Activity 3.1 - Writing The Main Application Logic In main.py

Activity 3.1: Writing The Main Application Logic In main.py

Objective:

This task involves integrating all the core modules and APIs into a single entry point (`main.py`) for the SmartSDLC application. The goal is to ensure the entire application flows properly when executed.

Structure of main.py:

The file acts as the central orchestrator of the application, performing the following actions:

1. Importing necessary modules and libraries.

2. Initializing FastAPI instance.

3. Including routers from modular files.

4. Handling middleware, CORS, or custom logic.

Code Example - main.py

```python
from fastapi import FastAPI

from routers import user_router, task_router

app = FastAPI(title="SmartSDLC App")

# Include routers
```

Activity 3.1 - Writing The Main Application Logic In main.py

```
app.include_router(user_router)

app.include_router(task_router)


@app.get("/")

def root():

    return {"message": "Welcome to SmartSDLC!"}


if __name__ == "__main__":

    import uvicorn

    uvicorn.run(app, host="0.0.0.0", port=8000)
```

Additional Features:

- Middleware for logging API calls.

- Error handling logic.

- Dependency injections (e.g., for DB sessions).

Diagram (Architecture Overview):

[See diagram below]

Conclusion:

The `main.py` file serves as the backbone of the SmartSDLC app, enabling clean module interaction and

Activity 3.1 - Writing The Main Application Logic In main.py

setting the foundation for deployment-ready architecture.

Challenges Faced:

- Managing dependencies between routers.

- Ensuring consistent error responses.

- Testing with tools like Postman.

```bash

# Run the app locally:

uvicorn main:app --reload

```