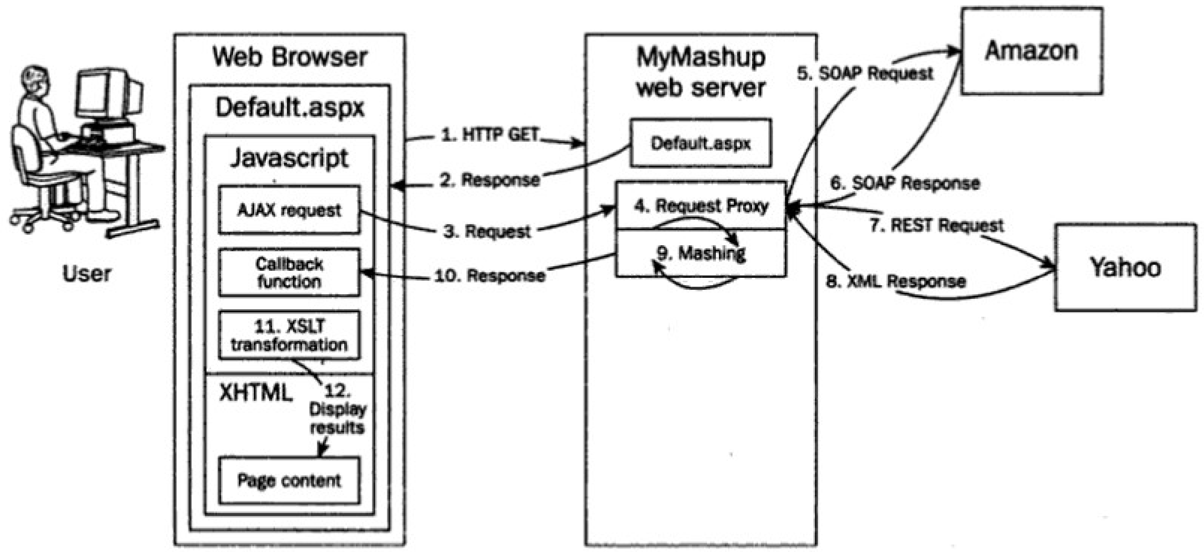
1. Explain in detail RUI implementation using AJAX with neat diagram.

The architecture of RUI using AJAX works in the following steps:



1. The web browser uses HTTP to send request for a page to the server which responds to the request in a standard manner
2. At this stage. No mashup content is present when the page is first time loaded by the browser. The browser downloads some JavaScript and the HTML for the page.
3. The browser issues another request to the server for additional content which is responded by the server in an asynchronous manner.
4. The server acts as a proxy and is responsible for forwarding requests coming in from the browser to the intended recipients
5. Amazon receives a SOAP request.

- As shown in fig. the architecture proceeds in further steps, where Amazon responds the request with appropriate data.
- A REST request is forwarded by the server to Yahoo, which returns the response in an XML format.
- The web server then performs all the mashing operations, combines the responses and then sends back to the browser in the standard format.

2. Explain characteristics of RIA

- **Direct interaction:** In a traditional page-based Web application, interaction is limited to a small group of standard controls: checkboxes, radio buttons and form fields. This severely hampers the creation of usable and engaging applications. An RIA can use a wider range of controls that allow greater efficiency and enhance the user experience. In RIAs, for example, users can interact directly with page elements through editing or drag-and-drop tools. They can also do things like pan across a map or other image.

- **Partial-page updating:** Standard HTML-based Web pages are loaded once. If you update something on a page, the change must be sent back to the server, which makes the changes and then resends the entire page. There's no other way to do it with HTTP and HTML. With traditional Web-based apps, network connectivity issues, processing limitations and other problems require users to wait while the entire page reloads. Even with broadband connections, wait times can be long and disruptive. But RIAs incorporate additional technologies, such as real-time streaming, high-performance client-side virtual machines, and local caching mechanisms that reduce latency (wait times) and increase responsiveness. A number of commercial development tools (see below) permit this partial-page updating.

- **Better feedback:** Because of their ability to change parts of pages without reloading, RIAs can provide the user with fast and accurate feedback, real-time confirmation of actions and choices, and informative and detailed error messages.

- **Consistency of look and feel:** With RIA tools, the user interface and experience with different browsers and operating systems can be more carefully controlled and made consistent.

- **Offline use**: When connectivity is unavailable, it might still be possible to use an RIA if the app is designed to retain its state locally on the client machine. (Developments in Web standards have also made it possible for some traditional Web applications to do that.)

- **Performance impact:** Depending on the application and network characteristics, RIAs can often perform better than traditional apps. In particular, applications that avoid round trips to the server by processing locally on the client are likely to be noticeably faster. Offloading such processing to the client machines can also improve server performance. The downside is that small, embedded and mobile devices -- which are increasingly common -- may not have the resources necessary to use such apps.

3. Write the code to create database connectivity using PHP.

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
```

```
// Create connection
$conn = new mysqli($servername, $username, $password);

// Check connection
if ($conn->connect_error) {
   die("Connection failed: " . $conn->connect_error);
}
echo "Connected successfully";
?>
```

4. Write a program to accept username and password fields from the user in form; store and display it using PHP-MySQL.

Index.php

```
<!DOCTYPE html>
<html lang="en" dir="ltr">
  <head>
    <title></title>
  </head>
  <style>
    .err{
      opacity: 0;
      color: red;
    }
  </style>
  <body>
    <fieldset>
      <legend>Login</legend>
            <form class="" onsubmit="return validate()" action="insert.php" method="post">
      Username : <input type="text" name="username">
      <span class="err">* enter username</span>
      <br><br>
      Password : <input type="password" name="password">
      <span class="err">* enter password</span>
      <br><br>
      <input type="submit" name="submit" value="submit">
    </form>
    </fieldset>
    <script>
    function validate(){
      var username = document.getElementsByName('username')[0];
      var password = document.getElementsByName('password')[0];
      if(username.value==''){
        username.nextElementSibling.style.opacity='1';
        return false;
      }
      else{
        username.nextElementSibling.style.opacity='0';
      }
      if(password.value==''){
        password.nextElementSibling.style.opacity='1';
        return false;
      }
      else{
        password.nextElementSibling.style.opacity='0';
      }
    }
    </script>
  </body></html>
```

insert.php

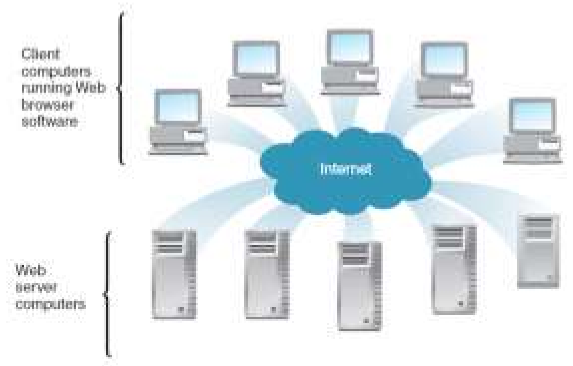
```
<!DOCTYPE html>
<html lang="en" dir="ltr">
  <head>
    <title>Display</title>
  </head>
  <body>
    <?php
      $username = '';
      $password = '';
      $usernameErr = $passwordErr = '';
      if(isset($_POST["submit"])){
          $username = $_POST['username'];
          $username = trim($username);
          $password = $_POST['password'];
      }

      $servername = "localhost";
      $username2 = "root";
      $password2 = "";
      $db = "mydb";

      $conn = mysqli_connect($servername,$username2,$password2,$db);
      if (!$conn) {
        die("Connection failed: " . mysqli_connect_error());
      }

      //check if username already exists
      $sql = "select username from login where username='$username'";
      $check = mysqli_query($conn,$sql);
      if(mysqli_num_rows($check) > 0){
        echo "Username already exists!!";
                      exit();
      }
      else {
        //insert into database
        $sql1 = "insert into login(username,password) values('$username','$password')";
        mysqli_query($conn,$sql1);
      }
      //dispay username and password
      $sql2 = "select * from login where username='$username'";
      $result = mysqli_query($conn,$sql2);
      $row = mysqli_fetch_assoc($result);
      echo "Username : ".$row['username']."<br>";
      echo "Password : ".$row['password']."<br>";
      mysqli_close($conn);
     ?>
  </body>
</html>
```

5. Write a program for selecting data from database and displaying it

on your browser using PHP.

```
<!DOCTYPE html>
<html lang="en" dir="ltr">
 <head>
 <title>Display</title>
 </head>
 <body>
 <?php
 $servername = "localhost";
 $username = "root";
 $password ="";
 $db = "mydb";
 $conn = mysqli_connect($servername,$username,$password,$db);
 if(!$conn){
 die("connection failed ".mysqli_connect_error());
 }
 $sql = "select * from users";
 $result = mysqli_query($conn,$sql);
 if(mysqli_num_rows($result) > 0){
 while($rows = mysqli_fetch_assoc($result)){
 echo "Id : ".$rows['id']."&emsp;"."Name : ".$rows['firstname']."
".$rows['lastname']."&emsp;"." Roll No. : ".$rows['rollNo']."<br><br>";
 }
 }
 mysqli_close($conn);
 ?>
 </body>
</html>
```

6. Explain the function of web server.



1) The client’s browser divides the URL into diff. parts dividing including address, path name and protocol.

2) DNS translates the domain name into the corresponding IP address .the numeric combination represents the site’s true address on the internet.

3) The browser now decides which protocol should be used .a protocol in common parlance is a language which the client’s to communicate with the server. FTP,HTTP are some such protocols.

4) The server sends a GET request to the web server to retrieve the address it has been given. It verifies given address exists, finds necessary files ,runs appropriate scripts exchanges cookies if necessary and returns back to the browser.

5) The browser now converts the data into HTML and displays results to the user. If it does not locate it sends an error message to the browser and to the client.

7. What are XML sitemap? Name some different types of sitemaps. Also explain main benefits of using XML sitemaps.

A Sitemap is an XML file that lists the URLs for a site. It allows webmasters to include additional information about each URL: when it was last updated, how often it changes, and how important it is in relation to other URLs in the site. This allows search engines to crawl the site more intelligently

There are up to six different kinds of sitemaps. HTML- and XML-sitemaps are the two sitemap-types that are seen most often.

1. HTML-Sitemap
2. XML-Sitemap
3. Image-Sitemap
4. Video-Sitemap
5. News-Sitemap
6. Mobile-Sitemap

Sitemaps are particularly beneficial on websites where:

- some areas of the website are not available through the browsable interface
- webmasters use rich Ajax, Silverlight, or Flash content that is not normally processed by search engines.
- The site is very large and there is a chance for the web crawlers to overlook some of the new or recently updated content
- When websites have a huge number of pages that are isolated or not well linked together, or
- When a website has few external links

8. What is python?

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

9. Short note on Django framework.

Django is a high-level Python Web framework encouraging rapid development and pragmatic, clean design. A web application framework is a toolkit of components all web applications need. The goal here is to allow developers to instead of implementing the same solutions over and over again, focus on the parts of their application that are new and unique to their project. In fact, Django is much more fully featured than many other frameworks out there. It takes care of a lot of the hassle of Web development, letting you focus on writing your application without any need to reinvent the wheel. It’s free and open source. Additionally, the Django framework enables you to model your domain and code classes, and before you know it, you already have an ORM. Let’s take a closer look to understand its acclaim better.

10. Write a code for “Hello World” using python.

```
# This program prints Hello, world!
print('Hello, world!')
```