Національний університет «Львівська політехніка»

Кафедра програмного забезпечення

**КУРСОВА РОБОТА**

**з дисципліни «Бази даних»**

на тему:

«Інформаційна система для управління бронюванням настільних ігор у клубі»

Виконав:  
студент спеціальності 121

«Інженерія програмного забезпечення»

групи П3-34

Музика Микола Сергійович

Керівник:

доцент кафедри програмного забезпечення,

Білоіваненко Максим Вікторович

Оцінка:

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

Кількість балів \_\_\_\_\_\_ Оцінка ECTS \_\_\_\_

Члени комісії \_\_\_\_\_\_\_\_\_\_\_\_ …

(підпис)

\_\_\_\_\_\_\_\_\_\_\_\_ Білоіваненко М.В.

(підпис)

Львів – 2025 рік

**Зміст**

[**Розділ 1. Аналіз предметної області та постановка завдання** 2](#_Toc198002192)

[**1.1. Опис предметної області** 2](#_Toc198002193)

[**1.2. Вимоги до обробки даних** 4](#_Toc198002194)

[**III. Постановка завдання** 6](#_Toc198002195)

[**Розділ 2. Розробка моделей зберігання даних** 7](#_Toc198002196)

[**I. Концептуальне проектування** 7](#_Toc198002197)

[**II. Вимоги до системи накопичення даних** 8](#_Toc198002200)

[**III. Логічне проектування схеми бази даних** 9](#_Toc198002201)

[**IV. Реалізація процедур бізнес-логіки (тригери, процедури)** 12](#_Toc198002202)

[**Розділ 3. Розробка програмного коду** 19](#_Toc198002209)

[**I. Обґрунтування обраної архітектури** 19](#_Toc198002210)

[**II. Структурна модель інформаційної системи** 20](#_Toc198002211)

[**III. Призначення модулів та компонентів системи** 21](#_Toc198002212)

[**IV. Опис реалізації функцій системи** 22](#_Toc198002213)

[**V. Опис реалізації елементів інтерфейсу** 22](#_Toc198002214)

[**Розділ 4. Опрацювання даних** 28](#_Toc198002238)

[**I. Характеристика первинного джерела** 28](#_Toc198002239)

[**II. Опис процесу генерації тестових даних** 28](#_Toc198002240)

[**III. Алгоритм трансформації даних при завантаженні** 29](#_Toc198002241)

[**Розділ 5. Функціональні можливості** 30](#_Toc198002242)

[**I. Опис інтерфейсу у відповідності до бізнес-процесів** 30](#_Toc198002243)

[**II. Засоби аналітичного представлення даних** 48](#_Toc198002248)

[**III. Засоби експорту даних**](#_Toc198002249) [49](#_Toc198002250)

[**IV. Особливості додаткових реалізованих функцій** 50](#_Toc198002251)

[**Список літератури** 50](#_Toc198002252)

[**Висновки** 51](#_Toc198002253)

[**I. Встановлені недоліки** 51](#_Toc198002254)

[**II. Перспективи покращення** 52](#_Toc198002255)

[**Додатки** 53](#_Toc198002256)

[**I. Скрипт створення бази даних** 53](#_Toc198002257)

[**II. Скрипт завантаження первинних даних** 58](#_Toc198002258)

[**III. Sql запити, що відповідають бізнес процесам** 59](#_Toc198002259)

[**IIV. Програмний код реалізації бізнес-логіки** 61](#_Toc198002264)

[**V. Код побудови інтерфейсу користувача** 79](#_Toc198002265)

**Розділ 1. Аналіз предметної області та постановка завдання**

**1.1. Опис предметної області**

У повсякденні люди часто намагаються проводити своє дозвілля цікаво та небанально, часто їх вибір активності падає на настільні ігри. Та що робити як немає потрібної гри вдома, чи немає місця де пограти компанією, в такому випадку люди шукають такі ігрові клуби.

Актуальність вибраної теми полягає у потребі автоматизації процесів, зберігання, аналізу та представлення даних про реєстрації користувачів як для індивідуального користування, так і для бізнес-задач.

Розроблена інформаційна система призначена для автоматизації процесів бронювання столів і настільних ігор у клубі, полегшуючи управління ігровими сесіями, взаємодію між учасниками та адміністративний контроль. Вона забезпечує зручний інтерфейс як для відвідувачів, які планують своє дозвілля, так і для персоналу клубу, який керує розкладом та ресурсами.

Система дозволяє користувачам обирати гру, дату, час і стіл, оформлювати бронювання для себе чи команди, переглядати наявність вільних слотів та отримувати підтвердження. Водночас адміністрація має можливість контролювати стан інвентарю, визначати статус оплати та відмічати відвідуваність гравців. Окремим функціоналом є формування команд для багатокористувацьких ігор із можливістю автоматичного підбору гравців до сформованої групи.

Поняття:

* Бронювання – дія користувача з формування групи з декількох учасників, резервування ігрового столу на певний час для певної гри в певну дату.
* Група (команда) – об’єднання кількох користувачів для спільної участі в грі або матчі.
* Стіл – фізичне місце у клубі, доступне для бронювання.
* Настільна гра – одиниця ігрового контенту, яку можна обрати для гри.
* Оплата – фіксація факту розрахунку за заброньовану гру/стіл, або факту відміни бронювання, чи очікування на оплату.
* Чемпіонат – ігрова подія з монополії, яка проводиться між командами та має переможця.
* Користувач – загальне поняття, що описує зареєстровану особу в системі, яка має доступ до функціоналу відповідно до своєї ролі.
* Роль користувача – визначає набір дозволів та функціональних можливостей у системі. Основні ролі включають:

1. Відвідувач – може переглянути список ігор, актуальний стан турнірної таблиці, обрахувати ціну бронювання та зареєструватись.
2. Менеджер – має доступ до всіх можливостей системи, може додавати, видаляти ігри, дивитись статистику, дивитись бронювання за будь яку дату, додавати дані про матчі чемпіонату, змінювати стан оплати та видаляти бронювання.
3. Робітник – має можливість переглядати всі бронювання за сьогодні, найближче бронювання до актуального часу в даний момент, змінювати стан оплати та видаляти бронювання.
4. Зареєстрований користувач – фізична особа, яка є зареєстрованою в системі та може створювати бронювання, переглядати всі бронювання в яких вона бере участь та налаштовувати деталі акаунту.

* Статистика – модуль, що дозволяє отримати агреговану інформацію про оплати бронювання, кількість бронювань за місяць та топ найпопулярніших ігор за весь час. Підтримує фільтрацію за датою.

Обсяг аналізу:

Система забезпечує:

* Управління процесом бронювання ігор та столів за часом і датою.
* Формування та керування ігровими групами.
* **Відображення розкладу та доступності ресурсів**.
* **Управління каталогом ігор та станом інвентарю**.
* **Візуалізацію бронювань і статистики за періодами**.
* Облік відвідуваності та стану оплати за гру.
* Експорт статистичних даних та історії бронювань у формати CSV, XML, JSON.
* Управління матчами чемпіонату з монополії.

**1.2. Вимоги до обробки даних**

У розглянутій системі обробка даних ґрунтується на автоматизації процесів бронювання, формування ігрових сесій та адміністрування розкладу. Система призначена для накопичення, структурування та аналізу даних про користувачів, доступність ігор та столів, історію бронювань і статистику участі в ігрових подіях. Всі дані зберігаються у реляційній базі даних, де доступ до інформації регулюється згідно з ролями користувачів.

#### **1.2.1. Процес бронювання ігор та столів**

Цей бізнес-процес охоплює основну взаємодію користувача з системою. Він включає:

1. Вибір гри, дати, періоду та часу;
2. Перевірку доступності бронювання;
3. Введення кількості гравців;
4. Збереження заявки у базі даних.

Накопичені дані:

* Інформація про користувача, гру, стіл, дату й час;
* Кількість гравців та їхні контактні дані;
* Статус бронювання (активне, скасоване, завершене).

Аналітична складова:

* Визначення пікових годин/днів;
* Статистика популярності ігор;
* Середнє навантаження на столи.

#### **1.2.2. Формування команд для групових ігор**

Система підтримує формування команд з декількох користувачів під час реєстрації бронювання:

1. Користувач задає кількість гравців;
2. Користувач вводить дані кожного користувача;
3. Система автоматично записує користувачів при створенні команди, якщо вони ще не були зареєстровані.

Зберігаються:

* Дані про склад команди;
* Персональні ігрові профілі;
* Історія участі в групових сесіях.

Аналітика:

* Взаємодії різних гравців;
* Рейтинг активних гравців.

#### **1.2.3. Адміністрування ігор та інвентарю**

Менеджер має можливість керувати каталогом ігор, бронюваннями та станом інвентарю:

1. Додавання, редагування або видалення ігор;
2. Оцінка фізичного стану гри та столу;
3. Позначення статусу оплати.

Дані для накопичення:

* Ознаки зношеності інвентарю;
* Статуси чистоти та доступності столів;
* Статуси оплати користувачів та суми оплати;

Аналітика:

* Визначення ігор, що потребують заміни;
* Частота користування столами;
* Контроль виконаних оплат.

#### **1.2.4. Участь у чемпіонатах**

Система дозволяє створення матчів чемпіонату (фіксацію результатів):

1. Створення події;
2. Реєстрація учасників;
3. Збереження результатів.

**III. Постановка завдання**

Інформаційна система призначена для автоматизації процесів управління, каталогізації та обліку настільних ігор у рамках клубу, спільноти або домашньої колекції. Вона забезпечує зручне зберігання інформації про ігри, авторів, жанри, кількість гравців, тривалість партії та поточний статус (в наявності, у користуванні, втрачена тощо).

Головною метою є підвищення організованості в роботі працівників з клієнтами, спрощення пошуку відповідних деталей бронювання, а також підтримка менеджменту оплати бронювань та їх самих. Очікувані переваги від впровадження системи:

1. Централізований облік усіх ігор з ключовими характеристиками.
2. Ефективне управління ігровими сесіями та ведення історії бронювань.
3. Зменшення ризику втрати даних про бронювання користувачів, чи бронювання на той самий столик в один час.
4. Швидке відображення всіх бронювань певного столика для його кращого менеджменту та бронювання за всі столики, щоб робітники буи готові до приходу клієнтів.
5. Зручний інтерфейс для як менеджера з робітниками, так і рядових зареєстрованих користувачів.

**Розділ 2. Розробка моделей зберігання даних**

**I. Концептуальне проектування**

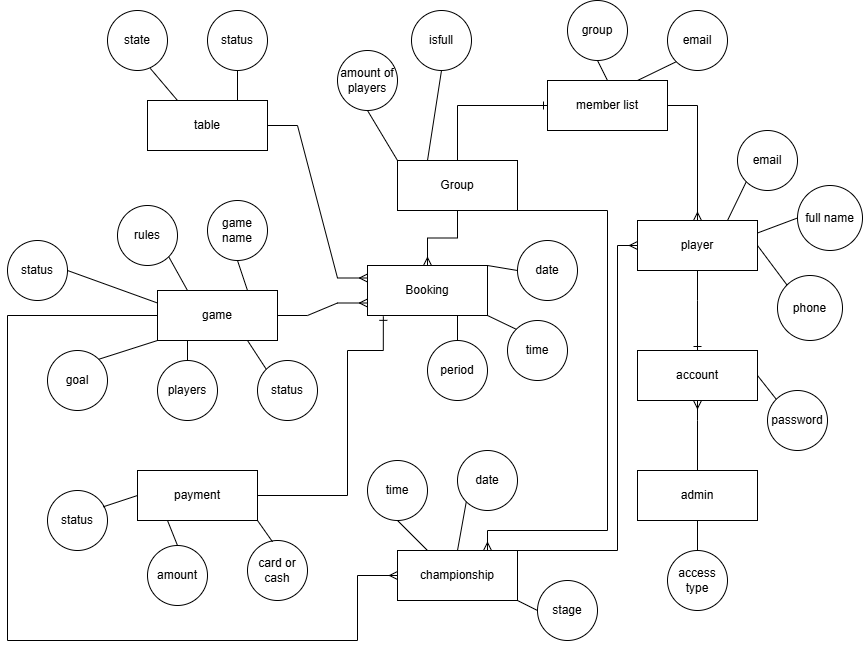
****

Рис. 2.1. ER-діаграма

Основні поняття та зв’язки:

* **Гравець (Player)** – особа, яка бере участь у грі, має ім’я, електронну пошту та номер телефону.
* **Обліковий запис (Account)** – запис для входу до системи, пов’язаний із гравцем через його email.
* **Група гравців (Group)** – команда гравців, яка може здійснювати бронювання або брати участь у турнірах.
* **Список учасників (MembersList)** – зв’язок між гравцями та групами, M:N.
* **Гра (Game)** – настільна гра з певними правилами, цілями, кількістю гравців та станом.
* **Стіл (Table)** – фізичне місце, де проходить гра. Має статус (зайнятий/вільний) та стан (чистий, брудний, непридатний).
* **Бронювання (Booking)** – запис, що фіксує резервування столу для певної групи на гру, дату та час.
* **Оплата (Payment)** – інформація про оплату бронювання (сума, статус, тип).
* **Адміністратор (Admin)** – працівник клубу, який має доступ до управління бронюваннями, таблицями, оплатами тощо.
* **Турнір (MonopolyChampionship)** – змагання між групами у грі «Монополія», з фіксацією переможця.

Взаємозв’язки між об’єктами моделі (ER-діаграма):

* **Обліковий запис** – **Гравець**: 1:1
* **Група гравців** – **Список учасників** – **Гравець**: M:N
* **Група гравців** – **Бронювання**: 1:N
* **Бронювання** – **Гра**, **Стіл**: N:1
* **Бронювання** – **Оплата**: 1:1
* **Турнір** – **Гра**: 1:1
* **Турнір** – **Група гравців, Гравець**: 1:N

**II. Вимоги до системи накопичення даних**

Оцінка обсягів даних:

* За день може бути до **100 бронювань** та **до 100 оплат**.
* Дані про ігри, столи, гравців та групи змінюються рідше.
* Матчі можуть проходити раз чи два на тиждень.

Частота додавання:

* Booking: кожне нове бронювання – запис.
* Payment: створюється автоматично при бронюванні (через тригер).
* MonopolyChampionship: запис додається після проведення матчу.
* MembersList: оновлюється при формуванні нової групи (автоматичне оновлення кількості гравців через тригер).

Очікувані запити:

* Пошук вільного столу на обрану дату та час.
* Отримання історії бронювань конкретного гравця або групи.
* Перегляд статистики участі в турнірах.
* Перевірка стану столів та ігор (нові, вживані, зіпсовані).
* Агрегація: кількість бронювань за місяць, найпопулярніші ігри, кількість бронювань з певною грою.

Обмеження доступу:

* Менеджер – повний доступ до всіх таблиць.
* Робітник – доступ до бронювань, оплат, гравців, груп, але без змін у структурі ігор.
* Зареєстрований користувач – перегляд ігор, створення бронювань, перегляд своїх записів.
* Відвідувач – лише перегляд доступних ігор та створення облікового запису.

**III. Логічне проектування схеми бази даних**

Таблиці мають зовнішні ключі відповідно до встановлених зв’язків та каскадне видалення (ON DELETE CASCADE) в деяких зв’язках.

Проектування бази:

Облікові записи (Account):

* **account\_id** (int, primary key): Унікальний ідентифікатор облікового запису.
* **password** (varchar(100)): Пароль користувача.
* **player\_email** (varchar(50), foreign key): Електронна пошта гравця (зв’язок з Player.email).

Адміністратори (Admin):

* **admin\_id** (int, primary key, foreign key): Ідентифікатор адміністратора (зв’язок з Account.account\_id).
* **access\_type** (varchar): Тип доступу (тільки 'employee' або 'manager').

Бронювання (Booking):

* **booking\_id** (int, primary key): Унікальний ідентифікатор бронювання.
* **group\_id** (int, foreign key): Ідентифікатор групи.
* **table\_id** (int, foreign key): Ідентифікатор столу.
* **game\_id** (int, foreign key): Ідентифікатор гри.
* **payment\_id** (int, foreign key): Ідентифікатор платежу.
* **date** (date): Дата бронювання.
* **time** (time): Час початку.
* **period** (time): Тривалість бронювання.

Ігри (Game):

* **game\_id** (int, primary key): Унікальний ідентифікатор гри.
* **game\_name** (varchar(100)): Назва гри.
* **rules** (text): Правила гри.
* **status** (varchar): Стан гри (допустимі значення: 'new', 'good', 'used', 'unusable').
* **players** (varchar(20)): Кількість гравців.
* **goal** (varchar): Мета гри.

Групи (Group):

* **group\_id** (int, primary key): Унікальний ідентифікатор групи.
* **players\_amount** (int): Кількість гравців (від 2 до 20).
* **is\_full** (boolean): Чи повна група.

Список учасників (Members\_List):

* **list\_id** (int, primary key): Унікальний ідентифікатор запису.
* **group\_id** (int, foreign key): Ідентифікатор групи.
* **email** (varchar(50), foreign key): Електронна пошта гравця.

Чемпіонат Монополії (Monopoly\_Championship):

* **play\_id** (int, primary key): Унікальний ідентифікатор гри чемпіонату.
* **group\_id** (int, foreign key): Ідентифікатор групи.
* **winner\_id** (varchar(50), foreign key): Електронна пошта переможця.
* **game\_id** (int, foreign key): Ідентифікатор гри.
* **date** (date): Дата проведення.
* **time** (time): Час проведення.
* **stage** (varchar): Стадія чемпіонату.

Платежі (Payment):

* **payment\_id** (int, primary key): Унікальний ідентифікатор платежу.
* **status** (varchar): Статус платежу (допустимі: 'booked', 'waiting', 'successful', 'declined', 'unbooked').
* **amount** (int): Сума платежу.
* **card\_or\_cash** (boolean): Тип оплати (картка або готівка).

Гравці (Player):

* **email** (varchar(50), primary key): Унікальна електронна адреса гравця.
* **full\_name** (varchar(100)): Повне ім’я гравця.
* **phone** (int): Номер телефону.

Столи (Table):

* **table\_id** (int, primary key): Унікальний ідентифікатор столу.
* **status** (boolean): Статус (активний/неактивний).
* **state** (varchar): Стан столу (можливі значення: 'clean', 'dirty', 'unusable').

Система індеків:

Для оптимізації роботи з базою даних заплановано створення наступних типів індексів:

1. Кластеризовані індекси на первинних ключах усіх таблиць — забезпечують швидкий доступ до даних за унікальним ідентифікатором:

* account.account\_id
* admin.admin\_id
* booking.booking\_id
* game.game\_id
* group.group\_id
* members\_list.list\_id
* monopoly\_championship.play\_id
* payment.payment\_id
* player.email
* table.table\_id

1. Некластеризовані індекси на зовнішніх ключах — пришвидшують операції JOIN між таблицями:

* booking.group\_id, booking.table\_id, booking.game\_id, booking.payment\_id
* members\_list.group\_id, members\_list.email
* monopoly\_championship.group\_id, monopoly\_championship.winner\_id, monopoly\_championship.game\_id
* account.player\_email
* admin.admin\_id

1. Композитні індекси для полів, що часто використовуються разом у запитах — у наявних скриптах композитні індекси не створювались, однак їх варто розглянути у разі частого використання запитів із кількома умовами (наприклад, date + group\_id у booking).
2. Фільтровані індекси для частих запитів з умовами WHERE — не створено, але доцільно реалізувати, наприклад:
   * Індекс на записи з status = 'successful' у таблиці payment
   * Індекс на table.status = true у таблиці table
3. Текстові індекси на поля, по яких проводиться пошук — не реалізовано явно, але можуть бути доцільними:
   * game.game\_name
   * player.full\_name
   * player.email

Для полів з великим обсягом даних (наприклад, text або varchar(max)), індекси не створюються, щоби уникнути зниження продуктивності.

**IV. Реалізація процедур бізнес-логіки (тригери, процедури)**

Для забезпечення цілісності даних та реалізації бізнес-правил у базі даних передбачається створення набору тригерів.

Основні тригери системи

### 1. Тригер перевірки конфлікту при бронюванні (trg\_check\_booking\_conflicts)

Тригер активується перед вставкою нового запису в таблицю booking і викликає функцію check\_booking\_conflicts. Функція перевіряє, чи не виникає конфліктів за часом по обраному столу (table\_id) та грі (game\_id). Якщо вказаний період уже зайнятий іншим бронюванням, вставка скасовується і повертається помилка з поясненням.

CREATE OR REPLACE TRIGGER trg\_check\_booking\_conflicts

BEFORE INSERT

ON public.booking

FOR EACH ROW

EXECUTE FUNCTION public.check\_booking\_conflicts();

### 2. Тригер створення оплати перед бронюванням (trg\_create\_payment\_before\_booking)

Цей тригер також спрацьовує перед вставкою в таблицю booking. Він викликає функцію create\_payment\_before\_booking, яка створює новий платіж на основі кількості гравців у групі (players\_amount) та тривалості бронювання (period). Обчислена сума додається в таблицю payment, після чого її payment\_id прив'язується до нового запису бронювання.

CREATE OR REPLACE TRIGGER trg\_create\_payment\_before\_booking

BEFORE INSERT

ON public.booking

FOR EACH ROW

EXECUTE FUNCTION public.create\_payment\_before\_booking();

### 3. Тригер оновлення кількості гравців у групі (trigger\_update\_players\_amount)

Цей тригер спрацьовує після додавання нового учасника до таблиці members\_list. Якщо кількість унікальних email-адрес для відповідної групи перевищує 2, функція update\_players\_amount оновлює відповідне поле players\_amount в таблиці group. Це дозволяє підтримувати актуальну інформацію про склад груп.

CREATE TRIGGER trigger\_update\_players\_amount

AFTER INSERT ON public.members\_list

FOR EACH ROW

EXECUTE FUNCTION update\_players\_amount();

Основні збережені процедури

### 1. check\_booking\_conflicts

**Призначення:**  
Запобігання дублюванню бронювань столів або ігор на один і той самий часовий інтервал.

**Логіка:**  
Підраховує кількість записів, які перетинаються з поточним запитом за table\_id та game\_id. Якщо знайдено конфлікти — генерує виключення з описом проблеми.

CREATE OR REPLACE FUNCTION public.check\_booking\_conflicts()

RETURNS trigger

LANGUAGE 'plpgsql'

COST 100

VOLATILE NOT LEAKPROOF

AS $BODY$

DECLARE

table\_conflict INTEGER;

game\_conflict INTEGER;

BEGIN

-- Перевірка конфлікту по столу

SELECT COUNT(\*) INTO table\_conflict

FROM public.booking

WHERE

table\_id = NEW.table\_id AND

date = NEW.date AND

(

(time < NEW.time + NEW.period::INTERVAL) AND

(time + period::INTERVAL > NEW.time)

);

IF table\_conflict > 0 THEN

RAISE EXCEPTION 'Table % is already booked for this time period.', NEW.table\_id;

END IF;

-- Перевірка конфлікту по грі

SELECT COUNT(\*) INTO game\_conflict

FROM public.booking

WHERE

game\_id = NEW.game\_id AND

date = NEW.date AND

(

(time < NEW.time + NEW.period::INTERVAL) AND

(time + period::INTERVAL > NEW.time)

);

IF game\_conflict > 0 THEN

RAISE EXCEPTION 'Game % is already booked for this time period.', NEW.game\_id;

END IF;

RETURN NEW;

END;

$BODY$;

ALTER FUNCTION public.check\_booking\_conflicts()

OWNER TO postgres;

### 2. create\_payment\_before\_booking

**Призначення:**  
Автоматичне створення пов'язаного платежу перед вставкою нового бронювання.

**Логіка:**  
Розраховує суму платежу на основі тривалості сесії та кількості гравців у групі.  
Використовується фіксована вартість (5 одиниць за годину на одного гравця).  
Платіж вставляється до таблиці payment, і його ID додається до нового запису бронювання.

CREATE OR REPLACE FUNCTION public.create\_payment\_before\_booking()

RETURNS trigger

LANGUAGE 'plpgsql'

COST 100

VOLATILE NOT LEAKPROOF

AS $BODY$

DECLARE

player\_count INTEGER;

period\_hours NUMERIC;

new\_payment\_id INTEGER;

BEGIN

-- Get number of players in the group

SELECT players\_amount INTO player\_count

FROM public.group

WHERE group\_id = NEW.group\_id;

-- Convert time period to total hours

SELECT EXTRACT(HOUR FROM NEW.period) + EXTRACT(MINUTE FROM NEW.period) / 60

INTO period\_hours;

-- Insert payment and retrieve payment\_id

INSERT INTO public.payment (status, card\_or\_cash, amount)

VALUES ('booked', true, CEIL(5 \* player\_count \* period\_hours))

RETURNING payment\_id INTO new\_payment\_id;

-- Assign the generated payment\_id to the booking row

NEW.payment\_id := new\_payment\_id;

RETURN NEW;

END;

$BODY$;

ALTER FUNCTION public.create\_payment\_before\_booking()

OWNER TO postgres;

### 3. update\_players\_amount

**Призначення:**  
Актуалізація кількості гравців у групі після додавання нового учасника.

**Логіка:**  
Якщо група має більше ніж 2 унікальних учасники, функція обчислює нову кількість гравців і оновлює відповідне поле в таблиці group.

CREATE OR REPLACE FUNCTION public.update\_players\_amount()

RETURNS trigger

LANGUAGE 'plpgsql'

COST 100

VOLATILE NOT LEAKPROOF

AS $BODY$

BEGIN

-- Only update if there are more than one member in the group

IF (

SELECT COUNT(DISTINCT email)

FROM public.members\_list

WHERE group\_id = NEW.group\_id

) > 2 THEN

UPDATE public."group"

SET players\_amount = (

SELECT COUNT(DISTINCT email)

FROM public.members\_list

WHERE group\_id = NEW.group\_id

)

WHERE group\_id = NEW.group\_id;

END IF;

RETURN NEW;

END;

$BODY$;

ALTER FUNCTION public.update\_players\_amount()

OWNER TO postgres;

**Розділ 3. Розробка програмного коду**

**I. Обґрунтування обраної архітектури**

Для розробки було обрано клієнт-серверну архітектуру з розділенням фронтенду та бекенду. Клієнтська частина реалізована у вигляді односторінкового додатку (SPA) на фреймворку Next.js, а серверна частина — за допомогою Node.js з використанням фреймворку Express.js. Це дозволяє досягти високої масштабованості, повторного використання коду та чіткого поділу відповідальності між частинами системи.

Мова програмування та фреймворк:

Основною мовою програмування обрано JavaScript, зокрема її сучасний синтаксис (ES6+). Для клієнтської частини застосовано React.js через фреймворк Next.js, а для серверної частини — Node.js, що забезпечує асинхронну обробку запитів. Це дозволило використовувати одну мову для всього стеку (full-stack JavaScript), що спрощує розробку, тестування і супровід системи.

**Робота з базою даних:**

У якості СУБД використано PostgreSQL, яка є потужною, стабільною і масштабованою реляційною системою з відкритим кодом. PostgreSQL підтримує складні запити, транзакції, перевірки цілісності та має широкі можливості для інтеграції з Node.js через бібліотеку pg.

Протоколи обміну:

Обмін даними між клієнтською та серверною частинами здійснюється за допомогою **HTTP/HTTPS** протоколів з використанням **RESTful API**. Сервер обробляє запити у форматі **JSON**, що є стандартом для веб-додатків. Обмін інформацією між клієнтом і сервером відбувається через HTTP-методи GET, POST, PUT, DELETE.

#### Архітектурні патерни:

Для організації коду серверної частини застосовано принципи **MVC-подібної структури**: маршрутизація (контролери), логіка обробки (сервіси) та взаємодія з базою даних (через SQL-запити). У фронтенді застосовано патерн **компонентної архітектури**, притаманний React-додаткам, а також використано **Redux Toolkit** для управління станом.

**II. Структурна модель інформаційної системи**

Клієнтська частина (Frontend)

Клієнтська частина реалізована у вигляді SPA з використанням Next.js. У додатку реалізовано функціонал авторизації, перегляду календаря бронювань, створення нових бронювань, перегляду матчів та детальної інформації про них. Для взаємодії з API використовується fetch, а стан додатку керується через Redux.

Серверна частина (Backend)

Сервер реалізовано на Node.js з використанням Express.js. Він відповідає за:

* зберігання і обробку інформації про ігри, матчі, бронювання;
* авторизацію користувачів у вигляді перевірки наявності емейлу користувача у списку активно авторизованих та відповідності емейлу до ключа що був динамічно створений при логіні;
* перевірку прав доступу до створення матчів/бронювань;
* взаємодію з базою даних через PostgreSQL.

База даних (Database)

У базі даних реалізовано наступні таблиці:

* player – інформація про гравців;
* group – групи гравців;
* members\_list – зв’язок між гравцями та групами;
* game, booking, monopoly\_championship – дані про ігри, бронювання, чемпіонати.

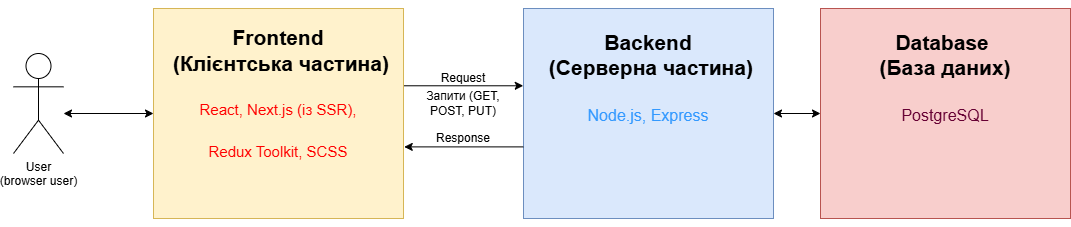


Рис. 3.2.1. Діаграма компонентів системи

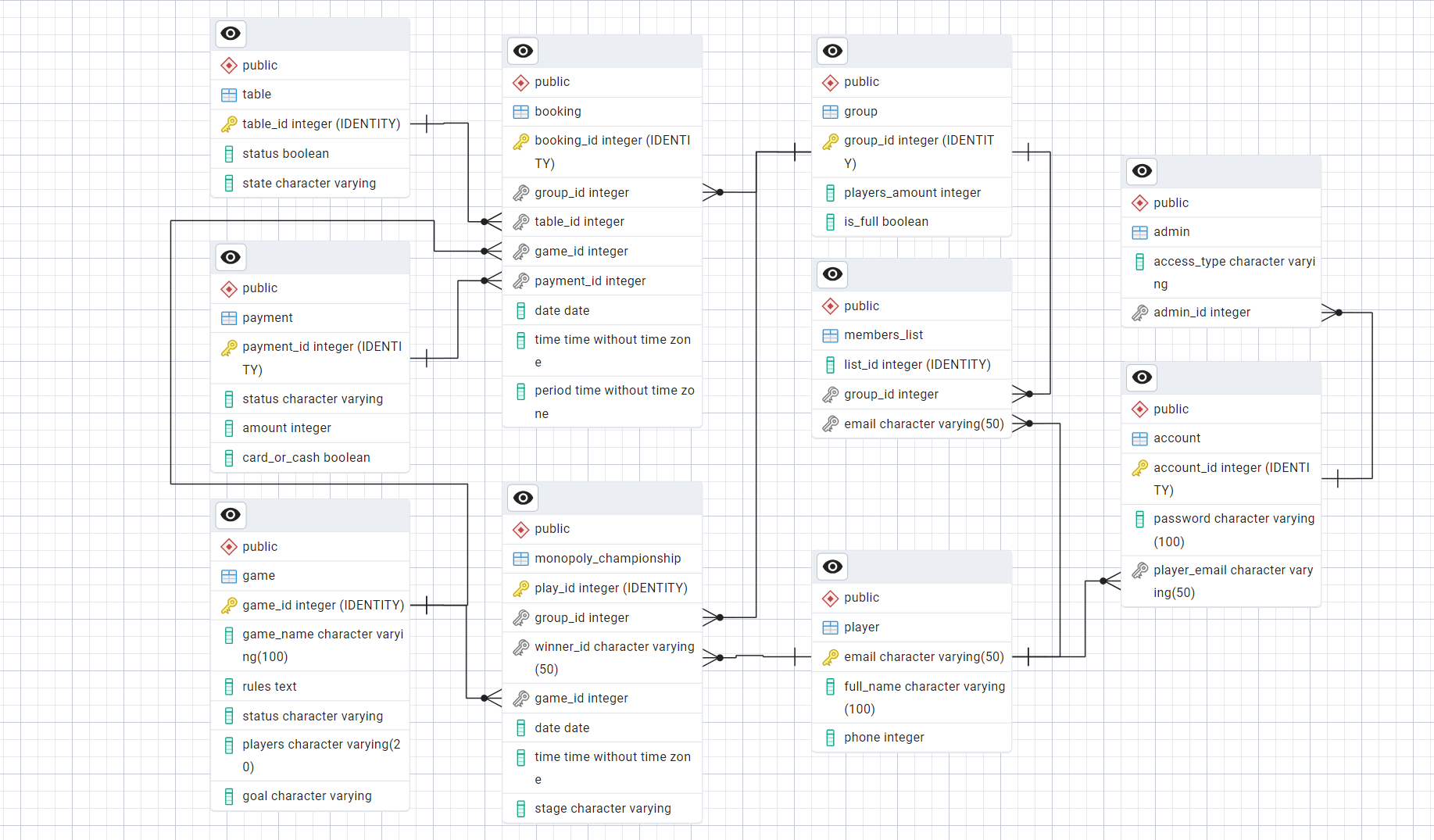


Рис. 3.2.2. Структурна модель бази даних (ER)

**III. Призначення модулів та компонентів системи**

#### 3.3.1. Presentation Layer

Відповідає за візуальне відображення даних користувачу. Усі компоненти Next.js, такі як User, EventDetails, Alert, реалізують логіку взаємодії та відображення даних. Використано бібліотеку **FullCalendar** для інтерактивного календаря.

#### 3.3.2. Application Layer

Містить логіку взаємодії з бекендом, наприклад, bookingsSlice.js реалізує функції для завантаження бронювань користувача. Тут же обробляються відповіді від API, включаючи перевірку автентифікації.

#### 3.3.3. Domain Layer

Реалізує основні бізнес-правила, зокрема перевірку доступу на сервері, логіку створення матчів (створення гравців, груп, матчів), бронювань (вибір столу, перевірка гравців).

#### 3.3.4. Infrastructure Layer

Інфраструктурний рівень відповідає за з’єднання з базою даних через бібліотеку pg, виконання SQL-запитів, обробку транзакцій, підключення middlewares.

**IV. Опис реалізації функцій системи**

#### 3.4.1. Багаторівнева система валідації

Валідація реалізована як на клієнті (перевірка полів перед відправкою), так і на сервері (перевірка типів, формату email, номера телефону, кількості гравців). Це дозволяє зменшити ймовірність помилок та забезпечити цілісність даних.

#### 3.4.2. Сповіщення у режимі реального часу

Функція сповіщень поки реалізована лише у вигляді інтерактивних поп-ап вікон (наприклад, про недоступність дати або підтвердження створення бронювання). У подальшому можлива інтеграція WebSocket.

#### 3.4.3. Асинхронність більшості сервісів

Усі запити до сервера виконуються асинхронно за допомогою async/await. Це забезпечує неблокуючу обробку операцій, зокрема взаємодії з базою даних.

#### 3.4.4. Реалізація авторизації на рівні БД

У системі передбачено багаторівневу авторизацію: користувачі, менеджери та працівники мають різні права доступу, що перевіряється як на клієнті, так і на сервері через змінну loggedUsers.

**V. Опис реалізації елементів інтерфейсу**

Для реалізації клієнтської частини було обрано React.js — сучасну JavaScript бібліотеку, яка широко використовується для створення динамічних односторінкових застосунків (SPA). Основні переваги React:

* Компонентна структура, що дозволяє повторно використовувати елементи інтерфейсу.
* Висока продуктивність завдяки використанню Virtual DOM.
* Велика спільнота та екосистема бібліотек.
* Сумісність з TypeScript для статичної типізації.

Також було використано **SCSS** для оптимізації та створення адаптивного дизайну.

## **1. Відвідувач**

Це роль користувача, який ще не створив обліковий запис. Може переглядати основну інформацію про сервіс.

### **Навігаційне меню**

Мінімальне — включає:

* **Main (Головна)** — короткий опис сервісу.
* Price — простий калькулятор для ціни бронювання.
* **Games List** — список наявних настільних ігор.
* **Championship** — відображення турнірної таблиці.
* **Register/Login** — форма реєстрації/авторизації.

### **Main (Головна сторінка)**

Інформаційна сторінка про сервіс.

**Елементи:**

* Загальний опис платформи.
* Кнопки реєстрації та входу.
* Кнопка обрахунку ціни.
* Пошук ігор.

### **Price**

Сторінка для обрахунку ціни потенційного бронювання.

**Елементи:**

* Пояснення цінової політики.
* Кнопки переходу до реєстрації та входу.
* Кнопка обрахунку ціни.

### **Games List (Список ігор)**

Огляд настільних ігор, доступних у закладі.

**Елементи:**

* Список наявних ігор.
* Сторінка кожної гри з описом, метою, правилами.

## **2. Зареєстрований користувач**

Має особистий кабінет і можливість бронювання.

### **Навігаційне меню**

* Main
* Add booking
* Account details
* Logout

### **Main (Бронювання)**

Можливість створення та перегляду бронювань.

**Елементи:**

* Календар з відмітками бронювань в датах.
* Інформація по існуючих бронюваннях.
* Ім’я користувача.
* Додавання бронювання.

### **Add booking (Додати бронювання)**

Можливість створення бронювань.

**Елементи:**

* Форма бронювання: дата, час, кількість осіб, обрана гра, список гравців з деталями.
* Кнопка створення бронювання.

### **Account details (Профіль користувача)**

Особисті дані та налаштування.

**Елементи:**

* ПІБ, логін, email.
* Можливість змінити пароль, повне ім’я чи номер.

## **3. Робітник**

Має доступ до керування ігровим простором, але не до адміністрування системи.

### **Навігаційне меню**

* Main
* Games State
* User account
* Logout

### **Main (Поточна зміна)**

Інформація про поточний стан закладу.

**Елементи:**

* Список активних бронювань на день для кожного столику окремо.
* Відображення всіх столиків закладу
* Інформація про найближче бронювання для кожного столику.
* Видалення бронювання та зміна стану оплати.

### **Games State(Ігровий інвентар)**

Управління станом ігор.

**Елементи:**

* Список усіх ігор.
* Статус кожної гри.
* Відповідні правила, кількість гравців та мета гри.

### **User account (Інтерфейс зареєстрованого користувача)**

Перехід до інтерфейсу зареєстрованого користувача.

Logout (вихід із системи до інтерфейсу відвідувача)

## **4. Менеджер**

Повний доступ до всіх функцій: адміністрування, статистика, управління персоналом.

### **Навігаційне меню**

* Main
* Games State
* Championship
* Statistics
* Export Bookings
* User account
* Logout

### **Main (Аналітика та огляд діяльності)**

Головна сторінка з схемою столів.

**Елементи:**

* Список активних бронювань на день для кожного столику окремо.
* Відображення всіх столиків закладу
* Інформація про найближче бронювання для кожного столику.
* Видалення бронювання та зміна стану оплати.
* Дата для обрання бронювань по даті.

### **Games State(Керування іграми)**

Повноцінне адміністрування ігрового інвентарю.

**Елементи:**

* Створення нових, видалення ігор.
* Список усіх ігор.
* Статус кожної гри.
* Відповідні правила, кількість гравців та мета гри.

### **Championship (Керування персоналом)**

Адміністрування чемпіонату.

**Елементи:**

* Схема турнірної таблиці.
* Додавання нових записів про проведені матчі.
* Відображення деталей вже доданих матчів.

### **Statistics (Звіти)**

Формування статистики (звітності) по роботі закладу.

**Елементи:**

* Автоматичне формування діаграм року вибраного в даті.
* Діаграми місячного доходу, кількості реєстрацій по днях місяця та топ найпопулярніших ігор.
* Виведення числових метрик.

### **Export Bookings (Звіти)**

Формування звітносту по роботі закладу у формі списку бронювань з деталями.

**Елементи:**

* Автоматичне формування таблиці зі списком бронювань та завантаження на локальний пристрій.

### **User account (Інтерфейс зареєстрованого користувача)**

Перехід до інтерфейсу зареєстрованого користувача.

Logout (вихід із системи до інтерфейсу відвідувача)

**Розділ 4. Опрацювання даних**

**I. Характеристика первинного джерела**

Первинним джерелом даних у процесі тестування системи бронювання виступають згенеровані штучно дані, які імітують типові сценарії реального користування сервісом настільних ігор. Оскільки система перебуває на етапі розробки, реальні бронювання ще не здійснюються, тому тестування функціоналу відбувається виключно на основі генерації умовних записів.

Згенеровані дані містять такі поля:

* Ім’я та прізвище користувача
* Дата та час бронювання
* Ідентифікатор гри
* Ідентифікатор столу
* Кількість гравців
* Статус бронювання (активне, скасоване, завершене)
* Дата створення бронювання

**II. Опис процесу генерації тестових даних**

Генерація даних реалізована за допомогою скрипта на Python із використанням бібліотек Faker, random та datetime.

Ключові етапи генерації:

* Імітація користувачів: створення імен та email-адрес на основі псевдовипадкових даних.
* Вибір гри та столу: ідентифікатори ігор та столів вибираються з заздалегідь заданих списків.
* Генерація дати та часу: випадкова дата бронювання в межах останніх 3 місяців.
* Кількість гравців: випадкове число в діапазоні від 2 до 6.
* Статус: випадковий вибір одного зі статусів (наприклад, "successful", "declined", "waiting").

**III. Алгоритм трансформації даних при завантаженні**

Усі згенеровані дані попередньо формуються у форматі JSON. Далі реалізовано наступні етапи:

1. Валідація: перевірка структури JSON, обов’язкових полів (дата, користувач, статус), відповідності типів даних.
2. Нормалізація: автоматичне приведення дат до UTC-формату, переведення імен у правильний регістр, очищення електронної пошти.
3. Мапінг: перетворення JSON у DTO-об'єкти системи (наприклад, BookingDTO) згідно з моделлю BookingEntity.
4. Збереження: імпорт у реляційну базу даних PostgreSQL (або вашу конкретну БД) через Spring Data JPA.

**Розділ 5. Функціональні можливості**

**I. Опис інтерфейсу у відповідності до бізнес-процесів**

### **5.1.1. Бізнес-процес ознайомлення з сервісом (роль — Відвідувач)**

Цей процес охоплює первинне знайомство користувача із платформою. Основне завдання — надати базову інформацію, заохотити зареєструватися та дозволити попередньо ознайомитись із іграми та ціноутворенням.

**Ключові дії користувача:**

* Перегляд сторінки **Main** з описом платформи, можливостями та закликом до реєстрації.

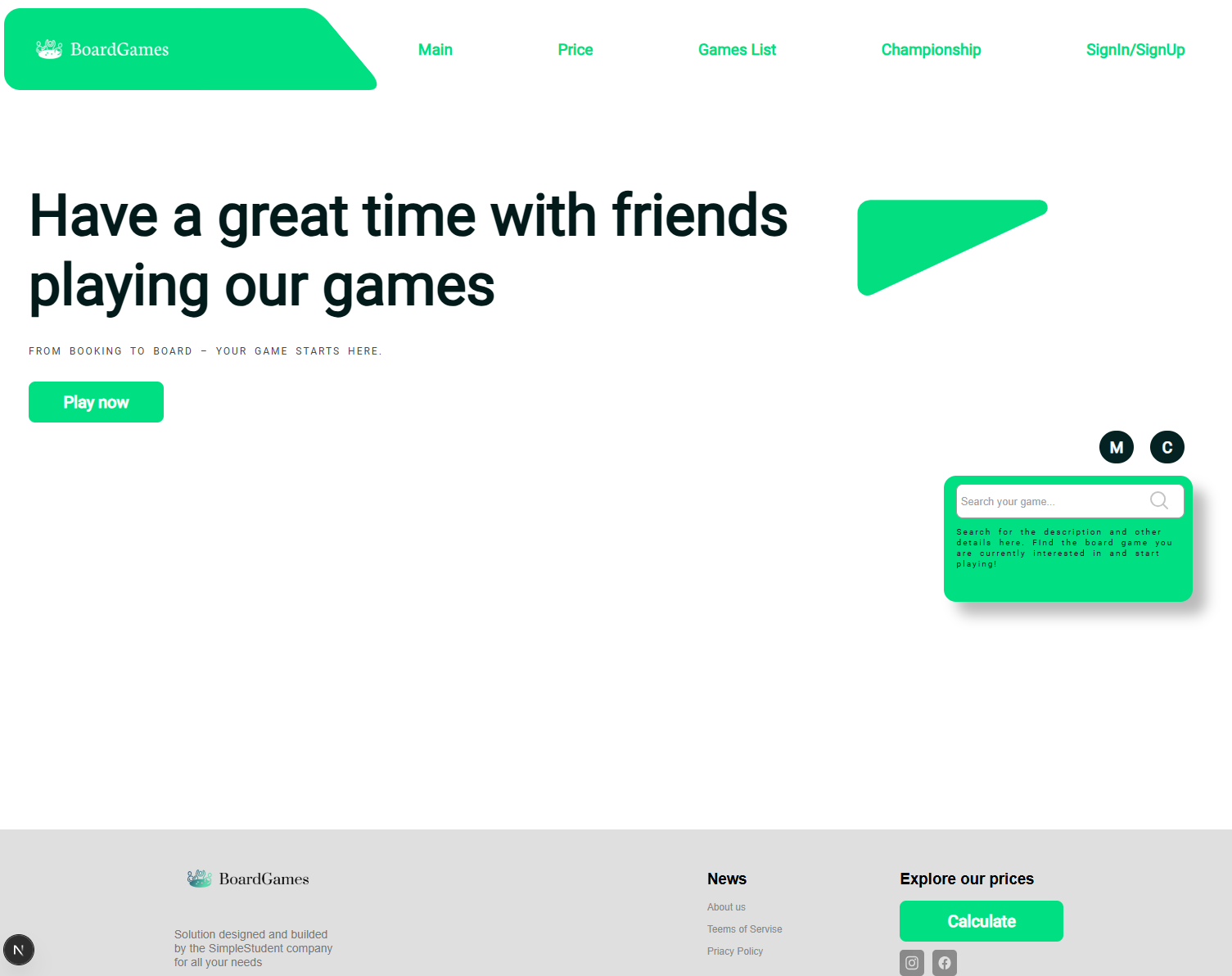


Рис. 5.1.1.1 Перегляд сторінки **Main**

* Використання **Games List** для перегляду асортименту ігор, їх правил та описів.



Рис. 5.1.1.2 **Games List** для перегляду

* Перевірка вартості потенційного бронювання через **Price** із калькулятором.



Рис. 5.1.1.3 **Price** із калькулятором

* Ознайомлення з активними матчами чемпіонату через **Championship**.

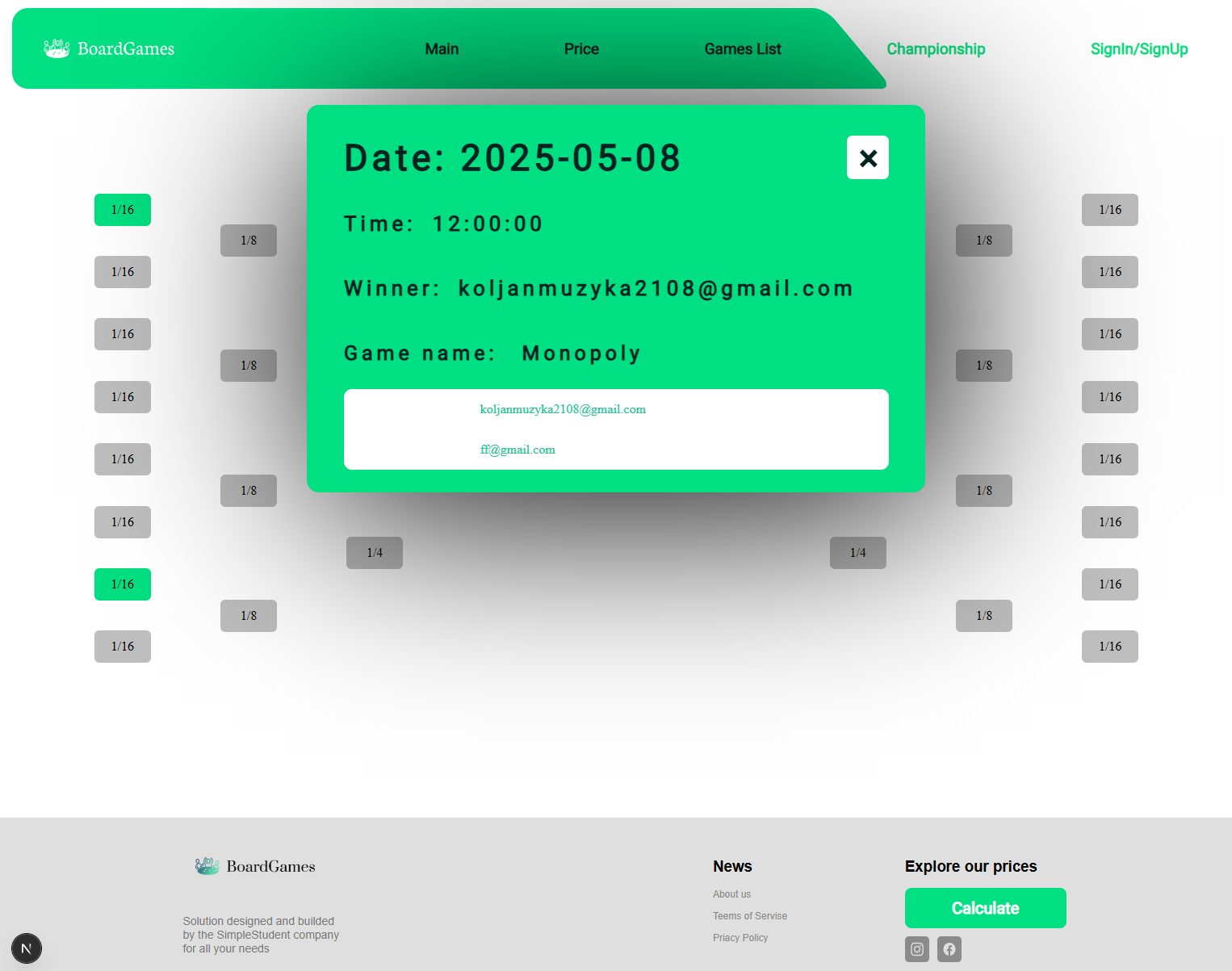
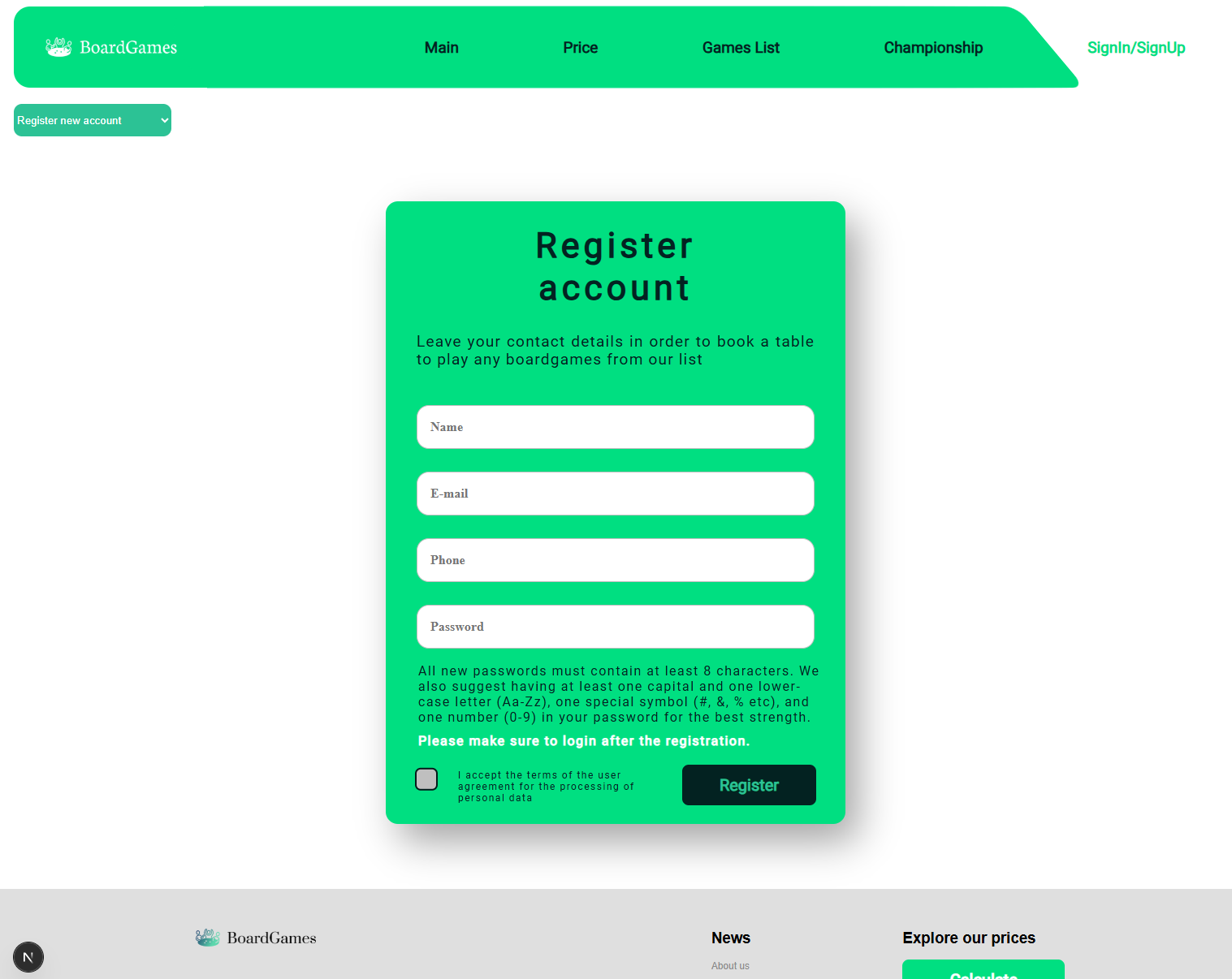


Рис. 5.1.1.4 **Championship**

* Переходи до сторінок **Register** та **Login** для подальшої реєстрації/авторизації.



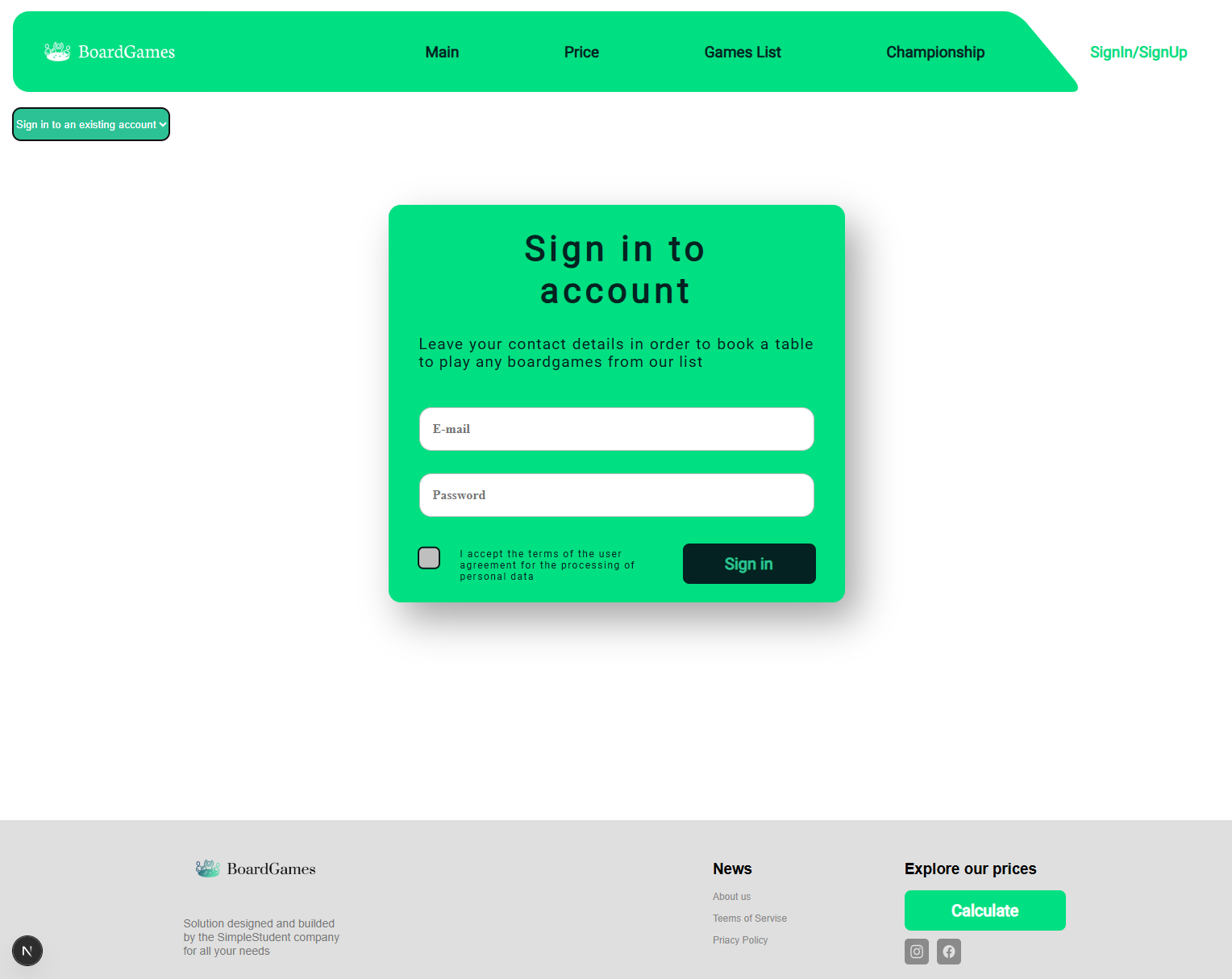


Рис. 5.1.1.4/5 Login/Register

Цей процес не передбачає створення бронювань, але готує користувача до них, демонструючи переваги сервісу.

### **5.1.2. Бізнес-процес створення бронювання (роль — Зареєстрований користувач)**

Цей бізнес-процес забезпечує можливість клієнтам самостійно створювати бронювання за обраними параметрами.

**Ключові етапи:**

* Користувач авторизується та переходить на сторінку **Main**, де бачить усі свої бронювання в календарному вигляді.

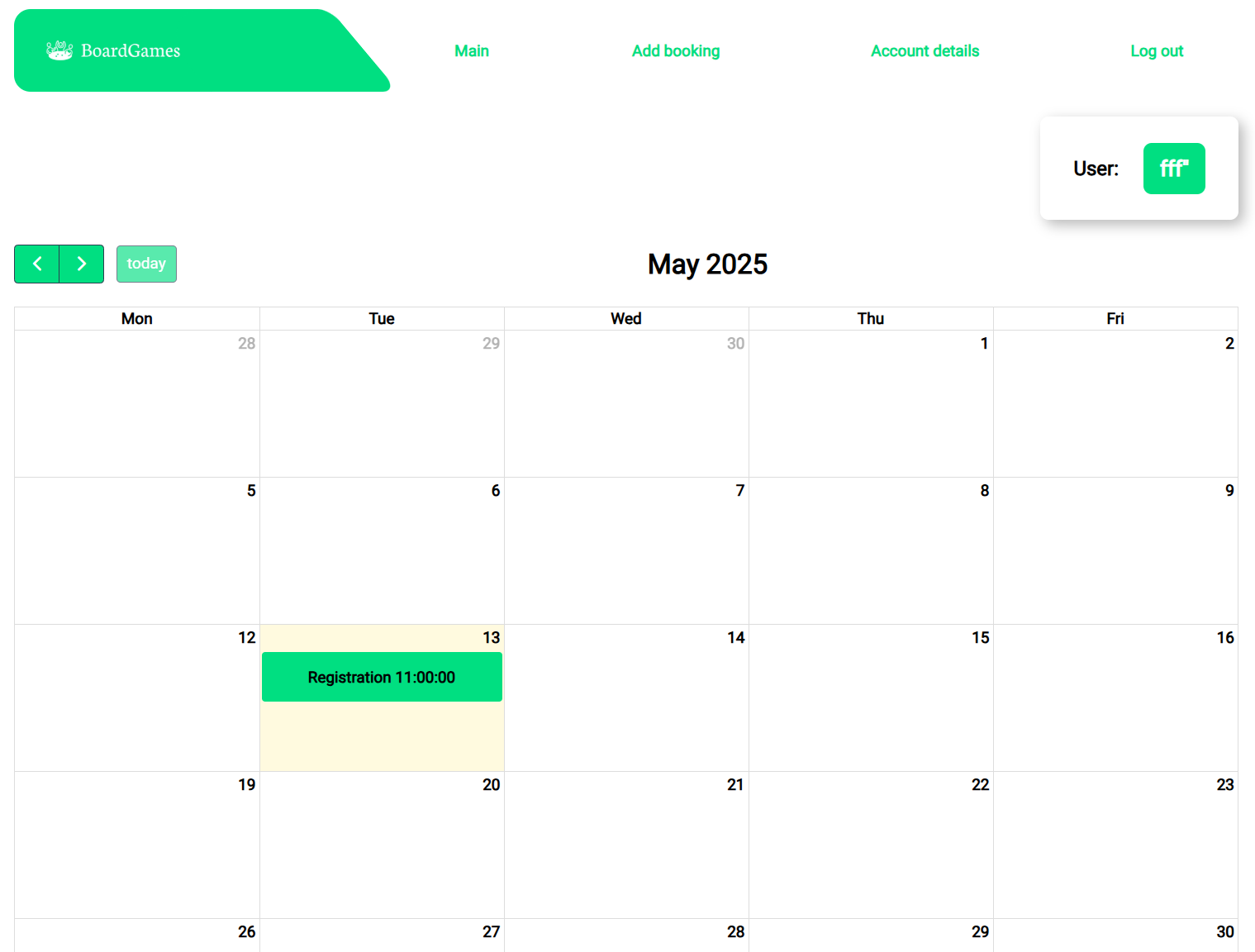


Рис. 5.1.2.1 Main календар

* За потреби — натискає **Add Booking** для створення нового запису.

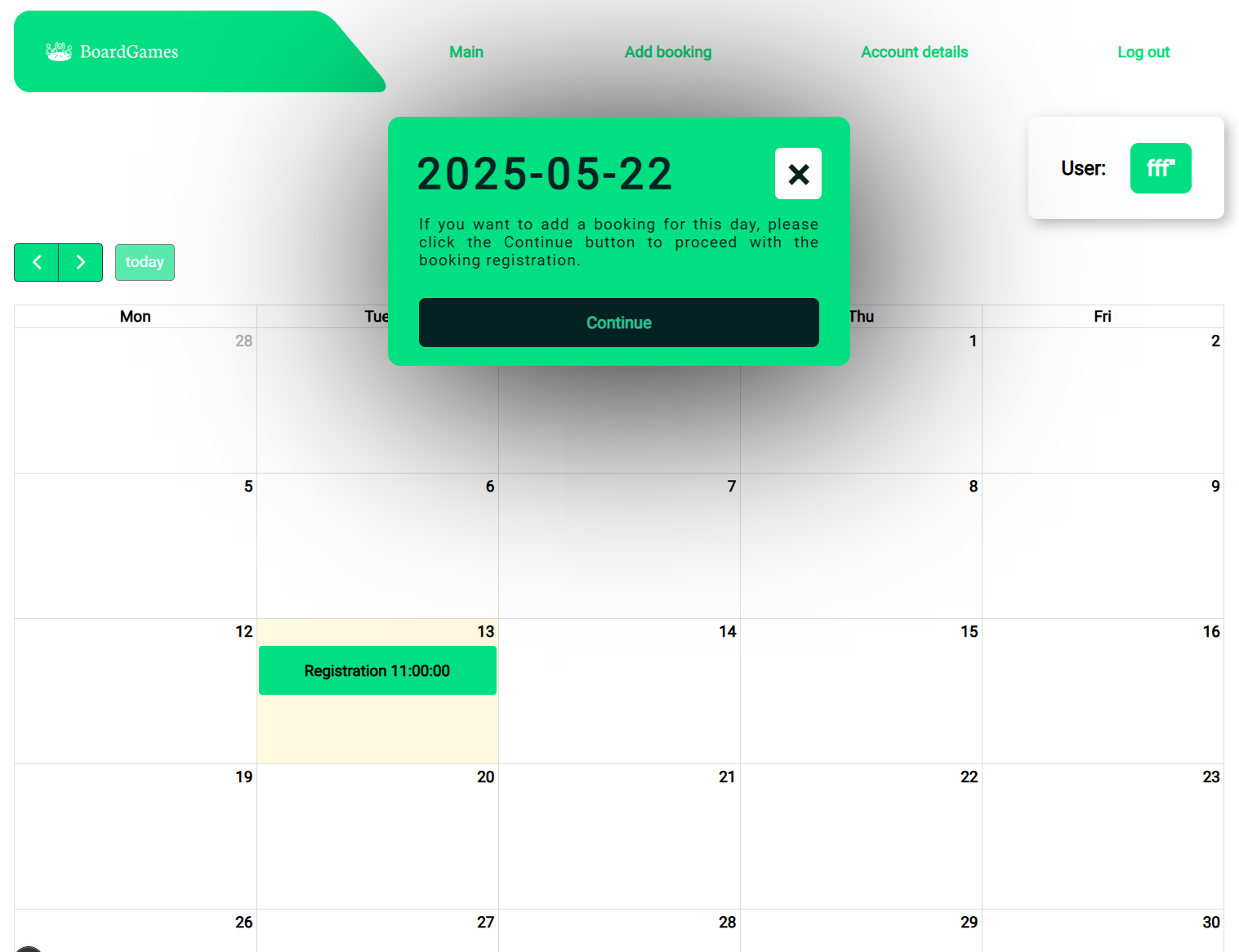


Рис. 5.1.2.2 Main календар(додавання букінгу)

* Вносить дані у форму: дата, час, гра, кількість учасників, список гравців.

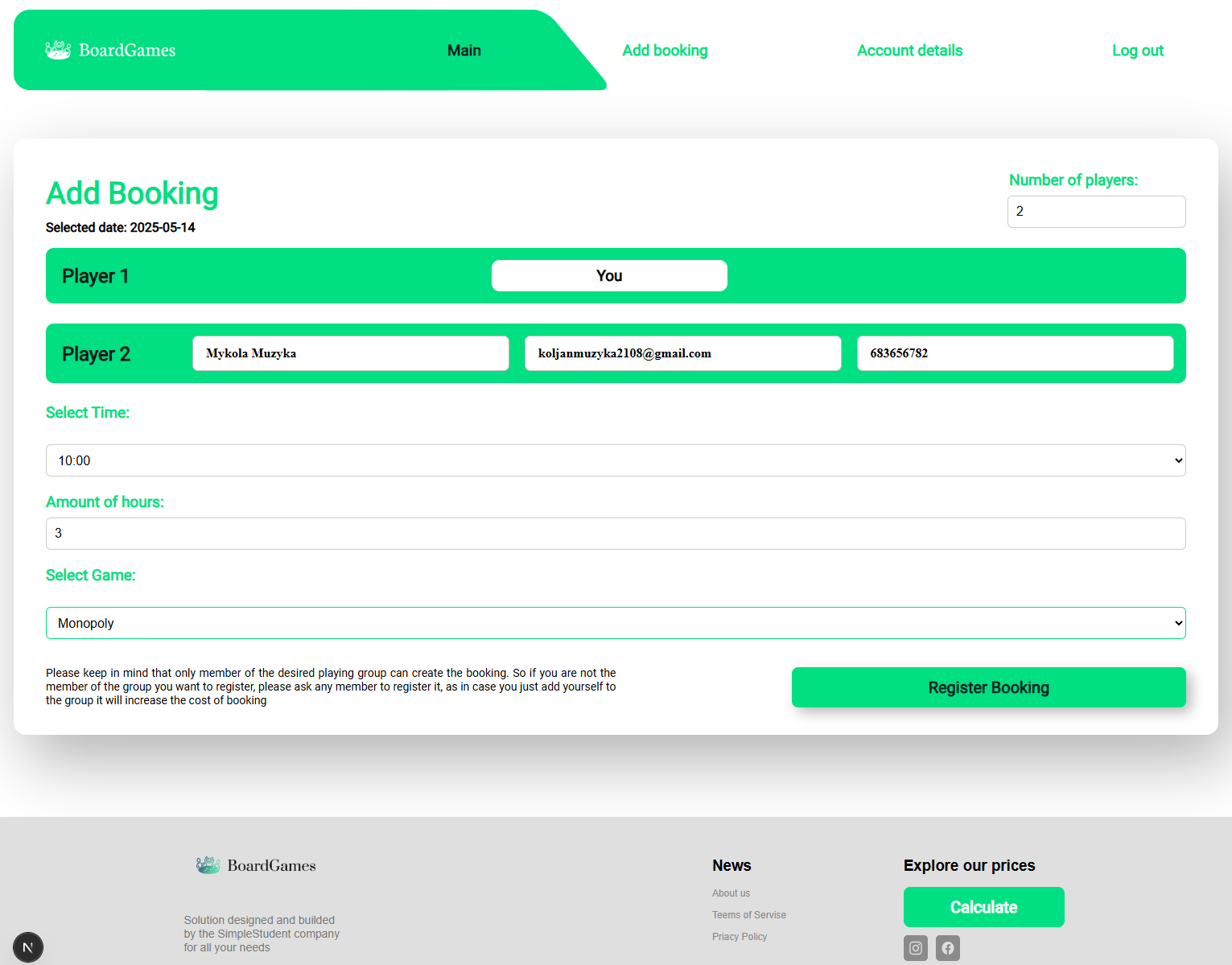
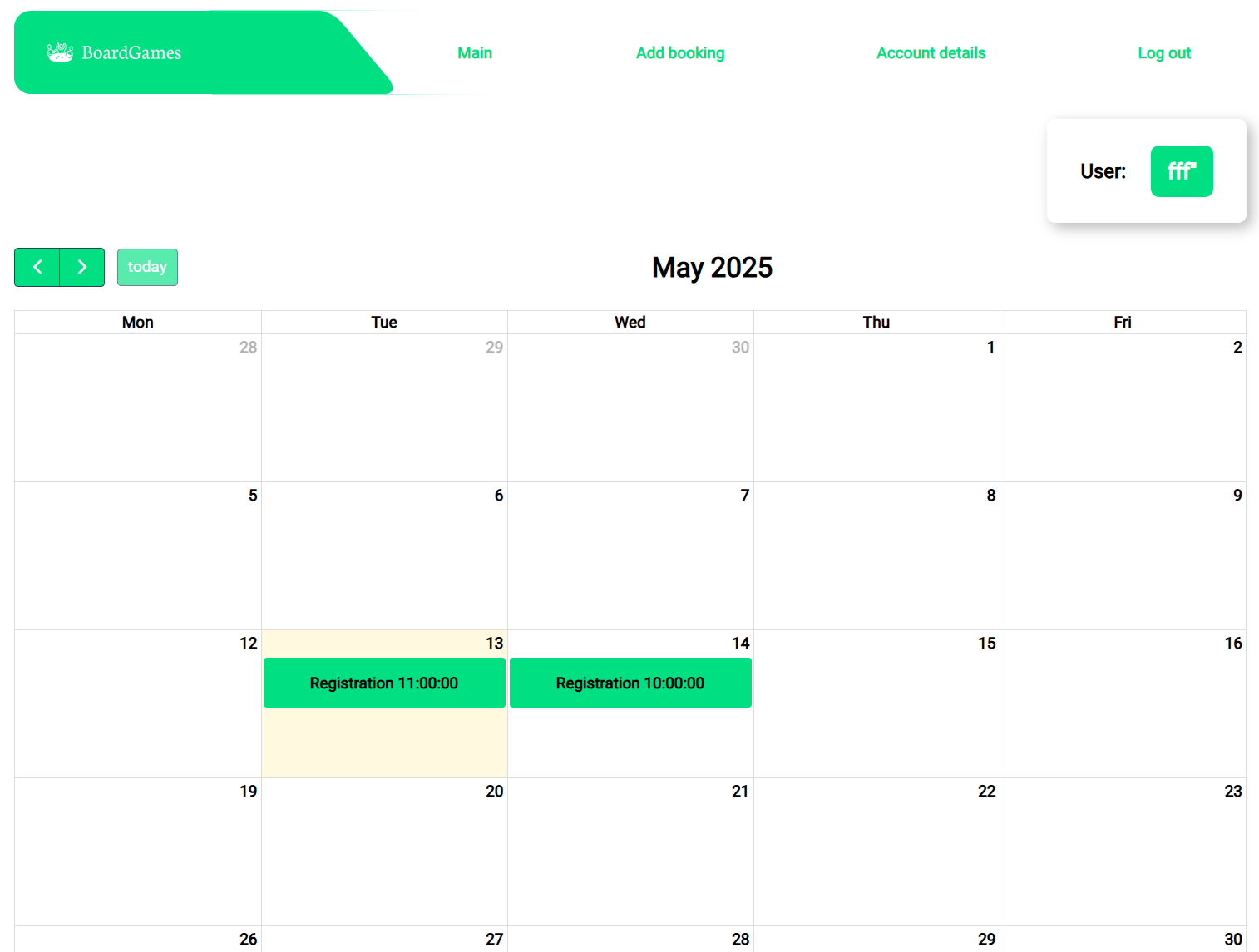


Рис. 5.1.2.3 Форма додавання нового букінгу

* Після підтвердження бронювання — запис зберігається та відображається в інтерфейсі.



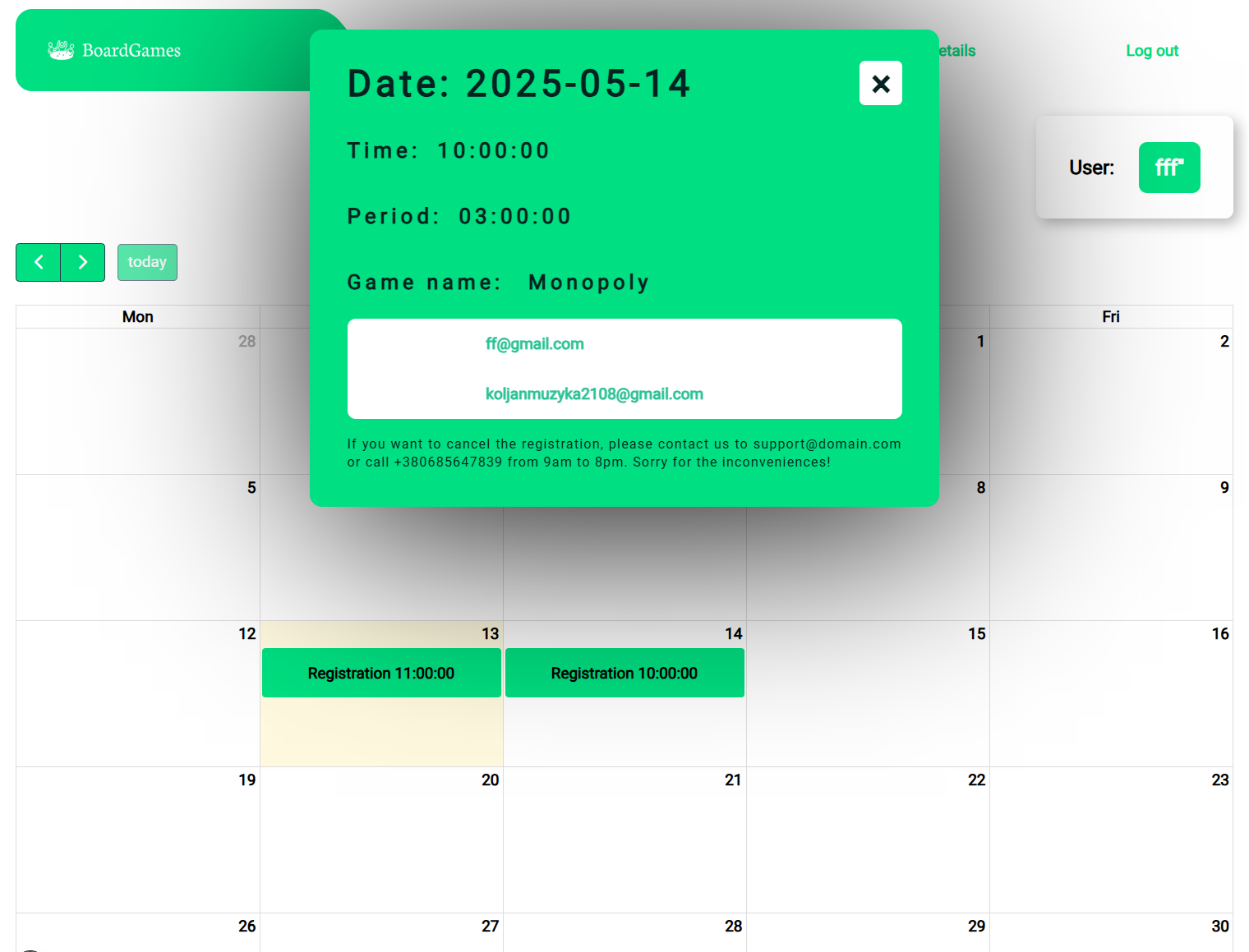


Рис. 5.1.2.4 Новий букінг та його інформація в календарі

* У розділі **Account Details** користувач переглядає або змінює свої особисті дані. Log out користувач може вийти з акаунту.

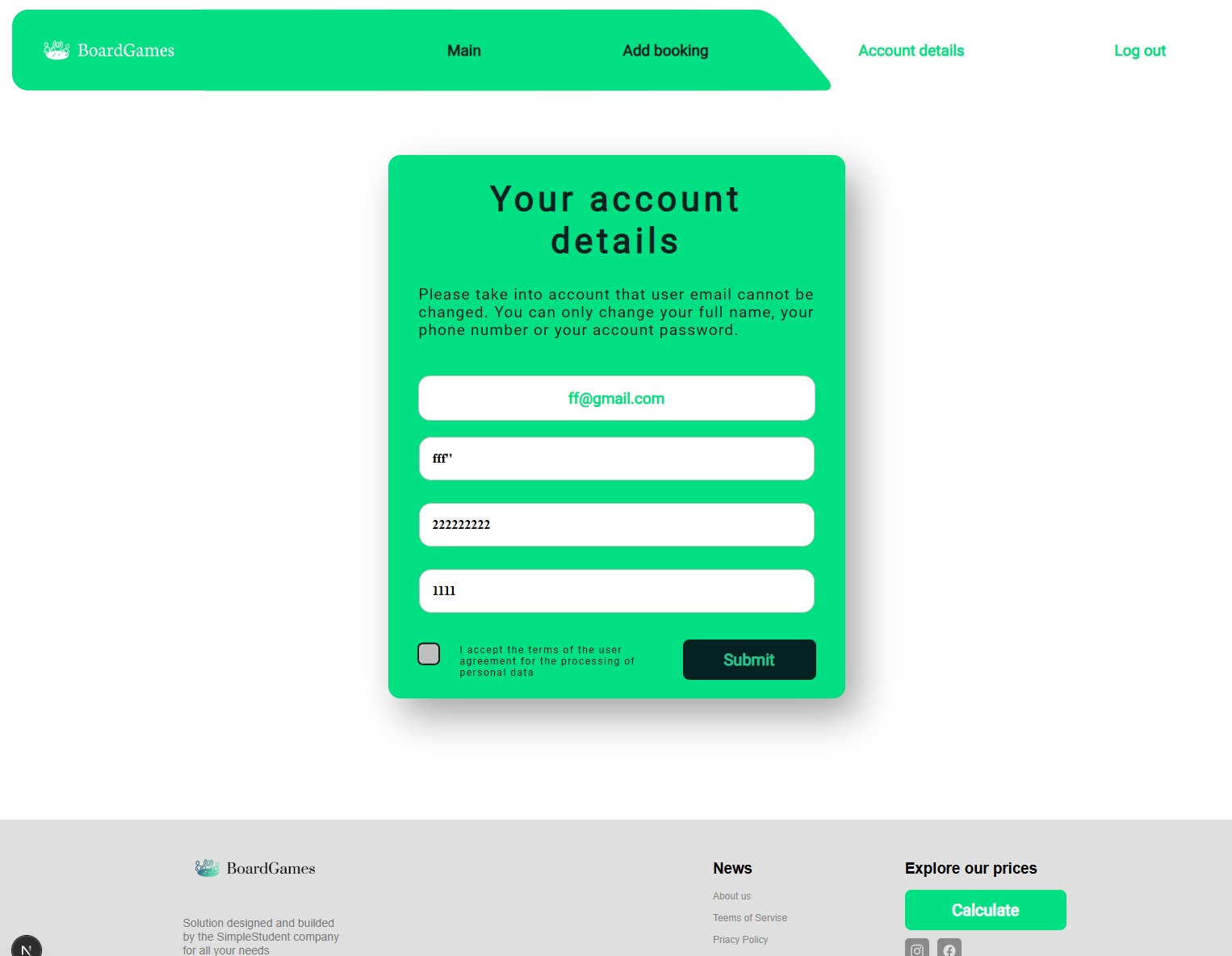


Рис. 5.1.2.5 Деталі акаунту

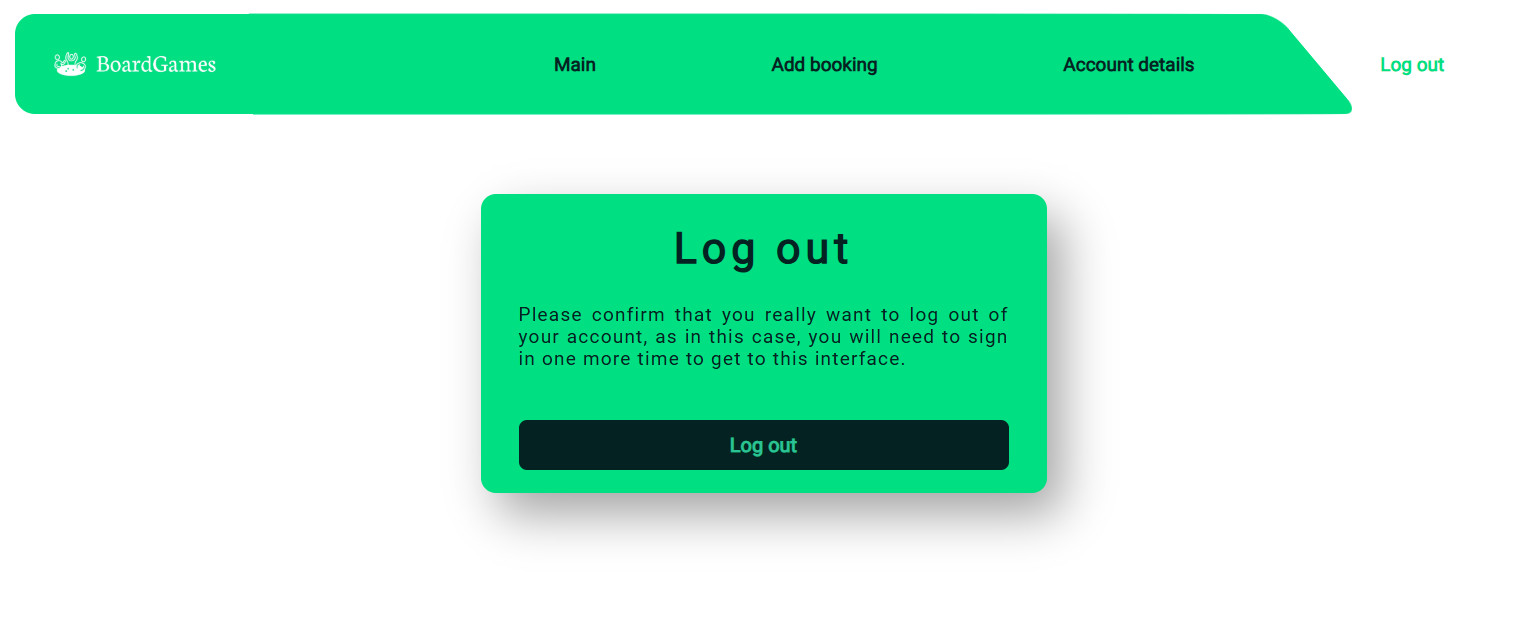


Рис. 5.1.2.6 Вихід із акаунту

Бізнес-процес включає всі етапи самостійної взаємодії клієнта з сервісом без залучення персоналу.

### **5.1.3. Бізнес-процес управління ігровим простором (роль — Робітник)**

Цей процес підтримує щоденну роботу співробітників закладу, пов’язану з координацією бронювань та контролем за станом інвентарю.

**Основні функції:**

* Відображення **Main** — перегляд поточних бронювань по столиках, інформація про найближчі ігри.

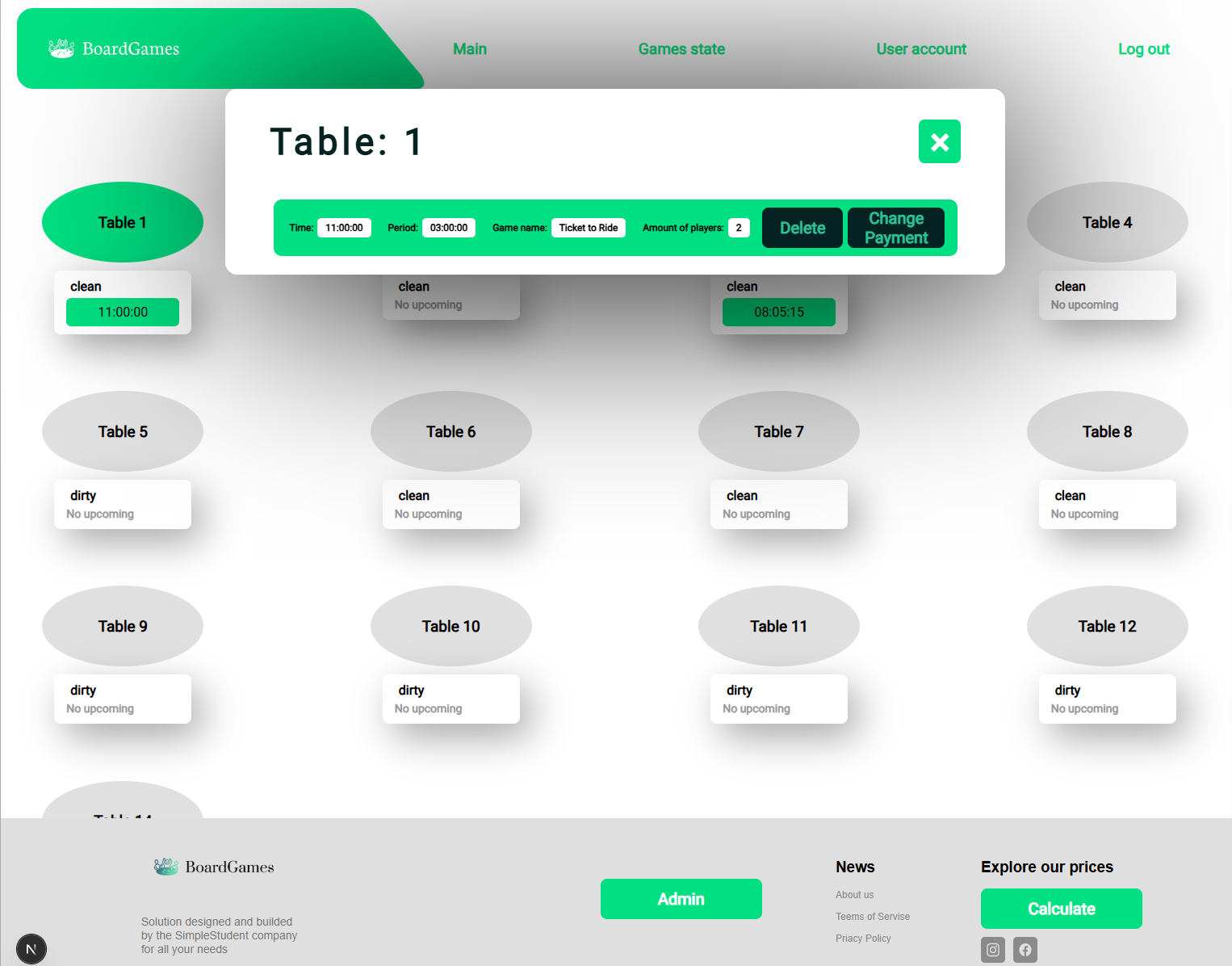


Рис. 5.1.3.1 Схема столів (Main)

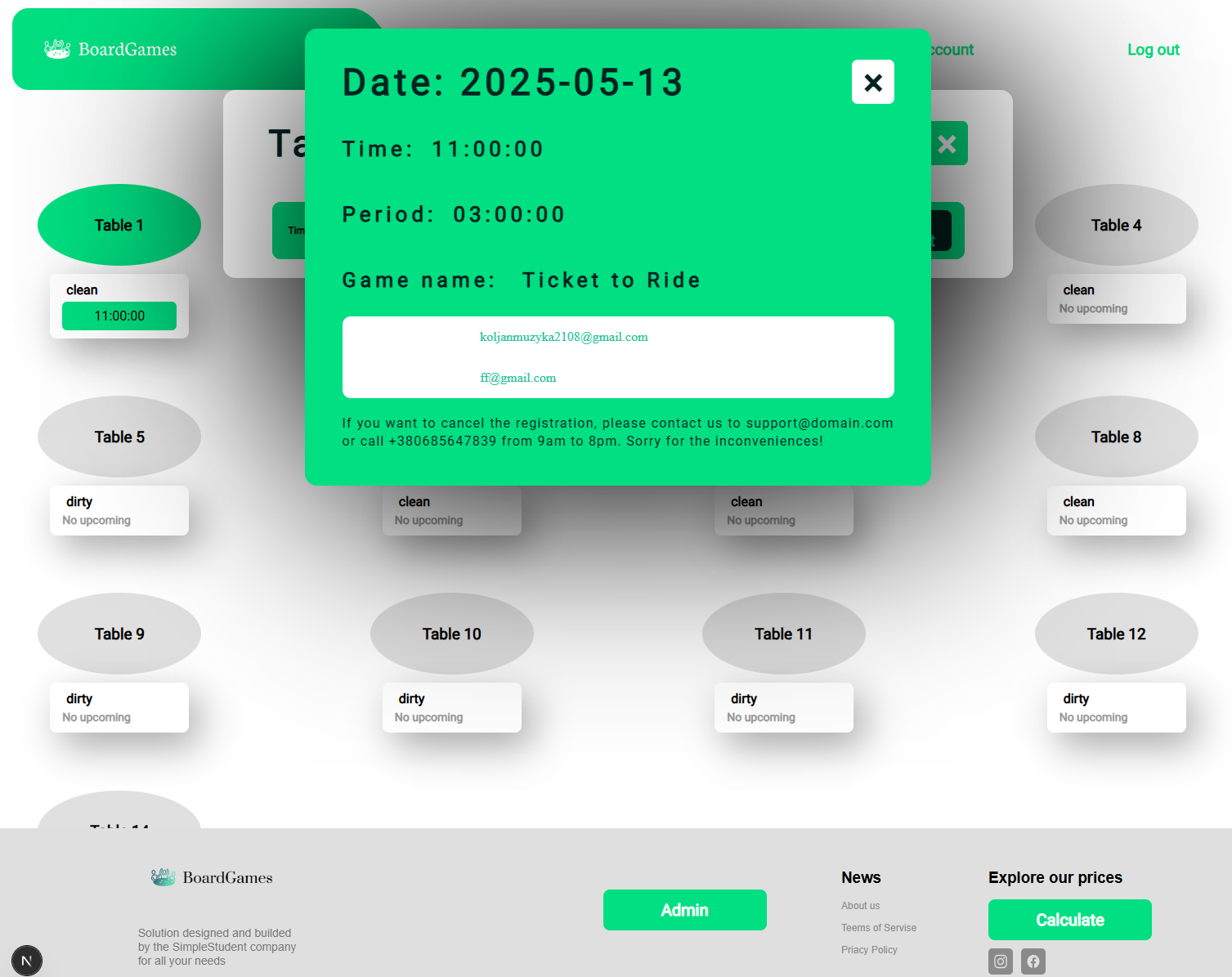


Рис. 5.1.3.2 Схема столів (Main) інформація по бронюванню

* Можливість скасування бронювання або зміни його статусу (наприклад, «оплачено»).

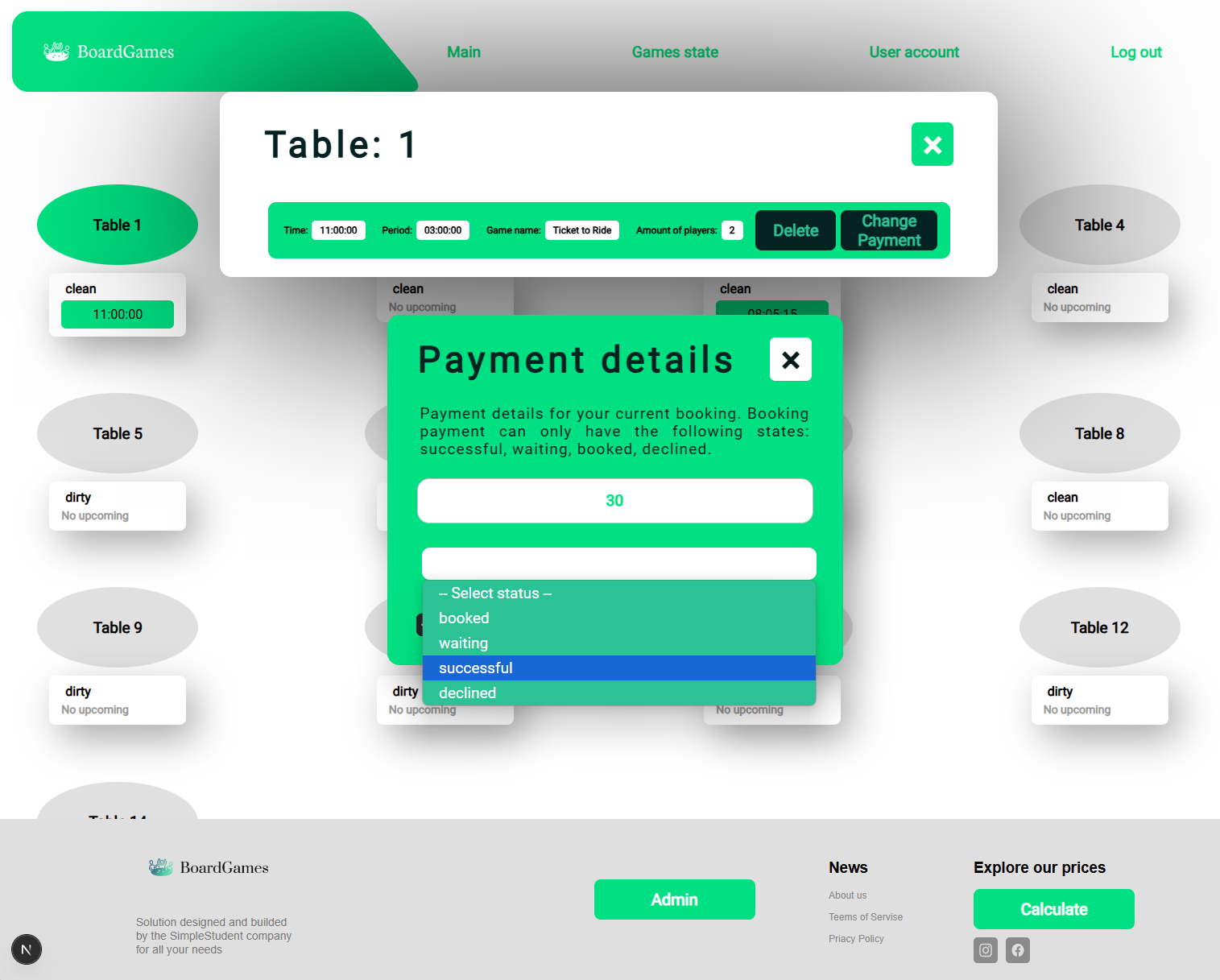


Рис. 5.1.3.2 Схема столів (Main) інформація по бронюванню

* У розділі **Games State** — перегляд ігор, їхнього стану, наявності пошкоджень або зношеності.

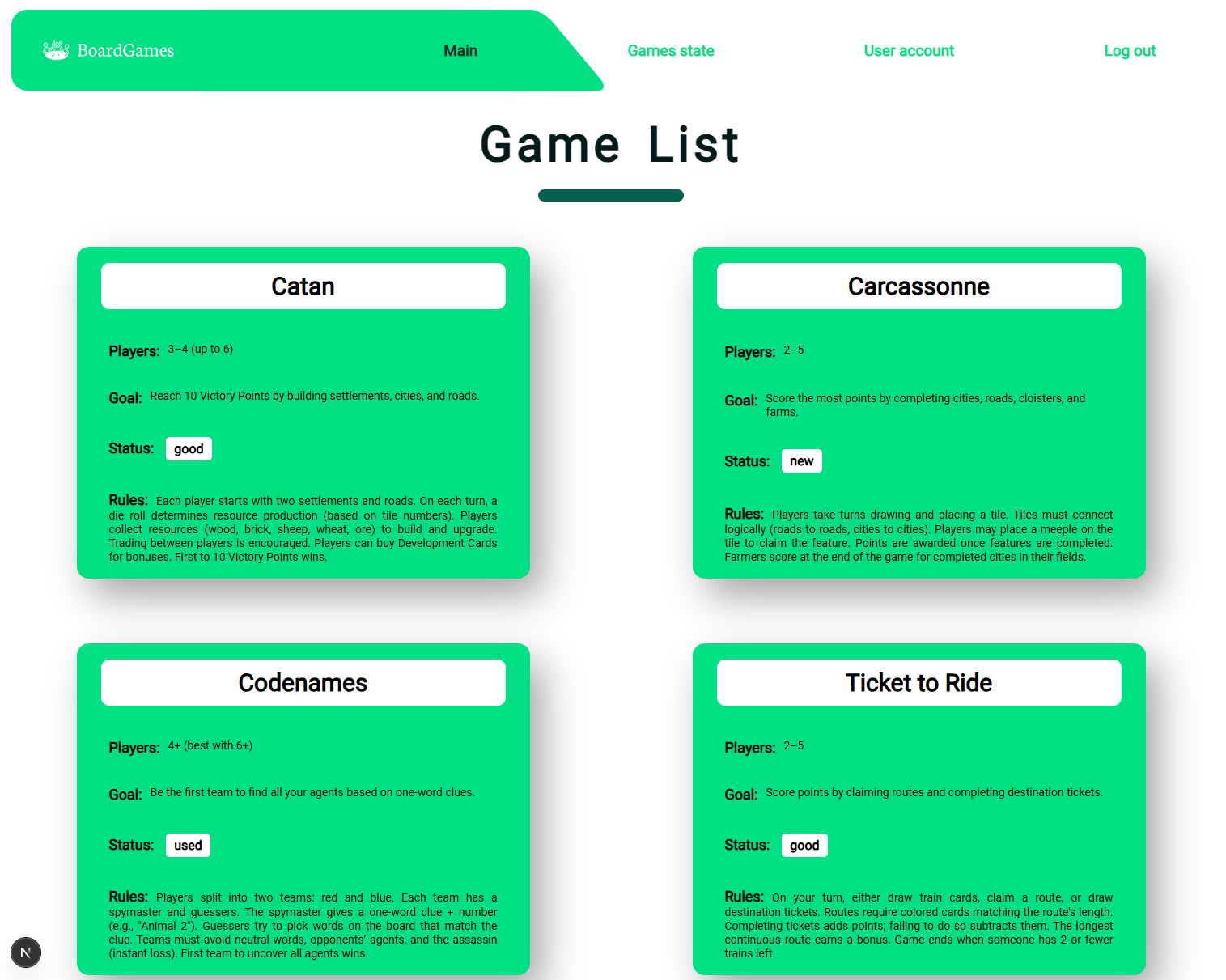


Рис. 5.1.3.3 Список ігор зі статусами

* Перехід до **User Account** для використання функціоналу звичайного користувача (наприклад, для створення власного бронювання).

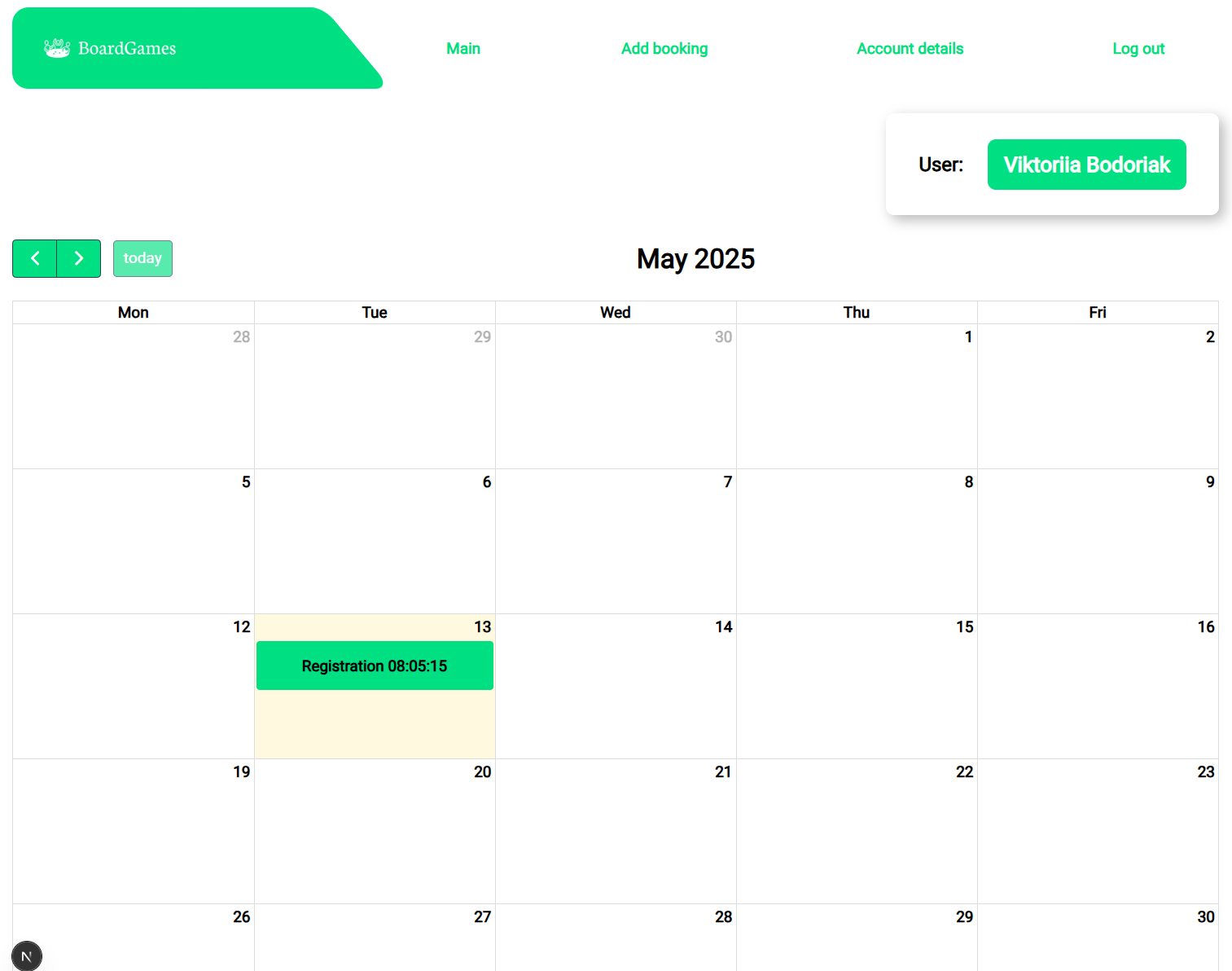


Рис. 5.1.3.4 Головна сторінка Користувача

* **Logout** — завершення зміни та вихід із системи.

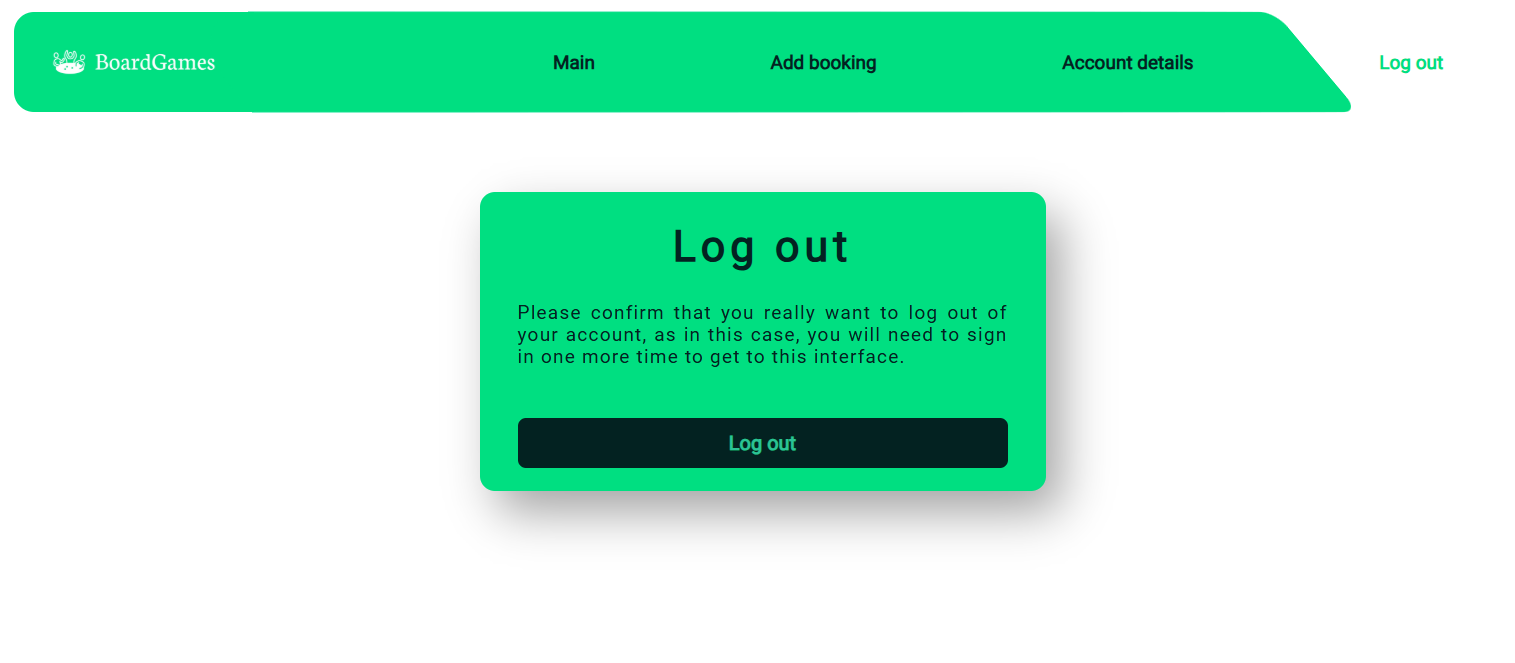


Рис. 5.1.3.5 Вихід із акаунту

Бізнес-процес дозволяє оперативно реагувати на ситуації в закладі та підтримувати порядок у грі та розкладі.

### **5.1.4. Бізнес-процес адміністрування сервісу (роль — Менеджер)**

Менеджер має найширший набір функцій і повний доступ до управління всіма об'єктами системи.

**Основні етапи:**

* **Main** — повний огляд стану столиків, бронювань на день та можливість обрати будь-яку дату.

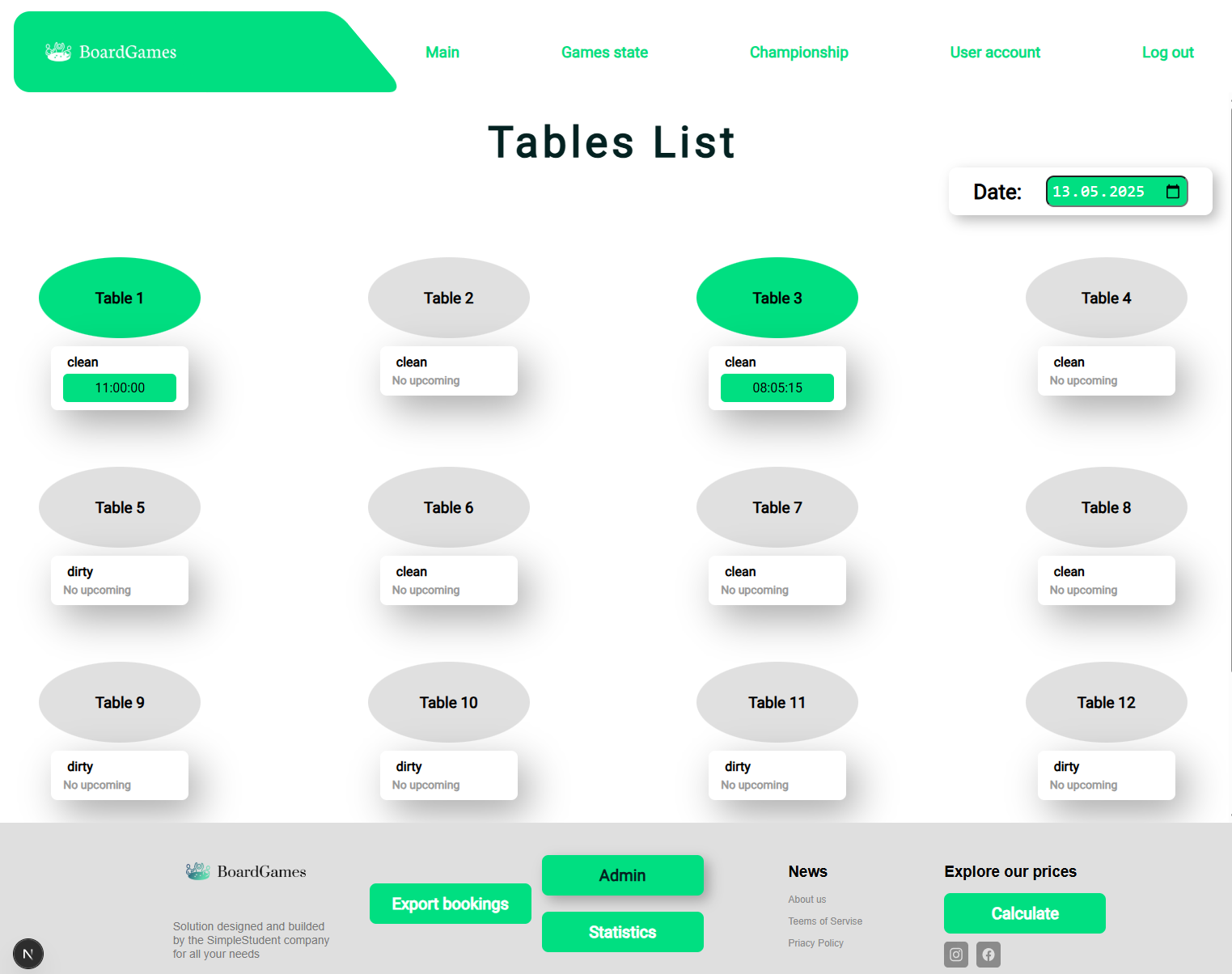
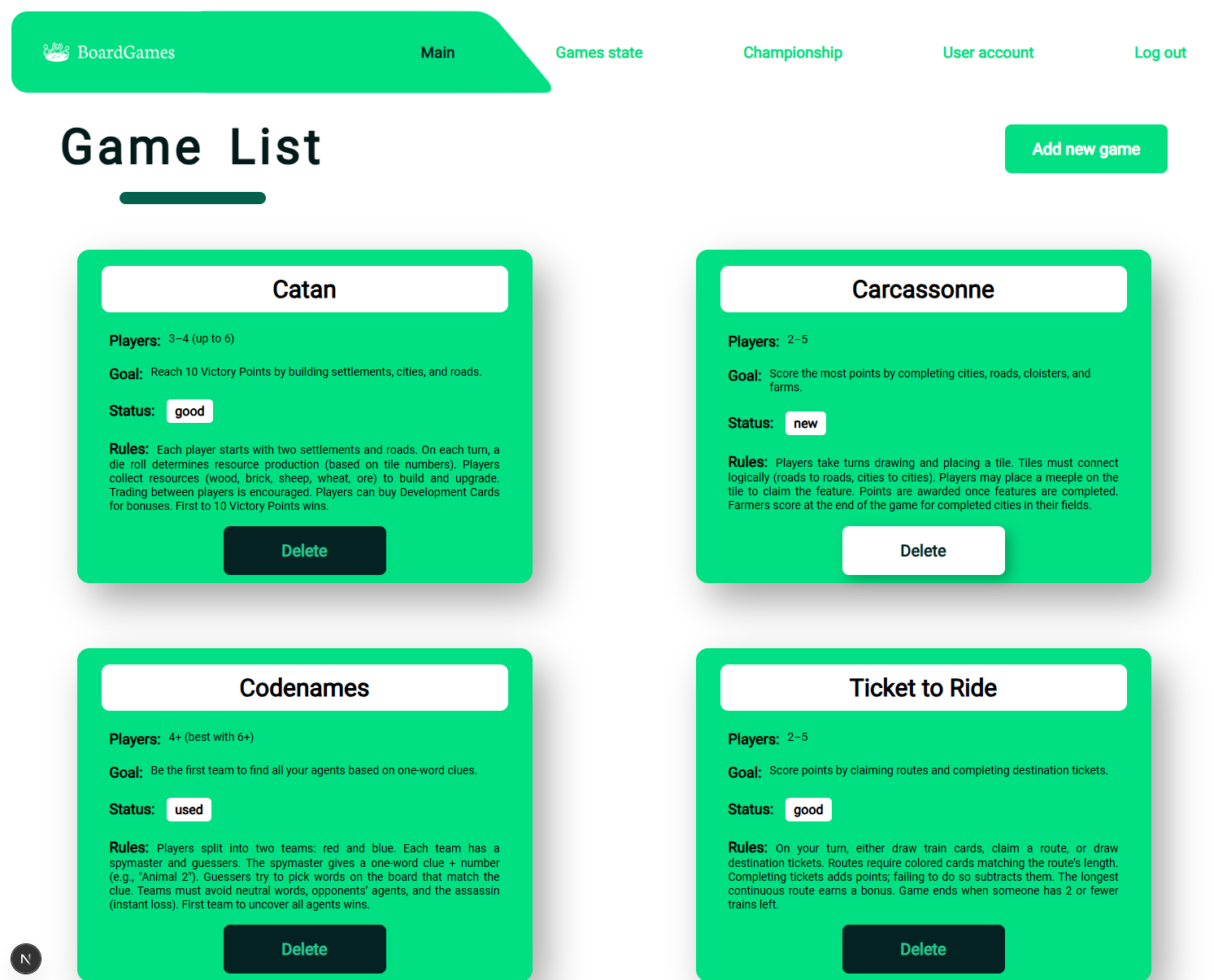
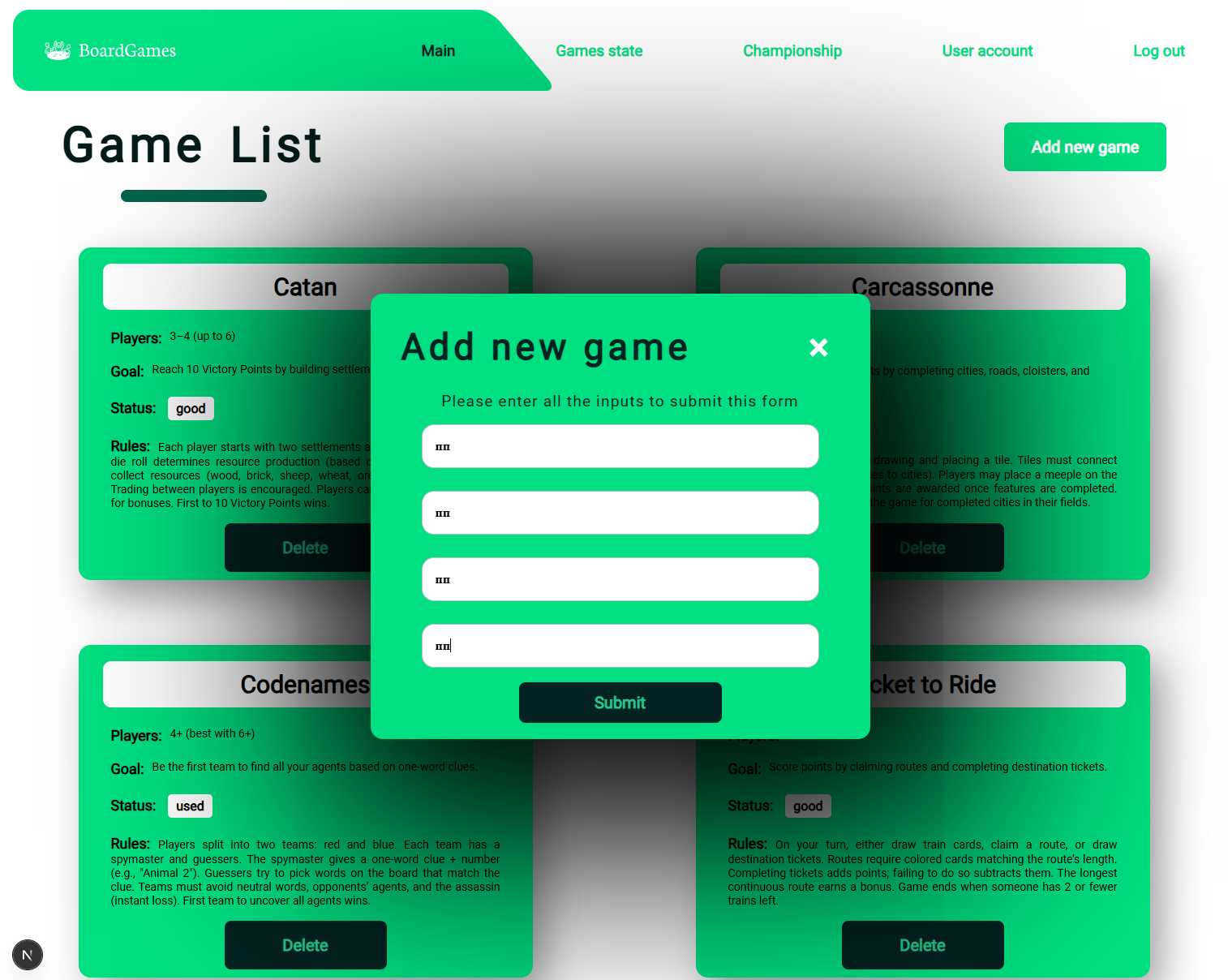
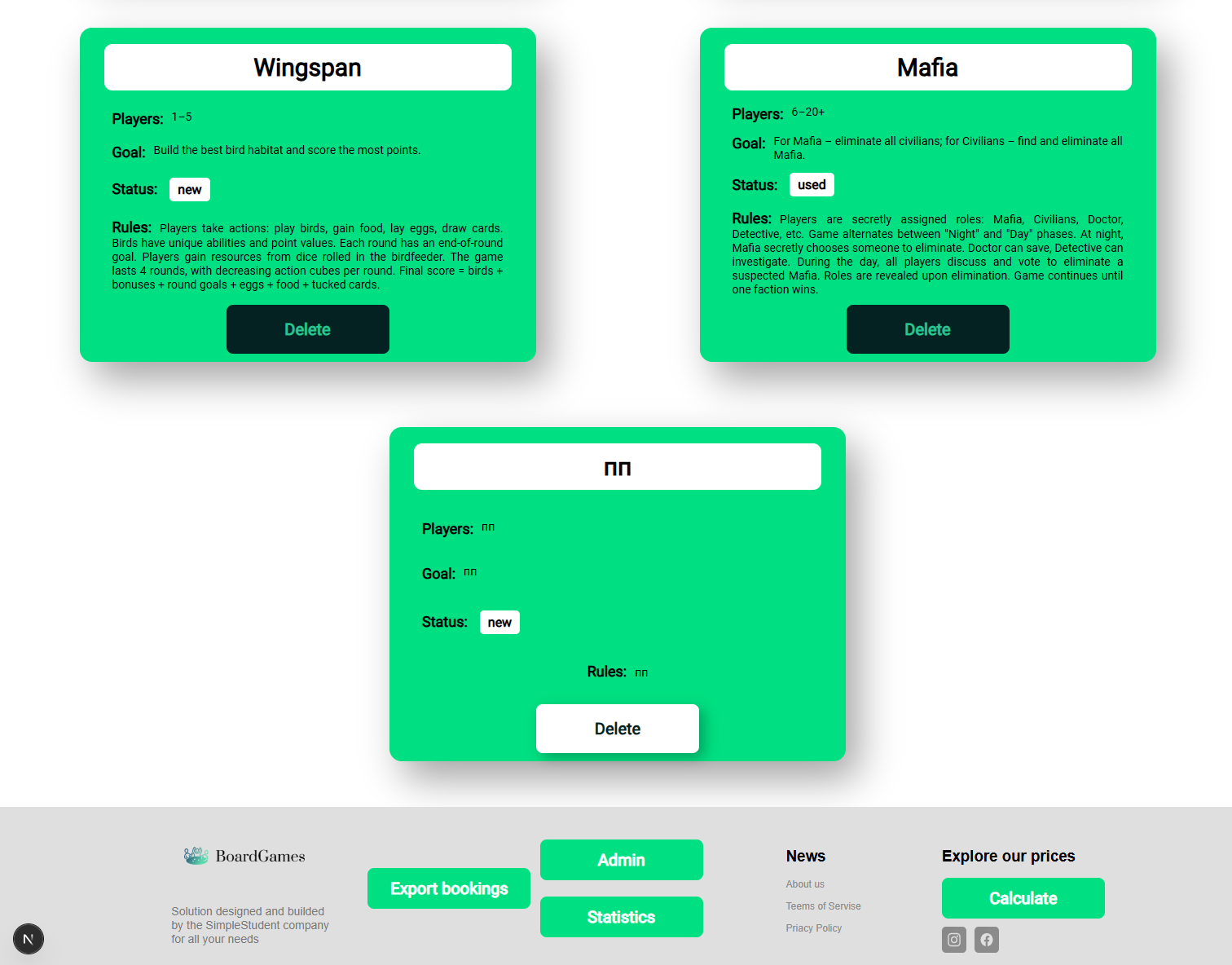


Рис. 5.1.4.1 Схема столів для менеджера з вибором дати

* Керування ігровим інвентарем у розділі **Games State** — додавання нових ігор, редагування, видалення.







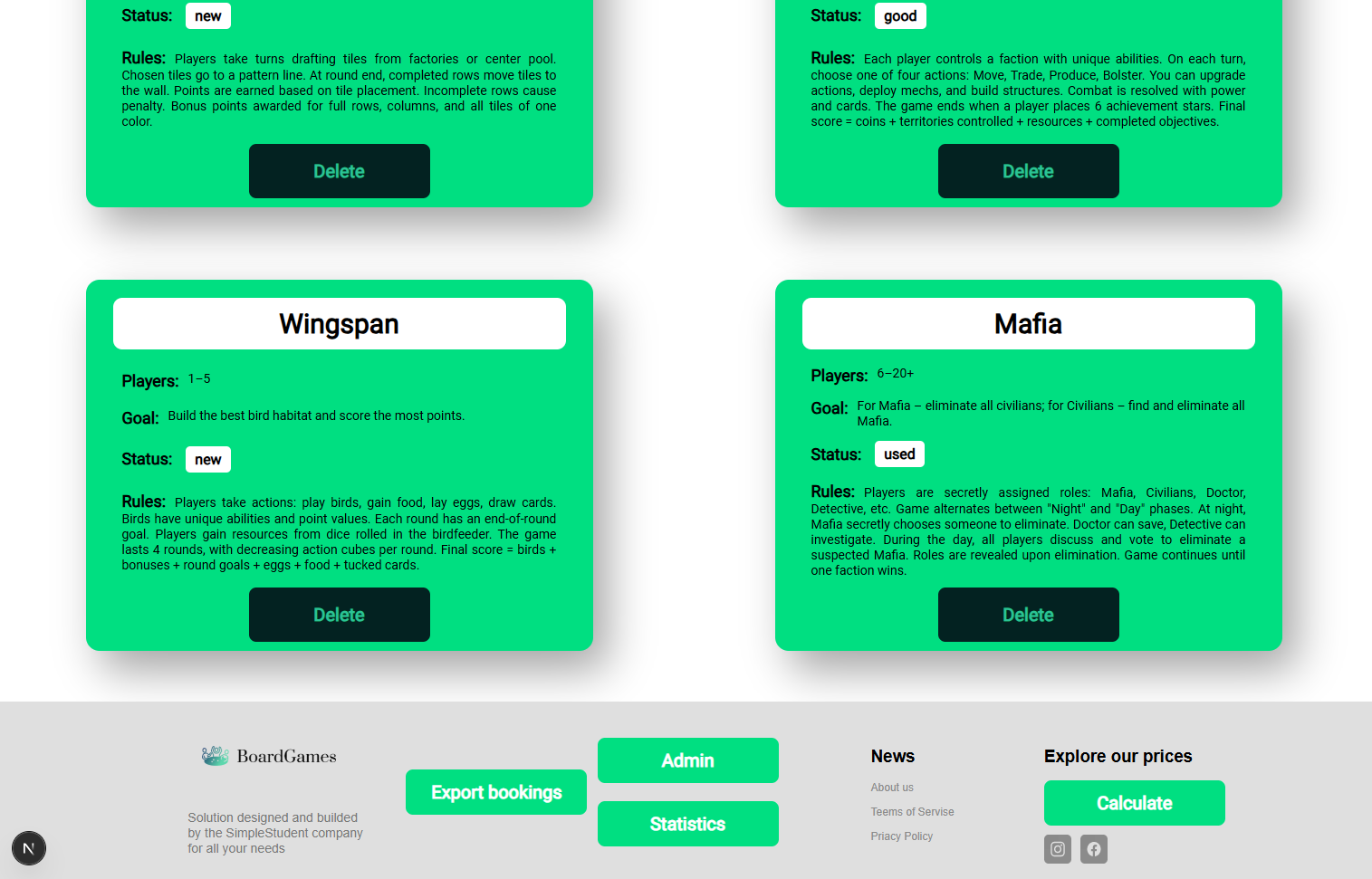


Рис. 5.1.4.2/3/4/5 Операції по додаванню та видаленню ігор зі сторони менеджера

* Робота з турнірами через **Championship** — ведення турнірної таблиці, додавання результатів матчів.

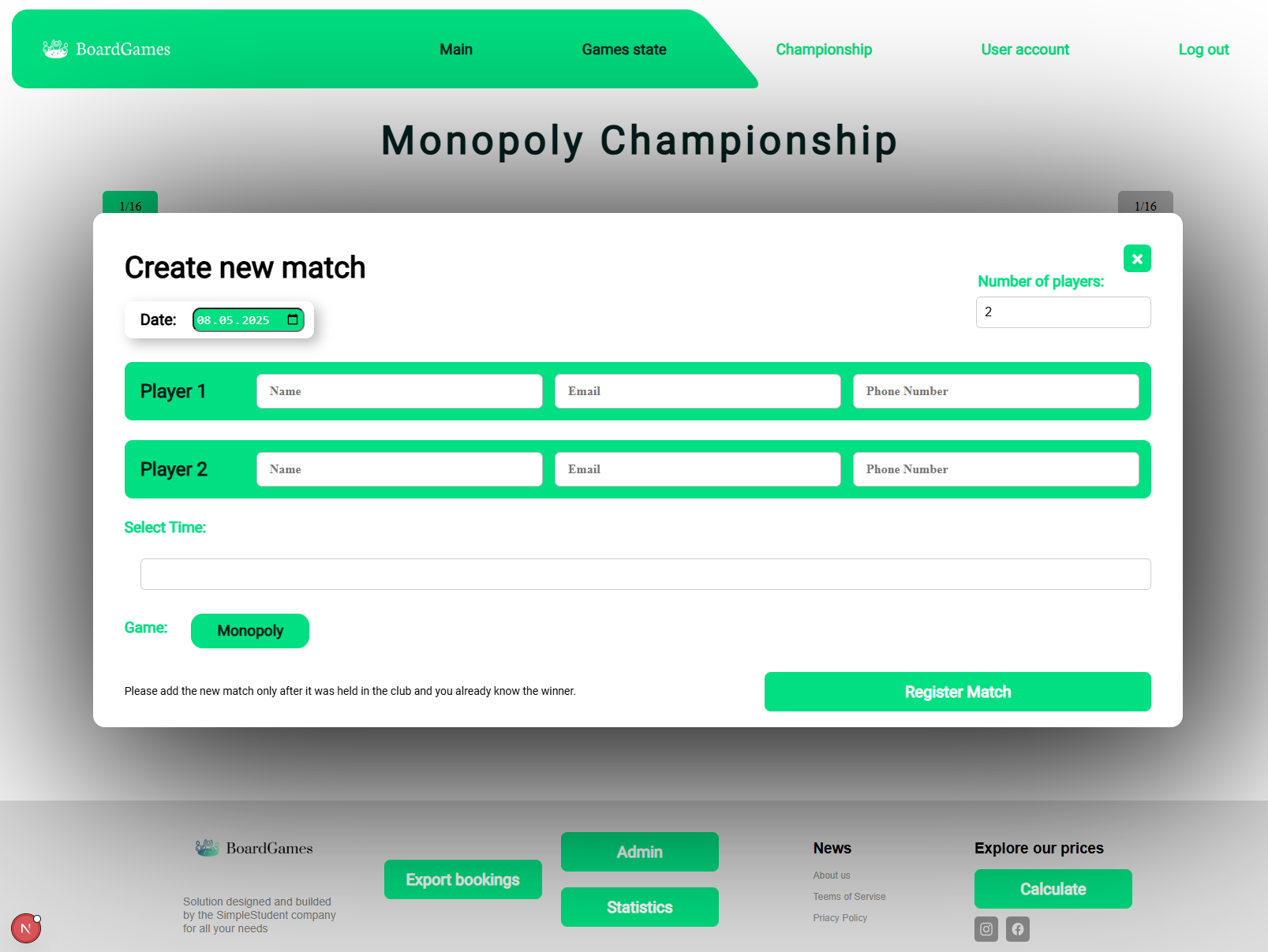
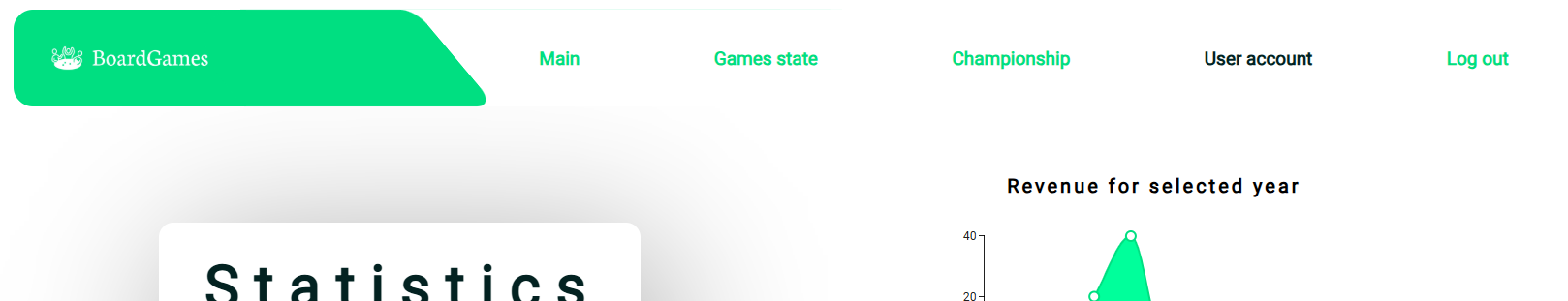


Рис. 5.1.4.5 Форма додавання матчу чемпіонату

* Можливість перемикання до інтерфейсу звичайного користувача або вихід із системи.



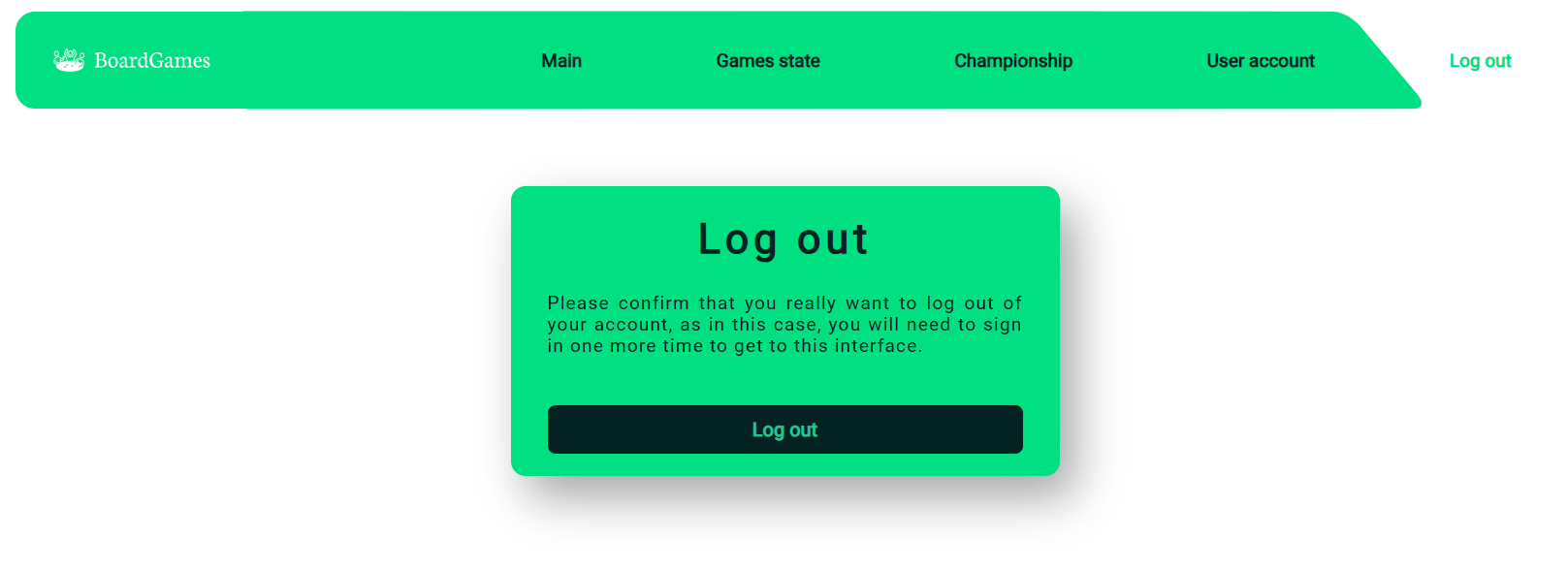


Рис. 5.1.4.6/7 Перехід до юзер акаунту та вихід

Цей бізнес-процес охоплює всі рівні операційного контролю та планування.

**II. Засоби аналітичного представлення даних**

У системі реалізовано візуалізацію ключових аналітичних даних у вигляді графіків та діаграм, доступну менеджеру в розділі Statistics. Сюди входить:

* Дохід по місяцях — графік зростання або зниження прибутку.
* Реєстрації користувачів — динаміка по днях у межах одного місяця.
* Популярні ігри — топ-5 або топ-10 найчастіше обираних ігор.
* Числові метрики — середня тривалість бронювання, середня кількість гравців.

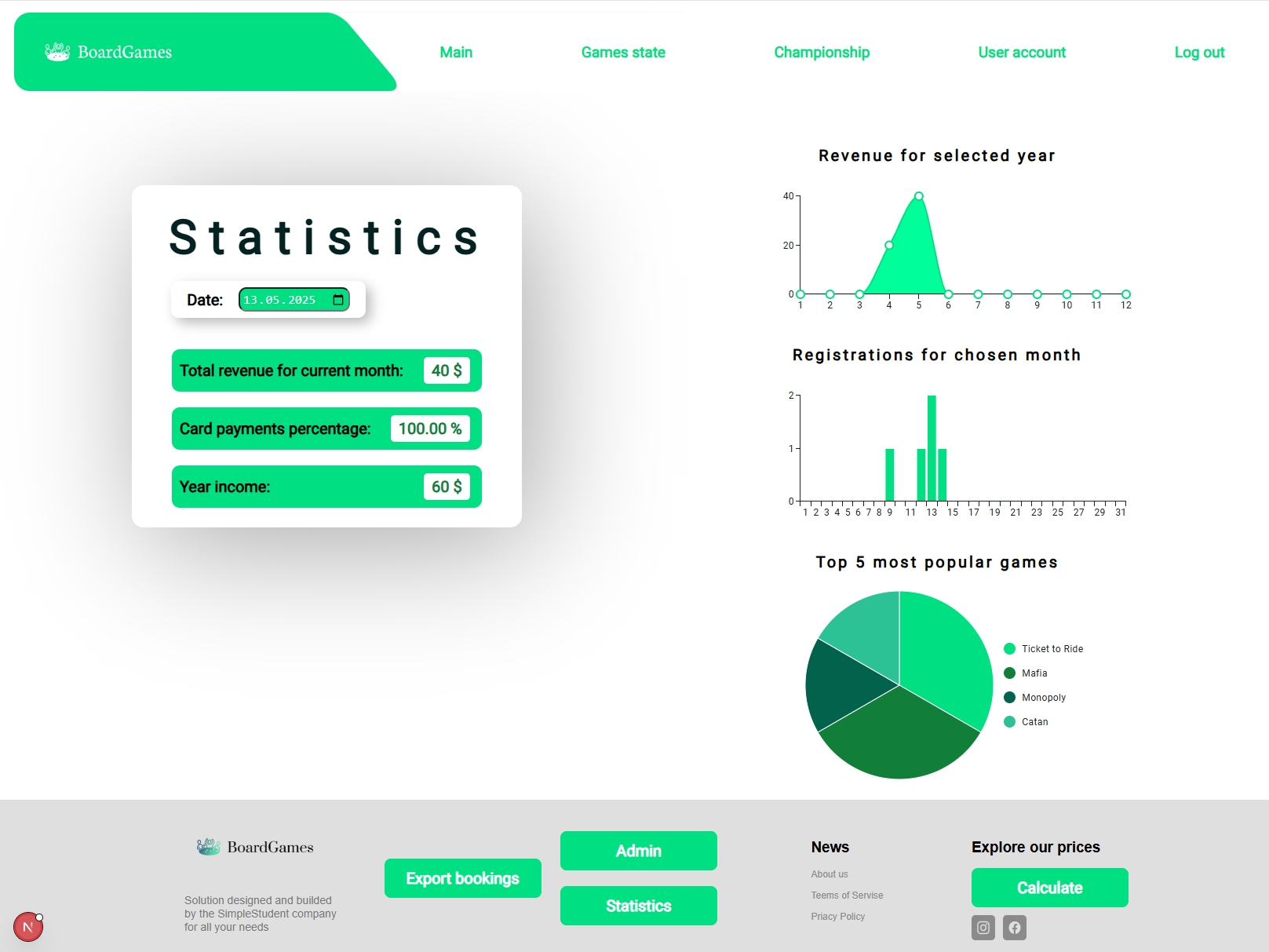


Рис. 5.2.1 Сторінка статистики

**III. Засоби експорту даних**

Менеджер має доступ до функції Export Bookings, яка дозволяє сформувати список усіх бронювань за обраний період. Експорт виконується у вигляді таблиці з такими полями:

* Дата
* Час
* Ім’я користувача
* Гра
* Кількість учасників
* Стан бронювання
* Стан оплати

Формат експорту — CSV або XLSX. Дані можуть бути відкриті в Excel або іншому табличному редакторі. Обмеження щодо кількості записів або фільтрів визначаються на етапі експорту.

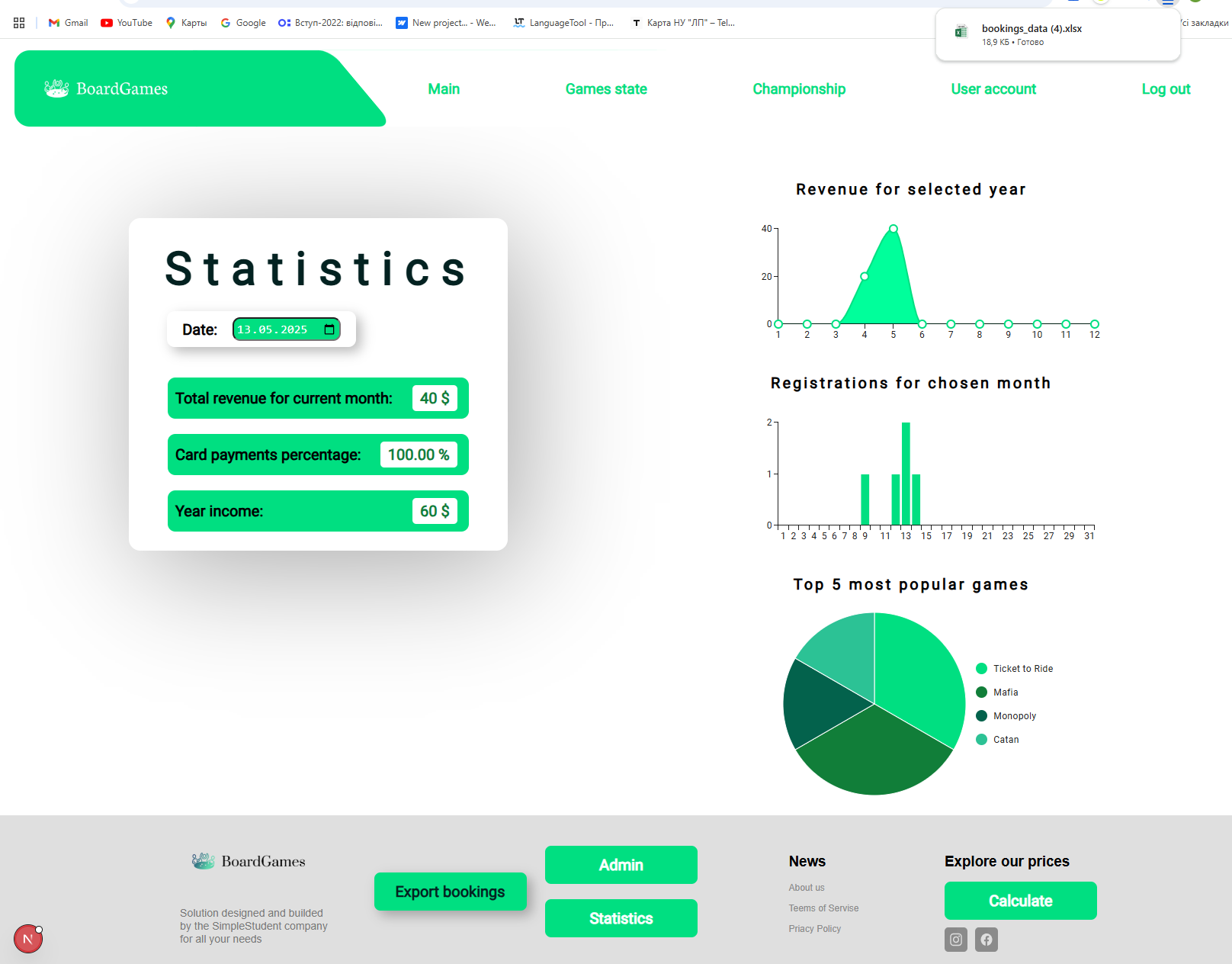


Рис. 5.3.1 Експорт даних

**IV. Особливості додаткових реалізованих функцій**

У системі реалізовано низку додаткових можливостей:

* Калькулятор ціни для гостей — не потребує авторизації.
* Чемпіонати — не тільки відображення, а й активне ведення матчів у менеджерському інтерфейсі.
* Ігровий інвентар — статус ігор може бути "активна", "зламано", "відсутня".
* Зміна статусу оплати — робітник або менеджер може вручну фіксувати оплату.

**Список літератури**

1. PostgreSQL Documentation (official): <https://www.postgresql.org/docs/current/>
2. SQL Style Guide by Simon Holywell: <https://www.sqlstyle.guide/>
3. Node.js Official Documentation: <https://nodejs.org/docs/latest/api/>
4. React Official Documentation: <https://reactjs.org/docs/getting-started.html>
5. Next.js Documentation (by Vercel): <https://nextjs.org/docs>

**Висновки**

**I. Встановлені недоліки**

У ході розробки інформаційної системи для автоматизації процесів бронювання настільних ігор у клубах було виявлено кілька недоліків:

Обмежена гнучкість налаштувань ролей користувачів: Роль користувача в системі є досить обмеженою. Хоча система включає основні ролі, такі як відвідувач, менеджер, робітник та зареєстрований користувач, не вистачає більш специфічних ролей, таких як аналітик або модератор, що дозволило б більш детально налаштувати доступ і функціональність.

Обмежена кількість доступних столів та ігор: Зараз система дозволяє бронювати лише обмежену кількість столів та ігор, що може обмежувати можливості для великих клубів або тих, хто працює з великою кількістю користувачів. Це може створити труднощі для забезпечення плавного процесу бронювання в пікові періоди.

Відсутність більш детального аналізу статистики: Наразі статистика в системі зводиться до базових параметрів, таких як кількість бронювань і найпопулярніші ігри, але відсутні глибші аналітичні інструменти, які дозволяють краще оцінити ефективність використання ресурсів (наприклад, час, коли ігри найбільш популярні, або середнє навантаження на столи).

Обмежене управління інвентарем: Функціонал управління фізичним станом інвентарю (наприклад, столів чи ігор) потребує вдосконалення. Поки що система дозволяє лише позначати статус ігор і столів як доступний або не доступний, але не включає механізмів для точного контролю за зношенням або необхідністю ремонту.

**II. Перспективи покращення**

Щоб підвищити ефективність і універсальність системи, доцільно впровадити такі покращення:

Розширення ролей користувачів та налаштувань доступу: Потрібно створити додаткові ролі, такі як аналітик чи модератор, з чітким визначенням прав доступу для кожної ролі. Це дозволить краще налаштувати доступ до окремих модулів системи і розширить можливості адміністрування.

Додавання підтримки більшої кількості столів та ігор: Розширення доступних столів та ігор дозволить залучити більше користувачів, зокрема в пікові періоди, та покращити ефективність використання інвентарю. Це також включає підтримку варіативності ігор залежно від кількості учасників.

Розширення аналітики та статистики: Доцільно вдосконалити статистику і додати нові аналітичні інструменти, такі як аналіз популярності ігор по годинах, визначення пікових періодів, а також середнє навантаження на столи та типи ігор, що використовуються. Це дозволить краще планувати ресурси та прогнозувати попит.

Покращення управління інвентарем: Важливо додати більш детальну інформацію про стан інвентарю, зокрема механізм для позначення зношення або необхідності ремонту ігор і столів. Крім того, варто реалізувати можливість управління статусом ігор (наприклад, через систему для перевірки наявності в інвентарі та очищення від пошкоджених екземплярів).

Вдосконалення інтерфейсу користувача: Важливо покращити інтерфейс користувача, зробивши його більш інтуїтивно зрозумілим та зручним для всіх ролей. Додавання функцій персоналізації профілю, можливість налаштування сповіщень, багатомовність та адаптація для мобільних пристроїв підвищать комфорт користувачів і зручність користування системою.

**Додатки**

**I. Скрипт створення бази даних**

-- Table: public.account

-- DROP TABLE IF EXISTS public.account;

CREATE TABLE IF NOT EXISTS public.account

(

account\_id integer NOT NULL GENERATED BY DEFAULT AS IDENTITY ( INCREMENT 1 START 1 MINVALUE 1 MAXVALUE 2147483647 CACHE 1 ),

password character varying(100) COLLATE pg\_catalog."default" NOT NULL,

player\_email character varying(50) COLLATE pg\_catalog."default" NOT NULL,

CONSTRAINT account\_pkey PRIMARY KEY (account\_id),

CONSTRAINT player\_email\_fk\_constr FOREIGN KEY (player\_email)

REFERENCES public.player (email) MATCH SIMPLE

ON UPDATE RESTRICT

ON DELETE RESTRICT

NOT VALID

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public.account

OWNER to postgres;

REVOKE ALL ON TABLE public.account FROM admin\_role;

REVOKE ALL ON TABLE public.account FROM user\_role;

REVOKE ALL ON TABLE public.account FROM visitor\_role;

GRANT DELETE, INSERT, SELECT, UPDATE ON TABLE public.account TO admin\_role;

GRANT ALL ON TABLE public.account TO manager\_role;

GRANT ALL ON TABLE public.account TO postgres;

GRANT UPDATE, INSERT, SELECT ON TABLE public.account TO user\_role;

GRANT SELECT, INSERT ON TABLE public.account TO visitor\_role;

-- Table: public.admin

-- DROP TABLE IF EXISTS public.admin;

CREATE TABLE IF NOT EXISTS public.admin

(

access\_type character varying COLLATE pg\_catalog."default" NOT NULL,

admin\_id integer NOT NULL,

CONSTRAINT admin\_fk FOREIGN KEY (admin\_id)

REFERENCES public.account (account\_id) MATCH SIMPLE

ON UPDATE RESTRICT

ON DELETE RESTRICT

NOT VALID,

CONSTRAINT access\_type\_check CHECK (access\_type::text = ANY (ARRAY['employee'::character varying, 'manager'::character varying]::text[]))

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public.admin

OWNER to postgres;

REVOKE ALL ON TABLE public.admin FROM admin\_role;

REVOKE ALL ON TABLE public.admin FROM visitor\_role;

GRANT SELECT ON TABLE public.admin TO admin\_role;

GRANT ALL ON TABLE public.admin TO manager\_role;

GRANT ALL ON TABLE public.admin TO postgres;

GRANT SELECT ON TABLE public.admin TO visitor\_role;

-- Table: public.booking

-- DROP TABLE IF EXISTS public.booking;

CREATE TABLE IF NOT EXISTS public.booking

(

booking\_id integer NOT NULL GENERATED BY DEFAULT AS IDENTITY ( INCREMENT 1 START 1 MINVALUE 1 MAXVALUE 2147483647 CACHE 1 ),

group\_id integer NOT NULL,

table\_id integer NOT NULL,

game\_id integer NOT NULL,

payment\_id integer NOT NULL,

date date NOT NULL,

"time" time without time zone NOT NULL,

period time without time zone NOT NULL,

CONSTRAINT booking\_pkey PRIMARY KEY (booking\_id),

CONSTRAINT book\_team\_fk\_const FOREIGN KEY (group\_id)

REFERENCES public."group" (group\_id) MATCH SIMPLE

ON UPDATE RESTRICT

ON DELETE CASCADE,

CONSTRAINT game\_fk\_constr FOREIGN KEY (game\_id)

REFERENCES public.game (game\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION,

CONSTRAINT payment\_fk\_constr FOREIGN KEY (payment\_id)

REFERENCES public.payment (payment\_id) MATCH SIMPLE

ON UPDATE RESTRICT

ON DELETE CASCADE,

CONSTRAINT table\_fk FOREIGN KEY (table\_id)

REFERENCES public."table" (table\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public.booking

OWNER to postgres;

REVOKE ALL ON TABLE public.booking FROM admin\_role;

REVOKE ALL ON TABLE public.booking FROM user\_role;

GRANT SELECT, UPDATE, INSERT, DELETE ON TABLE public.booking TO admin\_role;

GRANT ALL ON TABLE public.booking TO manager\_role;

GRANT ALL ON TABLE public.booking TO postgres;

GRANT INSERT, SELECT ON TABLE public.booking TO user\_role;

-- Index: idx\_booking\_date

-- DROP INDEX IF EXISTS public.idx\_booking\_date;

CREATE INDEX IF NOT EXISTS idx\_booking\_date

ON public.booking USING btree

(date ASC NULLS LAST)

TABLESPACE pg\_default;

-- Index: idx\_booking\_group\_id

-- DROP INDEX IF EXISTS public.idx\_booking\_group\_id;

CREATE INDEX IF NOT EXISTS idx\_booking\_group\_id

ON public.booking USING btree

(group\_id ASC NULLS LAST)

TABLESPACE pg\_default;

-- Trigger: trg\_check\_booking\_conflicts

-- DROP TRIGGER IF EXISTS trg\_check\_booking\_conflicts ON public.booking;

CREATE OR REPLACE TRIGGER trg\_check\_booking\_conflicts

BEFORE INSERT

ON public.booking

FOR EACH ROW

EXECUTE FUNCTION public.check\_booking\_conflicts();

-- Trigger: trg\_create\_payment\_before\_booking

-- DROP TRIGGER IF EXISTS trg\_create\_payment\_before\_booking ON public.booking;

CREATE OR REPLACE TRIGGER trg\_create\_payment\_before\_booking

BEFORE INSERT

ON public.booking

FOR EACH ROW

EXECUTE FUNCTION public.create\_payment\_before\_booking();

-- Table: public.game

-- DROP TABLE IF EXISTS public.game;

CREATE TABLE IF NOT EXISTS public.game

(

game\_id integer NOT NULL GENERATED BY DEFAULT AS IDENTITY ( INCREMENT 1 START 1 MINVALUE 1 MAXVALUE 2147483647 CACHE 1 ),

game\_name character varying(100) COLLATE pg\_catalog."default" NOT NULL,

rules text COLLATE pg\_catalog."default" NOT NULL,

status character varying COLLATE pg\_catalog."default" NOT NULL,

players character varying(20) COLLATE pg\_catalog."default" NOT NULL,

goal character varying COLLATE pg\_catalog."default" NOT NULL,

CONSTRAINT game\_pkey PRIMARY KEY (game\_id),

CONSTRAINT status\_check CHECK (status::text = ANY (ARRAY['new'::character varying, 'good'::character varying, 'used'::character varying, 'unusable'::character varying]::text[]))

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public.game

OWNER to postgres;

REVOKE ALL ON TABLE public.game FROM admin\_role;

REVOKE ALL ON TABLE public.game FROM user\_role;

REVOKE ALL ON TABLE public.game FROM visitor\_role;

GRANT SELECT ON TABLE public.game TO admin\_role;

GRANT ALL ON TABLE public.game TO manager\_role;

GRANT ALL ON TABLE public.game TO postgres;

GRANT SELECT ON TABLE public.game TO user\_role;

GRANT SELECT ON TABLE public.game TO visitor\_role;

-- Table: public.group

-- DROP TABLE IF EXISTS public."group";

CREATE TABLE IF NOT EXISTS public."group"

(

group\_id integer NOT NULL GENERATED BY DEFAULT AS IDENTITY ( INCREMENT 1 START 1 MINVALUE 1 MAXVALUE 2147483647 CACHE 1 ),

players\_amount integer NOT NULL,

is\_full boolean NOT NULL,

CONSTRAINT team\_pkey PRIMARY KEY (group\_id),

CONSTRAINT players\_amount\_check CHECK (players\_amount >= 2 AND players\_amount <= 20)

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public."group"

OWNER to postgres;

REVOKE ALL ON TABLE public."group" FROM admin\_role;

REVOKE ALL ON TABLE public."group" FROM user\_role;

REVOKE ALL ON TABLE public."group" FROM visitor\_role;

GRANT DELETE, UPDATE, SELECT, INSERT ON TABLE public."group" TO admin\_role;

GRANT ALL ON TABLE public."group" TO manager\_role;

GRANT ALL ON TABLE public."group" TO postgres;

GRANT SELECT, UPDATE, INSERT ON TABLE public."group" TO user\_role;

GRANT SELECT ON TABLE public."group" TO visitor\_role;

-- Table: public.members\_list

-- DROP TABLE IF EXISTS public.members\_list;

CREATE TABLE IF NOT EXISTS public.members\_list

(

list\_id integer NOT NULL GENERATED BY DEFAULT AS IDENTITY ( INCREMENT 1 START 1 MINVALUE 1 MAXVALUE 2147483647 CACHE 1 ),

group\_id integer NOT NULL,

email character varying(50) COLLATE pg\_catalog."default" NOT NULL,

CONSTRAINT members\_team\_id FOREIGN KEY (group\_id)

REFERENCES public."group" (group\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

NOT VALID,

CONSTRAINT player\_fk\_member FOREIGN KEY (email)

REFERENCES public.player (email) MATCH SIMPLE

ON UPDATE RESTRICT

ON DELETE RESTRICT

NOT VALID

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public.members\_list

OWNER to postgres;

REVOKE ALL ON TABLE public.members\_list FROM admin\_role;

REVOKE ALL ON TABLE public.members\_list FROM user\_role;

REVOKE ALL ON TABLE public.members\_list FROM visitor\_role;

GRANT UPDATE, SELECT, DELETE, INSERT ON TABLE public.members\_list TO admin\_role;

GRANT ALL ON TABLE public.members\_list TO manager\_role;

GRANT ALL ON TABLE public.members\_list TO postgres;

GRANT SELECT, INSERT ON TABLE public.members\_list TO user\_role;

GRANT SELECT ON TABLE public.members\_list TO visitor\_role;

-- Trigger: trigger\_update\_players\_amount

-- DROP TRIGGER IF EXISTS trigger\_update\_players\_amount ON public.members\_list;

CREATE OR REPLACE TRIGGER trigger\_update\_players\_amount

AFTER INSERT

ON public.members\_list

FOR EACH ROW

EXECUTE FUNCTION public.update\_players\_amount();

-- Table: public.monopoly\_championship

-- DROP TABLE IF EXISTS public.monopoly\_championship;

CREATE TABLE IF NOT EXISTS public.monopoly\_championship

(

play\_id integer NOT NULL GENERATED BY DEFAULT AS IDENTITY ( INCREMENT 1 START 1 MINVALUE 1 MAXVALUE 2147483647 CACHE 1 ),

group\_id integer NOT NULL,

winner\_id character varying(50) COLLATE pg\_catalog."default" NOT NULL,

game\_id integer NOT NULL,

date date NOT NULL,

"time" time without time zone NOT NULL,

stage character varying COLLATE pg\_catalog."default" NOT NULL,

CONSTRAINT monopoly\_championship\_pkey PRIMARY KEY (play\_id),

CONSTRAINT champ\_game\_fk FOREIGN KEY (game\_id)

REFERENCES public.game (game\_id) MATCH SIMPLE

ON UPDATE RESTRICT

ON DELETE RESTRICT,

CONSTRAINT champ\_team\_fk FOREIGN KEY (group\_id)

REFERENCES public."group" (group\_id) MATCH SIMPLE

ON UPDATE RESTRICT

ON DELETE RESTRICT,

CONSTRAINT winner\_fk FOREIGN KEY (winner\_id)

REFERENCES public.player (email) MATCH SIMPLE

ON UPDATE RESTRICT

ON DELETE RESTRICT

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public.monopoly\_championship

OWNER to postgres;

REVOKE ALL ON TABLE public.monopoly\_championship FROM admin\_role;

REVOKE ALL ON TABLE public.monopoly\_championship FROM visitor\_role;

GRANT SELECT ON TABLE public.monopoly\_championship TO admin\_role;

GRANT ALL ON TABLE public.monopoly\_championship TO manager\_role;

GRANT ALL ON TABLE public.monopoly\_championship TO postgres;

GRANT SELECT ON TABLE public.monopoly\_championship TO visitor\_role;

-- Index: idx\_champ\_date

-- DROP INDEX IF EXISTS public.idx\_champ\_date;

CREATE INDEX IF NOT EXISTS idx\_champ\_date

ON public.monopoly\_championship USING btree

(date ASC NULLS LAST)

TABLESPACE pg\_default;

-- Index: idx\_champ\_group\_id

-- DROP INDEX IF EXISTS public.idx\_champ\_group\_id;

CREATE INDEX IF NOT EXISTS idx\_champ\_group\_id

ON public.monopoly\_championship USING btree

(group\_id ASC NULLS LAST)

TABLESPACE pg\_default;

-- Table: public.payment

-- DROP TABLE IF EXISTS public.payment;

CREATE TABLE IF NOT EXISTS public.payment

(

payment\_id integer NOT NULL GENERATED BY DEFAULT AS IDENTITY ( INCREMENT 1 START 1 MINVALUE 1 MAXVALUE 2147483647 CACHE 1 ),

status character varying COLLATE pg\_catalog."default",

amount integer,

card\_or\_cash boolean,

CONSTRAINT payment\_pkey PRIMARY KEY (payment\_id),

CONSTRAINT payment\_status\_check CHECK (status::text = ANY (ARRAY['booked'::character varying, 'waiting'::character varying, 'successful'::character varying, 'declined'::character varying, 'unbooked'::character varying]::text[]))

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public.payment

OWNER to postgres;

REVOKE ALL ON TABLE public.payment FROM admin\_role;

REVOKE ALL ON TABLE public.payment FROM user\_role;

GRANT UPDATE, SELECT, DELETE, INSERT ON TABLE public.payment TO admin\_role;

GRANT ALL ON TABLE public.payment TO manager\_role;

GRANT ALL ON TABLE public.payment TO postgres;

GRANT INSERT, SELECT ON TABLE public.payment TO user\_role;

-- Table: public.player

-- DROP TABLE IF EXISTS public.player;

CREATE TABLE IF NOT EXISTS public.player

(

email character varying(50) COLLATE pg\_catalog."default" NOT NULL,

full\_name character varying(100) COLLATE pg\_catalog."default" NOT NULL,

phone integer NOT NULL,

CONSTRAINT player\_pkey PRIMARY KEY (email),

CONSTRAINT email\_unique UNIQUE (email)

INCLUDE(email)

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public.player

OWNER to postgres;

REVOKE ALL ON TABLE public.player FROM admin\_role;

REVOKE ALL ON TABLE public.player FROM user\_role;

REVOKE ALL ON TABLE public.player FROM visitor\_role;

GRANT DELETE, INSERT, SELECT, UPDATE ON TABLE public.player TO admin\_role;

GRANT ALL ON TABLE public.player TO manager\_role;

GRANT ALL ON TABLE public.player TO postgres;

GRANT UPDATE, INSERT, SELECT ON TABLE public.player TO user\_role;

GRANT SELECT, INSERT ON TABLE public.player TO visitor\_role;

-- Table: public.table

-- DROP TABLE IF EXISTS public."table";

CREATE TABLE IF NOT EXISTS public."table"

(

table\_id integer NOT NULL GENERATED BY DEFAULT AS IDENTITY ( INCREMENT 1 START 1 MINVALUE 1 MAXVALUE 2147483647 CACHE 1 ),

status boolean NOT NULL,

state character varying COLLATE pg\_catalog."default" NOT NULL,

CONSTRAINT table\_pkey PRIMARY KEY (table\_id),

CONSTRAINT state\_check CHECK (state::text = ANY (ARRAY['clean'::character varying, 'dirty'::character varying, 'unusable'::character varying]::text[]))

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public."table"

OWNER to postgres;

REVOKE ALL ON TABLE public."table" FROM admin\_role;

REVOKE ALL ON TABLE public."table" FROM user\_role;

GRANT SELECT ON TABLE public."table" TO admin\_role;

GRANT ALL ON TABLE public."table" TO manager\_role;

GRANT ALL ON TABLE public."table" TO postgres;

GRANT SELECT ON TABLE public."table" TO user\_role;

**II. Скрипт завантаження первинних даних**

**INSERT INTO public.player (email, full\_name, phone) VALUES**

**('alice@example.com', 'Alice Johnson', 123456789),**

**('bob@example.com', 'Bob Smith', 987654321),**

**('carol@example.com', 'Carol White', 456123789),**

**('dave@example.com', 'Dave Brown', 321654987);**

**-- ACCOUNT**

**INSERT INTO public.account (password, player\_email) VALUES**

**('hashedpass1', 'alice@example.com'),**

**('hashedpass2', 'bob@example.com'),**

**('hashedpass3', 'carol@example.com'),**

**('hashedpass4', 'dave@example.com');**

**-- ADMIN**

**INSERT INTO public.admin (access\_type, admin\_id) VALUES**

**('employee', 1),**

**('manager', 2);**

**-- GAME**

**INSERT INTO public.game (game\_name, rules, status, players, goal) VALUES**

**('Monopoly Classic', 'Standard Monopoly rules', 'new', '2-6', 'Bankruptcy'),**

**('Catan', 'Build settlements and roads', 'good', '3-4', '10 victory points');**

**-- TABLE**

**INSERT INTO public."table" (status, state) VALUES**

**(TRUE, 'clean'),**

**(TRUE, 'dirty'),**

**(FALSE, 'unusable');**

**-- GROUP**

**INSERT INTO public."group" (players\_amount, is\_full) VALUES**

**(3, FALSE),**

**(2, TRUE);**

**-- MEMBERS\_LIST**

**INSERT INTO public.members\_list (group\_id, email) VALUES**

**(1, 'alice@example.com'),**

**(1, 'bob@example.com'),**

**(1, 'carol@example.com'),**

**(2, 'dave@example.com'),**

**(2, 'alice@example.com');**

**-- PAYMENT**

**INSERT INTO public.payment (status, amount, card\_or\_cash) VALUES**

**('booked', 100, TRUE),**

**('successful', 150, FALSE),**

**('waiting', 75, TRUE);**

**-- BOOKING**

**INSERT INTO public.booking (group\_id, table\_id, game\_id, payment\_id, date, "time", period) VALUES**

**(1, 1, 1, 1, '2025-05-15', '14:00', '02:00'),**

**(2, 2, 2, 2, '2025-05-16', '16:00', '01:30');**

**-- MONOPOLY\_CHAMPIONSHIP**

**INSERT INTO public.monopoly\_championship (group\_id, winner\_id, game\_id, date, "time", stage) VALUES**

**(1, 'alice@example.com', 1, '2025-05-10', '12:00', 'quarter-final'),**

**(2, 'dave@example.com', 1, '2025-05-11', '15:00', 'semi-final');**

**III. Sql запити, що відповідають бізнес процесам**

### Відображення бронювань користувача:

Цей запит відображає всі бронювання для конкретної групи за допомогою ID групи, повертаючи відповідну інформацію про дату, час, період, стіл, гру та учасників групи.

sql

КопіюватиРедагувати

SELECT

b.booking\_id,

b.date,

b.time,

b.period,

b.table\_id,

g.game\_name,

gr.group\_id,

gr.players\_amount,

gr.is\_full,

(

SELECT array\_agg(ml.email)

FROM public.members\_list ml

WHERE ml.group\_id = gr.group\_id

) AS member\_emails

FROM public.booking b

JOIN public.game g ON b.game\_id = g.game\_id

JOIN public."group" gr ON b.group\_id = gr.group\_id

WHERE b.group\_id = ANY($1::int[])

ORDER BY b.date DESC, b.time DESC;

**Опис:**  
 Цей запит дозволяє отримати інформацію про всі бронювання, зроблені певною групою (за вказаними ID групи). Для кожного бронювання виводяться дані про гру, стіл, період, а також список учасників групи.

### Отримання списку матчів:

Цей запит відображає всі матчі чемпіонату, що проводяться, включаючи інформацію про гравців та переможця.

SELECT

mc.play\_id,

mc.date,

mc.time,

mc.stage,

mc.winner\_id AS winner\_email,

g.players\_amount,

ARRAY\_AGG(m.email) AS player\_emails,

ARRAY\_AGG(pl.full\_name) AS player\_names

FROM monopoly\_championship mc

JOIN "group" g ON mc.group\_id = g.group\_id

JOIN members\_list m ON m.group\_id = g.group\_id

JOIN player pl ON pl.email = m.email

GROUP BY mc.play\_id, mc.date, mc.time, mc.stage, mc.winner\_id, g.players\_amount

ORDER BY mc.date DESC, mc.time DESC;

**Опис:**  
 Цей запит дозволяє отримати всю необхідну інформацію про матчі чемпіонату, включаючи учасників, їхні електронні адреси та повні імена, а також інформацію про переможця.

### Перевірка всіх платежів:

Цей запит дозволяє отримати всі платежі, зроблені протягом певного року.

sql

КопіюватиРедагувати

SELECT DISTINCT

p.payment\_id,

p.amount,

p.status,

p.card\_or\_cash

FROM public.payment p

JOIN public.booking b ON p.payment\_id = b.payment\_id

WHERE EXTRACT(YEAR FROM b.date) = $1;

**Опис:**  
 Цей запит дозволяє отримати всю інформацію про платежі, зроблені протягом конкретного року, включаючи статус платежу та спосіб оплати.

### Створення нового бронювання:

Цей запит дозволяє створити нове бронювання для гри, перевіряючи наявність гравців, створюючи нову групу та бронюючи стіл.

sql

КопіюватиРедагувати

BEGIN;

-- Перевірка чи існують гравці або їх додавання

-- Створення нової групи

-- Створення бронювання та спроба забронювати стіл

COMMIT;

**Опис:**  
 Запит використовується для створення нового бронювання для гри. Він включає перевірку існуючих гравців, додавання нових учасників, створення групи та спробу забронювати стіл для гри.

**IIV. Програмний код реалізації бізнес-логіки**

Index.js:  
const express = require("express");

const cors = require("cors");

const { Client } = require("pg");

const app = express();

app.use(cors());

app.use(express.json());

const loggedUsers = {}; // for storing only users that are logged locally on their devices and then check it user email exist and match the created sessionId

app.get("/games", async (req, res) => {

    const client = new Client({

        host: "localhost",

        user: "visitor\_user",

        port: 5432,

        password: "1111",

        database: "BoardGames",

    });

    try {

        await client.connect();

        const result = await client.query("SELECT \* FROM public.game");

        await client.end();

        res.json(result.rows);

    } catch (err) {

        console.error("DB error:", err.message);

        res.status(500).json({ error: err.message });

    }

});

app.post("/game\_create", async (req, res) => {

    const { email, userId, game\_name, status, players, goal, rules } = req.body;

    const loggedUser = loggedUsers[email];

    if (

        !loggedUser ||

        loggedUser.id !== userId ||

        loggedUser.access !== "manager"

    ) {

        return res.status(403).json({ error: "Unauthorized: Invalid credentials or insufficient access." });

    }

    if (!game\_name || !status || !players || !goal || !rules) {

        return res.status(400).json({ error: "Missing required fields." });

    }

    const client = new Client({

        host: "localhost",

        user: "manager1",

        port: 5432,

        password: "1111111",

        database: "BoardGames",

    });

    try {

        await client.connect();

        const insertQuery = `

            INSERT INTO public.game (game\_name, status, rules, players, goal)

            VALUES ($1, $2, $3, $4, $5)

            RETURNING \*;

        `;

        const result = await client.query(insertQuery, [game\_name, status, rules, players, goal]);

        await client.end();

        res.status(201).json({

            success: true,

            message: "Game created successfully.",

            game: result.rows[0]

        });

    } catch (err) {

        console.error("DB error:", err.message);

        res.status(500).json({ error: err.message });

    }

});

app.post("/game\_delete", async (req, res) => {

    const { email, userId, game\_id } = req.body;

    const loggedUser = loggedUsers[email];

    if (

        !loggedUser ||

        loggedUser.id !== userId ||

        loggedUser.access !== "manager"

    ) {

        return res.status(403).json({ error: "Unauthorized: Invalid credentials or insufficient access." });

    }

    if (!game\_id) {

        return res.status(400).json({ error: "Missing game\_id in request body." });

    }

    const client = new Client({

        host: "localhost",

        user: "manager1",

        port: 5432,

        password: "1111111",

        database: "BoardGames",

    });

    try {

        await client.connect();

        // Step 1: Check for future bookings using this game\_id

        const now = new Date().toISOString(); // ISO format for timestamp comparison

        const checkQuery = `

            SELECT \* FROM public.booking

            WHERE game\_id = $1 AND (date::timestamp + time::time) > $2

        `;

        const checkResult = await client.query(checkQuery, [game\_id, now]);

        if (checkResult.rows.length > 0) {

            await client.end();

            return res.status(400).json({ error: "Cannot delete game: future bookings exist." });

        }

        // Step 2: Delete the game

        const deleteQuery = `DELETE FROM public.game WHERE game\_id = $1`;

        await client.query(deleteQuery, [game\_id]);

        await client.end();

        res.json({ success: true, message: "Game deleted successfully." });

    } catch (err) {

        console.error("DB error:", err.message);

        res.status(500).json({ error: err.message });

    }

});

app.get("/matches", async (req, res) => {

    const client = new Client({

        host: "localhost",

        user: "visitor\_user",

        port: 5432,

        password: "1111",

        database: "BoardGames",

    });

    try {

        await client.connect();

        const result = await client.query(`

            SELECT

                mc.play\_id,

                mc.date,

                mc.time,

                mc.stage,

                mc.winner\_id AS winner\_email,

                g.players\_amount,

                ARRAY\_AGG(m.email) AS player\_emails,

                ARRAY\_AGG(pl.full\_name) AS player\_names

            FROM monopoly\_championship mc

            JOIN "group" g ON mc.group\_id = g.group\_id

            JOIN members\_list m ON m.group\_id = g.group\_id

            JOIN player pl ON pl.email = m.email

            GROUP BY mc.play\_id, mc.date, mc.time, mc.stage, mc.winner\_id, g.players\_amount

            ORDER BY mc.date DESC, mc.time DESC;

        `);

        res.json(result.rows);

    } catch (err) {

        console.error("Error fetching matches:", err.message);

        res.status(500).json({ error: err.message });

    } finally {

        await client.end();

    }

});

app.post("/match\_create", async (req, res) => {

    const { email, userId, playerCount, players, winner, date, time, stage } = req.body;

    const loggedUser = loggedUsers[email];

    if (

        !loggedUser ||

        loggedUser.id !== userId ||

        loggedUser.access !== "manager"

    ) {

        return res.status(403).json({ error: "Unauthorized: Invalid credentials or insufficient access." });

    }

    console.log("Received data:", { playerCount, players, winner, date, time, stage });

    const client = new Client({

        host: "localhost",

        user: "manager1",

        port: 5432,

        password: "1111111",

        database: "BoardGames",

    });

    try {

        await client.connect();

        console.log("DB connected");

        await client.query("BEGIN");

        // Step 1: Ensure players exist

        for (const player of players) {

            const { email, full\_name, phone } = player;

            console.log(`Checking/creating player: ${email}`);

            const checkPlayer = await client.query("SELECT 1 FROM player WHERE email = $1", [email]);

            if (checkPlayer.rowCount === 0) {

                await client.query(

                    "INSERT INTO player (email, full\_name, phone) VALUES ($1, $2, $3)",

                    [email, full\_name, phone]

                );

                console.log(`Inserted player: ${email}`);

            }

        }

        // Step 2: Create new group

        const groupRes = await client.query(

            "INSERT INTO public.group (players\_amount, is\_full) VALUES ($1, $2) RETURNING group\_id",

            [playerCount, true]

        );

        const groupId = groupRes.rows[0].group\_id;

        console.log("Created group:", groupId);

        // Step 3: Add players to members\_list

        for (const player of players) {

            await client.query(

                "INSERT INTO members\_list (group\_id, email) VALUES ($1, $2)",

                [groupId, player.email]

            );

            console.log(`Added ${player.email} to group ${groupId}`);

        }

        // Step 4: Insert into monopoly\_championship

        const gameId = 1; // Hardcoded

        console.log("Inserting championship record:", {

            groupId,

            winner,

            gameId,

            stage,

            date,

            time,

        });

        await client.query(

            `INSERT INTO monopoly\_championship (group\_id, winner\_id, game\_id, stage, date, time)

            VALUES ($1, $2, $3, $4, $5, $6)`,

            [groupId, winner, gameId, stage, date, time]

        );

        await client.query("COMMIT");

        console.log("Match successfully created");

        res.json({ message: "Match created successfully" });

    } catch (err) {

        console.error("Match creation failed:", err);

        try {

            await client.query("ROLLBACK");

        } catch (rollbackErr) {

            console.error("Rollback failed:", rollbackErr);

        }

        res.status(500).json({ error: err.message });

    } finally {

        try {

            await client.end();

            console.log("DB connection closed");

        } catch (endErr) {

            console.error("Failed to close DB connection:", endErr);

        }

    }

});

app.post("/tables", async (req, res) => {

    const { email, userId} = req.body;

    const loggedUser = loggedUsers[email];

    if (

        !loggedUser ||

        loggedUser.id !== userId ||

        (loggedUser.access !== "manager" && loggedUser.access !== "employee")

    ) {

        return res.status(403).json({ error: "Unauthorized: Invalid credentials or insufficient access." });

    }

    const client = new Client({

        host: "localhost",

        user: "manager1",

        port: 5432,

        password: "1111111",

        database: "BoardGames",

    });

    try {

        await client.connect();

        const result = await client.query("SELECT \* FROM public.table");

        await client.end();

        res.json(result.rows);

    } catch (err) {

        console.error("DB error:", err.message);

        res.status(500).json({ error: err.message });

    }

});

app.post("/register", async (req, res) => {

    const { email, name, phone, password } = req.body;

    const client = new Client({

        host: "localhost",

        user: "visitor\_user",

        port: 5432,

        password: "1111",

        database: "BoardGames",

    });

    try {

        await client.connect();

        const playerCheck = await client.query("SELECT \* FROM player WHERE email = $1", [email]);

        if (playerCheck.rows.length === 0) {

            // Insert new player

            await client.query("INSERT INTO player (email, full\_name, phone) VALUES ($1, $2, $3)", [email, name, phone]);

        }

        // Check if account already exists

        const accountCheck = await client.query("SELECT \* FROM account WHERE player\_email = $1", [email]);

        if (accountCheck.rows.length > 0) {

            await client.end();

            return res.status(400).json({ error: "Account already exists" });

        }

        // Create account

        await client.query("INSERT INTO account (password, player\_email) VALUES ($1, $2)", [password, email]);

        const newAccount = await client.query("SELECT \* FROM account WHERE player\_email = $1", [email]);

        await client.end();

        res.json("registered");

    } catch (err) {

        console.error("Register error:", err.message);

        res.status(500).json({ error: err.message });

    }

});

app.post("/login", async (req, res) => {

    const { email, password } = req.body;

    const client = new Client({

        host: "localhost",

        user: "visitor\_user",

        port: 5432,

        password: "1111",

        database: "BoardGames",

    });

    try {

        await client.connect();

        // Check if account with correct password exists

        const accountRes = await client.query(

            "SELECT \* FROM account WHERE player\_email = $1 AND password = $2",

            [email, password]

        );

        if (accountRes.rows.length === 0) {

            await client.end();

            return res.status(401).json({ error: "Invalid email or password" });

        }

        // Get player info

        const playerRes = await client.query("SELECT \* FROM player WHERE email = $1", [email]);

        const adminRes = await client.query("SELECT access\_type FROM admin WHERE admin\_id = $1", [accountRes.rows[0].account\_id]);

        const access = adminRes.rows.length > 0 ? adminRes.rows[0].access\_type : "user";

        // ✅ Generate a unique ID based on timestamp and a random component

        const sessionId = `${Date.now()}-${Math.floor(Math.random() \* 100000)}`;

        // ✅ Log the user session

        loggedUsers[email] = {

            id: sessionId,

            access: access

        };

        await client.end();

        res.json({

            ...accountRes.rows[0],

            name: playerRes.rows[0].full\_name,

            phone: playerRes.rows[0].phone,

            access: adminRes.rows.length > 0 ? adminRes.rows[0].access\_type : null,

            userId: sessionId

        });

    } catch (err) {

        console.error("Login error:", err.message);

        res.status(500).json({ error: err.message });

    }

});

app.post("/logout", (req, res) => {

    const { email, userId } = req.body;

    const loggedUser = loggedUsers[email];

    if (

        !loggedUser ||

        loggedUser.id !== userId

    ) {

        return res.status(403).json({ error: "Invalid credentials. Logout failed." });

    }

    // Remove user from the loggedUsers list

    delete loggedUsers[email];

    res.json({ message: "Logout successful." });

});

app.put("/update\_user", async (req, res) => {

    const { email, userId, name, phone, password } = req.body;

    const loggedUser = loggedUsers[email];

    if (

        !loggedUser ||

        loggedUser.id !== userId

    ) {

        return res.status(403).json({ error: "Unauthorized: Invalid credentials or insufficient access." });

    }

    const client = new Client({

        host: "localhost",

        user: "user1",

        port: 5432,

        password: "11111",

        database: "BoardGames",

    });

    try {

    await client.connect();

      // Update player info

    await client.query(

        "UPDATE player SET full\_name = $1, phone = $2 WHERE email = $3",

        [name, phone, email]

    );

      // Update account password

    await client.query(

        "UPDATE account SET password = $1 WHERE player\_email = $2",

        [password, email]

    );

    await client.end();

    res.json({ message: "User updated successfully" });

    } catch (err) {

        console.error("Update error:", err.message);

        res.status(500).json({ error: err.message });

    }

});

app.post("/bookings", async (req, res) => {

    const {email, userId } = req.body;

    const loggedUser = loggedUsers[email];

    if (

        !loggedUser ||

        loggedUser.id !== userId

    ) {

        return res.status(403).json({ error: "Unauthorized: Invalid credentials or insufficient access." });

    }

    const client = new Client({

        host: "localhost",

        user: "user1",

        port: 5432,

        password: "11111",

        database: "BoardGames",

    });

    try {

        await client.connect();

        // Отримуємо всі group\_id для користувача

        const groupResult = await client.query(

            `SELECT DISTINCT group\_id FROM public.members\_list WHERE email = $1`,

            [email]

        );

        const groupIds = groupResult.rows.map(row => row.group\_id);

        if (groupIds.length === 0) {

            await client.end();

            return res.json({ bookings: [], groups: [] });

        }

        // Бронювання для цих груп

        const bookingResult = await client.query(`

            SELECT

                b.booking\_id,

                b.date,

                b.time,

                b.period,

                b.table\_id,

                g.game\_name,

                gr.group\_id,

                gr.players\_amount,

                gr.is\_full,

                (

                    SELECT array\_agg(ml.email)

                    FROM public.members\_list ml

                    WHERE ml.group\_id = gr.group\_id

                ) AS member\_emails

            FROM public.booking b

            JOIN public.game g ON b.game\_id = g.game\_id

            JOIN public."group" gr ON b.group\_id = gr.group\_id

            WHERE b.group\_id = ANY($1::int[])

            ORDER BY b.date DESC, b.time DESC

        `, [groupIds]);

        await client.end();

        res.json({

            bookings: bookingResult.rows

        });

    } catch (err) {

        console.error("DB error:", err.message);

        res.status(500).json({ error: err.message });

    }

});

app.post("/bookings\_admin", async (req, res) => {

    const {email, userId, year } = req.body;

    const loggedUser = loggedUsers[email];

    if (

        !loggedUser ||

        loggedUser.id !== userId ||

        (loggedUser.access !== "manager" && loggedUser.access !== "employee")

    ) {

        return res.status(403).json({ error: "Unauthorized: Invalid credentials or insufficient access." });

    }

    const client = new Client({

        host: "localhost",

        user: "manager1",

        port: 5432,

        password: "1111111",

        database: "BoardGames",

    });

    try {

        await client.connect();

        // ВСІ бронювання за вказаний рік

        const bookingResult = await client.query(`

            SELECT

                b.booking\_id,

                b.date,

                b.time,

                b.period,

                b.table\_id,

                b.payment\_id,

                g.game\_name,

                gr.group\_id,

                gr.players\_amount,

                gr.is\_full,

                (

                    SELECT array\_agg(ml.email)

                    FROM public.members\_list ml

                    WHERE ml.group\_id = gr.group\_id

                ) AS member\_emails

            FROM public.booking b

            JOIN public.game g ON b.game\_id = g.game\_id

            JOIN public."group" gr ON b.group\_id = gr.group\_id

            WHERE EXTRACT(YEAR FROM b.date) = $1

            ORDER BY b.date DESC, b.time DESC

        `, [year]);

        await client.end();

        res.json({

            bookings: bookingResult.rows

        });

    } catch (err) {

        console.error("DB error:", err.message);

        res.status(500).json({ error: err.message });

    }

});

app.post("/create\_booking", async (req, res) => {

    const { email, userId, date, time, period, game, players } = req.body;

    const loggedUser = loggedUsers[email];

    if (

        !loggedUser ||

        loggedUser.id !== userId

    ) {

        return res.status(403).json({ error: "Unauthorized: Invalid credentials or insufficient access." });

    }

    const client = new Client({

        host: "localhost",

        user: "user1",

        port: 5432,

        password: "11111",

        database: "BoardGames",

    });

    try {

        await client.connect();

        // Get game\_id from game name

        const gameRes = await client.query(

            "SELECT game\_id FROM public.game WHERE game\_name = $1",

            [game]

        );

        if (gameRes.rows.length === 0) {

            throw new Error(`Game "${game}" not found.`);

        }

        const gameId = gameRes.rows[0].game\_id;

        //Ensure all players exist or create them

        for (const player of players.slice(1)) {

            const { email, name, phone } = player;

            const existingPlayer = await client.query(

                "SELECT \* FROM public.player WHERE email = $1",

                [email]

            );

            const parsedPhone = parseInt(phone, 10);

            if (isNaN(parsedPhone)) {

                throw new Error(`Invalid phone number for ${email}`);

            }

            if (existingPlayer.rows.length === 0) {

                await client.query(

                    "INSERT INTO public.player (email, full\_name, phone) VALUES ($1, $2, $3)",

                    [email, name, parsedPhone]

                );

            }

        }

        // Create a new group

        const groupRes = await client.query(

            `INSERT INTO public.group (players\_amount, is\_full)

            VALUES ($1, $2)

            RETURNING group\_id`,

            [2, true] // temporary minimum valid value to pass CHECK constraint

        );

        const groupId = groupRes.rows[0].group\_id;

        //period conevrsion

        const periodHours = parseInt(period, 10);

        if (isNaN(periodHours) || periodHours <= 0) {

            throw new Error("Invalid period value");

        }

        const periodTime = `${String(periodHours).padStart(2, "0")}:00:00`;

        // Add the actual user (booking creator) to the group first

        await client.query(

            "INSERT INTO public.members\_list (group\_id, email) VALUES ($1, $2)",

            [groupId, email]

        );

        // Then add the rest of the valid players, skipping the first (empty) one

        for (const player of players.slice(1)) {

            await client.query(

                "INSERT INTO public.members\_list (group\_id, email) VALUES ($1, $2)",

                [groupId, player.email]

            );

        }

        // Try to book a table

        const tablesRes = await client.query("SELECT table\_id FROM public.table");

        const tableIds = tablesRes.rows.map(row => row.table\_id);

        let bookingCreated = false;

        let errorMessage = "All tables are taken or booking failed.";

        for (const tableId of tableIds) {

            try {

                await client.query(

                    `INSERT INTO public.booking

                    (date, time, period, table\_id, game\_id, group\_id, payment\_id)

                    VALUES ($1, $2, $3, $4, $5, $6, $7)`,

                    [date, time, periodTime, tableId, gameId, groupId, 0]

                );

                bookingCreated = true;

                break;

            } catch (err) {

                // Continue trying next table

                console.warn(`Table ${tableId} failed:`, err.message);

            }

        }

        await client.end();

        if (!bookingCreated) {

            return res.status(400).json({ error: errorMessage });

        }

        res.status(201).json({ message: "Booking created successfully." });

    } catch (err) {

        console.error("Error:", err.message);

        res.status(500).json({ error: err.message });

    }

});

app.post("/booking\_delete", async (req, res) => {

    const { email, userId, id } = req.body; // booking\_id

    const loggedUser = loggedUsers[email];

    if (

        !loggedUser ||

        loggedUser.id !== userId ||

        (loggedUser.access !== "manager" && loggedUser.access !== "employee")

    ) {

        return res.status(403).json({ error: "Unauthorized: Invalid credentials or insufficient access." });

    }

    const client = new Client({

        host: "localhost",

        user: "manager1",

        port: 5432,

        password: "1111111",

        database: "BoardGames",

    });

    try {

        await client.connect();

        // 1. Get related group\_id and payment\_id from the booking

        const bookingCheck = await client.query(

            `SELECT group\_id, payment\_id FROM public.booking WHERE booking\_id = $1`,

            [id]

        );

        if (bookingCheck.rows.length === 0) {

            await client.end();

            return res.status(404).json({ error: "Booking not found" });

        }

        const { group\_id, payment\_id } = bookingCheck.rows[0];

        // 2. Delete the booking

        await client.query(`DELETE FROM public.booking WHERE booking\_id = $1`, [id]);

        // 3. Delete the payment

        await client.query(`DELETE FROM public.payment WHERE payment\_id = $1`, [payment\_id]);

        // 4. Delete related members\_list rows

        await client.query(`DELETE FROM public.members\_list WHERE group\_id = $1`, [group\_id]);

        // 5. Delete the group

        await client.query(`DELETE FROM public."group" WHERE group\_id = $1`, [group\_id]);

        await client.end();

        res.json({ success: true, message: "Booking and related records deleted successfully." });

    } catch (err) {

        console.error("DB error:", err.message);

        res.status(500).json({ error: err.message });

    }

});

app.post("/checkAll\_payments", async (req, res) => {

    const { email, userId, year } = req.body;

    const loggedUser = loggedUsers[email];

    if (

        !loggedUser ||

        loggedUser.id !== userId ||

        (loggedUser.access !== "manager" && loggedUser.access !== "employee")

    ) {

        return res.status(403).json({ error: "Unauthorized: Invalid credentials or insufficient access." });

    }

    const client = new Client({

        host: "localhost",

        user: "manager1",

        port: 5432,

        password: "1111111",

        database: "BoardGames",

    });

    try {

        await client.connect();

        const result = await client.query(`

            SELECT DISTINCT

                p.payment\_id,

                p.amount,

                p.status,

                p.card\_or\_cash

            FROM public.payment p

            JOIN public.booking b ON p.payment\_id = b.payment\_id

            WHERE EXTRACT(YEAR FROM b.date) = $1

        `, [year]);

        await client.end();

        res.json(result.rows);

    } catch (err) {

        console.error("Database error:", err.message);

        res.status(500).json({ error: "Failed to retrieve payments" });

    }

});

app.put("/update\_payment", async (req, res) => {

    const { email, userId, payment\_id, status, amount, card\_or\_cash } = req.body;

    const loggedUser = loggedUsers[email];

    if (

        !loggedUser ||

        loggedUser.id !== userId ||

        (loggedUser.access !== "manager" && loggedUser.access !== "employee")

    ) {

        return res.status(403).json({ error: "Unauthorized: Invalid credentials or insufficient access." });

    }

    const client = new Client({

        host: "localhost",

        user: "manager1",

        port: 5432,

        password: "1111111",

        database: "BoardGames",

    });

    try {

        await client.connect();

        await client.query(

            "UPDATE payment SET status = $1, amount = $2, card\_or\_cash = $3 WHERE payment\_id = $4",

            [status, amount, card\_or\_cash, payment\_id]

        );

        await client.end();

        res.json({ message: "Payment updated successfully" });

    } catch (err) {

        console.error("Payment update error:", err.message);

        res.status(500).json({ error: err.message });

    }

});

app.listen(3001, () => {

    console.log("Server is running on http://localhost:3001");

});

**V. Код побудови інтерфейсу користувача**

Page.js:

"use client"

import styles from "./Main.module.scss";

import Image from "next/image";

import back from "../public/Untitled-1.svg"

import Button from "./\_components/Button";

import SearchBar from "./\_components/SearchBar/SearchBar";

import { useRouter } from 'next/navigation';

export default function Home() {

  const router = useRouter();

  function handleNav() {

      router.push('/login');

  }

  return (

    <div className={styles.main\_title}>

      <h2 className={styles.title}>Have a great time with friends<br/> playing our games</h2>

      <p>FROM BOOKING TO BOARD – YOUR GAME STARTS HERE.</p>

      <Image

        className={styles.main\_back\_img}

        src={back}

        alt="People playing a board game"

      />

      <Button action={handleNav} name={"button\_template1"}>Play now</Button>

      <SearchBar></SearchBar>

    </div>

  );

}

Loading.js

export default function loading() {

    return (

        <p className="loading">Loading ... </p>

    )

}

Main.module.scss

@import "./globals.scss";

.main\_title{

    padding: 50px;

    display: flex;

    flex-direction: column;

    align-items: start;

    text-align: start;

    border-radius: 0 20px 20px 20px;

    height: 620px;

    .main\_back\_img{

        position: absolute;

        width: 800px;

        left: 900px;

        top: 100px;

        filter: brightness(10%) saturate(100%) invert(63%) sepia(67%) saturate(706%) hue-rotate(103deg) brightness(96%) contrast(98%);

    }

    .title{

        color: $color\_7;

        @extend %font1;

        font-size: 70px;

        margin: 60px 0 20px 0;

        z-index: 2;

    }

    p{

        color: #484848;

        @extend %font1;

        font-size: 12px;

        letter-spacing: 2px;

        word-spacing: 4px;

        font-weight: 100;

        z-index: 2;

        margin-bottom: 30px;

    }

}

.main{

    overflow: auto;

    min-height: calc(100vh - 315px);

}

Layout.js

import Footer from "./\_components/Footer/Footer"

import Header from "./\_components/Header/Header"

import ReduxProvider from "./\_general\_state/Provider";

import styles from "./Main.module.scss";

export const metadata = {

    title: {

            template: "%s of BoardGaming",

            default: "BoardGaming",

        },

    description: 'Small boarding games rent business.',

    icons: {

        icon: '/default.png',

    },   // we can also do not write here icon and just name the picture icon.png inside of the public folder

}

export default function RootLayout({ children }) {

    return (

        <html lang="en">

            <body>

                <ReduxProvider>

                    <Header/>

                    <main className={styles.main}>

                        {children}

                    </main>

                    <Footer/>

                </ReduxProvider>

            </body>

        </html>

    )

}

Globals.scss

$color\_1: #ffff;

$color\_7: rgb(2, 27, 26);

$color\_5: rgb(3, 34, 33);

$color\_6: rgb(3, 98, 76);

$color\_3: rgb(44, 194, 149);

$color\_4: rgb(0, 223, 129);

$color\_8: rgb(17, 126, 57);

%font1 {

    font-family: "Roboto", serif;

    font-weight: 700;

    font-size: 19px;

}

%font2 {

    font-family: "Roboto", serif;

    font-weight: 100;

    font-size: 12px;

}

.button\_template{

    width: 160px;

    min-height: 50px;

    z-index: 3;

    @extend %font1;

    font-size: 20px;

    font-weight: 600;

    border: none;

    color: $color\_4;

    background-color: $color\_1;

    border-radius: 8px;

}

.button\_template1{

    width: 165px;

    min-height: 50px;

    z-index: 3;

    @extend %font1;

    font-size: 20px;

    font-weight: 600;

    border: none;

    color: $color\_1;

    background-color: $color\_4;

    border-radius: 8px;

    transition: 0.3s;

}

.button\_template1:hover{

    box-shadow: 0.4rem 0.4rem 1rem rgba(0, 0, 0, 0.274);

    transition: 0.3s;

    color: $color\_5;

}

.button\_template2{

    display: flex;

    justify-content: center;

    align-items: center;

    width: 160px;

    height: 50px;

    z-index: 3;

    @extend %font1;

    font-size: 20px;

    border: none;

    color: $color\_3;

    background-color: $color\_5;

    border-radius: 8px;

    transition: 0.3s;

}

.button\_template2:hover{

    box-shadow: 0.4rem 0.4rem 1rem rgba(0, 0, 0, 0.274);

    background-color: $color\_1;

    transition: 0.3s;

    color: $color\_5;

}

.loading{

    @extend %font1;

    font-size: 24px;

    color: $color\_6;

    display: flex;

    align-self: center;

    justify-self: center;

    margin: 100px 40px 40px 40px;

}

body{

    margin: 0;

    min-height: calc(100vh - 15px);

}

login/page.js

"use client"

import { useEffect, useState } from "react";

import Button from "../\_components/Button";

import styles from "./Login.scss";

import { useSelector, useDispatch } from "react-redux";

import { Register, Login, setError } from "../\_lib/accountSlice";

import { useRouter } from "next/navigation";

export default function Page() {

  const [sign, setSign] = useState("Register");

  const [email, setEmail] = useState("");

  const [name, setName] = useState("");

  const [phone, setPhone] = useState("");

  const [password, setPassword] = useState("");

  const [checked, setChecked] = useState(false);

  const dispatch = useDispatch();

  const router = useRouter();

  const { user, error, isLoading } = useSelector((state) => state.account);

  function handleAuthenticate(e){

    if(!checked){

      dispatch(setError("You have not confirmed the agreement"));

    }else{

      if(sign === "Register"){

        dispatch(Register({ email, name, phone: parseInt(phone), password }));

      }else{

        dispatch(Login({ email, password }));

      }

    }

  }

  useEffect(() => {

    if (user && user.email) {

      router.push("/user");

    }

  }, [user, router]);

  useEffect(() => {

    if (user && user.email) {

      router.push("/user");

    }

  }, []);

  return (

    <div >

      <select value={sign} onChange={(e) => setSign(e.target.value)}>

        <option value="Register">Register new account</option>

        <option value="Sign in">Sign in to an existing account</option>

      </select>

      <form id="sect7-log" className="section7-login">

          <h2 className="section7-consultation main-title-caption">{sign === "Register"? sign : sign + " to"} account</h2>

          <p className="section7-consultation-description">Leave your contact details in order to book a table to play any boardgames from our list</p>

          {sign === "Register" && <><input type="text" id="FromName" className="name \_req" required name="login" placeholder="Name" aria-describedby="Name" value={name} onChange={(e) => setName(e.target.value)} /> <br /></>}

          <input type="email" id="FromMail" className="name \_req" required aria-describedby="emailHelp"placeholder="E-mail" value={email} onChange={(e) => setEmail(e.target.value)}/> <br />

          {sign === "Register" && <><input type="phone" id="FromPhone" className="name \_req" aria-describedby="Phone"placeholder="Phone" value={phone} onChange={(e) => setPhone(e.target.value)}/> <br /></>}

          <input type="password" id="FromPassword" className="name \_req" required aria-describedby="Password"placeholder="Password" value={password} onChange={(e) => setPassword(e.target.value)}/> <br />

          {sign === "Register" &&<div className="section7-checkbox-label2">All new passwords must contain at least 8 characters.

          We also suggest having at least one capital and one lower-case letter (Aa-Zz), one special symbol (#, &, % etc), and one number (0-9) in your password for the best strength. <p className="strong">Please make sure to login after the registration.</p></div>}

          {error && <h2 className="error">{error}</h2>}

          <div className="check\_box\_sec">

              <input id="cons\_check" type="checkbox" required value={checked} onClick={() => setChecked((checked) => !checked)} className="section7-checkbox"/>

              <p className="section7-checkbox-label">I accept the terms of the user agreement for the processing of personal data</p>

              <Button action={handleAuthenticate} name={"button\_template2"}>

              <p>{isLoading ? "Loading..." : sign}</p>

              </Button>

          </div>

      </form>

    </div>

  );

}

login/Login.scss

@import "../globals.scss";

select{

    margin: 20px 0 0 20px;

    height: 40px;

    color: $color\_1;

    background-color: $color\_3;

    border-radius: 10px;

    border: none;

}

.name{

    padding-left: 16px;

    width: 470px;

    height: 50px;

    font-family: Poppins;

    font-weight: 600;

    font-size: 16px;

    line-height: 20px;

    border: 1px solid #bfbfbf;

    box-shadow: inset 0 0 0 55px #fff!important;

    border-radius: 15px;

    margin: 10px 0 0 0;

    outline: none;

}

.name:-webkit-autofill {

     -webkit-box-shadow: inset 0 0 0 55px #fff!important;

    box-shadow: inset 0 0 0 55px #fff!important;

}

.name :placeholder-shown{

    color: $color\_5;

    letter-spacing: 1px;

}

.main-title-btn{

    p{

        padding-left: 10px;

    }

    display: flex;

    justify-self: center;

    align-items: center;

    flex-direction: row;

}

.main-title-caption{

    @extend %font1;

    font-size: 53px;

    letter-spacing: 3px;

    font-weight: 900;

    color: $color\_5;

    margin: 8px 0 10px 0 ;

    font-weight: bolder;

    text-align: center;

    color: $color\_5;

}

.section7-login{

    position: relative;

    width: 550px;

    flex-direction: column;

    padding-left: 60px;

    margin: 20px 0 80px 0;

    background-color: $color\_4;

    padding: 8px;

    border-radius: 15px;

    display: flex;

    align-items: center;

    justify-self: center;

    box-shadow: 1rem 1.4rem 3rem rgba(0, 0, 0, 0.356);

}

.section7-login::after{

    content: "";

    top: 0;

    left: 0;

    position: absolute;

    width: 100%;

    height: 100%;

    opacity: 0;

    visibility: hidden;

    transition: 0.5s;

}

.section7-login.\_sending::after{

    opacity: 1;

    visibility: visible;

    transition: 0.5s;

}

.section7-consultation{

    text-align: start;

    font-size: 43px;

    margin-bottom: 10px;

    margin-top: 20px;

    width: 400px;

    text-align: center;

}

.section7-consultation-description{

    width: 540px;

    font-size: 19px;

    line-height: 22px;

    letter-spacing: 1.22px;

    color: $color\_5;

    margin: 0 0 7px 0;

    text-align: center;

    @extend %font2;

    font-weight: 200;

}

.img\_input1{

    position: absolute;

    top: 262px;

    margin-left: 450px;

    width: 20px;

    height: 22px;

}

.img\_input2{

    position: absolute;

    top: 360px;

    margin-left: 450px;

    width: 24px;

    height: 19px;

}

.section7-login{

    .main-title-btn{

        height: 58px;

        margin: 26px 0 0 0;

    }

}

.error{

    background-color: red;

    color: $color\_1;

    @extend %font1;

    padding: 10px;

    width: 86%;

    border-radius: 7px;

    text-align: center;

}

.not\_active{

    display: none;

}

.checkbox-div{

    flex-direction: row;

    font-size: 16px;

    width: 480px;

    height: 40px;

}

.section7-checkbox{

    display: grid;

    place-content: center;

    width: 20px;

    height: 20px;

    padding-bottom: 20px;

    margin: 2px 10px 0 1px;

}

.section7-checkbox::after{

    background-color: $color\_4;

}

.section7-checkbox-label{

    width: 260px;

    height: 40px;

    letter-spacing: 1.51px;

    color: $color\_5;

    margin: 6px 16px 0 25px;

    text-align: start;

    @extend %font2;

    color: $color\_5;

}

.section7-checkbox-label2{

    width: 90%;

    height: 110px;

    @extend %font2;

    font-size: 16px;

    letter-spacing: 1.51px;

    color: $color\_5;

    margin: 0 16px 0 25px;

    text-align: start;

    color: $color\_5;

    .strong{

        margin: 10px 0;

        font-weight: 900;

        color: #fff;

    }

}

.name.\_error{

    box-shadow: 0 0 15px red;

}

.check\_box\_sec{

    display: flex;

    flex-direction: row;

    margin: 15px 0 30px 0;

    input[type=checkbox] {

        -webkit-appearance: none;

        appearance: none;

        background-color: #bfbfbf;

        margin: 0;

        padding: 0;

        font: inherit;

        color: $color\_7;

        width: 28px;

        height: 28px;

        border: 0.15em solid $color\_7;

        border-radius: 7px;

        transform: translateY(-0.075em);

        display: grid;

        place-content: center;

        margin-top: 5px;

    }

    input[type=checkbox]::before {

        content: "";

        width: 28px;

        height: 28px;

        border-radius: 7px;

        transform: scale(0);

        background-image: url(../../public/Group427319235.svg);

        background-size: cover;

        transition: 120ms transform ease-in-out;

        box-shadow: inset 1em 1em var(--form-control-color);

        filter: saturate(0) brightness(70%) contrast(200%);

    }

    input[type="checkbox"]:checked::before {

        transform: scale(1);

    }

}

.img{

    width: 720px;

    height: 600px;

    margin:  35px 40px 40px 45px;

    background-size: cover;

    background-repeat: no-repeat;

}

prices/page.js

"use client"

import { useReducer, useState } from "react";

import Button from "../\_components/Button";

import styles from "./Prices.scss";

import { useRouter } from 'next/navigation';

const initialState = {

  hours: 0,

  people: 0,

  price: 0,

}

function reducer(state, action){

  switch(action.type){

    case 'setHours':

      return {

        ...state,

        hours: action.payload,

        price: state.people \* action.payload \* 5

      };

    case 'setPeople':

      return {

        ...state,

        people: action.payload,

        price: state.hours \* action.payload \* 5

      };

    case 'resetHours':

      return {

        ...state,

        hours: 0,

        price: 0,

      };

    case 'resetPeople':

      return {

        ...state,

        people: 0,

        price: 0,

      };

    default:

      throw new Error("Unknown action");

  }

}

export default function Home() {

  const [{hours, people, price}, dispatch] = useReducer(reducer, initialState);

  const router = useRouter();

  function handleSetHours(value){

    if(value < 5){

      dispatch({type: 'setHours', payload: value});

    }

  }

  function handleSetPeople(value){

    if(value < 10){

      dispatch({type: 'setPeople', payload: value});

    }

  }

  function handleMovetoLogin(){

    router.push('/login');

  }

  return (

    <>

      <div className="sections">

        <Section type={"Hours"} value={hours} setValue={handleSetHours} max={5} reset={() => dispatch({type: 'resetHours'})}/>

        <Section type={"People amount"} value={people} setValue={handleSetPeople} max={10} reset={() => dispatch({type: 'resetPeople'})}/>

      </div>

      <div className="total\_cost">

        <p className="total\_c\_desc">Your total cost for an <br/> entire group will be: </p>

        <p className="total\_c\_amount">{`$${price}`}</p>

        <Button action={handleMovetoLogin} width={200} name={"button\_template1"}>Book a table</Button>

      </div>

    </>

  );

}

function Section({type, value, setValue, max, reset}){

  return(

    <div className="confirmation\_bar  cart\_section">

          <h2 className="confirmation\_bar-caption">{type}</h2>

          <input

              type="text"

              id="promo\_check"

              className="name"

              value={value}

              onChange={(e) => setValue(e.target.value)}

              placeholder="Enter the amount"

          />

          <input

              type='range'

              className="name\_range"

              min='0'

              max={max}

              value={value}

              onChange={(e) => setValue(e.target.value)}

            />

          <div className="confirm\_bar\_cost">

              <p className="confirm\_barp\_cost \_amount">{`${type === "Hours" ? "Each group member will be charged $5.00 for each played hour" : "The price for one member per hour is multiplied by the amount of the group members"}`}</p>

          </div>

          <div className="main-title-btn\_sec">

              <Button action={reset} width={260} name={"button\_template2"}>Reset</Button>

          </div>

      </div>

  );

}

prices/Prices.scss

@import "../globals.scss";

.cart\_section{

    border-radius: 10px;

    box-shadow: 0px 4px 10px rgba(0, 0, 0, 0.15);

    background-color: $color\_4;

    display: flex;

    flex-direction: column;

    justify-content: space-between;

    width: 260px;

    padding: 30px;

    .name{

        width: 94%;

    }

    .name\_range{

        margin: 10px 0 0 50px;

        width: 60%;

    }

    .\_amount{

        text-align: justify;

    }

}

.confirmation\_bar-caption{

    width: fit-content;

    letter-spacing: 2px;

    @extend %font1;

    font-weight: 700;

    font-size: 24px;

    margin: 5px 0 15px 0;

    display: flex;

    justify-content: center;

}

.cart-title-btn2{

    transition: transform 0.5s ease;

    p{

        padding-left: 10px;

        font-size: 18px;

    }

    display: flex;

    justify-content: center;

    align-items: center;

    flex-direction: row;

    color: #000;

    background-color: #ffffff;

    border: 1.5px solid $color\_7;

    margin: 20px 0 20px 0;

    height: 64px;

    width: 100%;

}

.cart-title-btn{

    p{

        padding-left: 10px;

        font-size: 18px;

    }

    display: flex;

    justify-content: center;

    align-items: center;

    flex-direction: row;

    margin: 20px 0 20px 0;

    height: 64px;

    width: 100%;

}

.total\_cost{

    display: flex;

    justify-self: center;

    align-items: center;

    border-radius: 15px;

    box-shadow: 0px 4px 10px rgba(0, 0, 0, 0.15);

    padding: 20px;

    margin: 0 0 40px 0;

    .total\_c\_desc{

        @extend %font2;

        font-size: 14px;

        margin-right:20px;

    }

    .total\_c\_amount{

        @extend %font1;

        margin-right: 100px;

    }

}

.sections{

    margin: 80px 0;

    display: flex;

    flex-direction: row;

    justify-content: space-around;

    padding: 0 15%;

}

gamelist/page.js

"use client";

import { useEffect, useState } from "react";

import { useSelector, useDispatch } from "react-redux";

import { getGames } from "../\_lib/gameSlice.js";

import styles from "./GameList.module.scss";

import Loading from "../loading.js";

export default function Home() {

  const [selectedGameId, setSelectedGameId] = useState(null);

  const [display, setDisplay] = useState(null);

  const dispatch = useDispatch();

  useEffect(() => {

    dispatch(getGames());

  }, [dispatch]);

  const { games, isLoading, error } = useSelector((store) => store.game);

  const toggleDescription = (id) => {

    if(selectedGameId && selectedGameId === id){

      setSelectedGameId((prevId) => (prevId === id ? null : id));

    }else{

      setSelectedGameId(null)

      setTimeout(() => {

        setSelectedGameId((prevId) => (prevId === id ? null : id));

      }, 400);

    }

    setDisplay((prevId) => (prevId === id ? null : id));

  };

  if (isLoading) return <Loading />;

  if (error) return <p>Error loading games: {error}</p>;

  return (

    <div className={styles.container}>

      <h1 className={styles.Gametitle}>Game List</h1>

      <div className={styles.line}></div>

      <ul className={styles.GameList}>

        {games.map((game) => (

          <li key={game.game\_id} className={`${styles.game} ${display === game.game\_id ? styles.active : ""}`} onClick={() => toggleDescription(game.game\_id)}>

            <h2 className={styles.caption}>{game.game\_name}</h2>

            {selectedGameId === game.game\_id && (

              <>

                <p className={styles.players}>

                  <strong className={styles.desc\_cap}>Players:</strong> {game.players}

                </p>

                <p className={styles.rules}>

                  <strong className={styles.desc\_cap}>Goal of Game:</strong> {game.goal}

                </p>

                <p className={styles.rules}>

                  <strong className={styles.desc\_cap}>Rules:</strong> {game.rules}

                </p>

              </>

            )}

          </li>

        ))}

      </ul>

    </div>

  );

}

gamelist/GameList.module.scss

@import "../globals.scss";

.Gametitle{

    color: $color\_7;

    @extend %font1;

    font-size: 60px;

    margin: 60px 0 20px 0;

    letter-spacing: 5px;

    word-spacing: 10px;

    z-index: 2;

    display: flex;

    justify-self: center;

}

.line{

    background-color: $color\_6;

    width: 180px;

    height: 15px;

    border-radius: 10px;

    display: flex;

    justify-self: center;

}

.GameList{

    flex-wrap: wrap;

    display: flex;

    flex-direction: row;

    justify-content: space-around;

    padding: 0px;

    .game{

        list-style-type: none;

        position: relative;

        width: 380px;

        height: 76px;

        flex-direction: column;

        margin: 40px 20px ;

        background-color: $color\_4;

        border-radius: 15px;

        padding: 10px 40px;

        display: flex;

        align-items: center;

        justify-self: center;

        box-shadow: 1rem 1.4rem 3rem rgba(0, 0, 0, 0.356);

        transition: 0.3s ease-in-out;

        @extend %font1;

        font-weight: 200;

        font-size: 14px;

        .caption{

            @extend %font1;

            font-weight: 700;

            font-size: 30px;

            width: 100%;

            text-align: center;

            padding: 10px;

            margin: 10px 0;

            background-color: $color\_1;

            border-radius: 10px;

        }

        .desc\_cap{

            font-weight: 700;

            font-size: 18px;

            margin-right: 10px;

        }

        .players{

            width: 100%;

            transition: 0.3s ease-in-out;

            text-align: start;

        }

        .rules{

            transition: 0.3s ease-in-out;

            text-align: justify;

        }

    }

    .active{

        height: 350px;

    }

}

championship/page.js

"use client";

import { useEffect, useState } from "react";

import styles from "./Championship.module.scss";

import { useSelector, useDispatch } from "react-redux";

import { getMatches } from "@/app/\_lib/matchesSlice.js";

import CheckMatch from "../admin/champ/\_chack\_match/checkMatch.js";

function parseStage(stage) {

    const [id, round] = stage.split(".");

    return { matchId: parseInt(id, 10), round };

}

export default function Championship() {

    const [showMatch, setShowMatch] = useState(false);

    const [selectedMatch, setSelectedMatch] = useState({});

    const { matches } = useSelector((state) => state.matches);

    const dispatch = useDispatch();

    useEffect(() => {

        dispatch(getMatches());

    }, [dispatch]);

    const getRoundLabel = (i) => {

        return i === 0 ? "1/16" : i === 1 ? "1/8" : i === 2 ? "1/4" : i === 3 ? "1/2" : null;

    };

    return (

        <>

          <div className={styles.bracketWrapper}>

            <h2 className={styles.main\_title}>Monopoly Championship</h2>

            {showMatch && <CheckMatch active={showMatch} match={selectedMatch} handleActiveClick={setShowMatch} />}

            <div className={styles.bracket}>

                {/\* Left Side \*/}

                <div className={styles.side}>

                    {[...Array(4)].map((\_, i) => (

                        <div className={styles.round} key={`left-${i}`}>

                            {[...Array(2 \*\* (3 - i))].map((\_, j) => {

                                const matchId = i \* 10 + j + 100;

                                const round = getRoundLabel(i);

                                const isFilled = matches.some((match) => {

                                    const { matchId: id, round: r } = parseStage(match.stage);

                                    return id === matchId && r === round;

                                });

                                return (

                                    <div

                                        className={`${styles.match} ${isFilled ? styles.filled : ""}`}

                                        key={i \* 10 + j}

                                        onClick={() => {

                                            const stage = `${matchId}.${round}`;

                                            const matched = matches.find((match) => match.stage === stage);

                                            if (matched) {

                                                setSelectedMatch(matched);

                                                setShowMatch(true);

                                            }

                                        }}

                                    >

                                        {round}

                                    </div>

                                );

                            })}

                        </div>

                    ))}

                </div>

                {/\* Final \*/}

                <div className={styles.final}>

                    <div className={styles.match}

                        onClick={() => {

                            const stage = `${matchId}.${round}`;

                            const matched = matches.find((match) => match.stage === stage);

                            if (matched) {

                                setSelectedMatch(matched);

                                setShowMatch(true);

                            }

                        }}

                    > Final</div>

                </div>

                {/\* Right Side \*/}

                <div className={styles.side\_2}>

                    {[...Array(4)].map((\_, i) => (

                        <div className={styles.round} key={`right-${i}`}>

                            {[...Array(2 \*\* (3 - i))].map((\_, j) => {

                                const matchId = i \* 10 + j;

                                const round = getRoundLabel(i);

                                const isFilled = matches.some((match) => {

                                    const { matchId: id, round: r } = parseStage(match.stage);

                                    return id === matchId && r === round;

                                });

                                return (

                                    <div

                                        className={`${styles.match} ${isFilled ? styles.filled : ""}`}

                                        key={i \* 10 + j}

                                        onClick={() => {

                                            const stage = `${matchId}.${round}`;

                                            const matched = matches.find((match) => match.stage === stage);

                                            if (matched) {

                                                setSelectedMatch(matched);

                                                setShowMatch(true);

                                            }

                                        }}

                                    >

                                        {round}

                                    </div>

                                );

                            })}

                        </div>

                    ))}

                </div>

            </div>

        </div>

        </>

    );

}

championship/Championship.module.scss

@import "../globals.scss";

.bracketWrapper {

    padding: 20px;

    display: flex;

    justify-self: center;

    flex-direction: column;

    align-items: center;

    width: 90%;

}

.main\_title{

    @extend %font1;

    font-size: 50px;

    letter-spacing: 5px;

    font-weight: 900;

    color: $color\_5;

    margin: 15px 0 20px 0 ;

    font-weight: bolder;

    padding-left: 7px;

    color: $color\_5;

}

.bracket {

    display: flex;

    flex-direction: row;

    justify-content: center;

    width: 100%;

    flex-wrap: wrap;

    gap: 10px;

}

.side {

    width: 40vw;

    display: flex;

    flex-direction: row;

    justify-content: space-around;

    gap: 10px;

    height: 50vh;

}

.side\_2 {

    width: 40vw;

    display: flex;

    flex-direction: row-reverse;

    justify-content: space-around;

    gap: 10px;

    height: 50vh;

}

.round {

    display: flex;

    flex-direction: column;

    justify-content: space-around;

    gap: 10px;

    min-width: 70px;

}

.match {

    background-color: #bebebe;

    display: flex;

    align-items: center;

    justify-content: center;

    border-radius: 6px;

    width: 100%;

    height: 40px;

    transition: all 0.3s ease;

}

.match:hover {

    background-color: $color\_4;

    transform: scale(1.05);

}

.final {

    display: flex;

    align-items: center;

    justify-content: center;

    padding: 0 20px;

    .match {

        background-color: $color\_8;

        width: 80px;

        height: 40px;

    }

}

.filled{

    background-color: $color\_4;

}

// Responsive design

@media (max-width: 768px) {

    .round {

        min-width: 35px;

        .match {

            height: 25px;

        }

    }

    .final .match {

        width: 60px;

        height: 35px;

    }

}

@media (max-width: 480px) {

    .bracket {

        flex-direction: column;

        align-items: center;

    }

    .side {

        flex-direction: column;

        align-items: center;

    }

    .round {

        flex-direction: row;

        flex-wrap: wrap;

        justify-content: center;

    }

    .match {

        width: 80px;

        margin: 2px;

    }

    .final {

        padding: 20px 0;

    }

}

\_lib/accountSlice.js

import { createSlice } from "@reduxjs/toolkit";

const initialState = {

    user: {

        userId: "",

        email: "",

        name: "",

        phone: 0,

        password: "",

        access: "",

    },

    isLoading: false,

    error: null,

};

const accountSlice = createSlice({

    name: "account",

    initialState,

    reducers: {

        setAccount: (state, action) => {

            state.user = action.payload;

        },

        setLoading: (state, action) => {

            state.isLoading = action.payload;

        },

        setError: (state, action) => {

            state.error = action.payload;

        },

        resetAccount: (state, action) => {

            state.user = {};

        }

    },

});

export const { setAccount, setLoading, setError, resetAccount } = accountSlice.actions;

export default accountSlice.reducer;

export function Register(user) {

    return async (dispatch) => {

        try {

            dispatch(setLoading(true));

            dispatch(setError(null));

            const res = await fetch("http://localhost:3001/register", {

                method: "POST",

                headers: {

                    "Content-Type": "application/json",

                },

                body: JSON.stringify(user),

            });

            const data = await res.json();

            if (!res.ok) throw new Error(data.error || "Registration failed");

        } catch (error) {

            dispatch(setError(error.message));

        } finally {

            dispatch(setLoading(false));

        }

    };

}

export function Login(user) {

    return async (dispatch) => {

        try {

            dispatch(setLoading(true));

            dispatch(setError(null));

            const res = await fetch("http://localhost:3001/login", {

                method: "POST",

                headers: {

                    "Content-Type": "application/json",

                },

                body: JSON.stringify(user),

            });

            const data = await res.json();

            if (!res.ok) throw new Error(data.error || "Login failed");

            const transformedUser = {

                userId: data.userId,

                email: data.email || data.player\_email,

                name: data.name || "",

                phone: data.phone || 0,

                password: user.password,

                access: data.access || "",

            };

            console.log(transformedUser);

            dispatch(setAccount(transformedUser));

        } catch (error) {

            dispatch(setError(error.message));

        } finally {

            dispatch(setLoading(false));

        }

    };

}

export function userLogout(user) {

    return async (dispatch) => {

        try {

            dispatch(setLoading(true));

            dispatch(setError(null));

            const res = await fetch("http://localhost:3001/logout", {

                method: "POST",

                headers: {

                    "Content-Type": "application/json",

                },

                body: JSON.stringify(user),

            });

            const data = await res.json();

            dispatch(resetAccount());

        } catch (error) {

            dispatch(setError(error.message));

        } finally {

            dispatch(setLoading(false));

        }

    };

}

export function updateUser( name, phone, password ) {

    return async (dispatch, getState) => {

        try {

            dispatch(setLoading(true));

            dispatch(setError(null));

            console.log(name, phone, password);

            const { account: { user } } = getState();

            const res = await fetch("http://localhost:3001/update\_user", {

            method: "PUT",

            headers: {

                "Content-Type": "application/json",

            },

            body: JSON.stringify({

                email: user.email,

                userId: user.userId,

                name,

                phone,

                password,

            }),

            });

            const data = await res.json();

            if (!res.ok) throw new Error(data.error || "Update failed");

            dispatch(setAccount({

            ...user,

            name: name,

            phone: phone,

            password: password,

            }));

        } catch (error) {

            dispatch(setError(error.message));

        } finally {

            dispatch(setLoading(false));

        }

    };

}

\_lib/bookingsSlice.js

import { createSlice } from "@reduxjs/toolkit";

const initialState = {

    bookings: [],

    bookingIsLoading: false,

    bookingError: null,

};

const bookingSlice = createSlice({

    name: "booking",

    initialState,

    reducers: {

        setBookings: (state, action) => {

            state.bookings = action.payload;

        },

        setLoading: (state, action) => {

            state.bookingIsLoading = action.payload;

        },

        setError: (state, action) => {

            state.bookingError = action.payload;

        },

        resetBookings: (state, action) => {

            state.bookings = [];

        }

    },

});

export const { setBookings, setLoading, setError, resetBookings } = bookingSlice.actions;

export default bookingSlice.reducer;

export function checkBookings(user) {

    return async (dispatch) => {

        try {

            dispatch(setLoading(true));

            dispatch(setError(null));

            const res = await fetch("http://localhost:3001/bookings", {

                method: "POST",

                headers: {

                    "Content-Type": "application/json",

                },

                body: JSON.stringify({

                    email: user.email,

                    userId: user.userId,

                }),

            });

            if(!res.ok)throw new Error("Something went wrong with fetching bookings");

            const data = await res.json();

            dispatch(setBookings(data.bookings));

        }catch (error) {

            dispatch(setError(error.message));

        } finally {

            dispatch(setLoading(false));

        }

    };

};

export function checkAdminBookings(email, userId, year) {

    return async (dispatch) => {

        try {

            dispatch(setLoading(true));

            dispatch(setError(null));

            const res = await fetch("http://localhost:3001/bookings\_admin", {

                method: "POST",

                headers: {

                    "Content-Type": "application/json",

                },

                body: JSON.stringify({

                    email,

                    userId,

                    year,

                }),

            });

            if (!res.ok) throw new Error("Something went wrong with fetching bookings");

            const data = await res.json();

            dispatch(setBookings(data.bookings));

        } catch (error) {

            dispatch(setError(error.message));

        } finally {

            dispatch(setLoading(false));

        }

    };

};

export function createBooking(email, userId, date, time, period, game, players,) {

    return async (dispatch) => {

        try {

            dispatch(setLoading(true));

            dispatch(setError(null));

            const res = await fetch("http://localhost:3001/create\_booking", {

                method: "POST",

                headers: {

                    "Content-Type": "application/json",

                },

                body: JSON.stringify({

                    email,

                    userId,

                    date,

                    time,

                    period,

                    game,

                    players,

                }),

            });

            if(!res.ok)throw new Error("Something went wrong with fetching bookings");

        }catch (error) {

            dispatch(setError(error.message));

        } finally {

            dispatch(setLoading(false));

        }

    };

};

export function deleteBooking(email, userId, id) {

    return async (dispatch) => {

        try {

            dispatch(setLoading(true));

            dispatch(setError(null));

            const res = await fetch("http://localhost:3001/booking\_delete", {

                method: "POST",

                headers: {

                    "Content-Type": "application/json",

                },

                body: JSON.stringify({

                    id: id,

                    email,

                    userId,

                }),

            });

            if(!res.ok)throw new Error("Something went wrong with fetching bookings");

            const data = await res.json();

            dispatch(checkAdminBookings());

        }catch (error) {

            dispatch(setError(error.message));

        } finally {

            dispatch(setLoading(false));

        }

    };

}

\_lib/gameSlice.js

import { createSlice } from "@reduxjs/toolkit";

const initialState = {

    games: [],

    isLoading: false,

    error: null,

};

const gameSlice = createSlice({

    name: "game",

    initialState,

    reducers: {

        setGames: (state, action) => {

            state.games = action.payload;

        },

        setLoading: (state, action) => {

            state.isLoading = action.payload;

        },

        setError: (state, action) => {

            state.error = action.payload;

        },

    },

});

export const { setGames, setLoading, setError } = gameSlice.actions;

export default gameSlice.reducer;

export function getGames() {

    return async (dispatch) => {

        try {

            dispatch(setLoading(true));

            dispatch(setError(null));

            const res = await fetch("http://localhost:3001/games"); // Fetch the games data from backend

            const data = await res.json();

            // Dispatch action to set games in the Redux state

            dispatch(setGames(data));

        } catch (error) {

            // If there's an error, dispatch the error message

            dispatch(setError("Error fetching games"));

        } finally {

            dispatch(setLoading(false)); // Set loading to false after the fetch

        }

    };

}

export function createGame(email, userId, game\_name, players, goal, rules) {

    return async (dispatch) => {

        try {

            dispatch(setLoading(true));

            dispatch(setError(null));

            const res = await fetch("http://localhost:3001/game\_create", {

                method: "POST",

                headers: {

                    "Content-Type": "application/json",

                },

                body: JSON.stringify({

                    email,

                    userId,

                    game\_name,

                    status: "new",

                    players,

                    goal,

                    rules,

                }),

            });

            if (!res.ok) {

                const errorData = await res.json();

                throw new Error(errorData.error || "Failed to create game.");

            }

            const data = await res.json();

            dispatch(getGames());

        } catch (error) {

            dispatch(setError(error.message));

        } finally {

            dispatch(setLoading(false));

        }

    };

}

export function deleteGame(email, userId, id) {

    return async (dispatch) => {

        try {

            dispatch(setLoading(true));

            dispatch(setError(null));

            const res = await fetch("http://localhost:3001/game\_delete", {

                method: "POST",

                headers: {

                    "Content-Type": "application/json",

                },

                body: JSON.stringify({

                    email,

                    userId,

                    game\_id: id,

                }),

            });

            if(!res.ok)throw new Error("Something went wrong with fetching bookings");

            const data = await res.json();

            dispatch(getGames());

        }catch (error) {

            dispatch(setError(error.message));

        } finally {

            dispatch(setLoading(false));

        }

    };

}

\_lib/matchesSlice.js

import { createSlice } from "@reduxjs/toolkit";

const initialState = {

    matches: [],

    isLoading: false,

    error: null,

};

const matchSlice = createSlice({

    name: "match",

    initialState,

    reducers: {

        setMatches: (state, action) => {

            state.matches = action.payload;

        },

        setLoading: (state, action) => {

            state.isLoading = action.payload;

        },

        setError: (state, action) => {

            state.error = action.payload;

        },

    },

});

export const { setMatches, setLoading, setError } = matchSlice.actions;

export default matchSlice.reducer;

export function getMatches() {

    return async (dispatch) => {

        try {

            dispatch(setLoading(true));

            dispatch(setError(null));

            const res = await fetch("http://localhost:3001/matches"); // Fetch the games data from backend

            const data = await res.json();

            dispatch(setMatches(data));

        } catch (error) {

            dispatch(setError("Error fetching matches"));

        } finally {

            dispatch(setLoading(false));

        }

    };

}

export function createMatch(email, userId, playerCount, players, winner, date, time, stage) {

    return async (dispatch) => {

        try {

            dispatch(setLoading(true));

            dispatch(setError(null));

            const res = await fetch("http://localhost:3001/match\_create", {

                method: "POST",

                headers: {

                    "Content-Type": "application/json",

                },

                body: JSON.stringify({

                    email,

                    userId,

                    playerCount,

                    players,

                    winner,

                    date,

                    time,

                    stage,

                }),

            });

            if (!res.ok) {

                const errorData = await res.json();

                throw new Error(errorData.error || "Failed to create game.");

            }

            const data = await res.json();

            //dispatch(getMatches());

        } catch (error) {

            dispatch(setError(error.message));

        } finally {

            dispatch(setLoading(false));

        }

    };

}

\_lib/paymentsSlice.js

import { createSlice } from "@reduxjs/toolkit";

const initialState = {

    payments: [],

    isLoading: false,

    error: null,

};

const paymentSlice = createSlice({

    name: "payment",

    initialState,

    reducers: {

        setPayments: (state, action) => {

            state.payments = action.payload;

        },

        setLoading: (state, action) => {

            state.isLoading = action.payload;

        },

        setError: (state, action) => {

            state.error = action.payload;

        },

        resetPayments: (state, action) => {

            state.payments = [];

        }

    },

});

export const { setPayments, setLoading, setError, resetPayments } = paymentSlice.actions;

export default paymentSlice.reducer;

export function checkAllPayments(email, userId, year) {

    return async (dispatch) => {

        try {

            dispatch(setLoading(true));

            dispatch(setError(null));

            const res = await fetch("http://localhost:3001/checkAll\_payments", {

                method: "POST",

                headers: {

                    "Content-Type": "application/json",

                },

                body: JSON.stringify({

                    email,

                    userId,

                    year,

                }),

            });

            const data = await res.json();

            if (!res.ok) throw new Error(data.error || "Payments check failed");

            dispatch(setPayments(data));

        } catch (error) {

            dispatch(setError(error.message));

        } finally {

            dispatch(setLoading(false));

        }

    };

}

export function updatePayment( email, userId, id, status, amount, card\_or\_cash ) {

    return async (dispatch) => {

        try {

            dispatch(setLoading(true));

            dispatch(setError(null));

            const res = await fetch("http://localhost:3001/update\_payment", {

            method: "PUT",

            headers: {

                "Content-Type": "application/json",

            },

            body: JSON.stringify({

                email,

                userId,

                payment\_id: id,

                status,

                amount,

                card\_or\_cash,

            }),

            });

            const data = await res.json();

            if (!res.ok) throw new Error(data.error || "Update failed");

            dispatch(checkAllPayments());

        } catch (error) {

            dispatch(setError(error.message));

        } finally {

            dispatch(setLoading(false));

        }

    };

}

\_lib/tableSlice.js

import { createSlice } from "@reduxjs/toolkit";

const initialState = {

    tables: [],

    isLoading: false,

    error: null,

};

const tableSlice = createSlice({

    name: "account",

    initialState,

    reducers: {

        setTables: (state, action) => {

            state.tables = action.payload;

        },

        setLoading: (state, action) => {

            state.isLoading = action.payload;

        },

        setError: (state, action) => {

            state.error = action.payload;

        },

        resetTables: (state, action) => {

            state.tables = [];

        }

    },

});

export const { setTables, setLoading, setError, resetTables } = tableSlice.actions;

export default tableSlice.reducer;

export function getTables(email, userId) {

    return async (dispatch) => {

        try {

            dispatch(setLoading(true));

            dispatch(setError(null));

            const res = await fetch("http://localhost:3001/tables", {

                method: "POST",

                headers: {

                    "Content-Type": "application/json",

                },

                body: JSON.stringify({

                    email,

                    userId,

                }),

            });

            const data = await res.json();

            dispatch(setTables(data));

        } catch (error) {

            dispatch(setError("Error fetching games"));

        } finally {

            dispatch(setLoading(false));

        }

    };

}

\_general\_state/Provider.js

"use client";

import { Provider } from "react-redux";

import store from "./store";

export default function ReduxProvider({ children }) {

    return <Provider store={store}>

        {children}

    </Provider>;

}

\_general\_state/store.js

import { configureStore } from "@reduxjs/toolkit"

import gameReducer from "../\_lib/gameSlice.js";

import tableReducer from "../\_lib/tableSlice.js";

import accountReducer from "../\_lib/accountSlice.js";

import bookingReducer from "../\_lib/bookingsSlice.js";

import paymentReducer from "../\_lib/paymentsSlice.js";

import matchReducer from "../\_lib/matchesSlice.js";

const store = configureStore({

    reducer: {

        game: gameReducer,

        tables: tableReducer,

        account: accountReducer,

        bookings: bookingReducer,

        payments: paymentReducer,

        matches: matchReducer,

    }

});

export default store;

\_components/adminnavigation/AdminNavigation.js

'use client';

import Link from "next/link";

import styles from "./AdminNavigation.module.scss";

import Image from "next/image";

import logo from "@/public/cover.png";

import { usePathname } from 'next/navigation';

import { useState, useEffect } from "react";

export default function Navigation() {

    const pathname = usePathname();

    const [currentPage, setCurrentPage] = useState(0);

    const [ulStyle, setUlStyle] = useState("nav\_sec\_0")

    useEffect(() => {

        if (pathname.includes("games\_state\_list")) {

            setCurrentPage(1);

            setUlStyle("nav\_sec\_1");

        } else if (pathname.includes("touse")) {

            setCurrentPage(2);

            setUlStyle("nav\_sec\_2");

        } else if (pathname.includes("logout")) {

            setCurrentPage(3);

            setUlStyle("nav\_sec\_3");

        } else {

            setCurrentPage(0);

            setUlStyle("nav\_sec\_0");

        }

    }, [pathname]);

    return (

        <header className={styles.header}>

            <Link href="/" className={styles.logo\_sect\_1}>

                <Image

                    className={`${styles.logo\_img} ${styles.type2}`}

                    src={logo}

                    alt="Logo"

                />

            </Link>

            <ul className={styles[ulStyle]}>

                <li className={styles.nav\_li}>

                    <Link

                        className={`${styles.nav\_link} ${currentPage > 0 ? styles.active : ''}`}

                        href="/admin/"

                    >

                        Main

                    </Link>

                </li>

                <li className={styles.nav\_li}>

                    <Link

                        className={`${styles.nav\_link} ${currentPage > 1 ? styles.active : ''}`}

                        href="/admin/games\_state\_list"

                    >

                        Games state

                    </Link>

                </li>

                <li className={styles.nav\_li}>

                    <Link

                        className={`${styles.nav\_link} ${currentPage > 2 ? styles.active : ''}`}

                        href="/user"

                    >

                        User account

                    </Link>

                </li>

                <li className={styles.nav\_li}>

                    <Link

                        className={`${styles.nav\_link} ${currentPage > 2 ? styles.active : ''}`}

                        href="/admin/logout"

                    >

                        Log out

                    </Link>

                </li>

            </ul>

        </header>

    )

}

\_components/adminnavigation/AdminNavigation.module.scss

@import "../../globals.scss";

.header{

    display: flex;

    flex-direction: row;

    justify-content: space-between;

    align-items: center;

    margin: 15px 0 0 0;

    a{

        margin-left: 10px;

        height: 100px;

    }

    .nav\_sec\_0{

        background-color: $color\_1;

        margin: 0;

        height: 100;

        list-style: none;

        display: flex;

        flex-direction: row;

        justify-self: flex-end;

        align-items: center;

        justify-content: space-around;

        width: 70%;

        cursor: pointer;

    }

    .nav\_li{

        z-index: 1;

        padding: 10px;

        .active{

            color:$color\_5;

            transition: 0.3s;

        }

    }

    .nav\_link{

        color: $color\_4;

        text-decoration: none;

        @extend %font1;

        transition: 0.3s;

    }

    .nav\_link:hover{

        color: $color\_5;

        transition: 0.3s;

    }

    .nav\_sec\_0:before {

        content: ' ';

        display: block;

        background: $color\_4;

        width: 18%;

        height: 100px;

        position: absolute;

        top: 15;

        left: 216px;

        transform: skew(40deg);

        border-radius: 80px 15px 20px 0px  ;

        z-index: 0;

        transition: 0.3s;

    }

    .nav\_sec\_1{

        background-color: $color\_1;

        margin: 0;

        height: 100;

        list-style: none;

        display: flex;

        flex-direction: row;

        justify-self: flex-end;

        align-items: center;

        justify-content: space-around;

        width: 70%;

        cursor: pointer;

    }

    .nav\_sec\_1:before {

        content: ' ';

        display: block;

        background: $color\_4;

        width: 33%;

        height: 100px;

        position: absolute;

        top: 15;

        left: 216px;

        transform: skew(40deg);

        border-radius: 80px 15px 15px 0px  ;

        z-index: 0;

        transition: 0.3s;

    }

    .nav\_sec\_2{

        background-color: $color\_1;

        margin: 0;

        height: 100;

        list-style: none;

        display: flex;

        flex-direction: row;

        justify-self: flex-end;

        align-items: center;

        justify-content: space-around;

        width: 70%;

        cursor: pointer;

    }

    .nav\_sec\_2:before {

        content: ' ';

        display: block;

        background: $color\_4;

        width: 52%;

        height: 100px;

        position: absolute;

        top: 15;

        left: 216px;

        transform: skew(40deg);

        border-radius: 80px 15px 15px 0px  ;

        z-index: 0;

        transition: 0.3s;

    }

    .nav\_sec\_3{

        background-color: $color\_1;

        margin: 0;

        height: 100;

        list-style: none;

        display: flex;

        flex-direction: row;

        justify-self: flex-end;

        align-items: center;

        justify-content: space-around;

        width: 70%;

        cursor: pointer;

    }

    .nav\_sec\_3:before {

        content: ' ';

        display: block;

        background: $color\_4;

        width: 71%;

        height: 100px;

        position: absolute;

        top: 15;

        left: 216px;

        transform: skew(40deg);

        border-radius: 80px 15px 15px 0px  ;

        z-index: 0;

        transition: 0.3s;

    }

    .logo\_img{

        height: 100px;

        width: 240px;

        padding-bottom: 0;

        padding-right: 60px;

        transition: 0.3s;

        margin-left: 10px;

    }

    .type1{

        border-radius: 20px 20px 10px 0px;

        transition: 0.3s;

    }

    .type2{

        border-radius: 20px 20px 10px 20px;

        transition: 0.3s;

    }

}

\_components/Footer/Footer.js

'use client';

import styles from "./Footer.scss";

import Image from "next/image";

import instagram from "@/public/instagram.png";

import facebook from "@/public/facebook.png";

import logo from "@/public/logo.svg";

import Button from "../Button";

import { useRouter } from 'next/navigation';

import store from "@/app/\_general\_state/store";

import \* as XLSX from "xlsx";

import { saveAs } from "file-saver";

import { useSelector, useDispatch } from "react-redux";

import { checkAdminBookings } from "@/app/\_lib/bookingsSlice";

import { checkAllPayments } from "@/app/\_lib/paymentsSlice";

import { getGames } from "@/app/\_lib/gameSlice";

import { useState } from "react";

export default function Footer() {

    const dispatch = useDispatch();

    const router = useRouter();

    const { user } = useSelector((state) => state.account);

    const [selectedDate, setSelectedDate] = useState(() => {

        const today = new Date().toISOString().split("T")[0];

        return today;

    });

    function handleNav() {

        router.push('/prices');

    }

    function handleAdmin(){

        router.push('/admin');

    }

    function handleStatistics(){

        router.push('/admin/statistic');

    }

    const handleExport = async () => {

        try {

            await Promise.all([

                dispatch(checkAdminBookings(user.email, user.userId, new Date(selectedDate).getFullYear())),

                dispatch(checkAllPayments(user.email, user.userId, new Date(selectedDate).getFullYear())),

                dispatch(getGames()),

            ]);

            const state = store.getState();

            const bookings = state.bookings.bookings;

            const payments = state.payments.payments;

            const games = state.game.games;

            // Proceed with data export

            const combinedData = bookings.map((booking) => {

                const payment = payments.find(p => Number(p.payment\_id) === Number(booking.payment\_id));

                const game = games.find(g => g.game\_name === booking.game\_name);

                return {

                    BookingID: booking.booking\_id,

                    Date: new Date(booking.date).toLocaleDateString(),

                    Time: booking.time,

                    Duration: booking.period,

                    TableID: booking.table\_id,

                    GameName: booking.game\_name,

                    GameStatus: game?.status || "N/A",

                    GroupID: booking.group\_id,

                    PlayersCount: booking.players\_amount,

                    IsFull: booking.is\_full ? "Yes" : "No",

                    Members: booking.member\_emails.join(", "),

                    PaymentAmount: payment?.amount ?? "N/A",

                    PaymentStatus: payment?.status ?? "N/A",

                    PaidWithCard: payment ? (payment.card\_or\_cash ? "Card" : "Cash") : "N/A",

                };

            });

            const worksheet = XLSX.utils.json\_to\_sheet(combinedData);

            const workbook = XLSX.utils.book\_new();

            XLSX.utils.book\_append\_sheet(workbook, worksheet, "Bookings");

            const excelBuffer = XLSX.write(workbook, { bookType: "xlsx", type: "array" });

            const blob = new Blob([excelBuffer], { type: "application/octet-stream" });

            saveAs(blob, "bookings\_data.xlsx");

        } catch (err) {

            console.error("Failed to export bookings", err);

            alert("There was an error fetching the data. Please try again.");

        }

    };

    return (

        <footer>

            <div className={`${"footer\_section1"} ${user.access ==="manager"? "" : "sect\_1"}`}>

                <Image className="footer\_logo\_img" src={logo} width={200} height={200} alt=""/>

                <p className="part0-footer">Solution designed and builded by the SimpleStudent company for all your needs</p>

            </div>

            {user.access === "manager" &&

                <div className="footer\_section5">

                    <Button

                        name={"button\_template1"}

                        width={200}

                        action={handleExport}

                    >

                        Export bookings

                    </Button>

                </div>

            }

            {user.access  &&

                <>

                    <div className="footer\_section2">

                        <Button

                            name={"button\_template1"}

                            width={200}

                            action={handleAdmin}

                        >

                            Admin

                        </Button>

                        {user.access === "manager" &&

                            <Button

                                name={"button\_template1"}

                                width={200}

                                action={handleStatistics}

                            >

                                Statistics

                            </Button>

                        }

                    </div>

                </>

            }

            <div className="footer\_section">

                <h3>News</h3>

                <p className="part1-footer" >About us</p>

                <p className="part1-footer" >Teems of Servise</p>

                <p className="part1-footer" >Priacy Policy</p>

            </div>

            <div className="footer\_section">

                <h3>Explore our prices</h3>

                <div className="footer\_subsc-section">

                    <Button action={handleNav} name={"button\_template1"} width={"200px"}>Calculate</Button>

                </div>

                <div className="contact\_img\_section">

                <a href="https://www.instagram.com/" target="\_blank" rel="noopener referrer">

                    <Image className="footer\_contact\_img" src={instagram} alt="Instagram" />

                </a>

                <a href="https://www.facebook.com/" target="\_blank" rel="noopener referrer">

                    <Image className="footer\_contact\_img" src={facebook} alt="Facebook" />

                </a>

                </div>

            </div>

        </footer>

    )

}

\_components/Footer/Footer.scss

@import "../../globals.scss";

footer{

    font-family: 'Montserrat', sans-serif;

    display: flex;

    flex-direction: row;

    justify-content: space-evenly;

    height: 200px;

    background-color: #dfdfdf;

    padding: 0 10% 0 10%;

}

.footer\_section{

    display: flex;

    flex-direction: column;

    align-items: start;

    margin: 50px 0 0 30px;

    h3{

        margin: 0 0 15px 0;

    }

}

.footer\_section1{

    display: flex;

    flex-direction: column;

    align-items: start;

    margin: 0px 0 0 30px;

    h3{

        margin: 0 0 15px 0;

    }

}

.footer\_section2{

    display: flex;

    flex-direction: column;

    justify-content: center;

    align-content: center;

    margin: 0 5% 0 0 ;

    .button\_template1{

        margin: 10px 0;

    }

}

.footer\_section5{

    display: flex;

    flex-direction: column;

    justify-content: center;

    align-content: center;

    margin: 0 0 0 30px;

    .button\_template1{

        margin: 10px 0;

    }

}

.footer\_logo\_img{

    height: 120px;

}

.sect\_1{

    margin: 0 30% 0 0;

}

.part1-footer{

    color: #8A8A8A;

    width: 150px;

    font-size: 12px;

    font-weight: 500;

    line-height: 17px;

    font-weight: lighter;

    margin: 0 0 10px 0;

}

.part0-footer{

    color: #777777;

    width: 200px;

    font-size: 14px;

    font-weight: 500;

    line-height: 17px;

    font-weight: lighter;

    padding-left: 20px;

    margin-top: 0;

}

.uspace\_logo\_footer{

    width: fit-content;

    margin: 15px 0 0 0 ;

    background-color: $color\_3;

    -webkit-background-clip: text;

    -webkit-text-fill-color: transparent;

    background-clip: text;

}

.footer\_subsc-section{

    display: flex;

    flex-direction: row;

}

.footer-name{

    padding-left:10px;

    width: 180px;

    height: 40px;

    font-family: 'Montserrat';

    font-style: normal;

    font-weight: 500;

    font-size: 12px;

    line-height: 20px;

    border: 1.5px solid #bababa;

    border-radius: 7px;

    margin: 0 5px 0 0;

    outline: none;

}

.footer-name:-webkit-autofill {

    -webkit-box-shadow: inset 0 0 0 50px #fff!important;

   box-shadow: inset 0 0 0 50px #fff!important;

}

.footer-name:placeholder-shown{

    color: #bababa;

    font-size: 12px;

}

.button-subscription{

    width: 102px;

    height: 44px;

    background: #F15D38;

    box-shadow: 0px 4px 10px rgba(0, 0, 0, 0.15);

    border-radius: 5px;

    border: none;

    font-family: 'Inter';

    font-style: normal;

    font-weight: 600;

    font-size: 12px;

    line-height: 30px;

    margin: 0 0 15px 0;

    color: #FFFFFF;

}

.contact\_img\_section{

    display: flex;

    flex-direction: row;

    margin-top: 10px;

}

.footer\_contact\_img{

    padding-right: 10px;

    cursor: pointer;

}

.footer\_logo\_img{

    width: 220px;

}

\_components/Header/Header.js

"use client"

import Navigation from "../visitornavigation/Navigation.js"

import UserNavigation from "../usernavigation/UserNavigation.js"

import AdminNavigation from "../adminnavigation/AdminNavigation.js"

import { usePathname } from "next/navigation.js";

import { useSelector } from "react-redux";

import ManagerNavigation from "../../\_components/managernavigation/ManagerNavigation.js"

export default function Header() {

    const pathname = usePathname();

    const { user } = useSelector((state) => state.account);

    return (

        <header>

            {

                pathname.includes("user") ? <UserNavigation/> :

                pathname.includes("admin") ? (user.access === "manager"?

                    <ManagerNavigation/> :

                    <AdminNavigation/>

                ) : <Navigation/>

            }

        </header>

    )

}

\_components/managernavigation/ManagerNavigation.js

'use client';

import Link from "next/link";

import styles from "./ManagerNavigation.module.scss";

import Image from "next/image";

import logo from "@/public/cover.png";

import { usePathname } from 'next/navigation';

import { useState, useEffect } from "react";

export default function Navigation() {

    const pathname = usePathname();

    const [currentPage, setCurrentPage] = useState(0);

    const [ulStyle, setUlStyle] = useState("nav\_sec\_0")

    useEffect(() => {

        if (pathname.includes("games\_state\_list")) {

            setCurrentPage(1);

            setUlStyle("nav\_sec\_1");

        } else if (pathname.includes("champ")) {

            setCurrentPage(2);

            setUlStyle("nav\_sec\_2");

        } else if (pathname.includes("touse")) {

            setCurrentPage(3);

            setUlStyle("nav\_sec\_3");

        } else if (pathname.includes("logout")) {

            setCurrentPage(4);

            setUlStyle("nav\_sec\_4");

        }else {

            setCurrentPage(0);

            setUlStyle("nav\_sec\_0");

        }

    }, [pathname]);

    return (

        <header className={styles.header}>

            <Link href="/" className={styles.logo\_sect\_1}>

                <Image

                    className={`${styles.logo\_img} ${styles.type2}`}

                    src={logo}

                    alt="Logo"

                />

            </Link>

            <ul className={styles[ulStyle]}>

                <li className={styles.nav\_li}>

                    <Link

                        className={`${styles.nav\_link} ${currentPage > 0 ? styles.active : ''}`}

                        href="/admin/"

                    >

                        Main

                    </Link>

                </li>

                <li className={styles.nav\_li}>

                    <Link

                        className={`${styles.nav\_link} ${currentPage > 1 ? styles.active : ''}`}

                        href="/admin/games\_state\_list"

                    >

                        Games state

                    </Link>

                </li>

                <li className={styles.nav\_li}>

                    <Link

                        className={`${styles.nav\_link} ${currentPage > 2 ? styles.active : ''}`}

                        href="/admin/champ"

                    >

                        Championship

                    </Link>

                </li>

                <li className={styles.nav\_li}>

                    <Link

                        className={`${styles.nav\_link} ${currentPage > 3 ? styles.active : ''}`}

                        href="/user"

                    >

                        User account

                    </Link>

                </li>

                <li className={styles.nav\_li}>

                    <Link

                        className={`${styles.nav\_link} ${currentPage > 4 ? styles.active : ''}`}

                        href="/admin/logout"

                    >

                        Log out

                    </Link>

                </li>

            </ul>

        </header>

    )

}

\_components/managernavigation/ManagerNavigation.module.scss

@import "../../globals.scss";

.header{

    display: flex;

    flex-direction: row;

    justify-content: space-between;

    align-items: center;

    margin: 15px 0 0 0;

    a{

        margin-left: 10px;

        height: 100px;

    }

    .nav\_sec\_0{

        background-color: $color\_1;

        margin: 0;

        height: 100;

        list-style: none;

        display: flex;

        flex-direction: row;

        justify-self: flex-end;

        align-items: center;

        justify-content: space-around;

        width: 70%;

        cursor: pointer;

    }

    .nav\_li{

        z-index: 1;

        padding: 10px;

        .active{

            color:$color\_5;

            transition: 0.3s;

        }

    }

    .nav\_link{

        color: $color\_4;

        text-decoration: none;

        @extend %font1;

        transition: 0.3s;

    }

    .nav\_link:hover{

        color: $color\_5;

        transition: 0.3s;

    }

    .nav\_sec\_0:before {

        content: ' ';

        display: block;

        background: $color\_4;

        width: 16%;

        height: 100px;

        position: absolute;

        top: 15;

        left: 216px;

        transform: skew(40deg);

        border-radius: 80px 15px 20px 0px  ;

        z-index: 0;

        transition: 0.3s;

    }

    .nav\_sec\_1{

        background-color: $color\_1;

        margin: 0;

        height: 100;

        list-style: none;

        display: flex;

        flex-direction: row;

        justify-self: flex-end;

        align-items: center;

        justify-content: space-around;

        width: 70%;

        cursor: pointer;

    }

    .nav\_sec\_1:before {

        content: ' ';

        display: block;

        background: $color\_4;

        width: 28.5%;

        height: 100px;

        position: absolute;

        top: 15;

        left: 216px;

        transform: skew(40deg);

        border-radius: 80px 15px 15px 0px  ;

        z-index: 0;

        transition: 0.3s;

    }

    .nav\_sec\_2{

        background-color: $color\_1;

        margin: 0;

        height: 100;

        list-style: none;

        display: flex;

        flex-direction: row;

        justify-self: flex-end;

        align-items: center;

        justify-content: space-around;

        width: 70%;

        cursor: pointer;

    }

    .nav\_sec\_2:before {

        content: ' ';

        display: block;

        background: $color\_4;

        width: 44%;

        height: 100px;

        position: absolute;

        top: 15;

        left: 216px;

        transform: skew(40deg);

        border-radius: 80px 15px 15px 0px  ;

        z-index: 0;

        transition: 0.3s;

    }

    .nav\_sec\_3{

        background-color: $color\_1;

        margin: 0;

        height: 100;

        list-style: none;

        display: flex;

        flex-direction: row;

        justify-self: flex-end;

        align-items: center;

        justify-content: space-around;

        width: 70%;

        cursor: pointer;

    }

    .nav\_sec\_3:before {

        content: ' ';

        display: block;

        background: $color\_4;

        width: 60%;

        height: 100px;

        position: absolute;

        top: 15;

        left: 216px;

        transform: skew(40deg);

        border-radius: 80px 15px 15px 0px  ;

        z-index: 0;

        transition: 0.3s;

    }

    .nav\_sec\_4{

        background-color: $color\_1;

        margin: 0;

        height: 100;

        list-style: none;

        display: flex;

        flex-direction: row;

        justify-self: flex-end;

        align-items: center;

        justify-content: space-around;

        width: 70%;

        cursor: pointer;

    }

    .nav\_sec\_4:before {

        content: ' ';

        display: block;

        background: $color\_4;

        width: 75%;

        height: 100px;

        position: absolute;

        top: 15;

        left: 216px;

        transform: skew(40deg);

        border-radius: 80px 15px 15px 0px  ;

        z-index: 0;

        transition: 0.3s;

    }

    .logo\_img{

        height: 100px;

        width: 240px;

        padding-bottom: 0;

        padding-right: 60px;

        transition: 0.3s;

        margin-left: 10px;

    }

    .type1{

        border-radius: 20px 20px 10px 0px;

        transition: 0.3s;

    }

    .type2{

        border-radius: 20px 20px 10px 20px;

        transition: 0.3s;

    }

}

\_components/SearchBar/SearchBar.js

"use client"

import { useEffect, useRef, useState } from "react";

import { useRouter } from 'next/navigation';

import styles from "./SearchBar.module.scss";

import Image from "next/image";

import img from "@/public/Group27319068.svg";

export default function SearchBar({}) {

    const [query, setQuery] = useState("");

    const router = useRouter();

    const inputEl = useRef(null);

    useEffect(function(){

        function callback(e){

            if(e.code.toLowerCase() === "Enter".toLowerCase()){

                if(document.activeElement === inputEl.current)

                router.push("/gamelist");

                inputEl.current.focus();

            }

        }

        document.addEventListener("keydown", callback);

        return function(){

            document.removeEventListener("keydown", callback);

        }

    }, [query]);

    function handleOpen(){

        router.push("/gamelist");

    }

    return(

        <div className={styles.bottom\_part}>

            <div className={styles.link\_section}>

                <p className={styles.fast\_link} onClick={handleOpen}>M</p>

                <p className={styles.fast\_link} onClick={handleOpen}>C</p>

            </div>

            <div className={styles.SearchBar}>

                <input

                    className={styles.search}

                    //autoFocus  //this can be used for input to be the main focus after page reload

                    ref={inputEl} // to connect the ref to the element

                    type="text"

                    placeholder="Search your game..."

                    value={query}

                    onChange={(e) => setQuery(e.target.value)}

                />

                <Image

                    //onClick={setGameQuery(query)}

                    quality={100}

                    className={styles.search\_img}

                    src={img}

                    alt="White Logo Img"

                /> {/\* we can set for the image fill parameter and it will cover all the parent component\*/}

                <p className={styles.search\_description}>Search for the description and other details here. FInd the board game you are currently interested in and start playing!</p>

            </div>

        </div>

    );

}

\_components/SearchBar/SearchBar.module.scss

@import "../../globals.scss";

.bottom\_part{

    display: flex;

    flex-direction: column;

    align-items: end;

    justify-content: end;

    width: 100%;

    .link\_section{

        display: flex;

        flex-direction: row;

        .fast\_link{

            width: 40px;

            height: 40px;

            display: flex;

            align-items: center;

            justify-content: center;

            margin: 10px;

            background-color: $color\_5;

            border-radius: 40px;

            @extend %font1;

            font-weight: 700;

            color: $color\_1;

            padding-left: 2px;

            transition: 0.3s;

        }

        .fast\_link:hover{

            background-color: $color\_4;

            box-shadow: 0.1rem 0.3rem 0.5rem rgba(0, 0, 0, 0.274);

            transition: 0.3s;

        }

    }

    .SearchBar{

        margin-top: 5px;

        z-index: 3;

        background-color: $color\_4;

        display: flex;

        flex-direction: column;

        align-items: end;

        width: 20%;

        padding: 10px;

        border-radius: 15px;

        margin-left: 30px;

        box-shadow: 1rem 1rem 1rem rgba(0, 0, 0, 0.274);

        .search {

            justify-self: center;

            border: none;

            padding: 1.1rem 1.6rem;

            font-size: 0.8rem;

            border-radius: 0.5rem;

            width: 94%;

            transition: all 0.3s;

            color: var(--color-text);

            padding: 5px;

            height: 30px;

            border: 1px solid #9f9f9f;

            /\* background-color: var(--color-background-900); \*/

            background-color: $color\_1;

        }

        .search::placeholder {

            color: #9f9f9f;

        }

        .search:focus {

            outline: none;

            box-shadow: 0 2.4rem 2.4rem rgba(0, 0, 0, 0.1);

            transform: translateY(-1px);

        }

        .search\_description{

            color: $color\_5;

            @extend %font1;

            padding-left: 6px;

            font-size: 10px;

            font-weight: 200;

        }

        .search\_img{

            position: absolute;

            right:80px;

            top: 605;

        }

    }

}

\_components/usernavigation/UserNavigation.js

'use client';

import Link from "next/link";

import styles from "./UserNavigation.module.scss";

import Image from "next/image";

import logo from "@/public/cover.png";

import { usePathname } from 'next/navigation';

import { useState, useEffect } from "react";

export default function Navigation() {

    const pathname = usePathname();

    const [currentPage, setCurrentPage] = useState(0);

    const [ulStyle, setUlStyle] = useState("nav\_sec\_0")

    useEffect(() => {

        if (pathname.includes("add\_booking")) {

            setCurrentPage(1);

            setUlStyle("nav\_sec\_1");

        } else if (pathname.includes("account")) {

            setCurrentPage(2);

            setUlStyle("nav\_sec\_2");

        } else if (pathname.includes("logout")) {

            setCurrentPage(3);

            setUlStyle("nav\_sec\_3");

        } else {

            setCurrentPage(0);

            setUlStyle("nav\_sec\_0");

        }

    }, [pathname]);

    return (

        <header className={styles.header}>

            <Link href="/" className={styles.logo\_sect\_1}>

                <Image

                    className={`${styles.logo\_img} ${styles.type2}`}

                    src={logo}

                    alt="Logo"

                />

            </Link>

            <ul className={styles[ulStyle]}>

                <li className={styles.nav\_li}>

                    <Link

                        className={`${styles.nav\_link} ${currentPage > 0 ? styles.active : ''}`}

                        href="/user/"

                    >

                        Main

                    </Link>

                </li>

                <li className={styles.nav\_li}>

                    <div

                        className={`${styles.nav\_link} ${currentPage > 1 ? styles.active : ''}`}

                    >

                        Add booking

                    </div>

                </li>

                <li className={styles.nav\_li}>

                    <Link

                        className={`${styles.nav\_link} ${currentPage > 2 ? styles.active : ''}`}

                        href="/user/account"

                    >

                        Account details

                    </Link>

                </li>

                <li className={styles.nav\_li}>

                    <Link

                        className={`${styles.nav\_link} ${currentPage > 3 ? styles.active : ''}`}

                        href="/user/logout"

                    >

                        Log out

                    </Link>

                </li>

            </ul>

        </header>

    )

}

\_components/usernavigation/UserNavigation.module.scss

@import "../../globals.scss";

.header{

    display: flex;

    flex-direction: row;

    justify-content: space-between;

    align-items: center;

    margin: 15px 0 0 0;

    a{

        margin-left: 10px;

        height: 100px;

    }

    .nav\_sec\_0{

        background-color: $color\_1;

        margin: 0;

        height: 100;

        list-style: none;

        display: flex;

        flex-direction: row;

        justify-self: flex-end;

        align-items: center;

        justify-content: space-around;

        width: 70%;

        cursor: pointer;

    }

    .nav\_li{

        z-index: 1;

        padding: 10px;

        .active{

            color:$color\_5;

            transition: 0.3s;

        }

    }

    .nav\_link{

        color: $color\_4;

        text-decoration: none;

        @extend %font1;

        transition: 0.3s;

    }

    .nav\_link:hover{

        color: $color\_5;

        transition: 0.3s;

    }

    .nav\_sec\_0:before {

        content: ' ';

        display: block;

        background: $color\_4;

        width: 15%;

        height: 100px;

        position: absolute;

        top: 15;

        left: 216px;

        transform: skew(40deg);

        border-radius: 80px 15px 20px 0px  ;

        z-index: 0;

        transition: 0.3s;

    }

    .nav\_sec\_1{

        background-color: $color\_1;

        margin: 0;

        height: 100;

        list-style: none;

        display: flex;

        flex-direction: row;

        justify-self: flex-end;

        align-items: center;

        justify-content: space-around;

        width: 70%;

        cursor: pointer;

    }

    .nav\_sec\_1:before {

        content: ' ';

        display: block;

        background: $color\_4;

        width: 33%;

        height: 100px;

        position: absolute;

        top: 15;

        left: 216px;

        transform: skew(40deg);

        border-radius: 80px 15px 15px 0px  ;

        z-index: 0;

        transition: 0.3s;

    }

    .nav\_sec\_2{

        background-color: $color\_1;

        margin: 0;

        height: 100;

        list-style: none;

        display: flex;

        flex-direction: row;

        justify-self: flex-end;

        align-items: center;

        justify-content: space-around;

        width: 70%;

        cursor: pointer;

    }

    .nav\_sec\_2:before {

        content: ' ';

        display: block;

        background: $color\_4;

        width: 51%;

        height: 100px;

        position: absolute;

        top: 15;

        left: 216px;

        transform: skew(40deg);

        border-radius: 80px 15px 15px 0px  ;

        z-index: 0;

        transition: 0.3s;

    }

    .nav\_sec\_3{

        background-color: $color\_1;

        margin: 0;

        height: 100;

        list-style: none;

        display: flex;

        flex-direction: row;

        justify-self: flex-end;

        align-items: center;

        justify-content: space-around;

        width: 70%;

        cursor: pointer;

    }

    .nav\_sec\_3:before {

        content: ' ';

        display: block;

        background: $color\_4;

        width: 72%;

        height: 100px;

        position: absolute;

        top: 15;

        left: 216px;

        transform: skew(40deg);

        border-radius: 80px 15px 15px 0px  ;

        z-index: 0;

        transition: 0.3s;

    }

    .logo\_img{

        height: 100px;

        width: 240px;

        padding-bottom: 0;

        padding-right: 60px;

        transition: 0.3s;

        margin-left: 10px;

    }

    .type1{

        border-radius: 20px 20px 10px 0px;

        transition: 0.3s;

    }

    .type2{

        border-radius: 20px 20px 10px 20px;

        transition: 0.3s;

    }

}

\_components/visitornavigation/Navigation.js

'use client';

import Link from "next/link";

import styles from "./Navigation.module.scss";

import Image from "next/image";

import logo from "@/public/cover.png";

import { usePathname } from 'next/navigation';

import { useState, useEffect } from "react";

export default function Navigation() {

    const pathname = usePathname();

    const [currentPage, setCurrentPage] = useState(0);

    const [ulStyle, setUlStyle] = useState("nav\_sec\_0")

    useEffect(() => {

        if (pathname.includes("prices")) {

            setCurrentPage(1);

            setUlStyle("nav\_sec\_1");

        } else if (pathname.includes("gamelist")) {

            setCurrentPage(2);

            setUlStyle("nav\_sec\_2");

        }else if (pathname.includes("championship")) {

            setCurrentPage(3);

            setUlStyle("nav\_sec\_3");

        } else if (pathname.includes("login")) {

            setCurrentPage(4);

            setUlStyle("nav\_sec\_4");

        }else {

            setCurrentPage(0);

            setUlStyle("nav\_sec\_0");

        }

    }, [pathname]);

    return (

        <header className={styles.header}>

            <Link href="/" className={styles.logo\_sect\_1}>

                <Image

                    className={`${styles.logo\_img} ${styles.type2}`}

                    src={logo}

                    alt="Logo"

                />

            </Link>

            <ul className={styles[ulStyle]}>

                <li className={styles.nav\_li}>

                    <Link

                        className={`${styles.nav\_link} ${currentPage > 0 ? styles.active : ''}`}

                        href="/"

                    >

                        Main

                    </Link>

                </li>

                <li className={styles.nav\_li}>

                    <Link

                        className={`${styles.nav\_link} ${currentPage > 1 ? styles.active : ''}`}

                        href="/prices"

                    >

                        Price

                    </Link>

                </li>

                <li className={styles.nav\_li}>

                    <Link

                        className={`${styles.nav\_link} ${currentPage > 2 ? styles.active : ''}`}

                        href="/gamelist"

                    >

                        Games List

                    </Link>

                </li>

                <li className={styles.nav\_li}>

                    <Link

                        className={`${styles.nav\_link} ${currentPage > 3 ? styles.active : ''}`}

                        href="/championship"

                    >

                        Championship

                    </Link>

                </li>

                <li className={styles.nav\_li}>

                    <Link

                        className={`${styles.nav\_link} ${currentPage > 4 ? styles.active : ''}`}

                        href="/login"

                    >

                        SignIn/SignUp

                    </Link>

                </li>

            </ul>

        </header>

    )

}

\_components/visitornavigation/Navigation.module.scss

@import "../../globals.scss";

.header{

    display: flex;

    flex-direction: row;

    justify-content: space-between;

    align-items: center;

    margin: 15px 0 0 0;

    a{

        margin-left: 10px;

        height: 100px;

    }

    .nav\_sec\_0{

        background-color: $color\_1;

        margin: 0;

        height: 100;

        list-style: none;

        display: flex;

        flex-direction: row;

        justify-self: flex-end;

        align-items: center;

        justify-content: space-around;

        width: 70%;

        cursor: pointer;

    }

    .nav\_li{

        z-index: 1;

        padding: 10px;

        .active{

            color:$color\_5;

            transition: 0.3s;

        }

    }

    .nav\_link{

        color: $color\_4;

        text-decoration: none;

        @extend %font1;

        transition: 0.3s;

    }

    .nav\_link:hover{

        color: $color\_5;

        transition: 0.3s;

    }

    .nav\_sec\_0:before {

        content: ' ';

        display: block;

        background: $color\_4;

        width: 15%;

        height: 100px;

        position: absolute;

        top: 15;

        left: 216px;

        transform: skew(40deg);

        border-radius: 80px 15px 20px 0px  ;

        z-index: 0;

        transition: 0.3s;

    }

    .nav\_sec\_1{

        background-color: $color\_1;

        margin: 0;

        height: 100;

        list-style: none;

        display: flex;

        flex-direction: row;

        justify-self: flex-end;

        align-items: center;

        justify-content: space-around;

        width: 70%;

        cursor: pointer;

    }

    .nav\_sec\_1:before {

        content: ' ';

        display: block;

        background: $color\_4;

        width: 29%;

        height: 100px;

        position: absolute;

        top: 15;

        left: 216px;

        transform: skew(40deg);

        border-radius: 80px 15px 15px 0px  ;

        z-index: 0;

        transition: 0.3s;

    }

    .nav\_sec\_2{

        background-color: $color\_1;

        margin: 0;

        height: 100;

        list-style: none;

        display: flex;

        flex-direction: row;

        justify-self: flex-end;

        align-items: center;

        justify-content: space-around;

        width: 70%;

        cursor: pointer;

    }

    .nav\_sec\_2:before {

        content: ' ';

        display: block;

        background: $color\_4;

        width: 41%;

        height: 100px;

        position: absolute;

        top: 15;

        left: 216px;

        transform: skew(40deg);

        border-radius: 80px 15px 15px 0px  ;

        z-index: 0;

        transition: 0.3s;

    }

    .nav\_sec\_3{

        background-color: $color\_1;

        margin: 0;

        height: 100;

        list-style: none;

        display: flex;

        flex-direction: row;

        justify-self: flex-end;

        align-items: center;

        justify-content: space-around;

        width: 70%;

        cursor: pointer;

    }

    .nav\_sec\_3:before {

        content: ' ';

        display: block;

        background: $color\_4;

        width: 55.5%;

        height: 100px;

        position: absolute;

        top: 15;

        left: 216px;

        transform: skew(40deg);

        border-radius: 80px 15px 15px 0px  ;

        z-index: 0;

        transition: 0.3s;

    }

    .nav\_sec\_4{

        background-color: $color\_1;

        margin: 0;

        height: 100;

        list-style: none;

        display: flex;

        flex-direction: row;

        justify-self: flex-end;

        align-items: center;

        justify-content: space-around;

        width: 70%;

        cursor: pointer;

    }

    .nav\_sec\_4:before {

        content: ' ';

        display: block;

        background: $color\_4;

        width: 71%;

        height: 100px;

        position: absolute;

        top: 15;

        left: 216px;

        transform: skew(40deg);

        border-radius: 80px 15px 15px 0px  ;

        z-index: 0;

        transition: 0.3s;

    }

    .logo\_img{

        height: 100px;

        width: 240px;

        padding-bottom: 0;

        padding-right: 60px;

        transition: 0.3s;

        margin-left: 10px;

    }

    .type1{

        border-radius: 20px 20px 10px 0px;

        transition: 0.3s;

    }

    .type2{

        border-radius: 20px 20px 10px 20px;

        transition: 0.3s;

    }

}

\_components/Button.js

"use client"

import styles from "../globals.scss";

export default function Button({ children, name, action = ()=> console.log("no action"), width = 165, height=50 }) {

    return (

    <button

        className={name}

        onClick={(e) => {

            e.preventDefault();

            action();

        }}

        style={{ width: width ?? 'auto', height: height ?? 'auto' }}

    >

        {children}

    </button>

    );

}

user/page.js

"use client"

import styles from "./User.module.scss";

import Button from "../\_components/Button";

import EventDetails from "./\_event\_details/eventDetails.js";

import FullCalendar from '@fullcalendar/react'

import dayGridPlugin from '@fullcalendar/daygrid'

import interactionPlugin from "@fullcalendar/interaction"

import Loading from "@/app/loading.js";

import { useDispatch, useSelector } from "react-redux";

import { useRouter } from 'next/navigation';

import { useEffect, useState } from "react";

import { checkBookings } from "../\_lib/bookingsSlice.js";

export default function Home() {

  const [date, setDate] = useState("");

  const [active, setActive] = useState(false); // for the add\_booking

  const [showbooking, setShowbooking] = useState(false); // for the booking details

  const [selectedBooking, setSelectedBooking] = useState({});

  const { user } = useSelector((state) => state.account);

  const { bookings, bookingIsLoading } = useSelector((state) => state.bookings);

  const dispatch = useDispatch();

  const router = useRouter();

  useEffect(function(){

    if(user.email === ""){

      router.push("/");

    }

    dispatch(checkBookings(user));

  }, [user, router, dispatch]);

  const currentDate = new Date().toISOString().split('T')[0];

  //pop up windows

  const handleDateClick = (arg) => {

    if(!showbooking){

      if(arg.dateStr<currentDate){

        alert("Please choose the today or later date.");

      }else{

        setDate(arg.dateStr);

        setActive(true);

      }

    }

  }

  function bookingClick(info) {

    if(!active){

      const clickedDate = info.event.start.toISOString().split("T")[0];

      const foundBooking = bookings.find((booking) => {

        const bookingDate = new Date(booking.date).toISOString().split("T")[0];

        return clickedDate === bookingDate;

      });

      if (foundBooking) {

        setSelectedBooking(foundBooking);

        setShowbooking(true);

      }

    }

  }

  if (bookingIsLoading) return <Loading />;

  return (

    <div className={styles.container}>

      {user.email &&

        <>

          <div className={styles.container\_title}>User: <h2 className={styles.container\_title\_name}>{user.name}</h2></div>

          <Alert active={active} date={date} handleActiveClick={setActive}/>

          <EventDetails active={showbooking} booking={selectedBooking} handleActiveClick={setShowbooking} />

          <FullCalendar

            plugins={[ dayGridPlugin, interactionPlugin ]}

            headerToolbar={{

              left: 'prev,next today',

              center: 'title',

              right: '',

            }}

            events={bookings.map((booking) => ({

              title: `Registration ${booking.time}`,

              date: booking.date,

            }))}

            eventClick={(info) => bookingClick(info)}

            eventContent={renderEventContent}

            eventClassNames={() => styles.custom\_style}

            dateClick={handleDateClick}

            initialView="dayGridMonth"

            weekends={false}

          />

        </>

      }

    </div>

  );

}

function Alert({active, date, handleActiveClick}){

  const router = useRouter();

  function handleClick() {

    router.replace(`/user/add\_booking?date=${encodeURIComponent(date)}`);

    handleActiveClick(false);

  }

  return(

    <form className={`${styles.section7\_login} ${active ? styles.active : null}`}>

        <h2 className={styles.main\_title\_caption}>{date} <p className={styles.cross} onClick={() => handleActiveClick(false)}>&times;</p></h2>

        <p className={styles.section7\_consultation\_description}>If you want to add a booking for this day, please click the Continue button to proceed with the booking registration.</p>

        <div className={styles.check\_box\_sec}>

            <Button

                name={"button\_template2"}

                width={490}

                height={60}

                action={handleClick}

            >

              Continue

            </Button>

        </div>

    </form>

  );

}

function renderEventContent(eventInfo) {

  return(

    <>

      <h3>{eventInfo.event.title}</h3>

    </>

  )

}

user/User.module.scss

@import "../globals.scss";

.container{

    margin: 30px 20px;

    @extend %font1;

    display: flex;

    flex-direction: column;

    .container\_title{

        display: flex;

        align-self: flex-end;

        align-items: center;

        justify-content: space-around;

        font-size: 24px;

        padding: 10px 40px;

        box-shadow: 0.4rem 0.4rem 1rem rgba(0, 0, 0, 0.274);

        border-radius: 10px;

        margin-bottom: 30px;

        .container\_title\_name{

            padding: 15px 20px;

            border-radius: 10px;

            background-color: $color\_4;

            color: $color\_1;

            margin-left: 30px;

            font-size: 26px;

        }

    }

    .custom\_style {

        background-color: $color\_4 !important;

        height: 60px !important;

        display: flex;

        align-items: center;

        justify-content: center;

        font-weight: 500;

        border-radius: 4px;

    }

}

//for the calendar

:global(.fc-daygrid-day) {

    height: 100px;

}

:global(.fc-button ){

    background-color: $color\_4 !important;

    border: none;

    color: white;

    font-weight: bold;

    transition: 0.2s;

}

:global(.fc-button:hover) {

        background-color: $color\_4 !important;

        border: 1px solid black;

        transition: 0.2s;

}

.main\_title\_caption{

    @extend %font1;

    font-size: 53px;

    letter-spacing: 5px;

    font-weight: 900;

    color: $color\_5;

    margin: 30px 0 20px 0 ;

    font-weight: bolder;

    text-align: center;

    color: $color\_5;

    display: flex;

    flex-direction: row;

    justify-content: space-between;

    align-items: start;

    width: 90%;

    .cross{

        margin: 0;

        background-color: $color\_1;

        padding: 0px 8px 0px 15px;

        border-radius: 7px;

        cursor: pointer;

    }

}

.section7\_login{

    position: fixed;

    z-index: 3;

    width: 550px;

    flex-direction: column;

    margin-left: 0vw;

    padding-left: 60px;

    background-color: $color\_4;

    padding: 8px;

    border-radius: 15px;

    display: none;

    align-items: center;

    justify-self: center;

    align-self: center;

    box-shadow: 1rem 1.4rem 25rem rgb(0, 0, 0);

}

.section7\_login::after{

    content: "";

    top: 0;

    left: 0;

    position: absolute;

    width: 100%;

    height: 100%;

    opacity: 0;

    visibility: hidden;

    transition: 0.5s;

}

.section7\_consultation\_description{

    width: 490px;

    font-size: 19px;

    line-height: 22px;

    letter-spacing: 1.22px;

    color: $color\_5;

    margin: 0 0 35px 0;

    text-align: justify;

    @extend %font2;

    font-weight: 200;

}

.check\_box\_sec{

    margin-bottom: 15px;

}

.active{

    display: flex;

}

user/logout/Logout.js

"use client"

import Button from "@/app/\_components/Button";

import styles from "./Logout.scss";

import { useEffect } from "react";

import { useRouter } from 'next/navigation';

import { useDispatch, useSelector } from "react-redux";

import { userLogout } from "../../\_lib/accountSlice.js";

export default function LogOut() {

  const router = useRouter();

  const dispatch = useDispatch();

  const { user } = useSelector((state) => state.account);

  useEffect(function(){

    if(user.email === ""){

      router.push("/");

    }

  }, [user.email, router]);

  function handleClick() {

    dispatch(userLogout(user));

    router.replace('/');

  }

  return (

    <>

      {user.email &&

        <form id="sect7-log" className="section7-login">

          <h2 className="section7-consultation main-title-caption">Log out </h2>

          <p className="section7-consultation-description">Please confirm that you really want to log out of your account, as in this case, you will need to sign in one more time to get to this interface.</p>

          <div className="check\_box\_sec">

              <Button

                  name={"button\_template2"}

                  width={490}

                  action={handleClick}

              >

                Log out

              </Button>

          </div>

      </form>

      }

    </>

  );

}

user/logout/Logout.module.scss

@import "../../globals.scss";

.authenticate{

    select{

        margin: 20px 0 0 10px;

        height: 40px;

        color: $color\_1;

        background-color: $color\_3;

        border-radius: 10px;

        border: none;

    }

}

.main-title-caption{

    @extend %font1;

    font-size: 53px;

    letter-spacing: 5px;

    font-weight: 900;

    color: $color\_5;

    margin: 20px 0 30px 0 ;

    font-weight: bolder;

    text-align: center;

    color: $color\_5;

}

.section7-login{

    position: relative;

    width: 550px;

    flex-direction: column;

    padding-left: 60px;

    margin: 80px 0 80px 0;

    background-color: $color\_4;

    padding: 8px;

    border-radius: 15px;

    display: flex;

    align-items: center;

    justify-self: center;

    box-shadow: 1rem 1.4rem 3rem rgba(0, 0, 0, 0.356);

}

.section7-login::after{

    content: "";

    top: 0;

    left: 0;

    position: absolute;

    width: 100%;

    height: 100%;

    opacity: 0;

    visibility: hidden;

    transition: 0.5s;

}

.section7-consultation{

    text-align: start;

    font-size: 43px;

    width: 400px;

    text-align: center;

}

.section7-consultation-description{

    width: 490px;

    font-size: 19px;

    line-height: 22px;

    letter-spacing: 1.22px;

    color: $color\_5;

    margin: 0 0 35px 0;

    text-align: justify;

    @extend %font2;

    font-weight: 200;

}

.name.\_error{

    box-shadow: 0 0 15px red;

}

.check\_box\_sec{

    margin-bottom: 15px;

}

user/add\_booking/AddBooking.js

"use client";

import Button from "@/app/\_components/Button";

import styles from "./AddBooking.module.scss";

import { useRouter } from 'next/navigation';

import { useSelector, useDispatch } from "react-redux";

import { useSearchParams } from 'next/navigation';

import { getGames } from "@/app/\_lib/gameSlice.js";

import { createBooking } from "../../\_lib/bookingsSlice.js";

import { useEffect, useState } from "react";

const availableTimes = ["9:00", "10:00", "11:00", "12:00", "13:00", "14:00", "15:00", "16:00", "17:00", "18:00", "19:00", "20:00",];

const bookedGames = ["VR Arena"];

const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;

const phoneRegex = /^\d{9}$/;

export default function Home() {

    const searchParams = useSearchParams();

    const date = searchParams.get('date');

    const [playerCount, setPlayerCount] = useState(2);

    const [inputError, setInputError] = useState("");

    const [players, setPlayers] = useState([{ name: "", email: "", phone: "" }, { name: "", email: "", phone: "" }]);

    const [selectedTime, setSelectedTime] = useState("");

    const [selectedPeriod, setSelectedPeriod] = useState("");

    const [selectedGame, setSelectedGame] = useState("");

    const [availableGames, setAvailableGames] = useState([]);

    const { games, isLoading, error } = useSelector((store) => store.game);

    const { bookingError, bookingIsLoading } = useSelector((state) => state.bookings);

    const { user } = useSelector((state) => state.account);

    const dispatch = useDispatch();

    useEffect(() => {

        setInputError("");

        dispatch(getGames());

    }, [dispatch]);

    useEffect(() => {

        if (games && games.length > 0) {

            const gameNames = games.map((game) => game.game\_name);

            setAvailableGames(gameNames);

        }

    }, [games]);

    const router = useRouter();

    useEffect(function(){

        if(user.email === ""){

            router.push("/");

        }

    }, [user.email, router]);

    const handlePlayerChange = (index, field, value) => {

        const updated = [...players];

        updated[index][field] = value;

        setPlayers(updated);

    };

    const handlePlayerCountChange = (e) => {

        const count = parseInt(e.target.value);

        if(count > 1 && count < 13){

            setPlayerCount(count);

            const updatedPlayers = Array.from({ length: count }, (\_, i) => players[i] || { name: "", email: "", phone: "" });

            setPlayers(updatedPlayers);

        }

    };

    const handlePeriod = (value) => {

        const count = parseInt(value);

        if(count >= 1 && count < 5){

            setSelectedPeriod(count);

        }

    }

    const availableGamesFiltered = availableGames.filter(game => !bookedGames.includes(game));

    // to create new booking

    function handleCreateBooking(){

        setInputError("");

        const hasInvalidPlayer = players.some((player, index) => {

            if (index !== 0) {

                const hasMissingFields = !player.email || !player.name || !player.phone;

                const isInvalidEmail = !emailRegex.test(player.email || "");

                const isNotUserEmail = player.email === user.email;

                const isInvalidPhone = !phoneRegex.test(player.phone || "" );

            return hasMissingFields || isInvalidEmail || isInvalidPhone || isNotUserEmail;

            }

        });

        if (hasInvalidPlayer) {

            setInputError("Please enter valid name, email, and phone for some players");

            return;

        }

        if(!selectedTime){

            setInputError("Please choose the time");

            return;

        }

        if(!selectedPeriod){

            setInputError("Please choose the amount of hours");

            return;

        }

        if(!selectedGame){

            setInputError("Please choose the game you want to play");

            return;

        }

        dispatch(createBooking(user.email, user.userId, date, selectedTime, selectedPeriod, selectedGame, players));

        router.push("/user");

    }

    return (

        <div className={styles.addB\_container}>

            {user.email && <>

                <div className={styles.caption\_part}>

                    <h1 className={styles.title}>Add Booking</h1>

                    <p className={styles.date}>Selected date: {date}</p>

                </div>

                <div className={styles.formGroup1}>

                    <label>Number of players:</label>

                    <input

                        type="number"

                        min="2"

                        max="12"

                        value={playerCount}

                        onChange={handlePlayerCountChange}

                    />

                </div>

                <div className={styles.button\_part}>

                    {(bookingError || inputError) && <h2 className={styles.error}>{bookingError? bookingError: inputError}</h2>}

                    {players.map((player, index) => (

                        <div key={index} className={styles.playerForm}>

                            <h3>Player {index + 1}</h3>

                            {index === 0 ? (

                            <p className={styles.youLabel}>You</p>

                            ) : (

                            <>

                                <input

                                required

                                type="text"

                                placeholder="Name"

                                value={player.name}

                                onChange={(e) => handlePlayerChange(index, "name", e.target.value)}

                                />

                                <input

                                required

                                type="email"

                                placeholder="Email"

                                aria-describedby="emailHelp"

                                value={player.email}

                                onChange={(e) => handlePlayerChange(index, "email", e.target.value)}

                                />

                                <input

                                required

                                type="tel"

                                placeholder="Phone Number"

                                aria-describedby="Phone"

                                value={player.phone}

                                onChange={(e) => handlePlayerChange(index, "phone", e.target.value)}

                                />

                            </>

                            )}

                        </div>

                    ))}

                    <div className={styles.formGroup}>

                        <label>Select Time:</label>

                        <select value={selectedTime} onChange={(e) => setSelectedTime(e.target.value)}>

                            <option value="">-- Select Time --</option>

                            {availableTimes.map((time) => (

                                <option key={time} value={time}>{time}</option>

                            ))}

                        </select>

                    </div>

                    <div className={styles.formGroup}>

                        <label>Amount of hours:</label>

                        <input

                            type="number"

                            min="1"

                            max="4"

                            value={selectedPeriod}

                            onChange={(e) => handlePeriod(e.target.value)}

                        />

                    </div>

                    <div className={styles.formGroup}>

                        <label>Select Game:</label>

                        <select value={selectedGame} onChange={(e) => setSelectedGame(e.target.value)}>

                            <option value="">-- Select Game --</option>

                            {availableGamesFiltered.map((game) => (

                                <option key={game} value={game}>{game}</option>

                            ))}

                        </select>

                    </div>

                    <div className={styles.button\_sec}>

                        <p>Please keep in mind that only member of the desired playing group can create the booking. So if you are not the member of the group you want to register, please ask any member to register it, as in case you just add yourself to the group it will increase the cost of booking</p>

                        <Button

                            name={"button\_template1"}

                            width={490}

                            action={handleCreateBooking}

                        >

                            {bookingIsLoading ? "Loading..." : "Register Booking"}

                        </Button>

                    </div>

                </div>

            </>}

        </div>

    );

}

user/add\_booking/AddBooking.module.scss

@import "../../globals.scss";

.addB\_container{

    flex-wrap: wrap;

    @extend %font1;

    background-color: $color\_1;

    box-shadow: 1rem 2rem 5rem rgba(0, 0, 0, 0.301);

    border-radius: 15px;

    margin: 60px 20px 80px 20px;

    padding: 20px 40px;

    z-index: 4;

    display: flex;

    flex-direction: row;

    .caption\_part{

        width: 50%;

    }

    .formGroup {

        margin: 20px 0;

        display: flex;

        flex-direction: column;

        label {

            font-weight: bold;

            margin-bottom: 8px;

            color: $color\_4;

        }

        input,

        select {

            padding: 10px;

            border: 1px solid #ccc;

            border-radius: 6px;

            font-size: 16px;

            background-color: #fff;

            transition: border-color 0.2s ease;

            margin-left: 0px;

            color: #000;

            &:focus {

            outline: none;

            border-color: $color\_4;

            }

        }

    }

    .formGroup1 {

        width: 50%;

        margin: 20px 0;

        display: flex;

        flex-direction: column;

        align-items: flex-end;

        label {

            font-weight: bold;

            margin-bottom: 8px;

            color: $color\_4;

            margin-right: 60px;

        }

        input,

        select {

            padding: 10px;

            border: 1px solid #ccc;

            border-radius: 6px;

            font-size: 16px;

            background-color: #fff;

            transition: border-color 0.2s ease;

            width: 200px;

            &:focus {

            outline: none;

            border-color: $color\_4;

            }

        }

    }

    .button\_part{

        width: 100%;

        .error{

            background-color: red;

            color: $color\_1;

            @extend %font1;

            padding: 10px;

            width: 98.5%;

            border-radius: 7px;

            text-align: center;

        }

    }

    .playerForm {

        background-color: $color\_4;

        padding: 15px;

        border-radius: 10px;

        margin-bottom: 25px;

        display: flex;

        flex-direction: row;

        flex-wrap: wrap;

        justify-content: space-between;

        h3 {

            width: 10%;

            margin: 5px;

            color: $color\_7;

            font-size: 24px;

            display: flex;

            align-self: center;

        }

        .youLabel{

            width: 20%;

            margin: 0 40% 0 0;

            display: flex;

            justify-content: center;

            align-items: center;

            background-color: $color\_1;

            padding: 8px;

            border-radius: 10px;

        }

        input{

            padding-left: 16px;

            width: 27%;

            height: 40px;

            font-family: Poppins;

            font-weight: 600;

            font-size: 16px;

            line-height: 20px;

            border: 1px solid #bfbfbf;

            box-shadow: inset 0 0 0 55px #fff!important;

            border-radius: 7px;

            margin: 0;

            outline: none;

        }

        input:-webkit-autofill {

            -webkit-box-shadow: inset 0 0 0 55px #fff!important;

            box-shadow: inset 0 0 0 55px #fff!important;

        }

        input :placeholder-shown{

            color: $color\_5;

            letter-spacing: 1px;

        }

    }

    .submit {

        padding: 12px 24px;

        background-color: $color\_4;

        color: #fff;

        font-weight: bold;

        font-size: 16px;

        border: none;

        border-radius: 8px;

        cursor: pointer;

        transition: background-color 0.2s ease;

        &:hover {

            background-color: darken($color\_4, 10%);

        }

    }

    .title{

        color: $color\_4;

        margin-bottom: 10px;

    }

    .date{

        font-size: 16px;

        margin-top: 0px;

    }

    .button\_sec{

        display: flex;

        flex-direction: row;

        align-items: center;

        justify-content: space-between;

        p{

            font-size: 14px;

            font-weight: 200;

            width: 50%;

            text-align: justify;

        }

    }

}

user/account/page.js

"use client"

import Button from "@/app/\_components/Button";

import styles from "./Account.scss";

import { useEffect, useState } from "react";

import { useRouter } from 'next/navigation';

import { useDispatch, useSelector } from "react-redux";

import { setError, updateUser } from "@/app/\_lib/accountSlice";

export default function Home() {

  const router = useRouter();

  const dispatch = useDispatch();

  const [name, setName] = useState("");

  const [phone, setPhone] = useState(0);

  const [password, setPassword] = useState("");

  const [checked, setChecked] = useState(false);

  const { user, isLoading, error } = useSelector((state) => state.account);

  useEffect(function(){

    if(user.email === ""){

      router.push("/");

    }

  }, [user.email, router]);

  useEffect(() => {

    if (user) {

      dispatch(setError(""));

      setName(user.name || "");

      setPhone(user.phone || "");

      setPassword(user.password || "");

    }

  }, [user]);

  function handleClick() {

    dispatch(setError(""));

    if (!checked) {

      return dispatch(setError("You have not confirmed the agreement"));

    }

    if (!name.trim()) {

      return dispatch(setError("Please enter your new name"));

    }

    const phoneRegex = /^[0-9]{9}$/;

    if (!phoneRegex.test(phone)) {

      return dispatch(setError("Please enter a valid phone number(9 digits)"));

    }

    if (!password.trim()) {

      return dispatch(setError("Please type the password"));

    }

    dispatch(updateUser(name, phone, password));

  }

  return (

    <>

      {user.email &&

        <form id="sect7-log" className="section7-login">

          <h2 className="section7-consultation main-title-caption">Your account details </h2>

          <p className="section7-consultation-description">Please take into account that user email cannot be changed. You can only change your full name, your phone number or your account password.</p>

          <p className="email">{user.email}</p>

          <input type="text" className="name \_req" required name="login" placeholder="Name" value={name} onChange={(e) => setName(e.target.value)} /> <br />

          <input type="phone" className="name \_req" placeholder="Phone" value={phone} onChange={(e) => setPhone(e.target.value)}/> <br />

          <input type="text" className="name \_req" required aria-describedby="Password"placeholder="Password" value={password} onChange={(e) => setPassword(e.target.value)}/> <br />

          {error && <h2 className="error">{error}</h2>}

          <div className="check\_box\_sec">

              <input id="cons\_check" type="checkbox" required value={checked} onClick={() => setChecked((checked) => !checked)} className="section7-checkbox"/>

              <p className="section7-checkbox-label">I accept the terms of the user agreement for the processing of personal data</p>

              <Button action={handleClick} name={"button\_template2"}>

              <p>{isLoading ? "Loading..." : "Submit"}</p>

              </Button>

          </div>

      </form>

      }

    </>

  );

}

user/account/Account.scss

@import "../../globals.scss";

.authenticate{

    select{

        margin: 20px 0 0 10px;

        height: 40px;

        color: $color\_1;

        background-color: $color\_3;

        border-radius: 10px;

        border: none;

    }

}

.main-title-caption{

    @extend %font1;

    font-size: 53px;

    letter-spacing: 5px;

    font-weight: 900;

    color: $color\_5;

    margin: 20px 0 30px 0 ;

    font-weight: bolder;

    text-align: center;

    color: $color\_5;

}

.section7-login{

    position: relative;

    width: 550px;

    flex-direction: column;

    padding-left: 60px;

    margin: 80px 0 80px 0;

    background-color: $color\_4;

    padding: 8px;

    border-radius: 15px;

    display: flex;

    align-items: center;

    justify-self: center;

    box-shadow: 1rem 1.4rem 3rem rgba(0, 0, 0, 0.356);

}

.section7-login::after{

    content: "";

    top: 0;

    left: 0;

    position: absolute;

    width: 100%;

    height: 100%;

    opacity: 0;

    visibility: hidden;

    transition: 0.5s;

}

.section7-consultation{

    text-align: start;

    font-size: 43px;

    width: 400px;

    text-align: center;

}

.section7-consultation-description{

    width: 490px;

    font-size: 19px;

    line-height: 22px;

    letter-spacing: 1.22px;

    color: $color\_5;

    margin: 0 0 35px 0;

    text-align: justify;

    @extend %font2;

    font-weight: 200;

}

.name.\_error{

    box-shadow: 0 0 15px red;

}

.check\_box\_sec{

    margin-bottom: 15px;

}

.email{

    height: 54px;

    background-color: $color\_1;

    width: 490px;

    border: 1px solid rgba(0, 0, 0, 0.235);

    border-radius: 15px;

    margin: 10px 0 10px 0;

    padding: 0;

    display: flex;

    align-items: center;

    justify-content: center;

    color: $color\_4;

    @extend %font1;

}

user/\_event\_details/eventDetails.js

import Button from "@/app/\_components/Button";

import styles from "./EventDetails.module.scss";

export default function EventDetails({ active, booking, handleActiveClick}){

    return(

        <form className={`${styles.section7\_login} ${active ? styles.active : null}`}>

            { booking &&

                <>

                    <h2 className={styles.main\_title\_caption}>

                        Date: {booking.date ? new Intl.DateTimeFormat('en-CA').format(new Date(booking.date)) : "Invalid Date"}

                        <p className={styles.cross} onClick={() => handleActiveClick(false)}>&times;</p>

                    </h2>

                    <h4 className={`${styles.main\_title\_caption} ${styles.secondary}`}>Time: <p>{booking.time}</p></h4>

                    <h4 className={`${styles.main\_title\_caption} ${styles.secondary}`}>Period: <p>{booking.period}</p></h4>

                    <h4 className={`${styles.main\_title\_caption} ${styles.third}`}>Game name: <p>{booking.game\_name}</p></h4>

                    <div className={styles.player\_list}>

                    {booking?.member\_emails?.length > 0 && booking.member\_emails.map(player => (

                        <p key={player} className={styles.member}>{player}</p>

                    ))}

                    </div>

                    <p className={styles.section7\_consultation\_description}>If you want to cancel the registration, please contact us to support@domain.com or call +380685647839 from 9am to 8pm. Sorry for the inconveniences!</p>

                </>

            }

        </form>

    );

}

user/\_event\_details/eventDetails.js

@import "../../globals.scss";

.main\_title\_caption{

    @extend %font1;

    font-size: 45px;

    letter-spacing: 5px;

    font-weight: 900;

    color: $color\_5;

    margin: 30px 0 0 0 ;

    font-weight: bolder;

    text-align: center;

    color: $color\_5;

    display: flex;

    flex-direction: row;

    justify-content: space-between;

    align-items: start;

    width: 90%;

    .cross{

        margin: 0;

        background-color: $color\_1;

        padding: 0px 8px 0px 15px;

        border-radius: 7px;

        cursor: pointer;

    }

}

.section7\_login{

    position: fixed;

    z-index: 10;

    top: 40px;

    display: flex;

    justify-self: center;

    align-self: center;

    width: 750px;

    flex-direction: column;

    margin-left: 0vw;

    padding-left: 60px;

    background-color: $color\_4;

    padding: 8px;

    border-radius: 15px;

    display: none;

    align-items: center;

    justify-self: center;

    align-self: center;

    box-shadow: 1rem 1.4rem 25rem rgb(0, 0, 0);

}

.section7\_login::after{

    content: "";

    top: 0;

    left: 0;

    position: absolute;

    width: 100%;

    height: 100%;

    opacity: 0;

    visibility: hidden;

    transition: 0.5s;

}

.player\_list{

    margin: 20px 0;

    background-color: $color\_1;

    width: 90%;

    display: flex;

    flex-direction: column;

    align-items: start;

    justify-content: center;

    border-radius: 10px;

    .member{

        margin-left: 25%;

        color: $color\_3;

    }

}

.section7\_consultation\_description{

    width: 90%;

    font-size: 16px;

    line-height: 22px;

    letter-spacing: 1.22px;

    color: $color\_5;

    margin: 0 0 35px 0;

    text-align: justify;

    @extend %font2;

    font-weight: 200;

}

.active{

    display: flex;

}

.secondary{

    display: flex;

    flex-direction: row;

    align-items: center;

    justify-content: start;

    height: 50px;

    font-size: 26px;

    p{

        margin-left: 20px;

    }

}

.third{

    display: flex;

    flex-direction: row;

    align-items: center;

    justify-content: start;

    font-size: 25px;

    height: 50px;

    p{

        margin-left: 30px;

    }

}

admin/page.js

"use client"

import { useEffect, useState } from "react";

import { useRouter } from 'next/navigation';

import { useSelector, useDispatch } from "react-redux";

import { getGames } from "@/app/\_lib/gameSlice.js";

import { getTables } from "@/app/\_lib/tableSlice.js";

import { checkAdminBookings, resetBookings } from "../\_lib/bookingsSlice.js";

import styles from "./Admin.module.scss";

import EventDetails from "../user/\_event\_details/eventDetails.js";

import Bookings from "./\_bookings\_list/BookingsList.js";

import Loading from "@/app/loading.js";

export default function Home() {

  const router = useRouter();

  const [selectedDate, setSelectedDate] = useState(() => {

    const today = new Date().toISOString().split("T")[0];

    return today;

  });

  const [showAllBookings, setShowAllBookings] = useState(false);

  const [shownearestbooking, setShowNearestbooking] = useState(false); // for the booking details

  const [selectedBooking, setSelectedBooking] = useState({});

  const [selectedTable, setSelectedTable] = useState(0);

  const { user } = useSelector((state) => state.account);

  const { tables } = useSelector((state) => state.tables);

  const { bookings, bookingIsLoading} = useSelector((state) => state.bookings);

  const dispatch = useDispatch();

  useEffect(() => {

    if (!user.email) {

      router.push("/");

      return;

    }

    dispatch(getGames());

    dispatch(getTables(user.email, user.userId));

    if (user.access) {

      dispatch(checkAdminBookings(user.email, user.userId, new Date(selectedDate).getFullYear()));

    }

  }, [user, router, dispatch, selectedDate]);

  const getNearestBooking = (tableId) => {

    if (!selectedDate) return null;

    const filteredBookings = bookings

      .filter(b => {

        // Ensure b.date is parsed correctly as Date object

        const bookingDate = new Date(b.date);

        bookingDate.setDate(bookingDate.getDate() + 1);

        const bookingDateOnly = bookingDate.toISOString().split("T")[0];

        return b.table\_id === tableId && bookingDateOnly === selectedDate && b.time;

      })

      .sort((a, b) => {

        // Convert time to seconds for sorting

        const [ah, am, as = 0] = a.time.split(":").map(Number);

        const [bh, bm, bs = 0] = b.time.split(":").map(Number);

        const aTimeInSeconds = ah \* 3600 + am \* 60 + as;

        const bTimeInSeconds = bh \* 3600 + bm \* 60 + bs;

        return aTimeInSeconds - bTimeInSeconds;

      });

    return filteredBookings[0] || null;

  };

  const handleListClick = (booking, e) => {

    setSelectedBooking(booking);

    if(e.target.value !== ""){

      setShowNearestbooking(true);

    }

  };

  const handleClick = (booking) => {

    const clickedDate = booking.date;

    setSelectedBooking(booking);

    setShowNearestbooking(true);

  };

  const handleAllClick = (tableId) => {

    setSelectedTable(tableId);

    setShowAllBookings(true);

  };

  const handleDateClick = (value) =>{

    setSelectedDate(value);

  }

  if (bookingIsLoading) return <Loading />;

  return (

    <div className={styles.container}>

      {user.access &&

      <>

        <h2 className={styles.main\_title}>Tables List</h2>

        {user.access === "manager" &&

          <div className={styles.datePickerContainer}>

            <label htmlFor="datePicker">Date: </label>

            <input

              id="datePicker"

              type="date"

              value={selectedDate}

              onChange={(e) => handleDateClick(e.target.value)}

              className={styles.datePicker}

            />

        </div>

        }

        <div className={styles.tableGrid}>

          {tables.map(table => {

            const booking = getNearestBooking(table.table\_id);

            if(table.status === true){

              return (

                <div key={table.table\_id} className={styles.tableItem}>

                  <div className={`${styles.tableOval} ${booking ? styles.active : null}`} onClick={() => handleAllClick(table.table\_id)}>Table {table.table\_id}</div>

                  <div className={styles.tableDetails}>

                    <p className={styles.tableState}>{table.state}</p>

                    {booking ? (

                      <div className={styles.bookingBox} onClick={() => handleClick(booking)}>

                        {booking.time}

                      </div>

                    ) : (

                      <div className={styles.bookingBoxEmpty}>No upcoming</div>

                    )}

                  </div>

                </div>

              );

            }

          })}

        </div>

        <EventDetails

          active={shownearestbooking}

          booking={selectedBooking}

          handleActiveClick={setShowNearestbooking}

        />

        <Bookings

          user={user}

          active={showAllBookings}

          tableId={selectedTable}

          date={selectedDate}

          handleListClick={handleListClick}

          handleActiveClick={setShowAllBookings}

        />

        </>

      }

    </div>

  );

}

admin/Admin.module.scss

@import "../globals.scss";

.container{

    height: 900px;

    display: flex;

    flex-direction: column;

}

.main\_title{

    @extend %font1;

    font-size: 53px;

    letter-spacing: 5px;

    font-weight: 900;

    color: $color\_5;

    margin: 30px 0 0 0 ;

    font-weight: bolder;

    text-align: center;

    color: $color\_5;

}

.datePickerContainer{

    @extend %font1;

    display: flex;

    align-self: flex-end;

    align-items: center;

    justify-content: space-around;

    font-size: 26px;

    padding: 10px 30px;

    box-shadow: 0.4rem 0.4rem 1rem rgba(0, 0, 0, 0.274);

    border-radius: 10px;

    margin: 0 20px 30px 0;

    .datePicker{

        padding: 5px;

        border-radius: 10px;

        background-color: $color\_4;

        color: $color\_1;

        margin-left: 30px;

        font-size: 20px;

    }

}

.tableGrid {

    @extend %font1;

    display: grid;

    grid-template-columns: repeat(auto-fill, minmax(180px, 1fr));

    gap: 10%;

    padding: 22px;

    .tableItem {

        display: flex;

        flex-direction: column;

        align-items: center;

    }

    .tableOval {

        width: 200px;

        height: 100px;

        border-radius: 50%;

        background-color: #e0e0e0;

        display: flex;

        justify-content: center;

        align-items: center;

        font-weight: bold;

        margin-bottom: 10px;

        transition: 0.3s ease;

    }

    .active:hover {

        background-color: $color\_6;

        color: $color\_1;

        box-shadow: 0.8rem 1rem 3rem rgba(0, 0, 0, 0.356);

        transition: 0.3s ease;

    }

    .active{

        background-color: $color\_4;

    }

    .tableDetails{

        border-radius: 8px;

        padding: 5px 15px 10px 15px;

        width: 140px;

        box-shadow: 0.8rem 1rem 3rem rgba(0, 0, 0, 0.356);

        .tableState{

            margin: 5px;

            font-size: 16px;

        }

    }

    .bookingBox {

        background-color: $color\_4;

        padding: 8px 10px;

        border-radius: 6px;

        cursor: pointer;

        font-weight: 200;

        font-size: 16px;

        text-align: center;

        transition: 0.3s ease;

    }

    .bookingBox:hover {

        background-color: $color\_6;

        color: $color\_1;

    }

    .bookingBoxEmpty {

        color: #999;

        font-size: 14px;

    }

}

admin/statistic/page.js

"use client";

import { useEffect, useState } from "react";

import { useRouter } from "next/navigation";

import { useSelector, useDispatch } from "react-redux";

import { BarChart } from "@mui/x-charts/BarChart";

import styles from "./Statistic.module.scss";

import { checkAdminBookings } from "@/app/\_lib/bookingsSlice.js";

import { checkAllPayments } from "@/app/\_lib/paymentsSlice.js";

import { LineChart, PieChart } from "@mui/x-charts";

export default function Home() {

    const router = useRouter();

    const dispatch = useDispatch();

    const { user } = useSelector((state) => state.account);

    const { payments } = useSelector((state) => state.payments);

    const { bookings } = useSelector((state) => state.bookings);

    const [selectedDate, setSelectedDate] = useState(() =>

        new Date().toISOString().split("T")[0]

    );

    const [month, setMonth] = useState("");

    const [monthlyIncome, setMonthlyIncome] = useState(0);

    const [cardPercentage, setCardPercentage] = useState(0);

    const [yearIncome, setYearIncome] = useState(0);

    const [monthlyRevenueData, setMonthlyRevenueData] = useState([]);

    const [dailyRegistrations, setDailyRegistrations] = useState([]);

    const [popularGames, setPopularGames] = useState([]);

    useEffect(() => {

        if (user.email === "") {

        router.push("/");

        }

        if(user.access !== "manager"){

            router.push("/admin");

        }

    }, [user.access, user.email, router]);

    useEffect(() => {

        dispatch(checkAdminBookings(user.email, user.userId, new Date(selectedDate).getFullYear()));

        dispatch(checkAllPayments(user.email, user.userId, new Date(selectedDate).getFullYear()));

    }, [dispatch, selectedDate, user]);

    useEffect(() => {

        if (payments.length > 0 && bookings.length > 0) {

        // Log bookings data structure for debugging

            console.log("Bookings Data:", bookings);

        const currentDate = new Date(selectedDate);

        const currentMonth = currentDate.getMonth();

        const currentYear = currentDate.getFullYear();

        setMonth(currentMonth);

        let monthIncome = 0;

        let yearTotal = 0;

        let cardCount = 0;

        let totalPayments = 0;

        const incomeByMonth = Array(12).fill(0);

        payments.forEach((payment) => {

            if (payment.status !== "successful") return;

            // Find related booking for this payment

            const relatedBooking = bookings.find(

                (booking) => booking.payment\_id === payment.payment\_id

            );

            console.log(

                `Related Booking for Payment ${payment.payment\_id}: ${JSON.stringify(relatedBooking)}`

            );

            if (!relatedBooking) return;

            // Assuming the booking has a `date` field which is a string that can be parsed into a Date object

            const bookingDate = new Date(relatedBooking.date);

            const year = bookingDate.getFullYear();

            const month = bookingDate.getMonth();

            const amount = Number(payment.amount || 0);

            if (year === currentYear) {

                yearTotal += amount;

                incomeByMonth[month] += amount;

            if (month === currentMonth) {

                monthIncome += amount;

            }

            // Check for card payments

            if (payment.card\_or\_cash === true) cardCount++;

                totalPayments++;

            }

        });

        const daysInMonth = new Date(currentYear, currentMonth + 1, 0).getDate();

        const registrationsByDay = Array(daysInMonth).fill(0);

        // Count registrations for each day

        bookings.forEach((booking) => {

            const bookingDate = new Date(booking.date);

            const bookingYear = bookingDate.getFullYear();

            const bookingMonth = bookingDate.getMonth();

            if (bookingYear === currentYear && bookingMonth === currentMonth) {

                const day = bookingDate.getDate(); // 1 to 31

                registrationsByDay[day - 1] += 1;

            }

        });

        const gameCountMap = {};

        bookings.forEach((booking) => {

            const bookingDate = new Date(booking.date);

            const bookingYear = bookingDate.getFullYear();

            if (bookingYear === currentYear) {

                const gameName = booking.game\_name; // Adjust key as needed

                if (gameName) {

                    gameCountMap[gameName] = (gameCountMap[gameName] || 0) + 1;

                }

            }

        });

        const sortedGames = Object.entries(gameCountMap)

            .sort((a, b) => b[1] - a[1])

            .map(([name, count], index) => ({

                id: index,

                value: count,

                label: name

            }));

        setPopularGames(sortedGames);

        setDailyRegistrations(registrationsByDay);

        setMonthlyIncome(monthIncome);

        setYearIncome(yearTotal);

        setCardPercentage(totalPayments ? ((cardCount / totalPayments) \* 100).toFixed(2) : 0);

        setMonthlyRevenueData(incomeByMonth);

        } else {

            console.log("No payments or bookings found.");

        }

    }, [payments, bookings, selectedDate]);

    const handleDateClick = (value) => {

        setSelectedDate(value);

    };

    return (

        <>{user.access === "manager" &&

            <div className={styles.container}>

                <div className={styles.summed\_part}>

                    <h2 className={styles.main\_title} >Statistics</h2>

                    <div className={styles.datePickerContainer}>

                        <label htmlFor="datePicker">Date: </label>

                        <input

                            id="datePicker"

                            type="date"

                            value={selectedDate}

                            onChange={(e) => handleDateClick(e.target.value)}

                            className={styles.datePicker}

                        />

                    </div>

                    <h4 className={styles.secondary}>

                        Total revenue for current month: <p>{monthlyIncome} $</p>

                    </h4>

                    <h4 className={styles.secondary}>

                        Card payments percentage: <p>{cardPercentage} %</p>

                    </h4>

                    <h4 className={styles.secondary}>

                        Year income: <p>{yearIncome} $</p>

                    </h4>

                </div>

                <div className={styles.charts}>

                    <h4 className={styles.tit}>

                        Revenue for selected year

                    </h4>

                    <LineChart

                        height={190}

                        width={500}

                        xAxis={[{

                            data: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12],

                        }]}

                        series={[{

                            data: monthlyRevenueData,

                            area: true,

                            baseline: 'min',

                            color: 'rgb(0, 223, 129)',

                        }]}

                    />

                    <h4 className={styles.tit}>

                        Registrations for chosen month

                    </h4>

                    <BarChart

                        series={[{ data: dailyRegistrations, color: 'rgb(0, 223, 129)', }]}

                        height={200}

                        xAxis={[{

                            data: Array.from({ length: dailyRegistrations.length }, (\_, i) => (i + 1).toString())

                        }]}

                    />

                    {popularGames.length > 0 ? (

                        <>

                            <h4 className={styles.tit}>

                                Top 5 most popular games

                            </h4>

                            <PieChart

                                series={[

                                    {

                                        data: popularGames.map((game, index) => ({

                                            ...game,

                                            id: index,

                                            color: [

                                                'rgb(0, 223, 129)',

                                                ' rgb(17, 126, 57)',

                                                'rgb(3, 98, 76)',

                                                ' rgb(44, 194, 149)',

                                                'rgb(3, 34, 33)'

                                            ][index % 5] // cycle colors if more than 5

                                        }))

                                    }

                                ]}

                                width={250}

                                height={250}

                            />

                        </>

                    ) : (

                        <p>No game data available for selected year.</p>

                    )}

                </div>

            </div>

        }</>

    );

}

admin/statistic/Statistic.module.scss

@import "../../globals.scss";

.summed\_part{

    z-index: 3;

    background-color: $color\_1;

    padding: 8px;

    border-radius: 15px;

    padding:  15px 40px;

    margin: 70px 50px 0 0;

    display: flex;

    flex-direction: column;

    align-items: center;

    justify-self: start;

    align-self: start;

    box-shadow: 1rem 1.4rem 10rem rgba(0, 0, 0, 0.354);

    .main\_title{

        @extend %font1;

        font-size: 60px;

        letter-spacing: 16px;

        font-weight: 900;

        color: $color\_5;

        margin: 15px 0 20px 0 ;

        font-weight: bolder;

        padding-left: 7px;

        color: $color\_5;

    }

    .secondary{

        @extend %font1;

        display: flex;

        flex-direction: row;

        align-items: center;

        justify-content: space-between;

        font-size: 20px;

        width: 90%;

        margin: 10px 5px;

        padding: 10px;

        border-radius: 10px;

        background-color: $color\_4;

        p{

            background-color: $color\_1;

            color: $color\_8;

            margin: 0 5px;

            padding: 5px 10px;

            border-radius: 5px;

        }

    }

    .datePickerContainer{

        @extend %font1;

        display: flex;

        align-self: start;

        align-items: center;

        justify-content: space-around;

        font-size: 20px;

        padding: 8px 20px;

        box-shadow: 0.4rem 0.4rem 1rem rgba(0, 0, 0, 0.274);

        border-radius: 10px;

        margin-bottom: 30px;

        margin-left: 10px;

        .datePicker{

            padding: 3px;

            border-radius: 10px;

            background-color: $color\_4;

            color: $color\_1;

            margin-left: 20px;

            font-size: 16px;

        }

    }

}

.container{

    display: flex;

    flex-direction: row;

    margin: 50px 20px;

    padding: 0 30px;

    height: 800;

    justify-content: space-around;

}

.tit{

    @extend %font1;

    display: flex;

    flex-direction: row;

    align-items: center;

    justify-content: center;

    letter-spacing: 3px;

    font-size: 20px;

    width: 90%;

    margin: 10px 5px;

    padding: 10px;

    border-radius: 10px;

}

.charts{

    height: 700px;

    display: flex;

    align-content: space-between;

    flex-direction: column;

}

admin/logout/page.js

"use client"

import { useEffect } from "react";

import { useRouter } from 'next/navigation';

import { useSelector } from "react-redux";

import LogOut from "../../user/logout/page.js";

export default function Home() {

  const router = useRouter();

  const { user } = useSelector((state) => state.account);

  useEffect(function(){

    if(user.access === ""){

      router.push("/");

    }

  }, [user.access, router]);

  return (

    <>{user.access &&

        <LogOut/>

    }</>

  );

}

admin/\_game\_state\_list/page.js

"use client"

import { useEffect, useState } from "react";

import { useRouter } from 'next/navigation';

import { useSelector, useDispatch } from "react-redux";

import { getGames, deleteGame } from "../../\_lib/gameSlice.js";

import styles from "./Games.module.scss";

import Loading from "../../loading.js";

import Button from "@/app/\_components/Button.js";

import AddGame from "../\_add\_game/addGame.js";

export default function Home() {

  const router = useRouter();

  const [show, setShow] = useState(false);

  const { user } = useSelector((state) => state.account);

  const { games, isLoading, error } = useSelector((store) => store.game);

  const dispatch = useDispatch();

  useEffect(function(){

    if(user.access === ""){

      router.push("/");

    }

    dispatch(getGames());

  }, [user.access, router, dispatch]);

  function handleGameDelete(id){

    dispatch(deleteGame(user.email, user.userId, id));

  }

  if (isLoading) return <Loading />;

  if (error) return <p>Error loading games: {error}</p>;

  return (

    <div>{user.access &&

      <div className={styles.container}>

        <div className={user.access === "manager" ? styles.top\_container : null}>

          <div>

            <h1 className={styles.Gametitle}>Game List</h1>

            <div className={styles.line}></div>

          </div>

          {user.access === "manager" &&

            <div>

              <Button

                name={"button\_template1"}

                width={200}

                height={60}

                action={() => setShow(true)}

              >

                Add new game

              </Button>

            </div>

          }

        </div>

        <AddGame user={user} active={show} setActive={setShow}/>

        <ul className={styles.GameList}>

          {games.map((game) => (

            <li key={game.game\_id} className={`${styles.game} ${styles.active}`}>

              <h2 className={styles.caption}>{game.game\_name}</h2>

              <p className={styles.players}>

                <strong className={styles.desc\_cap}>Players:</strong> {game.players}

              </p>

              <p className={styles.players}>

                <strong className={styles.desc\_cap}>Goal:</strong> {game.goal}

              </p>

              <div className={styles.status}>

                <strong className={styles.desc\_cap}>Status:</strong> <p>{game.status}</p>

              </div>

              <p className={styles.rules}>

                <strong className={styles.desc\_cap}>Rules:</strong>{game.rules}

              </p>

              {user.access === "manager" &&

                <Button

                  name={"button\_template2"}

                  width={200}

                  height={60}

                  action={() => handleGameDelete(game.game\_id)}

                >

                  Delete

                </Button>

              }

            </li>

          ))}

        </ul>

      </div>

    }</div>

  );

}

admin/\_game\_state\_list/Games.module.scss

@import "../../globals.scss";

.top\_container{

    display: flex;

    flex-direction: row;

    align-items: center;

    justify-content: space-between;

    margin: 0px 80px;

}

.Gametitle{

    color: $color\_7;

    @extend %font1;

    font-size: 60px;

    margin: 30px 0 20px 0;

    letter-spacing: 5px;

    word-spacing: 10px;

    z-index: 2;

    display: flex;

    justify-self: center;

}

.line{

    background-color: $color\_6;

    width: 180px;

    height: 15px;

    border-radius: 10px;

    display: flex;

    justify-self: center;

}

.GameList{

    flex-wrap: wrap;

    display: flex;

    flex-direction: row;

    justify-content: space-around;

    padding: 0px;

    .game{

        list-style-type: none;

        position: relative;

        width: 480px;

        height: 300px;

        flex-direction: column;

        margin: 40px 20px ;

        background-color: $color\_4;

        border-radius: 15px;

        padding: 10px 40px;

        display: flex;

        align-items: center;

        justify-self: center;

        justify-content: space-between;

        box-shadow: 1rem 1.4rem 3rem rgba(0, 0, 0, 0.356);

        transition: 0.3s ease-in-out;

        @extend %font1;

        font-weight: 200;

        font-size: 14px;

        .caption{

            @extend %font1;

            font-weight: 700;

            font-size: 30px;

            width: 100%;

            text-align: center;

            padding: 10px;

            margin: 10px 0;

            background-color: $color\_1;

            border-radius: 10px;

        }

        .desc\_cap{

            font-weight: 700;

            font-size: 18px;

            margin-right: 10px;

        }

        .players{

            display: flex;

            flex-direction: row;

            width: 100%;

            transition: 0.3s ease-in-out;

            text-align: start;

            margin: 6px 0;

        }

        .status{

            display: flex;

            flex-direction: row;

            align-items: center;

            width: 100%;

            transition: 0.3s ease-in-out;

            text-align: start;

            margin: 6px 0;

            p{

                background-color: $color\_1;

                margin: 0 5px;

                padding: 5px 10px;

                border-radius: 5px;

                font-size: 16px;

                font-weight: 600;

            }

        }

        .rules{

            transition: 0.3s ease-in-out;

            text-align: justify;

            margin: 8px 0;

        }

    }

    .active{

        height: 390px;

    }

}

admin/champ/page.js

"use client";

import { useEffect, useState } from "react";

import AddMatch from "./\_create\_match/page";

import styles from "./Champ.module.scss";

import { useRouter } from 'next/navigation';

import { useSelector, useDispatch } from "react-redux";

import { getMatches } from "@/app/\_lib/matchesSlice.js";

import CheckMatch from "./\_chack\_match/checkMatch";

function parseStage(stage) {

    const [id, round] = stage.split(".");

    return { matchId: parseInt(id, 10), round };

}

export default function Championship() {

    const [active, setActive] = useState(false);

    const [showMatch, setShowMatch] = useState(false);

    const [selectedMatch, setSelectedMatch] = useState({});

    const [place, setPlace] = useState("");

    const { user } = useSelector((state) => state.account);

    const { matches } = useSelector((state) => state.matches);

    const dispatch = useDispatch();

    const router = useRouter();

    useEffect(() => {

        if (user.email === "") {

            router.push("/");

        }

        if (user.access !== "manager") {

            router.push("/admin");

        }

    }, [user.email, user.access, router]);

    useEffect(() => {

        dispatch(getMatches());

    }, [dispatch]);

    const getRoundLabel = (i) => {

        return i === 0 ? "1/16" : i === 1 ? "1/8" : i === 2 ? "1/4" : i === 3 ? "1/2" : null;

    };

    return (

        <>

            {user.access === "manager" &&

                <div className={styles.bracketWrapper}>

                    <h2 className={styles.main\_title}>Monopoly Championship</h2>

                    {active && <AddMatch handleActiveClick={setActive} place={place} />}

                    {showMatch && <CheckMatch active={showMatch} match={selectedMatch} handleActiveClick={setShowMatch} />}

                    <div className={styles.bracket}>

                        {/\* Left Side \*/}

                        <div className={styles.side}>

                            {[...Array(4)].map((\_, i) => (

                                <div className={styles.round} key={`left-${i}`}>

                                    {[...Array(2 \*\* (3 - i))].map((\_, j) => {

                                        const matchId = i \* 10 + j + 100;

                                        const round = getRoundLabel(i);

                                        const isFilled = matches.some((match) => {

                                            const { matchId: id, round: r } = parseStage(match.stage);

                                            return id === matchId && r === round;

                                        });

                                        return (

                                            <div

                                                className={`${styles.match} ${isFilled ? styles.filled : ""}`}

                                                key={i \* 10 + j}

                                                onClick={() => {

                                                    const stage = `${matchId}.${round}`;

                                                    const matched = matches.find((match) => match.stage === stage);

                                                    if (matched) {

                                                        setSelectedMatch(matched);

                                                        setShowMatch(true);

                                                    } else {

                                                        setActive(true);

                                                    }

                                                    setPlace(stage);

                                                }}

                                            >

                                                {round}

                                            </div>

                                        );

                                    })}

                                </div>

                            ))}

                        </div>

                        {/\* Final \*/}

                        <div className={styles.final}>

                            <div className={styles.match}

                            onClick={() => {

                                const stage = `${1}.${"final"}`;

                                const matched = matches.find((match) => match.stage === stage);

                                if (matched) {

                                    setSelectedMatch(matched);

                                    setShowMatch(true);

                                } else {

                                    setActive(true);

                                }

                                setPlace(stage);

                            }}

                            >

                                Final

                            </div>

                        </div>

                        {/\* Right Side \*/}

                        <div className={styles.side\_2}>

                            {[...Array(4)].map((\_, i) => (

                                <div className={styles.round} key={`right-${i}`}>

                                    {[...Array(2 \*\* (3 - i))].map((\_, j) => {

                                        const matchId = i \* 10 + j;

                                        const round = getRoundLabel(i);

                                        const isFilled = matches.some((match) => {

                                            const { matchId: id, round: r } = parseStage(match.stage);

                                            return id === matchId && r === round;

                                        });

                                        return (

                                            <div

                                                className={`${styles.match} ${isFilled ? styles.filled : ""}`}

                                                key={i \* 10 + j}

                                                onClick={() => {

                                                    const stage = `${matchId}.${round}`;

                                                    const matched = matches.find((match) => match.stage === stage);

                                                    if (matched) {

                                                        setSelectedMatch(matched);

                                                        setShowMatch(true);

                                                    } else {

                                                        setActive(true);

                                                    }

                                                    setPlace(stage);

                                                }}

                                            >

                                                {round}

                                            </div>

                                        );

                                    })}

                                </div>

                            ))}

                        </div>

                    </div>

                </div>

            }

        </>

    );

}

admin/champ/Champ.module.scss

@import "../../globals.scss";

.bracketWrapper {

    padding: 20px;

    display: flex;

    justify-self: center;

    flex-direction: column;

    align-items: center;

    width: 90%;

}

.main\_title{

    @extend %font1;

    font-size: 50px;

    letter-spacing: 5px;

    font-weight: 900;

    color: $color\_5;

    margin: 15px 0 20px 0 ;

    font-weight: bolder;

    padding-left: 7px;

    color: $color\_5;

}

.bracket {

    display: flex;

    flex-direction: row;

    justify-content: center;

    width: 100%;

    flex-wrap: wrap;

    gap: 10px;

}

.side {

    width: 40vw;

    display: flex;

    flex-direction: row;

    justify-content: space-around;

    gap: 10px;

    height: 50vh;

}

.side\_2 {

    width: 40vw;

    display: flex;

    flex-direction: row-reverse;

    justify-content: space-around;

    gap: 10px;

    height: 50vh;

}

.round {

    display: flex;

    flex-direction: column;

    justify-content: space-around;

    gap: 10px;

    min-width: 70px;

}

.match {

    background-color: #bebebe;

    display: flex;

    align-items: center;

    justify-content: center;

    border-radius: 6px;

    width: 100%;

    height: 40px;

    transition: all 0.3s ease;

}

.match:hover {

    background-color: $color\_4;

    transform: scale(1.05);

}

.final {

    display: flex;

    align-items: center;

    justify-content: center;

    padding: 0 20px;

    .match {

        background-color: $color\_8;

        width: 80px;

        height: 40px;

    }

}

.filled{

    background-color: $color\_4;

}

// Responsive design

@media (max-width: 768px) {

    .round {

        min-width: 35px;

        .match {

            height: 25px;

        }

    }

    .final .match {

        width: 60px;

        height: 35px;

    }

}

@media (max-width: 480px) {

    .bracket {

        flex-direction: column;

        align-items: center;

    }

    .side {

        flex-direction: column;

        align-items: center;

    }

    .round {

        flex-direction: row;

        flex-wrap: wrap;

        justify-content: center;

    }

    .match {

        width: 80px;

        margin: 2px;

    }

    .final {

        padding: 20px 0;

    }

}

admin/champ/\_create\_match/page.js

"use client";

import Button from "@/app/\_components/Button";

import styles from "./AddMatch.module.scss";

import { useSelector, useDispatch } from "react-redux";

import { getMatches, createMatch } from "@/app/\_lib/matchesSlice.js";

import { useEffect, useState } from "react";

const availableTimes = ["9:00", "10:00", "11:00", "12:00", "13:00", "14:00", "15:00", "16:00", "17:00", "18:00", "19:00", "20:00",];

const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;

const phoneRegex = /^\d{9}$/;

export default function AddMatch({handleActiveClick, place}) {

    const [selectedDate, setSelectedDate] = useState();

    const [playerCount, setPlayerCount] = useState(2);

    const [inputError, setInputError] = useState("");

    const [players, setPlayers] = useState([{ name: "", email: "", phone: "" }, { name: "", email: "", phone: "" }]);

    const [winner, setWinner] = useState("");

    const [selectedTime, setSelectedTime] = useState("");

    const game = "Monopoly";

    const { user } = useSelector((state) => state.account);

    const dispatch = useDispatch();

    useEffect(function(){

        setSelectedDate(new Date().toISOString().split("T")[0]);

    }, []);

    const handlePlayerChange = (index, field, value) => {

        const updated = [...players];

        updated[index][field] = value;

        setPlayers(updated);

    };

    const handleDateClick = (date) => setSelectedDate(date);

    const handlePlayerCountChange = (e) => {

        const count = parseInt(e.target.value);

        if (count >= 2 && count <= 12) {

            setPlayerCount(count);

            const updatedPlayers = Array.from({ length: count }, (\_, i) => players[i] || { name: "", email: "", phone: "" });

            setPlayers(updatedPlayers);

        }

    };

    const validPlayers = players.filter(

        p => p.name && emailRegex.test(p.email) && phoneRegex.test(p.phone)

    );

    const allEmailsValid = validPlayers.length === playerCount;

    // to create new booking

    function handleCreateMatch(){

        setInputError("");

        const hasInvalidPlayer = players.some((player, index) => {

            if (index !== 0) {

                const hasMissingFields = !player.email || !player.name || !player.phone;

                const isInvalidEmail = !emailRegex.test(player.email || "");

                const isInvalidPhone = !phoneRegex.test(player.phone || "" );

            return hasMissingFields || isInvalidEmail || isInvalidPhone;

            }

        });

        if (hasInvalidPlayer) {

            setInputError("Please enter valid name, email, and phone for some players");

            return;

        }

        if(!selectedTime){

            setInputError("Please choose the time");

            return;

        }

        if(!winner){

            setInputError("Please choose the winner of the match");

            return;

        }

        dispatch(createMatch(user.email, user.userId, playerCount, players, winner, selectedDate, selectedTime, place));

        dispatch(getMatches());

        handleActiveClick(false);

    }

    return (

        <div className={styles.addM\_container}>

            <div className={styles.sub\_container}>

                <div className={styles.caption\_part}>

                    <h1 className={styles.title}>Create new match</h1>

                    <div className={styles.datePickerContainer}>

                        <label htmlFor="datePicker">Date:</label>

                        <input

                            id="datePicker"

                            type="date"

                            value={selectedDate}

                            onChange={(e) => handleDateClick(e.target.value)}

                            className={styles.datePicker}

                        />

                    </div>

                </div>

                <div className={styles.formGroup1}>

                    <p className={styles.cross} onClick={() => handleActiveClick(false)}>&times;</p>

                    <label>Number of players:</label>

                    <input

                        type="number"

                        min="2"

                        max="12"

                        value={playerCount}

                        onChange={handlePlayerCountChange}

                    />

                </div>

                {inputError && <p className={styles.error}>{inputError}</p>}

                <div className={styles.button\_part}>

                    {players.map((player, index) => (

                        <div key={index} className={styles.playerForm}>

                            <h3>Player {index + 1}</h3>

                            <input

                                required

                                type="text"

                                placeholder="Name"

                                value={player.name}

                                onChange={(e) => handlePlayerChange(index, "name", e.target.value)}

                            />

                            <input

                                required

                                type="email"

                                placeholder="Email"

                                value={player.email}

                                onChange={(e) => handlePlayerChange(index, "email", e.target.value)}

                            />

                            <input

                                required

                                type="tel"

                                placeholder="Phone Number"

                                value={player.phone}

                                onChange={(e) => handlePlayerChange(index, "phone", e.target.value)}

                            />

                        </div>

                    ))}

                    <div className={styles.formGroup}>

                        <label>Select Time:</label>

                        <select value={selectedTime} onChange={(e) => setSelectedTime(e.target.value)}>

                            <option value="">-- Select Time --</option>

                            {availableTimes.map((time) => (

                                <option key={time} value={time}>{time}</option>

                            ))}

                        </select>

                    </div>

                    {allEmailsValid && (

                        <div className={styles.formGroup}>

                            <label>Select Winner:</label>

                            <select value={winner} onChange={(e) => setWinner(e.target.value)}>

                                <option value="">-- Choose Winner --</option>

                                {players.map((player, i) => (

                                    <option key={i} value={player.email}>{player.name || `Player ${i + 1}`} - {player.email}</option>

                                ))}

                            </select>

                        </div>

                    )}

                    <div className={styles.formGroup2}>

                        <label>Game:</label>

                        <p className={styles.game}>{game}</p>

                    </div>

                    <div className={styles.button\_sec}>

                        <p>Please add the new match only after it was held in the club and you already know the winner.</p>

                        <Button

                            name={"button\_template1"}

                            width={490}

                            action={handleCreateMatch}

                        >

                            Register Match

                        </Button>

                    </div>

                </div>

            </div>

        </div>

    );

}

admin/champ/\_create\_match/ AddMatch.module.scss

@import "../../../globals.scss";

.addM\_container{

    position: fixed;

    top: 0;

    left: 0;

    width: 100vw;

    height: 100vh;

    display: flex;

    justify-content: center;

    align-items: center;

    z-index: 9;

}

.sub\_container{

    position: fixed;

    flex-wrap: wrap;

    @extend %font1;

    background-color: $color\_1;

    box-shadow: 1rem 2rem 25rem 2rem rgb(0, 0, 0);

    border-radius: 15px;

    margin: 60px 20px 80px 20px;

    padding: 20px 40px;

    z-index: 4;

    display: flex;

    flex-direction: row;

    width: 80%;

}

.caption\_part{

    width: 50%;

}

.cross{

    margin: 0;

    height: 35px;

    width: 35px;

    background-color: $color\_4;

    border-radius: 7px;

    display: flex;

    justify-content: center;

    align-items: center;

    font-size: 26px;

    color: $color\_1;

    cursor: pointer;

}

.datePickerContainer{

    @extend %font1;

    display: flex;

    align-self: start;

    align-items: center;

    justify-content: space-around;

    font-size: 20px;

    padding: 8px 20px;

    box-shadow: 0.4rem 0.4rem 1rem rgba(0, 0, 0, 0.274);

    border-radius: 10px;

    margin-bottom: 30px;

    width: 200px;

    .datePicker{

        padding: 3px;

        border-radius: 10px;

        background-color: $color\_4;

        color: $color\_1;

        margin-left: 20px;

        font-size: 16px;

    }

}

.formGroup {

    margin: 20px 0;

    display: flex;

    flex-direction: column;

    label {

        font-weight: bold;

        margin-bottom: 8px;

        color: $color\_4;

    }

    input,

    select {

        padding: 10px;

        border: 1px solid #ccc;

        border-radius: 6px;

        font-size: 16px;

        background-color: #fff;

        transition: border-color 0.2s ease;

        &:focus {

        outline: none;

        border-color: $color\_4;

        }

    }

}

.formGroup2 {

    margin: 20px 0;

    display: flex;

    flex-direction: row;

    align-items: center;

    label {

        font-weight: bold;

        margin-bottom: 8px;

        color: $color\_4;

        margin-right: 30px;

    }

    .game{

        height: 44px;

        background-color: $color\_4;

        width: 150px;

        border-radius: 15px;

        margin: 10px 0 10px 0;

        padding: 0;

        display: flex;

        align-items: center;

        justify-content: center;

        color: $color\_7;

        @extend %font1;

    }

}

.formGroup1 {

    width: 50%;

    margin: 20px 0;

    display: flex;

    flex-direction: column;

    align-items: flex-end;

    label {

        font-weight: bold;

        margin-bottom: 8px;

        color: $color\_4;

        margin-right: 60px;

    }

    input,

    select {

        padding: 10px;

        border: 1px solid #ccc;

        border-radius: 6px;

        font-size: 16px;

        background-color: #fff;

        transition: border-color 0.2s ease;

        width: 200px;

        &:focus {

        outline: none;

        border-color: $color\_4;

        }

    }

}

.button\_part{

    width: 100%;

}

.error{

    background-color: red;

    color: $color\_1;

    @extend %font1;

    padding: 10px;

    width: 98.5%;

    border-radius: 7px;

    text-align: center;

}

.playerForm {

    background-color: $color\_4;

    padding: 15px;

    border-radius: 10px;

    margin-bottom: 25px;

    display: flex;

    flex-direction: row;

    flex-wrap: wrap;

    justify-content: space-between;

    h3 {

        width: 10%;

        margin: 5px;

        color: $color\_7;

        font-size: 24px;

        display: flex;

        align-self: center;

    }

    .youLabel{

        width: 20%;

        margin: 0 40% 0 0;

        display: flex;

        justify-content: center;

        align-items: center;

        background-color: $color\_1;

        padding: 8px;

        border-radius: 10px;

    }

    input{

        padding-left: 16px;

        width: 27%;

        height: 40px;

        font-family: Poppins;

        font-weight: 600;

        font-size: 16px;

        line-height: 20px;

        border: 1px solid #bfbfbf;

        box-shadow: inset 0 0 0 55px #fff!important;

        border-radius: 7px;

        margin: 0;

        outline: none;

    }

    input:-webkit-autofill {

        -webkit-box-shadow: inset 0 0 0 55px #fff!important;

        box-shadow: inset 0 0 0 55px #fff!important;

    }

    input :placeholder-shown{

        color: $color\_5;

        letter-spacing: 1px;

    }

}

.submit {

    padding: 12px 24px;

    background-color: $color\_4;

    color: #fff;

    font-weight: bold;

    font-size: 16px;

    border: none;

    border-radius: 8px;

    cursor: pointer;

    transition: background-color 0.2s ease;

    &:hover {

        background-color: darken($color\_4, 10%);

    }

}

.title{

    color: #000;

    margin-bottom: 20px;

}

.date{

    font-size: 16px;

    margin-top: 0px;

}

.button\_sec{

    display: flex;

    flex-direction: row;

    align-items: center;

    justify-content: space-between;

    p{

        font-size: 14px;

        font-weight: 200;

        width: 50%;

        text-align: justify;

    }

}

admin/champ/\_check\_match/checkMatch.js

import Button from "@/app/\_components/Button";

import styles from "./CheckMatch.module.scss";

export default function CheckMatch({ active, match, handleActiveClick}){

    return(

        <form className={`${styles.section7\_login} ${active ? styles.active : null}`}>

            { match &&

                <>

                    <h2 className={styles.main\_title\_caption}>

                        Date: {match.date ? new Intl.DateTimeFormat('en-CA').format(new Date(match.date)) : "Invalid Date"}

                        <p className={styles.cross} onClick={() => handleActiveClick(false)}>&times;</p>

                    </h2>

                    <h4 className={`${styles.main\_title\_caption} ${styles.secondary}`}>Time: <p>{match.time}</p></h4>

                    <h4 className={`${styles.main\_title\_caption} ${styles.secondary}`}>Winner: <p>{match.winner\_email}</p></h4>

                    <h4 className={`${styles.main\_title\_caption} ${styles.third}`}>Game name: <p>Monopoly</p></h4>

                    <div className={styles.player\_list}>

                    {match?.player\_emails?.length > 0 && match.player\_emails.map(player => (

                        <p key={player} className={styles.member}>{player}</p>

                    ))}

                    </div>

                </>

            }

        </form>

    );

}

admin/champ/\_check\_match/ CheckMatch.module.scss

@import "../../../globals.scss";

.main\_title\_caption{

    @extend %font1;

    font-size: 45px;

    letter-spacing: 5px;

    font-weight: 900;

    color: $color\_5;

    margin: 30px 0 0 0 ;

    font-weight: bolder;

    text-align: center;

    color: $color\_5;

    display: flex;

    flex-direction: row;

    justify-content: space-between;

    align-items: start;

    width: 90%;

    .cross{

        margin: 0;

        background-color: $color\_1;

        padding: 0px 8px 0px 15px;

        border-radius: 7px;

        cursor: pointer;

    }

}

.section7\_login{

    position: fixed;

    z-index: 10;

    width: 750px;

    flex-direction: column;

    margin-left: 0vw;

    padding-left: 60px;

    background-color: $color\_4;

    padding: 8px;

    border-radius: 15px;

    display: none;

    align-items: center;

    justify-self: center;

    align-self: center;

    box-shadow: 1rem 1.4rem 25rem rgb(0, 0, 0);

}

.section7\_login::after{

    content: "";

    top: 0;

    left: 0;

    position: absolute;

    width: 100%;

    height: 100%;

    opacity: 0;

    visibility: hidden;

    transition: 0.5s;

}

.player\_list{

    margin: 20px 0;

    background-color: $color\_1;

    width: 90%;

    display: flex;

    flex-direction: column;

    align-items: start;

    justify-content: center;

    border-radius: 10px;

    .member{

        margin-left: 25%;

        color: $color\_3;

    }

}

.section7\_consultation\_description{

    width: 90%;

    font-size: 16px;

    line-height: 22px;

    letter-spacing: 1.22px;

    color: $color\_5;

    margin: 0 0 35px 0;

    text-align: justify;

    @extend %font2;

    font-weight: 200;

}

.active{

    display: flex;

}

.secondary{

    display: flex;

    flex-direction: row;

    align-items: center;

    justify-content: start;

    height: 50px;

    font-size: 26px;

    p{

        margin-left: 20px;

    }

}

.third{

    display: flex;

    flex-direction: row;

    align-items: center;

    justify-content: start;

    font-size: 25px;

    height: 50px;

    p{

        margin-left: 30px;

    }

}

admin/\_change\_payment/payment.js

"use client";

import { useEffect, useState } from "react";

import { useDispatch, useSelector } from "react-redux";

import Button from "@/app/\_components/Button";

import styles from "./Payment.scss";

import { checkAllPayments, updatePayment, setError } from "@/app/\_lib/paymentsSlice.js";

export default function Payment({ booking, handleActiveClick, date }) {

    const [id, setID] = useState("");

    const [amount, setAmount] = useState("");

    const [status, setStatus] = useState("");

    const [card\_or\_cash, setCard\_or\_cash] = useState(false);

    const { payments, error, isLoading } = useSelector((state) => state.payments);

    const { user } = useSelector((state) => state.account);

    const dispatch = useDispatch();

    useEffect(() => {

        if (booking) {

            dispatch(setError(""));

            dispatch(checkAllPayments(user.email, user.userId, new Date(date).getFullYear()));

        }

    }, [dispatch, booking]);

    useEffect(() => {

        if (booking && payments.length > 0) {

            const relatedPayment = payments.find((payment) => payment.payment\_id === booking.payment\_id);

            if (relatedPayment) {

                setID(relatedPayment.payment\_id || "");

                setAmount(relatedPayment.amount || "");

                setStatus(relatedPayment.status || "");

                setCard\_or\_cash(relatedPayment.card\_or\_cash || false);

            }

        }

    }, [booking, payments]);

    function handleClick() {

        dispatch(setError(""));

        const validStatuses = ["booked", "waiting", "successful", "declined"];

        if (!validStatuses.includes(status)) {

            return dispatch(setError("Please enter a valid status"));

        }

        dispatch(updatePayment(user.email, user.userId, id, status, amount, card\_or\_cash));

        handleActiveClick(false);

    }

    return (

        <div className="authenticate">

            <div className="section7-login">

                <h2 className="main\_title\_caption">Payment details

                    <p className="cross" onClick={() => handleActiveClick(false)}>&times;</p>

                </h2>

                <p className="section7\_consultation\_description">

                    Payment details for your current booking. Booking payment can only have the following states: successful, waiting, booked, declined.

                </p>

                <p className="email">{amount}</p>

                <select className="name" value={status} onChange={(e) => setStatus(e.target.value)}>

                    <option value="">-- Select status --</option>

                    <option value="booked">booked</option>

                    <option value="waiting">waiting</option>

                    <option value="successful">successful</option>

                    <option value="declined">declined</option>

                </select><br />

                {error && <h2 className="error">{error}</h2>}

                <div className="check\_box\_sec">

                    <div className="check\_box\_sec2">

                        <input

                            id="cons\_check"

                            type="checkbox"

                            checked={card\_or\_cash}

                            onChange={() => setCard\_or\_cash(!card\_or\_cash)}

                            className="section7\_checkbox"

                        />

                        <p className="section7\_checkbox\_label">

                            If customer payed by the card it should be checked, cash = unchecked.

                        </p>

                    </div>

                    <Button action={handleClick} name="button\_template2">

                        <p>{isLoading ? "Loading..." : "Submit"}</p>

                    </Button>

                </div>

            </div>

        </div>

    );

}

admin/\_change\_payment/Payment.scss

@import "../../globals.scss";

.authenticate{

    z-index: 99;

    position: fixed;

    top: 0;

    left: 0;

    width: 100vw;

    height: 100vh;

    display: flex;

    justify-content: center;

    align-items: center;

    padding-bottom: 10px;

    select{

        margin: 20px 0 0 10px;

        height: 40px;

        color: $color\_1;

        background-color: $color\_3;

        border-radius: 10px;

        border: none;

    }

}

.main\_title\_caption{

    @extend %font1;

    width: 89%;

    font-size: 45px;

    letter-spacing: 5px;

    font-weight: 900;

    color: $color\_5;

    margin: 20px 0 30px 0 ;

    font-weight: bolder;

    text-align: center;

    color: $color\_5;

    display: flex;

    flex-direction: row;

    justify-content: space-between;

    align-content: space-between;

}

.section7-login{

    position: relative;

    width: 550px;

    flex-direction: column;

    padding-left: 60px;

    margin: 80px 0 80px 0;

    background-color: $color\_4;

    padding: 8px 10px 20px 10px;

    border-radius: 15px;

    display: flex;

    align-items: center;

    justify-self: center;

    box-shadow: 1rem 1.4rem 3rem rgba(0, 0, 0, 0.356);

    select{

        margin-left: 0px;

        height: 56px;

        width: 490px;

        border-radius: 15px;

        background-color: #fff;

        color: $color\_4;

        @extend %font1;

        border: 1px solid rgba(0, 0, 0, 0.235);

    }

}

.cross{

    margin: 0;

    background-color: $color\_1;

    padding: 0px 9px 0px 14px;

    border-radius: 7px;

    color: $color\_7;

    cursor: pointer;

}

.section7-login::after{

    content: "";

    top: 0;

    left: 0;

    position: absolute;

    width: 100%;

    height: 100%;

    opacity: 0;

    visibility: hidden;

    transition: 0.5s;

}

.section7-consultation{

    text-align: start;

    font-size: 43px;

    width: 400px;

    text-align: center;

}

.section7\_consultation\_description{

    width: 88%;

    font-size: 19px;

    line-height: 22px;

    letter-spacing: 1.22px;

    color: $color\_5;

    margin: 0 0 15px 0;

    text-align: justify;

    @extend %font2;

    font-weight: 200;

}

.name.\_error{

    box-shadow: 0 0 15px red;

}

.check\_box\_sec{

    margin-bottom: 15px;

}

.check\_box\_sec2{

    margin-top: 5px;

    display: flex;

    flex-direction: row;

}

.section7\_checkbox\_label{

    width: 260px;

    height: 40px;

    letter-spacing: 1.51px;

    color: $color\_5;

    margin: 5px 16px 0 25px;

    text-align: start;

    @extend %font2;

    color: $color\_5;

}

.email{

    height: 54px;

    background-color: $color\_1;

    width: 490px;

    border: 1px solid rgba(0, 0, 0, 0.235);

    border-radius: 15px;

    margin: 10px 0 10px 0;

    padding: 0;

    display: flex;

    align-items: center;

    justify-content: center;

    color: $color\_4;

    @extend %font1;

}

admin/\_bookings\_list/BookingsList.js

"use client"

import styles from "./BookingsList.module.scss";

import { useSelector, useDispatch } from "react-redux";

import { useEffect, useState } from "react";

import { deleteBooking } from "@/app/\_lib/bookingsSlice.js";

import Button from "@/app/\_components/Button";

import Payment from "../\_change\_payment/payment";

export default function Bookings({ user, active, tableId, date, handleListClick, handleActiveClick}){

    const { bookings } = useSelector((state) => state.bookings);

    const dispatch = useDispatch();

    const [bookingList, setBookingList] = useState([]);

    const [showPayment, setShowPayment] = useState(false);

    const [selectedBooking, setSelectedBooking] = useState({});

    useEffect(() => {

        const filteredBookings = bookings.filter(

            b => {

                const bookingDate = new Date(b.date);

                bookingDate.setDate(bookingDate.getDate() + 1);

                const bookingDateOnly = bookingDate.toISOString().split("T")[0];

                return b.table\_id === tableId && bookingDateOnly === date && b.time;

            }

        );

        setBookingList(filteredBookings);

    }, [bookings, tableId, date]);

    function handleDelete(id){

        handleActiveClick(false);

        dispatch(deleteBooking(user.email, user.userId, id));

    }

    function handlePayment(booking){

        setShowPayment(true);

        setSelectedBooking(booking);

    }

    return(

        <form className={`${styles.section7\_login} ${active ? styles.active : null}`}>

            <h2 className={styles.main\_title\_caption}>

                Table: {tableId}

                <p className={styles.cross} onClick={() => handleActiveClick(false)}>&times;</p>

            </h2>

            {showPayment && <Payment booking={selectedBooking} handleActiveClick={setShowPayment} date={date}/>}

            <div className={styles.list}>

                {bookingList.map((booking) => {

                    return (

                        <div key={booking.booking\_id} className={styles.booking} onClick={(e) => handleListClick(booking, e)}>

                            <h4 className={styles.secondary}>Time: <p>{booking.time}</p></h4>

                            <h4 className={styles.secondary}>Period: <p>{booking.period}</p></h4>

                            <h4 className={styles.secondary}>Game name: <p>{booking.game\_name}</p></h4>

                            <h4 className={styles.secondary}>Amount of players: <p>{booking.players\_amount}</p></h4>

                            <Button

                                name={"button\_template2"}

                                width={100}

                                height={50}

                                action={() => handleDelete(booking.booking\_id)}

                            >

                                Delete

                            </Button>

                            <Button

                                name={"button\_template2"}

                                width={120}

                                height={50}

                                action={() => handlePayment(booking)}

                            >

                                Change Payment

                            </Button>

                        </div>

                    );

                })}

            </div>

        </form>

    );

}

admin/\_bookings\_list/BookingsList.module.scss

@import "../../globals.scss";

.main\_title\_caption{

    @extend %font1;

    font-size: 45px;

    letter-spacing: 5px;

    font-weight: 900;

    color: $color\_5;

    margin: 30px 0;

    font-weight: bolder;

    text-align: center;

    color: $color\_5;

    display: flex;

    flex-direction: row;

    justify-content: space-between;

    align-items: start;

    width: 90%;

    .cross{

        margin: 0;

        background-color: $color\_4;

        padding: 0px 8px 0px 15px;

        border-radius: 7px;

        color: $color\_1;

        cursor: pointer;

    }

}

.section7\_login{

    position: fixed;

    z-index: 3;

    width: 950px;

    flex-direction: column;

    margin-left: 0vw;

    padding-left: 60px;

    background-color: $color\_1;

    padding: 8px;

    border-radius: 15px;

    display: none;

    align-items: center;

    justify-self: center;

    align-self: center;

    box-shadow: 1rem 1.4rem 25rem rgb(0, 0, 0);

    z-index: 5;

    .list{

        display: flex;

        align-items: center;

        flex-direction: column-reverse;

        width: 100%;

        .booking{

            background-color: $color\_4;

            border-radius: 10px;

            display: flex;

            flex-direction: row;

            align-items: center;

            justify-content: space-between;

            @extend %font2;

            padding: 10px 15px;

            width: 86%;

            margin: 15px 0;

            .secondary{

                font-size: 12px;

                margin: 0 5px;

                p{

                    background-color: $color\_1;

                    margin: 0 5px;

                    padding: 5px 10px;

                    border-radius: 5px;

                }

            }

        }

    }

}

.section7\_login::after{

    content: "";

    top: 0;

    left: 0;

    position: absolute;

    width: 100%;

    height: 100%;

    opacity: 0;

    visibility: hidden;

    transition: 0.5s;

}

.player\_list{

    margin: 20px 0;

    background-color: $color\_1;

    width: 90%;

    display: flex;

    flex-direction: column;

    align-items: start;

    justify-content: center;

    border-radius: 10px;

    .member{

        margin-left: 25%;

        color: $color\_3;

    }

}

.section7\_consultation\_description{

    width: 90%;

    font-size: 16px;

    line-height: 22px;

    letter-spacing: 1.22px;

    color: $color\_5;

    margin: 0 0 35px 0;

    text-align: justify;

    @extend %font2;

    font-weight: 200;

}

.active{

    display: flex;

}

.secondary{

    display: flex;

    flex-direction: row;

    align-items: center;

    justify-content: start;

    height: 50px;

    font-size: 26px;

    p{

        margin-left: 20px;

    }

}

.third{

    display: flex;

    flex-direction: row;

    align-items: center;

    justify-content: start;

    font-size: 25px;

    height: 50px;

    p{

        margin-left: 30px;

    }

}

admin/\_add\_game/addGame.js

"use client"

import { useState } from "react";

import Button from "../../\_components/Button";

import styles from "./addGame.module.scss";

import { createGame } from "@/app/\_lib/gameSlice.js";

import { useSelector, useDispatch } from "react-redux";

export default function AddGame({user, active, setActive}) {

    const [gameName, setGameName] = useState("");

    const [players, setPlayers] = useState("");

    const [goal, setGoal] = useState("");

    const [rules, setRules] = useState("");

    const [inputError, setInputError] = useState("");

    const dispatch = useDispatch();

    const { error, isLoading } = useSelector((state) => state.game);

    function handleCreate(){

        setInputError("");

        if(!gameName){

            setInputError("Please enter game name");

            return;

        }

        if(!players){

            setInputError("Please please enter amount of players");

            return;

        }

        if(!goal){

            setInputError("Please fill the goal of the game");

            return;

        }

        if(!rules){

            setInputError("Please fill the rules");

            return;

        }

        dispatch(createGame(user.email, user.userId, gameName, players, goal, rules));

        setActive(false);

    }

    return (

        <>{active ?

            <div className={styles.authenticate}>

            <form className={styles.section7\_login}>

                <h2 className={styles.main\_title\_caption}>

                    Add new game

                    <p className={styles.cross} onClick={() => setActive(false)}>&times;</p>

                </h2>

                <p className={styles.section7\_consultation\_description}>Please enter all the inputs to submit this form</p>

                <input type="text" className={styles.name} required name="login" placeholder="Game name"  value={gameName} onChange={(e) => setGameName(e.target.value)} /> <br />

                <input type="text" className={styles.name} required placeholder="Players" value={players} onChange={(e) => setPlayers(e.target.value)}/> <br />

                <input type="text" className={styles.name} required placeholder="Goal" value={goal} onChange={(e) => setGoal(e.target.value)}/> <br />

                <input type="text" className={styles.name} required placeholder="Rules" value={rules} onChange={(e) => setRules(e.target.value)}/> <br />

                {(inputError || error) && <h2 className={styles.error}>{error ? error : inputError}</h2>}

                <Button

                    action={() => handleCreate()}

                    name={"button\_template2"}

                    width={250}>

                    <p>{isLoading ? "Loading..." : "Submit"}</p>

                </Button>

            </form>

        </div>

        : null}</>

    );

}

admin/\_add\_game/addGame.module.scss

@import "../../globals.scss";

.authenticate {

    position: fixed;

    top: 0;

    left: 0;

    width: 100vw;

    height: 100vh;

    display: flex;

    justify-content: center;

    align-items: center;

    z-index: 9;

}

.section7\_login{

    position: fixed;

    z-index: 5;

    width: 600px;

    height: auto;

    flex-direction: column;

    margin-left: 0px auto;

    padding-left: 60px;

    background-color: $color\_4;

    padding: 8px 8px 20px 8px;

    top: 30%;

    border-radius: 15px;

    //display: none;

    display: flex;

    align-items: center;

    justify-self: center;

    align-self: center;

    box-shadow: 2rem 1.4rem 200rem 7rem rgb(0, 0, 0);

}

.main\_title\_caption{

    @extend %font1;

    font-size: 45px;

    letter-spacing: 5px;

    font-weight: 900;

    color: $color\_5;

    margin: 30px 0;

    font-weight: bolder;

    text-align: center;

    color: $color\_5;

    display: flex;

    flex-direction: row;

    justify-content: space-between;

    align-items: start;

    width: 90%;

    .cross{

        margin: 0;

        background-color: $color\_4;

        padding: 0px 8px 0px 15px;

        border-radius: 7px;

        color: $color\_1;

        cursor: pointer;

    }

}

.section7\_consultation\_description{

    width: 540px;

    font-size: 19px;

    line-height: 22px;

    letter-spacing: 1.22px;

    color: $color\_5;

    margin: 0 0 7px 0;

    text-align: center;

    @extend %font2;

    font-weight: 200;

}

.name{

    padding-left: 16px;

    width: 470px;

    height: 50px;

    font-family: Poppins;

    font-weight: 600;

    font-size: 16px;

    line-height: 20px;

    border: 1px solid #bfbfbf;

    box-shadow: inset 0 0 0 55px #fff!important;

    border-radius: 15px;

    margin: 10px 0 0 0;

    outline: none;

}

.name:-webkit-autofill {

    -webkit-box-shadow: inset 0 0 0 55px #fff!important;

    box-shadow: inset 0 0 0 55px #fff!important;

}

.name :placeholder-shown{

    color: $color\_5;

    letter-spacing: 1px;

}

.error{

    background-color: red;

    color: $color\_1;

    @extend %font1;

    padding: 10px;

    width: 77.5%;

    border-radius: 7px;

    text-align: center;

}