# Cinema app

Darmowy hosting do przetestowania : https://render.com/docs/free

1. Ogólne informacje

“Cinema App” to aplikacja do zarządzania kinem i rezerwacjami biletów. Całość została podzielona na dwie główne części:

**frontend** (React + Material UI),

**backend** (Node.js + Express + SQLite).

2. Uruchomienie projektu (potrzebujemy node i npm do uruchomienia projektu)

Instalacja paczek (w folderze backend):

npm install express@5 cors dotenv sqlite3 bcrypt jsonwebtoken nodemailer express-session

npm install --save-dev nodemon jest

Foldery:

frontend/ – zawiera kod interfejsu w React.

backend/ – zawiera serwer Express, pliki konfiguracyjne i bazę danych SQLite.

Konfiguracja serwera:

Plik app.js (backend) – konfiguracja Express, ustawienie Middlewares, routing.

Plik server.js (backend) – startuje nasłuchiwanie na porcie (np. 3001) i wywołuje initDb() (patrz niżej).

Uruchomienie:

Backend: npm run dev (z nodemon) w folderze backend.

Frontend: npm start (React) w folderze frontend.

3. Struktura backendu

backend/models/index.js

initDb() – funkcja wywoływana przy starcie serwera w server.js. Tworzy połączenie z plikiem bazy cinema.db (SQLite) i definiuje tabele (jeśli jeszcze nie istnieją).

getDb() – umożliwia dostęp do instancji bazy w innych miejscach (np. w kontrolerach).

backend/models/userModel.js

createUser(...) – wstawianie nowego użytkownika do bazy (np. podczas rejestracji).

getUserByEmail(...) – szukanie użytkownika po adresie e-mail (przy logowaniu).

getUserById(...) – pobieranie użytkownika na podstawie ID (przy weryfikacji sesji/tokena).

updateUser(...) – aktualizacja profilu użytkownika.

backend/models/roomModel.js – CRUD dla sal kinowych (nr sali, liczba rzędów/kolumn, layout).

backend/models/seanceModel.js – CRUD dla seansów (filmId, roomId, data, godzina, ceny biletów).

backend/models/ticketModel.js – rezerwacje/kupno biletów (userId, seatNumber itp.).

3.2. Kontrolery i obsługa logiki

backend/controllers/authController.js

register(...) – rejestracja nowego konta, sprawdzenie czy email nie istnieje, hashowanie hasła, zapis w bazie, zwrócenie ID usera.

login(...) – logowanie; sprawdza hasło, generuje token JWT z userId i role.

backend/controllers/userController.js

getProfile(...) – zwraca dane zalogowanego użytkownika (np. GET /api/users/me).

updateProfile(...) – aktualizuje dane zalogowanego użytkownika (np. PUT /api/users/me).

backend/controllers/roomController.js – zarządzanie salami: dodawanie, pobieranie, edycja, usuwanie.

backend/controllers/seanceController.js

getAllSeancesWithFilm(...) – pobiera seanse wraz z danymi o filmie i sali (JOIN).

getSeanceById(...) – szczegóły pojedynczego seansu.

backend/controllers/ticketController.js – tworzenie rezerwacji, sprawdzanie dostępności miejsc.

3.3. Routingi (endpointy)

backend/routes/authRoutes.js

/api/auth/register – rejestracja

/api/auth/login – logowanie

backend/routes/userRoutes.js

/api/users/me (GET/PUT) – profil użytkownika.

backend/routes/roomRoutes.js – /api/rooms – CRUD dla sal.

backend/routes/seanceRoutes.js – /api/seances – pobieranie seansów, tworzenie, edycja, etc.

backend/routes/ticketRoutes.js – /api/tickets – kupno / rezerwacja biletów.

Dodatkowo:

Middleware verifyToken() i isAdmin() – chroni endpointy, które wymagają autoryzacji.

4. Struktura frontendu

Frontend jest zbudowany w React (z Material UI) i został podzielony na komponenty i widoki (pages):

frontend/src/App.js – główny plik z routingiem (React Router).

frontend/src/pages – widoki takie jak:

HomePage.js – strona główna z karuzelą filmów i przyciskiem “Kup bilet”.

LoginPage.js, RegisterPage.js – logowanie i rejestracja użytkowników.

RepertuarPage.js – wyświetlanie dostępnych filmów w formie kart z przyciskiem do kupna biletu.

FilmDetailsPage.js – szczegóły konkretnego filmu (opis, gatunek, czas trwania) i przycisk “Kup bilet”.

SeatSelectionPage.js – interaktywna mapa sali kinowej (wybór miejsc, POST /api/tickets).

Panel użytkownika (np. UserDashboardPage, UserProfilePage, UserReservationsPage) – przeglądanie rezerwacji, edycja profilu itp.

Panel admina (np. AdminDashboardPage, AdminFilmsPage, AdminRoomsPage, AdminSeancesPage, AdminSnacksPage, AdminUsersPage) – zarządzanie całym systemem (CRUD na filmach, seansach, salach, snackach, użytkownikach).

5. Najważniejsze funkcjonalności

Rejestracja i logowanie (z JWT).

Panel użytkownika – aktualizacja profilu, przegląd rezerwacji.

Panel administratora – dodawanie nowych filmów, seansów, sal kinowych i snacków.

Widok sal kinowych – interaktywny wybór miejsc, z różnymi strefami (front, VIP, standard), animacje przy zaznaczaniu.

Kupno biletów – rezerwacja miejsc + generowanie biletów w bazie (tickets).

6. Przykladowy opis

“Cinema App” jest przykładem aplikacji full-stack z podziałem na frontend (React) i backend (Express + SQLite). Dzięki użyciu JWT i odpowiednich tras (endpoints), umożliwia zalogowanym użytkownikom przeglądanie filmów, rezerwacje biletów i modyfikowanie profilu, a administratorowi – pełną kontrolę nad repertuarem i salami.

Cała architektura składa się z (techiniczne aspekty):

Warstwy danych (SQLite + modele w folderze models),

Warstwy logiki (kontrolery w folderze controllers),

Warstwy routingu (pliki w routes),

Frontendu w React, który komunikuje się z backendem poprzez **REST API.**