Base URL

```
http://localhost:3000/api
```

Endpoints

1. **Authentication**

Register a New User

```
POST /auth/register
```

Request Body (JSON):

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

Response (JSON):

```
{
  "id": "uuid",
  "username": "string",
  "email": "string",
  "created_at": "timestamp"
}
```

Log In

```
POST /auth/login
```

Request Body (JSON):

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

Response (JSON):

```
{
  "token": "string"
}
```

- The returned `token` should be included in subsequent requests that require authentication, passed in the `Authorization` header.

Retrieve Current Authenticated User

```
GET /auth/me
```

Headers:

```
Authorization: <token>
```

(where `<token>` is the token returned from the `POST /auth/login` endpoint)

Response (JSON):

```
{
  "id": "uuid",
  "username": "string",
  "email": "string",
  "created_at": "timestamp"
}
```

- Requires valid `Authorization` header containing the token.

2. Users

Create a User

```
POST /users
```

> **Note**: This endpoint is functionally similar to `POST /auth/register`. Either can be used to create a user.

Request Body (JSON):

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

Response (JSON):

```
{
  "id": "uuid",
  "username": "string",
  "email": "string",
  "created_at": "timestamp"
}
```

Get All Users

```
GET /users
```

Response (JSON Array):

```
[
  {
    "id": "uuid",
    "username": "string",
    "email": "string",
    "created_at": "timestamp"
```

```
  }
]
```

Delete a User

```
DELETE /users/:id
```

Response (JSON):

```
{
  "id": "uuid",
  "username": "string",
  "email": "string"
}
```

---

3. Pictures

Create a Picture

```
POST /pictures
```

Request Body (JSON):

```
{
  "URL": "string",
  "caption": "string"
}
```

Response (JSON):

```
{
  "id": "uuid",
  "URL": "string",
  "caption": "string",
  "created_at": "timestamp"
}
```

Get All Pictures

```
GET /pictures
```

Response (JSON Array):

```
[
  {
    "id": "uuid",
    "URL": "string",
    "caption": "string",
    "created_at": "timestamp"
  }
]
```

Delete a Picture

```
DELETE /pictures/:id
```

Response (JSON):

```
{
  "id": "uuid",
  "URL": "string",
  "caption": "string"
}
```

---

4. Users & Pictures Relationship

Link a User to a Picture

```
POST /users/:userId/pictures/:pictureId
```

Response (JSON):

```
{
  "id": "uuid",
  "user_id": "uuid",
  "picture_id": "uuid",
  "created_at": "timestamp",
  "updated_at": "timestamp"
}
```

Get Pictures Uploaded by a Specific User

```
GET /users/:userId/pictures
```

Response (JSON Array):

```
[
  {
    "id": "uuid",
    "URL": "string",
    "caption": "string",
    "created_at": "timestamp"
  }
]
```

Remove a User-Picture Link

```
DELETE /users/:userId/pictures/:pictureId
```

Response (JSON):

```
{
  "id": "uuid",
  "user_id": "uuid",
  "picture_id": "uuid"
}
```

---

5. Error Handling

Errors are returned in the following format:

```
{
  "error": "Error message here"
}
```

- The status code will be set to the appropriate HTTP error code (e.g., 400, 401, 404, 500).

Notes

- The application seeds some initial data upon startup, but that does not affect how you use the endpoints.
- For any endpoint requiring authentication (currently `GET /auth/me`), make sure to include the token in the `Authorization` header.
- `POST /auth/register` and `POST /users` both create new users. The difference is primarily semantic; in the current implementation, both are open and do not require prior authentication.

End of Documentation