

DATA CONNECTIVITY

IT APPLICATION

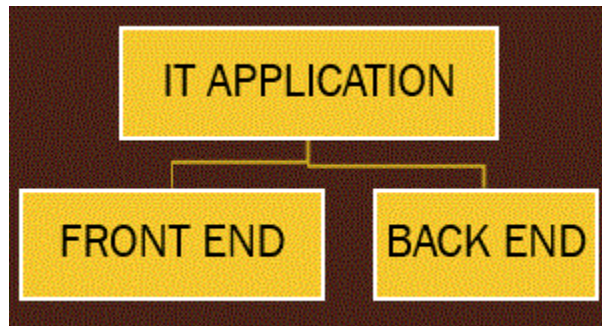
An IT application is a program or group of programs which provides ease and saves time and efforts in getting various jobs done.

Examples:

- Reservation of seat on flight/train
- Reservation of movie tickets
- Registering for new email account
- Buying things online
- Electronic billing system at shops

PARTS OF AN IT APPLICATION

A complete IT application has two parts: Front end and Back end.



EXAMPLE

Take an example of a restaurant.

- Front end of a restaurant:
 - ❖ The decorations, menus, wait-staff
 - ❖ As a customer, we see the front-end but we can't see the kitchen.

- ❑ Back end of a restaurant:
 - ❖ Kitchen, stock room etc.
 - ❖ As a customer, the kitchen and stockroom are out of view, but preparing food, keeping record of stock etc.

Take an example of a programming application: Billing system at Super Bazar.

- ❑ Front end:
 - ❖ The form on the computer screen where shop keeper enters information about the things purchased by us.
 - ❖ The form is made user friendly and attractive.
- ❑ Back end:
 - ❖ Database where the information about the customer and the things purchased, total amount paid etc. are stored permanently for future use.
 - ❖ As a customer, the database is out of view.

Thus, we can summarize front and back end in general view as:

- ❑ The front-end is the part a user sees and interacts with.
- ❑ The back-end is where all the behind-the-scenes processing happens which then returns the results to the front end.

FRONT END

The Front-End(GUI) helps the user to design forms to accept data and provide instructions for retrieving/storing information from/to the backend.

Some of the common tools to develop front end are: Java Net Beans, Java Script, ASP, JSP, Visual Basic, HTML.

BACKEND (DATABASE)

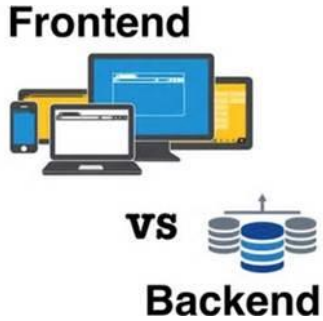
The backend helps to save data permanently in secondary storage devices and keeps data ready for future reference.

Example: MySQL, Dbase, SEQUEL Server etc.

NEED OF FRONT END AND BACKEND

- ❑ The user using the application is generally a lay man not a computer expert. He needs an interface which is simple and user friendly. He cannot directly work on back end. Thus, front end is needed to provide user with an interface to access or modify the information available in the database.
- ❑ The data entered in a front-end application gets stored in temporary memory (RAM) and cannot be retained for future use. Thus, back end is needed to store the data permanently.

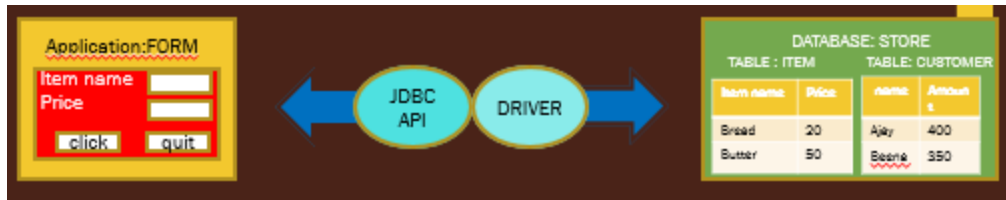
FRONTEND + BACKEND

<ul style="list-style-type: none">❑ A complete IT application requires creating an interface (the form), which a user may use for entering/accessing information and to store the information permanently for future use.❑ Thus both these concepts need to be combined/connected together to develop a complete application.❑ This is achieved with the help of database connectivity.	 <p>The diagram illustrates the relationship between the Frontend and Backend. On the left, the word "Frontend" is written above icons of a smartphone, a laptop, and a desktop monitor. In the center, the word "vs" is written. On the right, the word "Backend" is written above a database icon (a cylinder with a plus sign).</p>
---	---

DATABASE CONNECTIVITY

Database connectivity sets up communication between the front end and the back end. The two components essential to establish this connectivity are:

- ❑ **The JDBC API** - software used to provide RDBMS access and execute SQL statements within java code. API stands for Application Programming Interface.
- ❑ **The JDBC Driver for MySQL** - software component enabling a java application to interact with a MySQL database.



ADDING LIBRARIES

To establish data connectivity, we need:

- ❑ To add the **MySQL JDBC Driver Library**
- ❑ To import some basic class libraries:
 - ❖ **java.sql.DriverManager:** defines objects which can connect Java applications to a JDBC driver.
 - ❖ **com.mysql.jdbc.Connection:** represents a session/connection with a specific database.
 - ❖ **com.mysql.jdbc.Statement:** used to send and execute SQL statements to a database.

EXAMPLE

Consider the program to add new book to the library. Code for adding record is as follows:

<pre>String Accession_Number=jTextField1.getText(); String Title=jTextField2.getText(); String Author=jTextField3.getText(); String Price=jTextField4.getText(); try { Class.forName("java.sql.Driver"); Connection con =DriverManager.getConnection ("jdbc:mysql://localhost:3306/Slibrary", "root", "abcd"); Statement stmt=(Statement) con.createStatement(); String query="INSERT INTO library VALUES ("'+Accession_Number +'','"+Title+"','"+Author+"','"+Price+"')"; stmt.executeUpdate(query); } catch(Exception e) { JOptionPane.showMessageDialog (this, e.getMessage()) }</pre>	<div style="border: 1px solid #ccc; padding: 10px; width: fit-content; margin: auto;"> <p>SCHOOL LIBRARY</p> <p>Accession number <input type="text"/></p> <p>Title <input type="text"/></p> <p>Author <input type="text"/></p> <p>Price <input type="text"/></p> <p><input type="button" value="ADD RECORD"/> <input type="button" value="EXIT"/></p> </div>
---	--

EXAMPLE EXPLAINED

Command	Explanation
<pre>Class.forName("java.sql.Driver");</pre>	<p>In this step, Class.forName() method is called to load the driver class. The Driver class name is sent as an argument. Once loaded, the Driver class creates an object of itself. A client can then connect to the Database Server through JDBC Driver.</p>
<pre>Connection con = DriverManager.getConnection("jdbc:mysql: //localhost:3306/Slibrary","root","lib");</pre>	<p>The getConnection() method of the DriverManager class is used to establish a connection to the Slibrary database. The method uses a username (root), password(lib), and a jdbc url to establish a connection to the database and returns a connection object named con. Here 3306 is the Default Port number on which MySQL runs</p>
<pre>Statement stmt=(Statement) con.createStatement();</pre>	<p>The createStatement() method instantiate a Statement object called stmt from the connection object con. A statement object is used to send and execute SQL statements to a database.</p>
<pre>String query="INSERT INTO library VALUES ("'+Accession_Number+ "', '"+Title+"', '"+Author+"', '"+Price+'");";</pre>	<p>A variable query of type String is created and initialized with the SQL statement to be executed (in this case the INSERT INTO statement).</p>
<pre>stmt.executeUpdate(query);</pre>	<p>The executeUpdate() method executes the SQL statement stored in the variable query. These results in adding the values stored in the variables (Accession_Number,</p>

	Title, Author and Price) in the table Library of the Slibrary database.
catch(Exception e) { JOptionPane.showMessageDialog(this, e.getMessage()) } }	In case of an exception, retrieve the error message string using the getMessage() method and display it in a dialog box using the showMessageDialog() method.

[Click to view video on Java and MySQL connectivity](#)