**User Management API Documentation**

**Overview**

The **User Management API** is a RESTful API built with **FastAPI** and **SQLite**. It provides basic functionality to manage users, including creating, retrieving, and listing users. The API accepts JSON inputs and returns data in JSON format.

**Features**

* **Create User**: Add a new user to the system.
* **Get Users**: Retrieve a list of all users.
* **Swagger UI**: Interactive API documentation to test the API.

**Requirements**

Before setting up the API, make sure you have the following installed:

* **Python** 3.8+
* **Pip** for installing Python packages

**Dependencies**

Ensure all dependencies are installed via requirements.txt:

fastapi

uvicorn

sqlalchemy

pydantic

email-validator

pytest

To install dependencies, run:

pip install -r requirements.txt

**Project Structure**

my\_api\_project/

│

├── app/

│ ├── \_\_init\_\_.py

│ ├── main.py # Main application file with FastAPI routes

│ ├── models.py # Database models

│ ├── database.py # Database connection and session management

│ ├── routes/

│ │ ├── \_\_init\_\_.py

│ │ ├── user\_routes.py # Routes for user management (e.g., create, get users)

│ │

│ └── schemas.py # Pydantic models for request/response validation

│

├── tests/

│ ├── test\_main.py # Unit tests for the application

│

├── requirements.txt # Python dependencies

└── README.md # Documentation file (this one)

**Setup and Installation**

1. **Clone the repository** (if needed):

git clone <repository-url>

cd my\_api\_project

1. **Create a virtual environment** and activate it:

python -m venv venv

# On Windows:

venv\Scripts\activate

for deactivate use **deactivate**

1. **Install dependencies**:

pip install -r requirements.txt

1. **Run the application**:

uvicorn app.main:app --reload

The application will be running on http://127.0.0.1:8000.

**API Endpoints**

**1. GET / - Root Endpoint**

* **Description**: Returns a welcome message.
* **Response**:

{

"message": "Welcome to the User Management API!"

}

**2. POST /users/ - Create User**

* **Description**: Creates a new user with the provided data.
* **Request Body** (JSON):

{

"name": "John Doe",

"email": "john.doe@example.com"

}

* **Response** (JSON):

{

"id": 1,

"name": "John Doe",

"email": "john.doe@example.com"

}

**3. GET /users/ - Get All Users**

* **Description**: Retrieves a list of all users in the system.
* **Response** (JSON):

[

{

"id": 1,

"name": "John Doe",

"email": "john.doe@example.com"

},

{

"id": 2,

"name": "Jane Smith",

"email": "jane.smith@example.com"

}

]

**Testing**

Unit tests are written using **pytest**. To run the tests, use the following command:

pytest

The tests are located in the tests/ directory.

**Running the Application**

* After running uvicorn app.main:app --reload, you can access the interactive API documentation at:
  + **Swagger UI**: http://127.0.0.1:8000/docs
  + **ReDoc**: http://127.0.0.1:8000/redoc

**Database**

The application uses an SQLite database to store user data. The database is created automatically when the application runs for the first time.

* **Database Model** (SQLAlchemy):
  + User: Contains id, name, and email columns.

**Common Errors**

* **ModuleNotFoundError: No module named 'email\_validator'**: If you encounter this error, run the following command:

pip install email-validator

* **Invalid requirement error in requirements.txt**: Update your requirements.txt by explicitly adding email-validator and remove the extra pydantic[email]:

plaintext

Copy code

fastapi

uvicorn

sqlalchemy

pydantic

email-validator

pytest

**Usage Examples**

**Using Swagger UI**

1. Go to http://127.0.0.1:8000/docs.
2. Use the interactive interface to test the POST /users/ and GET /users/ endpoints.

**Using Postman**

1. Create a POST request to http://127.0.0.1:8000/users/ with the body:

{

"name": "Alice",

"email": "alice@example.com"

}

1. Create a GET request to http://127.0.0.1:8000/users/ to fetch all users.

**Testing the API**

**1. Writing Tests**

Test files are located in the tests/ folder. The provided test file test\_main.py includes tests for the root endpoint.

**2. Running Tests**

1. Ensure you are in the root directory (D:\Python\API\_Project).
2. Activate the virtual environment.
3. Run tests using pytest:

Pytest  
if doest work set “set PYTHONPATH=D:\Python\API\_Project” or \app

**Example:**

=========================================================================== test session starts ===========================================================================

platform win32 -- Python 3.13.1, pytest-6.2.5, py-1.11.0, pluggy-1.5.0

rootdir: D:\Python\API\_Project

plugins: anyio-4.7.0, pythonpath-0.7.4

collected 2 items

tests\test\_main.py .. [100%]

============================================================================ warnings summary =============================================================================

venv\Lib\site-packages\\_pytest\assertion\rewrite.py:958: 4 warnings

tests/test\_main.py: 11 warnings

D:\Python\API\_Project\venv\Lib\site-packages\\_pytest\assertion\rewrite.py:958: DeprecationWarning: ast.Str is deprecated and will be removed in Python 3.14; use ast.Constant instead

inlocs = ast.Compare(ast.Str(name.id), [ast.In()], [locs])

venv\Lib\site-packages\\_pytest\assertion\rewrite.py:961: 3 warnings

tests/test\_main.py: 11 warnings

D:\Python\API\_Project\venv\Lib\site-packages\\_pytest\assertion\rewrite.py:961: DeprecationWarning: ast.Str is deprecated and will be removed in Python 3.14; use ast.Constant instead

expr = ast.IfExp(test, self.display(name), ast.Str(name.id))

venv\Lib\site-packages\\_pytest\assertion\rewrite.py:1071: 5 warnings

tests/test\_main.py: 7 warnings

D:\Python\API\_Project\venv\Lib\site-packages\\_pytest\assertion\rewrite.py:1071: DeprecationWarning: ast.Str is deprecated and will be removed in Python 3.14; use ast.Constant instead

syms.append(ast.Str(sym))

venv\Lib\site-packages\\_pytest\assertion\rewrite.py:1073: 5 warnings

tests/test\_main.py: 7 warnings

D:\Python\API\_Project\venv\Lib\site-packages\\_pytest\assertion\rewrite.py:1073: DeprecationWarning: ast.Str is deprecated and will be removed in Python 3.14; use ast.Constant instead

expls.append(ast.Str(expl))

venv\Lib\site-packages\\_pytest\assertion\rewrite.py:823: 19 warnings

tests/test\_main.py: 33 warnings

D:\Python\API\_Project\venv\Lib\site-packages\\_pytest\assertion\rewrite.py:823: DeprecationWarning: ast.Str is deprecated and will be removed in Python 3.14; use ast.Constant instead

keys = [ast.Str(key) for key in current.keys()]

venv\Lib\site-packages\\_pytest\assertion\rewrite.py:933: 5 warnings

tests/test\_main.py: 7 warnings

D:\Python\API\_Project\venv\Lib\site-packages\\_pytest\assertion\rewrite.py:933: DeprecationWarning: ast.Str is deprecated and will be removed in Python 3.14; use ast.Constant instead

assertmsg = ast.Str("")

venv\Lib\site-packages\\_pytest\assertion\rewrite.py:935: 5 warnings

tests/test\_main.py: 8 warnings

D:\Python\API\_Project\venv\Lib\site-packages\\_pytest\assertion\rewrite.py:935: DeprecationWarning: ast.Str is deprecated and will be removed in Python 3.14; use ast.Constant instead

template = ast.BinOp(assertmsg, ast.Add(), ast.Str(explanation))

venv\Lib\site-packages\\_pytest\assertion\rewrite.py:947: 6 warnings

tests/test\_main.py: 8 warnings

D:\Python\API\_Project\venv\Lib\site-packages\\_pytest\assertion\rewrite.py:947: DeprecationWarning: ast.NameConstant is deprecated and will be removed in Python 3.14; use

ast.Constant instead

clear = ast.Assign(variables, ast.NameConstant(None))

app\database.py:9

D:\Python\API\_Project\app\database.py:9: MovedIn20Warning: The ``declarative\_base()`` function is now available as sqlalchemy.orm.declarative\_base(). (deprecated since: 2.0) (Background on SQLAlchemy 2.0 at: https://sqlalche.me/e/b8d9)

Base = declarative\_base()

venv\Lib\site-packages\pydantic\\_internal\\_config.py:295

D:\Python\API\_Project\venv\Lib\site-packages\pydantic\\_internal\\_config.py:295: PydanticDeprecatedSince20: Support for class-based `config` is deprecated, use ConfigDict

instead. Deprecated in Pydantic V2.0 to be removed in V3.0. See Pydantic V2 Migration Guide at https://errors.pydantic.dev/2.10/migration/

warnings.warn(DEPRECATION\_MESSAGE, DeprecationWarning)

venv\Lib\site-packages\pydantic\\_internal\\_config.py:345

D:\Python\API\_Project\venv\Lib\site-packages\pydantic\\_internal\\_config.py:345: UserWarning: Valid config keys have changed in V2:

\* 'orm\_mode' has been renamed to 'from\_attributes'

warnings.warn(message, UserWarning)

app\main.py:10

D:\Python\API\_Project\app\main.py:10: DeprecationWarning:

on\_event is deprecated, use lifespan event handlers instead.

Read more about it in the

[FastAPI docs for Lifespan Events](https://fastapi.tiangolo.com/advanced/events/).

@app.on\_event("startup")

venv\Lib\site-packages\fastapi\applications.py:4495

D:\Python\API\_Project\venv\Lib\site-packages\fastapi\applications.py:4495: DeprecationWarning:

on\_event is deprecated, use lifespan event handlers instead.

Read more about it in the

[FastAPI docs for Lifespan Events](https://fastapi.tiangolo.com/advanced/events/).

return self.router.on\_event(event\_type)

-- Docs: https://docs.pytest.org/en/stable/warnings.html

===================================================================== 2 passed, 149 warnings in 7.72s =====================================================================

**Conclusion**

This API provides a simple, extensible foundation for managing users. It is built with FastAPI and SQLite, making it suitable for small to medium-sized projects. You can extend the functionality by adding more endpoints (e.g., for updating or deleting users) as needed.

**Fllow Of Programm**

* uvicorn app.main:app --reload
* <http://127.0.0.1:8000/docs>
* [127.0.0.1:8000/users/](http://127.0.0.1:8000/users/)