3. Let's implement the routes to create an event, delete a single event and delete all events contained in the database:

```
@event_router.post("/new")
async def create_event(body: Event = Body(…)) -> dict:
    events.append(body)
    return {
        "message": "Event created successfully"
    }

@event_router.delete("/{id}")
async def delete_event(id: int) -> dict:
    for event in events:
        if event.id == id:
            events.remove(event)
            return { "message": "Event deleted
            successfully" }
    raise HTTPException(
        status_code=status. HTTP_404_NOT_FOUND,
        detail="Event with supplied ID does not exist"
    )
@event_router.delete("/)
async def delete_all_events() -> dict:
    events.clear()
    return {
        "message": "Events deleted successfully"
    }
```

We have successfully implemented the routes for events. The UPDATE route will be implemented in *Chapter 6*, *Connecting to a Database*, where we'll port our application to use an actual database.

4. Now that we have implemented the routes, let's update our route configuration to include the event route in `main.py`:

```
from fastapi import FastAPI
from routes.user import user_router
from routes.events import event_router
```

```
import uvicorn
app = FastAPI()

# Register routes

app.include_router(user_router, prefix="/user")
app.include_router(event_router, prefix="/event")

if __name__ == "__main__":
    uvicorn.run("main:app", host="0.0.0.0", port=8080,
    reload=True)
```

The application automatically reloads on every change. Let's test the routes:

- The `GET` route – the following operation returns an empty array, telling us that no data is present:

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

  Let's add data to our array next.

- The `POST` route – in the terminal, execute the following command:

```
(venv)$ curl -X 'POST' \
  'http://0.0.0.0:8080/event/new' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
  "id": 1,
  "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:

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

This operation was successful from the response received. Now, let's try to retrieve the specific event we just created:

- The `GET` route:

```
(venv)$ curl -X 'GET' \
  'http://0.0.0.0:8080/event/1' \
  -H 'accept: application/json'
```

Here is the response:

```
{
  "id": 1,
  "title": "FastAPI BookLaunch",
  "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"
}
```

Lastly, let's delete the event to confirm that the event route is working:

- The `DELETE` route – in the terminal, execute the following command:

```
(venv)$ curl -X 'DELETE' \
  'http://0.0.0.0:8080/event/1' \
  -H 'accept: application/json'
```

Here is the response:

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

If I retry the same command, I get the following response:

```
(venv)$ {
  "detail": "Event with supplied ID does not exist"
}
```

We have successfully implemented the routes and models for our planner application. We have also tested them to assess their working status.

Summary

In this chapter, we learned how to structure a FastAPI application, and implement routes, and models for an event-planning application. We made use of the basics of routing and the knowledge of routing and modeling that we learned in an earlier chapter.

In the next chapter, you will be introduced to connecting your application to SQL and NoSQL databases. You will continue building the event planner application by improving the existing application and adding more features. Before that, you will be introduced to what databases are, the different types, and how to use both (SQL and NoSQL) in a FastAPI application.