account_manager

---

account_manager

Modules

- (screens)/(auth)/login/page
- (screens)/(auth)/register/page
- (screens)/(base)/chat/[chatType]/[id]/page
- (screens)/(base)/group-settings/[groupId]/page
- (screens)/(base)/layout
- (screens)/(base)/page
- (screens)/(base)/profile/[id]/page
- (screens)/(base)/settings/page
- (screens)/layout
- colors
- components
- components/Aside
- components/Button
- components/Divider
- components/Header
- components/Icon
- components/Message
- components/ProfileImage
- components/Section
- components/TextInput
- components/UserCard
- config
- layout
- socket

account_manager

ChatApp

Project: Web Application for User Communication

- The application allows user database management through a login system. Users can only edit or delete their own accounts. They can send private messages to each other and create group chats. The project emphasizes simplicity, security, and an intuitive interface.
  ### Key Features:

- **User Database Management**: Secure login/authentication system.
- **Account Control**: Users can only modify or delete their own accounts.
- **Communication Tools**: Private messaging and group chat functionality.
- **Design Principles**: Focus on simplicity, security, and user-friendly interface. ## Instalation and run ### Frontend: Instalation:

```
cd website
npm i
```

Run for development `npm run dev`

Backend

Requirements

- Python 3.9+
- SQLite

Installation

```
cd backend\utils
pip install -r requirements.txt
```

Run Server

```
flask run --port=5000 --debug
```

account_manager

---

account_manager / colors

colors

Variables

- default

account_manager

---

account_manager / colors / default

Variable: default

`const` **default**: `object`

Defined in: colors.tsx:5

Obiekt colors zawierający stałe kolory używane w aplikacji

Type declaration

border

border: `string` = `"#7B7B7B"`

on_surface_gray

on_surface_gray: `string` = `"#565656"`

on_surface_light_gray

on_surface_light_gray: `string` = `"#A0A0A0"`

on_surface_white

on_surface_white: `string` = `"#EAE5E5"`

primary

primary: `string` = `"#680B8A"`

red

red: `string` = `"#FF0000"`

account_manager

---

account_manager / components

components

Functions

- Header
- Icon
- Message
- ProfileImage
- Section
- TextInput

- UserCard

References

Aside

Renames and re-exports default

Button

Renames and re-exports default

Divider

Renames and re-exports default

account_manager

---

account_manager / components/Aside

components/Aside

Functions

- default
- FriendsGroupButton
- UserCard

account_manager

---

account_manager / components/Aside / default

Function: default()

> **default**(`props`): `Element`

Defined in: components/Aside.tsx:21

Komponent Aside wyświetlający listę użytkowników lub grup oraz przycisk do tworzenia grup

Parameters

props

obiekt właściwości komponentu

groups `any`

lista grup do wyświetlenia

users `any`

lista użytkowników do wyświetlenia

Returns

`Element`

- komponent

account_manager

---

account_manager / components/Aside / FriendsGroupButton

Function: FriendsGroupButton()

> **FriendsGroupButton**(`props`): `Element`

Defined in: components/Aside.tsx:101

Przycisk wyboru pomiędzy listą przyjaciół a listą grup

Parameters

props

obiekt właściwości komponentu

callback `Function`

funkcja zmieniająca widok (przyjaciele/grupy)

checked `boolean`

czy przycisk jest aktualnie wybrany

label `string`

etykieta przycisku ("Friends" lub "Groups")

Returns

`Element`

- komponent

account__manager

---

account__manager / components/Aside / UserCard

Function: UserCard()

> **UserCard**(`props`): `Element`

Defined in: components/Aside.tsx:78

Komponent karty użytkownika

Parameters

props

obiekt właściwości komponentu

onClick? () => `void`

funkcja wywoływana po kliknięciu na kartę (opcjonalna)

user `any`

obiekt reprezentujący użytkownika lub grupę

Returns

`Element`

- komponent

account__manager

---

account__manager / components/Button

components/Button

Functions

- default

account_manager

---

account_manager / components/Button / default

Function: default()

> **default**(`props`): `Element`

Defined in: components/Button.tsx:12

Komponent Button wyświetlający przycisk z trzema wariantami stylów

Parameters

props

obiekt właściwości komponentu

label `string`

tekst na przycisku

onClick () => `void`

funkcja wywoływana po kliknięciu przycisku

type? `"filled"` | `"outlined"` | `"outlined2"` = `"filled"`

typ przycisku: “filled” | “outlined” | “outlined2” (domyślnie “filled”)

Returns

`Element`

- komponent

account_manager

---

account_manager / components/Divider

components/Divider

Functions

- default

account_manager

account_manager / components/Divider / default

Function: default()

> **default**(`props`): `Element`

Defined in: components/Divider.tsx:12

Komponent Divider wyświetlający poziomy lub pionowy pasek podziału

Parameters

props

obiekt właściwości komponentu

color

`string`

kolor tła paska

horizontal?

`boolean`

jeśli true, pasek jest poziomy

vertical?

`boolean`

jeśli true, pasek jest pionowy

Returns

`Element`

- komponent

account_manager

account_manager / components / Header

Function: Header()

> **Header**(`props`): `Element`

Defined in: components/Header.tsx:21

Komponent Header wyświetlający nagłówek strony

Parameters

props

obiekt właściwości komponentu

backArrow? `boolean`

czy wyświetlać strzałkę wstecz (opcjonalne)

header `any`

zawartość nagłówka

userProfileSrc? `string`

adres URL zdjęcia profilowego użytkownika (opcjonalne)

Returns

`Element`

- komponent

account_manager

---

account_manager / components / Icon

Function: Icon()

> **Icon**(`props`): `Element`

Defined in: components/Icon.tsx:12

Komponent Icon wyświetlający ikonę

Parameters

props

obiekt właściwości komponentu

onClick () => `void`

funkcja wywoływana po kliknięciu na ikonę

size `number`

rozmiar ikony

src? `string`

źródło obrazka (jeśli pusty, wyświetlany jest obraz domyślny)

Returns

`Element`

- komponent

account_manager

---

account_manager / components / Message

Function: Message()

> **Message**(`props`): `Element`

Defined in: components/Message.tsx:15

Komponent Message wyświetlający pojedynczą wiadomość

Parameters

props

obiekt właściwości komponentu

children `any`

zawartość wiadomości

isFirst? `Boolean`

czy jest to pierwsza wiadomość w grupie wiadomości (opcjonalne)

isLast? `Boolean`

czy jest to ostatnia wiadomość w grupie wiadomości (opcjonalne)

isUserAuthor? `Boolean`

czy autorem wiadomości jest użytkownik (domyślnie false)

name? `string`

imię autora wiadomości (opcjonalne)

src? `string`

adres URL zdjęcia profilowego autora wiadomości

Returns

`Element`

- komponent

account_manager

---

account_manager / components / ProfileImage

Function: ProfileImage()

> **ProfileImage**(`props`): `Element`

Defined in: components/ProfileImage.tsx:10

Komponent ProfileImage wyświetlający okrągłe zdjęcie profilowe z oznaczeniem aktywności

Parameters

props

obiekt właściwości komponentu

isActive? `Boolean`

czy użytkownik jest aktywny (opcjonalne)

size `number`

rozmiar obrazka (szerokość i wysokość w px)

src `string`

adres URL zdjęcia profilowego

Returns

`Element`

- komponent

account_manager

---

account_manager / components / Section

Function: Section()

> **Section**(`props`): `Element`

Defined in: components/Section.tsx:10

Komponent Section wyświetlający sekcję z nagłówkiem i zawartością

Parameters

props

obiekt właściwości komponentu

children

`any`

zawartość sekcji

header

`String`

nazwa sekcji

onClick?

() => `void`

funkcja wywoływana po kliknięciu na sekcję (opcjonalne)

Returns

`Element`

- komponent

account_manager

---

account_manager / components / TextInput

Function: TextInput()

> **TextInput**(`props`): `Element`

Defined in: components/TextInput.tsx:15

Komponent Input wyświetlający pole do wprowadzania textu z etykietą

Parameters

props

obiekt właściwości komponentu

label `string`

etykieta pola

long? `boolean`

czy użyć pola textarea zamiast input (opcjonalne)

password? `boolean`

czy pole jest typu hasło (opcjonalne)

placeholder? `string`

tekst zastępczy w polu (opcjonalny)

setValue `Function`

funkcja ustawiająca nową wartość pola

value `string`

aktualna wartość pola

Returns

`Element`

- komponent

account_manager

---

account_manager / components / UserCard

Function: UserCard()

> **UserCard**(`props`): `Element`

Defined in: components/UserCard.tsx:18

Komponent UserCard wyświetlający kartę użytkownika

Parameters

props

obiekt właściwości komponentu

buttons? `any`

opcjonalne przyciski do wyświetlenia obok nazwy (opcjonalne)

children? `any`

kontent karty (opcjonalne)

className? `string`

klasy (opcjonalne)

isActive `Boolean`

czy użytkownik jest aktywny

name `String`

imię użytkownika

onClick? () => `void`

funkcja wywoływana po kliknięciu na element z imieniem i nazwiskiem (opcjonalne)

onClickCard? () => `void`

funkcja wywoływana po kliknięciu na całą kartę (opcjonalne)

style? `any`

dodatkowe stylowanie (opcjonalne)

surname `String`

nazwisko użytkownika

Returns

`Element`

- komponent

account_manager

---

account_manager / components/Header

components/Header

References

default

Renames and re-exports Header

account_manager

---

account_manager / components/Icon

components/Icon

References

default

Renames and re-exports Icon

account_manager

---

account_manager / components/Message

components/Message

References

default

Renames and re-exports Message

account_manager

---

account_manager / components/ProfileImage

components/ProfileImage

References

default

Renames and re-exports ProfileImage

account_manager

---

account_manager / components/Section

components/Section

References

default

Renames and re-exports Section

account_manager

---

account_manager / components/TextInput

components/TextInput

References

default

Renames and re-exports TextInput

account_manager

---

account_manager / components/UserCard

components/UserCard

References

default

Renames and re-exports UserCard

account_manager

---

account_manager / config

config

Variables

- default

account_manager

---

account_manager / config / default

Variable: default

> `const` **default**: `object`

Defined in: config.tsx:4

Konfiguracja aplikacji zawierająca adres API

Type declaration

api

> **api**: `string` = `"http://localhost:5000"`

account_manager

---

account_manager / layout

layout

Variables

- metadata

Functions

- default

account_manager

---

account_manager / layout / default

Function: default()

> **default**(`props`): `Element`

Defined in: layout.tsx:21

Główny layout

Parameters

props

`Readonly`<{ `children`: `ReactNode`; }>

obiekt właściwości komponentu

Returns

`Element`

Layout

account_manager

---

account_manager / layout / metadata

Variable: metadata

> `const` **metadata**: `Metadata`

Defined in: layout.tsx:12

account_manager

---

account_manager / (screens)/(auth)/login/page

(screens)/(auth)/login/page

Functions

- default

account_manager

---

account_manager / (screens)/(auth)/login/page / default

Function: default()

default(): `Element`

Defined in: (screens)/(auth)/login/page.tsx:19

Strona logowania. Zarządza formularzem logowania użytkownika.

Returns

`Element`

Strona logowania.

account_manager

---

account_manager / (screens)/(auth)/register/page

(screens)/(auth)/register/page

Functions

- default

account_manager

---

account_manager / (screens)/(auth)/register/page / default

Function: default()

default(): `Element`

Defined in: (screens)/(auth)/register/page.tsx:19

Strona rejestracji użytkownika. Zarządza formularzem rejestracji oraz przesyłaniem danych do API.

Returns

`Element`

Strona rejestracji.

account_manager

---

account_manager / (screens)/(base)/chat/[chatType]/[id]/page

(screens)/(base)/chat/[chatType]/[id]/page

Functions

- default
- LabeledContent

account_manager

---

account_manager / (screens)/(base)/chat/[chatType]/[id]/page / default

Function: default()

> **default**(`props`): `Element`

Defined in: (screens)/(base)/chat/[chatType]/[id]/page.tsx:17

Strona chatu Umożliwia chatowanie prywatne oraz grupowe

Parameters

props

`any`

Właściwości komponentu (nieużywane)

Returns

`Element`

Strona czatu

account_manager

---

account_manager / (screens)/(base)/chat/[chatType]/[id]/page / LabeledContent

Function: LabeledContent()

> **LabeledContent**(`props`): `Element`

Defined in: (screens)/(base)/chat/[chatType]/[id]/page.tsx:196

Parameters

props

content `string`

Zawartość tekstowa

label `string`

Tekst etykiety

long? `boolean`

Flaga do wyświetlania dłuższego tekstu

Returns

`Element`

Komponent

account_manager

---

account_manager / (screens)/(base)/group-settings/[groupId]/page

(screens)/(base)/group-settings/[groupId]/page

Functions

- default
- LabeledContent
- LineInput
- ParticipantCard

account_manager

---

account_manager / (screens)/(base)/group-settings/[groupId]/page / default

Function: default()

> **default**(`props`): `Element`

Defined in: (screens)/(base)/group-settings/[groupId]/page.tsx:22

Strona ustawień grupy GroupSettings Umożliwia zarządzanie ustawieniami grupy, w tym: - wyświetlanie i edycję nazwy i opisu grupy, - dodawanie i usuwanie członków, - zmiana ról członków, - filtrowanie członków według ról (OWNER, ADMIN, USER).

Parameters

props

Właściwości komponentu

params `Promise`<{ `groupId`: `string`; }>

Promise zawierający ID grupy

Returns

`Element`

JSX element strony ustawień grupy

account_manager

---

account_manager / (screens)/(base)/group-settings/[groupId]/page / LabeledContent

Function: LabeledContent()

> **LabeledContent**(`props`): `Element`

Defined in: (screens)/(base)/group-settings/[groupId]/page.tsx:268

Parameters

props

children? `any`

Dzieci komponentu

content? `string`

Opcjonalna zawartość tekstowa

label `string`

Tekst etykiety

long? `boolean`

Flaga do wyświetlania dłuższego tekstu (większa czcionka)

Returns

`Element`

Komponent

account_manager

---

account_manager / (screens)/(base)/group-settings/[groupId]/page / LineInput

Function: LineInput()

> **LineInput**(`props`): `Element`

Defined in: (screens)/(base)/group-settings/[groupId]/page.tsx:283

Parameters

props

placeholder `string`

Tekst zastępczy

setValue `Function`

Funkcja aktualizująca wartość inputa

value `string`

Wartość inputa

Returns

`Element`

Komponent

account_manager

---

account_manager / (screens)/(base)/group-settings/[groupId]/page / ParticipantCard

Function: ParticipantCard()

> **ParticipantCard**(`props`): `Element`

Defined in: (screens)/(base)/group-settings/[groupId]/page.tsx:236

Parameters

props

Propsy komponentu

id `string`

ID użytkownika

isActive `boolean`

Status aktywności użytkownika

name `string`

Imię użytkownika

removeMember (`id`) => `void`

Funkcja do usuwania członka z grupy

role `string`

Rola użytkownika (OWNER, ADMIN, USER)

setRole (`id`, `role`) => `void`

Funkcja do zmiany roli użytkownika

surname `string`

Nazwisko użytkownika

userRole `string`

Rola aktualnego użytkownika

Returns

`Element`

account_manager

---

account_manager / (screens)/(base)/layout

(screens)/(base)/layout

Functions

- default

account_manager

---

account_manager / (screens)/(base)/layout / default

Function: default()

> **default**(`props`): `null` | `Element`

Defined in: (screens)/(base)/layout.tsx:18

Layout strony

Parameters

props

Właściwości komponentu

children

`any`

Elementy potomne renderowane wewnątrz layoutu

Returns

`null` | `Element`

Layout

account_manager

---

account_manager / (screens)/(base)/page

(screens)/(base)/page

Functions

- default

account_manager

---

account_manager / (screens)/(base)/page / default

Function: default()

> **default**(): `Element`

Defined in: (screens)/(base)/page.tsx:16

Strona główna Wyświetla dashboard użytkownika z listą znajomych, grup i zaproszeń do znajomych.

Returns

`Element`

Strona główna

account_manager

---

account_manager / (screens)/(base)/profile/[id]/page

(screens)/(base)/profile/[id]/page

Functions

- default
- Input
- LineInput

account_manager

---

account_manager / (screens)/(base)/profile/[id]/page / default

Function: default()

> **default**(`props`): `Element`

Defined in: (screens)/(base)/profile/[id]/page.tsx:18

Strona profilu użytkownika. Umożliwia przeglądanie profili urzytkowników

Parameters

props

Właściwości komponentu

params { `id`: `string`; }

Parametry trasy zawierające ID użytkownika

params.id `string`

Returns

`Element`

Strona profilu użytkownika

account_manager

---

account_manager / (screens)/(base)/profile/[id]/page / Input

Function: Input()

> **Input**(`props`): `Element`

Defined in: (screens)/(base)/profile/[id]/page.tsx:103

Komponent Input wyświetlający etykietę oraz pole tekstowe lub tekst tylko do odczytu.

Parameters

props

Właściwości komponentu

editable? `boolean`

Flaga czy pole jest edytowalne

label `string`

Etykieta pola

long? `boolean`

Flaga czy pole ma być długie (textarea) czy jednolinijkowe (input)

placeholder `string`

Tekst zastępczy pola

setValue `Function`

Funkcja aktualizująca wartość

value `string`

Wartość pola

Returns

`Element`

JSX z etykietą oraz odpowiednim polem (textarea/input lub span)

account_manager

---

account_manager / (screens)/(base)/profile/[id]/page / LineInput

Function: LineInput()

> **LineInput**(`props`): `Element`

Defined in: (screens)/(base)/profile/[id]/page.tsx:85

Komponent wyświetlający jednoliniowy tekst (tylko do odczytu).

Parameters

props

Właściwości komponentu

placeholder `string`

Tekst zastępczy wyświetlany, gdy wartość jest pusta

setValue `Function`

Funkcja ustawiająca wartość (nieużywana, bo komponent jest tylko do odczytu)

value `string`

Wyświetlana wartość tekstowa

Returns

`Element`

JSX z tekstem wyświetlanym w divie

account_manager

---

account_manager / (screens)/(base)/settings/page

(screens)/(base)/settings/page

Functions

- default

account_manager

---

account_manager / (screens)/(base)/settings/page / default

Function: default()

> **default**(): `Element`

Defined in: (screens)/(base)/settings/page.tsx:16

Strona ustawień profilu użytkownika Umożliwia edycję danych osobowych i społecznościowych użytkownika

Returns

`Element`

Strona ustawień

account_manager

---

account_manager / (screens)/layout

(screens)/layout

Functions

- default

account_manager

---

account_manager / (screens)/layout / default

Function: default()

> **default**(`props`): `Element`

Defined in: (screens)/layout.tsx:11

Layout z providerem socket.io

Parameters

props

obiekt właściwości komponentu

children `ReactNode`

elementy potomne renderowane wewnątrz providera

Returns

`Element`

Layout

account_manager

---

account_manager / socket

socket

Functions

- SocketProvider
- useSocket

account_manager

---

account_manager / socket / SocketProvider

Function: SocketProvider()

> **SocketProvider**(`children`): `Element`

Defined in: socket.tsx:21

Provider zarządzający połączeniem socket.io i udostępniający je komponentom potomnym

Parameters

children

elementy potomne, które mogą korzystać z kontekstu socket

children `ReactNode`

Returns

`Element`

account_manager

---

account_manager / socket / useSocket

Function: useSocket()

> **useSocket**(): `any`

Defined in: socket.tsx:67

Hook zwracający kontekst socket.io

Returns

`any`

Obiekt z właściwościami socket oraz isConnected

Throws

Błąd jeśli hook jest używany poza kontekstem

My Project

Generated by Doxygen 1.14.0

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

Namespace Documentation

5.1 app Namespace Reference

Namespaces

- namespace config
- namespace models
- namespace sockets

Functions

- create_app ()

Variables

- db = SQLAlchemy()
- jwt = JWTManager()
- socketio = SocketIO(cors_allowed_origins="*", async_mode="threading")

5.1.1 Function Documentation

5.1.1.1 create_app()

```
app.create_app ()
```

```
Funkcja tworz ca  aplikacj  Flask oraz inicjalizuje wszystkie niezb dne
rozszerzenia i blueprinty. Zwraca gotow  aplikacj  Flask.

Returns:
    app (Flask): gotowa aplikacja Flask
```

5.1.2 Variable Documentation

5.1.2.1 db

```
app.db = SQLAlchemy()
```

5.1.2.2 jwt

```
app.jwt = JWTManager()
```

5.1.2.3 socketio

```
app.socketio = SocketIO(cors_allowed_origins="*", async_mode="threading")
```

5.2 app.config Namespace Reference

Classes

- class Config

5.3 app.models Namespace Reference

Classes

- class Friends
- class Messages
- class MessageType
- class Role
- class Room
- class Room_Users
- class RoomType
- class UserPhoto
- class Users

5.4 app.sockets Namespace Reference

Functions

- register_socket_handlers (socketio)

5.4.1 Function Documentation

5.4.1.1 register_socket_handlers()

```
app.sockets.register_socket_handlers (
              socketio)
```

Chapter 6

Class Documentation

6.1 app.config.Config Class Reference

Static Public Attributes

- JWT_SECRET_KEY = os.getenv('JWT_SECRET_KEY', 'super-secret-key')
- SQLALCHEMY_DATABASE_URI = os.getenv('DATABASE_URL', 'sqlite:///database.db')
- bool SQLALCHEMY_TRACK_MODIFICATIONS = False
- JWT_ACCESS_TOKEN_EXPIRES = os.getenv('JWT_ACCESS_TOKEN_EXPIRES', timedelta(days=2))

6.1.1 Member Data Documentation

6.1.1.1 JWT_ACCESS_TOKEN_EXPIRES

```
app.config.Config.JWT_ACCESS_TOKEN_EXPIRES = os.getenv('JWT_ACCESS_TOKEN_EXPIRES', timedelta(days=2))
[static]
```

6.1.1.2 JWT_SECRET_KEY

```
app.config.Config.JWT_SECRET_KEY = os.getenv('JWT_SECRET_KEY', 'super-secret-key')  [static]
```

6.1.1.3 SQLALCHEMY_DATABASE_URI

```
app.config.Config.SQLALCHEMY_DATABASE_URI = os.getenv('DATABASE_URL', 'sqlite:///database.db')
[static]
```

6.1.1.4 SQLALCHEMY_TRACK_MODIFICATIONS

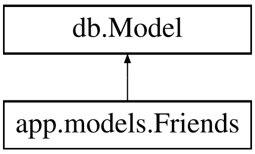
```
bool app.config.Config.SQLALCHEMY_TRACK_MODIFICATIONS = False  [static]
```

The documentation for this class was generated from the following file:

- config.py

6.2 app.models.Friends Class Reference

Inheritance diagram for app.models.Friends:



Static Public Attributes

- user_id = db.Column(db.Integer, db.ForeignKey('users.id'), primary_key=True)
- friend_id = db.Column(db.Integer, db.ForeignKey('users.id'), primary_key=True)
- created_at = db.Column(db.DateTime, default=datetime.utcnow)
- status = db.Column(db.Enum('pending', 'accepted', 'declined'), default='pending')

6.2.1 Member Data Documentation

6.2.1.1 created_at

```
app.models.Friends.created_at = db.Column(db.DateTime, default=datetime.utcnow)  [static]
```

6.2.1.2 friend_id

```
app.models.Friends.friend_id = db.Column(db.Integer, db.ForeignKey('users.id'), primary_↩
key=True)  [static]
```

6.2.1.3 status

```
app.models.Friends.status = db.Column(db.Enum('pending', 'accepted', 'declined'), default='pending')
[static]
```

6.2.1.4 user_id

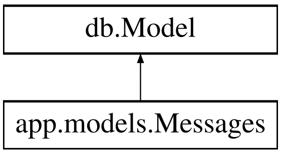
```
app.models.Friends.user_id = db.Column(db.Integer, db.ForeignKey('users.id'), primary_key=True)
[static]
```

The documentation for this class was generated from the following file:

- models.py

6.3 app.models.Messages Class Reference

Inheritance diagram for app.models.Messages:



Static Public Attributes

- id = db.Column(db.Integer, primary_key=True, autoincrement=True)
- user_id = db.Column(db.Integer, db.ForeignKey('users.id', ondelete='SET NULL'))
- room_id = db.Column(db.Integer, db.ForeignKey('room.id', ondelete='CASCADE'))
- content = db.Column(db.Text, nullable=False)
- message_type = db.Column(db.Enum(MessageType), default=MessageType.TEXT, nullable=False)
- timestamp = db.Column(db.DateTime, default=datetime.utcnow, index=True)
- image_path = db.Column(db.String(255))

6.3.1 Member Data Documentation

6.3.1.1 content

```
app.models.Messages.content = db.Column(db.Text, nullable=False)  [static]
```

6.3.1.2 id

```
app.models.Messages.id = db.Column(db.Integer, primary_key=True, autoincrement=True)  [static]
```

6.3.1.3 image_path

```
app.models.Messages.image_path = db.Column(db.String(255))  [static]
```

6.3.1.4 message_type

```
app.models.Messages.message_type = db.Column(db.Enum(MessageType), default=MessageType.TEXT,
nullable=False)  [static]
```

6.3.1.5 room_id

```
app.models.Messages.room_id = db.Column(db.Integer, db.ForeignKey('room.id', ondelete='CASCADE'))
[static]
```

6.3.1.6 timestamp

```
app.models.Messages.timestamp = db.Column(db.DateTime, default=datetime.utcnow, index=True)
[static]
```

6.3.1.7 user_id

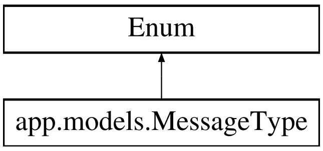
```
app.models.Messages.user_id = db.Column(db.Integer, db.ForeignKey('users.id', ondelete='SET
NULL')) [static]
```

The documentation for this class was generated from the following file:

- models.py

6.4 app.models.MessageType Class Reference

Inheritance diagram for app.models.MessageType:



Static Public Attributes

- str TEXT = 'TEXT'
- str IMAGE = 'IMAGE'
- str VIDEO = 'VIDEO'
- str FILE = 'FILE'

6.4.1 Member Data Documentation

6.4.1.1 FILE

```
str app.models.MessageType.FILE = 'FILE' [static]
```

6.4.1.2 IMAGE

```
str app.models.MessageType.IMAGE = 'IMAGE' [static]
```

6.4.1.3 TEXT

```
str app.models.MessageType.TEXT = 'TEXT' [static]
```

6.4.1.4 VIDEO

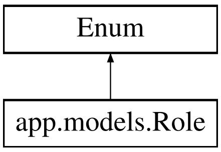
```
str app.models.MessageType.VIDEO = 'VIDEO'  [static]
```

The documentation for this class was generated from the following file:

- models.py

6.5 app.models.Role Class Reference

Inheritance diagram for app.models.Role:



Static Public Attributes

- str USER = 'USER'
- str ADMIN = 'ADMIN'
- str OWNER = 'OWNER'

6.5.1 Member Data Documentation

6.5.1.1 ADMIN

```
str app.models.Role.ADMIN = 'ADMIN'  [static]
```

6.5.1.2 OWNER

```
str app.models.Role.OWNER = 'OWNER'  [static]
```

6.5.1.3 USER

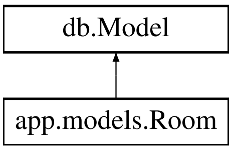
```
str app.models.Role.USER = 'USER'  [static]
```

The documentation for this class was generated from the following file:

- models.py

6.6 app.models.Room Class Reference

Inheritance diagram for app.models.Room:



Static Public Attributes

- id = db.Column(db.Integer, primary_key=True, autoincrement=True)
- type = db.Column(db.Enum(RoomType), nullable=False)
- name = db.Column(db.String(100), nullable=True)
- description = db.Column(db.Text, nullable=True)
- created_at = db.Column(db.DateTime, default=datetime.utcnow)
- messages = db.relationship('Messages', backref='room', lazy=True, cascade='all, delete-orphan')
- members = db.relationship('Room_Users', backref='room', lazy=True, cascade='all, delete-orphan')

6.6.1 Member Data Documentation

6.6.1.1 created_at

```
app.models.Room.created_at = db.Column(db.DateTime, default=datetime.utcnow) [static]
```

6.6.1.2 description

```
app.models.Room.description = db.Column(db.Text, nullable=True) [static]
```

6.6.1.3 id

```
app.models.Room.id = db.Column(db.Integer, primary_key=True, autoincrement=True) [static]
```

6.6.1.4 members

```
app.models.Room.members = db.relationship('Room_Users', backref='room', lazy=True, cascade='all,
delete-orphan') [static]
```

6.6.1.5 messages

```
app.models.Room.messages = db.relationship('Messages', backref='room', lazy=True, cascade='all,
delete-orphan') [static]
```

6.6.1.6 name

```
app.models.Room.name = db.Column(db.String(100), nullable=True)  [static]
```

6.6.1.7 type

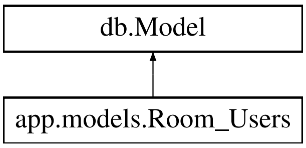
```
app.models.Room.type = db.Column(db.Enum(RoomType), nullable=False)  [static]
```

The documentation for this class was generated from the following file:

- models.py

6.7 app.models.Room_Users Class Reference

Inheritance diagram for app.models.Room_Users:



Static Public Attributes

- room_id = db.Column(db.Integer, db.ForeignKey('room.id', ondelete='CASCADE'), primary_key=True)
- user_id = db.Column(db.Integer, db.ForeignKey('users.id', ondelete='CASCADE'), primary_key=True)
- role = db.Column(db.Enum(Role), default=Role.USER, nullable=False)
- joined_at = db.Column(db.DateTime, default=datetime.utcnow)

6.7.1 Member Data Documentation

6.7.1.1 joined_at

```
app.models.Room_Users.joined_at = db.Column(db.DateTime, default=datetime.utcnow)  [static]
```

6.7.1.2 role

```
app.models.Room_Users.role = db.Column(db.Enum(Role), default=Role.USER, nullable=False)  [static]
```

6.7.1.3 room_id

```
app.models.Room_Users.room_id = db.Column(db.Integer, db.ForeignKey('room.id', ondelete='CASCADE'),
primary_key=True)  [static]
```

6.7.1.4 user_id

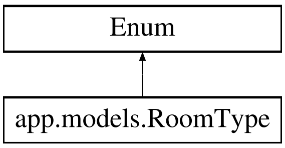
```
app.models.Room_Users.user_id = db.Column(db.Integer, db.ForeignKey('users.id', ondelete='CASCADE'),
primary_key=True) [static]
```

The documentation for this class was generated from the following file:

- models.py

6.8 app.models.RoomType Class Reference

Inheritance diagram for app.models.RoomType:



Static Public Attributes

- str PRIVATE = 'PRIVATE'
- str GROUP = 'GROUP'

6.8.1 Member Data Documentation

6.8.1.1 GROUP

```
str app.models.RoomType.GROUP = 'GROUP' [static]
```

6.8.1.2 PRIVATE

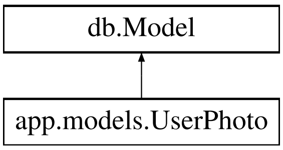
```
str app.models.RoomType.PRIVATE = 'PRIVATE' [static]
```

The documentation for this class was generated from the following file:

- models.py

6.9 app.models.UserPhoto Class Reference

Inheritance diagram for app.models.UserPhoto:



Static Public Attributes

- id = db.Column(db.Integer, primary_key=True)
- user_id = db.Column(db.Integer, db.ForeignKey('users.id'), nullable=False)
- photo_path = db.Column(db.String(255), nullable=False)
- is_profile = db.Column(db.Boolean, default=False)
- created_at = db.Column(db.DateTime, default=datetime.utcnow)

6.9.1 Member Data Documentation

6.9.1.1 created_at

```
app.models.UserPhoto.created_at = db.Column(db.DateTime, default=datetime.utcnow)  [static]
```

6.9.1.2 id

```
app.models.UserPhoto.id = db.Column(db.Integer, primary_key=True)  [static]
```

6.9.1.3 is_profile

```
app.models.UserPhoto.is_profile = db.Column(db.Boolean, default=False)  [static]
```

6.9.1.4 photo_path

```
app.models.UserPhoto.photo_path = db.Column(db.String(255), nullable=False)  [static]
```

6.9.1.5 user_id

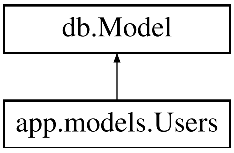
```
app.models.UserPhoto.user_id = db.Column(db.Integer, db.ForeignKey('users.id'), nullable=False)
[static]
```

The documentation for this class was generated from the following file:

- models.py

6.10 app.models.Users Class Reference

Inheritance diagram for app.models.Users:



Public Member Functions

- __repr__ (self)

Static Public Attributes

- id = db.Column(db.Integer, primary_key=True, autoincrement=True)
- email = db.Column(db.String(254), unique=True, nullable=False, index=True)
- name = db.Column(db.String(50), nullable=False)
- surname = db.Column(db.String(50), nullable=False)
- phone = db.Column(db.String(20), nullable=True)
- address = db.Column(db.String(100), nullable=True)
- facebook = db.Column(db.String(50), nullable=True)
- instagram = db.Column(db.String(50), nullable=True)
- linkedin = db.Column(db.String(50), nullable=True)
- password = db.Column(db.String(128), nullable=False)
- is_active = db.Column(db.Boolean, default=True)
- about_me = db.Column(db.TEXT(250), nullable=True)
- photos = db.relationship('UserPhoto', backref='user', lazy=True)
- created_at = db.Column(db.DateTime, default=db.func.current_timestamp())
- updated_at = db.Column(db.DateTime, default=db.func.current_timestamp(), onupdate=db.func.current_↩ timestamp())
- friends

6.10.1 Member Function Documentation

6.10.1.1 __repr__()

```
app.models.Users.__repr__ (
              self)
```

6.10.2 Member Data Documentation

6.10.2.1 about_me

```
app.models.Users.about_me = db.Column(db.TEXT(250), nullable=True)  [static]
```

6.10.2.2 address

```
app.models.Users.address = db.Column(db.String(100), nullable=True)  [static]
```

6.10.2.3 created_at

```
app.models.Users.created_at = db.Column(db.DateTime, default=db.func.current_timestamp())
[static]
```

6.10.2.4 email

```
app.models.Users.email = db.Column(db.String(254), unique=True, nullable=False, index=True)
[static]
```

6.10.2.5 facebook

```
app.models.Users.facebook = db.Column(db.String(50), nullable=True)  [static]
```

6.10.2.6 friends

```
app.models.Users.friends  [static]
```

Initial value:

```
=  db.relationship(
        'Users',
        secondary='friends',
        primaryjoin=(id == Friends.user_id),
        secondaryjoin=(id == Friends.friend_id),
        backref='friends_back',
        lazy='dynamic'
    )
```

6.10.2.7 id

```
app.models.Users.id = db.Column(db.Integer, primary_key=True, autoincrement=True)  [static]
```

6.10.2.8 instagram

```
app.models.Users.instagram = db.Column(db.String(50), nullable=True)  [static]
```

6.10.2.9 is_active

```
app.models.Users.is_active = db.Column(db.Boolean, default=True)  [static]
```

6.10.2.10 linkedin

```
app.models.Users.linkedin = db.Column(db.String(50), nullable=True)  [static]
```

6.10.2.11 name

```
app.models.Users.name = db.Column(db.String(50), nullable=False)  [static]
```

6.10.2.12 password

```
app.models.Users.password = db.Column(db.String(128), nullable=False)  [static]
```

6.10.2.13 phone

```
app.models.Users.phone = db.Column(db.String(20), nullable=True)  [static]
```

6.10.2.14 photos

```
app.models.Users.photos = db.relationship('UserPhoto', backref='user', lazy=True)  [static]
```

6.10.2.15 surname

```
app.models.Users.surname = db.Column(db.String(50), nullable=False)  [static]
```

6.10.2.16 updated_at

```
app.models.Users.updated_at = db.Column(db.DateTime, default=db.func.current_timestamp(),
onupdate=db.func.current_timestamp())  [static]
```

The documentation for this class was generated from the following file:

- models.py

Chapter 7

File Documentation

7.1 __init__.py File Reference

Namespaces

- namespace app

Functions

- app.create_app ()

Variables

- app.db = SQLAlchemy()
- app.jwt = JWTManager()
- app.socketio = SocketIO(cors_allowed_origins="∗", async_mode="threading")

7.2 config.py File Reference

Classes

- class app.config.Config

Namespaces

- namespace app
- namespace app.config

7.3 models.py File Reference

Classes

- class app.models.MessageType
- class app.models.RoomType
- class app.models.Role
- class app.models.Friends
- class app.models.Users
- class app.models.UserPhoto
- class app.models.Room
- class app.models.Room_Users
- class app.models.Messages

Namespaces

- namespace app
- namespace app.models

7.4 sockets.py File Reference

Namespaces

- namespace app
- namespace app.sockets

Functions

- app.sockets.register_socket_handlers (socketio)

Index