Back-End Puzzles: Hints + Solutions

./server/src/flickpics_api/auth.py

- (1) Remember we are using the “User” model. Consider the parameter we must use to retrieve the user.

```python
User.get(document_id=clerk_id)
```

- (2) Consider what part of the authorization header we want to look at here

```python
token
```

./server/src/flickpics_api/main.py

- (1) We want to connect to a MongoDB database using Motor here.

```python
motor.motor_asyncio.AsyncIOMotorClient(CONFIG.mongo_uri)
```

./server/src/flickpics_api/routers/users.py

- (1) We must create a new user with the name and id from the payload. Then, save this user.

```python
new_user = User(name=name, id=user_id)
await new_user.save()
```

- (2) Note: make sure to complete part (1) in this file first! Consider: what do we need to return here?

```python
new_user
```

- (3) Hint: The GET request URI is the user id.

```python
"/{user_id}"
```

- (4) Retrieve the movies where the creator_id is the function parameter.

```python
await Movie.find({"creator_id": user_id}).to_list(length=100)
```

- (5) Hint: This is an error; handle it.

```python
raise HTTPException(
    status.HTTP_404_NOT_FOUND,
    f"Movie lists with creator_id ({user_id}) were not found.",
)
```

./server/src/flickpics_api/routers/movies.py

- (1) Check what parameters fetch_from_tmdb() expects.

```python
request.tmdb_id
```

- (2) How would we retrieve the 10 most recent reviews?

```python
await Movie.all().sort(-Movie.created_at).limit(10).to_list()
```

- (3) Figure out the wrapper for this function. It works with a GET request.

```python
@router.get("/{movie_id}", response_model=Movie)
```

./server/src/flickpics_api/routers/lists.py

- (1) Find out how to get the movie list's name and find if a movie list with that name exists.

```python
await MovieList.find_one({"name": request.name})
```

- (2) Create a new movie list item with the elements in the request.

```python
movie_list = MovieList(
    name=request.name,
    description=request.description,
    movie_ids=request.movie_ids,
    creator_id=user.id,
)
```

- (3) Update list names, descriptions, and movie IDs based on what is provided in function parameters.

```python
if request.name:
    document.name = request.name

  if request.description:
    document.description = request.description

  if request.movie_ids:
    document.movie_ids = request.movie_ids
```

- (4) Get the list with the given ID.

```python
await MovieList.get(list_id)
```