Reddit Clone API Documentation

Base URL: `http://localhost:<PORT>/api`
Content-Type: `application/json`

POST /users/register

Create a new user account.

Auth: None

Request:

```
{
 "username": "string",
 "email": "string",
 "password": "string"
}
```

Response:

```
{
 "message": "User created successfully",
 "user": {
   "_id": "string",
   "username": "string",
   "email": "string",
   "avatar": "string",
   "karma": 0,
   "posts": [],
   "comments": [],
   "communities": [],
   "createdAt": "string",
   "updatedAt": "string"
 },
 "token": "string"
}
```

Errors:

```
{ "error": "All fields must be filled!" }
```

```
{ "error": "Incorrect email structure!" }
```

```
{ "error": "Email already exists!" }
```

```
{ "error": "Username already taken!" }
```

POST /users/login

Authenticate a user and return a token.

Auth: None

Request:

```
{
 "email": "string",
 "password": "string"
}
```

Response:

```
{
 "message": "Login successful",
 "user": {
   "_id": "string",
   "username": "string",
   "email": "string",
   "avatar": "string",
   "karma": 0,
   "posts": [],
   "comments": [],
   "communities": []
 },
 "token": "string"
}
```

Errors:

```
{ "error": "All fields must be filled!" }
```

```
{ "error": "User does not exist!" }
```

```
{ "error": "Incorrect password!" }
```

GET /users

Get all users.

Auth: None

Request: None

Response:

```
[
 {
   "_id": "string",
   "username": "string",
   "email": "string",
   "avatar": "string",
   "karma": 0,
   "posts": [],
   "comments": [],
   "communities": []
 }
]
```

Errors:

```
{ "error": "string" }
```

GET /users/:id

Get a specific user with all related data.

Auth: None

Request: None (id in URL path)

Response:

```
{
 "_id": "string",
 "username": "string",
 "email": "string",
 "avatar": "string",
 "karma": 0,
 "posts": [],
 "comments": [],
 "upvotedPosts": [],
 "downvotedPosts": [],
 "upvotedComments": [],
 "downvotedComments": []
}
```

Errors:

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

PATCH /users/:id

Update a user's profile.

Auth: None

Request:

```
{
 "username": "string",
 "email": "string",
 "avatar": "string"
}
```

Response:

```
{
 "message": "Profile updated successfully",
 "user": {
   "_id": "string",
   "username": "string",
   "email": "string",
   "avatar": "string",
   "karma": 0
 }
}
```

Errors:

```
{ "error": "Invalid user ID format" }
{ "error": "Incorrect email structure!" }
```

```
{ "error": "Email already exists!" }
```

```
{ "error": "Username already taken!" }
```

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

GET /users/:id/comments

Get all comments by a specific user.

Auth: None

Request: None (id in URL path)

Response:

```
[
 {
   "_id": "string",
   "text": "string",
   "user": {
     "_id": "string",
     "username": "string",
     "avatar": "string"
   },
   "post": {
     "_id": "string",
     "title": "string"
   },
   "karma": 0,
   "replies": [],
   "createdAt": "string",
   "updatedAt": "string"
 }
]
```

Errors:

```
{ "error": "string" }
```

GET /users/:id/upvoted

Get all posts upvoted by a specific user.

Auth: None

Request: None (id in URL path)

Response:

```
[
 {
   "_id": "string",
   "title": "string",
   "description": "string",
   "image": "string",
   "user": {
     "_id": "string",
     "username": "string",
     "avatar": "string"
   },
   "community": {
     "_id": "string",
     "name": "string",
     "logo": "string"
   },
   "votes": 0,
   "createdAt": "string",
   "updatedAt": "string"
 }
]
```

Errors:

```
{ "error": "string" }
```

GET /users/:id/downvoted

Get all posts downvoted by a specific user.

Auth: None

Request: None (id in URL path)

Response:

```
[
 {
   "_id": "string",
   "title": "string",
   "description": "string",
   "image": "string",
   "user": {
     "_id": "string",
     "username": "string",
     "avatar": "string"
   },
   "community": {
     "_id": "string",
     "name": "string",
     "logo": "string"
   },
   "votes": 0,
   "createdAt": "string",
   "updatedAt": "string"
 }
]
```

Errors:

```
{ "error": "string" }
```

GET /posts

Get all posts.

Auth: None

Request: None

Response:

```
[
 {
   "_id": "string",
   "title": "string",
   "description": "string",
   "image": "string",
   "user": {
     "_id": "string",
     "username": "string",
     "email": "string",
     "avatar": "string",
     "karma": 0
   },
   "community": {
     "_id": "string",
     "name": "string",
     "description": "string",
     "logo": "string",
     "backgroundImage": "string",
     "type": "string",
     "topic": "string"
   },
   "comments": [],
   "upvoters": [],
   "downvoters": [],
   "votes": 0,
   "createdAt": "string",
   "updatedAt": "string"
 }
]
```

Errors:

```
{ "error": "Failed to fetch posts" }
```

GET /posts/:id

Get a specific post with all comments.

Auth: None

Request: None (id in URL path)

Response:

```
{
 "_id": "string",
 "title": "string",
 "description": "string",
 "image": "string",
 "user": {
   "_id": "string",
   "username": "string",
   "email": "string",
   "avatar": "string",
   "karma": 0
 },
 "community": {
   "_id": "string",
   "name": "string",
   "description": "string",
   "logo": "string",
   "backgroundImage": "string",
   "type": "string",
   "topic": "string"
 },
 "comments": [],
 "upvoters": [],
 "downvoters": [],
 "votes": 0,
 "createdAt": "string",
 "updatedAt": "string"
}
```

Errors:

```
{ "error": "Post not found" }
```

```
{ "error": "Failed to fetch post" }
```

POST /posts

Create a new post.

Auth: None

Request:

```
{
 "title": "string",
 "description": "string",
 "image": "string",
 "user": "string",
 "community": "string"
}
```

Response:

```
{
 "_id": "string",
 "title": "string",
 "description": "string",
 "image": "string",
 "user": {
   "_id": "string",
   "username": "string",
   "email": "string",
   "avatar": "string",
   "karma": 0
 },
 "community": {
   "_id": "string",
   "name": "string",
   "description": "string",
   "logo": "string",
   "backgroundImage": "string",
   "type": "string",
   "topic": "string"
 },
 "comments": [],
 "upvoters": [],
 "downvoters": [],
 "votes": 0,
 "createdAt": "string",
 "updatedAt": "string"
}
```

Errors:

```
{ "error": "Failed to create post" }
```

PUT /posts/:id

Update a post.

Auth: None

Request:

```
{
 "title": "string",
 "description": "string",
 "image": "string"
}
```

Response:

```
{
 "_id": "string",
 "title": "string",
 "description": "string",
 "image": "string",
 "user": {
   "_id": "string",
   "username": "string",
   "email": "string",
   "avatar": "string",
   "karma": 0
 },
 "community": {
   "_id": "string",
   "name": "string",
   "description": "string",
   "logo": "string",
   "backgroundImage": "string",
   "type": "string",
   "topic": "string"
 },
 "comments": [],
 "upvoters": [],
 "downvoters": [],
 "votes": 0,
 "createdAt": "string",
 "updatedAt": "string"
}
```

Errors:

```
{ "error": "Post not found" }
```

```
{ "error": "Failed to update post" }
```

PATCH /posts/:id/vote

Upvote, downvote, or remove vote from a post.

Auth: None

Request:

```
{
 "delta": 1,
 "userId": "string"
}
```

Note: `delta` can be `1` (upvote), `-1` (downvote), or `0` (remove vote).

Response:

```
{
 "_id": "string",
 "title": "string",
 "description": "string",
 "image": "string",
 "user": {
   "_id": "string",
   "username": "string",
   "email": "string",
   "avatar": "string",
   "karma": 0
 },
 "community": {
   "_id": "string",
   "name": "string",
   "description": "string",
   "logo": "string",
   "backgroundImage": "string",
   "type": "string",
   "topic": "string"
 },
 "comments": [],
 "upvoters": [],
 "downvoters": [],
 "votes": 0,
 "createdAt": "string",
 "updatedAt": "string"
}
```

Errors:

```
{ "error": "userId required" }
```

```
{ "error": "No existing vote to remove" }
```

```
{ "error": "Post not found" }
{ "error": "Failed to update post vote" }
```

DELETE /posts/:id

Delete a post.

Auth: None

Request:

```
{
 "userId": "string"
}
```

Response:

```
{
 "message": "Post deleted"
}
```

Errors:

```
{ "error": "userId required" }
```

```
{ "error": "Not authorized" }
```

```
{ "error": "Post not found" }
```

```
{ "error": "Failed to delete post" }
```

GET /comments/:postId

Get all comments for a specific post.

Auth: None

Request: None (postId in URL path)

Response:

```
[
 {
   "_id": "string",
   "text": "string",
   "user": {
     "_id": "string",
     "username": "string",
     "avatar": "string"
   },
   "post": "string",
   "karma": 0,
   "parent": null,
   "replies": [],
   "createdAt": "string",
   "updatedAt": "string"
 }
]
```

Errors:

```
{ "error": "Failed to fetch comments" }
```

POST /comments

Create a new comment.

Auth: None

Request:

```
{
 "text": "string",
 "user": "string",
 "post": "string",
 "parent": "string"
}
```

Note: `parent` is optional. If provided, the comment is a reply to another comment.

Response:

```
{
 "_id": "string",
 "text": "string",
 "user": {
   "_id": "string",
   "username": "string",
   "email": "string",
   "avatar": "string",
   "karma": 0
 },
 "post": "string",
 "karma": 0,
 "parent": "string",
 "replies": [],
 "upvoters": [],
 "downvoters": [],
 "createdAt": "string",
 "updatedAt": "string"
}
```

Errors:

```
{ "error": "Failed to add comment" }
```

PATCH /comments/:id/vote

Upvote, downvote, or remove vote from a comment.

Auth: None

Request:

```
{
 "delta": 1,
 "userId": "string"
}
```

Note: `delta` can be `1` (upvote), `-1` (downvote), or `0` (remove vote).

Response:

```
{
 "_id": "string",
 "text": "string",
 "user": {
   "_id": "string",
   "username": "string",
   "email": "string",
   "avatar": "string",
   "karma": 0
 },
 "post": "string",
 "karma": 0,
 "parent": "string",
 "replies": [],
 "upvoters": [],
 "downvoters": [],
 "createdAt": "string",
 "updatedAt": "string"
}
```

Errors:

```
{ "error": "userId required" }
```

```
{ "error": "No existing vote to remove" }
```

```
{ "error": "Comment not found" }
```

```
{ "error": "Failed to update comment vote" }
```

DELETE /comments/:id

Delete a comment and all its replies.

Auth: None

Request:

```
{
 "userId": "string"
}
```

Response:

```
{
 "message": "Comment deleted"
}
```

Errors:

```
{ "error": "userId required" }
```

```
{ "error": "Not authorized" }
```

```
{ "error": "Comment not found" }
```

```
{ "error": "Failed to delete comment" }
```

GET /communities

Get all communities.

Auth: None

Request: None

Response:

```
[
 {
   "_id": "string",
   "name": "string",
   "description": "string",
   "logo": "string",
   "backgroundImage": "string",
   "type": "string",
   "topic": "string",
   "members": [],
   "posts": [],
   "createdBy": "string",
   "createdAt": "string",
   "updatedAt": "string"
 }
]
```

Errors:

```
{ "error": "Failed to fetch communities" }
```

GET /communities/:id

Get a specific community.

Auth: None

Request: None (id in URL path)

Response:

```
{
 "_id": "string",
 "name": "string",
 "description": "string",
 "logo": "string",
 "backgroundImage": "string",
 "type": "string",
 "topic": "string",
 "members": [],
 "posts": [],
 "createdBy": "string",
 "createdAt": "string",
 "updatedAt": "string"
}
```

Errors:

```
{ "error": "Community not found" }
```

```
{ "error": "Failed to fetch community" }
```

POST /communities

Create a new community.

Auth: None

Request:

```
{
 "name": "string",
 "description": "string",
 "logo": "string",
 "backgroundImage": "string",
 "type": "string",
 "topic": "string",
 "createdBy": "string"
}
```

Note: `type` must be one of: `"public"`, `"restricted"`, `"private"` (default: `"public"`).

Response:

```
{
 "_id": "string",
 "name": "string",
 "description": "string",
 "logo": "string",
 "backgroundImage": "string",
 "type": "string",
 "topic": "string",
 "members": [],
 "posts": [],
 "createdBy": "string",
 "createdAt": "string",
 "updatedAt": "string"
}
```

Errors:

```
{ "error": "Failed to create community" }
```

POST /communities/:id/join

Join a user to a community.

Auth: None

Request:

```
{
 "userId": "string"
}
```

Response:

```
{
 "message": "Joined community",
 "user": {
   "_id": "string",
   "username": "string",
   "email": "string",
   "communities": ["string"]
 },
 "community": {
   "_id": "string",
   "name": "string",
   "members": ["string"]
 }
}
```

Errors:

```
{ "error": "User or Community not found" }
```

```
{ "error": "Failed to join community" }
```

POST /communities/:id/leave

Remove a user from a community.

Auth: None

Request:

```
{
 "userId": "string"
}
```

Response:

```
{
 "message": "Left community",
 "user": {
   "_id": "string",
   "username": "string",
   "email": "string",
   "communities": []
 },
 "community": {
   "_id": "string",
   "name": "string",
   "members": []
 }
}
```

Errors:

```
{ "error": "User or Community not found" }
```

```
{ "error": "Failed to leave community" }
```

GET /communities/:id/posts

Get all posts in a specific community.

Auth: None

Request: None (id in URL path)

Response:

```
[
 {
   "_id": "string",
   "title": "string",
   "description": "string",
   "image": "string",
   "user": {
     "_id": "string",
     "username": "string"
   },
   "community": {
     "_id": "string",
     "name": "string"
   },
   "votes": 0,
   "createdAt": "string",
   "updatedAt": "string"
 }
]
```

Errors:

```
{ "error": "Failed to fetch community posts" }
```

DELETE /communities/:id

Delete a community and all associated posts and comments.

Auth: None

Request:

```
{
 "userId": "string"
}
```

Response:

```
{
 "message": "Community deleted successfully"
}
```

Errors:

```
{ "error": "Only the creator can delete this community" }
```

```
{ "error": "Community not found" }
```

```
{ "error": "Failed to delete community" }
```