Day 10
Dharunya s

add an additional table called users. put api post request to insert user into it in fast api. Additionally, when request to list music api is called, only a valid user is allowed to get the response else raise HTTPException with error code

```
from sqlalchemy import create_engine, Column, Integer, String
from sqlalchemy.orm import sessionmaker, Session, declarative_base
from fastapi import FastAPI, HTTPException, Request, Depends
from fastapi.templating import Jinja2Templates
from fastapi.responses import HTMLResponse
from pydantic import BaseModel
import uvicorn

MOVIE_DB_URL = "sqlite:///./movie.db"
engine = create_engine(MOVIE_DB_URL, connect_args={"check_same_thread": False})
SessionLocal = sessionmaker(autocommit=False, autoflush=False, bind=engine)
Base = declarative_base()

class User(Base):
    __tablename__ = "users"
    Id = Column(Integer, primary_key=True, index=True)
    Name = Column(String(50), unique=True, nullable=False)


class Movie(Base):
    __tablename__ = "movie"
    Id = Column(Integer, primary_key=True, index=True)
    SongName = Column(String(50), nullable=False)
    Duration = Column(Integer, nullable=False)


Base.metadata.create_all(bind=engine)

app = FastAPI()
templates = Jinja2Templates(directory="templates")

def get_db():
    db = SessionLocal()
    try:
        yield db
    finally:
        db.close()
```

```
def authenticate(name: str, db: Session):
    user = db.query(User).filter(User.Name == name).first()
    if not user:
        raise HTTPException(status_code=401, detail="User not allowed")
    return user

class UserSchema(BaseModel):
    Name: str
    class Config:
        orm_mode = True

class MovieSchema(BaseModel):
    SongName: str
    Duration: int
    class Config:
        orm_mode = True

class MovieDTO(BaseModel):
    Name: str
    class Config:
        orm_mode = True



@app.post("/add_user")
def add_user(user: UserSchema, db: Session = Depends(get_db)):
    existing = db.query(User).filter(User.Name == user.Name).first()
    if existing:
        raise HTTPException(status_code=400, detail="User already exists")
    new_user = User(id=user.Id,Name=user.Name)
    db.add(new_user)
    db.commit()
    db.refresh(new_user)
    return {"message": f"User {new_user.Name} added successfully"}

@app.post("/add_movie")
def add_song(m: MovieSchema,username: str,db: Session = Depends(get_db)):
    authenticate(username, db)
    mv = Movie(SongName=m.SongName, Duration=m.Duration)
    db.add(mv)
    db.commit()
    db.refresh(mv)
    return mv
```

```
@app.get("/list_html", response_class=HTMLResponse)
async def list_songs_html(
    request: Request,
    username: str,
    db: Session = Depends(get_db)
):
    authenticate(username, db)  # only registered users can view
    recs = db.query(Movie).all()
    return templates.TemplateResponse("songs.html", {"request": request, "movies":
recs})

@app.get("/getBy/{my_id}", response_model=MovieDTO)
async def get_movie_by_id(
    my_id: int,
    username: str,
    db: Session = Depends(get_db)
):
    authenticate(username, db)
    mv = db.query(Movie).filter_by(Id=my_id).first()
    if not mv:
        raise HTTPException(status_code=404, detail="Movie not found")
    return MovieDTO(Name=mv.SongName)

@app.middleware("http")
async def addmiddleware(request: Request, call_next):
    print("Middleware intercepted the call!")
    response = await call_next(request)
    return response

if __name__ == "__main__":
    uvicorn.run("day10:app", host="127.0.0.1", port=8000, reload=True)
```

username * required
string
(query)

Ramu

Request body required

application/json

Edit Value | Schema

```
{
  "SongName": "string",
  "Duration": 0
}
```

Execute | Clear

Responses

Curl

```
curl -X 'POST' \
  'http://127.0.0.1:8000/add_movie?username=Ramu' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
  "SongName": "string",
  "Duration": 0
}'
```

Request URL

```
http://127.0.0.1:8000/add_movie?username=Ramu
```

Server response

Code Details

Request URL

```
http://127.0.0.1:8000/add_movie?username=Ramu
```

Server response

Code	Details
200	Response body

```
{
  "Id": 12,
  "SongName": "string",
  "Duration": 0
}
```

Download

Response headers

```
content-length: 42
content-type: application/json
date: Tue,19 Aug 2025 12:26:13 GMT
server: uvicorn
```

Responses

Code	Description	Links
200	Successful Response	*No links*

Media type

application/json

Controls Accept header.

Example Value | Schema

```
"string"
```

422	Validation Error	*No links*

Media type

application/json

Example Value | Schema

```
{
  "detail": [
    {
      "loc": [
        "string",
        0
      ],
      "msg": "string",
      "type": "string"
    }
  ]
}
```

username * required

string

(query)

Ramu

Execute | Clear

Responses

Curl

```
curl -X 'GET' \
  'http://127.0.0.1:8000/list_html?username=Ramu' \
  -H 'accept: text/html'
```

Request URL

```
http://127.0.0.1:8000/list_html?username=Ramu
```

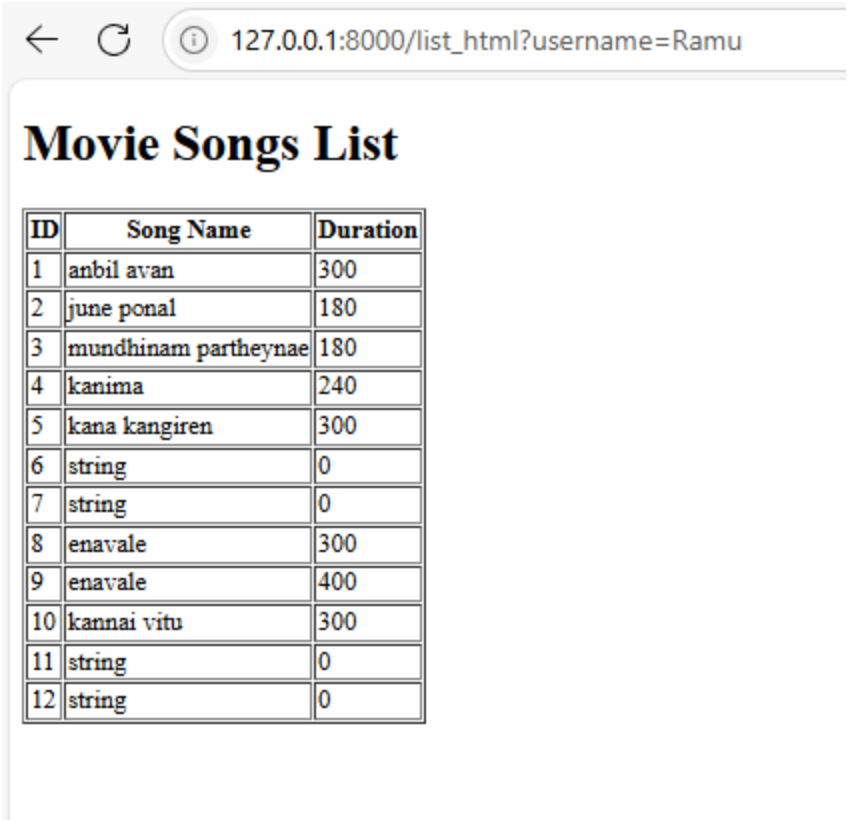
Server response

Code	Details
200	Response body

```
<!DOCTYPE html>
<html>
<head>
    <title>Movie Songs</title>
</head>
<body>
    <h1>Movie Songs List</h1>
    <table border="1">
        <tr>
            <th>ID</th>
            <th>Song Name</th>
            <th>Duration</th>
        </tr>

        <tr>
            <td>1</td>
            <td>anbil avan</td>
            <td>300</td>
        </tr>

        <tr>
            <td>2</td>
            <td>june ponal</td>
            <td>180</td>
        </tr>

        <tr>
            <td>3</td>
            <td>mundhinam partheynae</td>
```

Download

Response headers


127.0.0.1:8000/list_html?username=Ramu
Movie Songs List
ID
Song Name
Duration
1 anbil avan 300
2 june ponal 180
3 mundhinam partheynae 180
4 kanima 240
5 kana kangiren 300
6 string 0
7 string 0
8 enavale 300
9 enavale 400
10 kannai vitu 300
11 string 0
12 string 0