

USN

--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--



Internal Assessment Test 3 – May 2018

Sub:	Advanced Web Programming	Sub Code:	16MCA12	Branch:	MCA
Date:	22.05.2018	Duration:	90 mins	Max Marks:	50
		Sem / Sec:	IV A & B		OBE

Answer Any One FULL Question from each part.

	MARKS	CO	RBT
Part – 1			
<p>1 (a) What is JQuery . Explain the various JQuery selectors with examples.</p> <p>motto: Write less, do more. JQuery simplifies HTML document traversing, event handling, animating, and Ajax interactions for rapid web development. JQuery is a JavaScript toolkit designed to simplify various tasks by writing less code. Here is the list of important core features supported by JQuery:</p> <ul style="list-style-type: none"> • DOM manipulation: The JQuery made it easy to select DOM elements, negotiate them and modifying their content by using cross-browser open source selector engine called Sizzle. • Event handling: The JQuery offers an elegant way to capture a wide variety of events, such as a user clicking on a link, without the need to clutter the HTML code itself with event handlers. • AJAX Support: The JQuery helps you a lot to develop a responsive and featurerich site using AJAX technology. • Animations: The JQuery comes with plenty of built-in animation effects which you can use in your websites. • Lightweight: The JQuery is very lightweight library - about 19KB in size (Minified and gzipped). • Cross Browser Support: The JQuery has cross-browser support, and works well in IE 6.0+, FF 2.0+, Safari 3.0+, Chrome and Opera 9.0+ • Latest Technology: The JQuery supports CSS3 selectors and basic XPath syntax. 	[10]	CO1	L4
<p>2 (a) Write a JQuery script to limit character input in the text area including count of characters</p> <pre> <!DOCTYPE html> <html> <head> <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></scrip t> <script> var maxlength = 15; \$(document).ready(function(){ \$('#textarea').keypress(function() { var textlen = maxlength - \$(this).val().length; \$('#rchars').text(textlen); }); }); </pre>	[10]	CO1	L4

```

    });
  });

</script>
</head>
<body>
<form>
<label>Maximum 15 characters</label>
<textarea id="textarea" maxlength="15"></textarea>
  <span id="rchars">15</span> Character(s) Remaining
</form>
</body>

</html>

```

Part – II

3 (a) Explain file handling functions in PHP with examples
 FOPEN(filename, mode)
 Fread/fgets
 Fputs
 Fclose

[10]

CO1 L4

4 (a) Explain in brief PHP's XML facilities.

[10]

CO1 L4

5.xml

```

<?xml version="1.0" encoding="ISO-8859-1"?>
<student_info>
<student>
<usn>1CR17MCA01</usn>
<name>Ajay</name>
</student>
<student>
<usn>1CR17MCA02</usn>
<name>Akshatha</name>
</student>
<student>
<usn>1CR17MCA58</usn>
<name>Piyush</name>
</student>
<student>
<usn>1CR17MCA59</usn>
<name>Taj</name>
</student>
</student_info>

```

5.php

```

<html>
<body>

```

```
<form name="form1" method="post"
action="http://localhost//WEBDEMO/5.php">
    Enter Name <input type="text" name="stname">
    <input type="submit" name="submit" value="search">
</form>
</body>
</html>
```

```
<?php
$con = mysql_connect("localhost","root","");
if (!$con)
{
    die('Could not connect: ' . mysql_error());
}

mysql_select_db("web1", $con);
$lib = simplexml_load_file("5.xml");

$i = "delete from student";
$result = mysql_query($i);

foreach($lib as $stu)
{
    $usn= $stu->usn;
    $name=$stu->name;
    $i="insert into student(usn,name) values('$usn','$name)";
    mysql_query($i);
}

if(($_SERVER["REQUEST_METHOD"]=="POST") || ($_SERVER["REQUEST_METHOD"]=="post
"))
{
    $stname = $_POST["stname"];

    $result = mysql_query("SELECT * from student where name LIKE '%" . $stname . "%'");
    echo "<table border='1'><tr><th>USN</th><th>Name</th></tr>";

    while($row = mysql_fetch_array($result))
    {
        echo "<tr><td>" . $row['usn'] . "</td><td>" . $row['name'] . "</td></tr>";
    }
    echo "</table>";
}
?>
```


Part - III

- 5 (a) Explain built in methods for arrays and lists in Ruby.
Push, pop, shift, unshift, reverse, merge,
Addition, subtraction, union, intersection

[10] CO2 L4

- 6 (a) **Explain the directory structure for the Rails application?**

[10] CO2 L4

When you use the Rails helper script to create your application, it creates the entire directory structure for the application. Rails knows where to find things it needs within this structure, so you don't have to provide any input.

















Here is a top-level view of a directory tree created by the helper script at the time of application creation. Except for minor changes between releases, every Rails project will have the same structure, with the same naming conventions. This consistency gives you a tremendous advantage; you can quickly move between Rails projects without relearning the project's organization.

When we type

rails new bookstore

It creates a new application with the name bookstore.

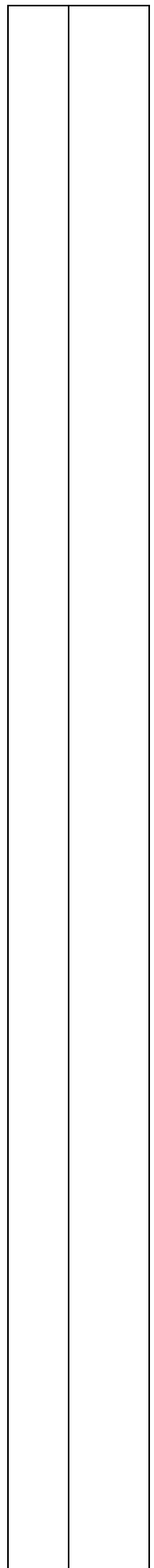
You will find a directory structure in Windows as follows –

 app	7/1/2015 5:06 PM	File folder
 bin	7/1/2015 5:06 PM	File folder
 config	7/1/2015 5:06 PM	File folder
 db	7/1/2015 5:06 PM	File folder
 lib	7/1/2015 5:06 PM	File folder
 log	7/1/2015 5:06 PM	File folder
 public	7/1/2015 5:06 PM	File folder
 test	7/1/2015 5:06 PM	File folder
 tmp	7/1/2015 5:06 PM	File folder
 vendor	7/1/2015 5:06 PM	File folder
 .gitignore	7/1/2015 5:06 PM	GITIGNORE File
 config.ru	7/1/2015 5:06 PM	RU File
 Gemfile	7/1/2015 5:06 PM	File
 Gemfile.lock	7/1/2015 5:07 PM	LOCK File
 Rakefile	7/1/2015 5:06 PM	File
 README.rdoc	7/1/2015 5:06 PM	RDOC File

- **app** – It organizes your application components. It's got subdirectories that hold the view (views and helpers), controller (controllers), and the backend business logic (models).
- **app/controllers** – The controllers subdirectory is where Rails looks to find

the controller classes. A controller handles a web request from the user.

- **app/helpers** – The helpers subdirectory holds any helper classes used to assist the model, view, and controller classes. This helps to keep the model, view, and controller code small, focused, and uncluttered.
- **app/models** – The models subdirectory holds the classes that model and wrap the data stored in our application's database. In most frameworks, this part of the application can grow pretty messy, tedious, verbose, and error-prone. Rails makes it dead simple!
- **app/view** – The views subdirectory holds the display templates to fill in with data from our application, convert to HTML, and return to the user's browser.
- **app/view/layouts** – Holds the template files for layouts to be used with views. This models the common header/footer method of wrapping views. In your views, define a layout using the `<tt>layout:default</tt>` and create a file named `default.html.erb`. Inside `default.html.erb`, call `<% yield %>` to render the view using this layout.
- **components** – This directory holds components, tiny self-contained applications that bundle model, view, and controller.
- **config** – This directory contains the small amount of configuration code that your application will need, including your database configuration (in `database.yml`), your Rails environment structure (`environment.rb`), and routing of incoming web requests (`routes.rb`). You can also tailor the behavior of the three Rails environments for test, development, and deployment with files found in the `environments` directory.
- **db** – Usually, your Rails application will have model objects that access relational database tables. You can manage the relational database with scripts you create and place in this directory.
- **doc** – Ruby has a framework, called RubyDoc, that can automatically generate documentation for code you create. You can assist RubyDoc with comments in your code. This directory holds all the RubyDoc-generated Rails and application documentation.
- **lib** – You'll put libraries here, unless they explicitly belong elsewhere (such as vendor libraries).
- **log** – Error logs go here. Rails creates scripts that help you manage various error logs. You'll find separate logs for the server (`server.log`) and each Rails environment (`development.log`, `test.log`, and `production.log`).
- **public** – Like the public directory for a web server, this directory has web files that don't change, such as JavaScript files (`public/javascripts`), graphics



(public/images), stylesheets (public/stylesheets), and HTML files (public).

- **script** – This directory holds scripts to launch and manage the various tools that you'll use with Rails. For example, there are scripts to generate code (generate) and launch the web server (server).
- **test** – The tests you write and those that Rails creates for you, all goes here. You'll see a subdirectory for mocks (mocks), unit tests (unit), fixtures (fixtures), and functional tests (functional).
- **tmp** – Rails uses this directory to hold temporary files for intermediate processing.
- **vendor** – Libraries provided by third-party vendors (such as security libraries or database utilities beyond the basic Rails distribution) go here.

Apart from these directories, there will be two files available in bookstore directory.

- **README** – This file contains a basic detail about Rail Application and description of the directory structure explained above.
- **Rakefile** – This file is similar to Unix Makefile, which helps with building, packaging and testing the Rails code. This will be used by rake utility supplied along with the Ruby installation.

Part – IV

7 (a) What is Web2.0 . Explain in detail folksonomy and convergence.

[10] CO1 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.

Elements of Web 2.0

- Wikis: Websites that enable users to contribute, collaborate and edit site content. Wikipedia is one of the oldest and best-known wiki-based sites.
- The increasing prevalence of Software as a Service ([SaaS](#)), [web apps](#) and [cloud computing](#) rather than locally-installed programs and services.
- Mobile computing, the trend toward users connecting from wherever they may be. That trend is enabled by the proliferation of smartphones, tablets and other mobile devices in conjunction with readily accessible Wi-Fi networks.
- [Mash-ups](#): Web pages or applications that integrate complementary elements from two or more sources.
- Social networking: The practice of expanding the number of one's business and/or social contacts by making connections through individuals. Social networking sites include [Facebook](#), [Twitter](#), [LinkedIn](#) and [Google+](#).
- Collaborative efforts based on the ability to reach large numbers of participants and their collective resources, such as [crowdsourcing](#), [crowdfunding](#) and [crowdsourcing testing](#).
- User-generated content ([UGC](#)): Writing, images, audio and video content -- among other possibilities -- made freely available online by the individuals who create it.
- Unified communications ([UC](#)): The integration of multiple forms of call and multimedia/cross-media message-management functions controlled by an individual user for both business and social purposes.
- Social curation: The collaborative sharing of content organized around one or more particular themes or topics. Social content curation sites include [Reddit](#), [Digg](#), [Pinterest](#) and [Instagram](#).

8 (a) Explain in detail SOAP, WSDL and REST.

[10] CO1 L4

The following list specifies the features of SOAP:

- SOAP is a communication protocol.
- SOAP is used for communication between applications.
- SOAP is a format for sending messages.
- SOAP is designed to communicate via Internet.
- SOAP is platform independent.
- SOAP is language independent.

- SOAP is simple and extensible.
- SOAP allows you to get around firewalls.
- SOAP will be developed as a W3C standard.

WSDL stands for Web Services Description Language. It is a xml document containing information about web services such as method name, method parameter etc.

REST stands for REpresentational State Transfer. It is a architectural style. It is not a protocol like SOAP.

Part - V

9 (a) What is D3. Explain SVG and the various shapes that can be created with neat example. [10]

D3.js is an open source JavaScript library that provides the facility for manipulating HTML documents based upon data, using JavaScript as the language for implementing the mapping of data to the documents. Hence, the name **D3 (Data Driven Documents)**. Many consider D3.js as a data visualization library. This may be correct, but D3.JS provides much more to its user than just visualization, such as:

Efficient selection of items in the HTML DOM.

Binding of data to visual elements.

Specifications on handling the addition and removal of data items.

The ability to style DOM elements dynamically.

Definition of an interaction model for the user with the data.

The ability to specify transitions between data visualizations based upon dynamic changes in data.

D3.js helps you bring data to life using **HTML**, **SVG**, and **CSS**. It focuses on the data, the way it is presented to the user, the changes in visualization with changes in data, and the way the user interacts with data through the visualization.

We are about to start on a fabulous journey of discovery with creating rich data visualizations with D3.js, and focusing on project-based learning of D3.js through practical examples. We will start out with the basic concepts, and then move through various examples of creating living data visualizations with D3.js.

In this first chapter, we will start with a brief overview of several of the concepts in D3.js, create a minimal D3.js application, and examine several of the tools that you can use to build D3.js applications..

We are about to start on a fabulous journey of discovery with creating rich data visualizations with D3.js, and focusing on project-based learning of D3.js through practical examples. We will start out with the basic concepts, and then move through various examples of creating living data visualizations with D3.js.

10.(a) Write a program to build a sample subway train status board. [10]

```
{
  "count": 26774.09756097561,
  "id": 1,
  "name": "Robert F. Kennedy Bridge Bronx Plaza"
}
```

```
<!DOCTYPE html>
<html>
<head>
```

CO1	L4
CO1	L4
CO1	L4


```
<meta charset="utf-8">
<style type="text/css">
div.chart
{
font-family:sans-serif;
font-size:0.7em;
}

div.bar
{
background-color:DarkRed;
color:white;
height:3em;
line-height:3em;
padding-right:1em;
margin-bottom:2px;
text-align:right;
}

</style>
<script src="d3.min.js"></script>
<script>
function draw(data) {
    "use strict"
    d3.select("body")
    .append("div")
    .attr("class","chart")
    .selectAll(".bar")
    .data(data)
    .enter()
    .append("div")
    .attr("class","bar")
    .style("width", function(d){return d.count/100 + "px"}) //267.740
    .style("outline", "1px solid black")
    .text(function(d){return d.name + " " + Math.round(d.count)});
}
</script>
</head>
<body>
<script>
d3.json("plaza_traffic.json", draw);
</script>
</body>
</html>
```