Architecture Styles

For

Hotel Management System

Table of Contents:

List Of Figures

Introduction:

Software architecture is constructing of high level structures of a software system. These structures are often complex as the size of the project increases. As the size and complexity of software systems increases, the design problem goes beyond the algorithms and data structures of the computation. In such cases Designing and specifying the overall system structure will be a new problem. There is no well-defined terminology or notation to characterize architectural structures, But a well defined software developers uses architectural styles to form a basis for the project. Architectural style sometimes also called as architectural pattern or design pattern is like a framework for an abstract class of similar systems. It provides solutions to frequently occurring problems and supports re usability. More specifically, an architectural style determines the vocabulary of components and connectors that can be used in instances of that style, together with a set of constraints on how they can be combined. They provide a common language for the developing team to understand and interact and ensure they are on the same boat. The architecture of a software system is almost never limited to a single architectural style. For larger system the architectural styles can be combined based on the needs. And most of the systems now need the combination.A combination of architecture styles is also useful especially when building a Web application for our Online Hotel Management System, where we can achieve effective separation of concerns by using the layered architecture style, centralized control through client server architecture and components access which modify data at a centralized data store through database centric style.

We chose 3 Architectural Styles combination models, that would be suitable to develop our Online Hotel Management System. They are:
1. Client Server Style
2. Layered Style
3. Database Centric Style

1) Client Server Style

Segregates the system into two applications, where the client makes requests to the server. In many cases, the server is a database with application logic represented as stored procedures. A server provides services to multiple instances of client systems. Client and servers are typically connected through a network. A request is usually made from client to a server and requests the service. There can be more than one servers to handle large systems. The architecture used here is a 3-tier client-server architecture. The third tier is the data tier where information is stored and retrieved.

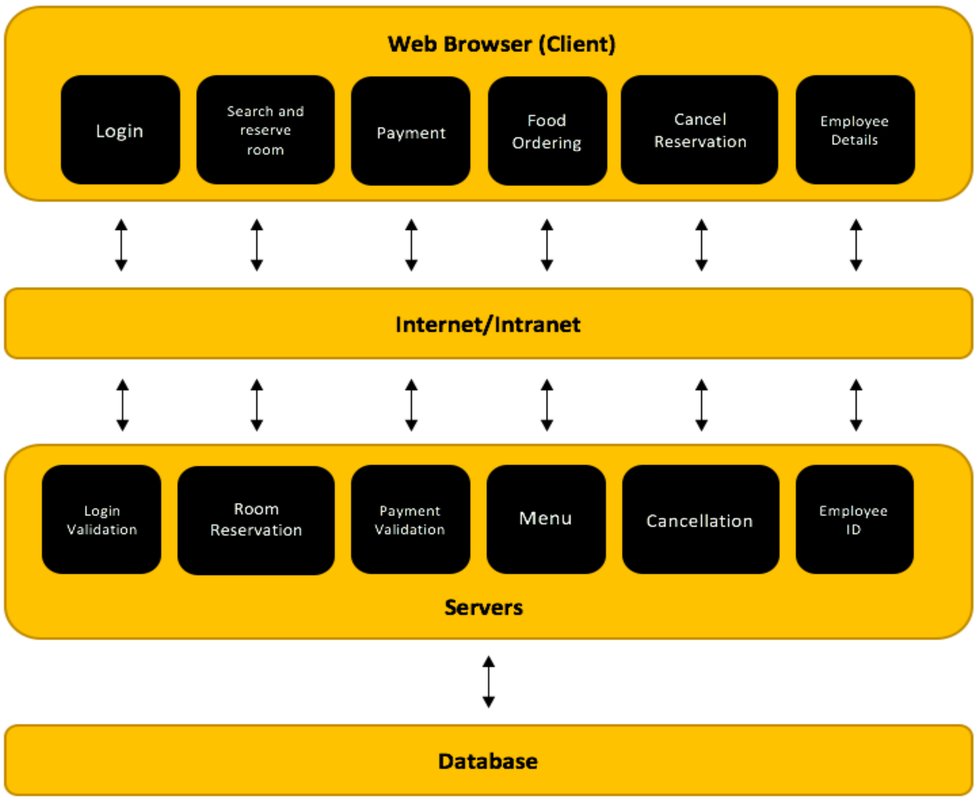


Fig1: Client-Server Architecture for Online Hotel Management System

Client : Web Browser / Mobile Application

The above figure explains the different client system that we have in the web portal i.e Login portal, Room reservation portal, Payment portal, Food ordering portal, Reservation cancellations and employee details portals in the Hotel Management System. The Actors – Owner, Manager, Receptionist, Staff, Chef, Developer and the Customer access the web browser (client) to retrieve the information they required with the valid login credentials. Client requests these services through the network using internet or intranet for server services. Server replies back with these services accessing information from the database.

<u>Reason why we chose Client Server Architecture style is:</u>

This 3-tier client/server architecture is used when

1. An effective distributed client/server design is needed that provides increased performance, flexibility, maintainability, reusability and scalability.
2. Application that needs many services
3. The application is programmed in different languages.
4. When the application being used is for longer periods of more than 3 years.
5. If there would be more than 50,000 transactions per day or more than 300 concurrent users on the same system accessing the same database.

Advantages of using Client Server Architectural Style:

1. Improved data sharing mechanism through servers. Transparency in network services depict that similar data is being shared among users.
2. Applications used for client/server model is built regardless of the hardware platform or technical background of the entitled software providing an open computing environment.
3. It offers centralized control. Servers help in administering the whole set-up.
4. Recovery from damage is easy and storage of backups is not complex.
5. It’s easy to replace, repair, upgrade and relocate a server while clients remain unaffected.
6. Servers have better control access and resources to ensure that only authorized clients can access or manipulate data

Such advantages of the client/server based architecture style makes it more usable and this architecture style is essential for a web based portal for the Hotel Management System.

Disadvantages of using Client Server Architectural Style:

1. When there are frequent simultaneous client requests, servers severely get overloaded, forming traffic congestion.
2. Since its centralized if a critical server fails, client requests are not accomplished.
3. It is expensive to install and might need professionals to manage.

2) Layered Style

Layered architecture provides high security such that no one can penetrate into the system, so we feel it as a good fit for our Hotel Management System as one of the architectural patterns. This architecture organizes the system into layers. The functionality within each layer is related by a common role or responsibility. A layer provides services to the layer above it so the lowest-level layers represent core services that are likely to be used throughout the system.

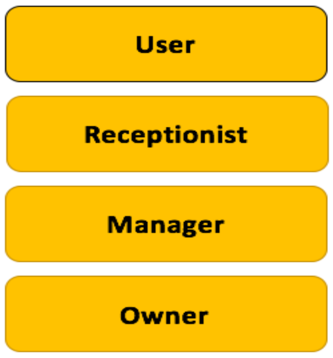


Fig2: Layered architecture for the different actors

Owner: (core actor)
The Owner is the complete holder of the hotel who has all rights to manage all actors and can review all reports related to the organization.

Manager:
The manager has authority to manage the staff and receptionist. He can add new features and also access reports of the hotel system except those related to finance and income.

Receptionist:
The receptionist is responsible for customer reservations and has the least access to the system functions.

Customer:
The customer is the top level actor who utilizes the services provided by the lower layers.

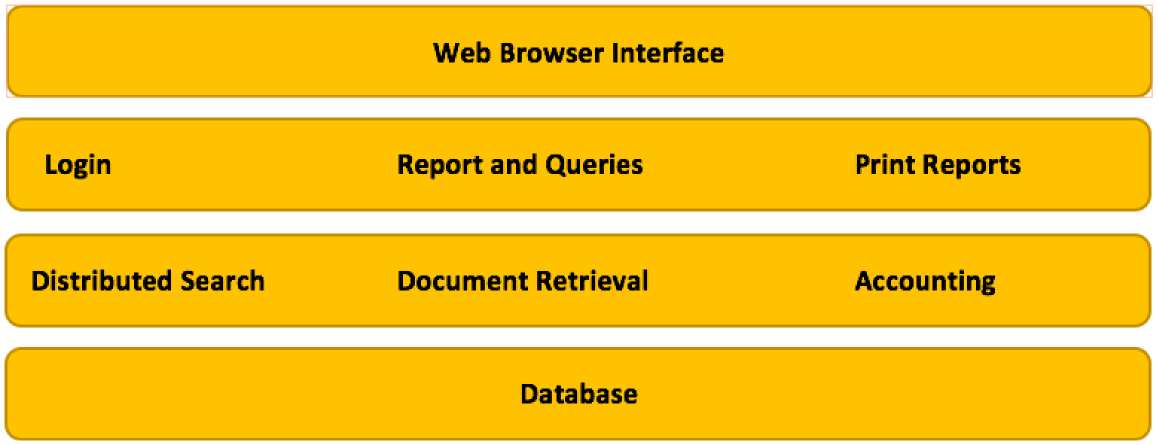


Fig3: Layered Architecture for Hotel Activities On Web Browser Interface/ Mobile App Interface

Layer1
Web Browser Interface / Mobile App Interface is the top layer which is accessed by the Customer, Receptionist, Manager, Owner to access any information and to perform any activity. This layer utilizes the services provided in the next layer.

Layer2

The second layer consists of the “Login”, “Reports & Queries”, and “Print Reports” components. All the actors login through the Web Browser Interface or Mobile Interface to access any Reports & Queries or to print them with the permissions levels they are assigned with.

<u>Layer3</u>
The third layer consists of Distributed Search, Document Retrieval and Accounting components.
The users can use the search functionality to find different types of documents. The receptionist can retrieve the documents related to the customer booking, the manager can retrieve the nonfinancial reports, and the owner can retrieve the reports related to staff, financial and nonfinancial by logging in. They can perform accounting on the reports.

<u>Layer4</u>
A database is the lowest layer which offers services to the higher layers. All kinds of information requested in the above layers is obtained through the database.

Layering is the ability to separate key functions into different logical locations where they can be executed, managed and changed with relative independence. So we selected Layer Style for our project. Unlike Pipe, Center or the Repository Model it has following advantages which makes our will strong to choose Layered style for online HMS.

1. **Abstraction**: It abstracts the view of a system as a whole while providing enough detail to understand the roles and responsibilities of individual layers and the relationship between them.
2. **Encapsulation**: No assumptions need to be made about data types, methods and properties, or implementation during design, as these features are not exposed at layer boundaries.
3. **Independence**: Independence between layers allows each layer to evolve independently.
4. **High cohesion**. Well-defined responsibility boundaries for each layer, and ensuring that each layer contains functionality directly related to the tasks of that layer, will help to maximize cohesion within the layer.
5. **Reusable**. Lower layers have no dependencies on higher layers, potentially allowing them to be reusable in other scenarios.
6. **Easy replacement:** Ability to replace one or several layer implementation with minimum effort and side effects.
7. **Clearly defined functional layers**. The separation between functionality in each layer is clear. Layers can also allow data to flow both up and down between the layers.
8. **Loose coupling**. Communication between layers is based on abstraction and events to provide loose coupling between layers.

<u>Disadvantages</u>:

1. Performance degrades if we have too many layers because it requires extra overhead to pass through these layers which slows down the process.
2. Difficult to efficiently assign functionalities to the right layer.
3. Can’t be used for simple applications because it adds complexity.
4. Development of user-intensive applications can sometime take longer if the layering prevents the use of user interface components that directly interact with the database.

3) Data-Centred (Repository) Style

Another architectural style or pattern that will be compatible with our online hotel management system is the database centric style.In this architecture, data is centralized and accessed frequently by other components which modify data. The three protocols that are required for these kind of architectural styles are communication protocol, data definition and data manipulation protocol. This architectural style will be well suited for the web applications that deal with large amount of data that will be manipulated from different access points. The means of communication between the components distinguishes the subtypes of the data centered architectural style:

Repository (Passive Data Store): A client sends a request to the system to perform a necessary action.

Blackboard (Active Data Store): The system sends notification and data to the subscribers when data of interest changes and is thus active.

The following diagram gives a brief syntax of the data centric architecture style:

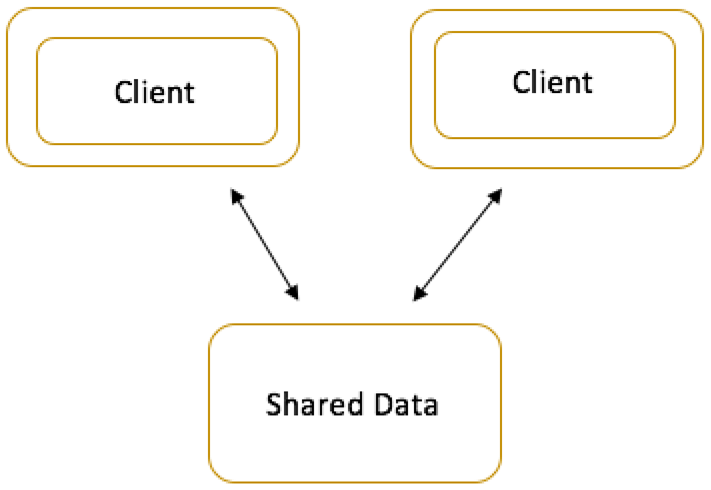
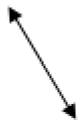
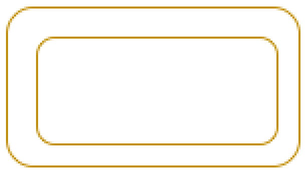


Fig4: General Syntax of Database Centric Architectural Style

The elements of the syntax conveys the following meaning:



-- Indicates the data flow



-- Double lined oval indicates the computational component or object

-- Oval indicates the passive data

In this architecture style clients are quite independent of each other and new clients can be added easily and be modified without affecting others.

Repository Architectural Style: In this style the database is passive and the clients of the data store are active. The clients control the logic flow. The components which participate check the data store for changes.

Blackboard Architectural Style: In this style the data store is active and the clients that interact with the data store remains passive. The logical flow is determined by the current data status in the data store. A blackboard component is used as a central data repository and for internal presentation and acted upon by the computational elements. The components in this style interact only through the blackboard component. The data store in this style alerts the clients whenever there is a change in the data store. Best example for this is WVU ecampus.

Therefore repository style subtype of database centric architectural style suits well for our online hotel management system as active clients involved in the process access data established in a centralized position. This architectural style is unique and used as data-store in web applications.

Advantages:

- The repository architectural style provides data integrity, backup and restore features.
- It reduces over the head of transient data between software components.
- It provides scalability, reusability of agents as they do not provide direct communication with each other.
- Blackboard Model provides concurrency that allows all knowledge sources to work in parallel as they independent of each other.

Disadvantages:

- There is a high dependency between data structure of the data store and its agents.
- Even a small change in data structure may affect its clients.
- Expensive.

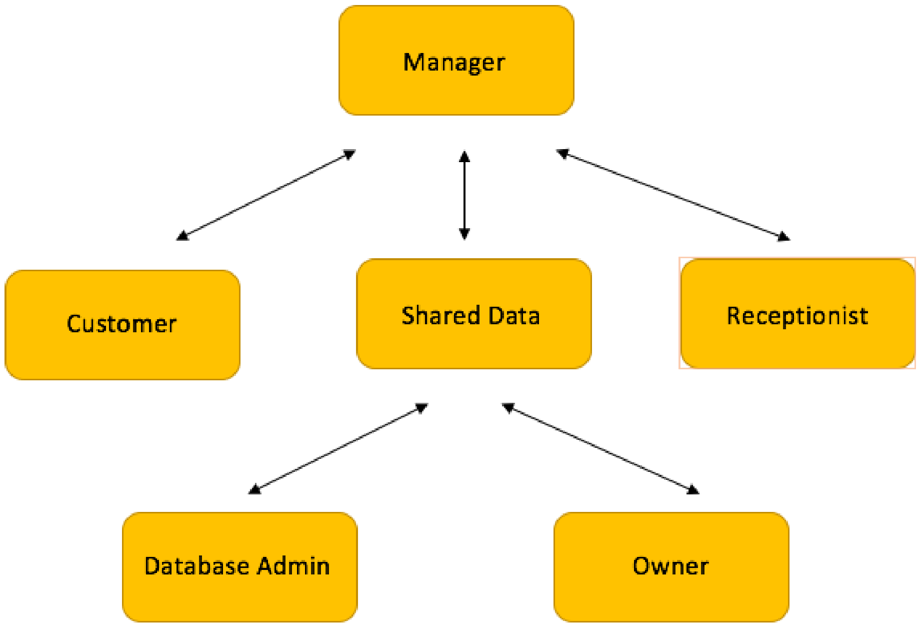


Fig5: Repository Architectural style for online hotel management system

In the above figure the distributed data over the network is shared by the clients of the organization. Some may have access to the data while others can have the privileges to modify, delete or update data stored in the database.

Comparison

	Client Server	**Layered**	**Database Centric**
Description	Partitions the system into two applications, where the client makes requests to the server and it responds.	Partitions the concerns of the application into stacked groups called layers.	Data can be accessed from a centralized location as distributed data by placing it on the network.
Cost	Higher cost due to complex structure with many servers and requires professionals to maintain it	The cost increases as the number of layers increases. The more abstract the system is, the lesser the cost.	The cost required to move the data store to the network server is high.
Performance	Performance may be unpredictable because it	Performance can be a problem because of multiple	The performance will be good if the

	depends on the network as well as the system.	levels of interpretation of a service request and it is processed at each layer.	server never slows down. If there are any up and downs, it may affect the performance.
Maintainability	Low rate of maintainability if servers are owned by different organizations	Separation of core concerns helps to identify dependencies, and organizes the code into more manageable sections.	Easy to maintain because the data is free from errors and inconsistency which, if contains may slow down the automation process.
Security	High security- all data is stored on the server, which generally offers a greater control of security than client machines.	Higher security can be provided with innermost layers protected with a high level of security validation.	Since the data stored on to the network server the security of the data is guaranteed because unauthorized access is not permitted.
Availability	Servers are independent of each other. Services and servers can be changed without affecting other parts of the system.	Includes redundant components so that it is possible to replace and update components without affecting the system.	The availability of the data store is guaranteed until some kind of fluctuations occur in the network.

Conclusion

All the above mentioned three architectures have their own advantages and disadvantages but after careful examination of pros and cons of each architecture style for our Online Hotel management system, we felt that the most suitable style for developing web/mobile application would be 'The Client-Server architecture', where all the operations are based on a request - response actions, which is well implemented through this style. It supports many thousands of transactions per day and ensures Atomicity, Consistency, Integrity and Durability (ACID) properties which is required in online transactions. These factors make the Client-Server architecture more efficient and suitable for our online hotel management system.