

```
    return {  
        "message": "Event deleted successfully."  
    }
```

Now that we have implemented the CRUD operations to our event routes, let's implement the routes for signing a user up and signing a user in.

routes/users.py

Let's start by updating the imports and creating a database instance:

```
from fastapi import APIRouter, HTTPException, status  
from database.connection import Database  
  
from models.users import User, UserSignIn  
  
user_router = APIRouter(  
    tags=["User"],  
)  
  
user_database = Database(User)
```

Next, update the POST route for signing new users:

```
@user_router.post("/signup")  
async def sign_user_up(user: User) -> dict:  
    user_exist = await User.find_one(User.email ==  
    user.email)  
    if user_exist:  
        raise HTTPException(  
            status_code=status.HTTP_409_CONFLICT,  
            detail="User with email provided exists  
            already."  
        )  
    await user_database.save(user)  
    return {  
        "message": "User created successfully"  
    }
```

In this code block, we check whether such a user with the email passed exists before adding them to the database. Let's add the route to sign users in:

```
@user_router.post("/signin")
async def sign_user_in(user: UserSignIn) -> dict:
    user_exist = await User.find_one(User.email ==
    user.email)
    if not user_exist:
        raise HTTPException(
            status_code=status.HTTP_404_NOT_FOUND,
            detail="User with email does not exist."
        )
    if user_exist.password == user.password:
        return {
            "message": "User signed in successfully."
        }
    raise HTTPException(
        status_code=status.HTTP_401_UNAUTHORIZED,
        detail="Invalid details passed."
    )
```

In this defined route, we first check whether the user exists before checking the validity of their credentials. The method of authentication used here is basic and *not recommended* in production. We'll take a look at proper authentication procedures in the next chapter.

Now that we have implemented the routes, let's start a MongoDB instance as well as our application. Create a folder to house our MongoDB database and start the MongoDB instance:

```
(venv)$ mkdir store
(venv)$ mongod --dbpath store
```

Next, in another window, start the application:

```
(venv)$ python main.py
INFO:      Uvicorn running on http://0.0.0.0:8080 (Press CTRL+C
to quit)
INFO:      Started reloader process [3744] using statreload
INFO:      Started server process [3747]
INFO:      Waiting for application startup.
INFO:      Application startup complete.
```

Let's test the event routes:

1. Create an event:

```
(venv)$ curl -X 'POST' \
  'http://0.0.0.0:8080/event/new' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
    "title": "FastAPI Book Launch",
    "image": "https://linktomyimage.com/image.png",
    "description": "We will be discussing the contents
of the FastAPI book in this event. Ensure to come
with your own copy to win gifts!",
    "tags": [
      "python",
      "fastapi",
      "book",
      "launch"
    ],
    "location": "Google Meet"
  }'
```

Here is the response from the preceding operation:

```
{
  "message": "Event created successfully"
}
```

2. Get all events:

```
(venv)$ curl -X 'GET' \  
  'http://0.0.0.0:8080/event/' \  
  -H 'accept: application/json'
```

The preceding request returns a list of events:

```
[  
  {  
    "_id": "624daab1585059e8a3fa77ac",  
    "title": "FastAPI Book Launch",  
    "image": "https://linktomyimage.com/image.png",  
    "description": "We will be discussing the contents  
of the FastAPI book in this event. Ensure to come  
with your own copy to win gifts!",  
    "tags": [  
      "python",  
      "fastapi",  
      "book",  
      "launch"  
    ],  
    "location": "Google Meet"  
  }  
]
```

3. Get an event:

```
(venv)$ curl -X 'GET' \  
  'http://0.0.0.0:8080/event/624daab1585059e8a3fa77ac' \  
  -H 'accept: application/json'
```

This operation returns the event that matches the supplied ID:

```
{  
  "_id": "624daab1585059e8a3fa77ac",  
  "title": "FastAPI Book Launch",  
  "image": "https://linktomyimage.com/image.png",  
  "description": "We will be discussing the contents  
of the FastAPI book in this event. Ensure to come  
with your own copy to win gifts!",  
}
```

```
"tags": [  
    "python",  
    "fastapi",  
    "book",  
    "launch"  
],  
"location": "Google Meet"  
}
```

4. Let's update the event location to Hybrid:

```
(venv)$ curl -X 'PUT' \  
  'http://0.0.0.0:8080/event/624daab1585059e8a3fa77ac' \  
 \  
 -H 'accept: application/json' \  
 -H 'Content-Type: application/json' \  
 -d '{  
  "location": "Hybrid"  
}'  
  
{  
  "_id": "624daab1585059e8a3fa77ac",  
  "title": "FastAPI Book Launch",  
  "image": "https://linktomyimage.com/image.png",  
  "description": "We will be discussing the contents  
of the FastAPI book in this event. Ensure to come  
with your own copy to win gifts!",  
  "tags": [  
    "python", "fastapi",  
    "book",  
    "launch"  
  ],  
  "location": "Hybrid"  
}
```