pr7-zentratech-dev-test-backend

Generated by Doxygen 1.10.0

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all documented 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

Namespace Documentation

4.1 app.admin Namespace Reference

Classes

- class ChatAdmin
- class ChatMessageAdmin
- class IntrestRequestAdmin

4.1.1 Detailed Description

```
@file admin.py
@brief Django admin configuration for the IntrestRequest model.
@details This file contains the admin interface customization for the
         IntrestRequest model, including how the model is displayed in
         the Django admin panel.
```

4.2 app.models Namespace Reference

Classes

- class Chat
- class ChatMessage
- class IntrestRequest

Variables

- **User** = get_user_model()

4.2.1 Detailed Description

```
@file models.py
@brief Model definitions for handling interest requests between users.
@details This file contains the model for managing interest requests,
         including the status of requests, timestamps, and relationships
         between users.
```

4.3 app.serializers Namespace Reference

Classes

- class ChatSerializer
- class IntrestRequestSerializer
- class MessageSerializer

4.3.1 Detailed Description

```
@file serializers.py
@brief Serializers for handling IntrestRequest model data.
@details This file contains the serializer for the IntrestRequest model, which includes
         field definitions, validation logic, and nested user serializers.
```

4.4 app.signals Namespace Reference

Functions

- addFriend (sender, instance, ∗∗kwargs)

Variables

- **User** = get_user_model()

4.4.1 Detailed Description

```
@file signals.py
@brief Signal handlers for IntrestRequest model.
@details This file contains the signal handlers that manage the actions triggered
         after saving an IntrestRequest instance, such as adding users as friends
         when a request is accepted.
```

4.4.2 Function Documentation

4.4.2.1 addFriend()

```
app.signals.addFriend (
              sender,
              instance,
             ** kwargs )
```

```
@brief Adds users as friends upon accepting an IntrestRequest.
@details This signal is triggered after an IntrestRequest instance is saved.
         If the request status is "accept", it adds the requesting user and
         the requested user as friends in the database.

@param sender The model class that sent the signal (IntrestRequest).
@param instance The actual IntrestRequest instance being saved.
@param kwargs Additional keyword arguments passed to the signal.
```

4.5 app.sockets Namespace Reference

Functions

- connect (sid, env, auth)
- connectChat (sid, data)
- messageRecieve (sid, data)

Variables

- **User** = get_user_model()
- **mgr** = socketio.AsyncRedisManager("redis://127.0.0.1:6379")
- sio

4.5.1 Detailed Description

```
@file sockets.py
@brief Socket.IO server setup and event handlers for chat functionality.
@details This file configures the Socket.IO server and defines event handlers for
         managing chat connections and message exchanges.
```

4.5.2 Function Documentation

4.5.2.1 connect()

```
app.sockets.connect (
              sid,
              env,
              auth )
```

```
@brief Handles client connections to the Socket.IO server.
@details This event handler is triggered when a client successfully connects to the server.
@param sid The session ID for the connected client.
@param env The environment in which the connection is established.
@param auth Authentication data provided by the client, if any.
```

4.5.2.2 connectChat()

```
app.sockets.connectChat (
              sid,
              data )
```

```
@brief Handles a client's request to join a chat room.
@details This event handler allows a client to join a specific chat room based on the chat ID provided.
@param sid The session ID for the connected client.
@param data A dictionary containing the chat ID.
```

4.5.2.3 messageRecieve()

```
app.sockets.messageRecieve (
              sid,
              data )
```

```
@brief Handles the reception of a message from a client.
@details This event handler processes incoming messages, saves them to the database,
         and broadcasts them to all clients in the relevant chat room.
@param sid The session ID for the connected client.
@param data A dictionary containing the sender's username and the message content.
```

4.5.3 Variable Documentation

4.5.3.1 sio

```
app.sockets.sio
```

Initial value:

```
00001 =  socketio.AsyncServer(
00002     async_mode="asgi", client_manager=mgr, cors_allowed_origins="*"
00003 )
```

4.6 app.tests Namespace Reference

Classes

- class ChatsViewTest
- class IndexViewTest
- class IntrestRequestExistsTest
- class IntrestRequestViewTest
- class ListUsersTest
- class MessageViewTest
- class TestSetup

Variables

- **User** = get_user_model()

4.6.1 Detailed Description

```
@file tests.py
@brief Unit tests for the chat functionality using Django's TestCase.
@details This file contains test cases for various views and models
         related to the chat application, including IntrestRequest,
         Chat, and ChatMessage models.
```

4.7 app.urls Namespace Reference

Variables

- list urlpatterns

4.7.1 Detailed Description

```
@file urls.py
@brief URL routing for the application.
@details This file defines the URL patterns for the application, mapping URLs
         to the appropriate views for handling requests.
```

4.7.2 Variable Documentation

4.7.2.1 urlpatterns

```
list app.urls.urlpatterns
```

Initial value:

```
00001 =  [
00002     # @brief Home page route.
00003     # @details Maps the root URL (”) to the IndexView, which handles the home page.
00004     path(”, views.IndexView.as_view(), name="index"),
00005
00006     # @brief Route for creating an interest request.
00007     # @details Maps the ’request’ URL to the IntrestRequestView, which handles the
00008     #          creation and status updatation of interest requests.
00009     path(’request’, views.IntrestRequestView.as_view(), name="request"),
00010
00011     # @brief Route for listing users.
00012     # @details Maps the ’list_users’ URL to the ListUsers view, which returns a list of users.
00013     path(’list_users’, views.ListUsers.as_view(), name="list_users"),
00014
00015     # @brief Route for checking if a request has been sent.
00016     # @details Maps the ’check_request_sent’ URL to the IntrestRequestExists view, which checks
00017     #          if an interest request has already been sent between users.
00018     path(’check_request_sent’, views.IntrestRequestExists.as_view(), name="check_request_sent"),
00019
00020     # @brief Route for list and create chats.
00021     # @details Maps the ’chats’ URL to the ChatsView view, which handles the
00022     #           createtion and listing chats.
00023     path(’chats’, views.ChatsView.as_view(), name="chats"),
00024
00025     # @brief Route for list and create messages.
00026     # @details Maps the ’messages’ URL to the MessageView view, which handles the
00027     #           createtion and listing messages.
00028     path(’messages’, views.MessageView.as_view(), name="messages"),
00029 ]
```

4.8 app.views Namespace Reference

Classes

- class ChatsView
- class IndexView
- class IntrestRequestExists
- class IntrestRequestView
- class ListUsers
- class MessageView

Variables

- **User** = get_user_model()

4.8.1 Detailed Description

```
@file views.py
@brief Contains the views for handling requests in the application.
@details This file defines the views for the application using Django's APIView class.
         It includes views for handling interest requests between users, such as
         creating, updating, and listing requests, as well as checking if a request exists.
```

4.9 authentication.admin Namespace Reference

Classes

- class CustomUserAdmin

4.9.1 Detailed Description

```
@file admin.py
@brief Admin configuration for CustomUser model.

This module defines the admin interface for managing the CustomUser model in the Django admin site.
It customizes the user creation and change forms, and defines how the CustomUser model should be displayed
in the admin interface, including which fields are shown and how they are grouped.

@module
```

4.10 authentication.forms Namespace Reference

Classes

- class CustomUserChangeForm
- class CustomUserCreationForm

4.10.1 Detailed Description

```
@file forms.py
@brief Forms for managing CustomUser model.

This module defines forms for creating and changing instances of the CustomUser model.
It extends Django's built-in `UserCreationForm` and `UserChangeForm` to include additional fields
specific to the CustomUser model.

@module
```

4.11 authentication.manager Namespace Reference

Classes

- class UserManager

4.11.1 Detailed Description

```
@file manager.py
@brief Custom user manager for handling user creation and superuser management.

This module defines a custom user manager that extends Django's `BaseUserManager`.
It provides methods for creating regular users and superusers with additional validation and settings.

@module
```

4.12 authentication.models Namespace Reference

Classes

- class CustomUser

4.12.1 Detailed Description

```
@file models.py
@brief Defines the custom user model for the application.
@details This file contains the CustomUser model, which extends Django's AbstractUser model to include additio
```

4.13 authentication.serializers Namespace Reference

Classes

- class userSerializer

Variables

- **User** = get_user_model()

4.13.1 Detailed Description

```
@file serializers.py
@brief Serializer definitions for the User model.
@details This file contains the serializer for the CustomUser model, which includes field definitions, validat
```

4.14 authentication.urls Namespace Reference

Variables

- list urlpatterns

4.14.1 Detailed Description

```
@file urls.py
@brief URL routing definitions for the authentication app.
@details This file defines the URL patterns that route requests to the appropriate views in the authentication
```

4.14.2 Variable Documentation

4.14.2.1 urlpatterns

```
list authentication.urls.urlpatterns
```

Initial value:

```
00001 =  [
00002     # ---------------------------------------------------------------
00003     # @brief URL pattern for user signup.
00004     # @details Routes to the SignUpUser view which handles user registration.
00005     # @route /signup
00006     # ---------------------------------------------------------------
00007     path('signup', views.SignUpUser.as_view()),
00008
00009     # ---------------------------------------------------------------
00010     # @brief URL pattern for user login.
00011     # @details Routes to the LoginUser view which handles user authentication.
00012     # @route /login
00013     # ---------------------------------------------------------------
00014     path('login', views.LoginUser.as_view()),
00015
00016     # ---------------------------------------------------------------
00017     # @brief URL pattern to check username availability.
00018     # @details Routes to the UserAvailability view which checks if a username is available.
00019     # @route /username_availability
00020     # ---------------------------------------------------------------
00021     path('username_availability', views.UserAvailability.as_view()),
00022
00023     # ---------------------------------------------------------------
00024     # @brief URL pattern to get the authenticated user's information.    # @details Routes to the
      UserIsAuthenticated view which returns the currently authenticated user's data.
00025     # @route /get_user
00026     # ---------------------------------------------------------------
00027     path('get_user', views.UserIsAuthenticated.as_view()),
00028 ]
```

4.15 authentication.views Namespace Reference

Classes

- class LoginUser
- class SignUpUser
- class UserAvailability
- class UserIsAuthenticated

Functions

- getUserToken (user)

Variables

- **User** = get_user_model()

4.15.1 Detailed Description

```
@file views.py
@brief API views for user authentication and registration.

This module contains API views for user authentication and registration using Django Rest Framework.
It includes views for checking username availability, user registration, and user login. JWT tokens
are utilized for authentication, and user data is managed using Django's built-in User model.

@module
```

4.15.2 Function Documentation

4.15.2.1 getUserToken()

```
authentication.views.getUserToken (
              user )
```

```
@brief Generates JWT tokens for the given user.

This function creates and returns refresh and access tokens using the `RefreshToken`
class from the `rest_framework_simplejwt` package.

@param user The user object for whom tokens are generated.
@return A dictionary containing 'refresh' and 'access' tokens as strings.
```

4.16 manage Namespace Reference

Functions

- main ()

4.16.1 Detailed Description

```
Django's command-line utility for administrative tasks.
```

4.16.2 Function Documentation

4.16.2.1 main()

```
manage.main ( )
```

```
Run administrative tasks.
```

4.17 pr7_zentra_test.asgi Namespace Reference

Variables

- **django_asgi_app** = get_asgi_application()
- **application** = socketio.ASGIApp(sio, django_asgi_app)

4.17.1 Detailed Description

```
ASGI config for pr7_zentra_test project.

It exposes the ASGI callable as a module-level variable named ``application``.

For more information on this file, see
https://docs.djangoproject.com/en/5.1/howto/deployment/asgi/
```

4.18 pr7_zentra_test.settings Namespace Reference

Variables

- **BASE_DIR** = Path(__file__).resolve().parent.parent
- str **SECRET_KEY** = 'django-insecure-@om_bd∗$oapc(6fva52r=5^c(jy2lonk@ct∗cdsao&q_bxr1a2'
- bool **DEBUG** = True
- list **ALLOWED_HOSTS** = [ ]
- list INSTALLED_APPS
- list MIDDLEWARE
- str **STATICFILES_STORAGE** = 'whitenoise.storage.CompressedManifestStaticFilesStorage'
- str **ROOT_URLCONF** = 'pr7_zentra_test.urls'
- list TEMPLATES
- str **ASGI_APPLICATION** = 'pr7_zentra_test.asgi.application'
- dict DATABASES
- list AUTH_PASSWORD_VALIDATORS
- bool **CORS_ORIGIN_ALLOW_ALL** = True
- dict REST_FRAMEWORK

4.18.1 Detailed Description

```
Django settings for pr7_zentra_test project.

Generated by 'django-admin startproject' using Django 5.1.

For more information on this file, see
https://docs.djangoproject.com/en/5.1/topics/settings/

For the full list of settings and their values, see
https://docs.djangoproject.com/en/5.1/ref/settings/
```

4.18.2 Variable Documentation

4.18.2.1 AUTH_PASSWORD_VALIDATORS

list pr7_zentra_test.settings.AUTH_PASSWORD_VALIDATORS

Initial value:

```
00001 =  [
00002     {
00003         'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
00004     },
00005     {
00006         'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
00007     },
00008     {
00009         'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
00010     },
00011     {
00012         'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
00013     },
00014 ]
```

4.18.2.2 DATABASES

dict pr7_zentra_test.settings.DATABASES

Initial value:

```
00001 =  {
00002     'default': {
00003         'ENGINE': 'django.db.backends.sqlite3',
00004         'NAME': BASE_DIR / 'db.sqlite3',
00005     }
00006 }
```

4.18.2.3 INSTALLED_APPS

list pr7_zentra_test.settings.INSTALLED_APPS

Initial value:

```
00001 =  [
00002     'django.contrib.admin',
00003     'django.contrib.auth',
00004     'django.contrib.contenttypes',
00005     'django.contrib.sessions',
00006     'django.contrib.messages',
00007     'django.contrib.staticfiles',
00008     'rest_framework',
00009     'app.apps.AppConfig',
00010     'authentication.apps.AuthenticationConfig'
00011 ]
```

4.18.2.4 MIDDLEWARE

list pr7_zentra_test.settings.MIDDLEWARE

Initial value:

```
00001 =  [
00002     'django.middleware.security.SecurityMiddleware',
00003     'django.contrib.sessions.middleware.SessionMiddleware',
00004     'django.middleware.common.CommonMiddleware',
00005     'django.middleware.csrf.CsrfViewMiddleware',
00006     'django.contrib.auth.middleware.AuthenticationMiddleware',
00007     'django.contrib.messages.middleware.MessageMiddleware',
00008     'django.middleware.clickjacking.XFrameOptionsMiddleware',
00009     'corsheaders.middleware.CorsMiddleware',
00010     'whitenoise.middleware.WhiteNoiseMiddleware',
00011 ]
```

4.18.2.5 REST_FRAMEWORK

dict pr7_zentra_test.settings.REST_FRAMEWORK

Initial value:

```
{
    'DEFAULT_AUTHENTICATION_CLASSES': (
        'rest_framework_simplejwt.authentication.JWTAuthentication',
    )
}
```

4.18.2.6 TEMPLATES

list pr7_zentra_test.settings.TEMPLATES

Initial value:

```
[
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]
```

4.19 pr7_zentra_test.urls Namespace Reference

Variables

- list urlpatterns

4.19.1 Detailed Description

```
URL configuration for pr7_zentra_test project.

The `urlpatterns` list routes URLs to views. For more information please see:
    https://docs.djangoproject.com/en/5.1/topics/http/urls/
Examples:
Function views
    1. Add an import:  from my_app import views
    2. Add a URL to urlpatterns:  path('', views.home, name='home')
Class-based views
    1. Add an import:  from other_app.views import Home
    2. Add a URL to urlpatterns:  path('', Home.as_view(), name='home')
Including another URLconf
    1. Import the include() function: from django.urls import include, path
    2. Add a URL to urlpatterns:  path('blog/', include('blog.urls'))
```

4.19.2 Variable Documentation

4.19.2.1 urlpatterns

list pr7_zentra_test.urls.urlpatterns

Initial value:

```
00001 =  [
00002     path('admin/', admin.site.urls),
00003     path('', include('app.urls')),
00004     path('auth/', include('authentication.urls')),
00005 ]
```

4.20 pr7_zentra_test.wsgi Namespace Reference

Variables

- **application** = get_wsgi_application()

4.20.1 Detailed Description

WSGI config for pr7_zentra_test project.

It exposes the WSGI callable as a module-level variable named ``application``.

For more information on this file, see
https://docs.djangoproject.com/en/5.1/howto/deployment/wsgi/

4.21 server Namespace Reference

Variables

- **log_level**
- **access_log**

4.21.1 Detailed Description

@file server.py
@brief Entry point for running the ASGI application with Uvicorn.
@details This script starts the ASGI server using Uvicorn to serve the application
defined in the `pr7_zentra_test` module. The server runs with debug logging
and access logging enabled.

Chapter 5

Class Documentation

5.1 app.apps.AppConfig Class Reference

Public Member Functions

- None **ready** (self)

Static Public Attributes

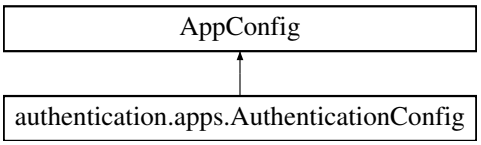
- str **default_auto_field** = 'django.db.models.BigAutoField'
- str **name** = 'app'

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

- src/app/apps.py

5.2 authentication.apps.AuthenticationConfig Class Reference

Inheritance diagram for authentication.apps.AuthenticationConfig:



Static Public Attributes

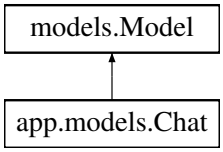
- str **default_auto_field** = 'django.db.models.BigAutoField'
- str **name** = 'authentication'

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

- src/authentication/apps.py

5.3 app.models.Chat Class Reference

Inheritance diagram for app.models.Chat:



Static Public Attributes

- initiator
- acceptor
- **short_id** = models.CharField(max_length=255, default=uuid.uuid4, unique=True)

5.3.1 Detailed Description

```
@class Chat
@brief Model representing a chat session between two users.
@details This model defines a chat session, including the users involved and a unique identifier for the chat.
```

5.3.2 Member Data Documentation

5.3.2.1 acceptor

```
app.models.Chat.acceptor  [static]
```

Initial value:

```
=  models.ForeignKey(
        User, on_delete=models.DO_NOTHING, related_name="acceptor_name"
    )
```

5.3.2.2 initiator

```
app.models.Chat.initiator  [static]
```

Initial value:

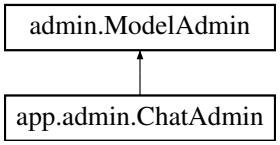
```
=  models.ForeignKey(
        User, on_delete=models.DO_NOTHING, related_name="initiator_chat"
    )
```

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

- src/app/models.py

5.4 app.admin.ChatAdmin Class Reference

Inheritance diagram for app.admin.ChatAdmin:



Static Public Attributes

- list list_display

5.4.1 Detailed Description

```
@brief Admin interface for the Chat model.
@details This class customizes the Django admin interface for the
         Chat model, specifying which fields are displayed
         in the list view.
```

5.4.2 Member Data Documentation

5.4.2.1 list_display

```
list app.admin.ChatAdmin.list_display  [static]
```

Initial value:

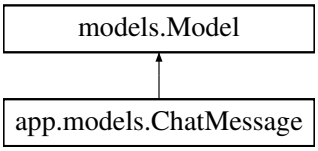
```
=  [
        "short_id",
        "initiator",
        "acceptor",
    ]
```

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

- src/app/admin.py

5.5 app.models.ChatMessage Class Reference

Inheritance diagram for app.models.ChatMessage:



Static Public Attributes

- **chat** = models.ForeignKey(Chat, on_delete=models.CASCADE, related_name="messages")
- **sender** = models.ForeignKey(User, on_delete=models.DO_NOTHING)
- **text** = models.TextField()
- **created_at** = models.DateTimeField(default=timezone.now)

5.5.1 Detailed Description

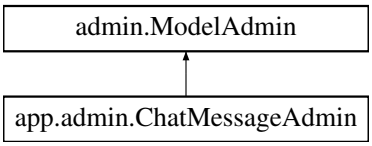
```
@class ChatMessage
@brief Model representing a message in a chat session.
@details This model defines a message sent within a chat session, including the sender, the message text, and
```

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

- src/app/models.py

5.6 app.admin.ChatMessageAdmin Class Reference

Inheritance diagram for app.admin.ChatMessageAdmin:



Static Public Attributes

- list list_display

5.6.1 Detailed Description

```
@brief Admin interface for the ChatMessage model.
@details This class customizes the Django admin interface for the
         ChatMessage model, specifying which fields are displayed
         in the list view.
```

5.6.2 Member Data Documentation

5.6.2.1 list_display

```
list app.admin.ChatMessageAdmin.list_display  [static]
```

Initial value:

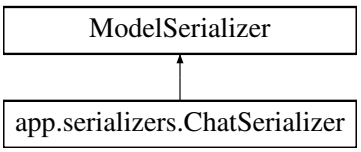
```
=  [
        "created_at",
        "chat",
        "sender",
        "text"
    ]
```

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

- src/app/admin.py

5.7 app.serializers.ChatSerializer Class Reference

Inheritance diagram for app.serializers.ChatSerializer:



Classes

- class Meta

Public Member Functions

- validate (self, data)

Static Public Attributes

- **initiator** = userSerializer(read_only=True)
- **acceptor** = userSerializer(read_only=True)

5.7.1 Detailed Description

```
@brief Serializer for the Chat model.
@details This serializer handles the serialization of Chat instances, including nested user data.
```

5.7.2 Member Function Documentation

5.7.2.1 validate()

```
app.serializers.ChatSerializer.validate (
              self,
              data )
```

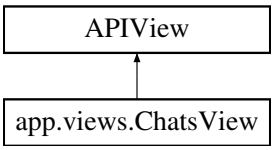
```
@brief Validates the data before saving.
@details This method performs additional validation checks if needed.
@param data The data dictionary that needs validation.
@return Returns the validated data.
```

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

- src/app/serializers.py

5.8 app.views.ChatsView Class Reference

Inheritance diagram for app.views.ChatsView:



Public Member Functions

- get (self, request)
- post (self, request)

Static Public Attributes

- list **permission_classes** = [IsAuthenticated]

5.8.1 Detailed Description

```
@brief View for handling chat operations.
@details This view handles GET and POST requests for retrieving and creating chats.
         Only authenticated users are allowed to access this view.
```

5.8.2 Member Function Documentation

5.8.2.1 get()

```
app.views.ChatsView.get (
              self,
              request )
```

```
@brief Handles GET requests to retrieve the list of chats.
@param request The HTTP request object.
@return Response A Response object containing the list of chats.
```

5.8.2.2 post()

```
app.views.ChatsView.post (
              self,
              request )
```

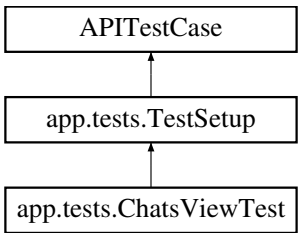
```
@brief Handles POST requests to create a new chat.
@param request The HTTP request object containing the chat data.
@return Response A Response object containing the created chat data.
```

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

- src/app/views.py

5.9 app.tests.ChatsViewTest Class Reference

Inheritance diagram for app.tests.ChatsViewTest:



Public Member Functions

- test_get_chats (self)
- test_post_chat (self)

Public Member Functions inherited from app.tests.TestSetup

- setUp (self)
- get_jwt_token (self, user)
- auth_headers (self, token)
- tearDown (self)

Additional Inherited Members

Public Attributes inherited from app.tests.TestSetup

- **user1**
- **user2**
- **intrest_request**
- **chat**
- **chat_message**
- **token**

5.9.1 Detailed Description

```
@brief Test case for the Chats view.
@details Tests the Chats view for retrieving and creating chat instances.
```

5.9.2 Member Function Documentation

5.9.2.1 test_get_chats()

```
app.tests.ChatsViewTest.test_get_chats (
              self )
```

```
@brief Tests retrieving chat instances.
@details Ensures that the GET request to the Chats view returns the correct list of chats.
```

5.9.2.2 test_post_chat()

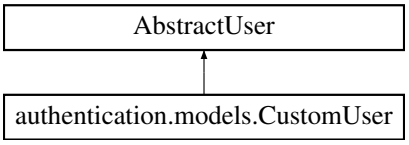
```
app.tests.ChatsViewTest.test_post_chat (
              self )

@brief Tests creating a new chat instance.
@details Ensures that the POST request to the Chats view successfully creates a new chat.
```

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

- src/app/tests.py

5.10 authentication.models.CustomUser Class Reference

Inheritance diagram for authentication.models.CustomUser:



Static Public Attributes

- **friends** = models.ManyToManyField('self', symmetrical=True)
- list **REQUIRED_FIELDS** = [ ]
- **objects** = UserManager()

5.10.1 Detailed Description

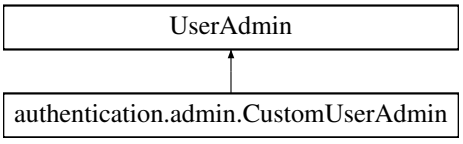
```
@brief A many-to-many relationship representing the user's friends.
@details This field allows each user to have multiple friends who are also users of the system.
         The relationship is symmetrical, meaning if user A is friends with user B, user B is also
         friends with user A.
```

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

- src/authentication/models.py

5.11 authentication.admin.CustomUserAdmin Class Reference

Inheritance diagram for authentication.admin.CustomUserAdmin:



Static Public Attributes

- **add_form** = CustomUserCreationForm
- **form** = CustomUserChangeForm
- **model** = CustomUser
- list list_display
- list **fieldsets**

5.11.1 Detailed Description

```
@brief Admin class for the CustomUser model.

This class defines the forms, fieldsets, and list display settings for the CustomUser model in the
Django admin interface.
```

5.11.2 Member Data Documentation

5.11.2.1 list_display

```
list authentication.admin.CustomUserAdmin.list_display  [static]
```

Initial value:

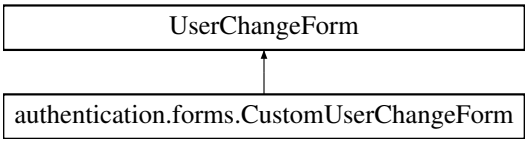
```
=  [
        'username',
        'email',
        'first_name',
        'last_name',
        'is_staff',
        'is_active',
        'is_superuser'
    ]
```

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

- src/authentication/admin.py

5.12 authentication.forms.CustomUserChangeForm Class Reference

Inheritance diagram for authentication.forms.CustomUserChangeForm:



Classes

- class Meta

5.12.1 Detailed Description

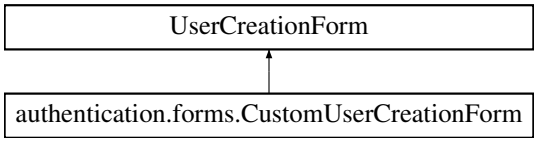
```
@brief Form for changing existing CustomUser instances.

This form extends the built-in UserChangeForm to include additional fields for the CustomUser model.
```

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

- src/authentication/forms.py

5.13 authentication.forms.CustomUserCreationForm Class Reference

Inheritance diagram for authentication.forms.CustomUserCreationForm:



Classes

- class Meta

5.13.1 Detailed Description

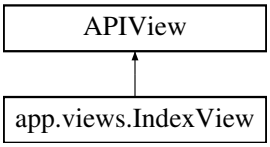
```
@brief Form for creating new CustomUser instances.

This form extends the built-in UserCreationForm to include additional fields for the CustomUser model.
```

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

- src/authentication/forms.py

5.14 app.views.IndexView Class Reference

Inheritance diagram for app.views.IndexView:



Public Member Functions

- get (self, request)

5.14.1 Detailed Description

```
@brief View for returning a simple greeting message.
@details This view handles GET requests and returns a response with a greeting message.
```

5.14.2 Member Function Documentation

5.14.2.1 get()

```
app.views.IndexView.get (
              self,
              request )
```

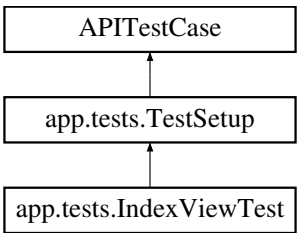
```
@brief Handles GET requests for the IndexView.
@param request The HTTP request object.
@return Response A Response object containing a greeting message.
```

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

- src/app/views.py

5.15 app.tests.IndexViewTest Class Reference

Inheritance diagram for app.tests.IndexViewTest:



Public Member Functions

- test_index_view (self)

Public Member Functions inherited from app.tests.TestSetup

- setUp (self)
- get_jwt_token (self, user)
- auth_headers (self, token)
- tearDown (self)

Additional Inherited Members

Public Attributes inherited from app.tests.TestSetup

- **user1**
- **user2**
- **intrest_request**
- **chat**
- **chat_message**
- **token**

5.15.1 Detailed Description

```
@brief Test case for the index view.
@details Tests the index view's response status and message content.
```

5.15.2 Member Function Documentation

5.15.2.1 test_index_view()

```
app.tests.IndexViewTest.test_index_view (
              self )
```

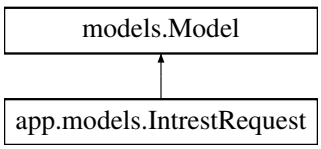
```
@brief Tests the index view.
@details Ensures that the index view returns a 200 OK status and the expected message.
```

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

- src/app/tests.py

5.16 app.models.IntrestRequest Class Reference

Inheritance diagram for app.models.IntrestRequest:



Static Public Attributes

- tuple STATUS_CHOICES
- **dt** = models.DateTimeField(default=timezone.now)
- **request_from** = models.ForeignKey(User, on_delete=models.CASCADE, related_name="request_from")
- **request_to** = models.ForeignKey(User, on_delete=models.CASCADE, related_name="request_to")
- **status** = models.CharField(max_length=1000, choices=STATUS_CHOICES, default="pending")

5.16.1 Detailed Description

```
@class IntrestRequest
@brief Model representing an interest request between users.
@details This model handles the storage and management of interest requests
         that one user sends to another. It includes fields for the request's
         status, the users involved, and a timestamp.
```

5.16.2 Member Data Documentation

5.16.2.1 STATUS_CHOICES

```
tuple app.models.IntrestRequest.STATUS_CHOICES  [static]
```

Initial value:

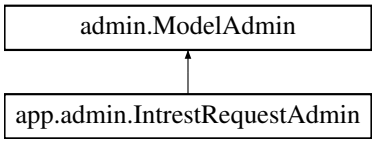
```
=  (
        ("pending", "PENDING"),
        ("accept", "ACCEPT"),
        ("reject", "REJECT")
    )
```

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

- src/app/models.py

5.17 app.admin.IntrestRequestAdmin Class Reference

Inheritance diagram for app.admin.IntrestRequestAdmin:



Static Public Attributes

- list list_display

5.17.1 Detailed Description

```
@brief Admin interface for the IntrestRequest model.
@details This class customizes the Django admin interface for the
         IntrestRequest model, specifying which fields are displayed
         in the list view.
```

5.17.2 Member Data Documentation

5.17.2.1 list_display

```
list app.admin.IntrestRequestAdmin.list_display  [static]
```

Initial value:

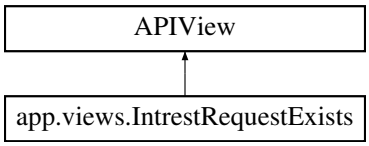
```
=  [
        "request_from",
        "request_to",
        "status"
    ]
```

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

- src/app/admin.py

5.18 app.views.IntrestRequestExists Class Reference

Inheritance diagram for app.views.IntrestRequestExists:



Public Member Functions

- post (self, request)

Static Public Attributes

- list **permission_classes** = [IsAuthenticated]

5.18.1 Detailed Description

```
@brief View for checking if an interest request has been sent.
@details This view handles POST requests to check if an interest request has already
         been sent from the current user to another user.
```

5.18.2 Member Function Documentation

5.18.2.1 post()

```
app.views.IntrestRequestExists.post (
              self,
              request )
```

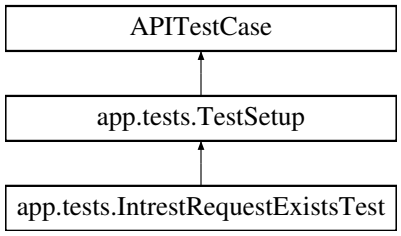
```
@brief Handles POST requests to check if an interest request exists.
@param request The HTTP request object containing the username to check against.
@return Response A Response object indicating whether the request exists.
```

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

- src/app/views.py

5.19 app.tests.IntrestRequestExistsTest Class Reference

Inheritance diagram for app.tests.IntrestRequestExistsTest:



Public Member Functions

- test_intrest_request_exists (self)

Public Member Functions inherited from app.tests.TestSetup

- setUp (self)
- get_jwt_token (self, user)
- auth_headers (self, token)
- tearDown (self)

Additional Inherited Members

Public Attributes inherited from app.tests.TestSetup

- **user1**
- **user2**
- **intrest_request**
- **chat**
- **chat_message**
- **token**

5.19.1 Detailed Description

```
@brief Test case for checking if an interest request exists.
@details Tests the IntrestRequestExists view for checking if a request has been sent.
```

5.19.2 Member Function Documentation

5.19.2.1 test_intrest_request_exists()

```
app.tests.IntrestRequestExistsTest.test_intrest_request_exists (
              self )
```

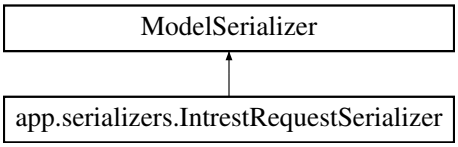
```
@brief Tests checking if an interest request exists.
@details Ensures that the POST request to the IntrestRequestExists view returns the correct status.
```

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

- src/app/tests.py

5.20 app.serializers.IntrestRequestSerializer Class Reference

Inheritance diagram for app.serializers.IntrestRequestSerializer:



Classes

- class Meta

Public Member Functions

- validate (self, data)

Static Public Attributes

- **request_to** = userSerializer(read_only=True)
- **request_from** = userSerializer(read_only=True)

5.20.1 Detailed Description

```
@brief Serializer for the IntrestRequest model.
@details This serializer handles the serialization and deserialization of
         IntrestRequest instances, including validation and nested user serializers.
```

5.20.2 Member Function Documentation

5.20.2.1 validate()

```
app.serializers.IntrestRequestSerializer.validate (
              self,
              data )
```

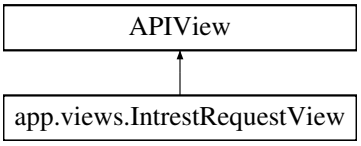
```
@brief Validates the data before saving.
@details Checks if the 'request_from' and 'request_to' fields are included in the data
         unless the request is partial (e.g., partial update).
@param data The data dictionary that needs validation.
@return Returns the validated data if all required fields are present.
@throws ValidationError if any required field is missing in the data.
```

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

- src/app/serializers.py

5.21 app.views.IntrestRequestView Class Reference

Inheritance diagram for app.views.IntrestRequestView:



Public Member Functions

- get (self, request)
- post (self, request)
- patch (self, request)

Static Public Attributes

- list **permission_classes** = [IsAuthenticated]

5.21.1 Detailed Description

```
@brief View for handling IntrestRequest operations.
@details This view handles GET, POST, and PATCH requests for managing IntrestRequest instances.
         Only authenticated users are allowed to access this view.
```

5.21.2 Member Function Documentation

5.21.2.1 get()

```
app.views.IntrestRequestView.get (
              self,
              request )
```

```
@brief Handles GET requests to retrieve pending IntrestRequest instances.
@param request The HTTP request object.
@return Response A Response object containing the serialized data of pending requests.
```

5.21.2.2 patch()

```
app.views.IntrestRequestView.patch (
              self,
              request )
```

```
@brief Handles PATCH requests to update an existing IntrestRequest.
@param request The HTTP request object containing the update data.
@return Response A Response object containing the updated data or an error message.
```

5.21.2.3 post()

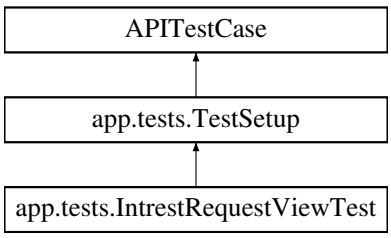
```
app.views.IntrestRequestView.post (
              self,
              request )


@brief Handles POST requests to create a new IntrestRequest.
@param request The HTTP request object containing the request data.
@return Response A Response object containing the status and message of the request.
```

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

- src/app/views.py

5.22 app.tests.IntrestRequestViewTest Class Reference

Inheritance diagram for app.tests.IntrestRequestViewTest:



Public Member Functions

- test_get_intrest_requests (self)
- test_post_intrest_request (self)
- test_patch_intrest_request (self)

Public Member Functions inherited from app.tests.TestSetup

- setUp (self)
- get_jwt_token (self, user)
- auth_headers (self, token)
- tearDown (self)

Additional Inherited Members

Public Attributes inherited from app.tests.TestSetup

- **user1**
- **user2**
- **intrest_request**
- **chat**
- **chat_message**
- **token**

5.22.1 Detailed Description

```
@brief Test case for the IntrestRequest view.
@details Tests the IntrestRequest view for GET, POST, and PATCH methods.
```

5.22.2 Member Function Documentation

5.22.2.1 test_get_intrest_requests()

```
app.tests.IntrestRequestViewTest.test_get_intrest_requests (
              self )
```

```
@brief Tests retrieving interest requests.
@details Ensures that the GET request to the IntrestRequest view returns the correct number of requests.
```

5.22.2.2 test_patch_intrest_request()

```
app.tests.IntrestRequestViewTest.test_patch_intrest_request (
              self )
```

```
@brief Tests updating an existing interest request.
@details Ensures that the PATCH request to the IntrestRequest view updates the request's status.
```

5.22.2.3 test_post_intrest_request()

```
app.tests.IntrestRequestViewTest.test_post_intrest_request (
              self )
```

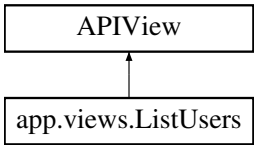
```
@brief Tests sending a new interest request.
@details Ensures that the POST request to the IntrestRequest view successfully creates a new request.
```

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

- src/app/tests.py

5.23 app.views.ListUsers Class Reference

Inheritance diagram for app.views.ListUsers:



Public Member Functions

- get (self, request)

Static Public Attributes

- list **permission_classes** = [IsAuthenticated]

5.23.1 Detailed Description

```
@brief View for listing users excluding those with existing interest requests.
@details This view handles GET requests to retrieve a list of users that the
         current user has not sent an interest request to.
```

5.23.2 Member Function Documentation

5.23.2.1 get()

```
app.views.ListUsers.get (
              self,
              request )
```

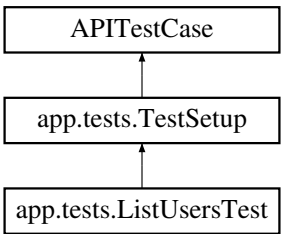
```
@brief Handles GET requests to list users.
@param request The HTTP request object.
@return Response A Response object containing a list of serialized user data.
```

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

- src/app/views.py

5.24 app.tests.ListUsersTest Class Reference

Inheritance diagram for app.tests.ListUsersTest:



Public Member Functions

- test_list_users (self)
- test_list_users_with_query (self)

Public Member Functions inherited from app.tests.TestSetup

- setUp (self)
- get_jwt_token (self, user)
- auth_headers (self, token)
- tearDown (self)

Additional Inherited Members

Public Attributes inherited from app.tests.TestSetup

- **user1**
- **user2**
- **intrest_request**
- **chat**
- **chat_message**
- **token**

5.24.1 Detailed Description

```
@brief Test case for the ListUsers view.
@details Tests the ListUsers view for listing users and querying users by username.
```

5.24.2 Member Function Documentation

5.24.2.1 test_list_users()

```
app.tests.ListUsersTest.test_list_users (
              self )
```

```
@brief Tests listing all users.
@details Ensures that the GET request to the ListUsers view returns the correct list of users.
```

5.24.2.2 test_list_users_with_query()

```
app.tests.ListUsersTest.test_list_users_with_query (
              self )
```

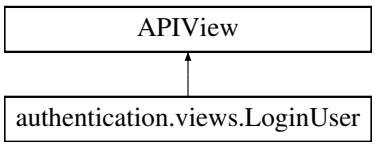
```
@brief Tests querying users by username.
@details Ensures that the GET request with a query string returns the correct list of users matching the query
```

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

- src/app/tests.py

5.25 authentication.views.LoginUser Class Reference

Inheritance diagram for authentication.views.LoginUser:



Public Member Functions

- post (self, request)

5.25.1 Detailed Description

```
@brief API view to handle user login.

This view processes POST requests for user authentication. It validates the credentials, generates
JWT tokens upon successful login, and returns them in the response.
```

5.25.2 Member Function Documentation

5.25.2.1 post()

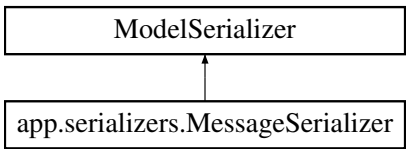
```
authentication.views.LoginUser.post (
              self,
              request )


@brief Handles POST requests for user login.

@param request The HTTP request object containing username and password.
@return Response A Response object with login status, JWT tokens, and user data upon success.
```

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

- src/authentication/views.py

5.26 app.serializers.MessageSerializer Class Reference

Inheritance diagram for app.serializers.MessageSerializer:



Classes

- class Meta

Static Public Attributes

- **sender** = userSerializer(read_only=True)

5.26.1 Detailed Description

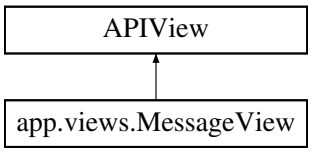
```
@brief Serializer for the ChatMessage model.
@details This serializer handles the serialization of chat messages, excluding the 'chat' field.
```

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

- src/app/serializers.py

5.27 app.views.MessageView Class Reference

Inheritance diagram for app.views.MessageView:



Public Member Functions

- get (self, request)

Static Public Attributes

- list **permission_classes** = [IsAuthenticated]

5.27.1 Detailed Description

```
@brief View for handling chat messages.
@details This view handles GET requests to retrieve chat messages for a specific chat.
         Only authenticated users are allowed to access this view.
```

5.27.2 Member Function Documentation

5.27.2.1 get()

```
app.views.MessageView.get (
              self,
              request )
```

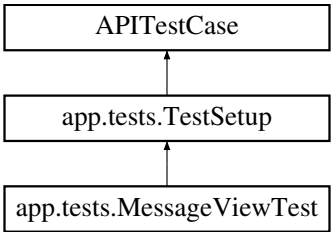
```
@brief Handles GET requests to retrieve chat messages.
@param request The HTTP request object containing the chat ID.
@return Response A Response object containing the list of chat messages.
```

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

- src/app/views.py

5.28 app.tests.MessageViewTest Class Reference

Inheritance diagram for app.tests.MessageViewTest:



Public Member Functions

- test_get_messages (self)
- test_get_messages_no_chat_id (self)

Public Member Functions inherited from app.tests.TestSetup

- setUp (self)
- get_jwt_token (self, user)
- auth_headers (self, token)
- tearDown (self)

Additional Inherited Members

Public Attributes inherited from app.tests.TestSetup

- **user1**
- **user2**
- **intrest_request**
- **chat**
- **chat_message**
- **token**

5.28.1 Detailed Description

```
@brief Test case for the Message view.
@details Tests the Message view for retrieving chat messages.
```

5.28.2 Member Function Documentation

5.28.2.1 test_get_messages()

```
app.tests.MessageViewTest.test_get_messages (
              self )

@brief Tests retrieving messages for a specific chat.
@details Ensures that the GET request to the Message view with a chat ID returns the correct list of messages.
```

5.28.2.2 test_get_messages_no_chat_id()

```
app.tests.MessageViewTest.test_get_messages_no_chat_id (
              self )

@brief Tests retrieving messages without specifying a chat ID.
@details Ensures that the GET request to the Message view without a chat ID returns an empty list.
```

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

- src/app/tests.py

5.29 app.serializers.ChatSerializer.Meta Class Reference

Static Public Attributes

- **model** = Chat
- list **fields** = ["short_id", "initiator", "acceptor"]

5.29.1 Detailed Description

```
@brief Meta options for the ChatSerializer.
@details This inner class defines the model being serialized and the fields to include
         in the serialized representation.
```

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

- src/app/serializers.py

5.30 app.serializers.IntrestRequestSerializer.Meta Class Reference

Static Public Attributes

- **model** = IntrestRequest
- list fields

5.30.1 Detailed Description

```
@brief Meta options for the IntrestRequestSerializer.
@details This inner class defines the model being serialized and the fields to include
         in the serialized representation.
```

5.30.2 Member Data Documentation

5.30.2.1 fields

```
list app.serializers.IntrestRequestSerializer.Meta.fields  [static]
```

Initial value:

```
=  [
            'id',
            'request_from',
            'request_to',
            'status'
        ]
```

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

- src/app/serializers.py

5.31 app.serializers.MessageSerializer.Meta Class Reference

Static Public Attributes

- **model** = ChatMessage
- tuple **exclude** = ("chat",)

5.31.1 Detailed Description

```
@brief Meta options for the MessageSerializer.
@details This inner class defines the model being serialized and the fields to include
         in the serialized representation.
```

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

- src/app/serializers.py

5.32 authentication.forms.CustomUserChangeForm.Meta Class Reference

Static Public Attributes

- **model** = CustomUser
- tuple **fields** = ('friends',)

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

- src/authentication/forms.py

5.33 authentication.forms.CustomUserCreationForm.Meta Class Reference

Static Public Attributes

- **model** = CustomUser
- tuple **fields** = ('friends',)

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

- src/authentication/forms.py

5.34 authentication.serializers.userSerializer.Meta Class Reference

Static Public Attributes

- **model** = User
- list fields
- dict **extra_kwargs** = {'password': {'write_only': True}}

5.34.1 Detailed Description

```
@brief Meta configuration for the userSerializer.
@details Specifies the model to be serialized and the fields to include.
         Also defines additional keyword arguments for certain fields.
```

5.34.2 Member Data Documentation

5.34.2.1 fields

```
list authentication.serializers.userSerializer.Meta.fields  [static]
```

Initial value:

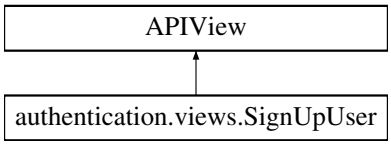
```
= [
            'username',
            'email',
            'password',
            'first_name',
            'last_name'
        ]
```

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

- src/authentication/serializers.py

5.35 authentication.views.SignUpUser Class Reference

Inheritance diagram for authentication.views.SignUpUser:



Public Member Functions

- post (self, request)

5.35.1 Detailed Description

```
@brief API view to handle user registration.

This view handles POST requests for creating a new user. It validates the input data using
a serializer, creates a user, and generates JWT tokens.
```

5.35.2 Member Function Documentation

5.35.2.1 post()

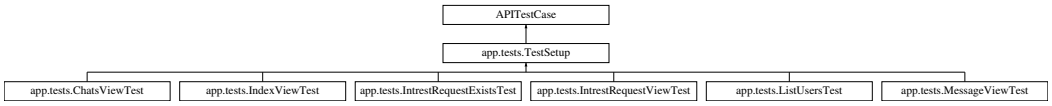
```
authentication.views.SignUpUser.post (
              self,
              request )

@brief Handles POST requests for user registration.

@param request The HTTP request object containing user registration data.
@return Response A Response object with registration status and JWT tokens upon success.
```

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

- src/authentication/views.py

5.36 app.tests.TestSetup Class Reference

Inheritance diagram for app.tests.TestSetup:



Public Member Functions

- setUp (self)
- get_jwt_token (self, user)
- auth_headers (self, token)
- tearDown (self)

Public Attributes

- **user1**
- **user2**
- **intrest_request**
- **chat**
- **chat_message**
- **token**

5.36.1 Detailed Description

```
@brief Setup for the test cases.
@details This class sets up the initial data required for the test cases,
         including creating users, IntrestRequest, Chat, and ChatMessage instances.
         It also provides utility methods to get JWT tokens and set authorization headers.
```

5.36.2 Member Function Documentation

5.36.2.1 auth_headers()

```
app.tests.TestSetup.auth_headers (
              self,
              token )
```

```
@brief Returns the authorization headers for the test requests.
@param token The JWT token to be included in the authorization header.
@return A dictionary containing the authorization header.
```

5.36.2.2 get_jwt_token()

```
app.tests.TestSetup.get_jwt_token (
              self,
              user )
```

```
@brief Generates a JWT token for a given user.
@param user The user object for which the JWT token is generated.
@return The JWT token as a string.
```

5.36.2.3 setUp()

```
app.tests.TestSetup.setUp (
              self )
```

```
@brief Initializes test data before each test case.
@details Creates two users, a chat interest request, a chat, and a chat message.
         Also obtains a JWT token for authentication in test requests.
```

5.36.2.4 tearDown()

```
app.tests.TestSetup.tearDown (
              self )
```

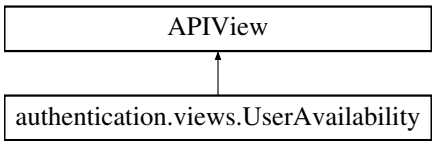
```
@brief Cleans up after each test case.
@details This method is called after each test case to perform any necessary cleanup.
```

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

- src/app/tests.py

5.37 authentication.views.UserAvailability Class Reference

Inheritance diagram for authentication.views.UserAvailability:



Public Member Functions

- post (self, request)

5.37.1 Detailed Description

```
@brief API view to check the availability of a username.

This view handles POST requests to determine if a given username is available in the database.
```

5.37.2 Member Function Documentation

5.37.2.1 post()

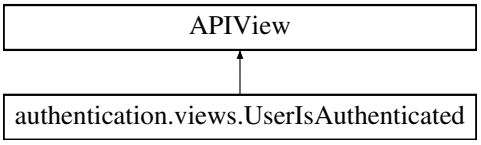
```
authentication.views.UserAvailability.post (
              self,
              request )


@brief Handles POST requests to check username availability.

@param request The HTTP request object containing the username to check.
@return Response A Response object indicating whether the username is available.
```

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

- src/authentication/views.py

5.38 authentication.views.UserIsAuthenticated Class Reference

Inheritance diagram for authentication.views.UserIsAuthenticated:



Public Member Functions

- get (self, request)

Static Public Attributes

- list **permission_classes** = [IsAuthenticated]

5.38.1 Detailed Description

```
@brief API view to get the authenticated user's information.

This view handles GET requests and returns the currently authenticated user's data.
```

5.38.2 Member Function Documentation

5.38.2.1 get()

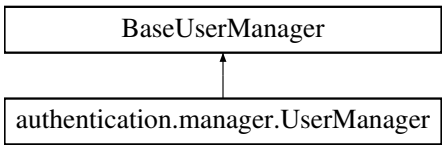
```
authentication.views.UserIsAuthenticated.get (
              self,
              request )

@brief Handles GET requests to return the authenticated user's data.

@param request The HTTP request object.
@return Response A Response object containing the user's data.
```

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

- src/authentication/views.py

5.39 authentication.manager.UserManager Class Reference

Inheritance diagram for authentication.manager.UserManager:



Public Member Functions

- create_user (self, username, password=None, ∗∗extra_fields)
- create_superuser (self, username, password=None, ∗∗extra_fields)

5.39.1 Detailed Description

```
@brief Custom manager for CustomUser model.

This class provides custom methods for creating users and superusers, including validation
for required fields and setting default values for superuser attributes.
```

5.39.2 Member Function Documentation

5.39.2.1 create_superuser()

```
authentication.manager.UserManager.create_superuser (
              self,
              username,
              password = None,
             ** extra_fields )

@brief Creates and returns a superuser with default admin permissions.

This method sets default values for `is_staff`, `is_superuser`, and `is_active`
fields to ensure the user has full admin privileges.

@param username The username for the superuser.
@param password The password for the superuser (optional).
@param extra_fields Additional fields to set for the superuser.

@return User A User object with superuser permissions.
```

5.39.2.2 create_user()

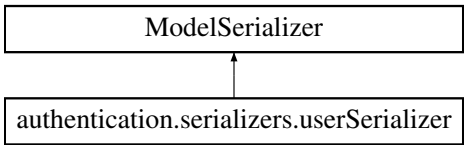
```
authentication.manager.UserManager.create_user (
              self,
              username,
              password = None,
             ** extra_fields )

@brief Creates and returns a regular user with an encrypted password.

@param username The username for the new user.
@param password The password for the new user (optional).
@param extra_fields Additional fields to set for the user.

@return User A User object with the provided username and other fields.

@raise ValueError If the username is not provided.
```

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

- src/authentication/manager.py

5.40 authentication.serializers.userSerializer Class Reference

Inheritance diagram for authentication.serializers.userSerializer:



Classes

- class Meta

Public Member Functions

- validate (self, data)
- create (self, validated_data)

5.40.1 Member Function Documentation

5.40.1.1 create()

```
authentication.serializers.userSerializer.create (
              self,
              validated_data )

@brief Creates a new CustomUser instance.
@details Handles the creation of a new user using the validated data.
@param validated_data The validated data for creating the user.
@return The newly created CustomUser instance.
```

5.40.1.2 validate()

```
authentication.serializers.userSerializer.validate (
              self,
              data )


@brief Validates the input data for the serializer.
@details Ensures that all required fields are present unless it's a partial update.
@param data The input data to validate.
@return The validated data if all required fields are present.
@throws ValidationError if any required field is missing.
```

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

- src/authentication/serializers.py