Laboratory Activity – Software Architecture

Part A: System Architecture Design

Chosen Architecture Pattern: Layered Architecture (Presentation–Service–Data)

This pattern separates the system into logical layers to improve maintainability, scalability, and separation of concerns.

Architecture Description

1. Presentation Layer

- **Provides the user interface of the system.**
- **Accepts user input and displays output.**
- **Sends user requests to the Service Layer.**

2. Service Layer

- **Contains the business logic of the system.**
- **Processes user requests received from the Presentation Layer.**
- **Validates data before sending it to the Data Layer.**

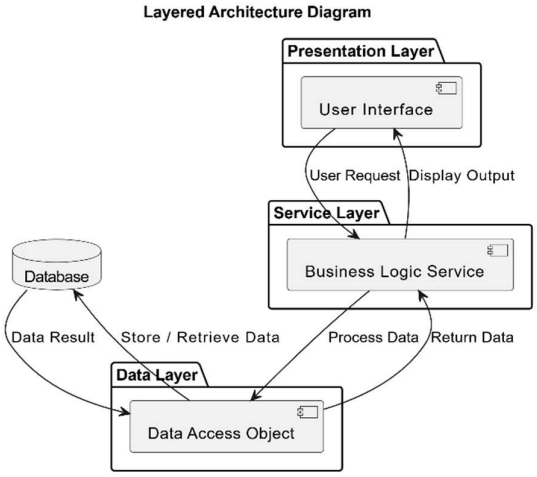
3. Data Layer

- **Handles data access logic.**
- **Communicates with the database to store or retrieve data.**
- **Returns processed data to the Service Layer.**

4. Database

- **Stores all persistent system data.**
- **Responds to data queries from the Data Layer.**

PlantUML Architecture Diagram (for draw.io)



Flow Explanation

- **User → UI:** User sends a request.
- **UI → Service Layer:** UI forwards the request for processing.
- **Service → Data Layer:** Service Layer asks DAO to interact with the database.
- **Data Layer → Database:** DAO executes database operations.
- **Database → Data Layer:** Database sends data results back to DAO.
- **Data Layer → Service Layer:** DAO returns processed data.
- **Service Layer → UI:** Business Logic Service prepares final output and displays it to the user.

PART B: Project Setup Procedure

Project Name Example: Student Management System

Folder Structure

AGD_StudentAttendanceSystem/ │ ├─ config/ │ └─ DIConfig.java │ ├─ docs/ │ └─ (UML diagrams, reports) │ ├─ src/ │ ├─ presentation/ │ │ ├─ controllers/ │ │ └─ views/ │ │ │ ├─ service/ │ │ └─ AttendanceService.java │ │ └─ data/ # Data access layer │ └─ AttendanceDAO.java │ ├─ test/ │ └─ README.md

Folder Descriptions

Folder	Purpose
src/presentation/views	User Interface pages for attendance management.
src/presentation/controllers	Controllers that handle requests from UI and call Service Layer.
src/service	Business logic: calculating attendance, reporting, validation.
src/data	DAO: handles database operations like storing/retrieving attendance records.
config	Dependency injection setup and other configurations.
test	Unit and integration tests for service and data layers.
docs	UML diagrams, flowcharts, and report files.
README.md	Overview, setup, and instructions for running the project.

Dependency Injection (DI) allows you to decouple layers, making it easier to test and maintain.

Benefit

Simplicity

Performance

Easier Testing

Centralized Management

Faster Development

Monolithic Architecture Used

This laboratory of **Student Attendance Management System** is implemented using a **Monolithic Architecture**, where all system components are developed, deployed, and managed as a **single unified application**.

Github Repository Submission

Create Repository

1. Name: AGD_`StudentAttendanceSystem`
2. `Initialized Git Repository`
   - gh repo clone Dadacay/AGD_StudentAttendanceSystem
3. `Added base folder Structure`
   - src/, config/, test/, docs/
4. `Pushed project to Github`

   git add .

   git commit -m "Initial project structure and implementation"

   git push origin main

Sample Github Link Format

https://github.com/Dadacay/AGD_StudentAttendanceSystem.git