

## Data management ctd.

Validating read\_user(s)

# UserOut model

- will include the auto-generated `_id`
- In most cases, we will want to exclude password from UserOut

```
class UserOut(BaseModel):  
    _id:str  
    # password: str  
    username:str
```

# Validating DB output

- What do you notice about the output?

```
# read and validate
uid = x.inserted_id
ud = mycol.find_one({'_id':uid})
u = UserOut(**ud)
print(u)
```

# Using an alias

- recall that mongo requires a unique `_id` field
- however, pydantic ignores fields with a leading underscore
- one way to address this is to use an alias

```
# user_models.py
from pydantic import Field
# [...]
class UserOut(BaseModel):
    id: str = Field(alias='_id')
    # password: str
    username: str
```

## read\_users: validating queries

- We should validate queries before read\_users
- For now, we just support query by username

```
# in user_models.py
class UserQuery(BaseModel):
    username: str
```

## read\_users: validating collections

- When we get the results back, we need to validate
- We can validate all at once by using a UserCollection model

```
class UserCollection(BaseModel):  
    users: list[UserOut]
```

## reading users with query

- We still want to support default read all users

```
# create some users before this
q = UserQuery(username='joe')
docs = mycol.find(q.model_dump())
uc = UserCollection(users=docs)
```

# Updating UserManager

```
class UserManager:
    # [...]
    def read_user(self, user_id:str) -> UserOut | None:
        '''read user and return'''

    def read_users(self, query:UserQuery=None) -> UserCollection:
        ''' read users by query, or read all users, and return'''
```



# Testing

- The tests will be almost the same, except for:
  - creating appropriate objects for method calls
  - dealing with returned objects

```
class TestUserManager(unittest.TestCase):  
    # [...]  
    def test_basic(self):  
        ''' reset, create user, read user '''  
  
    def test_reads(self):  
        ''' reset, create users, read users '''
```

# Activity

- Complete implementation and testing for create and read.
- Do the same for update and delete.