Bahria University
Discovering Knowledge
Department of Software Engineering

SEN-221-SOFTWARE DESIGN & ARCHITECTURE SPRING 2022

COURSE INSTRUCTOR ENGR. MAJID KALEEM

Assignment No.	01
Assignment Title	Description of various Architectural Styles
Course Learning Outcome	CLO-02 *"Describe various architectural and design styles and patterns suitable for a given scenario."*
Full Name	Muhammad Junaid Saleem Qadri
Semester	BSE 4 B
Submission Deadline	03rd April 2022
REG	70003

INSTRUCTIONS:

- USE ONLY TIMES NEW ROMAN SIZE 12 FONT.
- EACH HEADING (UNDERLINED, BOLD AND IN CAPITAL LETTERS) AND EXAMPLE MUST START FROM A NEW LINE.
- UPLOAD SOFTCOPY ON LMS AS A PDF FILE.
- DO NOT EDIT (THIS) ASSIGNMENT FILE GIVEN AS A PDF FILE.
- LAST PAGE OF YOUR ASSIGNMENT MUST CONTAIN SOURCES/REFERENCES (USE IEEE REFERENCING STYLE).
- NO MAKEUP ASSIGNMENTS WILL BE GIVEN & DATE WILL NOT BE EXTENDED.
- VIOLATION OF ANY OF THE INSTRUCTIONS MENTIONED HERE WILL RESULT IN MARKS DEDUCTION.

BAHRIA UNIVERSITY (KARACHI CAMPUS)

Software Design & Architecture (SEN-221)

ASSIGNMENT # 1 - Spring 2022

Based on: CLO-2

Class: **BSE-4B** Submission Deadline: **03rd April 22**

Course Instructor: **ENGR. MAJID KALEEM** Max Marks: **05**

1. Suppose you have to design & develop an online LMS System. You may consider various architectural and design style to address this situation. Keep LMS in mind, describe (separately) each of the architectural styles given below with respect to the following questions:

1. ***What*** are they?
2. ***Where*** are they used?
3. ***Why*** are they used?
4. ***How*** are they used / implemented?

a) Client/Server Architecture
b) Component-Based Architecture
c) MVC Architecture
d) Cloud Application Architecture

Follow the format/sample as mentioned below to answer this question:

CLIENT/SERVER ARCHITECTURE

WHAT?
…………………………………………………………………………………………
…………………………………………………………………………………………

WHERE?
…………………………………………………………………………………………
…………………………………………………………………………………………

WHY?
…………………………………………………………………………………………
…………………………………………………………………………………………

HOW?
…………………………………………………………………………………………
…………………………………………………………………………………………

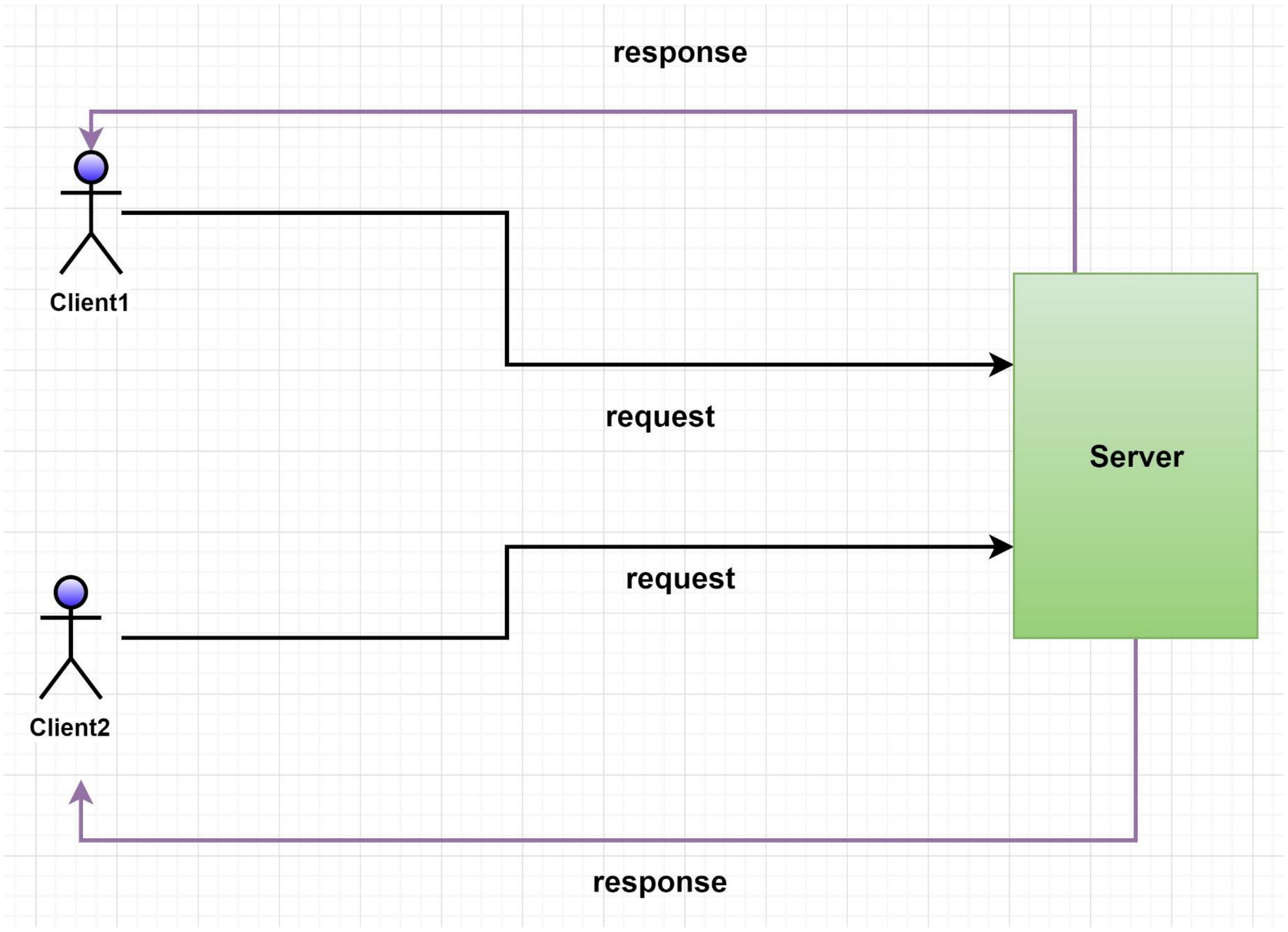
Follow the above-mentioned format to answer a, b, c, & d. While describing/answering each of the above, you may use relevant diagrams and examples to support your answer.

G☺□d Luck!

A) CLIENT/SERVER ARCHITECTURE

WHAT?

Client-server architecture is a computing model in which the **server** hosts, delivers and manages most of the resources and services to be consumed by the **client**. This type of architecture has one or more client computers connected to a central server over a network or internet connection. Client-server architecture is also known as a **networking computing model** or **client-server network** because all the requests and services are delivered over a network [1].



WHERE?

- The client-server architecture is most useful for applications that require a separation or abstraction of concerns between the client and the server; it is meant for systems with high interoperability. The client-server architectural style helps applications improve performance in scalability [1].

- In systems that need separation of functionality, the client-server architecture design is most applicable. Request validation and input could be handled from the client side while the load balancer routes the request to the server for adequate processing. The server will be responsible for processing the client's request, and returning the result via the right protocol. These layers (client and server) complete tasks independently and they are useful for abstracting functionality;
  for example, the client does not need to know how the server handles user authentication or request validation [1].

- With the separation of functionality comes the ability of each layer to function more efficiently at large scale. Modern techniques have been developed within the client-server architecture to solve scalability challenges like load balancing, sharing, and partitioning. These techniques provide performance improvements for multiple requests on the server side of the architecture and will be useful for software programs that deal with multiple requests/users [1].

WHY?

We are in an era where information technology plays a critical role in business applications, considered as an area an organization would highly invest in order to widen the opportunities available to compete the global market. **"A competitive global economy will ensure obsolescence and obscurity to those who cannot or are unwilling to compete"(**Client/Server Architecture,2011), according to this statement it's necessary for organizations sustain its market position by reengineering prevailing organizational structures and business practices to achieve their business goals. In short it's a basic need to evolve with the change of technological aspects. Therefore organizations should undergo a mechanism to retrieve and process its corporate data to make business procedures more efficient to excel or to survive in the global market. The client/server model brings out a logical perspective of **distributed corporative processing** where a server handles and processes all client requests. This can be also viewed as a revolutionary milestone to the data processing industry. Client/server computing has a vast progression in the computer industry leaving any area or corner untouched. Often hybrid skills are required for the development of client/server applications including database design, transaction processing, communication skills, graphical user interface design and development etc. Advanced applications require expertise of distributed objects and component infrastructures. Most commonly found client/server strategy today is PC LAN implementation optimized for the usage of group/batch. This has basically given threshold to many new distributed enterprises as it eliminates host-centric computing [2].

HOW?

In **client-server architecture**, many clients (remote processors) request and receive service from a **centralized server** (host computer). Client computers provide an interface to allow a computer user to request services of the server and to display the results the server returns. Servers wait for requests to arrive from clients and then respond to them. Ideally, a server provides a standardized transparent interface to clients so that clients need not be aware of the specifics of the system (i.e., the hardware and software) that is providing the service. Clients are often situated at workstations or on personal computers, while servers are located elsewhere on the network, usually on more powerful machines. This computing model is especially effective when clients and the server each have distinct tasks that they routinely perform. In hospital data processing,
for example, a client computer can be running an application program for entering patient information while the server computer is running another program that manages the database in which the information is permanently stored. Many clients can access the server's information simultaneously, and, at the same time, a client computer can perform other tasks, such as sending e-mail [3].

EXAMPLES

1. Web Servers:

A robust computational device that can manage many websites is like a web server. Installing numerous kinds of web server applications on this computer, such as Apache or Microsoft IIS, offers links to the various web pages hosted on the online, and such servers are connected to the Internet by higher-speed connections that offer ultra-speed data transfer speeds [4].

2. File Servers:

File server is dedicated systems that allow users to access for all files. It works like as centralized file storage location, and it can be accessed by several terminal systems [5].

B) Component-Based Architecture

WHAT?

A **component based** architecture is a type of application architecture composed of independent, modular, and reusable building blocks called **components**. When designing an app following component-based architecture principles, developers combine, reuse, and version these objects, saving a lot of time from building every inch of an app from scratch [6]. Component-based architecture focuses on the decomposition of the design into individual functional or logical components that represent well-defined communication interfaces containing methods, events, and properties. It provides a higher level of abstraction and divides the problem into sub-problems, each associated with component partitions [7].

WHERE?

Component based architecture stays up-to-date, without rebuilding it from scratch. That makes component based architecture a better fit for companies with complex, monolithic codebases. Using components turns a monolith into software building blocks. And these components can be combined, reused, and versioned [8].

WHY?

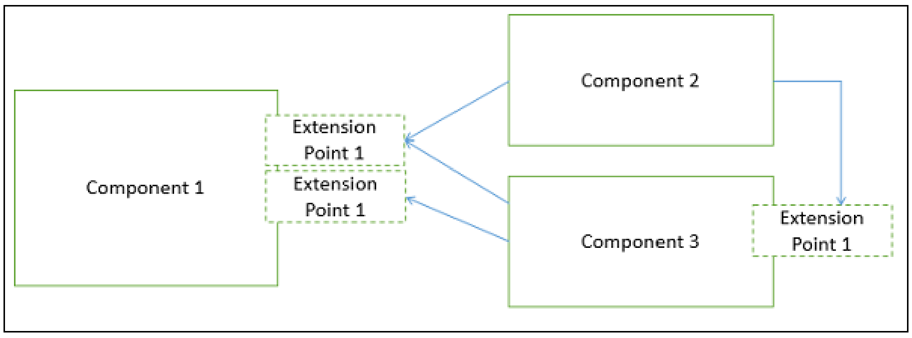
The primary objective of component-based architecture is to ensure **component reusability**. A component **encapsulates** functionality and behaviors of a software element into a **reusable** and **self-deployable** binary unit. There are many standard component frameworks such as COM/DCOM, JavaBean, EJB, CORBA, .NET, web services, and grid services. These technologies are widely used in local desktop GUI application design such as graphic JavaBean components, MS ActiveX components, and COM components which can be reused by simply drag and drop operation [7].

HOW?

A component-level design can be represented by using some intermediary representation (e.g. graphical, tabular, or text-based) that can be translated into source code. The design of data structures, interfaces, and algorithms should conform to well-established guidelines to help us avoid the introduction of errors [7].

- The software system is decomposed into reusable, cohesive, and encapsulated component units.
- Each component has its own interface that specifies required ports and provided ports; each component hides its detailed implementation.
- A component should be extended without the need to make internal code or design modifications to the existing parts of the component.
- Depend on abstractions component do not depend on other concrete components, which increase difficulty in expendability.
- Connectors connected components, specifying and ruling the interaction among components. The interaction type is specified by the interfaces of the components.

- Components interaction can take the form of method invocations, asynchronous invocations, broadcasting, message driven interactions, data stream communications, and other protocol specific interactions.
- For a server class, specialized interfaces should be created to serve major categories of clients. Only those operations that are relevant to a particular category of clients should be specified in the interface.
- A component can extend to other components and still offer its own extension points. It is the concept of plug-in based architecture. This allows a plugin to offer another plugin API.



EXAMPLES

Examples of component based models are:

- Enterprise JavaBeans (EJB) model,
- Component Object Model (COM) model,
- .NET model,
- X-MAN component model, and
- Common Object Request Broker Architecture (CORBA) component model [9].

C) MVC Architecture

WHAT?

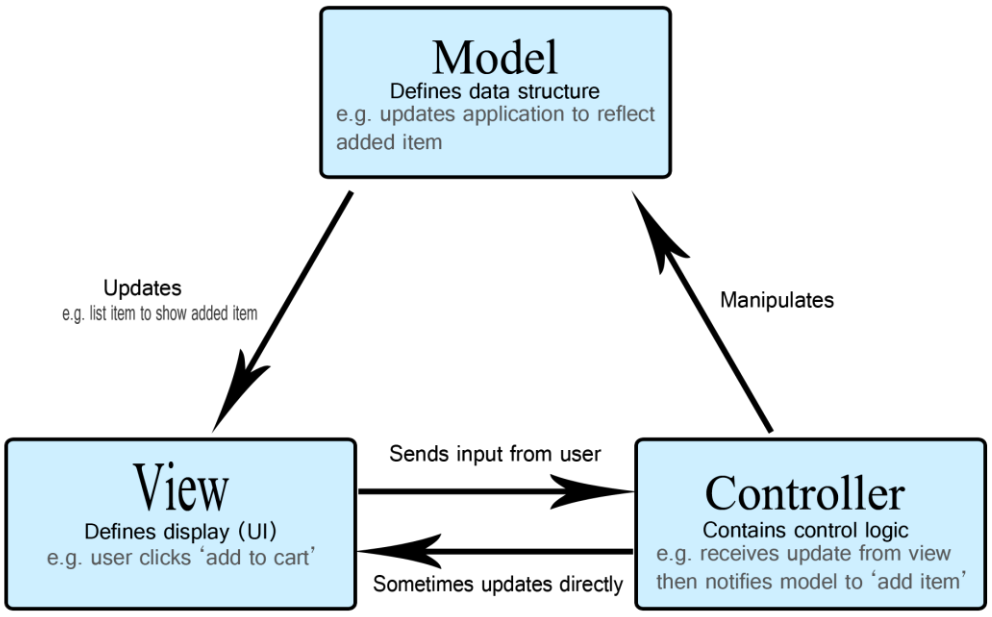
Model view controller (MVC) [10] is an architectural pattern
usually used in web-based applications. It provides three main
layers; model, view, and controller. Many developers use MVC as a
standard design pattern. It is a complete framework. MVC provide
three types of classes:

A. Model- Model classes are used to implement the logic
of data domains. These classes are used to retrieve, insert
or update the data into the database associated with our
application.

B. View- Views are used to prepare the interface of our
application. By using that interface users interact with our
application.

C. Controller- Controller classes are used to respond to the
user’s requests. Controller classes perform the users requested
actions. These classes work with model classes and select the
appropriate view that should be displayed to the user according
to user requests.

MVC pattern architecture is basically a three-layered
architecture. It separates the characteristics of application. Its
first layer is related to the user input logic, second layer is related
to the business logic and third layer is used to implement user
interface logic. MVC provide very loose coupling among these
three layers. MVC pattern are used to specify the location of
each logic in application [11]. MVC patterns provide the facility of
parallel development. It means that each layer of the application
independent of each other i.e. three developer can work on the
single application simultaneously [12]. One developer will be
working on user input logic (controller logic), other developer will be working on the user interface
logic (view) and third developer
will be working on the business logic (model) at the same time.


Model
Defines data structure
e.g. updates application to reflect added item
Updates
e.g. list item to show added item
Manipulates
View
Defines display (UI)
e.g. user clicks 'add to cart'
Sends input from user
Sometimes updates directly
Controller
Contains control logic
e.g. receives update from view then notifies model to 'add item'

WHERE?

If LMS application are making an with enough serious stimulating on the client side to refuse to go along with JavaScript alone. If LMS are developing an application which have a very high lifting on the server side and a little communication on the client side then we should not use the MVC pattern architecture, instead we should use simple setup such as web-based form model. The following are some characteristics that will help us whether to use MVC architecture in our LMS application or not:

i. LMS application needs asynchronous communication on the backend.

ii. LMS application has a functionality which results in not to reload a full page for example uploading huge amount of Assignment on a LMS at the same time.

iii. Manipulation of data is mostly on the client side (browser) rather than server side.

iv. Same type of data is being delivered in different ways on a single page (navigation).

v. When LMS application has many insignificant connections that are used to modify data (button, switches).

WHY?

As we divide the logic of our application into three tasks (input logic, business logic, interface logic), testing of these components would become very easy. Testability is very fast and flexible, as we can use any unit testing framework compatible with MVC framework. It is an extensible and pluggable framework. We can design the components of LMS application in such a way that these are easily replaceable or can be modified easily. We can plug our own view engine, URL routing strategy, action method constraint serialization. Instead of depending on class to create objects we use a technique dependency injection (DI) which enable us to inject object into classes. Another technique inversion of control (IOC) is used to show dependency among objects, it specifies that which object need other which object. MVC provide URL mapping component that helps us to build using understandable and searchable URLs. Instead of using file name extensions MVC support URL naming patterns that are very useful for search engine optimization (SEO) and representational state transfer (REST) addressing. Some frameworks of MVC such as ASP.NET MVC framework provide us some built in features such as form authentication, session management, transactional business logic, web application security, object relational mapping, localization, membership and roles and URL authorization etc. The most popular frameworks available today are backbone.js, ember.js; angular.js and knockout.js.

HOW?

Before we start the development of application we should decide that which type of architecture we should use to develop our application. We should examine the functional components of our application and determine the way through which they will communicate with each other. There are many design patterns are available which can be used to develop an application. Here we are going to use MVC architecture pattern. MVC pattern divides our application into three major components i.e. Model, View and Controller. Model is used to communicate with our database. View is used to get data from model and present it to the user. And finally, the controller is used to define the application behavior. It interacts with the inputs of the users. Normally in web-based application user’s inputs are HTTP GET and HTTP POST.

After creating front-end of our application, we enabled our application that it can communicate with the university database, for which we add sample data into the database and set up a data source and a connection pool on Glassfish server to proceed by JSP pages which will test the data source. We also use the JSTL library and SQL tag library to retrieve and show the Assigment,etc of different courses.

D) CLOUD APPLICATION ARCHITECTURE

WHAT?

Cloud computing has changed the thinking of industrial and scientific people in many ways. Minimizing the cost, it provides the solution of IT infrastructure easily. Cloud computing is the technique that is defined as on demand delivery of IT resources through the internet with pay-as you-go pricing system. There are two kinds of people involved to this system. One of them is service provider and other one is subscriber. Service providers are actually company's IT people or a third party or a combination of company and the third party. On the other hand subscriber may be anyone who takes the services from the service providers.

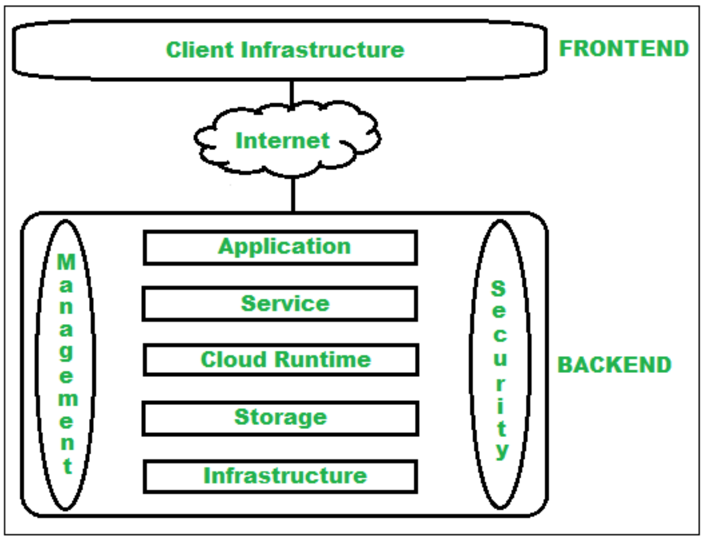
Cloud providers give the services of computing resources like databases, computing power and storage to the users where users need not to buy or become owner or maintain the physical data servers and centers [13]. Users can access the data hosted on cloud storage from any place in any time. Having internet connection to Laptop, Tab, Desktop and Smart Phone, they can manage or insert the data very smoothly on time [14].

Cloud Computing services

Cloud is visible in every sector of life wherever it may be office or home. People need to communicate with their relatives, friends and colleagues. So they need various apps and devices which are interacted with the cloud . It can be classified into three main services:

1. Software-as-a-Service (SaaS),
2. Infrastructure-as-a-Service (IaaS) and
3. Platform-as-a-Service (PaaS)

This classification creates the cloud stack as mentioned in fig.2. In this cloud stack, SaaS is on the top, PaaS is in the middle, and IaaS is on the bottom. Users do not interact with the platform or infrastructure on which it runs. Rather they interact with the software that is hosted on the cloud. So, SaaS is kept on the top of the cloud stack.



WHY?

On-demand self-service: Users can get the services of cloud computing whenever they need.
Resource pooling: Several resources are pooled simultaneously and provided to the users dynamically.
Rapid elasticity: Scalable services are provided by rapid elasticity. If the users need extra space on the cloud, they can take the permission by this.
Broad network access: All the services those are available on the cloud can be accessed by the users by following some methods.
Measured service: Resource utilization is definitely monitored and controlled. So transparent service is provided to customer and provider.
Pay as you go: User needs not to pay extra charge for the service that he taken from the provider. He needs to pay only for the service he has taken. Instead of extra payment, provider gives some extra spaces always for free [15].
Easy maintenance: Maintenance of the servers is very easy and the downtime is ignorable.
Automatic System: Demand of the users may be measured automatically. Since the overall scenario of the service is monitored, it is possible to report to the user about their total usages of the services.
Security: This is a very much important issue of cloud computing. If any server even gets damaged, data will not get lost because it takes a snapshot of the stored data. The storage devices are used to store the data that is about to impossible for any other person to utilize or

hack.
Device and Location Independence: Cloud computing allows the users to access the system from any location using any kind of device (PC or Mobile) via internet by any browser.

HOW?

Cloud applications are best deployed as a collection of cloud services, or APIs. LMS build up from the data to the services and then combine those services into composite services or complete composite applications.

This is service-based or service-oriented architecture, at its essence.When developing an application architecture for the cloud, we deal with complex distributed systems that can take advantage of loosely coupled applications built on many services that can also be decoupled from the data .We separate the LMS application services physically, executing on the proper machine instances, and service/API managers and governance technology that provide services directories can help track the many services that make up your application.

Decouple the data
If we tightly couple the data to the application, it won't find a good home in the cloud. Private and public clouds are complex distributed systems that work best with application architectures that break out processing and data into separate components. Database reads and writes across the open Internet can cause latency, and database communications may determine how close your data sits to the services and applications that need to leverage it.

Consider communications between application components
Decoupling applications, both data and services, doesn't mean our application is properly architected for the cloud.

Focus on designing applications that optimize communications between application components. For example, combine communications into a single stream of data or a group of messages, rather than constantly communicating as if the application components reside on a single platform.

Model and design for performance and scaling
Extend considerations around how application components communicate to include overall performance as well. This includes understanding how the application will scale under an increasing load.

Designing for performance means first building a model that represents how the application behaves under an increasing load. If 1,000 or more users log on at the same time, how will the application handle the increased traffic on the network, the increased load on the application servers, and the load placed on the back-end databases.

In some cases, cloud service providers offer auto-scaling capabilities, where provisioning occurs automatically. The most efficient path, however, lies in understanding the application's workload profile and defining the path to scaling the application, as well as putting mechanisms in place to ensure that it will, indeed, scale.

Finally, monitor overall application performance using application-aware performance monitoring tools, and create interfaces within the application to better enable performance monitoring. How the application provisions and de-provisions resources should be innate to the application as well.

Make security systemic within the application
we build LMS applications, security is typically an afterthought. When hosting an application in the cloud, however, security should be a high priority. Your cloud-based application architecture should make security systemic to the application—it should be designed and built into the application architecture.

Generally speaking, cloud-based applications should leverage identity and access management (IAM). Enterprises that develop mature IAM capabilities can reduce their security costs and, more importantly, become significantly more agile at configuring security for cloud-based applications. Indeed, IAM will be a part of more than 50 percent of existing applications that migrate to the public cloud and nearly 90 percent of new applications built on clouds.

our core objective is to design security into the application and take advantage of the native features of both the cloud and the IAM system you use. However, each application has its own requirements based upon the needs of the business, and security always differs from one enterprise to another.

REFERNCES

[1] cs.waterloo.ca, ‘Client Server’. [Online]. Available:
https://cs.uwaterloo.ca/~m2nagapp/courses/CS446/1195/Arch_Design_Activity/ClientServer.pd
f
[2] wiki.org, ‘Client Sever Architecture, The Purpose of Client Server Architecture’. [Online].
Available: https://cio-
wiki.org/wiki/Client_Server_Architecture#The_Purpose_of_Client.2FServer_Architecture.5B3.
5D
[3] Britannica.com, ‘Client Server Architecture’. [Online]. Available:
https://www.britannica.com/technology/client-server-architecture
[4] teachcomputerscience.com, ‘Client Server Architecture’. [Online]. Available:
https://teachcomputerscience.com/client-server-architecture/
[5] digitalthinkerhelp.com, ‘What is Client Server Architecture Diagram Types Examples
Components. [Online]. Available: http://digitalthinkerhelp.com/what-is-client-server-
architecture-diagram-types-examples-components/
[6] outsystems.com, ‘Component Architecture’. [Online]. Available:
https://www.outsystems.com/blog/posts/component-architecture/
[7] tutorialspoint.com, ‘Component based architecture’. [Online]. Available:
https://www.tutorialspoint.com/software_architecture_design/component_based_architecture.ht
m
[8] perforce.com, ‘Component Based Development’. [Online]. Available:
https://www.perforce.com/blog/vcs/component-based-development
[9] wikipedia.org, ‘Component based software engineering'. [Online]. Available:
https://en.wikipedia.org/wiki/Com
[10] Gupta P, Govil MC (2010) MVC Design pattern for the multi framework
distributed applications using XML, spring and struts framework.
International Journal on Computer Science and Engineering 2(4): 1047-
1051.
[11] Shu-qiang H, Huan-ming Z (2008) Research on improved MVC design
pattern based on struts and XSL. 2008 International Symposium on
Information Science and Engineering, pp. 451-455.
[12] Selfa DM, Carrillo M, Boone MDR (2006) A database and web application
based on MVC architecture. 16th International Conference on Electronics,
Communications and Computers (CONIELECOMP’06), p. 48.
[13] https://aws.amazon.com/what-is-cloud-computing/
[14] Rafat Ara, Md. Abdur Rahim," An Online Based
Inventory Management System Implementation in
Printing Business", International Journal of Emerging
Technologies and Innovative Research, ISSN:2349-
5162, Vol.5, Issue 11, page no. pp176-179, November
2018. Available at:
http://www.jetir.org/papers/JETIR1811B29.pdf
[15] https://data-flair.training/blogs/features-of-cloud   computing/