API Reference

Social Media Authentication API

API Version: 1.0.0

Social Media API - Authentication

CONTACT

EMAIL: sahirchawla2004@gmail.com

INDEX

1. ACTION 3

1.1 GET /action/follow/{userId} 3

1.2 GET /action/unfollow/{userId} 3

1.3 GET /action/followers/{userId} 4

1.4 GET /action/following/{userId} 5

1.5 GET /action/followers 6

1.6 GET /action/following 6

2. AUTH 8

2.1 POST /auth/register 8

2.2 POST /auth/login 8

3. POSTS 10

3.1 GET /posts 10

3.2 POST /posts 10

3.3 GET /posts/user/{userId} 11

3.4 GET /posts/{postId} 12

3.5 PUT /posts/{postId} 13

3.6 DELETE /posts/{postId} 13

3.7 GET /posts/view/latest-posts 14

4. PROFILE 16

4.1 GET /profile 16

4.2 PUT /profile 16

4.3 DELETE /profile 17

4.4 GET /profile/{userId} 18

API

1. ACTION

1.1 GET /action/follow/{userId}

Follow a user

Follows a user with the provided user ID.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*userId	string	ID of the user to follow

RESPONSE

STATUS CODE - 200: User followed successfully

RESPONSE MODEL - application/json

```
{
   message  string
}
```

STATUS CODE - 400: User ID is required or Cannot follow your own profile

RESPONSE MODEL - application/json

```
{
   error  string
}
```

STATUS CODE - 403: User not authenticated

RESPONSE MODEL - application/json

```
{
   error  string
}
```

STATUS CODE - 500: Internal server error

RESPONSE MODEL - application/json

```
{
   error  string
}
```

1.2 GET /action/unfollow/{userId}

Unfollow a user

Unfollows a user with the provided user ID.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*userId	string	ID of the user to unfollow

RESPONSE

STATUS CODE - 200: User unfollowed successfully

RESPONSE MODEL - application/json

```
{
   message  string
}
```

STATUS CODE - 400: User ID is required or Cannot unfollow your own profile

RESPONSE MODEL - application/json

```
{
   error  string
}
```

STATUS CODE - 403: User not authenticated

RESPONSE MODEL - application/json

```
{
   error  string
}
```

STATUS CODE - 500: Internal server error

RESPONSE MODEL - application/json

```
{
   error  string
}
```

1.3 `GET /action/followers/{userId}`

Get followers of a user

Retrieves the list of followers for a user with the provided user ID.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*userId	string	ID of the user to get followers for

RESPONSE

STATUS CODE - 200: List of followers of user with {userid}

RESPONSE MODEL - application/json

```
[{
```
Array of object:
```
   _id      string
   username string
}]
```

STATUS CODE - 400: User ID is required

RESPONSE MODEL - application/json

```
{
   error  string
}
```

STATUS CODE - 500: Internal server error

RESPONSE MODEL - application/json

```
{
   error  string
}
```

1.4 `GET /action/following/{userId}`

Get followings of a user

Retrieves the list of followings for a user with the provided user ID.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*userId	string	ID of the user to get followings for

RESPONSE

STATUS CODE - 200: List of followings of user with {userid}

RESPONSE MODEL - application/json

```
[{
```
Array of object:
```
   _id      string
   username string
}]
```

STATUS CODE - 400: User ID is required

RESPONSE MODEL - application/json

```
{
   error  string
}
```

STATUS CODE - 500: Internal server error

RESPONSE MODEL - application/json

```
{
   error  string
}
```

1.5 GET /action/followers

Get current user followers

Retrieves the list of followers for the authenticated user.

REQUEST

No request parameters

RESPONSE

STATUS CODE - 200: List of followers of logged in user

RESPONSE MODEL - application/json

```
[{
Array of object:
   _id       string
   username  string
}]
```

STATUS CODE - 403: User not authenticated

RESPONSE MODEL - application/json

```
{
   error  string
}
```

STATUS CODE - 500: Internal server error

RESPONSE MODEL - application/json

```
{
   error  string
}
```

1.6 GET /action/following

Get current user following users

Retrieves the list of users being followed by the authenticated user.

REQUEST

No request parameters

RESPONSE

STATUS CODE - 200: List of followings of logged in user

RESPONSE MODEL - application/json

```
[{
Array of object:
   _id       string
   username  string
}]
```

STATUS CODE - 403: User not authenticated

RESPONSE MODEL - application/json

```
{
  error string
}
```

STATUS CODE - 500: Internal server error

RESPONSE MODEL - application/json

```
{
  error string
}
```

2. AUTH

Authentication operations

2.1 POST /auth/register

Register a new user

Registers a new user with the provided details.

REQUEST

REQUEST BODY - application/json

```
{
   username           string
   password           string
   bio                string
   profilePictureURL  string
}
```

RESPONSE

STATUS CODE - 200: User registered successfully

RESPONSE MODEL - application/json

```
{
   message  string
}
```

STATUS CODE - 400: Username already taken or registration failed

RESPONSE MODEL - application/json

```
{
   error  string
}
```

STATUS CODE - 500: Internal server error

RESPONSE MODEL - application/json

```
{
   error  string
}
```

2.2 POST /auth/login

User login

Authenticates a user based on the provided username.

REQUEST

REQUEST BODY - application/json

```
{
    username  string
    password  string
  }
```

RESPONSE

STATUS CODE - 200: Login successful

RESPONSE MODEL - application/json

```
{
   token  string
}
```

STATUS CODE - 400: Invalid username/password

RESPONSE MODEL - application/json

```
{
   error  string
}
```

STATUS CODE - 500: Internal server error

RESPONSE MODEL - application/json

```
{
   error  string
}
```

3. POSTS

3.1 `GET /posts`

Get user posts

Retrieves the posts created by the authenticated user.

REQUEST

No request parameters

RESPONSE

STATUS CODE - 200: User posts retrieved successfully

RESPONSE MODEL - application/json

```
{
   username   string
   posts [{
   Array of object:
      _id         string
      userID      string
      textContent string
      createdAt   string
   }]
}
```

STATUS CODE - 403: User not authenticated

RESPONSE MODEL - application/json

```
{
   error  string
}
```

STATUS CODE - 500: Internal server error

RESPONSE MODEL - application/json

```
{
   error  string
}
```

3.2 `POST /posts`

Create a new post

Creates a new post for the authenticated user.

REQUEST

REQUEST BODY - application/json

```
{
   textContent  string
}
```

RESPONSE

STATUS CODE - 200: Post created successfully

RESPONSE MODEL - application/json

```
{
  message  string
}
```

STATUS CODE - 403: User not authenticated

RESPONSE MODEL - application/json

```
{
  error  string
}
```

STATUS CODE - 500: Internal server error

RESPONSE MODEL - application/json

```
{
  error  string
}
```

3.3 `GET /posts/user/{userId}`

Get posts by user ID

Retrieves the posts created by a user with the provided user ID.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*userId	string	ID of the user to get posts for

RESPONSE

STATUS CODE - 200: User posts retrieved successfully

RESPONSE MODEL - application/json

```
{
  username   string
  posts [{
  Array of object:
    _id          string
    userID       string
    textContent  string
    createdAt    string
  }]
}
```

STATUS CODE - 400: User ID is required

RESPONSE MODEL - application/json

```
{
    error  string
  }
```

STATUS CODE - 500: Internal server error

RESPONSE MODEL - application/json

```
{
  error  string
}
```

3.4 GET /posts/{postId}

Get post by post ID

Retrieves a post with the provided post ID for the authenticated user.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*postId	string	ID of the post to retrieve

RESPONSE

STATUS CODE - 200: Post retrieved successfully

RESPONSE MODEL - application/json

```
{
  _id          string
  userID       string
  textContent  string
  createdAt    string
}
```

STATUS CODE - 400: User ID is required

RESPONSE MODEL - application/json

```
{
  error  string
}
```

STATUS CODE - 404: Invalid Post ID

RESPONSE MODEL - application/json

```
{
  error  string
}
```

STATUS CODE - 500: Internal server error

RESPONSE MODEL - application/json

```
{
  error  string
}
```

3.5 `PUT /posts/{postId}`

Update post by post ID

Updates a post with the provided post ID for the authenticated user.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*`postId`	`string`	ID of the post to update

REQUEST BODY - application/json

```
{
   textContent  string
}
```

RESPONSE

STATUS CODE - 200: Post updated successfully

RESPONSE MODEL - application/json

```
{
   message  string
}
```

STATUS CODE - 403: User not authenticated

RESPONSE MODEL - application/json

```
{
   error  string
}
```

STATUS CODE - 404: Invalid Post ID

RESPONSE MODEL - application/json

```
{
   error  string
}
```

STATUS CODE - 500: Internal server error

RESPONSE MODEL - application/json

```
{
   error  string
}
```

3.6 `DELETE /posts/{postId}`

Delete post by post ID

Deletes a post with the provided post ID for the authenticated user.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*postId	string	ID of the post to delete

RESPONSE

STATUS CODE - 200: Post deleted successfully

RESPONSE MODEL - application/json

```
{
   message  string
}
```

STATUS CODE - 403: User not authenticated

RESPONSE MODEL - application/json

```
{
   error  string
}
```

STATUS CODE - 404: Invalid Post ID

RESPONSE MODEL - application/json

```
{
   error  string
}
```

STATUS CODE - 500: Internal server error

RESPONSE MODEL - application/json

```
{
   error  string
}
```

3.7 GET /posts/view/latest-posts

Get latest posts from followed users

Retrieves the latest posts from users being followed by the authenticated user.

REQUEST

No request parameters

RESPONSE

STATUS CODE - 200: posts of users followed by current retrieved successfully

RESPONSE MODEL - application/json

```
{
   posts [{
   Array of object:
      _id          string
      userID       string
      textContent  string
      createdAt    string
    }]
  }
```

STATUS CODE - 403: User not authenticated

RESPONSE MODEL - application/json

```
  {
    error  string
  }
```

STATUS CODE - 500: Internal server error

RESPONSE MODEL - application/json

```
  {
    error  string
  }
```

4. PROFILE

4.1 `GET /profile`

View authenticated user's profile

Retrieves the profile of the authenticated user.

REQUEST

No request parameters

RESPONSE

STATUS CODE - 200: Profile retrieved successfully

RESPONSE MODEL - application/json

STATUS CODE - 403: User not authenticated

RESPONSE MODEL - application/json

```
{
  error  string
}
```

STATUS CODE - 500: Internal server error

RESPONSE MODEL - application/json

```
{
  error  string
}
```

4.2 `PUT /profile`

Update authenticated user's profile

Updates the profile of a user by their user ID.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*userId	string	User ID of the profile to be updated

REQUEST BODY - application/json

```
{
  username*         string
  bio               string
  profilePictureURL string
}
```

RESPONSE

STATUS CODE - 200: User profile updated successfully

RESPONSE MODEL - application/json

```
{
   message  string
}
```

STATUS CODE - 403: User not authenticated

RESPONSE MODEL - application/json

```
{
   error  string
}
```

STATUS CODE - 500: Internal server error

RESPONSE MODEL - application/json

```
{
   error  string
}
```

4.3 DELETE /profile

Delete user profile by user ID

Deletes the profile of authenticated user

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*userId	string	User ID of the profile to be deleted

RESPONSE

STATUS CODE - 200: User profile deleted successfully

RESPONSE MODEL - application/json

```
{
   message  string
}
```

STATUS CODE - 403: User not authenticated or rate limiter

RESPONSE MODEL - application/json

```
{
   error  string
}
```

STATUS CODE - 500: Internal server error

RESPONSE MODEL - application/json

```
{
   error  string
}
```

4.4 GET /profile/{userId}

View user profile by user ID

Retrieves the profile of a user by their user ID.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*userId	string	User ID of the profile to be retrieved

RESPONSE

STATUS CODE - 200: User profile retrieved successfully

RESPONSE MODEL - application/json

STATUS CODE - 400: User ID is required

RESPONSE MODEL - application/json

```
{
   error  string
}
```

STATUS CODE - 500: Internal server error

RESPONSE MODEL - application/json

```
{
   error  string
}
```