**Python Homework**

**Use given `User` structure and implement below APIs through FastAPI.**

※You can add log you need in your functions.

|  |
| --- |
| {“user\_id”:{  “name”: “{user\_name}”,  “email”:”{user\_email}”  },  }  user = {  1:{"name":"Wanyu\_Lee",  "email":"Wanyu\_Lee@pegatroncorp.com"},  2:{"name":"Sandy1\_Zeng",  "email":"Sandy1\_Zeng@pegatroncorp.com"}  } |

1. **Get User Information**

|  |  |
| --- | --- |
| **URL** | */user/{user\_id}* |
| **Method** | GET |
| **URL Params:** | **N/A** |
| **Success Response:** | **Status Code** : 200 **Content :**  {  “name”: “{user\_name}”,  “email”:”{user\_email}”  } |
| **Error Response:** | **Scenario:** User didn’t input user\_id **Status Code: 400**  **Content:** {“msg”:”Please input user\_id”}  **Scenario:** User not exist  **Status Code: 404**  **Content:** {“msg”:”User not exist.”} |
| **Sample Call** | curl -X GET /user/1  **Response :**  {"name":"Wanyu\_Lee",  "email":”Wanyu\_Lee@pegatroncorp.com”  } |

1. **Add New User**

|  |  |
| --- | --- |
| **URL** | */user* |
| **Method** | POST |
| **Content Type** | application/json |
| **Data Parameter** | |  |  |  |  | | --- | --- | --- | --- | | **Key** | **Type** | **Description** | **Required** | | name | string | User name | Y | | email | string | User email | Y | |
| **Success Response:** | **Status Code** : 201 **Content :**  {“msg”:” Add new user succeed.”,  “user\_id”:”{user\_id}”  } |
| **Error Response:** | **Scenario:** User didn’t input data parameter or wrong data type **Status Code: 400**  **Content:** {“msg”:”Wrong input parameters.”}  **Scenario:** Add failed.  **Status Code: 500**  **Content:** {“msg”:”Add new user failed.”} |
| **Sample Call** | curl -X POST  -d ‘{“name”:”Jason\_Yang”,”email”:”Jason\_Yang@pegatroncorp.com”} ’  -H "Content-Type: application/json"  /user  **Response :**  {“msg”:” Add new user succeed.”, “user\_id”: 3} |

**※Hint: BaseModel**

1. **Update User Data**

|  |  |
| --- | --- |
| **URL** | */user/{user\_id}* |
| **Method** | PUT |
| **Content Type** | application/json |
| **Data Parameter** | |  |  |  |  | | --- | --- | --- | --- | | **Key** | **Type** | **Description** | **Required** | | name | string | User name | N | | email | string | User email | N | |
| **Success Response:** | **Status Code** : 200 **Content :**  {“msg”:” Update user{user\_id}’s data succeed.”} |
| **Error Response:** | **Scenario:** User didn’t input data parameter or wrong data type **Status Code: 400**  **Content:** {“msg”:”Wrong input parameters.”}  **Scenario:** User not exist  **Status Code: 404**  **Content:** {“msg”:”User not exist.”}  **Scenario:** Update failed.  **Status Code: 500**  **Content:** {“msg”:” Update user{user\_id}’s data failed.”} |
| **Sample Call** | curl -X PUT  -d ‘{“name”:”Yuwan\_Lee” } ’  -H "Content-Type: application/json"  /user/1  Response: {“msg”:” Update user1’s data succeed.”} |

1. **Delete User**

|  |  |
| --- | --- |
| **URL** | */user/{user\_id}* |
| **Method** | DELETE |
| **Success Response:** | **Status Code** : 200 **Content :**  {“msg”:” Delete user{user\_id} succeed.”} |
| **Error Response:** | **Scenario:** User didn’t input data parameter or wrong data type **Status Code: 400**  **Content:** {“msg”:”Wrong input parameters.”}  **Scenario:** User not exist  **Status Code: 404**  **Content:** {“msg”:”User not exist.”}  **Scenario:** Delete failed.  **Status Code: 500**  **Content:** {“msg”:” Delete user{user\_id} Failed.”} |
| **Sample Call** | curl -X DELETE /user/1  **Response:**  {“msg”:” Delete user1 succeed.”} |

**Docker Homework**

Dockerized your FastAPI project above.

1. **Write a Dockerfile**
   * **Base image:** python:3-alpine3.9
   * Add all packages you need and FastAPI project above
   * Timezone : Asia/Taipei
   * Export a port not used on testnode05.
2. **Build Image**
   * Image name: ctrdc-dev-dockerhub.pegatroncorp.com/training\_intern/{your OA name}:latest
3. **Push image to Harbor**
   * docker login ctrdc-dev-dockerhub.pegatroncorp.com
   * docker push ctrdc-dev-dockerhub.pegatroncorp.com/training\_intern/{your OA name}:latest
4. **Write a docker-compose file to run your image**
   * Container name: {your name}
   * network\_mode: bridge
   * Expose port
   * Setup cpu and memory limit for your container.