CMR
INSTITUTE OF
TECHNOLOGY

USN | 1 | C | | | | | | | | |

Internal Assessment Test 3 – May 2017

Sub:	Advanced Web Programming							Code:	13MCA43
Date:	31-05-17	Duration:	90 mins	Max Marks:	50	Sem:	IV	Branch:	MCA

Note: Answer any 5 questions. All questions carry equal marks. Total marks: 50

			Marks	OBE CO	OBE RBT
1	a.	Explain the string methods in Ruby.	[5]	CO5	L4

Ruby has many built in methods to work with strings. Strings in Ruby by default are mutable and can be changed in place or a new string can be returned from a method.

Length:

- The `.length` property returns the number of characters in a string including white-space.

```
"Hello".length
# returns: 5
"Hello World!".length
# returns: 12
```

Count:

- The `.count` method counts how many times a specific character(s) is found in a string.
- This method is case-sensitive.

```
"HELLO".count('L')
# returns: 2
"HELLO WORLD!".count('LO')
# returns: 1
```

Insert:

- The `.insert` method inserts a string into another string before a given index.

```
"Hello".insert(3, "hi5")
# returns:
Helhi5lo
# "hi5" is inserted into the string right before the second 'l' which is
at index 3
```

Upcase:

- The `.upcase` method transforms all letters in a string to uppercase.

```
"Hello".upcase
# returns:
HELLO
```

Downcase:

- The `.downcase` method transforms all letters in a string to lowercase.

```
"Hello".downcase
# returns:
hello
```

Capitilize:

- The `.capitalize` method make the first letter in a string uppercase and the rest of the string lowercase.

```
"HELLO".capitalize
# returns:
Hello
"HELLO, HOW ARE YOU?".capitalize
# returns:
Hello, how are you?
```

Note that the first letter is only capitilized if it is at the beginning of the string.

```
"-HELLO".capitalize
"1HELLO".capitalize
# returns:
-hello
1hello
```

Reverse:

- The `.reverse` method reverses the order of the characters in a string.

```
"Hello World!".reverse
# returns:
"!dlroW olleH"
```

Split:

- The `.split` takes a strings and *splits* it into an array, then returns the array.
- The default method splits the string based on whitespace, unless a different separator is provided (see second example).

```
"Hello, how are you?".split
# returns:
["Hello,", "how", "are", "you?"]
"H-e-l-l-o".split('-')
# returns:
["H", "e", "l", "l", "o"]
```

Chop:

- The `.chop` method removes the last character of the string.
- A new string is returned, unless you use the `.chop!` method which mutates the original string.

```
"Name".chop
# returns:
Nam
name = "Batman"
name.chop
name == "Batma" # returns false
name = "Batman"
name.chop!
name == "Batma" # returns true
```

Strip:

- The `.strip` method removes the leading and trailing whitespace on strings, including tabs, newlines, and carriage returns (`\t`, `\n`, `\r`).

```
"  Hello  ".strip
# returns:
Hello
```

Chomp:

- The `.chomp` method removes the last character in a string, only if it's a carriage return or newline (`\r`, `\n`).
- This method is commonly used with the `gets` command to remove returns from user input.

```
"hello\r".chomp
# returns:
hello
"hello\t".chomp
# returns:
hello\t #because tabs and other whitespace remain intact when using
`chomp`
```

To Integer:

- The `.to_i` method converts a string to an integer.

```
"15".to_i
# returns:
15 #integer
```

	b	Explain the various iterators in Ruby.
		
		The each iterator returns all the elements of an array or a hash.
		
		## Syntax
		``
		collection.each do |variable|
		   code
		end
		``
		Executes *code* for each element in *collection*. Here, *collection* could be an array or a ruby hash.
		
		## Example
		``
		#!/usr/bin/ruby
		
		ary = [1,2,3,4,5]
		ary.each do |i|
		   puts i
		end
		``
		# Ruby collect Iterator
		The *collect* iterator returns all the elements of a collection.
		
		## Syntax
		``
		collection = collection.collect
		``
		The *collect* method need not always be associated with a block. The *collect* method returns the entire collection, regardless of whether it is an array or a hash.
		
		## Example
		``
		#!/usr/bin/ruby
		
		a = [1,2,3,4,5]
		b = Array.new
		b = a.collect
		puts b
		``
			[5]	CO5	L4

2	a.	Explain Web services. Explain Web Service Architecture in detail. Web services are open standard (XML, SOAP, HTTP etc.) based Web applications that interact with other web applications for the purpose of exchanging data. A web service is a collection of open protocols and standards used for exchanging data between applications or systems. Software applications written in various programming languages and running on various platforms can use web services to exchange data over computer networks like the Internet in a manner similar to inter-process communication on a single	[10]	CO5	L4

computer.

Web services are self-contained, modular, distributed, dynamic applications that can be described, published, located, or invoked over the network to create products, processes, and supply chains. These applications can be local, distributed, or web-based. Web services are built on top of open standards such as TCP/IP, HTTP, Java, HTML, and XML.

Web services are XML-based information exchange systems that use the Internet for direct application-to-application interaction. These systems can include programs, objects, messages, or documents.

A web service is a collection of open protocols and standards used for exchanging data between applications or systems. Software applications written in various programming languages and running on various platforms can use web services to exchange data over computer networks like the Internet in a manner similar to inter-process communication on a single computer. This interoperability (e.g., between Java and Python, or Windows and Linux applications) is due to the use of open standards.

A complete web service is, therefore, any service that:

- Is available over the Internet or private (intranet) networks
- Uses a standardized XML messaging system
- Is not tied to any one operating system or programming language
- Is self-describing via a common XML grammar
- Is discoverable via a simple find mechanism

A web service is a method of communication between two devices over a network. There are two ways to view the web service architecture

1. To examine the individual roles of each service actor
2. Examine the emerging web service protocol stack

Web Service Roles

There are three major roles within the web service architecture:

1. Service Provider

This is the provider of the web service. The service provider implements the service and makes it available on the Internet.

2. Service Requestor

This is any consumer of the web service. The requestor utilizes an existing web service by opening a network connection and sending an XML request.

3. Service Registry

This is a logically centralized directory of services. The registry provides a central place where developers can publish new services or find existing ones. It therefore serves as a centralized clearing house for companies and their services.

Web Service Protocol Stack

		A second option for viewing the web service architecture is to examine the emerging web service protocol stack. The stack is still evolving, but currently has four main layers. 1. Service Transport This layer is responsible for transporting messages between applications. Currently, this layer includes Hyper Text Transport Protocol (HTTP), Simple Mail Transfer Protocol (SMTP), File Transfer Protocol (FTP), and newer protocols such as Blocks Extensible Exchange Protocol (BEEP). 2. XML Messaging This layer is responsible for encoding messages in a common XML format so that messages can be understood at either end. Currently, this layer includes XML-RPC and SOAP. 3. Service Description This layer is responsible for describing the public interface to a specific web service. Currently, service description is handled via the Web Service Description Language (WSDL). 4. Service Discovery This layer is responsible for centralizing services into a common registry and providing easy publish/find functionality. Currently, service discovery is handled via Universal Description, Discovery, and Integration (UDDI). As web services evolve, additional layers may be added and additional technologies may be added to each layer.			
3	a.	Build a Rails application to find the product of two integers and return the result to the client . **Step 3 : Click on "Git bash"** The CMD prompt points to $ with "/c/sites" directory **Step 4:** create new bookstore using the following syntax **$ rails new addnumbers** **Step 5: C**hange the directory the following syntax $ **cd addnumbers** **Step 6:** Run the server using following command ***$ rails server*** Then open Firefox Web browser and type http://localhost:3000 Which gives ruby on rails welcome page. **Step 9:** Use rake command to flush function paths to routes.rb **$ rake routes** **Step 10:** Run the server again ***$ rails server*** **Step 11:** Creating a new controller which helps to enter numbers ***$ rails generate controller addnu result***	[10]	CO5	L4

		Go to C:\Sites\addnumbers\app\controllers\ addnu_controller.rb **Add this code in this page** **class AddnuController < ApplicationController** **def result** **@a = params[:a]** **@b = params[:b]** **@c = @a.to_i * @b.to_i** **end** **end** **Go to C:\Sites\addnumbers\app\views\addnu\ result.html.erb** **Add this code here** **<html>** **<title> welcome template for books </title>** **<body>** ** sum = <%=@c%> ** **<form action="result">** **<input type="text" name="a" />** **<input type="text" name="b" />** **<input type=submit value="search" />** **</form>** **</body>** **</html>** **Run the server again using** **Go to Gitbash and type** **rails server** **Go to browser and type http://localhost:3000/addnu/result**			
4	a.	Explain pattern matching in Ruby. Explain Remembering and substitution pattern matching with examples A *regular expression* is a special sequence of characters that helps you match or find other strings or sets of strings using a specialized syntax held in a pattern. Remembering pattern matching $1, $2 implicit variables store the matches Substitution Sub and gsub	[6]	CO5	L4
	b.	Explain SOAP message structure. <?xml version="1.0"?> <SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2001/12/soap-envelope" SOAP-ENV:encodingStyle="http://www.w3.org/2001/12/soap-encoding">	[4]	CO5	L4

```
<SOAP-ENV:Header>
  ...
  ...
</SOAP-ENV:Header>

<SOAP-ENV:Body>
  ...
  ...
  <SOAP-ENV:Fault>
    ...
    ...
  </SOAP-ENV:Fault>
  ...
</SOAP-ENV:Body>

</SOAP_ENV:Envelope>
```

5 a. Explain Rails MVC Architecture with a neat diagram. [10] CO5 L4

The **M**odel **V**iew **C**ontroller principle divides the work of an application into three separate but closely cooperative subsystems.

Model (ActiveRecord)

It maintains the relationship between the objects and the database and handles validation, association, transactions, and more.

This subsystem is implemented in ActiveRecord library, which provides an interface and binding between the tables in a relational database and the Ruby program code that manipulates database records. Ruby method names are automatically generated from the field names of database tables.

View (ActionView)

It is a presentation of data in a particular format, triggered by a controller's decision to present the data. They are script-based template systems like JSP, ASP, PHP, and very easy to integrate with AJAX technology.

This subsystem is implemented in ActionView library, which is an **Embedded Ruby** (ERb) based system for defining presentation templates for data presentation. Every Web connection to a Rails application results in the displaying of a view.

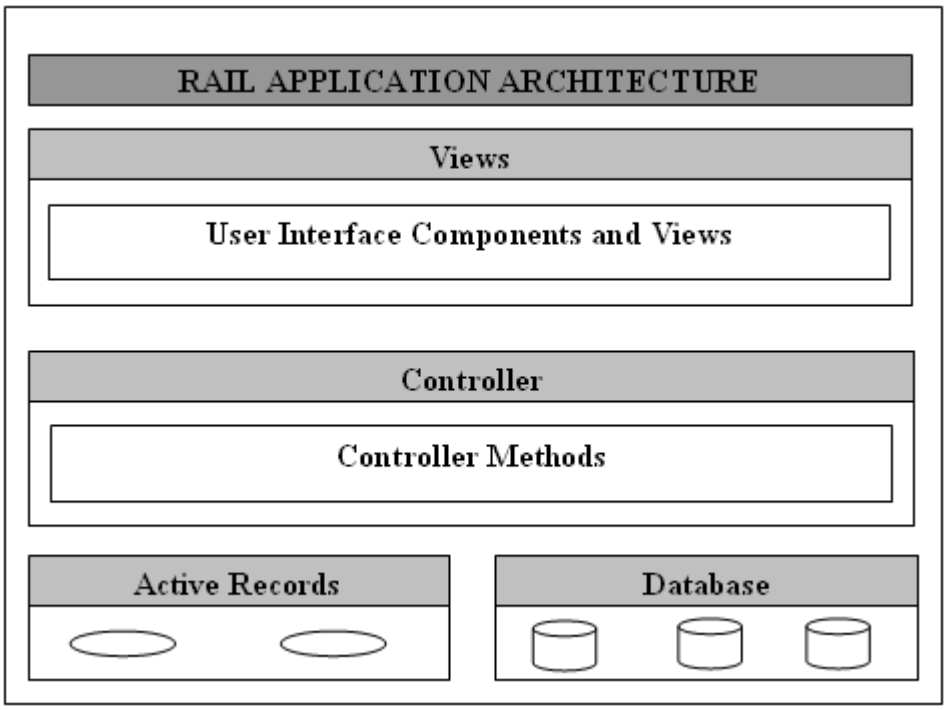
Controller (ActionController)

The facility within the application that directs traffic, on the one hand, querying the models for specific data, and on the other hand, organizing that data (searching, sorting, messaging it) into a form that fits the needs of a given view.

This subsystem is implemented in ActionController, which is a data broker sitting between ActiveRecord (the database interface) and ActionView (the presentation engine).

Pictorial Representation of MVC Framework

Given below is a pictorial representation of Ruby on Rails Framework –



6	a.	Explain WEB2.0. Explain the key features of web 2.0	[10]	CO5	L4

Web 2.0 describes World Wide Web sites that emphasize user-generated content, usability, and interoperability. A Web 2.0 site may allow users to interact and collaborate with each other in a social media dialogue as creators of user-generated content in a virtual community, in contrast to Web sites where people are limited to the passive viewing of content. Examples of Web 2.0 include social networking sites, blogs, wikis, folksonomies, video sharing sites, hosted services, Web applications, and mashups.

Web 2.0 is the current state of online technology as it compares to the early days of the Web, characterized by greater user interactivity and collaboration, more pervasive network connectivity and enhanced communication channels. One of the most significant differences between Web 2.0 and the traditional World Wide Web (WWW, retroactively referred to as Web 1.0) is greater collaboration among Internet users, content providers and enterprises. Originally, data was posted on Web sites, and users simply viewed or downloaded the content. Increasingly, users have more input into the nature and scope of Web content and in some cases exert real-time control over it. The social nature of Web 2.0 is another major difference between it and the original, static Web. Increasingly, websites enable community-based input, interaction, content-sharing and collaboration. Types of social media sites and applications include forums, microblogging, social networking, social bookmarking, social curation, and wikis.

Key Features of WEB 2.0

Folksonomy : Free Classification of Information

Rich User Experience

User as a Contributor

Long Tail

		User Participation Basic Trust Dispersion			
7	a.	Explain SAAS & Social networking SAAS – Software As A Service Software as a Service (SaaS) is a software distribution model in which applications are hosted by a vendor or service provider and made available to customers over a network, typically the Internet. SaaS is becoming an increasingly prevalent delivery model as underlying technologies that support Web services and service-oriented architecture (SOA) mature and new developmental approaches, such as Ajax, become popular. Meanwhile, broadband service has become increasingly available to support user access from more areas around the world. SaaS is typically accessed by users using a thin client via a web browser. SaaS has become a common delivery model for many business applications, including office and messaging software etc SaaS applications are often updated more frequently than traditional software,[18] in many cases on a weekly or monthly basis. This is enabled by several factors: • The application is hosted centrally, so an update is decided and executed by the provider, not by customers. • The application only has a single configuration, making development testing faster. • The application vendor does not have to expend resources updating and maintaining backdated versions of the software, because there is only a single version.[19] • The application vendor has access to all customer data, expediting design and regression testing. • The solution provider has access to user behavior within the application The social network is a theoretical construct useful in the social sciences to study relationships between individuals, groups, organizations, or even entire societies (social units, see differentiation). A **social network** is a social structure made up of a set of social actors (such as individuals or organizations), sets of dyadic ties, and other social interactions between actors. The social network perspective provides a set of methods for analyzing the structure of whole social entities as well as a variety of theories explaining the patterns observed in these structures. Social networks and the analysis of them is an inherently interdisciplinary academic field which emerged from social psychology ,sociology, statistics, and graph theory. The term is used to describe a social structure determined by such interactions. The ties through which any given social unit connects represent the convergence of the various social contacts of that unit. This theoretical approach is, necessarily, relational. An axiom of the social	[10]	CO5	L4

network approach to understanding social interaction is that social phenomena should be primarily conceived and investigated through the properties of relations between and within units, instead of the properties of these units themselves.

Thus, one common criticism of social network theory is that individual agency is often ignored.

A **social networking service** (also **social networking site**, **SNS** or **social media**) is a platform to build social networks or social relations among people who share similar interests, activities, backgrounds or real-life connections. The variety of stand-alone and built-in social networking services currently available in the online space introduces challenges of definition.

8	a.	Explain arrays in Ruby. With examples, explain the various array functions	[10]	CO5	L4

Ruby arrays are ordered, integer-indexed collections of any object. Each element in an array is associated with and referred to by an index.

Ruby arrays can hold objects such as String, Integer, Fixnum, Hash, Symbol, even other Array objects. Ruby arrays are not as rigid as arrays in other languages. Ruby arrays grow automatically while adding elements to them.

Creating Arrays:

There are many ways to create or initialize an array. One way is with the *new*class method:

```
names = Array.new
```

You can set the size of an array at the time of creating array:

```
names = Array.new(20)
```

```
#!/usr/bin/ruby


names = Array.new(20)

puts names.size  # This returns 20

puts names.length # This also returns 20


#!/usr/bin/ruby

names = Array.new(4, "mac")

puts "#{names}"
```

Array Built-in Methods:

```
#!/usr/bin/ruby

digits = Array(0..9)

num = digits.at(6)

puts "#{num}"
```


Course Outcomes		PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8
CO1:	Develop Web apps using various development languages and tools	1	-	3	-	-	-	3	3
CO2:	Build the ability to select the essential technology needed to develop and implement web applications	2	2		-	-	1	2	3
CO3:	Design dynamic web applications using PERL CGI - MySQL		3	3	1	-	1	3	3

CO4:	Design dynamic web applications using PHP MySQL	-	-	3	2	-	-	3	3
CO5:	Ruby Rails application development	1	-	2	-	-	-	3	3
CO6:	Develop Web apps using various development languages and tools	-	-	-	1	2	2	-	-

Cognitive level	KEYWORDS
L1	List, define, tell, describe, identify, show, label, collect, examine, tabulate, quote, name, who, when, where, etc.
L2	summarize, describe, interpret, contrast, predict, associate, distinguish, estimate, differentiate, discuss, extend
L3	Apply, demonstrate, calculate, complete, illustrate, show, solve, examine, modify, relate, change, classify, experiment, discover.
L4	Analyze, separate, order, explain, connect, classify, arrange, divide, compare, select, explain, infer.
L5	Assess, decide, rank, grade, test, measure, recommend, convince, select, judge, explain, discriminate, support, conclude, compare, summarize.