

Department of Information Technology, S.G.S.I.T.S.

Web Engineering (IT 38504)

Lab Assignment 7

Mini Project – Artelic – Image Hosting Website

Source Code - <https://github.com/yash1823/artelic>

Submitted to – Dr. Lalit Purohit Sir

Submitted By – Yash Saraswat 0801IT191101,

6th Semester B.Tech. IT

Developed with : Tanishq Rathore 0801IT191092

1. INTRODUCTION

Our website “Artelic” has been developed to provide a hassle-free and easy-to-go art-hosting platform for artists all around the globe. This software is supported to eliminate and in most case reduce the hardships faced by artists to properly display their art for the world to see and earn their living from it.

The application has reduced as much as possible to avoid errors while taking data inputs from users. It also provides error messages while users are entering or they have entered invalid data. One major plus point of Artelic is that no formal knowledge is needed for the user to use this system. Thus, all our website provides is user-friendly.

Artelic, as described above, can lead to an error-free, secure, reliable art-hosting system. It can assist the user to concentrate solely on their art’s exposure rather to concentrate on the record keeping. Thus it will help users in better utilization of their resources.

1.1 SCOPE AND OBJECTIVE

The main goal was that the project’s website would act as art-hosting platform for artists around the world.

For this reason the website was designed taking into consideration the following:

- To be simple and light
- To be extremely easy to use

2. MODULES

A Web module is the smallest deployable and usable unit of Web resources.

- **Registration Module**
 1. Visitors can register an account.
- **Login Module**
 1. Registered users can login and log out of the system.
- **Main-Page Module**
 1. Registered users can visit and browse personal homepage.
 2. Registered users can also access entry-point to other resources (gallery, upload, about).
- **Gallery Module**
 1. Registered users can visit and browse gallery.
- **Upload Art Module**
 1. Registered users can select their art to be uploaded.
 2. Registered users can upload their selected art
- **About Module**
 1. Registered users can browse the about(about developers) section.

3. TECHNOLOGIES USED

Front-end Design: HTML, CSS, Bootstrap

Hypertext Markup Language (HTML) is the standard markup language for creating web pages and web applications. With Cascading Style Sheets (CSS) and JavaScript, it forms a triad of cornerstone technologies for the World Wide Web. Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language like HTML. CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript. CSS is designed to enable the separation of presentation and content, including layout, colors, and fonts. This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple web pages to share formatting by

specifying the relevant CSS in a separate .css file, and reduce complexity and repetition in the structural content.

Bootstrap is a free and open-source front-end library for designing websites and web applications. It contains HTML- and CSS-based design templates for typography, forms, buttons, navigation and other interface components, as well as optional JavaScript extensions. Unlike many web frameworks, it concerns itself with front-end development only.

Client side validation: JavaScript

JavaScript often abbreviated as JS, is a high-level, interpreted programming language. It is a language which is also characterized as dynamic, weakly typed, prototype-based and multi-paradigm. Alongside HTML and CSS, JavaScript is one of the three core technologies of the World Wide Web. JavaScript enables interactive web pages and thus is an essential part of web applications. The vast majority of websites use it, and all major web browsers have a dedicated JavaScript engine to execute it.

Server side validation and Business Logic: Java

Then we have Java, the 500-pound gorilla of internet backends. Java started as an all-purpose low-level language in 1995 and was a huge success due to its focus on object-oriented programming. Since then, it has been nothing but a victory lap for it, as almost every single device in the world has a bit of Java in it.

As for server-side scripting, it's really powerful, it's flexible, it's extremely fast, and like any other low-level language, a good developer can deeply optimize it for peak performance.

Using the plethora of tools it offers, namely JSPs, Servlets, JSTL we have solidified the backend of this web project

Database: MySQL

MySQL is an open source relational database management system. For proprietary use, several paid editions are available, and offer additional functionality. In this project MySQL has been used to store, update, retrieve and delete related to user's data and other additional data about projects.

Web Server: Apache

The Apache HTTP Server, colloquially called Apache, is a free and open-source cross-platform web server, released under the terms of Apache License 2.0. Apache is developed and maintained by an open community of developers under the auspices of the Apache Software Foundation. The Apache HTTP Server is cross-platform; as of 1 June 2017 92% of Apache HTTP Server copies run on Linux distributions. Version 2.0 improved support for non-Unix operating systems such as Windows and OS/2. Old versions of Apache were ported to run on OpenVMS and NetWare.

Functional Requirements:

1. Authentication of user must be done whenever he/she logs into the system.
2. It should be in a simple language and unambiguous in nature to make itself user friendly and easy to navigate
3. The system should be able to comply with the legal constraints and follow the regulation of its governing body
4. It should be able to maintain the art and user records
5. It should be able to inform of the transactions' correction and cancellation
6. External interface should be present so that an outside user is able to use the system

Other Non-Functional Requirements:**Performance Requirement:**

The proposed system that we are going to develop will be used as the chief system within the different users.

Therefore, it is expected that the database would perform functionally all the requirements that are specified for the users.

- The performance of the system should be fast and accurate.
- Artelic shall handle expected and unexpected errors in way that prevent loss in information and long downtime period. Thus it should have inbuilt error testing to identify search or data check/fetch.
- The system should be able to handle certain amount of data. Thus it should accommodate large number of data entry of users without any fault.

Satisfy Requirement:

The database may get crashed at any certain point of time due to virus or operating system failure, Therefore, it is required to take the database backup.

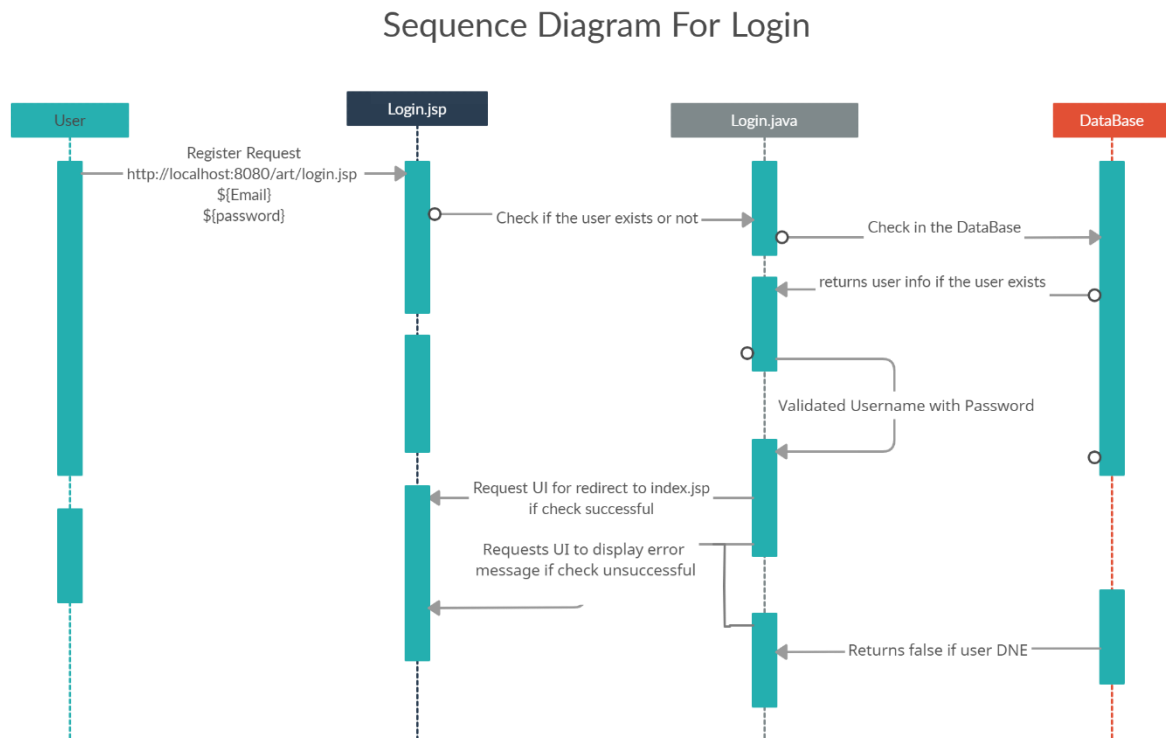
Security Requirement:

- System will use secure database
- Normal users are only limited to allowed functionalities, they cannot edit or modify anything except upload their art.
- Proper user authentication should be provided.
- No one should be able to hack user's detail or any other

information.

- Only admin has rights to update the database.

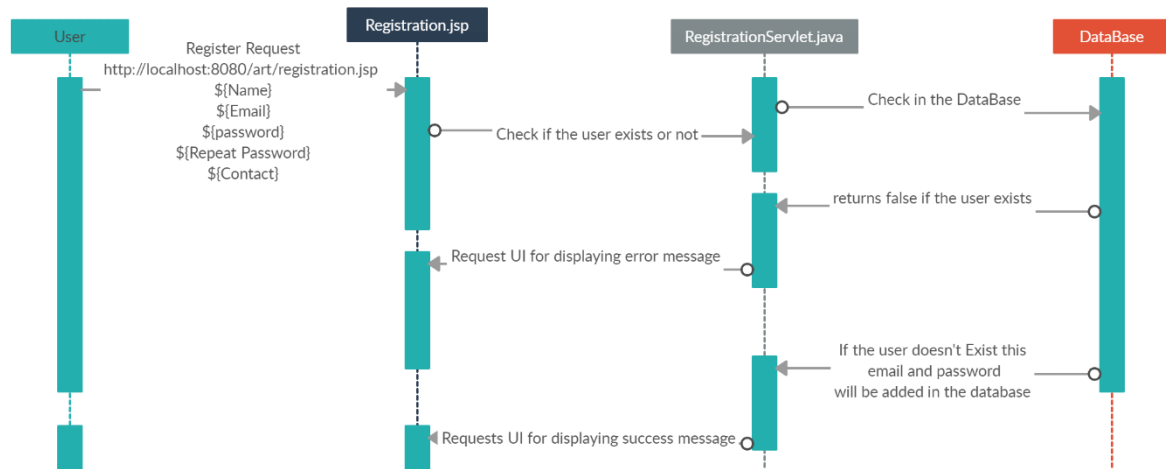
4.Design Diagrams



The basic flow of user login is as follows:

1. The user accesses the login page **login.jsp** and inputs his/her email and password.
2. The **login.jsp** page sends the login request together with the user's input information to the **Login.java Servlet**
3. **LoginServlet** queries the **database** to check for credentials and returns user info objects
4. **LoginServlet** verifies the request object data and the query returned data objects credentials
5. If the password is wrong, login fails and then returns to the login page **login.jsp**, reporting an error. Otherwise, login succeeds and then jumps to the user's personal homepage **index.jsp**

Sequence Diagram For Registering



The basic flow of user registration is as follows:

1. The user accesses **registration.jsp** page with registration credentials namely:

uname,

uemail,

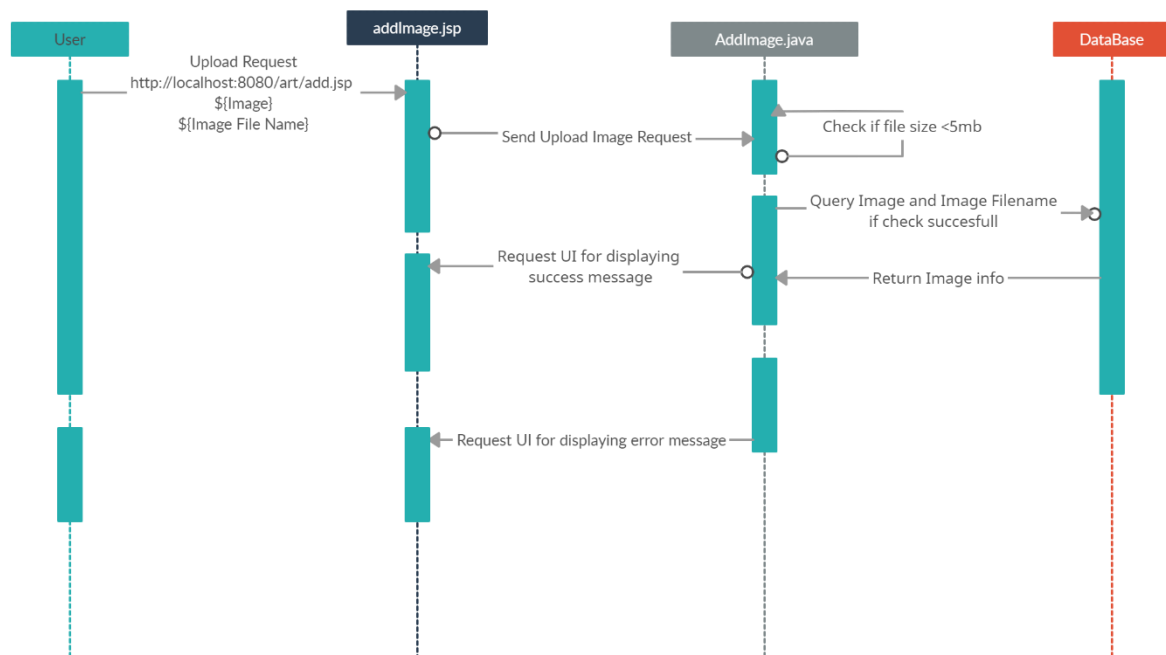
upwd,

reupwd,

contact

2. The **registration.jsp** page sends the registration request together with the user's input information to the **RegistrationServlet.java Servlet**
3. **RegistrationServlet** queries the **database** to check for if user exists or not and returns status
4. If the user exists then status is set as “failed” and **registration.jsp** is requested to display the error message.
5. If the user DNE then **RegistrationServlet** stores information in the Database and jumps to **registration.jsp** page to display success message.

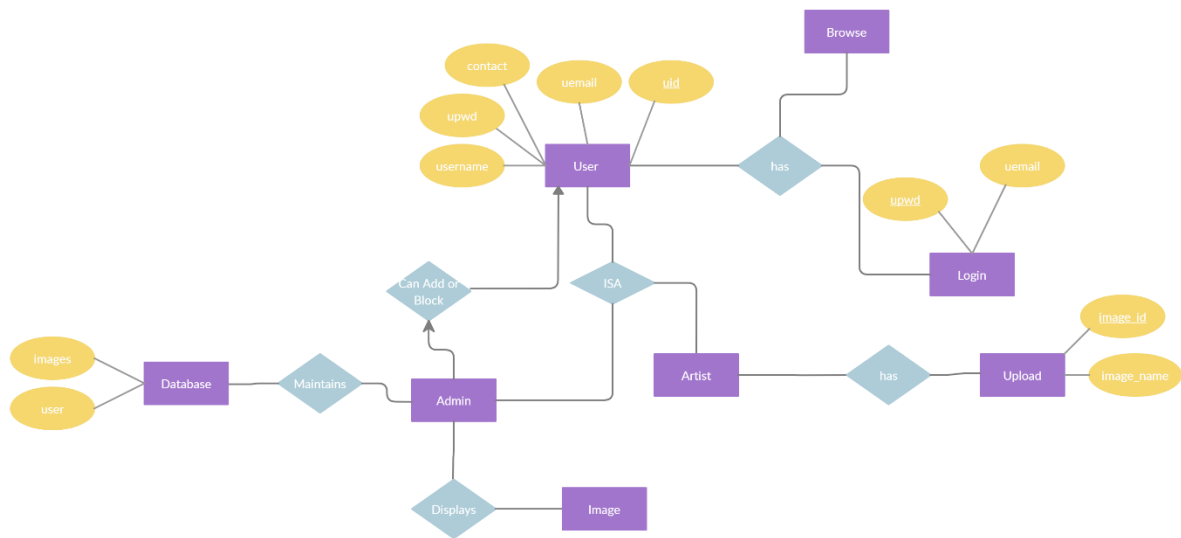
Sequence Diagram For Upload



The basic flow of upload is as follows:

1. The user accesses the **addImage.jsp** and selects image to be uploaded from their side.
2. The **addImage.jsp** page sends the upload request together with the user's image information to the **AddImage.java Servlet**, this servlet firstly checks if the file size is less than 5mb.
3. **AddImageServlet** queries the **database** if check is completed and stores the image info and image.
4. **AddImageServlet** jumps to **addImage.jsp** to display success message.
5. If check fails, then servlet jumps to **addImage.jsp** to display error message.

5. ER diagram with explanation



- **User**

- In this system , user has login functionality
- Every user has individual and unique login credentials to do through the registration portal.
- Users can navigate the website through and through after login.(**Browse**)
- Users can be **Admin or Artist**

- **Admin**

- Admin can modify **database**
- Admin can add users and provide them user id and password which is required by the users to access this website
- Admin can add and remove **image** in gallery

- **Artist**

- Artists can **upload** their art on the website's database

Database tables:

| Name | Engine | Version | Row Format |
|-------|--------|---------|------------|
| image | InnoDB | 10 | Dynamic |
| users | InnoDB | 10 | Dynamic |

Image Table:

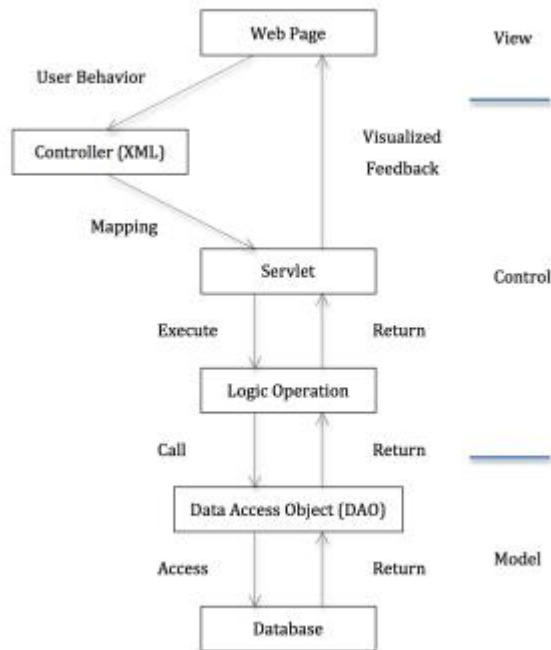
| Field | Type | Null | Key | Default | Extra |
|---------------|-------------|------|-----|---------|----------------|
| imageId | int | NO | PRI | NULL | auto_increment |
| imageFileName | varchar(60) | YES | | NULL | |

Users Table:

| Field | Type | Null | Key | Default | Extra |
|---------|-------------|------|-----|---------|----------------|
| id | int | NO | PRI | NULL | auto_increment |
| uname | varchar(50) | NO | | NULL | |
| uemail | varchar(50) | NO | UNI | NULL | |
| umobile | varchar(20) | NO | | NULL | |
| upwd | varchar(20) | NO | | NULL | |

6. MVC Architecture and components:

We use MVC framework as the overall structure in the design of this system. And the system is divided into 3 layers: view, control and model layer, which not only makes functions of different levels to be independent of each other, but also achieve the loose coupling of the system . This structure greatly facilitates the system development and testing, and lays a solid foundation for the future improvement and expansion of the system . The overall structure of the system is as shown in



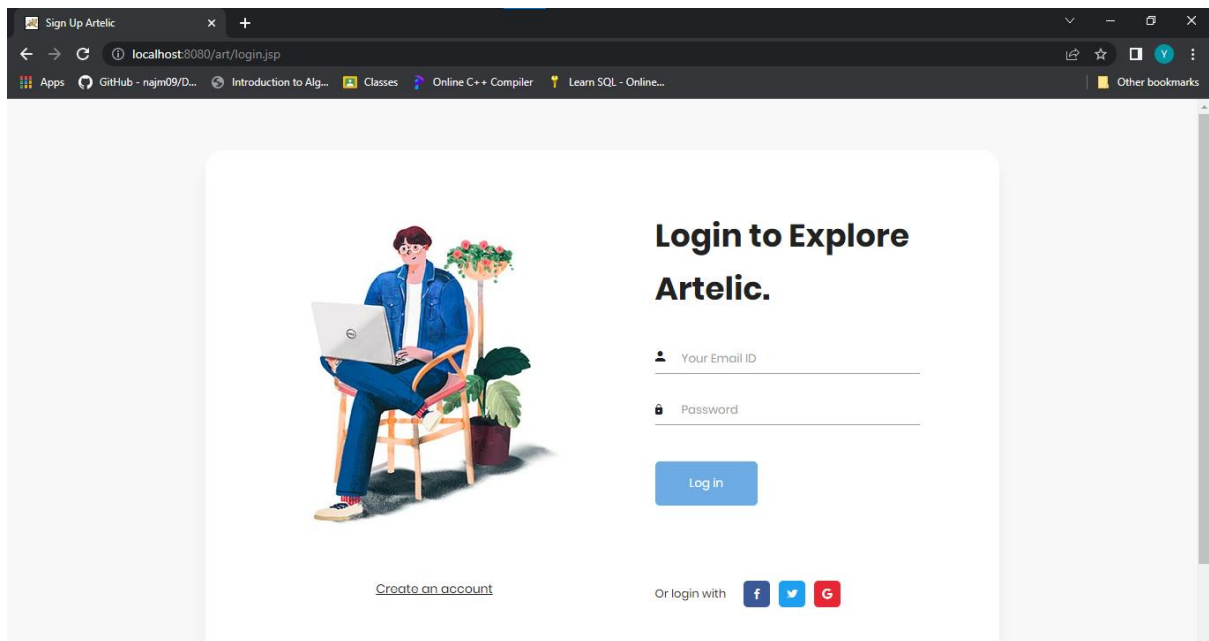
View layer: view layer is mainly used to provide users with visualized operations, to facilitate interactions between users and the system. A good view layer should contain only the user visualization interface and the data of user operation, it should avoid containing business logic operation of the system and operation on the database.

Control layer: as the core of the MVC structure, control layer is mainly used to receive data from the view layer, to perform business logic operation and to call the functions of model layer. When the view layer sends a request to the control layer, the web application service normally filters and match the requested URL with the controller (xml configuration file), then sends it to the specified Servlet.

Model layer: model layer is mainly used for the data operation, such as CRUD (create, retrieve, update and delete) operation. In this system, we also import DAO (data access object) Entity, encapsulation of the database, and data objects oriented operation, to make the model layer to be loose coupling and manageable

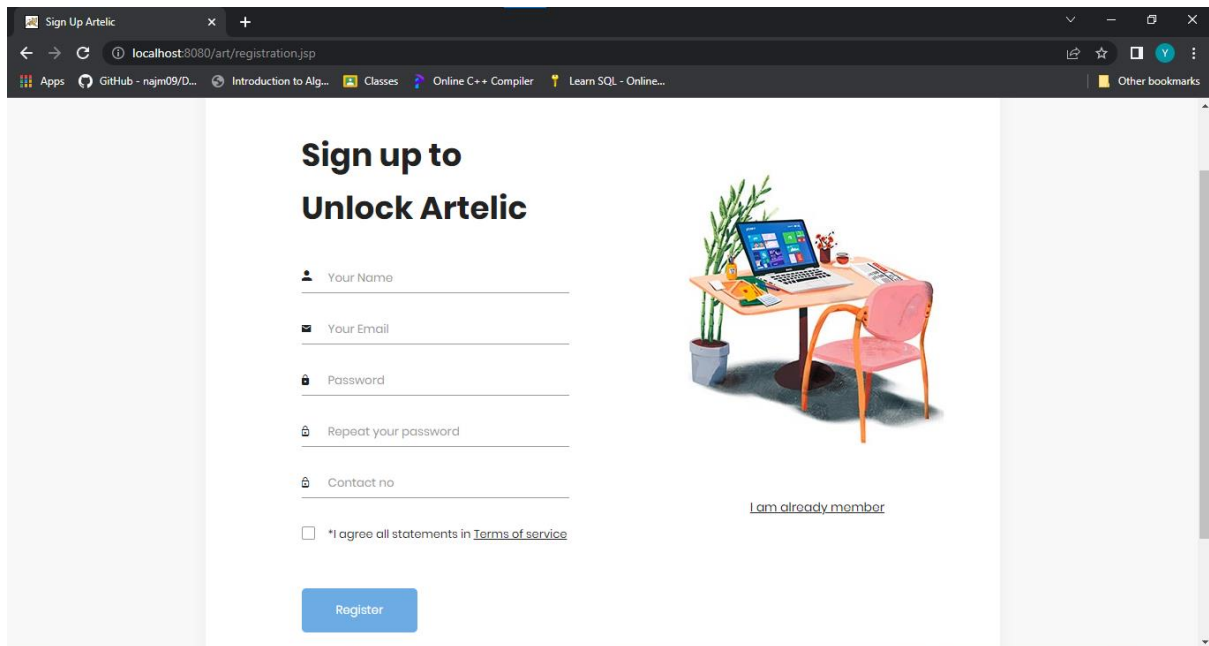
JSPs:

Login.jsp :



1. This jsp page is the one which acts as a proxy home page for unregistered users.
2. It's main duty is to provide "login" request to the Login.java servlet with the user's input, so as to provide the login functionality
3. If status is set as failed then error message is displayed here.
4. Redirection to registration.jsp is also provided in it for registering the user.

Registration.jsp :



Sign up to
Unlock Artelic

Your Name

Your Email

Password

Repeat your password

Contact no

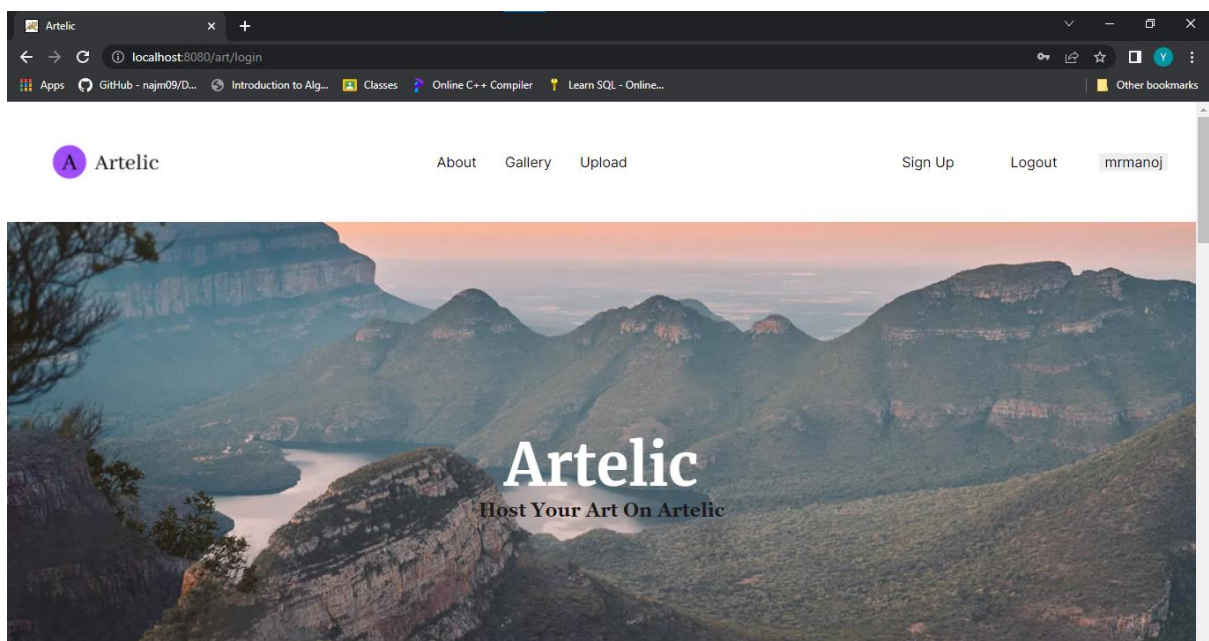
☐ *I agree all statements in [Terms of service](#)

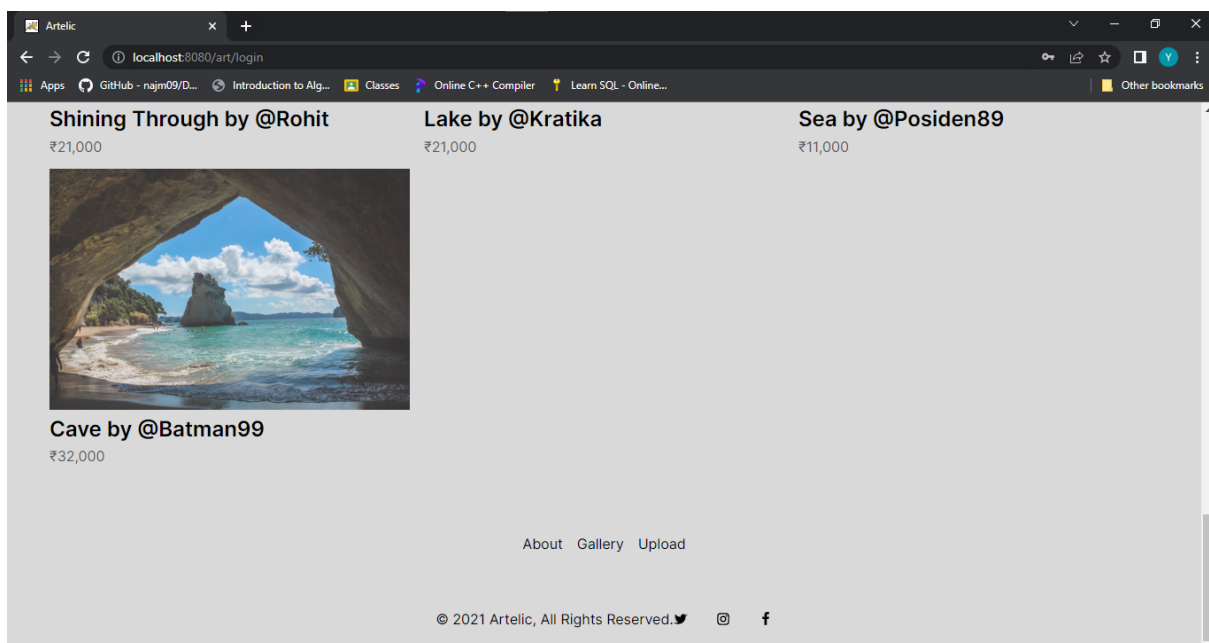
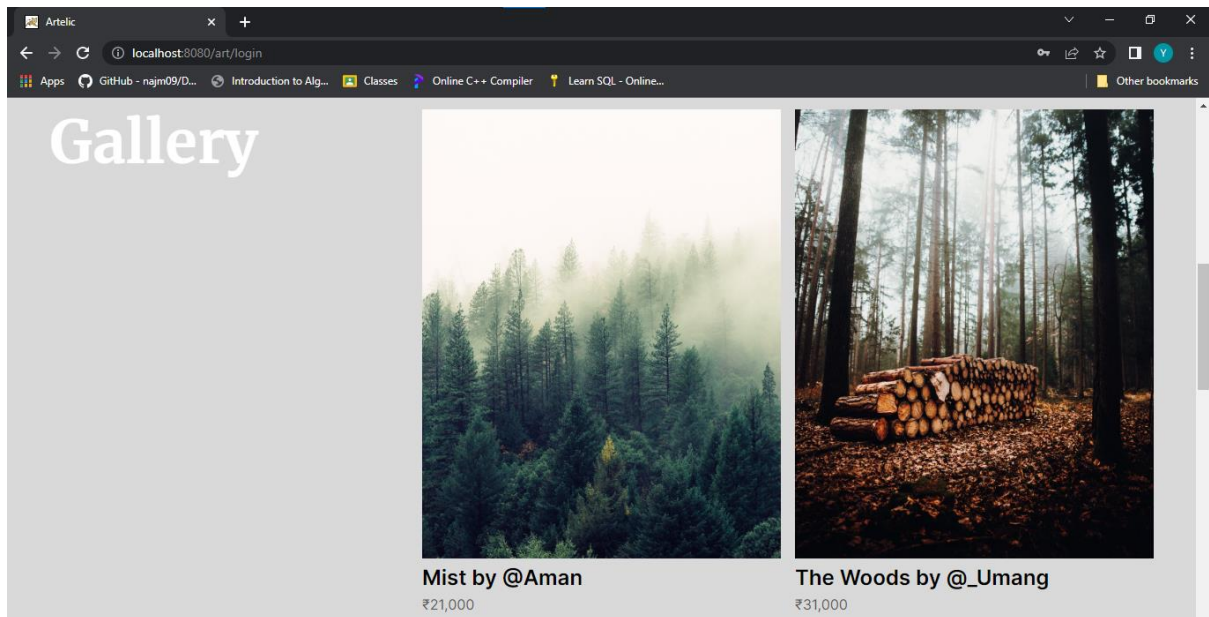
Register

[I am already member](#)

1. This jsp page is for providing user with a UI for registration.
2. It's main duty is to provide "register" request to the Registration.java servlet with the user's input, so as to provide the registration functionality
3. If status is set as failed then error message is displayed here.
4. If status is set as success then it success message is displayed here
5. Redirection to login.jsp is also provided in it for login functionality

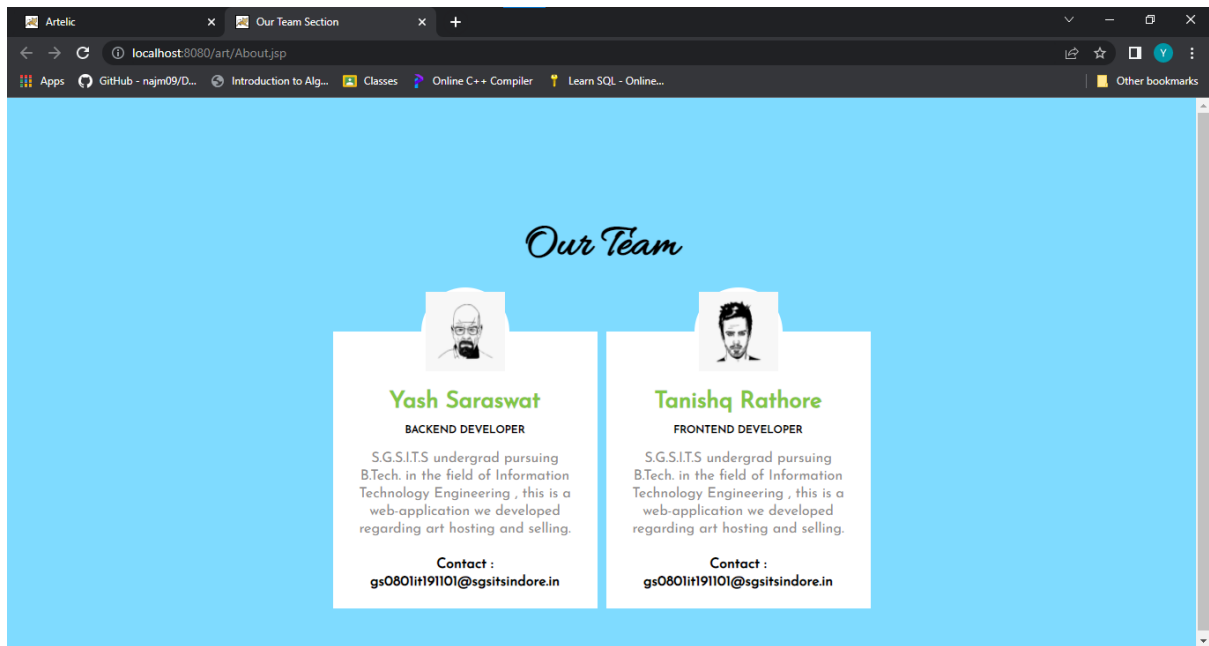
index.jsp:





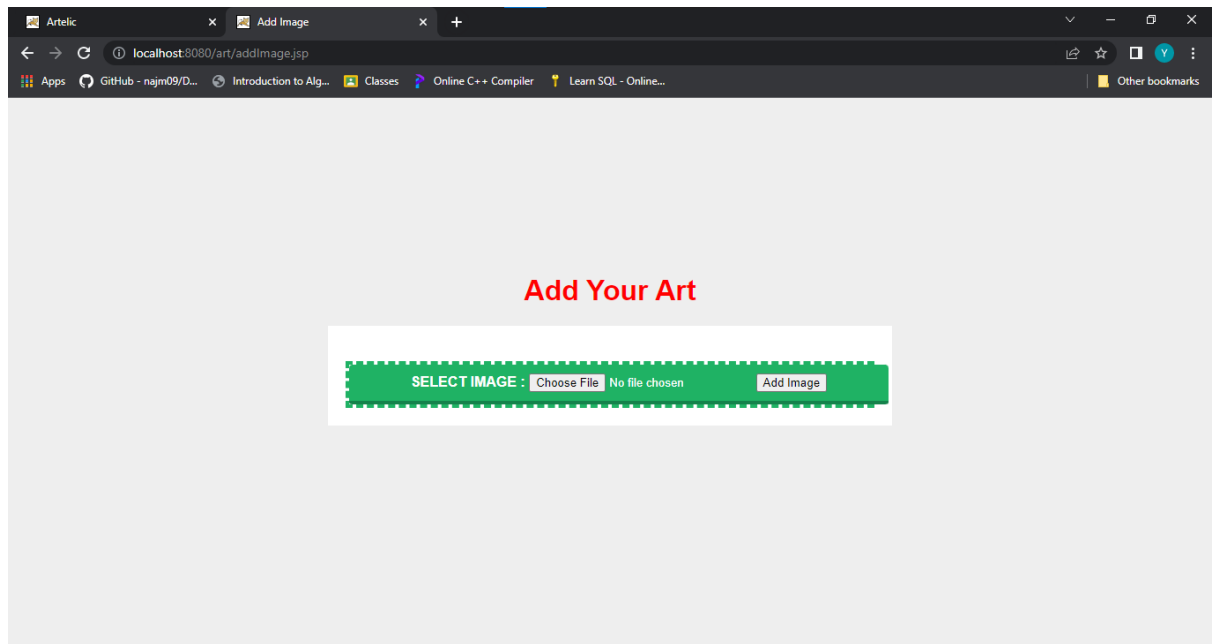
1. This jsp is the actual home page of the website
2. Here users can “browse” the website and avail the access to other pages and functionalities:
 - About
 - Upload
 - Sign-up
 - Logout

About.jsp



1. This page displays information about the developers of the page.

addImage.jsp:



1. Here we provide the user access to upload functionality where they can upload their art onto our database.
2. If the upload is successful then we display success message here else the error message.

Servlets:

We have 2 packages namely:

1. com.art.registration : This package includes three servlets

- **Login.java :**
 - This servlet handles “/login” request and creates session.
 - It then processes the user info to validate him/her for authentication by querying the database and retrieving the responses and ultimately working on to facilitate the login for the user.
 - This servlet redirects to jsp(s) and also sets status for the attribute in the jsp which displays success status.
 - This also has server side checks for login.
- **Logout.java :**
 - This servlet handles “/logout” requests
 - It invalidates the session and redirects to our proxy home page.
- **RegistrationServlet.java :**
 - This servlet handles “/register” requests
 - It then validates the user information for further registration processing.
 - After server side validations this servlet queries and gets responses from the data base for registering the user
 - This servlet sets status for the registration process which are then displayed on registration.jsp

2. img.artelic: In this package we have one servley

- **addImage.java:**
 - This servlet responds to “/AddImage” requests
 - Before querying , file size is checked in it
 - Then server side checks are also places here , majorly checking if the file name is null or not
 - Uploading of image in database takes place here.
 - Status is set which is then displayed on addImage.jsp

Servlets' Source Code:

RegistrationServlet.java:

```
package com.art.registration;

import jakarta.servlet.RequestDispatcher;
import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.sql.SQLIntegrityConstraintViolationException;

/**
 * Servlet implementation class RegistrationServlet
 */
@WebServlet("/register") public class RegistrationServlet extends
HttpServlet
{
    private
        static final long serialVersionUID = 1L;

    protected
        void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException
        {

            String uname = request.getParameter("name");
            String uemail = request.getParameter("email");
            String upwd = request.getParameter("pass");
            String Reupwd = request.getParameter("re_pass");
            String umobile = request.getParameter("contact");

            RequestDispatcher dispatcher = null;
            Connection con = null;

            if (uname == null || uname.equals(""))
            {
                request.setAttribute("status", "invalidName");
                dispatcher =
request.getRequestDispatcher("registration.jsp");
                dispatcher.forward(request, response);
            }
        }
    }
}
```



```

    }

    else if (uemail == null || uemail.equals(""))
    {
        request.setAttribute("status", "invalidEmail");
        dispatcher =
request.getRequestDispatcher("registration.jsp");
        dispatcher.forward(request, response);
    }
    else if (upwd == null || upwd.equals(""))
    {
        request.setAttribute("status", "invalidPassword");
        dispatcher =
request.getRequestDispatcher("registration.jsp");
        dispatcher.forward(request, response);
    }

    else if (umobile == null || umobile.equals(""))
    {
        request.setAttribute("status", "invalidMobile");
        dispatcher =
request.getRequestDispatcher("registration.jsp");
        dispatcher.forward(request, response);
    }
    else if (umobile.length() != 10)
    {
        request.setAttribute("status", "invalidMobileLength");
        dispatcher =
request.getRequestDispatcher("registration.jsp");
        dispatcher.forward(request, response);
    }
    else if (!upwd.equals(Reupwd))
    {
        request.setAttribute("status", "PasswordsNotMatching");
        dispatcher =
request.getRequestDispatcher("registration.jsp");
        dispatcher.forward(request, response);
    }

    else
    {

        try
        {
            Class.forName("com.mysql.cj.jdbc.Driver");
            con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/youtube?use
SSL=false", "root", "Yash@007");

```

```

        PreparedStatement pst = con.prepareStatement("insert
into youtube.users(uname,upwd,uemail,umobile) values(?,?,?,?)");
        pst.setString(1, uname);
        pst.setString(2, upwd);
        pst.setString(3, uemail);
        pst.setString(4, umobile);

        try
        {
            int rowCount = pst.executeUpdate(); // Might
throw duplicate value error
            dispatcher =
request.getRequestDispatcher("registration.jsp");

            if (rowCount > 0)
            {
                request.setAttribute("status", "success");
            }

            else
            {
                request.setAttribute("status", "failed");
                response.sendRedirect("registration.jsp");
            }
            dispatcher.forward(request, response);
        }
        catch (SQLIntegrityConstraintViolationException e)
        {
            System.out.println("Error Found!! And Caught
:)");

            request.setAttribute("status", "EmailExists");
            dispatcher =
request.getRequestDispatcher("registration.jsp");
            dispatcher.forward(request, response);
        }
    }
    catch (Exception e)
    {
        e.printStackTrace();
    }
    finally
    {
        try
        {
            con.close();
        }
        catch (SQLException e)
        {

```

```

    }
    }
    }
    }
    e.printStackTrace();
}

```

Login.java:

```
package com.art.registration;

import jakarta.servlet.RequestDispatcher;
import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import jakarta.servlet.http.HttpSession;

import java.io.IOException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

/**
 * Servlet implementation class Login
 */
@WebServlet("/login")
public class Login extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void doPost(HttpServletRequest request,
        HttpServletResponse response) throws ServletException, IOException {
        String uemail = request.getParameter("username");
        String upwd = request.getParameter("password");
        HttpSession session = request.getSession();//Session
Creation
        RequestDispatcher dispatcher = null;
        Connection con = null;

        if(uemail==null || uemail.equals("")) {
            request.setAttribute("status", "invalidEmail");
            dispatcher = request.getRequestDispatcher("login.jsp");
            dispatcher.forward(request, response);
        }
    }
}
```

```

    }
    if(upwd==null || upwd.equals("")) {
        request.setAttribute("status", "invalidUpwd");
        dispatcher = request.getRequestDispatcher("login.jsp");
        dispatcher.forward(request, response);
    }

    try {
        Class.forName("com.mysql.cj.jdbc.Driver");
        con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/youtube","r
oot","Yash@007");
        PreparedStatement pst = con.prepareStatement("select *
from users where uemail = ? and upwd = ?");

        pst.setString(1, uemail);//for Place Holders
        pst.setString(2, upwd);

        ResultSet rs = pst.executeQuery();

        if(rs.next()) {
            session.setAttribute("name", rs.getString("uname"));
            dispatcher =
request.getRequestDispatcher("index.jsp");
        }
        else {
            request.setAttribute("status", "failed");
            dispatcher =
request.getRequestDispatcher("login.jsp");
        }

        dispatcher.forward(request, response);

    }catch(Exception e) {
        e.printStackTrace();
    }finally {
        try {
            con.close();
        } catch (SQLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
}
}

```

```
}
```

Logout.java:

```
package com.art.registration;

import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import jakarta.servlet.http.HttpSession;

import java.io.IOException;

/**
 * Servlet implementation class Logout
 */
@WebServlet("/logout")
public class Logout extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void doGet(HttpServletRequest request,
        HttpServletResponse response) throws ServletException, IOException {
        HttpSession session = request.getSession();
        session.invalidate();
        response.sendRedirect("login.jsp");
    }
}
```

AddImage.java:

```
package img.artelic;

import jakarta.servlet.RequestDispatcher;
import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.MultipartConfig;
```

```

import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import jakarta.servlet.http.Part;

import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.sql.*;

/**
 * Servlet implementation class AddImage
 */
@MultipartConfig(maxFileSize = 1024 * 1024 * 5)//For Preventing DOS
@WebServlet("/AddImage")
public class AddImage extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void doPost(HttpServletRequest request,
        HttpServletResponse response) throws ServletException, IOException {

        System.out.println("In the dopost method of addimage
        servlet");
        Part file = request.getPart("image");
        String imageFileName = file.getSubmittedFileName();//to get
        selected image filename
        System.out.println("Selected Image File Name
        :"+imageFileName);
        RequestDispatcher dispatcher = null;

        if(imageFileName=="||imageFileName==null) {

            request.getSession().setAttribute("status","SelectFile")
;
            response.sendRedirect("addImage.jsp");
            System.out.println("Empty File Name Hai");
        }
        else {
            String uploadPath="C:/Users/HP/eclipse-
            workspace/art/images/"+imageFileName;
            System.out.println("Upload Path :"+uploadPath);

            //Uploading our selected image in the images folder
            try{
                FileOutputStream fos = new FileOutputStream(uploadPath);
                InputStream is = file.getInputStream();

```

```

        byte[] data = new byte[is.available()];
        is.read(data);
        fos.write(data);
        fos.close();
    }

    catch(Exception e) {
        e.printStackTrace();
    }

    //jdbc code
    Connection connection = null;
    try {
        Class.forName("com.mysql.cj.jdbc.Driver");
        connection=DriverManager.getConnection("jdbc:mysql://localhost:3306/youtube","root","Yash@007");
        PreparedStatement stmt;
        //Creating Query
        String query ="insert into image(imageFileName)
values(?)";
        stmt = connection.prepareStatement(query);
        stmt.setString(1, imageFileName);

        int row=stmt.executeUpdate();
        dispatcher =
request.getRequestDispatcher("addImage.jsp");

        if(row>0) {
            System.out.println("Image Added into DB
Successfully");
            request.setAttribute("status", "success");
        }
        else {
            System.out.println("Failed to upload image.");
            request.getSession().setAttribute("status","UploadFa
iled");
            response.sendRedirect("addImage.jsp");
        }
        dispatcher.forward(request, response);
    }
    catch(Exception e) {
        System.out.println(e);
    }
}
}
}
}

```