Architectural Patterns & Semester Project Implementation

Task1：

1. Client—Server Architecture

   Client—Server Architecture is a client sends a request to a server, and the server processes the request and returns a response. It only has two layers, one is the front—end(Client) and the other is the back—end(Server). The back—end server will handle all data operations and business logic.

2. Multi—tier Client—Server Architecture

   Multi—tier Client—Server Architecture adds more intermediate layers on the Client—Server Architecture. It separating different services such as business logic and data storage. It usually is a three—tier architecture (three—tier architecture：Presentation layer, Application layer, Data layer).

3. Service—Oriented Architecture

   Service Oriented Architecture divides business functions into multiple independent services. Each service can be used by multiple clients.

Task2：

Advantages：

Client—Server：Simple structure, easy to implement. Centralized data management on servers with low cost

Multi—tier Client—Server：Improve scalability through layering. Better security because of separation of data layer and business layer. Strong maintainability,

each layer can be independently modified.

Service—Oriented: High modularity and reusable services. Can quickly adapt to changes in request numbers.

Disadvantages:

Client—Server: Servers can easily become performance bottlenecks. Limited scalability, difficult to support large—scale users.

Multi—tier Client—Server: Complex structure, high development and deployment costs. Additional server resources are required

Service—Oriented: Complex service governance is required. Remote calling increases communication overhead. Requires the highest development and maintain costs.

Use cases:

Client—Server: Small applications (such as LAN file sharing).

Multi—tier Client—Server: Large scale web applications (such as banking systems, e—commerce).

Service—Oriented: Enterprise level system integration (such as hospital information management system).

Task3:

We think that the most suitable architecture for our semester project Bill Buddy is the Multi—tier Client Server Architecture.

Reason：

1.Support real—time data updates

Bill Buddy needs to automatically calculate fees and display users' balances in real time, and a multi—layer architecture can ensure that the front—end directly obtains the latest data through APIs.

2.Enhance system maintainability

The business logic layer is responsible for automatic calculations, while the database layer is responsible for storing and querying data. This hierarchical approach helps to expand or modify features in the future.

3.Improve security

Due to the involvement of user identity authentication and financial data, this architecture allows us to perform identity verification at the business logic layer, avoiding direct access to the database by the front—end.

Task4：

Key Feature：automated cost calculation and real—time synchronization

Bill Buddy needs to automatically calculate each person's shared amount and update the balance in real—time when users add bills.

Implementation：

Front end (React)：

Send a request to add a new bill and display the current user's balance changes.

Business logic layer (Spring Boot)：

Process cost allocation logic, calculate the payable amount for each user, store it in the database, and trigger notification function.

Database layer (MySQL):

Store user billing and balance data, and support transaction management to ensure data consistency.