

2.3 Microsoft Access :

Microsoft Access is a pseudo-relational database engine from Microsoft. Access uses the Jet Database Engine for data storage. Access is used for both small and large database deployments. This is partly due to its easy-to-use graphical interface, as well as its interoperability with other applications and platforms such as Microsoft's own SQL Server database engine and Visual Basic for Applications (VBA).

HOW TO INSTALL MICROSOFT ACCESS : Microsoft access is a part of the Microsoft Office suite of applications that also includes Word, Outlook and Excel, among others.

HOW TO LAUNCH :

Click on start and then find the microsoft office there can get microsoft access, click on that.

2.4.WAMP Server :

WampServer refers to a [software stack](#) for the [Microsoft Windows](#) operating system, created by Romain Bourdon and consisting of the [Apache web server](#), [OpenSSL](#) for SSL support, [MySQL](#) database and [PHP](#) programming language.

INSTALLING WAMP SERVER :

Step 1: Download the WAMP Server

Go to the official website www.wampserver.com/en/ and download the WampServer setup. There are two versions of WampServer are available i.e. 64-bits (x64) and 32-bits (x86), choose according to your computer's configuration.

Step 2: Initiate WAMP Server Install Process

Step 3: Select Location/Destination to Install WAMP

Step 4: Select Start Menu Folder to Install WAMP

Step 5: Ready to Install WAMP

Step 6: WAMPInstallation Complete

2.5 Jersey Java API :

Jersey RESTful Web Services framework is open source, production quality, framework for developing RESTful Web Services in Java that provides support for JAX-RS APIs and serves as a JAX-RS (JSR 311 & JSR 339) Reference Implementation.

Download the Jersey Jar file from

<http://repo1.maven.org/maven2/org/glassfish/jersey/bundles/jaxrs-ri/2.29.1/jaxrs-ri-2.29.1.zip>

Which includes the jar file which needs to be imported into the preferred IDE.

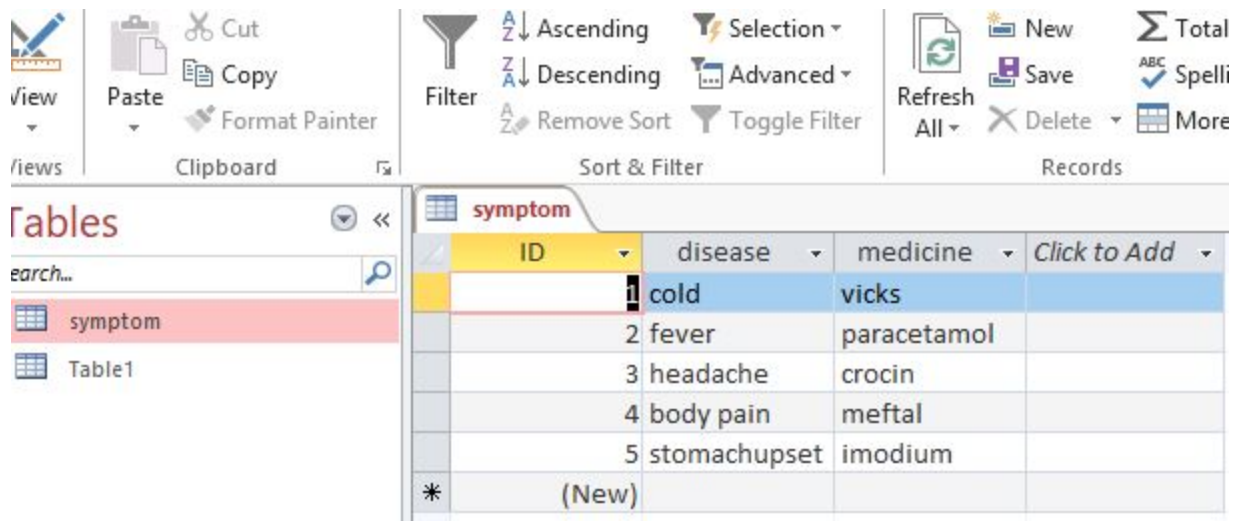
3. COMMANDS / CONFIGURATION / PROGRAMS/ OUTPUT

3.1 RESTFUL PHP SERVICE AND JAVA CLIENT

The project involves a front end, coded using Java SWING. Backend uses the facilities of PHP. The initial page, describes the patient details, which includes the name, contact information, age, blood group and the status of the patient appointment. After submitting the details, the information is stored in the database and then the next window for selecting the symptoms is shown where the patient can select the symptoms from the window and using the data the prescription is retrieved from the database. After the prescription the patient can view the details entered using the patient details button which retrieves the data from database.

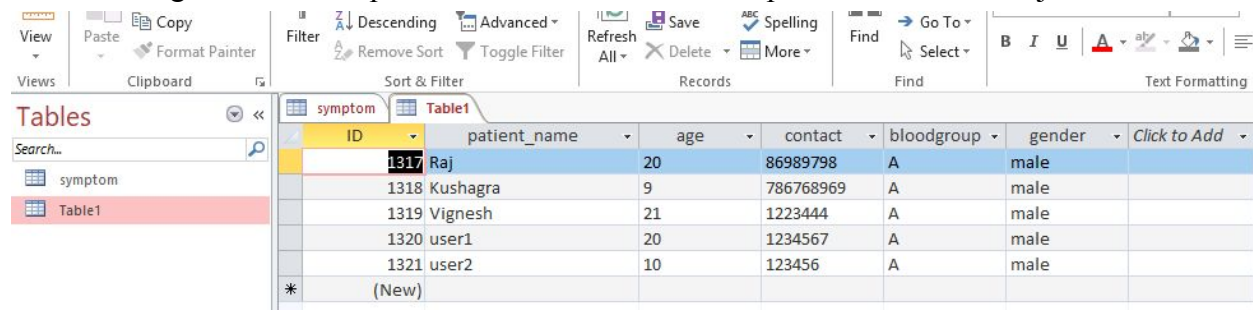
1. Creating Database using Microsoft Access:

The first step in creating the web service is to create a Database and it is done by using Microsoft Access or this service. To create a Microsoft Access database go to the Access application given with the Microsoft Office product and start the program. After starting the program select on creating a new table and name the table with proper column definition. There are two tables created or this which are named symptoms and Table1 which contain the symptom list and patient details respectively.



ID	disease	medicine	Click to Add
1	cold	vicks	
2	fever	paracetamol	
3	headache	crocin	
4	body pain	meftal	
5	stomachupset	imodium	
(New)			

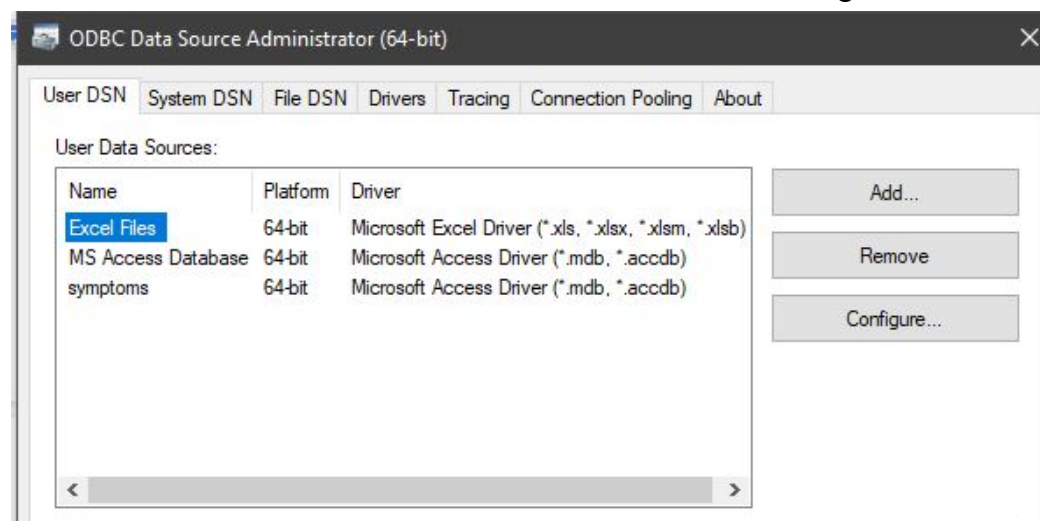
The below figure shows the patient details stored when the patient enter it in the java client.



ID	patient_name	age	contact	bloodgroup	gender	Click to Add
1317	Raj	20	86989798	A	male	
1318	Kushagra	9	786768969	A	male	
1319	Vignesh	21	1223444	A	male	
1320	user1	20	1234567	A	male	
1321	user2	10	123456	A	male	
(New)						

2. Creating a ODBC Connection:

Type ODBC in the start menu and the following window will open. Next a DSN is created, a DSN is a Data Source Network which is a connection to the string of the database location.



3. Creating the ODBC Connection in PHP:

After creating the ODBC database file the php.ini configuration file is changed so that php can use the ODBC connection to retrieve the database from the source.

```
;extension=php_pdo_oci.dll  
extension=php_pdo_odbc.dll  
;extension=php_pdo_pgsql.dll  
extension=php_pdo_sqlite.dll
```

4. PHP File to insert the patient details into the database:

A php file is created to insert the patient details entered in the client side. It takes the values from the URL and inserts it into the respective columns.

```
<?php  
$conn=odbc_connect('symptoms1','','');  
$name_val=$_GET['name'];  
$age_val=$_GET['age'];  
$contact_val=$_GET['contact'];  
$blood_val=$_GET['blood'];  
$gender_val=$_GET['gender'];  
$sql = "INSERT INTO Table1(patient_name,age,contact,bloodgroup,gender) VALUES('$name_val'," .  
        . "'$age_val','$contact_val','$blood_val','$gender_val')";  
$rs=odbc_exec($conn,$sql);
```

5. Creating the Java Client for entering the Patient Details:

The java client can be created using the Java Swing and AWT libraries and the entered values are stored in a private variables which is then sent as URL to the web service.

The values from the text fields can be retrieved and sent as URL as shown below

```
String name_val=jTextField1.getText();
String age_val=jTextField2.getText();
String contact_val=jTextField3.getText();
String blood_val=jTextField4.getText();
String gender_val=jTextField5.getText();
Client client = Client.create();
String php_url= "http://localhost:8080/networklab4/get_patient_details.php?name="+
+name_val+"&age="+age_val+"&contact="+contact_val+"&blood="+blood_val+"&gender="+gender_val;
```

Jersey Java API is used for the java client to retrieve the data from the php web service in a plain text format, this can be replaced with JSON format also but as the data is in single line format there is no need for complications.

```
Client client = Client.create();
WebResource webResource = client.resource(php_url);
ClientResponse response = webResource.accept("").get(ClientResponse.class);
String output2 = response.getEntity(String.class);
jTextArea1.append("The REST service URL is:"+php_url+"\n");
jTextArea1.append("-----Prescription-----\n");
jTextArea1.append(output2);
jTextArea1.setText(php_url);
new ui_prescription(name_val).setVisible(true);
```

After the java patient details client is done submitting the ui_prescription window is opened for the patient to enter the symptoms. The highlighted area is the REST URL which communicates with the web service.

Enter Patient Details

Name: Ramesh

Age: 20

Contact No.: 12345677

Blood Group: O-

Gender: Female

Status: =Ramesh&age=20&contact=12345677&blood=O-&gender=Female

Submit

After clicking the submit button the next window is opened with the web service being called and the data entered can be verified with the database.

ID	patient_name	age	contact	bloodgroup	gender	Click to Add
1317	Raj	20	86989798	A	male	
1318	Kushagra	9	786768969	A	male	
1319	Vignesh	21	1223444	A	male	
1320	user1	20	1234567	A	male	
1321	user2	10	123456	A	male	
1322	Ramesh	20	12345677	O-	Female	
*	(New)					

6. PHP File for the Java Prescription Client to communicate:

PHP runs on wamp server which is used to host sites either on localhost or via ip. Creating a new php file for the java client to communicate and the odbc connection is made as shown in the below figure.

```

function get_from_db($symp){
    $conn=odbc_connect('symptoms1','','');
    $sql = "SELECT * FROM symptom where disease='".$symp."'";
    $rs=odbc_exec($conn,$sql);
    while (odbc_fetch_row($rs)){
        $med_name=odbc_result($rs,"medicine");
        echo $med_name;
        echo "\n";
    }
}

```

7. Making the PHP file to receive the variables in URL:

Using the following code shown below the php file will take the variables from the url when called.

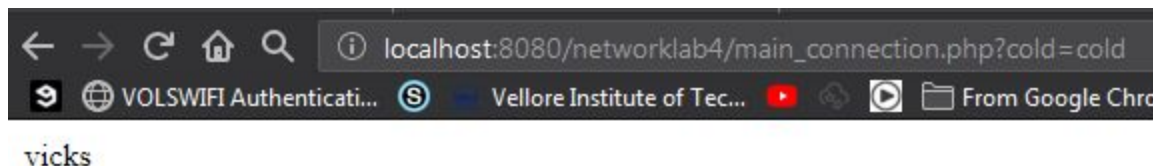
```

if (isset($_GET['cold'])) {
    if($_GET['cold']=="cold"){
        $symp="cold";
        get_from_db($symp);
    }
}

if (isset($_GET['fever'])) {
    if($_GET['fever']=="fever"){
        $symp="fever";
        get_from_db($symp);
    }
}

```

To check whether the URL is working, a sample url with the values is provided and the result is observed. As shown the URL is working and is retrieving the data from the database.



8. Java Prescription Client for entering Patient Symptoms:

The Prescription client takes the values from the URL and returns it in the plain text format in the text area as shown below.

Welcome Ramesh

Prescription Details

Cold ☒ Yes

Fever ☐ Yes

Headache ☒ Yes

Body Pain ☒ Yes

Stomach Upset ☐ Yes

Status

You selected cold
 You selected headache
 You selected bodypain
 The REST service URL is:http://localhost:8080/networklab4/main_co
 -----Prescription-----
 vicks
 crocin
 meftal

The patient details button in the above figure takes the values of the patient entered in the first window and returns it from the database.

Welcome Ramesh

Prescription Details

Cold ☒ Yes

Fever ☐ Yes

Headache ☒ Yes

Body Pain ☒ Yes

Stomach Upset ☐ Yes

Status

The REST service URL is:http://localhost:8080/networklab4/datafrom
 -----PATIENT DATA-----
 name: Ramesh
 age: 20
 contactno.:12345677
 bloodgroup:O-
 gender:Female

It communicates with another php service file where the database returns the row values of the key passed through the URL.


```

<?php
function data_from_db($name){
    //echo $name;
    $conn=odbc_connect('symptoms1','','');
    $sql = "SELECT * FROM Table1 where patient_name='".$name."'";
    $rs=odbc_exec($conn,$sql);
    while (odbc_fetch_row($rs)){
        $db_data=odbc_result($rs,"patient_name");
        $db_age=odbc_result($rs,"age");
        $db_contact=odbc_result($rs,"contact");
        $db_bloodgroup=odbc_result($rs,"bloodgroup");
        $db_gender=odbc_result($rs,"gender");
        echo "name: ".$db_data."\n";
        echo "age: ".$db_age."\n";
        echo "contactno.: ".$db_contact."\n";
        echo "bloodgroup: ".$db_bloodgroup."\n";
        echo "gender: ".$db_gender."\n";
        echo "\n";
    }
}

```

3.2 Python Simple REST API Creation:

1. Simple Hello world API

```

from flask import Flask, url_for
app = Flask(__name__)

@app.route('/')
def api_root():
    return 'Hello World...!!'

if __name__ == '__main__':
    app.run()

```