3>Бронювання: Стіл №<?= $table['table_number'] ?> (<?=
ucfirst($table['category']) ?>) <em><?=$table['room_type'] === 'vip' ? 'VIP' : ''
?></em></h3>
                <table class="manager-table">
                    <thead>
                        <tr>
                            <th>Час</th>
                            <th>Клієнт</th>
                            <th>Контакти</th>
                            <th>Дії</th>
                        </tr>
```

```
                    </thead>
                    <tbody>
                        <?php if (isset($all_bookings[$table['id']])): ?>
                            <?php foreach ($all_bookings[$table['id']] as $b): ?>
                                <tr>
                                    <td><?= substr($b['start_time'], 0, 5) ?> - <?=
substr($b['end_time'], 0, 5) ?></td>
                                    <td><?= htmlspecialchars($b['full_name']) ?></td>
                                    <td><?= htmlspecialchars($b['email']) ?></td>
                                    <td>
                                        <div class="action-btns">
                                            <a href="/poolking/edit-booking?id=<?=
$b['id'] ?>" class="btn-sm btn-edit">Змінити</a>
                                            <form
action="/poolking/manager/actions.php" method="POST" style="display:inline;"
onsubmit="return confirm('Видалити бронювання та повідомити клієнта?')">
                                                <input type="hidden"
name="booking_id" value="<?= $b['id'] ?>">
                                                <input type="hidden" name="action"
value="cancel">
                                                <button type="submit" class="btn-sm
btn-cancel">Скасувати</button>
                                            </form>
                                        </div>
                                    </td>
                                </tr>
                            <?php endforeach; ?>
                        <?php else: ?>
                            <tr><td colspan="4">На цей стіл бронювань немає</td></tr>
                        <?php endif; ?>
                    </tbody>
                </table>
            </div>
        <?php endforeach; ?>
    </div>
</div>
```

М.2 Повний код файлу actions.php

```
<?php
session_start();
require_once '../config/db.php';
require_once '../includes/functions.php';
checkRole('manager');
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $booking_id = intval($_POST['booking_id']);
    $action = $_POST['action'];
    $sql = "SELECT b.*, u.email, u.full_name, t.table_number
            FROM bookings b
            JOIN users u ON b.user_id = u.id
            JOIN billiard_tables t ON b.table_id = t.id
            WHERE b.id = ?";
    $stmt = $pdo->prepare($sql);
    $stmt->execute([$booking_id]);
    $data = $stmt->fetch();
    if ($data) {
        if ($action === 'cancel') {
```

```
            $update = $pdo->prepare("DELETE FROM bookings WHERE id = ?");
            $update->execute([$booking_id]);

            $subject = "Скасування вашого бронювання у PoolKing";
            $message = "
                <h2>Шановний(а) {$data['full_name']}!</h2>

                <p>На жаль, ваше бронювання стола №{$data['table_number']} на дату
{$data['booking_date']} ({$data['start_time']}) було <b>скасовано
адміністрацією</b>.</p>
                <p>Для уточнення деталей зв'яжіться з менеджером по телефону
+38(067)123-45-67.</p>
                <p><b>Просимо вибачення за незручності!</b></p>
            ";
            sendStatusEmail($data['email'], $subject, $message);
        }
    }
    header("Location: /poolking/manager/dashboard?date=" . $data['booking_date']);
    exit();
}
```

ДОДАТОК Н

Повний код файлу functions.php

```
<?php
function checkRole($requiredRole) {
    if (!isset($_SESSION['user_id'])) {
        header("Location: /poolking/login");
        exit();
    }
    if ($_SESSION['role'] !== $requiredRole) {
        header("Location: /poolking/");
        exit;
    }
}

function isLoggedIn() {
    return isset($_SESSION['user_id']);
}

function clean($data) {
    return htmlspecialchars(strip_tags(trim($data)));
}

function sendStatusEmail($to, $subject, $message) {
    $headers = "MIME-Version: 1.0" . "\r\n";
    $headers .= "Content-type:text/html;charset=UTF-8" . "\r\n";
    $headers .= "From: PoolKing <poolking@gmail.com>" . "\r\n";
    return mail($to, $subject, $message, $headers);
}
```

БІБЛІОГРАФІЧНА ДОВІДКА

Тема бакалаврської роботи: **Розробка web-сайту більярдного клубу з функцією пріоритетного бронювання засобами HTML, PHP та SQL**

Обсяг пояснювальної записки 66 аркушів:

9 таблиць;

30 рисунків;

12 додатків.

Дата завершення роботи: *11 червня 2026р*.

Підпис студента- _____Агафонкін Б. В.

Протокол аналізу звіту подібності науковим керівником

Заявляю, що я ознайомився (-лась) з Повним звітом подібності, який був згенерований Системою виявлення і запобігання плагіату щодо роботи:

Автор: Агафонкін Богдан Владиславович

Співавтор:

Назва: 6Диплом

Науковий керівник: Гарасимів Тарас Григорович

Підрозділ: Каф. КСМ

Коефіцієнт подібності 1:4.03%

Мікропробіли: 4

Заміна букв: 0

Інтервали: 0

Білі знаки: 18

Дата створення звіту: 2026-06-09 03:42:30.0

Після аналізу Звіту подібності констатую наступне:

☒ **Запозичення, виявлені в роботі є законними і не є плагіатом. Рівень подібності не перевищує допустимої межі. Таким чином робота незалежна і приймається.**

☐ **Запозичення не є плагіатом, але перевищено граничне значення рівня подібностей. Таким чином робота повертається на доопрацювання.**

☐ **Виявлено запозичення і плагіат або навмисні текстові спотворення (маніпуляції), як передбачувані спроби укриття плагіату, які роблять роботу невідповідною вимогам законодавства (Ст. 32. ЗУ Про вищу освіту, пункт 3.1, Ст. 42. ЗУ Про освіту) та вимог НАЗЯВО (Критерій 5), а також кодексу етики і процедурам. Таким чином робота не приймається.**

Обгрунтування:

2026-06-09 Надія Ширмовська

Дата експерт