CMR
INSTITUTE OF
TECHNOLOGY

USN

Internal Assessment Test 3– Mar. 2023

Sub:	**Cloud Computing**							**Sub Code:**	**20MCA342**
Date:	**14/3/2023**	**Duration:**	**90 min's**	**Max Marks:**	**50**	**Sem:**	**III**	**Branch:**	**MCA**

Note : Answer FIVE FULL Questions, choosing ONE full question from each Module

		MARKS	OBE	
			CO	RBT
	PART I			
1	Illustrate Microsoft windows Azure platform architecture for cloud computing	[10]	CO4	L3
	OR			
2	Discuss application services of Google AppEngine.	[10]	CO4	L3
	PART II			
3	Explain Communication services of AWS	[10]	CO4	L3
	OR			
4	Discuss the storage services offered by Amazon Web Services (AWS)	[10]	CO4	L3
	PART III			
5	Explain in detail about Dropbox and Icloud	[10]	CO5	L4
	OR			
6	Explain the salesforce and Force.com architecture	[10]	CO5	L4
	PART IV			
7	Describe how cloud computing technology can be applied to support remote ECG monitoring.	[10]	CO5	L4
	OR			
8	Briefly explain architecture and overview of the Jeeva Portal	[10]	CO5	L4
	PARTV			
9	Discuss social networking application Facebook	[10]	CO5	L4
	OR			
10	Discuss Google Docs and Cloud Desktops EyeOS and XIOS/3	[10]	CO5	L4

Q1) Illustrate Microsoft windows Azure platform architecture for cloud computing

The Windows Azure platform is made up of a foundation layer and a set of developer services that can be used to build scalable applications. These services cover compute, storage, networking, and identity management, which are tied together by middleware called AppFabric. This scalable computing environment is hosted within Microsoft datacenters and accessible through the Windows Azure Management Portal. Alternatively, developers can recreate a Windows Azure environment (with limited capabilities) on their own machines for development and testing purposes. In this section, we provide an overview of the Azure middleware and its services.

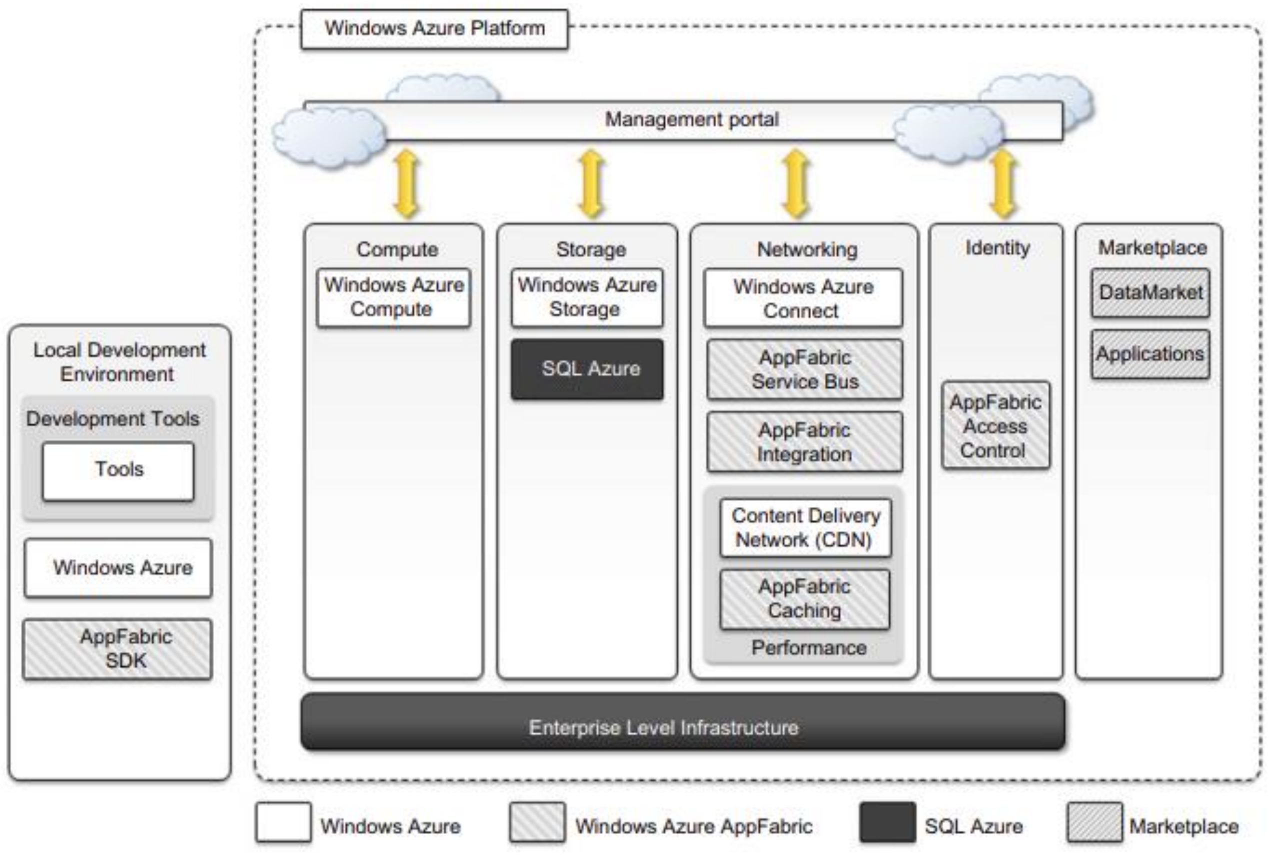

FIGURE 9.3

Microsoft Windows Azure Platform Architecture.

Compute services

Compute services are the core components of Microsoft Windows Azure, and they are delivered by means of the abstraction of roles. A role is a runtime environment that is customized for a specific compute task. Roles are managed by the Azure operating system and instantiated on demand in order to address surges in application demand. Currently, there are three different roles: Web role, Worker role, and Virtual Machine (VM) role.

Web role

The Web role is designed to implement scalable Web applications. Web roles represent the units of deployment of Web applications within the Azure infrastructure.

Worker role
Worker roles are designed to host general compute services on Azure. They can be used to quickly provide compute power or to host services that do not communicate with the external world through HTTP.

Virtual machine role
The Virtual Machine role allows developers to fully control the computing stack of their compute service by defining a custom image of the Windows Server 2008 R2 operating system and all the service stack required by their applications. The Virtual Machine role is based on the Windows Hyper-V virtualization technology, which is natively integrated in the Windows server technology at the base of Azure.

Storage services
Blobs Azure allows storing large amount of data in the form of binary large objects (BLOBs) by means of the blobs service. This service is optimal to store large text or binary files. Two types of blobs are available:
- Block blobs
- Page blobs

Azure drive Page blobs can be used to store an entire file system in the form of a single Virtual Hard Drive (VHD) file
Tables constitute a semi structured storage solution, allowing users to store information in the form of entities with a collection of properties.
Queue storage allows applications to communicate by exchanging messages through durable queues, thus avoiding lost or unprocessed messages. Applications enter messages into a queue, and other applications can read them in a first-in, first-out (FIFO) style.

Core infrastructure:

AppFabric
AppFabric is a comprehensive middleware for developing, deploying, and managing applications on the cloud or for integrating existing applications with cloud services. AppFabric implements an optimized infrastructure supporting scaling out and high availability; sandboxing a multitenancy; state management; and dynamic address resolution and routing. On top of this infrastructure, the middleware offers a collection of services that simplify many of the common tasks in a distributed application, such as communication, authentication and authorization, and data access. These services are available through language-agnostic interfaces, thus allowing developers to build heterogeneous applications.

Q2) Discuss application services of Google AppEngine

Applications hosted on AppEngine take the most from the services made available through the runtime environment. These services simplify most of the common operations that are performed in Web applications: access to data, account management, integration of external resources, messaging and communication, image manipulation, and asynchronous computation.

UrlFetch

Web 2.0 has introduced the concept of composite Web applications. Different resources are put together and organized as meshes within a single Web page. Meshes are fragments of HTML generated in different ways. They can be directly obtained from a remote server or rendered from an XML document retrieved from a Web service, or they can be rendered by the browser as the result of an embedded and remote component. A common characteristic of all these examples is the fact that the resource is not local to the server and often not even in the same administrative domain. Therefore, it is fundamental for Web applications to be able to retrieve remote resources. The sandbox environment does not allow applications to open arbitrary connections through sockets, but it does provide developers with the capability of retrieving a remote resource through HTTP/HTTPS by means of the UrlFetch service.
Applications can make synchronous and asynchronous Web requests and integrate the resources obtained in this way into the normal requesthandling cycle of the application. One of the interesting features of UrlFetch is the ability to set deadlines for requests so that they can be completed (or aborted) within a given time. Moreover, the ability to perform such requests asynchronously allows the applications to continue with their logic while the resource is retrieved in the background. UrlFetch is not only used to integrate meshes into a Web page but also to leverage remote Web services in accordance with the SOA reference model for distributed applications.

MemCache

AppEngine provides developers with access to fast and reliable storage, which is DataStore. Despite this, the main objective of the service is to serve as a scalable and long-term storage, where data are persisted to disk redundantly in order to ensure reliability and availability of data against failures. This design poses a limit on how much faster the store can be compared to other solutions, especially for objects that are frequently accessed—for example, at each Web request. AppEngine provides caching services by means of MemCache. This is a distributed in-memory cache that is optimized for fast access and provides developers with a volatile store for the objects that are frequently accessed. The caching algorithm implemented by MemCache will automatically remove the objects that are rarely accessed. The use of MemCache can significantly reduce the access time to data; developers can structure their applications so that each object is first looked up into MemCache and if there is a miss, it will be retrieved from DataStore and put into the cache for future lookups.

Mail and instant messaging

Communication is another important aspect of Web applications. It is common to use email for following up with users about operations performed by the application. Email can also be used to trigger activities in Web applications. To facilitate the implementation of such tasks, AppEngine provides developers with the ability to send and receive mails through Mail. The service allows sending email on behalf of the application to specific user accounts. It is also possible to include several types of attachments and to target multiple recipients. Mail operates asynchronously, and in case of failed delivery the sending address is notified through an email detailing the error. AppEngine provides also another way to communicate with the external world: the Extensible Messaging and Presence Protocol (XMPP). Any chat service that supports XMPP, such as Google Talk, can send and receive chat messages to and from the Web application, which is identified by its own address. Even though the chat is a communication medium mostly used for human interactions, XMPP can be conveniently used to connect the Web application with chat bots or to implement a small administrative console.

Account management

Web applications often keep various data that customize their interaction with users. These data normally go under the user profile and are attached to an account. AppEngine simplifies account management by allowing developers to leverage Google account management by means of Google Accounts. The integration with the service also allows Web applications to offload the implementation of authentication capabilities to Google's authentication system. Using Google Accounts, Web applications can conveniently store profile settings in the form of key-value pairs, attach them to a given Google account, and quickly retrieve them once the user authenticates. With respect to a custom solution, the use of Google Accounts requires users to have a Google account, but it does not require any further implementation. The use of Google Accounts is particularly advantageous for developing Web applications within a corporate environment using Google Apps. In this case, the applications can be easily integrated with all the other services (and profile settings) included in Google Apps.

Image manipulation

Web applications render pages with graphics. Often simple operations, such as adding watermarks or applying simple filters, are required. AppEngine allows applications to perform image resizing, rotation, mirroring, and enhancement by means of Image Manipulation, a service that is also used in other Google products. Image Manipulation is mostly designed for lightweight image processing and is optimized for speed.

Q3) Explain Communication services of AWS

Amazon provides facilities to structure and facilitate the communication among existing applications and services residing within the AWS infrastructure. These facilities can be organized into two major categories: virtual networking and messaging.

Virtual networking

Virtual networking comprises a collection of services that allow AWS users to control the connectivity to and between compute and storage services. Amazon Virtual Private Cloud (VPC) an Amazon Direct Connect provides connectivity solutions in terms of infrastructure; Route 53 facilitates connectivity in terms of naming.

Amazon VPC provides a great degree of flexibility in creating virtual private networks within the Amazon infrastructure and beyond. The service providers prepare either templates covering most of the usual scenarios or a fully customizable network service for advanced configurations. Prepared templates include public subnets, isolated networks, private networks accessing Internet through network address translation (NAT), and hybrid networks including AWS resources and private resources.

Messaging

Messaging services constitute the next step in connecting applications by leveraging AWS capabilities. The three different types of messaging services offered are Amazon Simple Queue Service (SQS), Amazon Simple Notification Service (SNS), and Amazon Simple Email Service (SES).

Amazon SQS constitutes disconnected model for exchanging messages between applications by means of message queues, hosted within the AWS infrastructure. Using the AWS console or directly the underlying Web service AWS, users can create an unlimited number of message queues and configure them to control their access. Applications can send messages to any queue they have access to. These messages are securely and redundantly stored within the AWS infrastructure for a limited period of time, and they can be accessed by other (authorized) applications. While a message is being read, it is kept locked to avoid spurious processing from other applications. Such a lock will expire after a given period.

Amazon SNS provides a publish-subscribe method for connecting heterogeneous applications. With respect to Amazon SQS, where it is necessary to continuously poll a given queue for a new message to process, Amazon SNS allows applications to be notified when new content of interest is available. This feature is accessible through a Web service whereby AWS users can create a topic, which other applications can subscribe to. At any time, applications can publish content on a given topic and subscribers can be automatically notified. The service provides subscribers with different notification models (HTTP/HTTPS, email/email JSON, and SQS).

Amazon SES provides AWS users with a scalable email service that leverages the AWS infrastructure. Once users are signed up for the service, they have to provide an email that SES will use to send emails on their behalf. To activate the service, SES will send an email to verify the given address and provide the users with the necessary information for the activation. Upon verification, the user is given an SES sandbox to test the service, and he can request access to the production version. Using SES, it is possible to send either SMTP-compliant emails or raw emails by specifying email headers and Multipurpose Internet Mail Extension (MIME) types. Emails are queued for delivery, and the users are notified of any failed delivery. SES also provides a wide range of statistics that help users to improve their email campaigns for effective communication with custom

Q4) Discuss the storage services offered by Amazon Web Services (AWS)

Amazon Simple Storage Service (S3)

As the name suggests, S3 has been designed to provide a simple storage service that's accessible through a Representational State Transfer (REST) interface, which is quite similar to a distributed file system but which presents some important differences that allow the infrastructure to be highly efficient:

• The storage is organized in a two-level hierarchy. S3 organizes its storage space into buckets that cannot be further partitioned. This means that it is not possible to create directories or other kinds of physical groupings for objects stored in a bucket. Despite this fact, there are few limitations in naming objects, and this allows users to simulate directories and create logical groupings.

• Stored objects cannot be manipulated like standard files. S3 has been designed to essentially provide storage for objects that will not change over time. Therefore, it does not allow renaming, modifying, or relocating an object. Once an object has been added to a bucket, its content and position is immutable, and the only way to change it is to remove the object from the store and add it again.

• Content is not immediately available to users. The main design goal of S3 is to provide an eventually consistent data store. As a result, because it is a large distributed storage facility, changes are not immediately reflected. For instance, S3 uses replication to provide redundancy and efficiently serve objects across the globe; this practice introduces latencies when adding objects to the store—especially large ones—which are not available instantly across the entire globe.

• Requests will occasionally fail. Due to the large distributed infrastructure being managed, requests for object may occasionally fail. Under certain conditions, S3 can decide to drop a request by returning an internal server error. Therefore, it is expected to have a small failure rate during day-to-day operations, which is generally not identified as a persistent failure.

Access to S3 is provided with RESTful Web services. These express all the operations that can be performed on the storage in the form of HTTP requests (GET, PUT, DELETE, HEAD, and POST), which operate differently according to the element they address. As a rule of thumb PUT/ POST requests add new content to the store, GET/HEAD requests are used to retrieve content and information, and DELETE requests are used to remove elements or information attached to them.

Amazon elastic block store

The Amazon Elastic Block Store (EBS) allows AWS users to provide EC2 instances with persistent storage in the form of volumes that can be mounted at instance startup. They accommodate up to 1 TB of space and are accessed through a block device interface, thus allowing users to format them according to the needs of the instance they are connected to (raw storage, file system, or other).

The content of an EBS volume survives the instance life cycle and is persisted into S3. EBS volumes can be cloned, used as boot partitions, and constitute durable storage since they rely on S3 and it is possible to take incremental snapshots of their content. EBS volumes normally reside within the same availability zone of the EC2 instances that will use them to maximize the I/O performance. It is also possible to connect volumes located in different availability zones. Once mounted as volumes, their content is lazily loaded in the background and according to the request made by the operating system. This reduces the number of I/O requests that go to the network. Volume images cannot be shared among instances, but multiple (separate) active volumes can be created from them. In addition, it is possible to attach multiple volumes to a single instance or create a volume from a given snapshot and modify its size, if the formatted file system allows such an operation.

Amazon ElastiCache

ElastiCache is an implementation of an elastic in-memory cache based on a cluster of EC2 instances. It provides fast data access from other EC2 instances through a Memcached-compatible protocol so that existing applications based on such technology do not need to be modified and can transparently migrate to ElastiCache. ElastiCache is based on a cluster of EC2 instances running the caching software, which is made available through Web services. An ElastiCache cluster can be dynamically resized according to the demand of the client applications. Furthermore, automatic patch management and failure detection and recovery of cache nodes allow the cache cluster to keep running without administrative intervention from AWS users, who have only to elastically size the cluster when needed.

Structured storage solutions

Traditionally, RDBMS have been the common data back-end for a wide range of applications, even though recently more scalable and lightweight solutions have been proposed. Amazon provides applications with structured storage services in three different forms: preconfigured EC2 AMIs, Amazon Relational Data Storage (RDS), and Amazon SimpleDB.

Q5) Explain in detail about Dropbox and Icloud

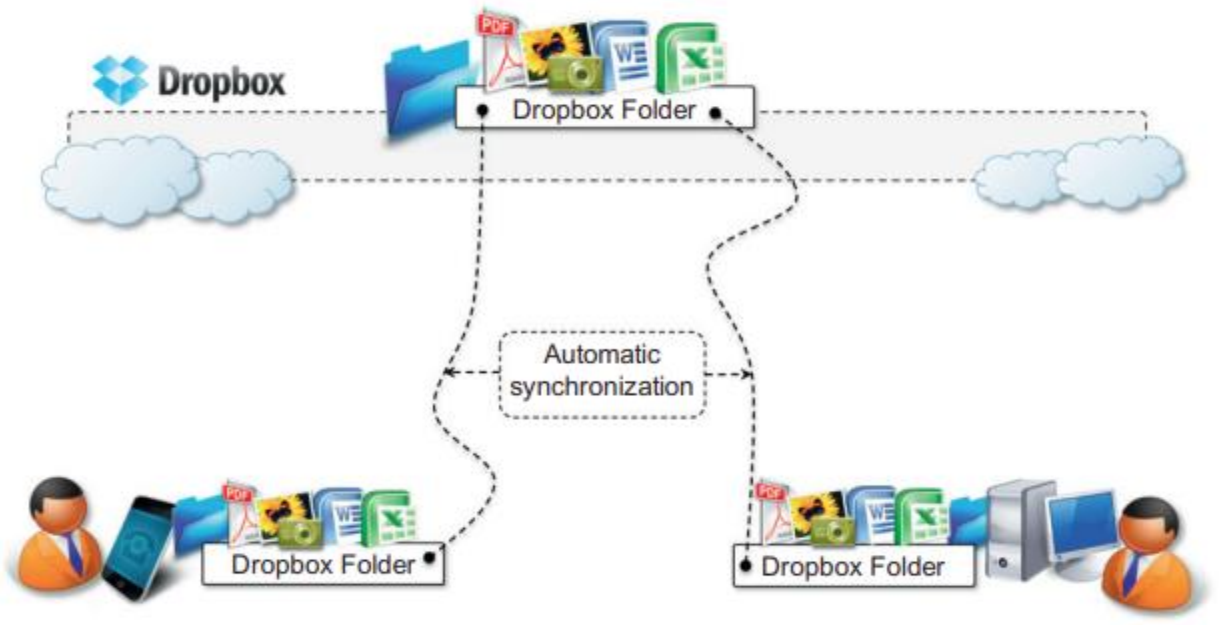

FIGURE 10.6

Dropbox usage scenario.

The most popular solution for online document storage is Dropbox, an online application that allows users to synchronize any file across any platform and any device in a seamless manner (see Figure 10.6). Dropbox provides users with a free amount of storage that is accessible through the abstraction of a folder. Users can either access their Dropbox folder through a browser or by downloading and installing a Dropbox client, which provides access to the online storage by means of a special folder. All the modifications into this folder are silently synched so that changes are notified to all the local instances of the Dropbox folder across all the devices.

The key advantage of Dropbox is its availability on different platforms (Windows, Mac, Linux, and mobile) and the capability to work seamlessly and transparently across all of them. Another interesting application in this area is iCloud, a cloud-based document-sharing application provided by Apple to synchronize iOS-based devices in a completely transparent manner. Unlike Dropbox, which provides synchronization through the abstraction of a local folder, iCloud has been designed to be completely transparent once it has been set up. Documents, photos, and videos are automatically synched as changes are made, without any explicit operation. This allows the system to efficiently automate common operations without any human intervention: taking a picture with your iPhone and having it automatically available in iPhoto on your Mac at home; editing a document on the iMac at home and having the changes updated in your iPad.

Unfortunately, this capability is limited to iOS devices, and currently there are no plans to provide iCloud with a Web-based interface that would make user content accessible from even unsupported platforms. There are other solutions for online document sharing, such as Windows Live, Amazon Cloud Drive, and

CloudMe, that are popular and that we did not cover. These solutions offer more or less the same capabilities of those we've discussed, with different levels of integration between platform and devices.

Q6) Explain the salesforce and Force.com architecture

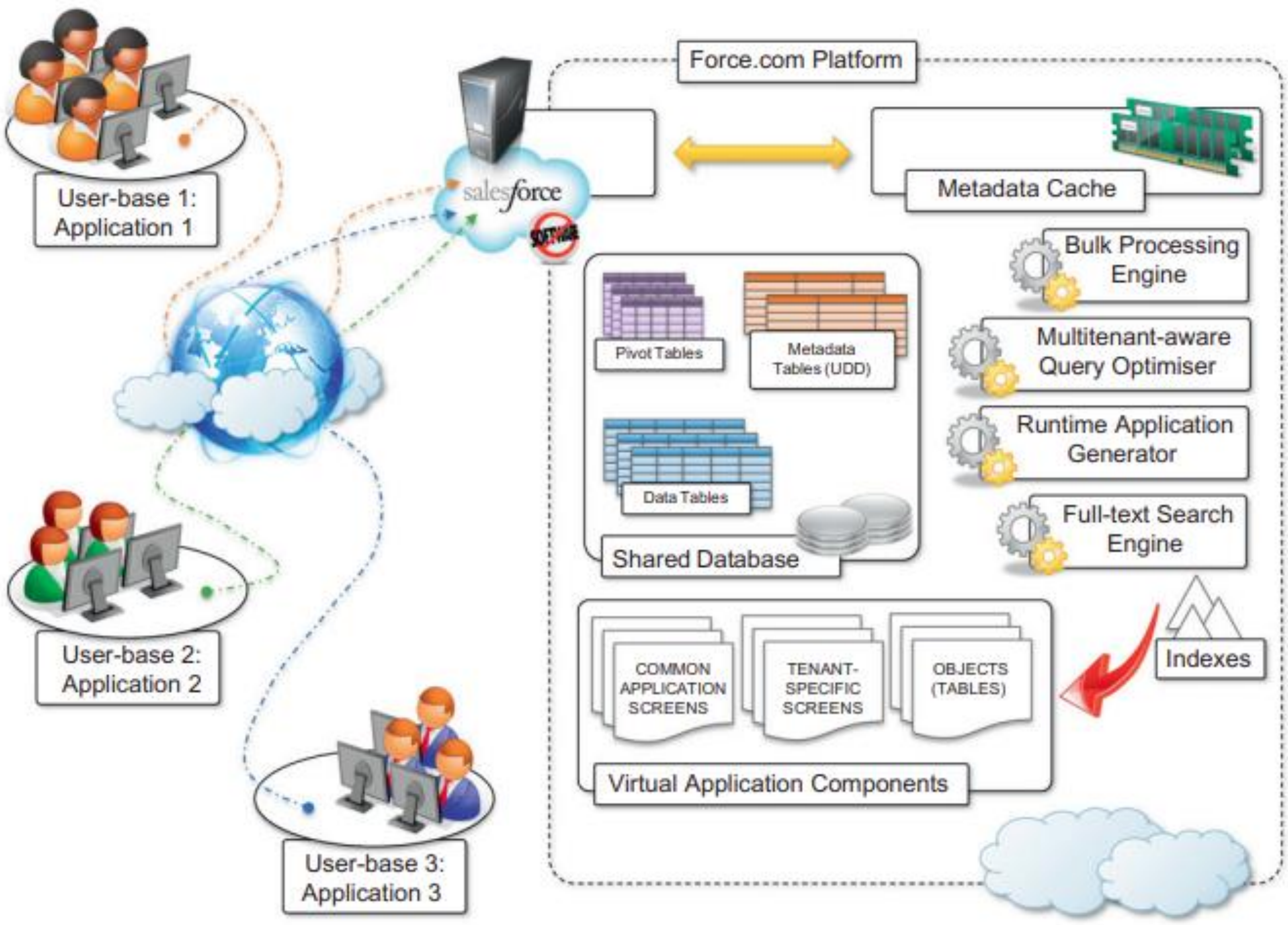
User-base 1: Application 1
User-base 2: Application 2
User-base 3: Application 3
salesforce
Force.com Platform
Metadata Cache
Bulk Processing Engine
Multitenant-aware Query Optimiser
Runtime Application Generator
Full-text Search Engine
Indexes
Pivot Tables
Metadata Tables (UDD)
Data Tables
Shared Database
COMMON APPLICATION SCREENS
TENANT-SPECIFIC SCREENS
OBJECTS (TABLES)
Virtual Application Components

FIGURE 10.5

Salesforce.com and Force.com architecture.

Salesforce.com is probably the most popular and developed CRM solution available today. As of today more than 100,000 customers have chosen Safesforce.com to implement their CRM solutions. The application provides customizable CRM solutions that can be integrated with additional features developed by third parties. Salesforce.com is based on the Force.com cloud development platform. This represents scalable and high-performance middleware executing all the operations of all Salesforce.com applications. The architecture of the Force.com platform is shown in Figure 10.5. Initially designed to support scalable CRM applications, the platform has evolved to support the entire life cycle of a wider range of cloud applications by implementing a flexible and scalable infrastructure. At the core of the platform resides its metadata architecture, which provides the system with flexibility and scalability. Rather than being built on top of specific components and tables, application core logic and business rules are saved as metadata into the Force.com store. Both application structure and application data are stored in the store.
A runtime engine executes application logic by retrieving its metadata and then performing the operations on the data. Although running in isolated containers, different applications logically share the same database

structure, and the runtime engine executes all of them uniformly. A full-text search engine supports the runtime engine. This allows application users to have an effective user experience despite the large amounts of data that need to be crawled. The search engine maintains its indexing data in a separate store and is constantly updated by background processes triggered by user interaction. Users can customize their application by leveraging the “native” Force.com application framework or by using programmatic APIs in the most popular programming languages.

The application framework allows users to visually define either the data or the core structure of a Force.com application, while the programmatic APIs provide them with a more conventional way for developing applications that relies on Web services to interact with the platform. Customization of application processes and logic can also be implemented by developing scripts in APEX. This is a Java-like language that provides object-oriented and procedural capabilities for defining either scripts executed on demand or triggers. APEX also offers the capability of expressing searches and queries to have complete access to the data managed by the Force.com platform.

Q7) Describe how cloud computing technology can be applied to support remote ECG monitoring

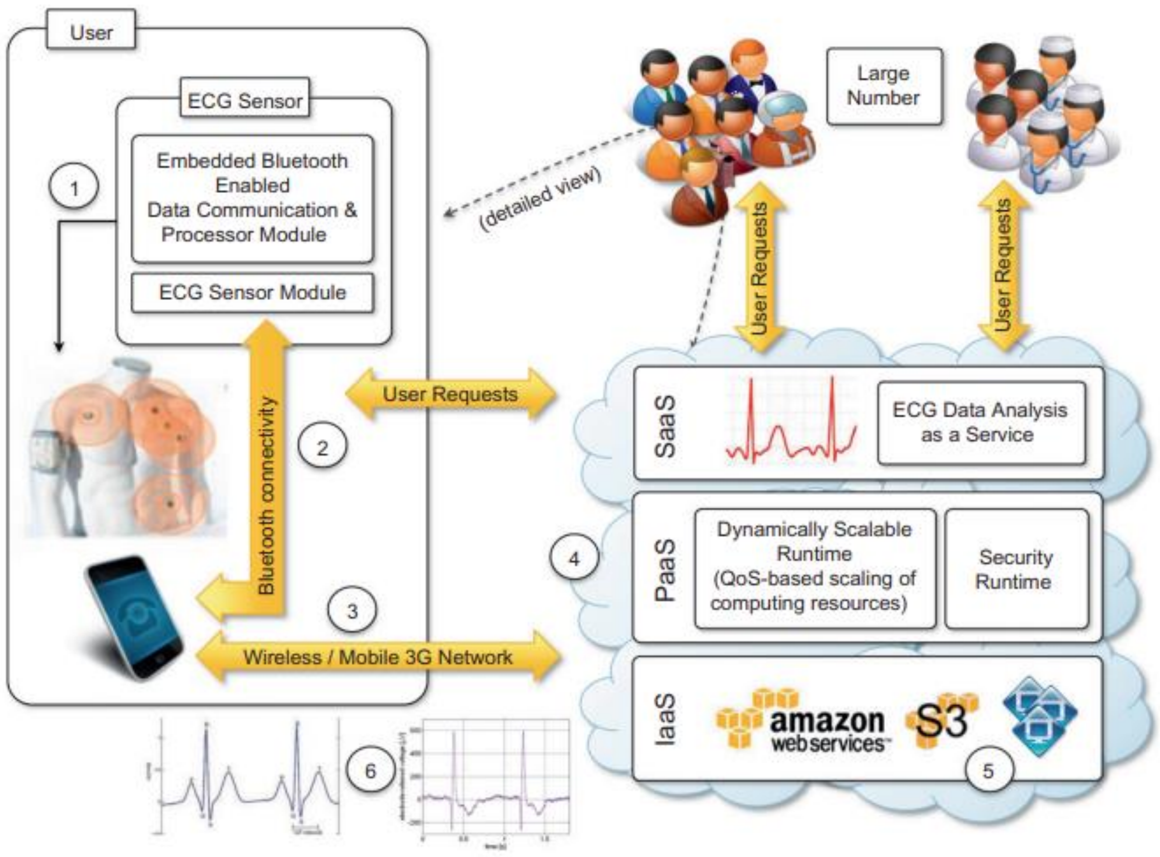
User
ECG Sensor
Embedded Bluetooth Enabled Data Communication & Processor Module
ECG Sensor Module
1
2
3
4
5
6
Bluetooth connectivity
User Requests
Wireless / Mobile 3G Network
(detailed view)
Large Number
User Requests
User Requests
SaaS
ECG Data Analysis as a Service
PaaS
Dynamically Scalable Runtime (QoS-based scaling of computing resources)
Security Runtime
IaaS
amazon webservices
S3

FIGURE 10.1
An online health monitoring system hosted in the cloud.

The capillary development of Internet connectivity and its accessibility from any device at any time has made cloud technologies an attractive option for developing health-monitoring systems. ECG data analysis and monitoring constitute a case that naturally fits into this scenario. ECG is the electrical manifestation of the contractile activity of the heart's myocardium. This activity produces a specific waveform that is repeated over time and that represents the heartbeat. The analysis of the shape of the ECG waveform is used to identify arrhythmias and is the most common way to detect heart disease. Cloud computing technologies allow the remote monitoring of a patient's heartbeat data, data analysis in minimal time, and the notification of first-aid personnel and doctors should these data reveal potentially dangerous conditions. This way a patient at risk can be constantly monitored without going to a hospital for ECG analysis. At the same time, doctors and first-aid personnel can instantly be notified of cases that require their attention.

An illustration of the infrastructure and model for supporting remote ECG monitoring is shown in Figure 10.1. Wearable computing devices equipped with ECG sensors constantly monitor the patient's heartbeat. Such information is transmitted to the patient's mobile device, which will eventually forward it to the cloud-hosted Web service for analysis. The Web service forms the front-end of a platform that is entirely hosted in the cloud and that leverages the three layers of the cloud computing stack: SaaS, PaaS, and IaaS. The Web service constitute the SaaS application that will store ECG data in the Amazon S3 service and issue a processing request to the scalable cloud platform. The runtime platform is composed of a dynamically sizable number of instances running the workflow engine and Aneka. The number of workflow engine instances is controlled according to the number of requests in the queue of each instance, while Aneka controls the number of EC2 instances used to execute the single tasks defined by the workflow engine for a single ECG processing job. Each of these jobs consists of a set of operations involving the extraction of the waveform from the heartbeat data and the comparison of the waveform with a reference waveform to detect anomalies. If anomalies are found, doctors and first-aid personnel can be notified to act on a specific patient.

Even though remote ECG monitoring does not necessarily require cloud technologies, cloud computing introduces opportunities that would be otherwise hardly achievable. The first advantage is the elasticity of the cloud infrastructure that can grow and shrink according to the requests served. As a result, doctors and hospitals do not have to invest in large computing infrastructures designed after capacity planning, thus making more effective use of budgets. The second advantage is ubiquity. Cloud computing technologies have now become easily accessible and promise to deliver systems with minimum or no downtime. Computing systems hosted in the cloud are accessible from any Internet device through simple interfaces (such as SOAP and REST-based Web services). This makes these systems not only ubiquitous, but they can also be easily integrated with other systems maintained on the hospital's premises.

Finally, cost savings constitute another reason for the use of cloud technology in healthcare. Cloud services are priced on a pay-per-use basis and with volume prices for large numbers of service requests. These two models provide a set of flexible options that can be used to price the service, thus actually charging costs based on effective use rather than capital costs.

Q8) Briefly explain architecture and overview of the Jeeva Portal

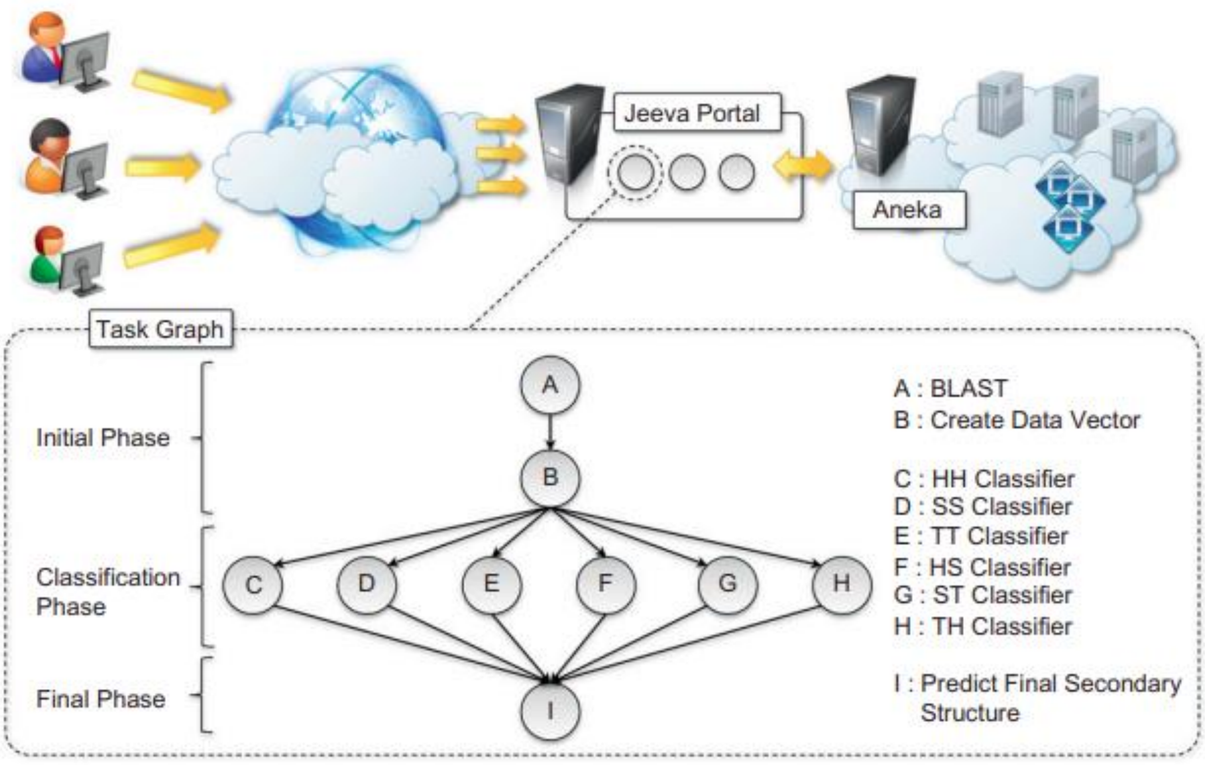

FIGURE 10.2

Architecture and overview of the Jeeva Portal.

One project that investigates the use of cloud technologies for protein structure prediction is Jeeva—an integrated Web portal that enables scientists to offload the prediction task to a computing cloud based on Aneka (see Figure 10.2). The prediction task uses machine learning techniques (support vector machines) for determining the secondary structure of proteins. These techniques translate the problem into one of pattern recognition, where a sequence has to be classified into one of three possible classes (E, H, and C). A popular implementation based on support vector machines divides the pattern recognition problem into three phases: initialization, classification, and a final phase. Even though these three phases have to be executed in sequence, it is possible to take advantage of parallel execution in the classification phase, where multiple classifiers are executed concurrently.

This creates the opportunity to sensibly reduce the computational time of the prediction. The prediction algorithm is then translated into a task graph that is submitted to Aneka. Once the task is completed, the middleware makes the results available for visualization through the portal. The advantage of using cloud technologies (i.e., Aneka as scalable cloud middleware) versus conventional grid infrastructures is the capability to leverage a scalable computing infrastructure that can be grown and shrunk on demand. This concept is distinctive of cloud technologies and constitutes a strategic advantage when applications are offered and delivered as a service.

Q9) Discuss social networking application Facebook

Facebook is probably the most evident and interesting environment in social networking. With more than 800 million users, it has become one of the largest Websites in the world. To sustain this incredible growth, it has been fundamental that Facebook be capable of continuously adding capacity and developing new scalable technologies and software systems while maintaining high performance to ensure a smooth user experience. Currently, the social network is backed by two data centers that have been built and optimized to reduce costs and impact on the environment. On top of this highly efficient infrastructure, built and designed out of inexpensive hardware, a completely customized stack of opportunely modified and refined open-source technologies constitutes the back-end of the largest social network.

Taken all together, these technologies constitute a powerful platform for developing cloud applications This platform primarily supports Facebook itself and offers APIs to integrate third-party applications with Facebook's core infrastructure to deliver additional services such as social games and quizzes created by others. The reference stack serving Facebook is based on LAMP (Linux, Apache, MySQL, and PHP). This collection of technologies is accompanied by a collection of other services developed in-house. These services are developed in a variety of languages and implement specific functionalities such as search, news feeds, notifications, and others. While serving page requests, the social graph of the user is composed. The social graph identifies a collection of interlinked information that is of relevance for a given user. Most of the user data are served by querying a distributed cluster of MySQL instances, which mostly contain key-value pairs. These data are then cached for faster retrieval. The rest of the relevant information is then composed together using the services mentioned before. These services are located closer to the data and developed in languages that provide better performance than PHP.

The development of services is facilitated by a set of internally developed tools. One of the core elements is Thrift. This is a collection of abstractions (and language bindings) that allow cross-language development. Thrift allows services developed in different languages to communicate and exchange data. Bindings for Thrift in different languages take care of data serialization and deserialization, communication, and client and server boilerplate code. This simplifies the work of the developers, who can quickly prototype services and leverage existing ones. Other relevant services and tools are Scribe, which aggregates streaming log feeds, and applications for alerting and monitoring.

Q10) Discuss Google Docs and Cloud Desktops EyeOS and XIOS/3

Google Docs

Google Docs is a SaaS application that delivers the basic office automation capabilities with support for collaborative editing over the Web. The application is executed on top of the Google distributed computing infrastructure, which allows the system to dynamically scale according to the number of users using the service. Google Docs allows users to create and edit text documents, spreadsheets, presentations, forms, and drawings. It aims to replace desktop products such as Microsoft Office and OpenOffice and provide similar interface and functionality as a cloud service. It supports collaborative editing over the Web for most of the applications included in the suite. This eliminates tedious emailing and synchronization tasks when documents need to be edited by multiple users. By being stored in the Google infrastructure, these documents are always available from anywhere and from any device that is connected to the Internet.

Moreover, the suite allows users to work offline if Internet connectivity is not available. Support for various formats such as those that are produced by the most popular desktop office solutions allows users to easily import and move documents in and out of Google Docs, thus eliminating barriers to the use of this application. Google Docs is a good example of what cloud computing can deliver to end users: ubiquitous access to resources, elasticity, absence of installation and maintenance costs, and delivery of core functionalities as a service

Cloud Desktops EyeOS and XIOS/3

EyeOS1 is one of the most popular Web desktop solutions based on cloud technologies. It replicates the functionalities of a classic desktop environment and comes with pre-installed applications for the most common file and document management tasks (see Figure 10.7). Single users can access the EyeOS desktop environment from anywhere and through any Internet-connected device, whereas organizations can create a private EyeOS Cloud on their premises to virtualize the desktop environment of their employees and centralize their management. The EyeOS architecture is quite simple:

On the server side, the EyeOS application maintains the information about user profiles and their data, and the client side constitutes the access point for users and administrators to interact with the system. EyeOS stores the data about users and applications on the server file system. Once the user has logged in by providing credentials, the desktop environment is rendered in the client's browser by downloading all the JavaScript libraries required to build the user interface and implement the core functionalities of EyeOS.

Each application loaded in the environment communicates with the server by using AJAX; this communication model is used to access user data as well as to perform application operations: editing documents, visualizing images, copying and saving files, sending emails, and chatting. EyeOS also provides APIs for developing new applications and integrating new capabilities into the system. EyeOS applications are server-side components that are defined by at least two files (stored in the eyeos/apps/appname directory): appname.php and appname.js. The first file defines and implements all the operations that the application exposes; the JavaScript file contains the code that needs to be loaded in the browser in order to provide user interaction with the application.

Xcerion XML Internet OS/3 (XIOS/3) is another example of a Web desktop environment. The service is delivered as part of the CloudMe application, which is a solution for cloud document storage. The key differentiator of XIOS/3 is its strong leverage of XML, used to implement many of the tasks of the OS: rendering user interfaces, defining application business logics, structuring file system organization, and even application development.The architecture of the OS concentrates most of the functionalities on the client side while implementing server-based functionalities by means of XML Web services. The client side renders the user interface, orchestrates processes, and provides data-binding capabilities on XML data that is exchanged with Web services.

The server is responsible for implementing core functions such as transaction management for documents edited in a collaborative mode and core logic of installed applications into the environment. XIOS/3 also provides an environment for developing applications (XIDE), which allows users to quickly develop complex applications by visual tools for the user interface and XML documents for business logic. XIOS/3 is released as open-source software and implements a marketplace where third parties can easily deploy applications that can be installed on top of the virtual desktop environment.

It is possible to develop any type of application and feed it with data accessible through XML Web services: developers have to define the user interface, bind UI components to service calls and operations, and provide the logic on how to process the data. XIDE will package this information into a proper set of XML documents, and the rest will be performed by an XML virtual machine implemented in XIOS. XIOS/3 is an advanced Web desktop environment that focuses on the integration of services into the environment by means of XML-based services and that simplifies collaboration with peers.