

Dokumentacja techniczna - system rezerwacji sal konferencyjnych

Jędrzej Ciechanowski, Kacper Leszczyński, Tomasz Świątek

15 stycznia 2025

Spis treści

1 Cele projektu	3
2 Opis stosu technologicznego i wykorzystywanych narzędzi	3
2.1 Backend	3
2.2 Frontend	3
2.3 Inne	3
3 Diagram ERD	4
4 Widoki i projekt graficzny	5
4.1 Ekrany logowania	5
4.2 Główne ekrany użytkownika	7
4.2.1 Ekran rezerwacji	7
4.2.2 Ekran szczegółowych danych rezerwacji	8
4.2.3 Ekran rezerwowania spotkania	9
4.2.4 Ekran dostępnych pokoi do rezerwacji	10
4.2.5 Ekran szczegółowego opisu pokoju	11
4.2.6 Ekran rezerwacji pokoju	12
4.2.7 Ekran wydarzeń	13
4.2.8 Ekran powiadomień	14
4.2.9 Ekran profil użytkownika	15
4.3 Główny ekran administratora	16
4.3.1 Ekran dodawania sali konferencyjnej	17
4.3.2 Ekran zarządzania pokojami	18
4.3.3 Ekran profil administratora	19
5 Dokumentacja punktów końcowych systemu	19
5.1 Endpointy związane z autoryzowaniem użytkownika	19
5.1.1 Login	19
5.1.2 Register	20
5.1.3 Refresh JWT Token	21
5.2 Endpointy przeznaczone dla użytkownika	21
5.2.1 Rooms	21
5.2.2 Room by ID	22
5.2.3 Available Rooms	22
5.2.4 Profile	23
5.2.5 Notifications	23
5.2.6 Update Notification Status	23
5.2.7 Reserve	24
5.2.8 Meetings	24
5.2.9 Reservations	25
5.2.10 Reservation by ID	25
5.2.11 Get Users	26
5.3 Endpointy przeznaczone dla admina	26
5.3.1 Add Room	26
5.3.2 Delete Room	27
5.3.3 Edit User Role	27
5.3.4 Edit User Password	28

5.3.5 Delete User	29
6 Wykorzystane rozwiązania produkcyjne	29

1 Cele projektu

Celem projektu było zaprojektowanie i stworzenie aplikacji mobilnej pozwalającej na rezerwacje sal konferencyjnych wewnętrz organizacji. Aplikacja ta usprawniłaby korzystanie z takich sal przez pracowników organizacji adaptującej zaprojektowaną przez nas aplikację. Aplikacja ma na celu umożliwienie użytkownikom przeglądanie dostępnych sal, sprawdzanie ich wyposażenia oraz rezerwowania ich na określony czas. Aplikacja przesyła powiadomienia o nadchodzących spotkaniach, zaplanowanych w aplikacji, dzień i godzinę przed ich rozpoczęciem.

2 Opis stosu technologicznego i wykorzystywanych narzędzi

2.1 Backend

Do stworzenia backendu aplikacji użyliśmy języka **Python** wraz z frameworkiem **Flask**. Flask to lekki framework umożliwiający szybsze tworzenie API, zarządzanie autoryzacją oraz zarządzanie danymi. W celu uproszczenia operacji na bazie danych użyliśmy biblioteki ORM flask-SQLAlchemy. W celu uproszczenia operacji związanych z generowaniem i autoryzacją za pomocą Tokenów JWT użyliśmy biblioteki Flask-JWT-Extended.

Systemem zarządzania bazą danych który wybraliśmy jest **PostgreSQL**, jeden z najpopularniejszych lub, wedle niektórych badań, najpopularniejszy taki system na świecie.

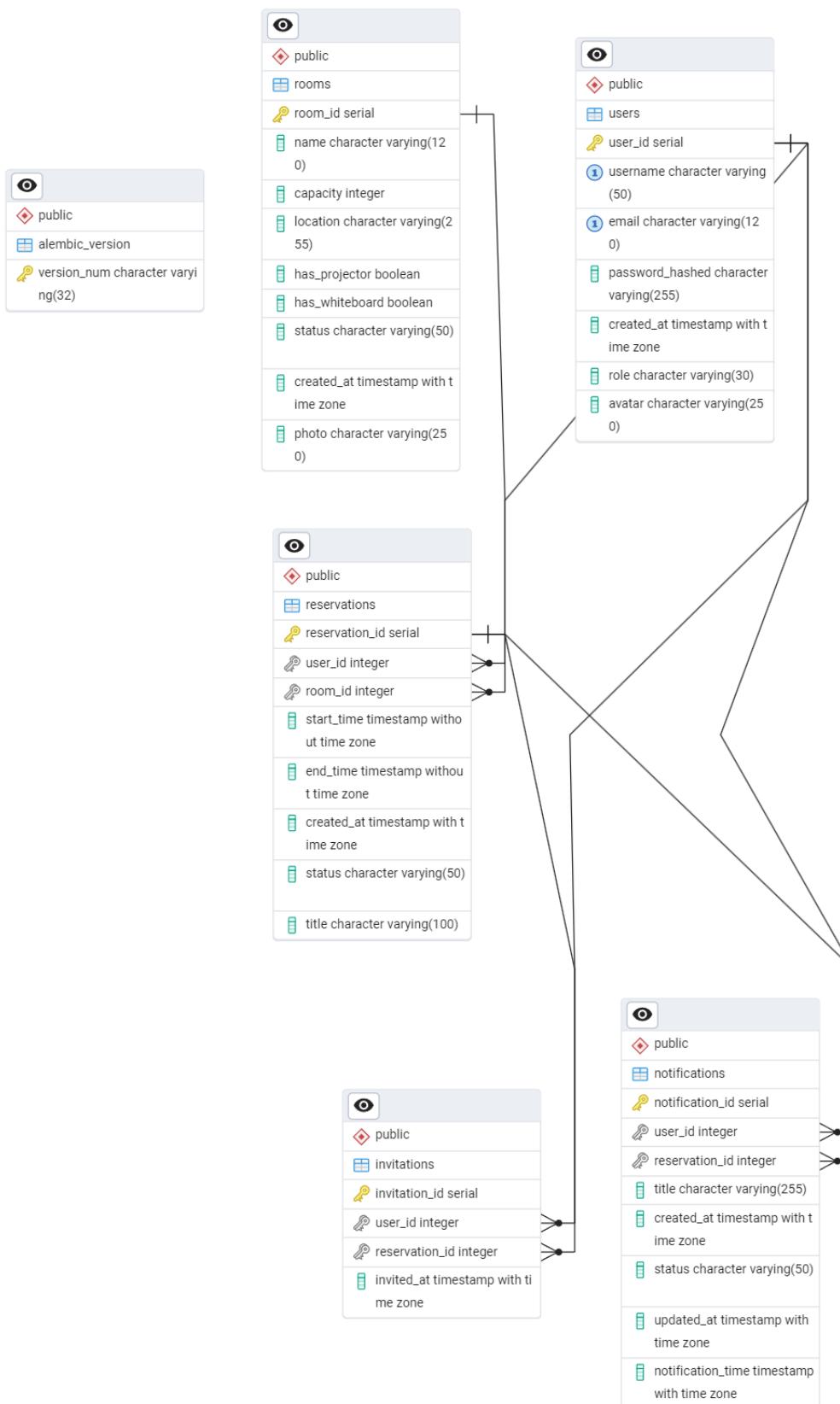
2.2 Frontend

Do stworzenia frontendu aplikacji wykorzystaliśmy framework **React Native**. React Native to nowoczesne rozwiązanie umożliwiające tworzenie aplikacji mobilnych na platformy iOS oraz Android z wykorzystaniem jednego wspólnego kodu.

2.3 Inne

Jako platformę do przechowywania naszego kodu wybraliśmy **Github** (link do repozytorium: <https://github.com/rikardoricz/conference-room-booking>), największą tego typu platformę opartą o system kontroli wersji Git. Tam też znajduje się instrukcja uruchomienia aplikacji - plik README.md. Do testowania api używaliśmy **Postmana**, oprogramowania dedykowanego właśnie do tego. Podczas programowania projektu korzystaliśmy z **Visual Studio Code**, jednego z najpopularniejszych IDE, oraz **NeoVim**. Do wygenerowania diagramu ERD wykorzystaliśmy funkcje wbudowaną w **pgAdmin4**, jest to popularne narzędzie wykorzystywane do zarządzania bazami danych stworzonymi przy pomocy PostgreSQL. Projekt wizualny całego projektu stworzyliśmy w **Figma**. Za pomocą **Docker** i Docker Compose zkonteneryzowaliśmy backend oraz bazę danych. Zbudowany obraz Docker został zapisany w rejestrze Docker: <https://hub.docker.com/r/rikardoricz/conf-room-booking>

3 Diagram ERD



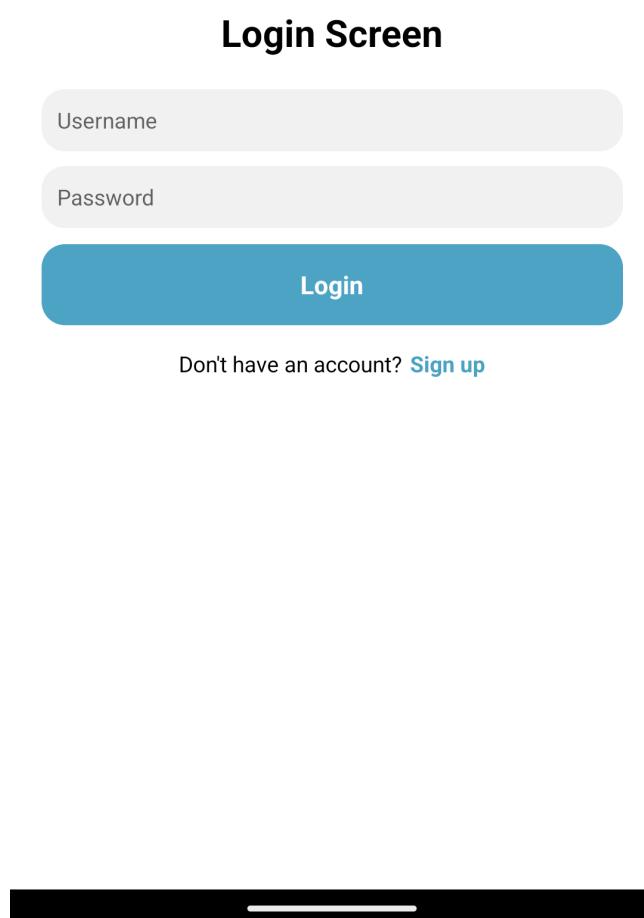
Rysunek 1: Schemat diagramu ERD

4 Widoki i projekt graficzny

4.1 Ekrany logowania

Ekran logowania umożliwia użytkownikowi wprowadzenie swojego loginu i hasła, a system sprawdza, czy użytkownik istnieje w bazie danych. W przypadku błędnych danych użytkownik otrzymuje komunikat o nieudanym logowaniu.

12:14 ⓘ 



Rysunek 2: Ekran logowania

Ekran rejestracji pozwala nowemu użytkownikowi stworzyć konto, sprawdzając, czy spełnia wymagania dotyczące adresu e-mail, hasła oraz jego potwierdzenia. System waliduje dane, zapewniając, że użytkownik jest unikalny. Po pomyślnym uzupełnieniu formularza użytkownik otrzymuje komunikat o udanej rejestracji i może przejść do ekranu logowania.

12:16 ⓘ 🔍 🔍 🔍

Signup Screen

The screenshot shows a mobile application's sign-up interface. At the top, there are four light-gray rounded rectangular input fields stacked vertically. The first field is labeled "Username", the second "Email", the third "Password", and the fourth "Confirm password". Below these fields is a large, horizontally stretched blue button with the white text "Sign up" in its center. At the bottom of the screen, centered, is the text "Already have an account? [Login](#)". The background of the entire screen is white. At the very bottom, there is a thick black horizontal bar, likely representing the device's physical home button or a decorative element.

Username

Email

Password

Confirm password

Sign up

Already have an account? [Login](#)

Rysunek 3: Ekran Rejestrowania

4.2 Główne ekranu użytkownika

4.2.1 Ekran rezerwacji

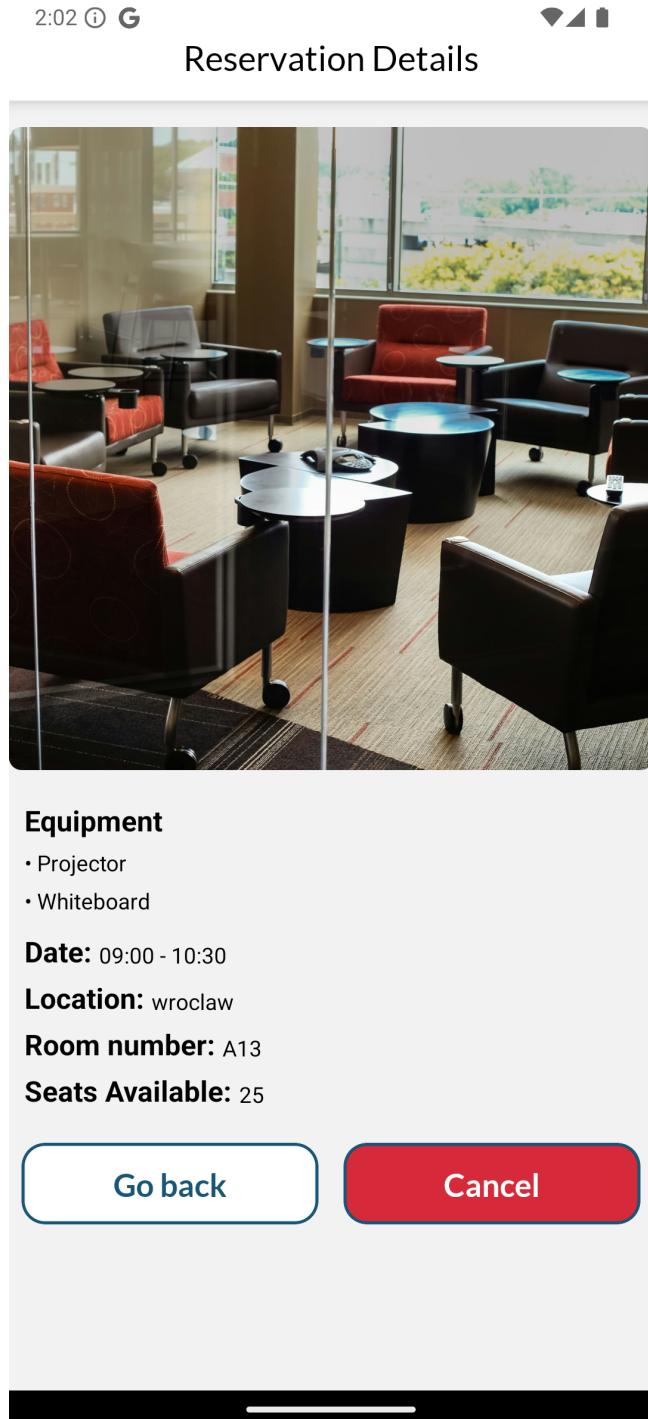
Ekran rezerwacji ma układ, w którym wyświetlana jest lista rezerwacji utworzonych przez użytkownika. Każda rezerwacja jest przedstawiona w formie karty, zawierającej kluczowe informacje: nazwa rezerwacji, lokalizację, czas oraz liste dostępnego wyposażenia sali. Rezerwacje są pogrupowane w sposób umożliwiający łatwe przeglądanie. W prawym dolnym rogu ekranu znajduje się przycisk "Dodaj rezerwację", który umożliwia użytkownikowi utworzenie nowej rezerwacji.



Rysunek 4: Ekran Rezerwacji

4.2.2 Ekran szczegółowych danych rezerwacji

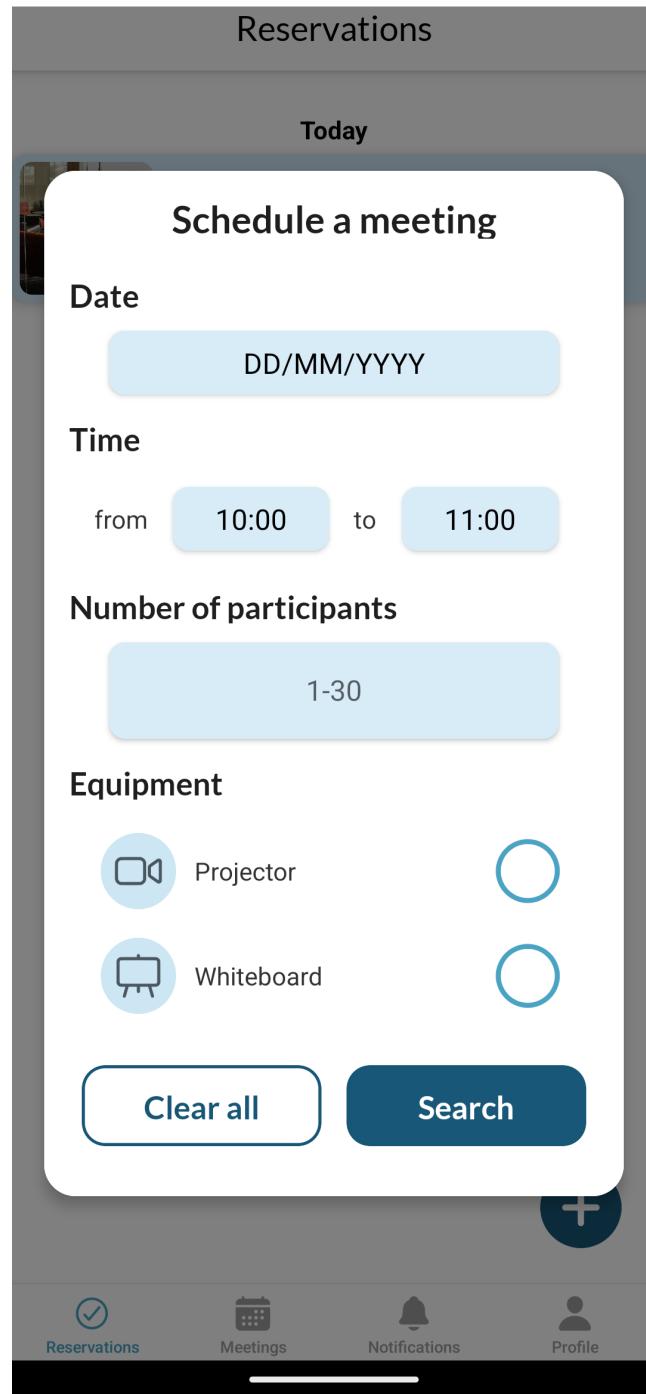
Z ekranu Rysunek: 4.2.7 możemy przejść do ekranu szczegółowych danych rezerwacji klikając na na wybrane spotkanie. Na ekranie tym można znaleźć więcej informacji na temat spotkania. Dodatkowo użytkownik ma możliwość anulowania rezerwacji, jeśli zdecyduje, że nie chce dłużej z niej korzystać. Po kliknięciu przycisku cancel dostajemy komunikat czy napewno chcemy anulować daną rezerwację. Jeśli zdecydujemy, że tak rezerwacja zostaje anulowana i zostaje ona usunięta z bazy danych.



Rysunek 5: Ekran Rezerwacji

4.2.3 Ekran rezerwowania spotkania

Z ekranu rezerwacji Rysunek: 4.2.7 możemy przejść do ekranu rezerwacji spotkań, klikając na znak "+" znajdujący się w prawym dolnym rogu. Po jego naciśnięciu otwiera się ekran, na którym należy wybrać interesującą nas datę, godzinę oraz liczbę uczestników spotkania. Po dobrani odpowiednich parametrów możemy przejść do ekranu dostępnych pokoi Rysunek 7.



Rysunek 6: Ekran rezerwowania spotkania

4.2.4 Ekran dostępnych pokoi do rezerwacji

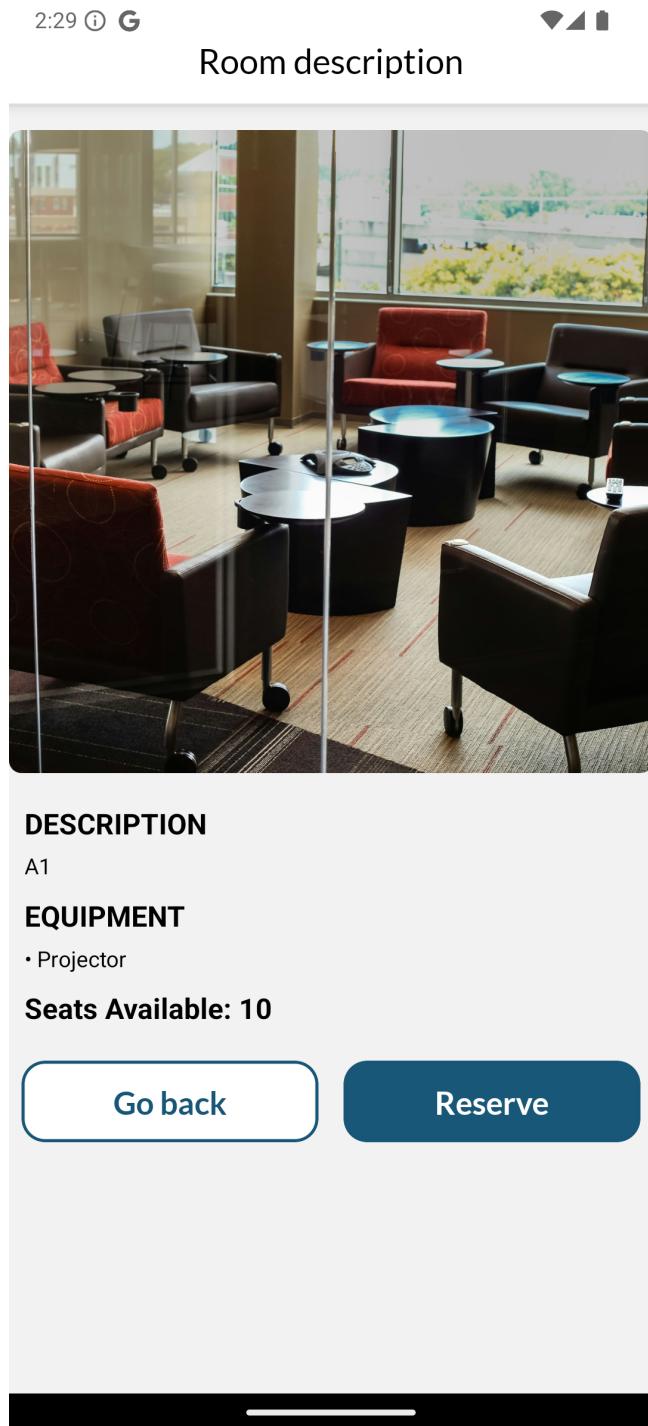
Na tym ekranie użytkownik może wybrać spośród dostępnych pokoi (w tym przypadku dostępny jest tylko jeden). Po dokonaniu wyboru pokoju, użytkownik zostaje przeniesiony na ekran z opisem pokoju Rysunek 8



Rysunek 7: Ekran dostępnych pokoi do rezerwacji

4.2.5 Ekran szczegółowego opisu pokoju

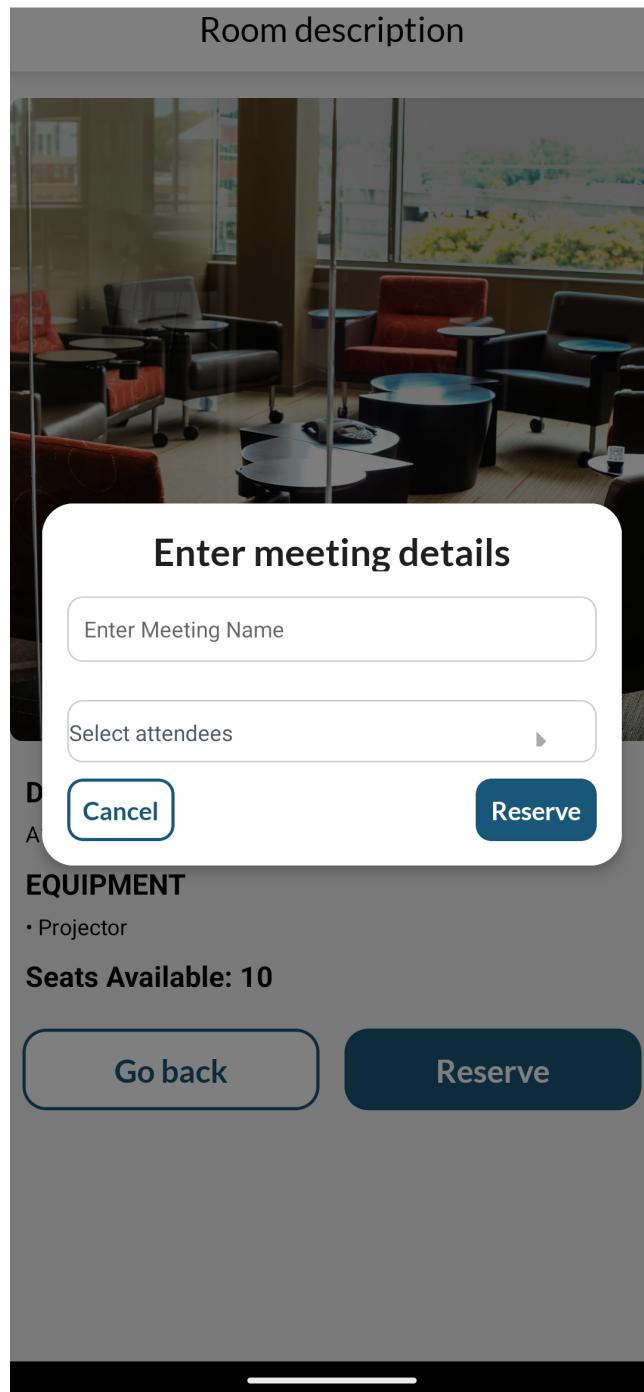
Na tym ekranie, po zapoznaniu się z opisem pokoju, użytkownik może zdecydować, czy chce wrócić do poprzedniego ekranu Rysunek: 7, czy kontynuować i przejść do rezerwacji Rysunek: 9.



Rysunek 8: Ekran szczegółowego opisu pokoju

4.2.6 Ekran rezerwacji pokoju

Na tym ekranie użytkownik ustawia nazwę spotkania, wybiera uczestników i rezerwuje spotkanie.



Rysunek 9: Ekran rezerwacji pokoju

4.2.7 Ekran wydarzeń

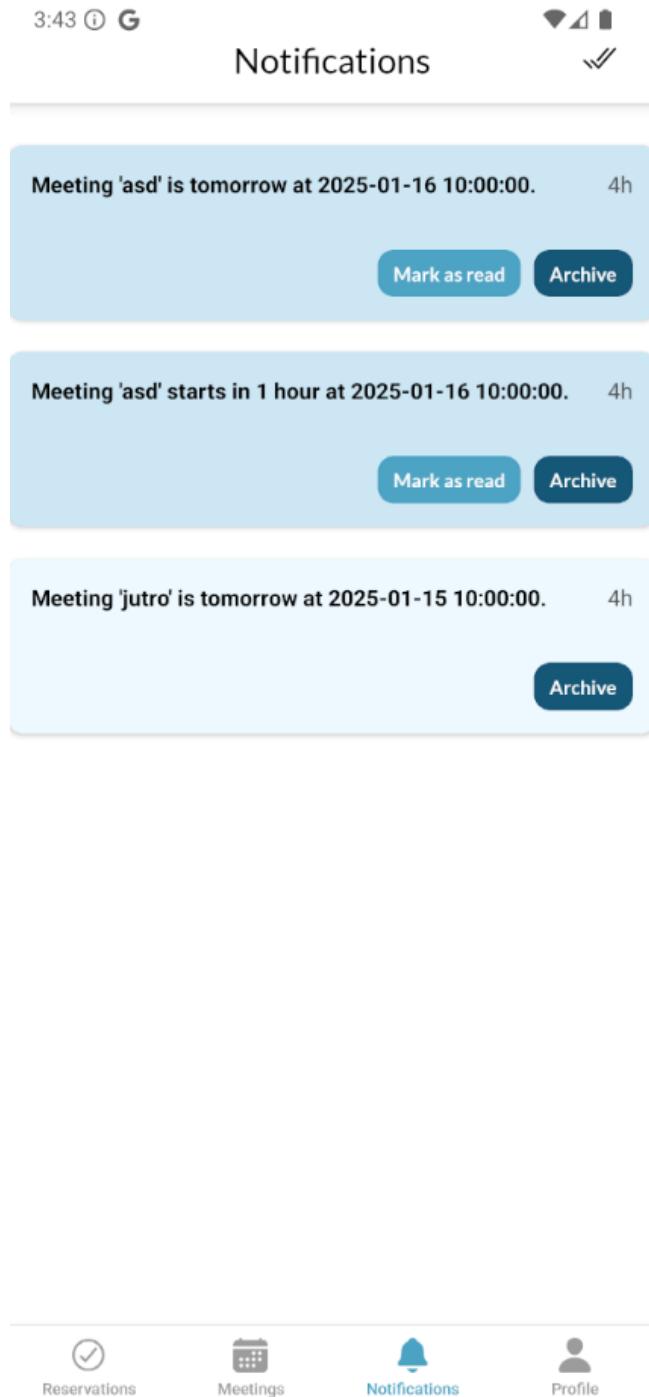
Ekran wydarzeń wyświetla listę wszystkich spotkań, w których bierzesz udział. W głównej części ekranu znajdują się karty spotkań, z podstawowymi informacjami, takimi jak: lokalizacja, czas oraz nazwa spotkania. Jeśli któryś ze spotkań jest organizowane przez ciebie Podobnie jak w przypadku ekranu rezerwacji Rysunek: , w prawym dolnym rogu znajduje się przycisk umożliwiający tworzenie nowych spotkań.



Rysunek 10: Ekran spotkań

4.2.8 Ekran powiadomień

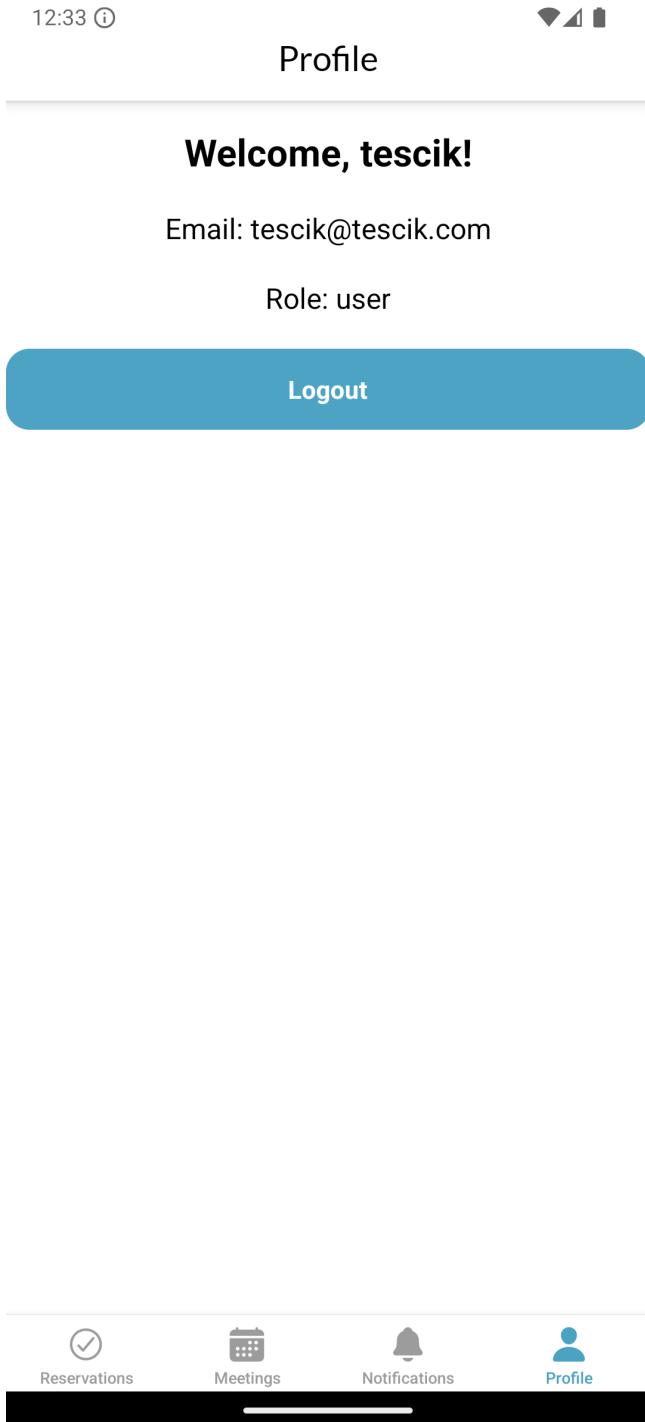
Ekran powiadomień wyświetla użytkownikowi powiadomienia o nadchodzących spotkaniach. Użytkownik otrzymuje powiadomienie dzień przed spotkaniem oraz godzinę przed jego rozpoczęciem. Powiadomienia można oznaczyć jako przeczytane lub wybrać opcję 'Archive', po której powiadomienie znika z ekranu, ale zostaje zapisane w bazie danych.



Rysunek 11: Ekran powiadomień

4.2.9 Ekran profil użytkownika

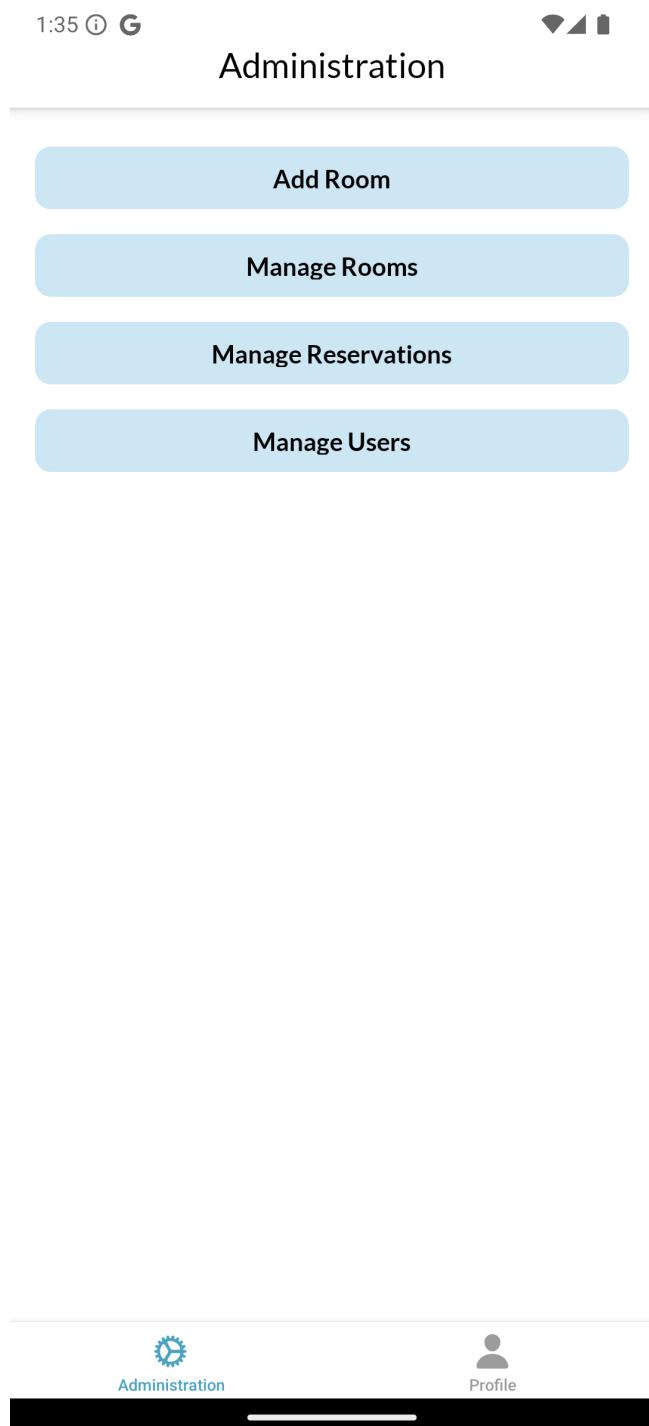
Ekran profilu użytkownika pozwala na wylogowanie się z aplikacji. Pokazuje również aktualny adres e-mail oraz rolę pełnioną w systemie, np. użytkownik (user) lub administrator (admin).



Rysunek 12: Ekran profil użytkownika

4.3 Główny ekran administratora

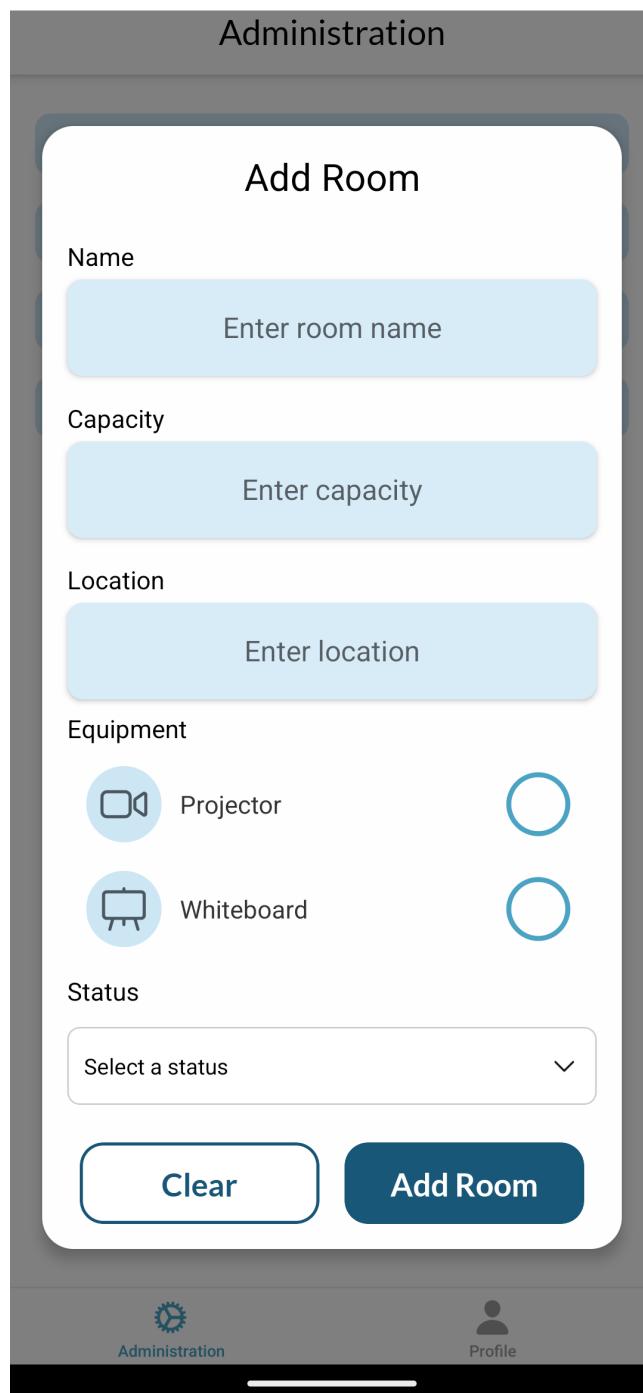
Panel administratora to sekcja aplikacji, która umożliwia zarządzanie różnymi aspektami systemu. Administrator ma dostęp do kilku kluczowych opcji, w tym dodawania nowych sal, zarządzania istniejącymi salami, resetowania haseł użytkowników oraz zarządzania kontami użytkowników. Administrator może dodać nowe sale do systemu, wprowadzając szczegóły takie jak m.in. nazwa sali, lokalizacja, pojemność i dostępne wyposażenie. Kolejna opcja dla administratora to Manager Rooms umożliwiająca usuwanie istniejących sal w systemie. Kolejne pole to reset password, opcja pozwala administratorowi zresetować hasło dowolnego użytkownika w systemie. Ostatnie pole to manage users gdzie administrator może usuwać konta.



Rysunek 13: Ekran zarządzania administratora

4.3.1 Ekran dodawania sali konferencyjnej

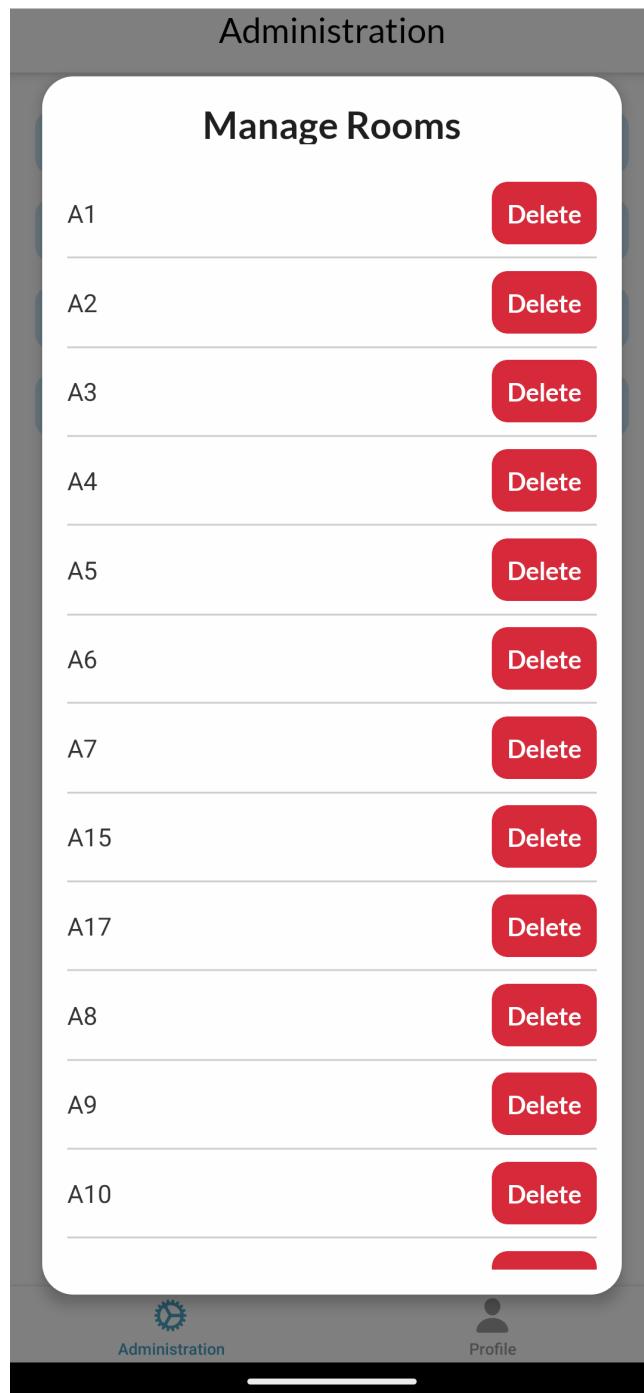
Ekran dodawania sali konferencyjnych umożliwia administratorowi dodanie nowej sali do systemu. Składa się z kilku pól formularza, w tym: Nazwa, Pojemność, Lokalizacja. Poniżej administrator może wybrać również dostępne wyposażenie projektor i/lub tablicę. Na dole ekranu znajduje się pole Status, w którym administrator wybiera status sali: dostępna lub w konserwacji. Na dole formularza znajdują się dwa przyciski Wyczyść do usunięcia wprowadzonych danych oraz przycisk Dodaj salę który zatwierdza formularz i dodaje nową salę do systemu.



Rysunek 14: Ekran dodawania sali konferencyjnych

4.3.2 Ekran zarządzania pokojami

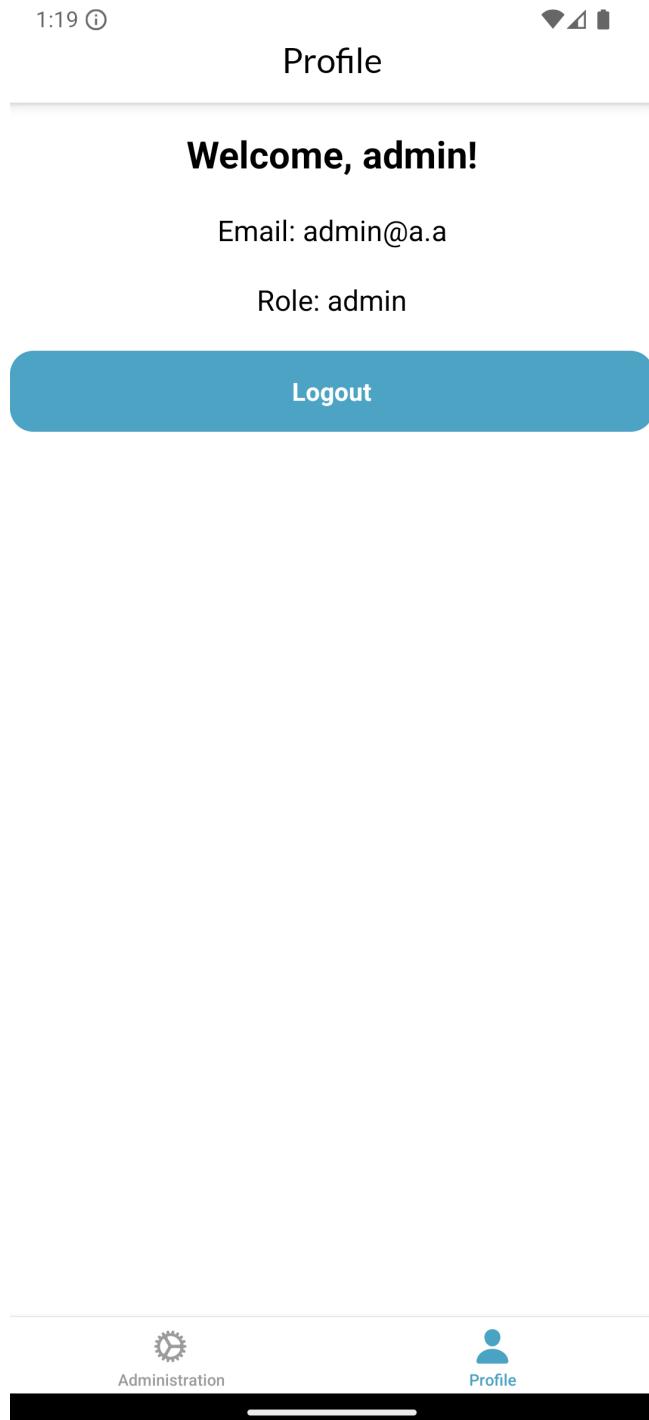
Ekran zarządzania pokojami umożliwia administratorowi usuwanie sal z systemu. Obecnie dostępna jest tylko opcja usunięcia pokoju, jednak w przyszłości planowane jest dodanie funkcji edytowania istniejących pokoi.



Rysunek 15: Ekran zarządzania pokojami

4.3.3 Ekran profil administratora

Ekran profilu administratora pozwala na wylogowanie się z aplikacji. Pokazuje również aktualny adres e-mail oraz rolę pełnioną w systemie, np. użytkownik (user) lub administrator (admin).



Rysunek 16: Ekran profil administratora

5 Dokumentacja punktów końcowych systemu

5.1 Endpointy związane z autoryzowaniem użytkownika

5.1.1 Login

- URL: /login
- Metody: POST, GET

- **Opis:** Endpoint umożliwiający logowanie użytkownika przy użyciu nazwy użytkownika i hasła.

Struktura żądania:

Body:

```
{
  "username": "ondre",
  "password": "password"
}
```

Przykłady odpowiedzi:

- **200 OK**

```
{
  "message": "Login Successful, <User: ondre>",
  "access_token": "access_token",
  "refresh_token": "refresh_token",
  "user_id": 1,
  "role": "user"
}
```

- **400 Bad Request:** Brakujące dane w żądaniu.

```
{
  "msg": "Missing fields: username, password"
}
```

- **401 Unauthorized:** Niepoprawna nazwa użytkownika lub hasło.

```
{
  "msg": "Login Unsuccessful. Please check username and password."
}
```

5.1.2 Register

- **URL:** /register
- **Metody:** POST, GET
- **Opis:** Endpoint umożliwiający rejestrację nowego użytkownika.

Struktura żądania:

Body:

```
{
  "username": "ondre",
  "email": "ondre@example.com",
  "password": "password"
}
```

Przykłady odpowiedzi:

- **200 OK**

```
{
  "msg": "Register Successful, <User: ondre>",
  "access_token": "access_token",
  "refresh_token": "refresh_token"
}
```

- **400 Bad Request:** Brakujące dane w żądaniu.

```
{  
    "msg": "Missing fields: username, email, password"  
}
```

- **409 Conflict: Email już istnieje.**

```
{  
    "msg": "Email already used"  
}
```

- **409 Conflict: Uzytkownik już istnieje.**

```
{  
    "msg": "Username already used"  
}
```

5.1.3 Refresh JWT Token

- **URL:** /refresh
- **Metody:** POST
- **Opis:** Endpoint do odświeżania tokenu dostępu przy użyciu tokenu odświeżającego.
- **Wymagania:** Token odświeżający musi być dołączony w nagłówkach.

Struktura żądania:

Nagłówki:

Authorization: Bearer <refresh_token>

Przykłady odpowiedzi:

- **200 OK**

```
{  
    "access_token": "access_token",  
    "user_id": 1,  
    "role": "user"  
}
```

- **422 Unprocessable entity: Niewłaściwy token JWT**

```
{  
    "msg": "Only refresh tokens are allowed"  
}
```

5.2 Endpointy przeznaczone dla użytkownika

5.2.1 Rooms

- **URL:** /rooms
- **Metody:** GET
- **Opis:** Endpoint do pobierania listy wszystkich dostępnych pokoi.

Przykłady odpowiedzi:

- 200 OK

```
\begin{verbatim}
[
  {
    "id": 1,
    "name": "Room 1",
    "capacity": 10,
    "location": "Building A",
    "has_projector": true,
    "has_whiteboard": false,
    "status": "available"
  },
  ...
]
```

5.2.2 Room by ID

- **URL:** /rooms/{id}
- **Metody:** GET
- **Opis:** Endpoint do pobierania informacji o pojedynczym pokoju o danym identyfikatorze.

Przykłady odpowiedzi:

- 200 OK

```
{
  "id": 1,
  "name": "Room 1",
  "capacity": 10,
  "location": "Building A",
  "has_projector": true,
  "has_whiteboard": false,
  "status": "available",
  "created_at": "Fri, 03 Jan 2025 17:31:18 GMT",
  "reservations": [
    {
      "reservation_id": 1,
      "reservation_start": "Mon, 13 Jan 2025 23:00:00 GMT",
      "reservation_end": "Tue, 14 Jan 2025 00:00:00 GMT"
    }
  ]
}
```

5.2.3 Available Rooms

- **URL:** /rooms/available
- **Metody:** GET
- **Opis:** Endpoint do pobierania listy dostępnych pokoi w zadanym przedziale czasowym.

Struktura zapytania:

Query Parameters:

startTime: "2025-01-13T10:00:00",
endTime: "2025-01-13T12:00:00"

Przykłady odpowiedzi:

- 200 OK

```
[  
 {  
   "id": 1,  
   "name": "Room 1",  
   "capacity": 10,  
   "location": "Building A",  
   "has_projector": true,  
   "has_whiteboard": false,  
   "status": "available"  
,  
 ...  
 ]
```

5.2.4 Profile

- URL: /profile
- Metody: GET
- Opis: Endpoint do pobierania informacji o aktualnym użytkowniku.

Przykłady odpowiedzi:

- 200 OK

```
{  
   "username": "ondre",  
   "email": "ondre@example.com",  
   "role": "user",  
   "user_id": 1,  
   "created_at": "Fri, 13 Dec 2024 19:47:14 GMT"  
}
```

5.2.5 Notifications

- URL: /notifications
- Metody: GET
- Opis: Endpoint do pobierania listy powiadomień dla aktualnego użytkownika.

Przykłady odpowiedzi:

```
[  
 {  
   "notification_id": 1,  
   "user_id": 1,  
   "reservation_id": 1,  
   "title": "Meeting 'MEETING 2' starts in 1 hour at 2025-01-13 22:27:00.",  
   "created_at": "2025-01-13T22:21:52.935448+01:00",  
   "updated_at": "2025-01-13T22:21:50.757810+01:00",  
   "status": "unread"  
,  
 ...  
 ]
```

5.2.6 Update Notification Status

- URL: /notifications/{notification_id}
- Metody: PATCH
- Opis: Endpoint do aktualizacji statusu powiadomienia.

Struktura żądania:

Body:

```
{  
    "status": "read"  
}
```

Przykłady odpowiedzi:

- 200 OK

```
{  
    "msg": "Notification status updated successfully",  
    "notification_id": 1,  
    "updated_at": "2025-01-13T23:52:24.649556+01:00",  
    "status": "read"  
}
```

5.2.7 Reserve

- URL: /reserve
- Metody: POST
- Opis: Endpoint do tworzenia rezerwacji pokoju.

Struktura żądania:

Body:

```
{  
    "start_time": "2025-01-14 00:10:00",  
    "end_time": "2025-01-14 00:29:00",  
    "room_id": 1,  
    "title": "TITLE",  
    "attendees": [2, 3]  
}
```

Przykłady odpowiedzi:

- 200 OK

```
{  
    "msg": "Reservation Successful, reservation id: 1. Invitations sent to 2 attendees."  
}
```

5.2.8 Meetings

- URL: /meetings
- Metody: GET
- Opis: Endpoint do pobierania listy spotkań (zarówno rezerwacji użytkownika, jak i spotkań, do których użytkownik został zaproszony).

Przykłady odpowiedzi:

- 200 OK

```
[  
    {  
        "reservation_id": 1,  
        "user_id": 1,  
        "room_id": 1,  
        "start_time": "Mon, 13 Jan 2025 23:00:00 GMT",  
        "end_time": "Tue, 14 Jan 2025 00:00:00 GMT",  
    }
```

```

        "created_at": "Mon, 13 Jan 2025 21:17:38 GMT",
        "title": "TITLE",
        "link_to_photo": "../assets/room1.jpg"
    },
    ...
]

```

5.2.9 Reservations

- **URL:** /reservations
- **Metody:** GET
- **Opis:** Endpoint do pobierania listy rezerwacji aktualnego użytkownika.

Przykłady odpowiedzi:

- **200 OK**

```

[
{
    "reservation_id": 1,
    "user_id": 1,
    "room_id": 1,
    "start_time": "Mon, 13 Jan 2025 23:00:00 GMT",
    "end_time": "Tue, 14 Jan 2025 00:00:00 GMT",
    "created_at": "Mon, 13 Jan 2025 21:17:38 GMT",
    "title": "Meeting",
    "has_projector": true,
    "has_whiteboard": false,
    "link_to_photo": "../assets/room1.jpg"
},
...
]

```

5.2.10 Reservation by ID

- **URL:** /reservations/{id}
- **Metody:** GET, DELETE
- **Opis:** Endpoint do pobierania szczegółów rezerwacji lub jej usunięcia.

Przykłady odpowiedzi (GET):

- **200 OK**

```
{
    "reservation_id": 1,
    "user_id": 1,
    "room_id": 1,
    "start_time": "Mon, 13 Jan 2025 23:00:00 GMT",
    "end_time": "Tue, 14 Jan 2025 00:00:00 GMT",
    "created_at": "Mon, 13 Jan 2025 21:20:25 GMT",
    "title": "TITLE",
    "link_to_photo": "../assets/room1.jpg"
}
```

Przykłady odpowiedzi (DELETE):

- **200 OK**

```
{
    "msg": "Reservation with ID 1 has been deleted."
}
```

5.2.11 Get Users

- **URL:** /users
- **Metody:** GET
- **Opis:** Endpoint do pobierania listy użytkowników o roli "user".

Przykłady odpowiedzi:

- **200 OK**

```
[  
  {  
    "user_id": 1,  
    "username": "ondre"  
  },  
  ...  
]
```

5.3 Endpointy przeznaczone dla admina

5.3.1 Add Room

- **URL:** /add-room
- **Metody:** POST
- **Opis:** Endpoint umożliwiający dodanie nowego pokoju do systemu. Wymaga roli administratora.

Struktura żądania:

Body:

```
{  
  "name": "Pojek1",  
  "capacity": 10,  
  "location": "Building 1",  
  "has_projector": true,  
  "has_whiteboard": true,  
  "status": "available"  
}
```

Przykłady odpowiedzi:

- **200 OK**

```
{  
  "msg": "Successfully added room, <Room id: 1>"  
}
```

- **400 Bad Request:** Brakujące dane w żądaniu.

```
{  
  "msg": "Missing json in request"  
}
```

- **409 Conflict:** Błąd autoryzacji.

```
{  
  "msg": "Authorization Error! Invalid role"  
}
```

5.3.2 Delete Room

- **URL:** /delete-room
- **Metody:** DELETE
- **Opis:** Endpoint umożliwiający usunięcie pokoju z systemu. Wymaga roli administratora.

Struktura żądania:

Body:

```
{  
    "room_id": 1  
}
```

Przykłady odpowiedzi:

- **200 OK**

```
{  
    "msg": "Reservation with ID 1 has been deleted."  
}
```

- **400 Bad Request:** Brakujące dane w żądaniu.

```
{  
    "msg": "No room-id provided"  
}
```

- **404 Not Found:** Pokój nie został znaleziony.

```
{  
    "msg": "Room not found"  
}
```

- **409 Conflict:** Błąd autoryzacji.

```
{  
    "msg": "Authorization Error! Invalid role"  
}
```

5.3.3 Edit User Role

- **URL:** /edit-role
- **Metody:** POST
- **Opis:** Endpoint umożliwiający zmianę roli użytkownika w systemie. Wymaga roli admina.

Struktura żądania:

Body:

```
{  
    "username": "ondre",  
    "role": "admin"  
}
```

Przykłady odpowiedzi:

- 200 OK

```
{  
    "msg": "User role updated successfully!"  
}
```

- 400 Bad Request: Brakujące dane w żądaniu.

```
{  
    "msg": "Missing json in request"  
}
```

- 403 Forbidden: Błąd autoryzacji.

```
{  
    "msg": "Authorization Error! Invalid role"  
}
```

- 404 Not Found: Użytkownik nie został znaleziony.

```
{  
    "msg": "User not found"  
}
```

5.3.4 Edit User Password

- URL: /edit-user-password
- Metody: POST
- Opis: Endpoint umożliwiający zmianę hasła użytkownika. Wymaga roli admina.

Struktura żądania:

Body:

```
{  
    "username": "ondre",  
    "password": "new_password"  
}
```

Przykłady odpowiedzi:

- 200 OK

```
{  
    "msg": "User password updated successfully!"  
}
```

- 400 Bad Request: Brakujące dane w żądaniu.

```
{  
    "msg": "Missing json in request"  
}
```

- 403 Forbidden: Błąd autoryzacji.

```
{  
    "msg": "Authorization Error! Invalid role"  
}
```

- **404 Not Found:** Użytkownik nie został znaleziony.

```
{  
    "msg": "User not found"  
}
```

5.3.5 Delete User

- **URL:** /delete-user
- **Metody:** DELETE
- **Opis:** Endpoint umożliwiający usunięcie użytkownika z systemu. Wymaga roli admina.

Struktura żądania:

Body:

```
{  
    "username": "ondre"  
}
```

Przykłady odpowiedzi:

- **200 OK**

```
{  
    "msg": "User deleted successfully!"  
}
```

- **400 Bad Request:** Brakujące dane w żądaniu.

```
{  
    "msg": "Missing json in request"  
}
```

- **403 Forbidden:** Błąd autoryzacji.

```
{  
    "msg": "Authorization Error! Invalid role"  
}
```

- **404 Not Found:** Użytkownik nie został znaleziony.

```
{  
    "msg": "User not found"  
}
```

6 Wykorzystane rozwiązania produkcyjne

- Skonteneryzowaliśmy backend i bazę danych projektu używając do tego Docker'a oraz Docker Compose.
- Wrzucenie skonteneryzowanego backendu i bazy danych na fizyczny serwer domowy (dostęp przez VPN).
- Wrażliwe zmienne są przechowywane w pliku .env.
- Autoryzacja za pomocą tokenu JWT.