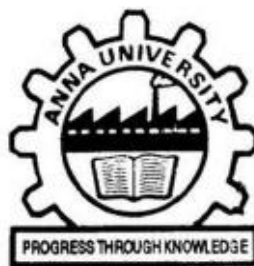


DMC 8211

**MASTER OF
COMPUTER APPLICATION**

**INTERNET PROGRAMMING
LABORATORY**



**CENTRE FOR DISTANCE AND ONLINE EDUCATION
ANNA UNIVERSITY
CHENNAI - 600 025**



Course Writer

Dr.G.Rajesh

Asst. Professor

Department of Information Technology
MIT Campus, Anna University

Chennai - 600 025.

Course Reviewer

Dr.G.S.Mahalakmi

Associate Professor

Department of Computer Science and
Engineering

Anna University, Chennai – 600 025.

INTERNET PROGRAMMING LABORATORY

SYLLABUS

- 1. Create an event registration application using javascript.** It should implement different widgets for registration form and registered records view using tabs. It should perform the form validation.
- 2. Create a javascript application in an Object Oriented way using Classes and Modules.**
It should also use browser storage for persistence.
- 3. Build a web application using Gradle.** The server side of the application should implement RESTful APIs using Servlet and do necessary logging. The client side of the application should be a single page application which consumes the RESTful APIs through AJAX.
- 4. Build a chat application using WebSocket.**
- 5. Create a Spring MVC application.** The application should handle form validation, file upload, session tracking.
- 6. Implement a RESTful Spring Boot application using Spring REST, Spring Security and Spring Cache.**
- 7. Design a complex system using JPA and Hibernate.** The system should have multiple entities and relationships between the entities. The database schema should be generated through Hibernate. Provide RESTful endpoints for CRUD operations for the defined entities. Also, support pagination and searching using JPA's JPQL and Criteria API.
- 8. Create a Spring RESTful Application with Spring Data JPA.** Support pagination and searching using Specifications.
- 9. Create a React application with different components and interactions between the components.**
- 10. Develop a full-stack application using React and Spring.** Make use of Spring REST, Spring Security, Spring Data JPA, Hibernate, Spring Boot, Gradle and React's higher order component.



INTERNET PROGRAMMING LABORATORY

PRACTICAL EXERCISES

Ex.No	Experiment	Pg.No
Lesson 1	EVENT REGISTRATION AND VALIDATION	5-10
Lesson 2	BEAN CLASS	11-30
Lesson 3	WEB APPLICATION USING GRADLE	31-35
Lesson 4	CHAT APPLICATION USING WEBSOCKETS	36-47
Lesson 5	SPRING MVC APPLICATION	48-53
Lesson 6	RESTful SPRING BOOT APPLICATION	54-62
Lesson 7	HIBERNATE	63-66
Lesson 8	SPRING RESTful APPLICATION WITH SPRING JPA	67-70
Lesson 9	REACT APPLICATION	71-77
Lesson 10	FULL-STACK APPLICATION USING REACT AND SPRING	78-134

PRACTICAL EXERCISES

1. EVENT REGISTRATION AND VALIDATION

AIM:

Create an event registration application using javascript. It should implement different widgets for registration form and registered records view using tabs. It should perform the form validation.

PROGRAM:

index.html

```
<!DOCTYPE html>

<html>

<head>

<title>Registration Form</title>

<link rel="stylesheet" type="text/css" href="stylesheet.css" />

<script type="text/javascript" src="script.js"></script>

</head>

<body>

<form method="post" target="#" onsubmit="return validate()">

<h1>Even Registration Form</h1>

Name<br>

<input type="text" name="name"><br>

Email<br>

<input type="text" name="email"><br>

Retype email<br>

<input type="text" name="retype_email"><br>Create a password<br>

<input type="password" name="password"><br>Retype password<br>

<input type="password" name="retype_password"><br>
```

```
<label>
```

```
<input type="checkbox" name="terms">By signing up this registration form
```

```
you agree to your
```

```
<a href="#service">Terms of Service</a> and <a href="#policy">Privacy
```

```
Policy</a>
```

```
</label>
```

```
<div class="button">
```

```
<input type="submit" class="button" value="Set Up Account">
```

```
</div>
```

```
</form>
```

```
</body>
```

```
</html>
```

```
script.js
```

```
function validate() {
```

```
    var result = "";
```

```
    result += validateName();
```

```
    result += validateEmail();
```

```
    result += validatePassword();
```

```
    result += validateTerms();
```

```
    if (result == "") return true;
```

```
    alert ("Validation Result:\n\n" + result);
```

```
    return false;
```

```
}
```

```
function validateName() {
```

```
    var name = document.getElementsByName("name")[0].value;
```

```
    if (name.length<= 3
```

```

        return "Name should be at least three characters.\n";

        return "";

    }

    function validatePassword() {
        var password = valueOf("password");
        var retype = valueOf("retype_password");
        if (password.length <= 6)
            return "Password should be at least 6 characters.\n";
        if (password != retype) return "Passwords do not match.\n";
        return "";
    }

    function validateEmail() {
        var email = valueOf("email");
        var retype = valueOf("retype_email");
        if (email.indexOf('@') == -1)
            return "Email should be a valid address.\n";
        if (email != retype)
            return "Email addresses do not match.\n";
        return "";
    }

    function validateTerms() {
        var terms = document.getElementsByName("terms")[0];
        if (!terms.checked)
            return "Please accept the Terms of Service and Privacy Policy";
        return "";
    }

    function valueOf(name) {
        return document.getElementsByName(name)[0].value;
    }

```

stylesheet.css

```

body {
    font-family: Verdana, sans-serif;
}

input {

```

```
border-radius: 5px;
padding: 5px;
}

div.button {
text-align: right;
}

a {

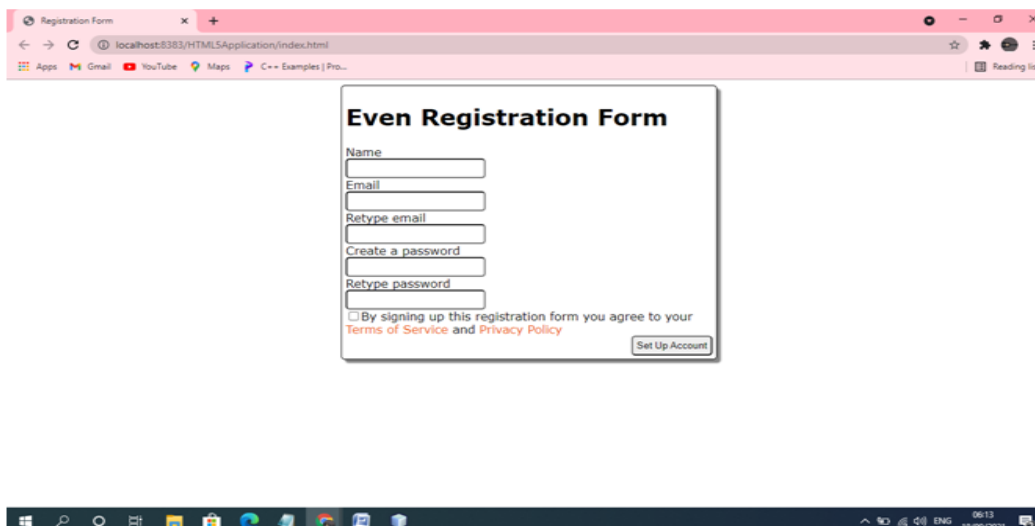
text-decoration: none;
color: #e51;

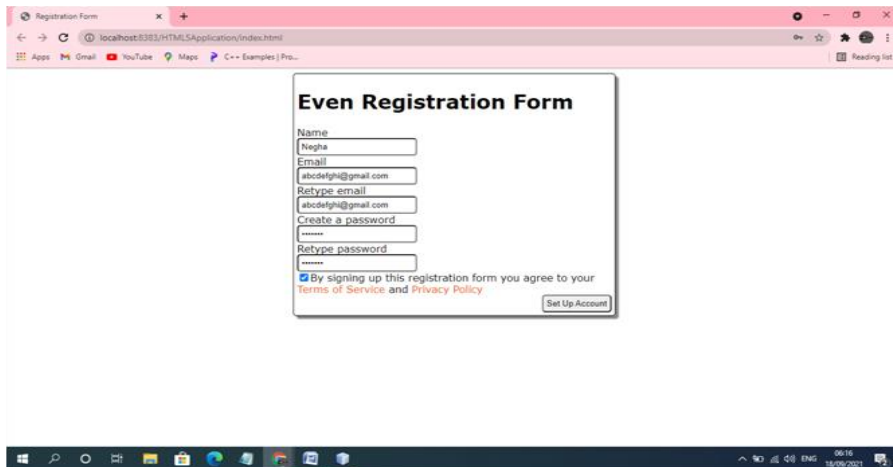
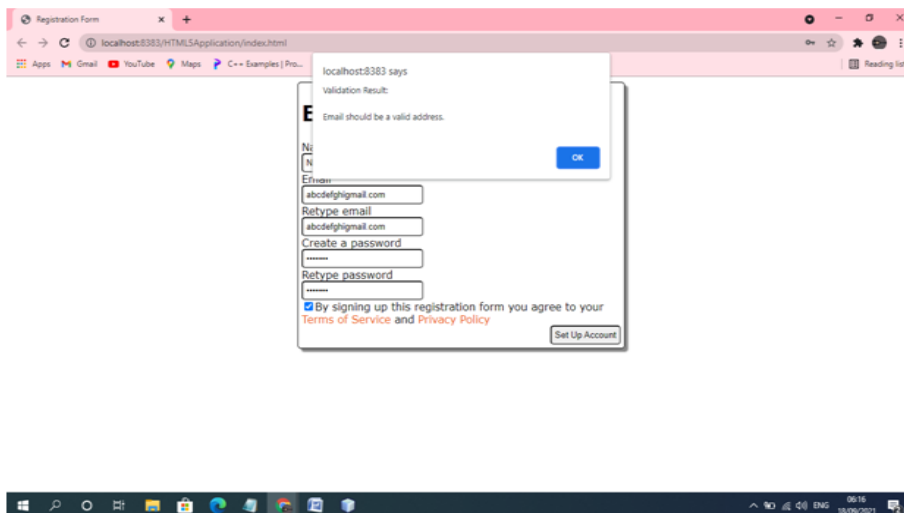
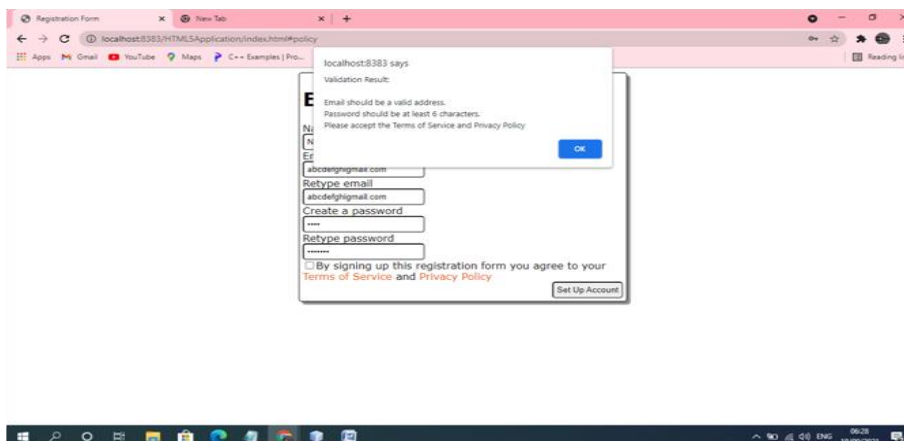
}

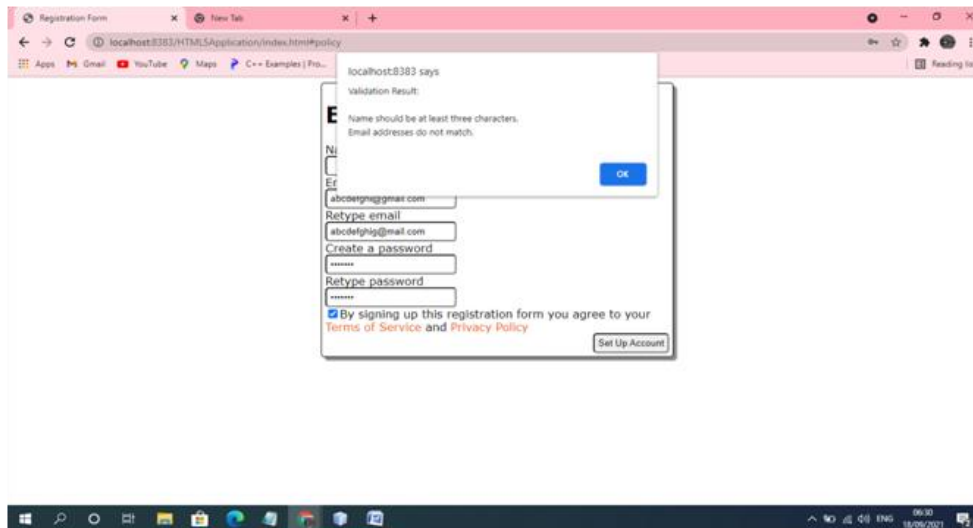
form {
width: 480px;
margin: auto;
padding: 5px;
border: solid black 1px;
border-radius: 5px;
box-shadow: 5px 5px 2px #888;

}
```

OUTPUT:





RESULT:

Thus the program for Event Registration and validation is implemented successfully.

2. BEAN CLASS

AIM:

To create a javascript application in an object oriented way using classes and modules. It should also use browser storage for persistence.

PROGRAM:

```
/*  
    *To change this license header, choose License Headers in Project Properties.  
    *To change this template file, choose Tools | Templates  
    *and open the template in the editor.  
    */  
package org.example.model;  
/**  
 *  
 * @author ADMIN  
 */  
public class Device {  
    private int id;  
    private String name;  
    private String status;  
    private String type;  
    private String description;  
    public Device() {  
    }  
    public int getId() {  
        return id;  
    }  
    public String getName() {  
        return name;  
    }  
    public String getStatus() {  
        return status;  
    }  
}
```

```
}  
public String getType() {  
    return type;  
}  
public String getDescription() {  
    return description;  
}  
public void setId(int id) {  
    this.id = id;  
}  
public void setName(String name) {  
    this.name = name;  
}  
public void setStatus(String status) {  
    this.status = status;  
}  
public void setType(String type) {  
    this.type = type;  
}  
public void setDescription(String description) {  
    this.description = description;  
}  
}
```

Devicesessionhandlerclass

```
/*  
  
 * To change this license header, choose License Headers in Project Properties.  
  
 * To change this template file, choose Tools | Templates  
  
 * and open the template in the editor.  
  
*/
```



```
package org.example.websocket;
import java.io.IOException;
import java.util.ArrayList;

import javax.faces.bean.ApplicationScoped;

import java.util.HashSet;
import java.util.List;
import java.util.Set;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.json.JsonObject;
import javax.json.spi.JsonProvider;
import javax.websocket.Session;
import org.example.model.Device;
/**
 *
 * @author ADMIN
 */
@ApplicationScoped
public class DeviceSessionHandler {
    private int deviceId = 0;
    private final Set<Session> sessions = new HashSet<>();
    private final Set<Device> devices = new HashSet<>();
    public void addSession(Session session) {
sessions.add(session);

        for (Device device : devices) {
JsonObjectaddMessage = createAddMessage(device);
sendToSession(session, addMessage);
        }
    }

    public void removeSession(Session session) {
```

```
sessions.remove(session);
    }
public List<Device>getDevices() {
    return new ArrayList<>(devices);
}

    public void addDevice(Device device) {
device.setId(deviceId);

devices.add(device);
deviceId++;

JsonObjectaddMessage = createAddMessage(device);
sendToAllConnectedSessions(addMessage);
    }

    public void removeDevice(int id) {

Device device = getDeviceById(id);

if (device != null) {
devices.remove(device);

JsonProvider provider = JsonProvider.provider();
JsonObjectremoveMessage = provider.createObjectBuilder()
.add("action", "remove")

.add("id", id)

.build();

sendToAllConnectedSessions(removeMessage);
    }
}

public void toggleDevice(int id) {

JsonProvider provider =

JsonProvider.provider(); Device device =
getDeviceById(id);
```

```
        if (device != null) {  
            if ("On".equals(device.getStatus())) {  
device.setStatus("Off");  
            } else {  
device.setStatus("On");  
            }  
  
        JsonObjectupdateDevMessage = provider.createObjectBuilder()  
        .add("action", "toggle")  
        .add("id", device.getId())  
        .add("status", device.getStatus())  
        .build();  
        sendToAllConnectedSessions(updateDevMessage);  
    }  
}  
  
private Device getDeviceById(int id) {  
    for (Device device : devices) {  
        if (device.getId() == id) {  
            return device;  
        }  
    }  
  
    return null;  
}  
  
private JsonObjectcreateAddMessage(Device device) {  
    JsonProvider provider = JsonProvider.provider();  
    JsonObjectaddMessage = provider.createObjectBuilder()  
    .add("action", "add")
```

```
.add("id", device.getId())

.add("name", device.getName())

.add("type", device.getType())

.add("status", device.getStatus())

.add("description", device.getDescription())

.build();

    return addMessage;

}

private void sendToAllConnectedSessions(JsonObject message) {

    for (Session session : sessions) {
sendToSession(session, message);

    }

}

private void sendToSession(Session session, JsonObject message) {

    try {
session.getBasicRemote().sendText(message.toString());

    } catch (IOException ex) {

sessions.remove(session);

        Logger.getLogger(DeviceSessionHandler.class.getName()).log(Level.
SEVERE, null, ex);

    }

}

}
```

Devicewebsocketservlet

/*

* To change this license header, choose License Headers in Project Properties.



```
* To change this template file, choose Tools | Templates
* and open the template in the editor.
*/

package org.example.websocket;

/**
 *
 * @author ADMIN
 */

import javax.faces.bean.ApplicationScoped;
import javax.websocket.OnClose;
import javax.websocket.OnError;
import javax.websocket.OnMessage;
import javax.websocket.OnOpen;
import javax.websocket.Session;
import javax.websocket.server.ServerEndpoint;
import javax.inject.Inject;
import java.io.StringReader; import javax.json.Json;
import javax.json.JsonObject;
import javax.json.JsonReader;
import org.example.model.Device;
import java.util.logging.Logger;
import java.util.logging.Level;

@ApplicationScoped
@ServerEndpoint("/actions")
public class DeviceWebSocketServer {
    @Inject
    private DeviceSessionHandlersessionHandler;

    @OnOpen
    public void open(Session session) {
        sessionHandler.addSession(session);
    }
}
```



```
@OnMessage
public void handleMessage(String message, Session session) {
    try (JsonReader reader = Json.createReader(new StringReader(message))) {
        JsonObject jsonMessage = reader.readObject();
        if ("add".equals(jsonMessage.getString("action"))) {
            Device device = new Device();
            device.setName(jsonMessage.getString("name"));
            device.setDescription(jsonMessage.getString("description"));
            device.setType(jsonMessage.getString("type"));
            device.setStatus("Off");
            sessionHandler.addDevice(device);
        }
        if ("remove".equals(jsonMessage.getString("action"))) {
            int id = (int) jsonMessage.getInt("id");
            sessionHandler.removeDevice(id);
        }
        if ("toggle".equals(jsonMessage.getString("action"))) {
            int id = (int) jsonMessage.getInt("id");
            sessionHandler.toggleDevice(id);
        }
    }
}

@OnClose
public void close(Session session) {
    sessionHandler.removeSession(session);
}

@OnError
public void onError(Throwable error) {
    Logger.getLogger(DeviceWebSocketServer.class.getName()).log(Level.
SEVERE, null, error);
}
}
```

INDEX.HTML

```
<!DOCTYPE html>

<!--
```

To change this license header, choose License Headers in Project Properties.

To change this template file, choose Tools | Templates

and open the template in the editor.

-->

<html>

<head>

<title></title>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<script src="websocket.js"></script>

<link rel="stylesheet" type="text/css" href="style.css">

</head>

<body>

<div id="wrapper">

<h1>Java Websocket Home</h1>

<p>Welcome to the Java WebSocket Home. Click the Add a device button to start adding devices.</p>

<div id="addDevice">

<div class="button">Add a device</div>

<form id="addDeviceForm">

<h3>Add a new device</h3>

Name: <input type="text" name="device_name" id="device_name">

Type:

<select id="device_type">

<option name="type" value="Appliance">Appliance</option>

```
<option name="type" value="Electronics">Electronics</option>

<option name="type" value="Lights">Lights</option>

<option name="type" value="Other">Other</option>

</select></span><br>
<span>Description:<br />
<textarea name="description" id="device_description" rows="2" cols="50"></
textarea>
</span><br>
<input type="button" class="button" value="Add" onclick=formSubmit()>
<input type="reset" class="button" value="Cancel" onclick=hideForm()>

</form>

</div>

<br />

<h3>Currently connected devices:</h3>

<div id="content">

</div>

</div>

</body>

</html>
```

STYLE.CSS

```
/*
```

To change this license header, choose License Headers in Project Properties.

To change this template file, choose Tools | Templates

and open the template in the editor.

```
*/
```

```
/*
```

Created on : 21 Sep, 2021, 1:35:40 PM

Author : ADMIN

*/

body {

font-family: Arial, Helvetica, sans-
serif; font-size: 80%;

background-color: #1f1f1f;

}

#wrapper {

width:

960px;

margin:

auto; text-

align: left;

color: #d9d9d9;

}

p {

text-align: left;

}

.button {

display: inline;

color: #fff;

background-color: #f2791d;

padding: 8px;

margin: auto;

border-radius:

8px;

-moz-border-radius: 8px;

-webkit-border-radius:

8px; box-shadow: none;

```
border: none;

}

.button:hover {

background-color: #ffb15e;

}

.button a, a:visited, a:hover, a:active {

color: #fff;

text-decoration: none;

}

#addDevice {

text-align: center;

width: 960px;

margin: auto;

margin-bottom: 10px;

}

#addDeviceForm {

text-align: left;

width: 400px;

margin: auto;

padding: 10px;

}

#addDeviceForm span {

display: block;

}

#content {

margin: auto;
```

```
        width: 960px;
    }

    .device {

        width: 180px;

        height: 110px;

        margin: 10px;

        padding: 16px;

color: #fff;

        vertical-align: top;
        border-radius: 8px;
        -moz-border-radius: 8px;
        -webkit-border-radius: 8px;

        display: inline-block;
    }

    .device.off {

        background-color: #c8cccf;

    }

    .device span {

display: block;

    }

    .deviceName {

        text-align: center;

        font-weight: bold;

        margin-bottom: 12px;

    }

    .removeDevice { margin-top: 12px;

text-align: center;
```

```
}  
  
.device.Appliance { background-color: #5eb85e;  
}  
  
.device.Appliance a:hover {  
color: #a1ed82;  
}  
  
.device.Electronics {  
background-color: #0f90d1;  
}  
.device.Electronics a:hover {  
color: #4badd1;  
}  
  
.device.Lights {  
background-color: #c2a00c;  
}  
  
.device.Lights a:hover {  
color: #fad232;  
}  
  
.device.Other {  
background-color: #db524d;  
}  
  
.device.Other a:hover {  
color: #ff907d;  
}  
  
.device a {  
text-decoration: none;  
}
```



```
.device a:visited, a:active, a:hover {
```

```
color: #fff;
```

```
}
```

```
.device a:hover {
```

```
text-decoration: underline;
```

```
}
```

WEBSOCKET.JS

```
/*
```

```
*To change this license header, choose License Headers in Project Properties.
```

```
*To change this template file, choose Tools | Templates
```

```
*and open the template in the editor.
```

```
*/
```

```
window.onload = init;
```

```
var socket = new WebSocket("ws://localhost:8080/WebsocketHome/actions");
```

```
socket.onmessage = onMessage;
```

```
function onMessage(event) {
```

```
    var device = JSON.parse(event.data);
```

```
    if (device.action === "add") {
```

```
printDeviceElement(device);
```

```
    }
```

```
    if (device.action === "remove") {
```

```
document.getElementById(device.id).remove();
```

```
        //device.parentNode.removeChild(device);
```

```
    }
```

```
    if (device.action === "toggle") {
```

```
        var node = document.getElementById(device.id);
```

```
        var statusText = node.children[2];
```

```

    if (device.status === "On") {
        statusText.innerHTML = "Status: " + device.status + " (<a href=\'#\''
        OnClick=toggleDevice(\" + device.id + \")>Turn off</a>\"");
    } else if (device.status === "Off") {
        statusText.innerHTML = "Status: " + device.status + " (<a href=\'#\''
        OnClick=toggleDevice(\" + device.id + \")>Turn on</a>\"");
    }
}

function addDevice(name, type, description) {
    var DeviceAction = {
        action: "add",
        name: name,
        type: type,
        description: description
    };

    socket.send(JSON.stringify(DeviceAction));
}

function removeDevice(element) {
    var id = element;
    var DeviceAction = {
        action: "remove",
        id: id
    };
    socket.send(JSON.stringify(DeviceAction));
}

function toggleDevice(element) {

```



```
var id = element;
var DeviceAction = {
    action: "toggle",
    id: id
};

socket.send(JSON.stringify(DeviceAction));

}

function printDeviceElement(device) {
    var content = document.getElementById("content");
    var deviceDiv = document.createElement("div");
    deviceDiv.setAttribute("id", device.id);
    deviceDiv.setAttribute("class", "device " + device.type);
    content.appendChild(deviceDiv);

    var deviceName = document.createElement("span");
    deviceName.setAttribute("class", "deviceName");
    deviceName.innerHTML = device.name;
    deviceDiv.appendChild(deviceName);

    var deviceType = document.createElement("span");
    deviceType.innerHTML = "<b>Type:</b> " + device.type;
    deviceDiv.appendChild(deviceType);

    var deviceStatus = document.createElement("span");
    if (device.status === "On") {
        deviceStatus.innerHTML = "<b>Status:</b> " + device.status + " (<a href='\"#\"'\"
        OnClick=toggleDevice(\" + device.id + \")>Turn off</a>\"");

        } else if (device.status === "Off") {

        deviceStatus.innerHTML = "<b>Status:</b> " + device.status + " (<a href='\"#\"'\"
        OnClick=toggleDevice(\" + device.id + \")>Turn on</a>\"");
        //deviceDiv.setAttribute("class", "device off");

    }

    deviceDiv.appendChild(deviceStatus);
```

```
var deviceDescription = document.createElement("span");
deviceDescription.innerHTML = "<b>Comments:</b> " + device.description;
deviceDiv.appendChild(deviceDescription);

var removeDevice = document.createElement("span");
removeDevice.setAttribute("class", "removeDevice");
removeDevice.innerHTML = "<a href='\"#\"' OnClick=removeDevice(\"\" + device.id + \")>Remove device</a>";
deviceDiv.appendChild(removeDevice);
}

function showForm() {
document.getElementById("addDeviceForm").style.display = "";
}

function hideForm() {
document.getElementById("addDeviceForm").style.display =
"none";
}

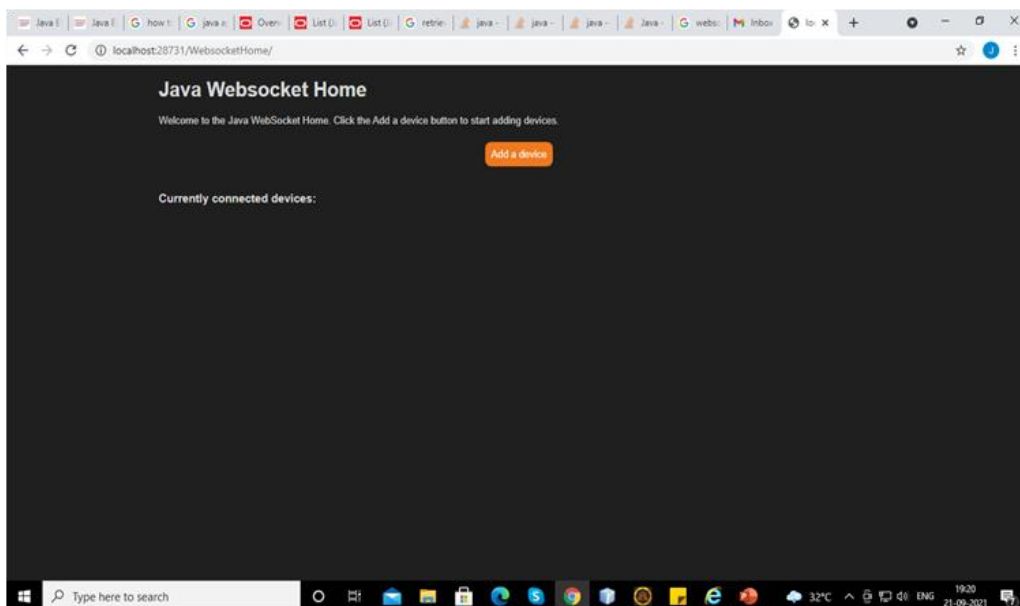
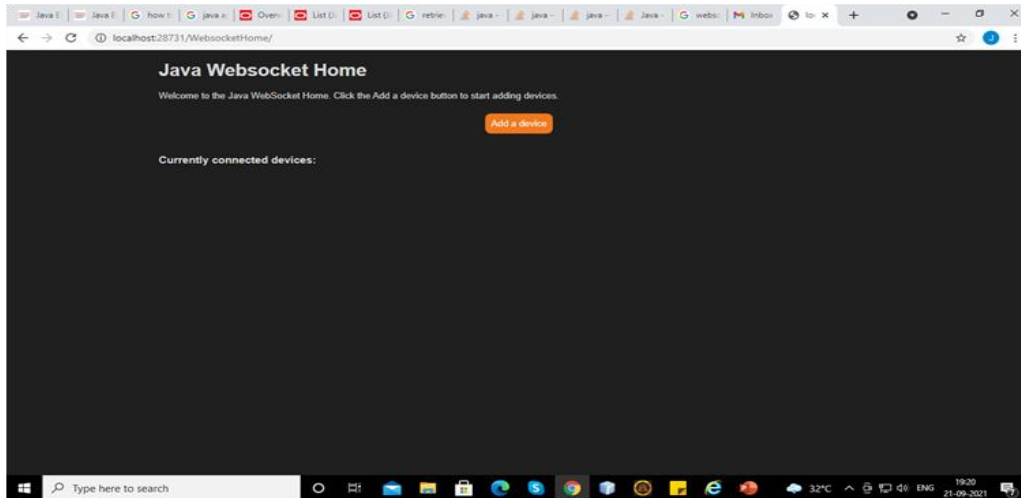
function formSubmit() {

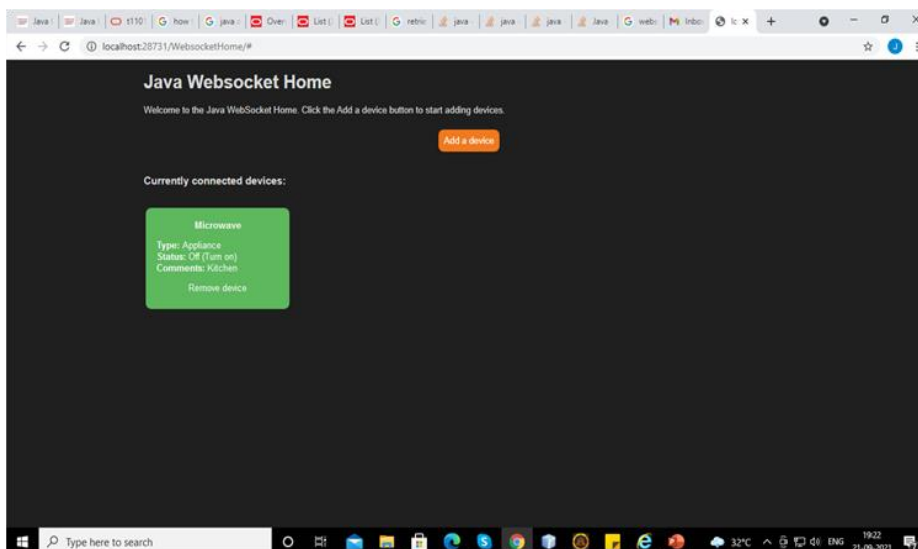
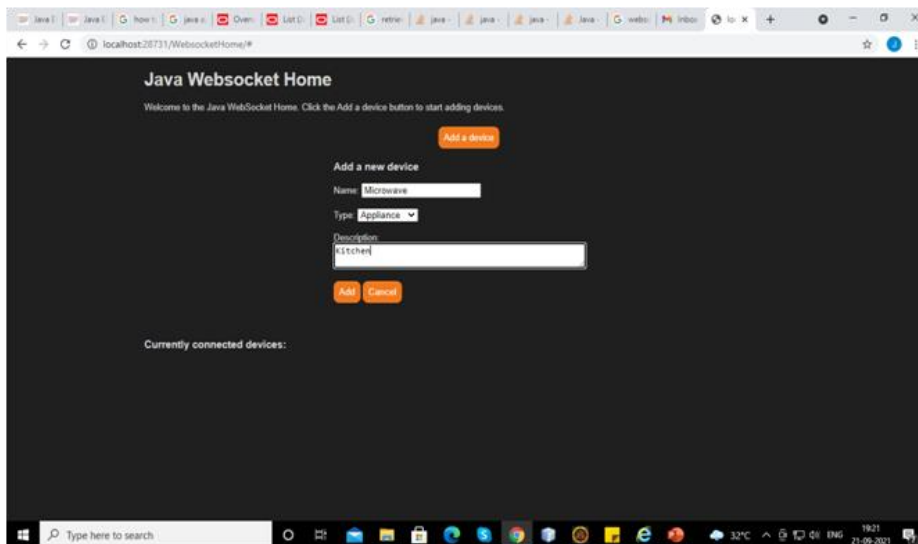
var form = document.getElementById("addDeviceForm");
var name = form.elements["device_name"].value;
var type = form.elements["device_type"].value;
var description = form.elements["device_description"].value;
hideForm();

document.getElementById("addDeviceForm").reset();
addDevice(name, type, description);
}

function init() {
hideForm();
}
```

OUTPUT:





RESULT

Thus the program for javascript application in an object oriented way using classes and modules is implemented successfully.

3. WEB APPLICATION USING GRADLE

AIM:

To build a web application using Gradle.

PROGRAM:

```
<%@tag description="Default Page template" pageEncoding="UTF-8"%>
<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<% @taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>
<% @attribute name="title" required="false" %>

<!DOCTYPE html>

<html>

<head>

    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Stormpath Webapp Sample | <c:out value="${!empty title ? title :
''}"/></title>
    <link href="//maxcdn.bootstrapcdn.com/bootstrap/3.3.2/css/bootstrap.min.
css" rel="stylesheet">
    <%-- <link href="${pageContext.request.contextPath}/assets/css/style.css"
rel="stylesheet" --%>
    <!-- HTML5 shim and Respond.js for IE8 support of HTML5 elements and
media queries -->
    <!-- WARNING: Respond.js doesn't work if you view the page via file:// -->
    <!--[if lt IE 9]>
        <script src="https://oss.maxcdn.com/html5shiv/3.7.2/html5shiv.min.js"></
script>
        <script src="https://oss.maxcdn.com/respond/1.4.2/respond.min.js"></script>
    <![endif]-->
    <!-- jQuery (necessary for Bootstrap's JavaScript plugins) -->
```

```

<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.2/jquery.min.
js"></script>
<script src="//maxcdn.bootstrapcdn.com/bootstrap/3.3.2/js/bootstrap.min.
js"></script>
</head>
<body>

<div class="container">

<div class="header">

<ul class="nav nav-pills pull-right">

<c:set var="uri" value="${requestScope['javax.servlet.forward.
request_uri']}'"/>

<li<c:if test="${fn:endsWith(uri, '/')}"> class="active"</c:if><a
href="${pageContext.request.contextPath}"/>Home</a></li>

<%-- Change upper right context menu depending on if the user is
logged in or not: --%>

<c:choose>

<c:when test="${!empty account}">

<li<c:if test="${fn:endsWith(uri, 'dashboard')}"> class="active"</
c:if><a href="${pageContext.request.contextPath}/dashboard">Dashboard</
a></li>

<li><a href="${pageContext.request.contextPath}/logout">Logout</
a></li>

</c:when>

<c:otherwise>

<li<c:if test="${fn:endsWith(uri, 'login')}"> class="active"</
c:if><a href="${pageContext.request.contextPath}/login">Login</a></li>

</c:otherwise>

</c:choose>

</ul>

<h3 class="text-muted">Stormpath Webapp Sample</h3>

</div>

```




```
<jsp:doBody/>

</div>

</body>

</html>
```

Home.jsp:

```
<% @ page session="false"%>

<% @ page contentType="text/html;charset=UTF-8" pageEncoding="UTF-8"
%>

<% @ taglib prefix="t" tagdir="/WEB-INF/tags" %>

<% @ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

<t:page>

  <jsp:attribute name="title">Welcome!</jsp:attribute>

  <jsp:body>

    <div class="jumbotron" id="welcome">

      <h1>Welcome to the Stormpath Webapp Sample Application!</h1>

      <p class="lead">

        <br/>

        <br/>

        Welcome to this <i>gloriously simple</i>

        <a href="https://docs.stormpath.com/java/servlet-plugin/">Stormpath

Java Webapp</a> sample application!

        <ul>

          <li>First, take a look through this very basic site.</li>

          <li>Then, check out this project's source code

            <a href="https://github.com/stormpath/stormpath-sdk-java/

examples/servlet">on GitHub</a>.</li>

          <li>Lastly, integrate Stormpath into your own sites!</li>
```


</p>

<h2>What This Sample App Demonstrates</h2>

<p>This simple application demonstrates how easy it is to register, login, and securely authenticate

users on your website using the Stormpath Servlet Plugin.</p>

<p>Not a Stormpath user yet? Go
signup

now!</p>

<p class="bigbutton"><a class="bigbuttonbtn btn-danger"
href="{pageContext.request.contextPath}/register" role="button">Register</
a></p>

</div>

</jsp:body>

</t:page>

HomeController.java

```
package tutorial;
```

```
import javax.servlet.ServletException;
```

```
import javax.servlet.http.HttpServlet;
```

```
import javax.servlet.http.HttpServletRequest;
```

```
import javax.servlet.http.HttpServletResponse;
```

```
import java.io.IOException;
```

```
public class HomeController extends HttpServlet {
```

```
    public static final String VIEW_TEMPLATE_PATH = "/WEB-INF/jsp/home.
```

```
jsp”;
```

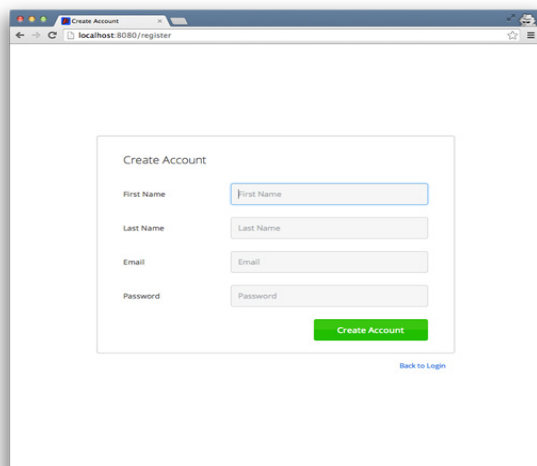
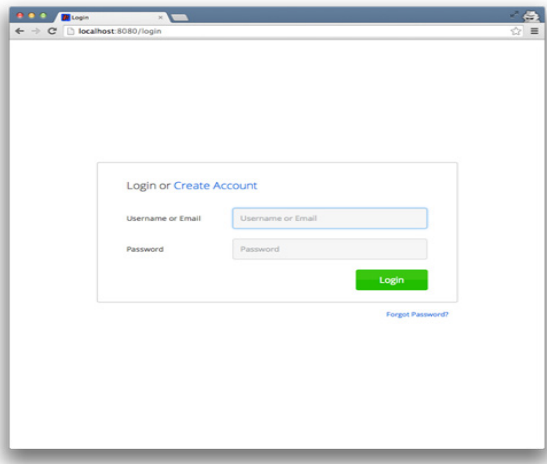
```
@Override
```

```
protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException,  
IOException {
```

```
req.getRequestDispatcher(VIEW_TEMPLATE_PATH).forward(req, resp);
```

```
}}
```

OUTPUT:



RESULT:

Thus the program for web application using gradle is implemented successfully.

4. CHAT APPLICATION USING WEBSOCKETS

AIM:

To create a chat application using web socket in java.

PROGRAM:

ChatServer.java

/*

*To change this license header, choose License Headers in Project Properties.

*To change this template file, choose Tools | Templates

*and open the template in the editor.

*/

package chat;

import java.io.DataInputStream;

import java.io.DataOutputStream;

import java.net.ServerSocket;

import java.net.Socket;

/**

*

*@author sabar

*/

public class chatserver extends javax.swing.JFrame {

static ServerSocket ss;

static Socket s;

static DataInputStream dis;

static DataOutputStream dout;

/**

* Creates new form chatserver

```
*/  
  
public chatserver() {  
initComponents();  
}  
/**  
  
* This method is called from within the constructor to initialize the form.  
* WARNING: Do NOT modify this code. The content of this method is always  
* regenerated by the Form Editor.  
*/  
  
@SuppressWarnings("unchecked")  
// <editor-fold defaultstate="collapsed" desc="Generated Code">  
private void initComponents() {  
    jScrollPane1 = new javax.swing.JScrollPane();  
    msg_area = new javax.swing.JTextArea();  
    msg_text = new javax.swing.JTextField();  
    msg_send = new javax.swing.JButton();  
    jLabel1 = new javax.swing.JLabel();  
    setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);  
    msg_area.setColumns(20);  
    msg_area.setRows(5);  
    jScrollPane1.setViewportView(msg_area);  
    msg_send.setText("Send");  
  
    jLabel1.setFont(new java.awt.Font("Tahoma", 1, 24)); // NOI18N  
    jLabel1.setText("Server");  
    javax.swing.GroupLayout layout = new javax.swing.  
    GroupLayout(getContentPane());  
    getContentPane().setLayout(layout);  
  
    layout.setHorizontalGroup(  
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
```



```
.addComponent(jScrollPane1, javax.swing.GroupLayout.DEFAULT_SIZE, 452,
Short.MAX_VALUE)

.addGroup(layout.createSequentialGroup())

.addContainerGap()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
LEADING))

.addGroup(layout.createSequentialGroup())

.addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 124,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addGap(0, 0, Short.MAX_VALUE))

.addGroup(layout.createSequentialGroup())

.addComponent(msg_text)

.addGap(18, 18, 18)

.addComponent(msg_send, javax.swing.GroupLayout.PREFERRED_SIZE, 84,
javax.swing.GroupLayout.PREFERRED_SIZE)))

.addContainerGap())

);

layout.setVerticalGroup(
layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
.addGroup(layout.createSequentialGroup())

.addContainerGap()

.addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 28,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.
UNRELATED)

.addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 240,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addGap(31, 31, 31)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.  BASELINE)

.addComponent(msg_text, javax.swing.GroupLayout.PREFERRED_SIZE, 41,
javax.swing.GroupLayout.PREFERRED_SIZE)
```



```
.addComponent(msg_send, javax.swing.GroupLayout.PREFERRED_SIZE, 41,
javax.swing.GroupLayout.PREFERRED_SIZE))

.addContainerGap()

);
pack();
} // </editor-fold>
private void msg_sendActionPerformed(java.awt.event.ActionEvent evt)
{
try{
    String msg="";
    msg=msg_text.getText();
    dout.writeUTF(msg);
    msg_text.setText("");
    }
catch(Exception e)
    {
        //handle the exception
    }
}

/**
 * @param args the command line arguments
 */

public static void main(String args[]) {

    /* Set the Nimbus look and feel */
    //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code
(optional) ">

    /* If Nimbus (introduced in Java SE 6) is not available, stay with the default
look and feel.
    * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
```

```

*/

try {
    for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.
        UIManager.getInstalledLookAndFeels()) {
        if ("Nimbus".equals(info.getName())) {
            javax.swing.UIManager.setLookAndFeel(info.getClassName());
            break;
        }
    }
} catch (ClassNotFoundException ex) {
    java.util.logging.Logger.getLogger(chatserver.class.getName()).log(java.util.
        logging.Level.SEVERE, null, ex);
} catch (InstantiationException ex) {
    java.util.logging.Logger.getLogger(chatserver.class.getName()).log(java.util.
        logging.Level.SEVERE, null, ex);
} catch (IllegalAccessException ex) {
    java.util.logging.Logger.getLogger(chatserver.class.getName()).log(java.util.
        logging.Level.SEVERE, null, ex);
} catch (javax.swing.UnsupportedLookAndFeelException ex) {
    java.util.logging.Logger.getLogger(chatserver.class.getName()).log(java.util.
        logging.Level.SEVERE, null, ex);
}
//</editor-fold>

/* Create and display the form */
java.awt.EventQueue.invokeLater(new Runnable() {
    public void run() {
        new chatserver().setVisible(true);
    }
});

try {
    String msgin = "";

```



```
ss = new ServerSocket(1201);

s = ss.accept();

dis = new DataInputStream(s.getInputStream());
dout = new DataOutputStream(s.getOutputStream());
while (!msgin.equals("exit")) {
msgin = dis.readUTF();
msg_area.setText(msg_area.getText() + "\n Client: " + msgin);

}

} catch (Exception e) {

//handle the exception here

}

}

// Variables declaration - do not modify

private javax.swing.JLabel jLabel1;
private javax.swing.JScrollPane jScrollPane1;
private static javax.swing.JTextAreamsg_area;
private javax.swing.JButtonmsg_send;
private javax.swing.JTextFieldmsg_text;
// End of variables declaration

}
```

ChatClient.java:

```
package chat;

import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.net.Socket;
/**
 *
 * @author sabar
```



```

*/

public class chatclient extends javax.swing.JFrame { static Socket s;
    static DataInputStream dis;
    static DataOutputStream dout;
    /**
     * Creates new form chatclient
     */

    public chatclient() {
initComponents();
    }

    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */

    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {
        jScrollPane1 = new javax.swing.JScrollPane();
        msg_area = new javax.swing.JTextArea();
        msg_text = new javax.swing.JTextField();
        msg_send = new javax.swing.JButton();
        jLabel1 = new javax.swing.JLabel();
        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
        msg_area.setColumns(20);
        msg_area.setRows(5);
        jScrollPane1.setViewportView(msg_area);
        msg_send.setText("Send");
        jLabel1.setFont(new java.awt.Font("Tahoma", 1, 18)); // NOI18N
        jLabel1.setText("Client");

```



```
javax.swing.GroupLayout layout = new javax.swing.  
GroupLayout(getContentPane());  
getContentPane().setLayout(layout);  
  
layout.setHorizontalGroup(  
layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  
.addComponent(jScrollPane1)  
.addGroup(layout.createSequentialGroup()  
.addGap(10, 10, 10)  
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.  
LEADING)  
.addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 118,  
javax.swing.GroupLayout.PREFERRED_SIZE)  
.addGroup(layout.createSequentialGroup()  
.addComponent(msg_text, javax.swing.GroupLayout.PREFERRED_SIZE,  
320, javax.swing.GroupLayout.PREFERRED_SIZE)  
.addGap(10, 10, 10)  
.addComponent(msg_send)))  
.addGap(47, 47, Short.MAX_VALUE))  
);  
  
layout.setVerticalGroup(  
layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  
.addGroup(layout.createSequentialGroup()  
.addGap(10, 10, 10)  
.addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 20,  
javax.swing.GroupLayout.PREFERRED_SIZE)  
.addGap(10, 10, 10)  
.addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE,  
275, javax.swing.GroupLayout.PREFERRED_SIZE)  
.addGap(10, 10, 10)  
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
```

LEADING)

```
.addComponent(msg_text)

.addComponent(msg_send,    javax.swing.GroupLayout.DEFAULT_SIZE,    39,
Short.MAX_VALUE))

.addContainerGap()

);

pack();

} // </editor-fold>

private void msg_sendActionPerformed(java.awt.event.ActionEvent evt)
{
try{
    String msg="";
    msg=msg_text.getText();
    dout.writeUTF(msg);
    msg_text.setText("");
    }
catch(Exception e)
    {
        //handle the exception
    }
}

/**
 * @param args the command line arguments
 */
public static void main(String args[]) {
    /* Set the Nimbus look and feel */
    //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code
(optional) ">
    /* If Nimbus (introduced in Java SE 6) is not available, stay with the
default look and feel.
    * For details see http://download.oracle.com/javase/tutorial/uiswing/
lookandfeel/plaf.html
    */
    try {
```

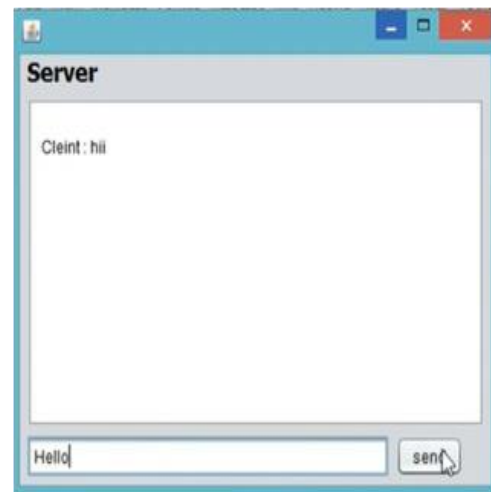
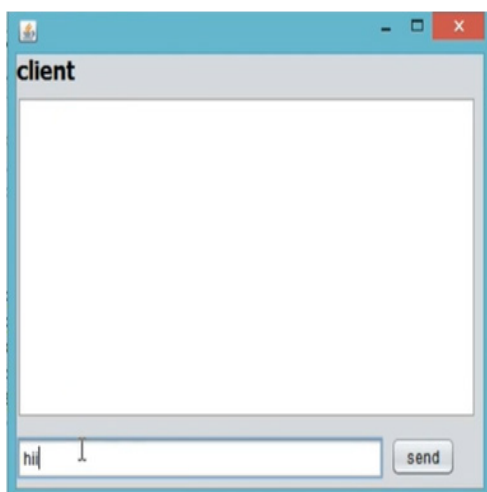
```
        for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.
UIManager.getInstalledLookAndFeels()) {
            if ("Nimbus".equals(info.getName())) {
                javax.swing.UIManager.setLookAndFeel(info.getClassName());
                break;
            }
        }
    } catch (ClassNotFoundException ex) {
        java.util.logging.Logger.getLogger(chatclient.class.getName()).log(java.util.
logging.Level.SEVERE, null, ex);
    } catch (InstantiationException ex) {
        java.util.logging.Logger.getLogger(chatclient.class.getName()).log(java.util.
logging.Level.SEVERE, null, ex);
    } catch (IllegalAccessException ex) {
        java.util.logging.Logger.getLogger(chatclient.class.getName()).log(java.util.
logging.Level.SEVERE, null, ex);
    } catch (javax.swing.UnsupportedLookAndFeelException ex) {
        java.util.logging.Logger.getLogger(chatclient.class.getName()).log(java.util.
logging.Level.SEVERE, null, ex);
    }
}
//</editor-fold>

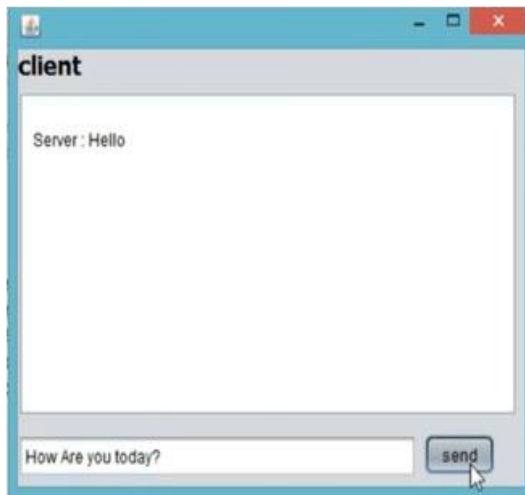
/* Create and display the form */
java.awt.EventQueue.invokeLater(new Runnable() {
    public void run() {
        new chatclient().setVisible(true);
    }
});
try {
    String msgin = "";
```

s=new Socket("127.0.0.1",1201);//ip address is of localhost because server is running on the same machine

```
dis = new DataInputStream(s.getInputStream());
dout = new DataOutputStream(s.getOutputStream());
while (!msgin.equals("exit")) {
msgin = dis.readUTF();
msg_area.setText(msg_area.getText() + "\n Server: " + msgin);
}
} catch (Exception e) {
//handle the exception here
}
}
// Variables declaration - do not modify
private javax.swing.JLabel jLabel1;
private javax.swing.JScrollPane jScrollPane1;
private static javax.swing.JTextAreamsg_area;
private javax.swing.JButtonmsg_send;
private javax.swing.JTextFieldmsg_text;
// End of variables declaration
}
```

OUTPUT:





RESULT:

Hence the above code is implemented successfully and output is verified.



5. SPRING MVC APPLICATION

AIM:

Create a spring MVC applications. The application should have handle form validation, file upload and session tracking.

PROGRAM:

Web.xml:

```
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee" xmlns:xsi="http://www.
w3.org/2001/XMLSchema"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaeehttp://xmlns.jcp.org/ xml/ns/javaee/web-
app_3_1.xsd" version="3.1">
<display-name>FileUpload</display-name>

<!-- Spring MVC dispatcher servlet -->

<servlet>

<servlet-name>mvc-dispatcher</servlet-name>

<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-
class>

<init-param>

<param-name>contextConfigLocation</param-name>

<param-value>

    /WEB-INF/spring-mvc.xml,

</param-value>

</init-param>

<load-on-startup>1</load-on-startup>

</servlet>

<servlet-mapping>

    <servlet-name>mvc-dispatcher</servlet-name>

    <url-pattern>/</url-pattern>

</servlet-mapping>

<listener>

<listener-class>org.springframework.web.context.ContextLoaderListener</
```




```
listener-class>
```

```
</listener>
```

```
</web-app>
```

uploadFile.jsp:

```
<% @ page language="java" contentType="text/html; charset=ISO-8859-1"
```

```
pageEncoding="ISO-8859-1"%>
```

```
<% @ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
```

```
<% @ taglib prefix="spring" uri="http://www.springframework.org/tags"%>
```

```
<% @ taglib prefix="form" uri="http://www.springframework.org/tags/
form"%>
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
```

```
<html>
```

```
<head>
```

```
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
```

```
<title>Hello</title>
```

```
</head>
```

```
<body>
```

```
<form:form method="POST" action="uploadFile" enctype="multipart/
formdata" modelAttribute="fileUploadModel">
```

```
File to Upload: <input type="file" name="file"></br> </br>
```

```
Name: <input type="text" name="name"></br /> </br>
```

```
<input type="submit" value="Upload"></br>
```

```
<form:errors path="file" style="color:red;"/>
```

```
</form:form>
```

```
</body>
```

```
</html>
```

FileuploadSucess.jsp

```
<html>
```

```
<body>
```

```
<h2>Spring MVC file upload Success</h2>
```

```
FileName :<strong> ${ fileName} </strong> - Uploaded Successful.
```

```
</body>
```

```
</html>
```

FileuploadFailure.jsp:

```
<html>
```

```
<body>
```

```
<h2>Spring MVC file upload failure</h2>
```

```
FileName :<strong> ${fileName} </strong> - Upload error due to some technical  
issue Please try again later.
```

```
</body>
```

```
</html>
```

Controller.jsp:

```
import java.io.BufferedOutputStream;
import java.io.File;
import java.io.FileOutputStream;
import javax.servlet.ServletContext;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.context.ServletContextAware;
import org.springframework.web.multipart.MultipartFile;
import org.springframework.web.servlet.ModelAndView;
import com.kb.model.FileUploadModel;
import com.kb.validator.CustomFileValidator;

@Controller
```



```
public class FileUploadController implements ServletContextAware {
    private ServletContext servletContext;

    @Autowired
    CustomFileValidator customFileValidator;

    @RequestMapping(value = "/uploadFile", method = RequestMethod.GET)
    public String uploadFileFormDisplay(Model model) {
        model.addAttribute("fileUploadModel", new FileUploadModel());
        return "uploadFile";
    }

    @RequestMapping(value = "/uploadFile", method = RequestMethod.POST)
    public String uploadFileHandler(Model model, @
    ModelAttribute FileUploadModel fileUploadModel,
    BindingResult bindingResult) {
        MultipartFile file = fileUploadModel.getFile();
        customFileValidator.validate(fileUploadModel, bindingResult);
        if (bindingResult.hasErrors()) { return "uploadFile";
        }

        String fileName = file.getOriginalFilename();
        model.addAttribute("fileName", fileName);
        try {
            byte[] bytes = file.getBytes();
            BufferedOutputStream stream = new BufferedOutputStream(
            new FileOutputStream(new File(servletContext.getRealPath("/") + "/" +
            fileName)));
            stream.write(bytes);
            stream.close();
            return "fileUploadSuccess";
        } catch (Exception e) {
            return "fileUploadFailure";
        }
    }
}
```

```

    }

    public void setServletContext(ServletContext servletContext) {
this.servletContext = servletContext;
    }
}

```

Validation.jsp:

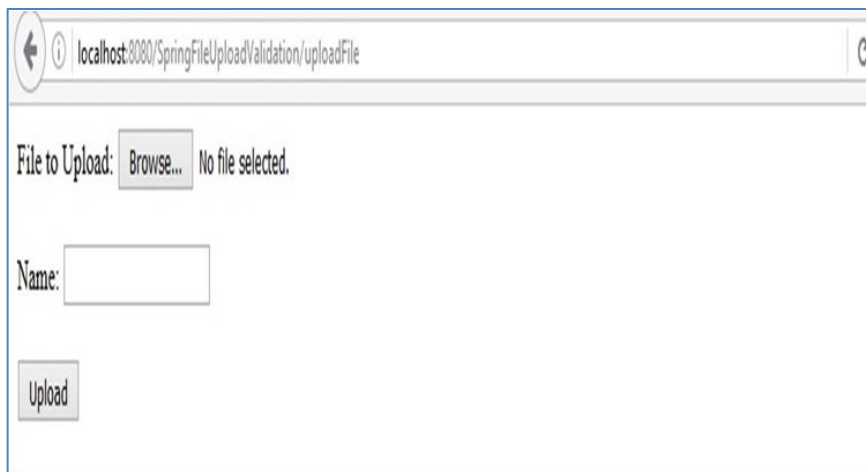
```

import org.springframework.stereotype.Component;
import org.springframework.validation.Errors;
import org.springframework.validation.Validator;
import org.springframework.web.multipart.MultipartFile;
import com.kb.model.FileUploadModel;
@Component
public class CustomFileValidator implements Validator{
    public static final String PNG_MIME_TYPE="image/png";
    public static final long TEN_MB_IN_BYTES = 10485760;
    @Override
    public boolean supports(Class<?> clazz) {
        return FileUploadModel.class.isAssignableFrom(clazz);
    }
    @Override
    public void validate(Object target, Errors errors) {
        FileUploadModel fileUploadModel = (FileUploadModel)target;
        MultipartFile file = fileUploadModel.getFile();
        if(file.isEmpty()){
            errors.rejectValue("file", "upload.file.required");
        }
        else if(!PNG_MIME_TYPE.equalsIgnoreCase(file.getContentType())){
            errors.rejectValue("file", "upload.invalid.file.type");
        }
    }
}

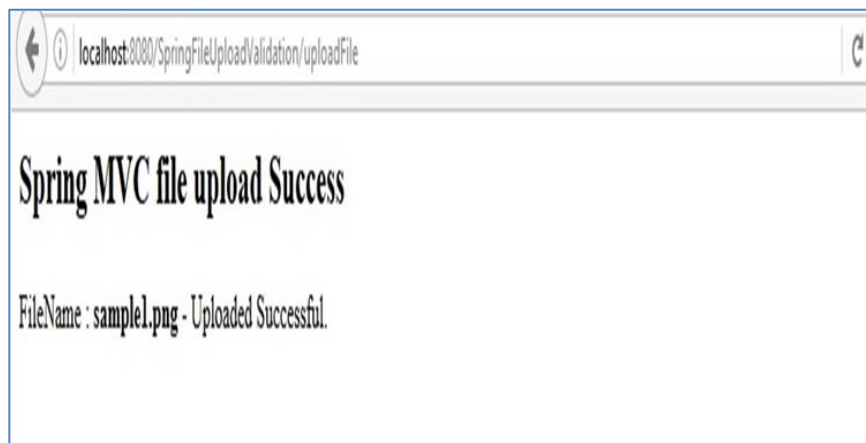
```

```
else if(file.getSize() > TEN_MB_IN_BYTES){  
errors.rejectValue("file", "upload.exceeded.file.size");  
}  
}  
}
```

OUTPUT:



The screenshot shows a web browser window with the address bar displaying 'localhost:8080/SpringFileUploadValidation/uploadFile'. The page content includes a 'File to Upload:' label, a 'Browse...' button, and the text 'No file selected.'. Below this is a 'Name:' label followed by an empty text input field. At the bottom of the form is an 'Upload' button.



The screenshot shows the same web browser window after a successful upload. The page displays the message 'Spring MVC file upload Success' in a large, bold font. Below this message, it says 'FileName : sample1.png - Uploaded Successful.'.

RESULT:

Thus the application should have handle form validation, file upload, session tracking using spring MVC application has been executed successfully.



6. RESTful SPRING BOOT APPLICATION

AIM:

To implement a RESTful Spring Boot application using Spring REST, Spring Security and Spring Cache.

PROGRAM:

Model:

Coupon.java

```
package com.example.couponservices.model;

import lombok.Data;
import javax.persistence.*;
import java.math.BigDecimal;

@Entity
@Data

public class Coupon {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "identity")
    private Long identity;
    @Column(name = "code_number")
    private String code_number;
    @Column(name = "discount_percent")
    private BigDecimal discount_percent;
    @Column(name = "exp_date")
    private String expDate;
}
```

Role.java

```
package com.example.couponservices.model;

import java.util.Set;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
```



```
import javax.persistence.GenerationType;

import javax.persistence.Id;
import javax.persistence.ManyToMany;
import org.springframework.security.core.GrantedAuthority;

@Entity
public class Role implements GrantedAuthority {
    private static final long serialVersionUID = 1L;

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long identity;
    private String first_name;

    @ManyToMany(mappedBy = "roles")
    private Set<User> users;

    public Long getIdentity() {
        return id;
    }

    public void setIdentity(Long id) {
        this.id = id;
    }

    public String getFirstName() {
        return name;
    }

    public void setFirstName(String name) {
        this.name = name;
    }

    @Override
    public String getAuthority() { return name;
    }
}
```

User.java



```
package com.example.couponservices.model;

import java.util.Set;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.JoinTable;
import javax.persistence.ManyToMany;

@Entity
public class User {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String firstName;
    private String lastName;
    private String email_id;
    private String password;
    @ManyToMany(fetch = FetchType.EAGER)
    @JoinTable(name = "user_role", joinColumns = @JoinColumn(name =
"user_id"), inverseJoinColumns = @JoinColumn(name = "role_id"))
    private Set<Role> roles;

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getFirstName() {
        return firstName;
    }
}
```



```
    }

    public void setFirstName(String firstName) {
this.firstName = firstName;
    }

    public String getLastName() {
        return lastName;
    }

    public void setLastName(String lastName) {
this.lastName = lastName;
    }

    public String getEmailId() {
        return email;
    }

    public void setEmail(String email) {
this.email = email;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
this.password = password;
    }

    public Set<Role>getRoles() {
        return roles;
    }

    public void setRoles(Set<Role> roles) {
this.roles = roles;
    }
}
```



View:

CouponRepo.java

```
package com.example.couponservices.repository;

import com.example.couponservices.model.Coupon;
import org.springframework.data.jpa.repository.JpaRepository;

public interface CouponRepo extends JpaRepository<Coupon,Long> {

    Coupon findByCode(String code);

}
```

RoleRepo.java

```
package com.example.couponservices.repository;

import com.example.couponservices.model.Role;
import com.example.couponservices.model.User;
import org.springframework.data.jpa.repository.JpaRepository;

public interface RoleRepo extends JpaRepository<Role,Long> {

}
```

CouponRepo.java

```
package com.example.couponservices.repository;

import com.example.couponservices.model.Coupon;
import org.springframework.data.jpa.repository.JpaRepository;

public interface CouponRepo extends JpaRepository<Coupon,Long> {

    Coupon findByCode(String code);

}
```

View:

CouponRestController.java

```
package com.example.couponservices.controller;

import com.example.couponservices.model.Coupon;
import com.example.couponservices.repository.CouponRepo;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;
```



@RestController

```
@RequestMapping("/couponapi")
public class CouponRestController {
    @Autowired
    CouponRepocouponRepo;
    @PostMapping("/coupons")
    public Coupon create(@RequestBody Coupon coupon) {
        return couponRepo.save(coupon);
    }
    @GetMapping("/coupons/{code}")
    public Coupon getCoupon(@PathVariable String code) {
        return couponRepo.findByCode(code);
    }
}
```

Security:

UserDetailsServiceImp.java

```
package com.example.couponservices.security;
import com.example.couponservices.model.User;
import com.example.couponservices.repository.UserRepo;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.core.userdetails.Username
NotFoundException;
import org.springframework.stereotype.Service;

@Service
public class UserDetailsServiceImpl implements UserDetailsService{
    @Autowired
    private UserRepouserRepo;
    @Override
```



```
public UserDetailsloadUserByUsername(String username) throws
UsernameNotFoundException {
    User user = userRepo.findByEmail(username);
    if(user == null)
        throw new UsernameNotFoundException("User Not found for email " +
        username);
    return new org.springframework.security.core.userdetails.User(user.
    getEmail(),user.getPassword(),user.getRoles());
}
}
```

WebSecurityConfig.java

```
package com.example.couponservices.security;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.http.HttpMethod;
import org.springframework.security.config.annotation.authentication.builders.
AuthenticationManagerBuilder;

import org.springframework.security.config.annotation.web.builders.
HttpSecurity;

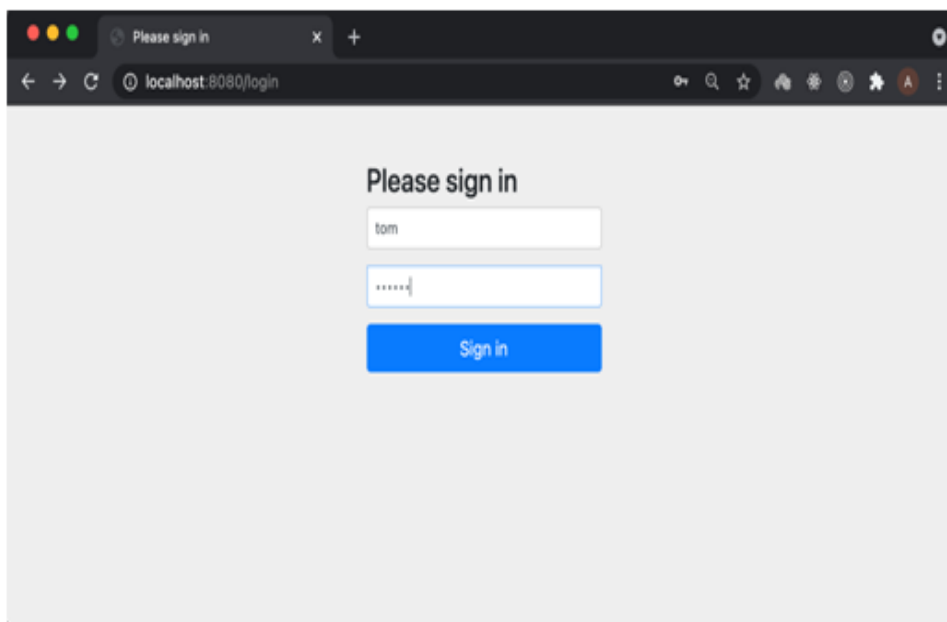
import org.springframework.security.config.annotation.web.configuration.
WebSecurityConfigurerAdapter;

import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.crypto.password.PasswordEncoder;

@Configuration
public class SWebSecurityConfig extends WebSecurityConfigurerAdapter {
    @Autowired
    private UserDetailsServiceImpl userDetailsService;
    @Override
    protected void configure(AuthenticationManagerBuilder auth) throws
Exception {
```

```
auth.userService(userDetailsService);
}
@Override
protected void configure(HttpSecurity http) throws Exception {
http.httpBasic();
http.authorizeRequests().mvcMatchers(HttpMethod.GET, "/couponapi/
coupons/{code:^(A-Z)*$}").hasAnyRole("USER", "ADMIN")
.mvcMatchers(HttpMethod.POST, "/couponapi/coupons").hasRole("ADMIN").
anyRequest().denyAll()
.and().csrf().disable();
}
@Bean
public PasswordEncoderpasswordEncoder() {
return new BCryptPasswordEncoder();
}
}
```

OUTPUT:





RESULT:

Thus the RESTful Spring Boot application using Spring REST, Spring Security and Spring Cache has been executed successfully.



7. HIBERNATE

AIM:

To design a complex system using JPA and Hibernate.

EMPLOYEE:

```
package Proj

import javax.persistence.*;
import Proj.Account;

@Entity
@Table(name="employee")
public class Employee;
{
    @Id
    @GeneratedValue(strategy=GenerationType.AUTO)
    @Column(name = "id")
    private int id;
    @Column(name="email")
    private String email;
    @Column(name="name");
    private String name;
    @OneToOne
    @JoinColumn(name="ACCOUNT_ID",referencedColumnName = "id")
    private Account account;
    public int getId() { return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
```

```
        this.name = name;
    }
    public String getEmail() {
        return email;
    }
    public void setEmail(String email) {
        this.email = email;
    }
    public Account getAccount() {
        return account;
    }
    public void setAccount(Account account) {
        this.account = account;
    }
}
```

EMPLOYEE ACCOUNT:

```
package Proj
import javax.persistence.*;
import Proj.Employee;
@Entity
@Table(name="account")
public class Account
{
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Column(name = "id")
    private int id;
    @Column(name="accno")
    private String accno;
    @OneToOne(mappedBy="account")
    private Employee employee;
```




```
public int getId() {
    return id;
}

public void setId(int id) {
    this.id = id;
}

public String getAccno()
{
    return accno;
}

public void setAccno(String accno)
{
    this.accno=accno;
}

public Employee getEmployee() {
    return employee;
}

public void setEmployee(Employee employee) {
    this.employee = employee;
}
}
```

CLIENT:

```
package Proj; import Proj.*;
import org.hibernate.*;

import org.hibernate.boot.Metadata;
import org.hibernate.boot.MetadataSources;
import org.hibernate.boot.registry.StandardServiceRegistry;
import org.hibernate.boot.registry.StandardServiceRegistryBuilder;

public class Clienttruncate {
    public static void main(String[] args) {
```



```
StandardServiceRegistryssr=new StandardServiceRegistryBuilder().
configure("hibernate.cfg.xml").build();

Metadata meta=new MetadataSources(ssr).getMetadataBuilder().build();

SessionFactory factory=meta.getSessionFactoryBuilder().build();

Session session=factory.openSession();

Transaction t=session.beginTransaction();

Account account = new Account();
account.setAccno("123-345-65495");

Employee emp = new Employee();
emp.setEmail("john@gmail.com");
emp.setName("userjohn");
session.saveOrUpdate(account);
emp.setAccount(account);
session.saveOrUpdate(emp);
t.commit();
session.close();

}

}
```

RESULT:

Thus the hibernate has been executed successfully.

8. SPRING RESTFUL APPLICATION WITH SPRING JPA

AIM:

To create a spring restful application with spring jpa and support with searching and pagination.

PROGRAM:

PaginationAndSortingApplication.java:

```
package com.lynas.paginationandsorting;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;
import lombok.RequiredArgsConstructor;
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.Bean;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.PageRequest;
import org.springframework.data.domain.Pageable;
import org.springframework.data.domain.Sort;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import java.util.Optional;

@SpringBootApplication
public class PaginationAndSortingApplication {
    public static void main(String[] args) {
        SpringApplication.run(PaginationAndSortingApplication.class, args);
    }
}
```



@Bean

```
public CommandLineRunnerinit(StudentRepository repository) {  
    return args -> {  
repository.save(new Student(null, "sazzad", "one", 1995));  
repository.save(new Student(null, "Rony", "one", 1993));  
repository.save(new Student(null, "Naim", "one", 1996));  
repository.save(new Student(null, "Dania", "one", 1995));  
repository.save(new Student(null, "Mamun", "one", 1993));  
repository.save(new Student(null, "Rimi", "one", 1995));  
repository.save(new Student(null, "Habib", "two", 1991));  
repository.save(new Student(null, "Shail", "two", 1999));  
repository.save(new Student(null, "Pranjol", "two", 1998));  
repository.save(new Student(null, "Shohag", "two", 1992));  
repository.save(new Student(null, "Ramjan", "two", 1993));  
repository.save(new Student(null, "Ashik", "two", 1996));  
repository.save(new Student(null, "Kibria", "two", 1995));  
repository.save(new Student(null, "Aurik", "three", 1997));  
repository.save(new Student(null, "Tanvir", "three", 1998));  
repository.save(new Student(null, "Nazmul", "three", 1995));  
repository.save(new Student(null, "Mizan", "three", 1996));  
repository.save(new Student(null, "Anik", "three", 1998));  
repository.save(new Student(null, "Mehedi", "three", 1997));  
repository.save(new Student(null, "Shahadat", "four", 1999));  
repository.save(new Student(null, "Dipak", "four", 1995));  
repository.save(new Student(null, "Mukta", "four", 1997));  
repository.save(new Student(null, "Rabaet", "four", 1997));  
repository.save(new Student(null, "Lopa", "four", 1995));  
repository.save(new Student(null, "Markes", "five", 1997));  
repository.save(new Student(null, "Valentino", "five", 1996));  
repository.save(new Student(null, "Fotik", "five", 1991));  
repository.save(new Student(null, "Lubna", "five", 1992));  
    };  
}
```

```

    }
}
@Data
@AllArgsConstructor
@NoArgsConstructor
@Entity
class Student {
    @Id
    @GeneratedValue
    private Long id;
    private String name;
    private String course;
    private int admissionYear;
}

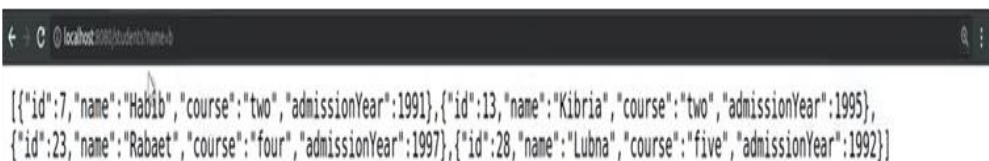
interface StudentRepository extends JpaRepository<Student, Long> {
    @Query("select s from Student s where name like %?1%")
    Page<Student>findByName(String name, Pageable pageable);
}

@RestController
@RequiredArgsConstructor
Class StudentController {
    private final StudentRepository repository;
    @GetMapping("/students")
    public Page<Student>findAll(
        @RequestParam Optional<String> name,
        @RequestParam Optional<Integer> page,
        @RequestParam Optional<String>sortBy) {
        // Sort by added
        return repository.findByName(name.orElse("_"),
            new PageRequest(
page.orElse(0), 5,
Sort.Direction.ASC, sortBy.orElse("id"))));

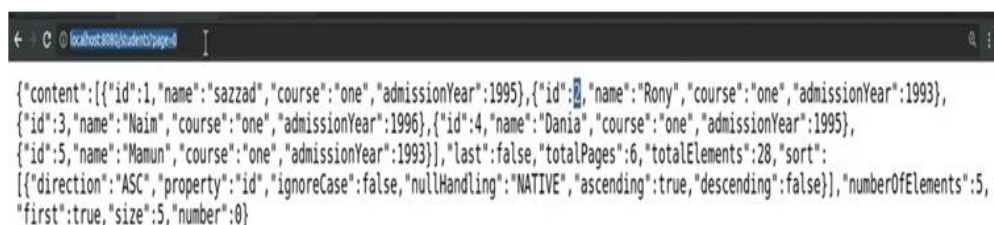
```

```
}  
}  
  
PaginationAndSortingApplicationTests.java:  
  
package com.lynas.paginationandsorting;  
  
import org.junit.Test;  
  
import org.junit.runner.RunWith;  
  
import org.springframework.boot.test.context.SpringBootTest;  
  
import org.springframework.test.context.junit4.SpringRunner;  
  
@RunWith(SpringRunner.class)  
  
@SpringBootTest  
  
public class PaginationAndSortingApplicationTests {  
  
    @Test  
  
    public void contextLoads() {  
  
    }  
  
}
```

OUTPUT:



```
localhost:8080/students/home  
[{ "id": 7, "name": "Habib", "course": "two", "admissionYear": 1991 }, { "id": 13, "name": "Kibria", "course": "two", "admissionYear": 1995 },  
{ "id": 23, "name": "Rabaet", "course": "four", "admissionYear": 1997 }, { "id": 28, "name": "Lubna", "course": "five", "admissionYear": 1992 } ]
```



```
localhost:8080/students/page  
{ "content": [ { "id": 1, "name": "sazzad", "course": "one", "admissionYear": 1995 }, { "id": 2, "name": "Rony", "course": "one", "admissionYear": 1993 },  
{ "id": 3, "name": "Naim", "course": "one", "admissionYear": 1996 }, { "id": 4, "name": "Dania", "course": "one", "admissionYear": 1995 },  
{ "id": 5, "name": "Mamun", "course": "one", "admissionYear": 1993 } ], "last": false, "totalPages": 6, "totalElements": 28, "sort":  
[ { "direction": "ASC", "property": "id", "ignoreCase": false, "nullHandling": "NATIVE", "ascending": true, "descending": false }, { "numberElements": 5,  
"first": true, "size": 5, "number": 0 } ]
```

RESULT:

Thus a spring restful application is built using spring jpa with the support of pagination and searching.



9. REACT APPLICATION

AIM :

To Create React Application with different components and interactions between components.

PROGRAM:

```
//creating exercise tracker app
```

```
//MODULES
```

Exercise.model.js

```
const mongoose = require('mongoose');  
//import mongoose from mongoose;  
const Schema = mongoose.Schema;  
const exerciseSchema = new Schema({  
  username : {type: String , required: true},  
  description: {type: String,required:true},  
  duration:{type: Number, required:true},  
  date:{type: Date,required:true},  
},  
{  
  timestamps: true,  
});  
const Exercise= mongoose.model('Exercise',exerciseSchema);  
module.exports = Exercise;
```

User.model.js

```
const mongoose = require('mongoose');  
//import mongoose from 'mongoose';  
const Schema = mongoose.Schema;  
const userSchema = new Schema({  
  username:{  
    type: String,  
    required : true,
```

```

        unique: true,
        trim: true,
        minlength: 3
    },
},
{
    timestamps: true,
});
const User = mongoose.model('User',userSchema);
module.exports=User;
//ROUTES
Exercise.js
const router = require('express').Router();
let Exercise = require('../models/exercise.model');
router.route('/').get((req,res)=>{
    Exercise.find()
        .then(exercises =>res.json(exercises))
        .catch(err =>res.status(400).json('Error:' + err));
    console.log('helo');
});
router.route('/add').post((req,res)=>{
    const username=req.body.username;
    const description=req.body.description;
    const duration =Number(req.body.duration);
    const date=Date.parse(req.body.date);
    const newExercise = new Exercise({
        username,
        description,
        duration,date,
    });
    newExercise.save()

```




```

        .then(() => res.json('Exercise added..!!!'))
        .catch(err => res.status(400).json('Error : ' + err));
    });

    router.route('/:id').get((req, res) => {
        Exercise.findById(req.params.id)
        .then(exercise => res.json(exercise))
        .catch(err => res.status(400).json('Error' + err));
    }
    );

    router.route('/:id').delete((req, res) => {
        Exercise.findByIdAndDelete(req.params.id)
        .then(() => res.json('Exercise deleted..'))
        .catch(err => res.status(400).json('Error' + err));
    });

    router.route('/update/:id').post((req, res) => {
        Exercise.findById(req.params.id)
        .then(exercise => {
            exercise.username = req.body.username;
            exercise.description = req.body.description;
            exercise.duration = req.body.duration;
            exercise.save()
            .then(() => res.json('Exercise updated'))
            .catch(err => res.status(400).json('Error:' + err));
        })
        .catch(err => res.status(400).json('Error' + err));
    });

    module.exports = router;

```

User.js

```

const router = require('express').Router();
let User = require('../../models/user.model');

```

```
router.route('/').get((req,res)=>{
  User.find()
  .then(users=>res.json(users))
  .catch(err =>res.status(400).json('Error:' + err));
});
router.route('/add').post((req,res)=>{
  const username = req.body.username;
  const newUser= new User({username});
  newUser.save()
  .then(()=>res.json('User added !'))
  .catch(err =>res.status(400).json('Error:' + err));
});
module.exports=router;
```

Server.js

```
const express = require('express');
const cors = require('cors');
const mongoose =require('mongoose');
require('dotenv').config();
const app=express();
const port=process.env.PORT || 5000;
app.use(cors());
app.use(express.json());
const uri = process.env.ATLAS_URI;
mongoose.connect(uri,{useNewUrlParser: true , useCreateIndex: true,
useUnifiedTopology: true});
const connection = mongoose.connection;
connection.once('open',()=>{
  console.log("connected to MongoDB");
})
const exerciseRouter = require('./routes/exercises');
```



```
const userRouter = require('./routes/users');
app.use('/exercises', exerciseRouter);
app.use('/users', userRouter);
app.listen(port, () => {
  console.log(`Server is listening at port : ${port}`);
});
```

App.js

```
import React from 'react';
import { BrowserRouter as Router, Route } from 'react-router-dom';
import 'bootstrap/dist/css/bootstrap.min.css';
import Navbar from './components/navbar.component';
import ExercisesList from './components/exercises-list.component';
import EditExercise from './components/edit-exercise.component';
import CreateExercise from './components/create-exercise.component';
import CreateUser from './components/create-user.component';
function App() {
  return (
    <Router>
      <div className="container">

        <Navbar />
        <br/>
        <Route path="/" exact component={ExercisesList} />
        <Route path="/edit/:id" component={EditExercise} />
        <Route path="/create" component={CreateExercise} />
        <Route path="/user" component={CreateUser} />
      </div>
    </Router>
  );
}
export default App;
```

Index.js

```
import React from 'react';
import ReactDOM from 'react-dom';
import App from './App';
import reportWebVitals from './reportWebVitals';

ReactDOM.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>,
  document.getElementById('root')
);
```

OUTPUT :

ExerciseTracker

[Exercises](#) [Create Exercise log](#) [Create User](#)

Logged Exercises

Username	Description	Duration	Date	Actions
priya	run	25	2021-07-01	edit delete
surya	jogging	30	2021-07-14	edit delete

ExerciseTracker

[Exercises](#) [Create Exercise log](#) [Create User](#)

Logged Exercises

Username	Description	Duration	Date	Actions
priya	run	25	2021-07-01	edit delete
surya	jogging	30	2021-07-14	edit delete



ExerciseTracker

Exercises Create Exercise log Create User

Create New Exercise

Username:

Priya

Description:

Jogging

Duration (in minutes):

30

Date:

09/17/2021

Create Exercise Log

RESULT:

To create react application with different components and interactions between components.

10. FULL-STACK APPLICATION USING REACT AND SPRING

AIM:

Create anfull-stack application using React and Spring.

PROGRAM:

User.java

```
package io.agileintelllligence.fullstack.domain;

import com.fasterxml.jackson.annotation.JsonFormat;
import com.fasterxml.jackson.annotation.JsonIgnore;
import lombok.Data;
import org.hibernate.annotations.CreationTimestamp;
import org.hibernate.annotations.UpdateTimestamp;
import org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.userdetails.UserDetails;
import javax.persistence.*;
import javax.validation.constraints.Email;
import javax.validation.constraints.NotBlank;
import java.util.ArrayList;
import java.util.Collection; import
java.util.Date; import java.util.List;

@Entity
@Data

public class User implements UserDetails {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    @Email(message = "Username must be an email")
    @NotBlank(message="Username is required")
    @Column(unique = true)
    private String username;
    @NotBlank(message="Please Enter your full name")
    private String fullName;
```

```
@NotBlank(message="Password field is required")
```

```
private String password;
```

```
@Transient
```

```
private String confirmPassword;
```

```
@CreationTimestamp
```

```
@JsonFormat(pattern="yyyy-mm-dd")
```

```
@Column(updatable =false)
```

```
private Date created_At;
```

```
@UpdateTimestamp
```

```
@JsonFormat(pattern="yyyy-mm-dd")
```

```
private Date updated_At;
```

```
@OneToMany(cascade=CascadeType.REFRESH,fetch=FetchType.
```

```
EAGER,mappedBy="user",orphanRemoval = true)
```

```
private List<Project> projects = new ArrayList<Project>();
```

```
public User() {
```

```
}
```

```
/*
```

UserDetails Interface Methods

```
*/
```

```
@Override
```

```
@JsonIgnore
```

```
public Collection<? extends GrantedAuthority>getAuthorities() {
```

```
    return null;
```

```
}
```

```
@Override
```

```
@JsonIgnore
```

```
public booleanisAccountNonExpired() {
```

```
    return true;
```

```

    }
    @Override
    @JsonIgnore
    public boolean isAccountNonLocked() {
        return true;
    }
    @Override
    @JsonIgnore
    public boolean isCredentialsNonExpired() {
        return true;
    }
    @Override
    @JsonIgnore
    public boolean isEnabled() {
        return true;
    }
}

```

Project Task.java

```

package io.agileintelllligence.fullstack.domain;
import com.fasterxml.jackson.annotation.JsonFormat;
import com.fasterxml.jackson.annotation.JsonIgnore;
import lombok.Data;
import org.hibernate.annotations.CreationTimestamp;
import org.hibernate.annotations.UpdateTimestamp;
import org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.userdetails.UserDetails;
import javax.persistence.*;
import javax.validation.constraints.Email;
import javax.validation.constraints.NotBlank;
import java.util.ArrayList; import java.util.Collection;
import java.util.Date;

```




```
import java.util.List;

@Entity
@Data

public class User implements UserDetails {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    @Email(message = "Username must be an email")
    @NotBlank(message="Username is required")
    @Column(unique = true)
    private String username;
    @NotBlank(message="Please Enter your full name")
    private String fullName;
    @NotBlank(message="Password field is required")
    private String password;
    @Transient
    private String confirmPassword;
    @CreationTimestamp
    @JsonFormat(pattern="yyyy-mm-dd")
    @Column(updatable =false)
    private Date created_At;
    @UpdateTimestamp
    @JsonFormat(pattern="yyyy-mm-dd")
    private Date updated_At;

    @OneToMany(cascade=CascadeType.REFRESH,fetch=      FetchType.
EAGER,mappedBy="user",orphanRemoval = true)
    private List<Project> projects = new ArrayList<Project>();
    public User() {
    }
    /*

UserDetails Interface Methods

*/
```

```
@Override
@JsonIgnore
public Collection<? extends GrantedAuthority>getAuthorities() {
    return null;
}
@Override
@JsonIgnore
public booleanisAccountNonExpired() {
    return true;
}
@Override
@JsonIgnore
public booleanisAccountNonLocked() {
    return true;
}
@Override
@JsonIgnore
public booleanisCredentialsNonExpired() {
    return true;
}
@Override
@JsonIgnore
public booleanisEnabled() {
    return true;
}
}
```

Project.java

```
package io.agileintelllignce.fullstack.domain;
import com.fasterxml.jackson.annotation.JsonFormat;
import com.fasterxml.jackson.annotation.JsonIgnore;
import lombok.Data;
import org.hibernate.annotations.CreationTimestamp;
```



```
import org.hibernate.annotations.UpdateTimestamp;
import javax.persistence.*;
import javax.validation.constraints.NotBlank;
import javax.validation.constraints.Size;
import java.util.Date;

@Entity
@Data
public class Project {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    @NotBlank(message="Project Name Required")
    private String projectName;
    @NotBlank(message="Project Identifier Required")
    @Size(min=4,max=5,message="Please USe 4-5 Characters")
    @Column(updatable = false,unique = true)
    private String projectIdentifier;
    @NotBlank(message="Project Description Required")
    private String description;
    @JsonFormat(pattern="yyyy-mm-dd")
    private Date startDate;
    @JsonFormat(pattern="yyyy-mm-dd")
    private Date endDate;
    @CreationTimestamp
    @JsonFormat(pattern="yyyy-mm-dd")
    @Column(updatable =false)
    private Date created_At;
    @UpdateTimestamp
    @JsonFormat(pattern="yyyy-mm-dd")
    private Date updated_At;
    @OneToOne(fetch=FetchType.EAGER,cascade=CascadeType.
ALL,mappedBy="project")
```



```
@JsonIgnore
```

```
private Backlog backlog;
```

```
@ManyToOne(fetch=FetchType.LAZY)
```

```
@JsonIgnore
```

```
private User user;
```

```
private String projectLeader; public Project() {
```

```
}
```

```
}
```

Backlog.java

```
package io.agileintelllligence.fullstack.domain; import
```

```
com.fasterxml.jackson.annotation.JsonIgnore; import lombok.Data;
```

```
import javax.persistence.*; import
```

```
java.util.ArrayList; import java.util.List;
```

```
@Entity
```

```
@Data
```

```
public class Backlog {
```

```
    @Id
```

```
    @GeneratedValue(strategy = GenerationType.IDENTITY)
```

```
private Long id;
```

```
private Integer PTSequence = 0;
```

```
private String projectIdentifier;
```

```
@Column(insertable = false,updatable=false)
```

```
private int project_id;
```

```
public Backlog() {
```

```
}
```

```
//OnetoOne with Project
```

```
@OneToOne(fetch=FetchType.EAGER)
```

```
@JoinColumn(name = "project_id",nullable=false)
```

```
@JsonIgnore
```

```
private Project project;
```

```
//OnetoMany with ProjectTask
@OneToMany(cascade=CascadeType.REFRESH,fetch=FetchType.
EAGER,mappedBy="backlog",orphanRemoval = true)
private List<ProjectTask>projectTasks = new ArrayList<>();
}
```

CONTROLLER

BacklogController.java

```
package io.agileintelligence.fullstack.web;
import io.agileintelligence.fullstack.domain.ProjectTask;
import io.agileintelligence.fullstack.services.MapValidationErrorMessage;
import io.agileintelligence.fullstack.services.ProjectTaskService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.*;
import javax.validation.Valid;
import java.security.Principal;

@RestController
@RequestMapping("/api/backlog")
@CrossOrigin
public class BacklogController {
    @Autowired
    private ProjectTaskService projectTaskService;
    @Autowired
    private MapValidationErrorMessage mapValidationErrorMessage;
    @PostMapping("/{backlog_id}")
    public ResponseEntity<?> addPTtoBacklog(@Valid @RequestBody
    ProjectTask projectTask,
    BindingResult result, @PathVariable String backlog_id, Principal principal) {
    ResponseEntity<?> errorMap = mapValidationErrorMessage.
```

```
MapValidationErrorResponse(result);
    if (errorMap != null) return errorMap;
    ProjectTask projectTask1 = projectTaskService.addProjectTask(backlog_id,
projectTask,principal.getName());
    return new ResponseEntity<ProjectTask>(projectTask1, HttpStatus.
CREATED);
}
@GetMapping("/{backlog_id}")
    public Iterable<ProjectTask>getProjectBacklogs(@PathVariable String
backlog_id,Principal principal) {
        return projectTaskService.findByBacklogId(backlog_id,principal.
getName());
    }
@GetMapping("/{backlog_id}/{pt_id}")
    public ResponseEntity<?>getProjectBySequence(@PathVariable String
backlog_id, @PathVariable String pt_id,Principal principal) {
ProjectTaskprojectTask1=projectTaskService.findByProjectSequence(backlog_
id, pt_id,principal.getName());

    return new ResponseEntity<ProjectTask>(projectTask1, HttpStatus.OK);
}
@PatchMapping("/{backlog_id}/{pt_id}")
    public ResponseEntity<?>updateProjectTask(@Valid @RequestBody
ProjectTaskprojectTask, BindingResult result,
        @PathVariable String backlog_id, @PathVariable
String pt_id,Principal principal){
ResponseEntity<?>errorMap = mapValidationErrorResponse.
MapValidationErrorResponse(result);
    if (errorMap != null) return errorMap;
ProjectTaskupdatedTask = projectTaskService.
updateProject(projectTask,backlog_id,pt_id,principal.getName());
    return new ResponseEntity<ProjectTask>(updatedTask,HttpStatus.OK);
```

```
}  
@DeleteMapping("/{backlog_id}/{pt_id}")  
public ResponseEntity<?>deleteProject(@PathVariable String backlog_id, @  
PathVariable String pt_id,Principal principal) {  
    projectTaskService.deleteProject(backlog_id,pt_id,principal.getName());  
    return new ResponseEntity<String>("Project with id:"+pt_id+" in the  
"+backlog_id+" has been deleted successfully",HttpStatus.OK);  
}  
}
```

ProjectController.java

```
package io.agileintelligence.fullstack.web;  
import io.agileintelligence.fullstack.domain.Project;  
import io.agileintelligence.fullstack.services.MapValidationErrorResponse;  
import io.agileintelligence.fullstack.services.ProjectService;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.http.HttpStatus;  
import org.springframework.http.ResponseEntity;  
import org.springframework.validation.BindingResult;  
import org.springframework.validation.FieldError;  
import org.springframework.web.bind.annotation.*;  
import javax.validation.Valid;  
import java.security.Principal;  
import java.util.HashMap; import java.util.List;  
import java.util.Map;  
@RestController  
@RequestMapping("/api/project")  
@CrossOrigin  
public class ProjectController {  
    @Autowired  
    private ProjectServiceprojectService;  
    @Autowired  
    private MapValidationErrorResponsemapValidationErrorResponse;  
    @PostMapping("")
```

```
public ResponseEntity<?>createNewProject(@Valid @RequestBody Project
project, BindingResult result, Principal principal){

    ResponseEntity<?>errorMap          =          mapValidationErrorMessageService.
    MapValidationErrorMessageService(result);
        if(errorMap!=null) return  errorMap;
        Project project1 = projectService.saveOrUpdateProject(project, principal.
getName());
        return new ResponseEntity<Project>(project1, HttpStatus.CREATED);
    }
    @GetMapping("/{projectId}")
        public ResponseEntity<?>getProjectById(@PathVariable String projectId,
Principal principal){
        Project project = projectService.findProjectByIdentifier(projectId, principal.
getName());
        return new ResponseEntity<Project>(project, HttpStatus.OK);
    }
    @GetMapping("/all")
        public  Iterable<Project>getAllProjects(Principal  principal){return
projectService.findAllProjects(principal.getName());}

    @DeleteMapping("/{projectId}")

        public ResponseEntity<?>deleteProject(@PathVariable String projectId,
Principal principal){
projectService.deleteProjectByIdentifier(projectId, principal.getName());

        return new ResponseEntity<String>("Project with ID: '"+projectId+"' was
deleted", HttpStatus.OK);

    }

}
UserController.java
package io.agileintelllligence.fullstack.web;
import io.agileintelllligence.fullstack.domain.User;
```



```
import io.agileintelllligence.fullstack.payload.JWTLoginSucessResponse;
import io.agileintelllligence.fullstack.payload.LoginRequest;
import io.agileintelllligence.fullstack.security.JwtTokenProvider;
import io.agileintelllligence.fullstack.services.MapValidationErrorResponse;
import io.agileintelllligence.fullstack.services.UserService;
import io.agileintelllligence.fullstack.validator.UserValidator;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.security.authentication.AuthenticationManager;
import org.springframework.security.authentication.
UsernamePasswordAuthenticationToken;
import org.springframework.security.core.Authentication;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.*;
import javax.validation.Valid;
import static io.agileintelllligence.fullstack.security.SecurityConstants.TOKEN_
PREFIX;

@RestController

@RequestMapping("/api/users")

@CrossOrigin

public class UserController {

    @Autowired
    private MapValidationErrorResponsemapValidationErrorResponse;

    @Autowired
    private UserServiceuserService;

    @Autowired
    private UserValidatoruserValidator;

    @Autowired
    private JwtTokenProvidertokenProvider;

    @Autowired
```



```
private AuthenticationManager authenticationManager;

@PostMapping("/login")
public ResponseEntity<?> authenticateUser(@Valid @RequestBody
LoginRequest loginRequest, BindingResult result)
{
    ResponseEntity<?> errorMap = mapValidationErrorResponse.
    MapValidationErrorResponse(result);
    if(errorMap != null)
        return errorMap;
    Authentication authentication = authenticationManager.authenticate(
        new UsernamePasswordAuthenticationToken(
loginRequest.getUsername(),
loginRequest.getPassword()
        )
    );
    SecurityContextHolder.getContext().setAuthentication(authentication);
    String jwt = TOKEN_PREFIX + tokenProvider.generateToken(authentication);
    return ResponseEntity.ok(new JWTLoginSuccessResponse(true, jwt));
}

@PostMapping("/register")
public ResponseEntity<?> registerUser(@Valid @RequestBody User user,
BindingResult result)
{
    //Validate Passwords Match
    userValidator.validate(user, result);
    ResponseEntity<?> errorMap = mapValidationErrorResponse.
    MapValidationErrorResponse(result);
    if (errorMap != null )
        return errorMap;
    User newUser = userService.saveUser(user);
    return new ResponseEntity<User>(newUser, HttpStatus.CREATED);
}
```

```
}
```

SERVICE CLASS

UserService.java

```
package io.agileintelligence.fullstack.services;
import io.agileintelligence.fullstack.domain.User;
import io.agileintelligence.fullstack.exceptions.
UsernameAlreadyExistsException;
import io.agileintelligence.fullstack.repositories.UserRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.crypto.bcrypt.BCrypt;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.stereotype.Service;
@Service
public class UserService {
    @Autowired
    private UserRepositoryuserRepository;
    @Autowired
    private BCryptPasswordEncoderbCryptPasswordEncoder;
    public User saveUser(User newUser)
    {
        try {
            newUser.setPassword(bCryptPasswordEncoder.encode(newUser.
getPassword()));
            //Username must be unique
            newUser.setUsername(newUser.getUsername());
            //Make Sure Password and ConfirmPassword Match
            //Dont Persist ConfirmPass
            newUser.setConfirmPassword("");
            return userRepository.save(newUser);
        } catch (Exception e)
        {
            throw new UsernameAlreadyExistsException("Username:" + newUser.
```

```

        getUsername() + “ already exists”);
    }
}

```

ProjectTaskService.java

```

package io.agileintelligence.fullstack.services;
import io.agileintelligence.fullstack.domain.Backlog;
import io.agileintelligence.fullstack.domain.Project;
import io.agileintelligence.fullstack.domain.ProjectTask;
import io.agileintelligence.fullstack.exceptions.ProjectNotFoundException;
import io.agileintelligence.fullstack.repositories.BacklogRepository;
import io.agileintelligence.fullstack.repositories.ProjectRepository;
import io.agileintelligence.fullstack.repositories.ProjectTaskRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
@Service
public class ProjectTaskService {
    @Autowired
    private BacklogRepository backlogRepository;
    @Autowired
    private ProjectTaskRepository projectTaskRepository;
    @Autowired
    private ProjectRepository projectRepository;
    @Autowired
    private ProjectService projectService;

    public ProjectTask addProjectTask(String projectIdIdentifier,
    ProjectTask projectTask, String username) {
        //Exceptions: Project not found

        //PTs to be added to a specific project, project != null, BL exists

        Backlog backlog = projectService.
findProjectByIdentifier(projectIdentifier, username).getBacklog();

        //backlogRepository.findById(projectIdentifier);

```

```
//set the bl to pt

projectTask.setBacklog(backlog);
//we want our project sequence to be like this: IDPRO-1 IDPRO-2 ...100
101
    Integer BacklogSequence = backlog.getPTSequence();
    // Update the BL SEQUENCE
    BacklogSequence++;
    backlog.setPTSequence(BacklogSequence);
    //Add Sequence to Project Task
    projectTask.setProjectSequence(projectIdentifier + "-" + BacklogSequence);
    projectTask.setProjectIdentifier(projectIdentifier);
    //INITIAL status when status is null
    if (projectTask.getStatus() == "" || projectTask.getStatus() == null) {
    projectTask.setStatus("TO_DO");
    }
    //INITIAL priority when priority null
    if (projectTask.getPriority() == null || projectTask.getPriority() == 0) {
    projectTask.setPriority(3);
    }
    return projectTaskRepository.save(projectTask);
}

public Iterable<ProjectTask>findByBacklogId(String backlog_id,String
username) {
    projectService.findProjectByIdentifier(backlog_id,username);
    return projectTaskRepository.findByProjectIdentifierOrderByPriority(backlog_
id);
}

public ProjectTaskfindByProjectSequence(String backlog_id, String
sequence,String username) {
    projectService.findProjectByIdentifier(backlog_id,username);
    ProjectTaskprojectTask = projectTaskRepository.
findByProjectSequence(sequence);
```



```
        if (projectTask == null) {
            throw new ProjectNotFoundException("Project Task id:" + sequence +
" does not exist");
        }
        if (!projectTask.getProjectIdentifier().equals(backlog_id)) {
            throw new ProjectNotFoundException("Project Task id:" + sequence + "
does not exist in the project" + backlog_id);
        }
        return projectTask;
    }

    public ProjectTaskupdateProject(ProjectTaskupdatedProject, String backlog_
id, String sequence,String username) {
ProjectTaskprojectTask1                =                findByProjectSequence(backlog_id,
sequence,username);

        projectTask1 = updatedProject;

        return projectTaskRepository.save(projectTask1);
    }
    public voiddeleteProject(String backlog_id, String sequence,String username)
    {
ProjectTask        projectTask1        =        findByProjectSequence(backlog_id,
sequence,username);
projectTaskRepository.delete(projectTask1);
    }
}
```

ProjectService.java

```
package io.agileintelllligence.fullstack.services;
import io.agileintelllligence.fullstack.domain.Backlog;
import io.agileintelllligence.fullstack.domain.Project;
import io.agileintelllligence.fullstack.domain.User;
import io.agileintelllligence.fullstack.exceptions.ProjectIdException;
import io.agileintelllligence.fullstack.repositories.BacklogRepository;
```

```
import io.agileintelllligence.fullstack.repositories.ProjectRepository;
import io.agileintelllligence.fullstack.repositories.UserRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import io.agileintelllligence.fullstack.exceptions.ProjectNotFoundException;
@Service
public class ProjectService {
    @Autowired
    private ProjectRepository projectRepository;
    @Autowired
    private BacklogRepository backlogRepository;
    @Autowired
    private UserRepository userRepository;
    public Project saveOrUpdateProject(Project project, String username){
        if(project.getId() != null){
            Project existingProject = projectRepository.findByProjectIdentifier(project.
getProjectIdentifier());
            if(existingProject != null && (!existingProject.getProjectLeader().
equals(username))){
                throw new ProjectNotFoundException("Project not found in your
account");
            }else if(existingProject == null){
                throw new ProjectNotFoundException("Project with ID: '"+project.
getProjectIdentifier()+"' cannot be updated because it doesn't exist");
            }
        }
    }
    try{
        User user = userRepository.findByUsername(username);
        project.setUser(user);
        project.setProjectLeader(user.getUsername());
    }
```

```
project.setProjectIdentifier(project.getProjectIdentifier().toUpperCase());

    if(project.getId()==null){

        Backlog backlog = new Backlog();
project.setBacklog(backlog);

backlog.setProject(project);
backlog.setProjectIdentifier(project.getProjectIdentifier().toUpperCase());

    }

    if(project.getId()!=null){

project.setBacklog(backlogRepository.findByProjectIdentifier(project.
getProjectIdentifier().toUpperCase()));

    }

    return projectRepository.save(project);

} catch (Exception e){

    throw new ProjectIdException("Project ID '"+project.getProjectIdentifier().
toUpperCase()+"' already exists");

}

}

public Project findProjectByIdentifier(String projectId, String username){
    //Only want to return the project if the user looking for it is the owner
    Project project = projectRepository.findByProjectIdentifier(projectId.
toUpperCase());
    if(project == null){

        throw new ProjectIdException("Project ID '"+projectId+"' does not
exist");

    }

    if(!project.getProjectLeader().equals(username)){

        throw new ProjectNotFoundException("Project not found in your
account");

    }

}
```




```

    }

    return project;

}

public Iterable<Project>findAllProjects(String username){
    return
    projectRepository.findAllByProjectLeader(username);
}

public void deleteProjectByIdentifier(String projectid, String username){
projectRepository.delete(findProjectByIdentifier(projectid, username));
}

}

MapErrorValidationService
package io.agileintelligence.fullstack.services;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.stereotype.Service;
import org.springframework.validation.BindingResult;
import org.springframework.validation.FieldError;
import java.util.HashMap;
import java.util.Map;
@Service
public class MapValidationErrorResponseService {
public ResponseEntity<?>MapValidationErrorResponse(BindingResult result) {
if (result.hasErrors()) {
    Map<String, String>mp = new HashMap<String,
    String>(); for (FieldError error :result.getFieldErrors()) {
mp.put(error.getField(), error.getDefaultMessage());
    }

    return new ResponseEntity<Map<String, String>>(mp, HttpStatus.
BAD_REQUEST);

```

```

    }
    return null;
}
}

```

SECURITY

SecurityConfig.java

```

package io.agileintelligence.fullstack.security;

import io.agileintelligence.fullstack.services.CustomUserDetailsService;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.authentication.AuthenticationManager;
import org.springframework.security.authentication.
UsernamePasswordAuthenticationToken;

import org.springframework.security.config.BeanIds;

import org.springframework.security.config.annotation.authentication.builders.
AuthenticationManagerBuilder;

import org.springframework.security.config.annotation.method.configuration.
EnableGlobalMethodSecurity;

import org.springframework.security.config.annotation.web.builders.
HttpSecurity;

import org.springframework.security.config.annotation.web.configuration.
EnableWebSecurity;

import org.springframework.security.config.annotation.web.configuration.
WebSecurityConfigurerAdapter;

import org.springframework.security.config.http.SessionCreationPolicy;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.web.authentication.
UsernamePasswordAuthenticationFilter;

import static io.agileintelligence.fullstack.security.SecurityConstants.H2_URL;

```

```
import static io.agileintelligence.fullstack.security.SecurityConstants.SIGN_
UP_URLS;
@Configuration
@EnableWebSecurity
@EnableGlobalMethodSecurity( securedEnabled = true,
    jsr250Enabled = true, prePostEnabled = true
)

public class SecurityConfig extends WebSecurityConfigurerAdapter {

    @Autowired

    private JwtAuthenticationEntryPointunauthorizedHandler;

    @Autowired
    private CustomUserDetailsServicecustomUserDetailsService;
    @Autowired
    private BCryptPasswordEncoderbCryptPasswordEncoder;
    @Bean
    public JwtAuthenticationFilterjwtAuthenticationFilter(){
        return new JwtAuthenticationFilter();
    }
    @Override
    protected void
    configure(AuthenticationManagerBuilderauthenticationManagerBuilder)
    throws Exception {

        authenticationManagerBuilder.userDetailsService(customUserDetailsService).
        passwordEncoder(bCryptPasswordEncoder);

    }
    @Override
    @Bean(Beans.AUTHENTICATION_MANAGER)
    protected AuthenticationManagerauthenticationManager() throws Exception
    {
        return super.authenticationManager();
    }
}
```

```
@Override
protected void configure(HttpSecurity http) throws Exception {
    http.cors().and().csrf().disable().
        exceptionHandling().authenticationEntryPoint(unauthorizedHandler).and()
        .sessionManagement()
        .sessionCreationPolicy(SessionCreationPolicy.STATELESS)
        .and()
        .headers().frameOptions().sameOrigin()
        .and()
        .authorizeRequests()
        .antMatchers("/",
            "/favicon.ico", "/**/*.png",
            "/**/*.gif",
            "/**/*.svg",
            "/**/*.jpg",
            "/**/*.html",
            "/**/*.css", "/**/*.js"
        ).permitAll()
        .antMatchers(SIGN_UP_URLS).permitAll()
        .antMatchers(H2_URL).permitAll()
        .anyRequest().authenticated();
    http.addFilterBefore(jwtAuthenticationFilter(),
        UsernamePasswordAuthenticationFilter.class);
}
}
```

JwtTokenProvider.java

```
package io.agileintelllligence.fullstack.security;
import io.agileintelllligence.fullstack.domain.User;
```



```
import io.jsonwebtoken.*;
import org.springframework.security.core.Authentication;
import org.springframework.stereotype.Component;
import java.util.Date;
import java.util.HashMap;
import java.util.Map;
import static io.agileintelligence.fullstack.security.SecurityConstants.
EXPIRATION_TIME;
import static io.agileintelligence.fullstack.security.SecurityConstants.SECRET;
@Component
public class JwtTokenProvider {
    public String generateToken(Authentication authentication) {
        User user = (User) authentication.getPrincipal();
        Date now = new Date(System.currentTimeMillis());
        Date expiryDate = new Date(now.getTime() + EXPIRATION_TIME);
        String userId = Long.toString(user.getId());
        Map<String, Object> claims = new HashMap<String, Object>();
        claims.put("id", (Long.toString(user.getId())));
        claims.put("username", user.getUsername());
        claims.put("fullName", user.getFullName());
        return Jwts.builder()
            .setSubject(userId)
            .setClaims(claims)
            .setIssuedAt(expiryDate)
            .signWith(SignatureAlgorithm.HS512, SECRET)
            .compact();
    }

    public boolean validateToken(String token) { try {
        Jwts.parser().setSigningKey(SECRET).parseClaimsJws(token);
        return true;
    } catch (SignatureException ex) {
        System.out.println("Invalid Jwt Signature");
    }
}
```

```
        } catch (MalformedJwtException ex) {
System.out.println("Invalid Jwt Token");
        } catch (ExpiredJwtException ex) {
System.out.println("Expired Jwt Token");
        } catch (UnsupportedJwtException ex) {
System.out.println("Unsupported Jwt Token");
        } catch (IllegalArgumentException ex) {
System.out.println("Jwt claims string is empty");
        }
        return false;
    }

    public Long getUserIdFromJWT(String token) {
        Claimsclaims=Jwts.parser().setSigningKey(SECRET).parseClaimsJws(token).
getBody();
        String id = (String)claims.get("id");
        return Long.parseLong(id);
    }
}
```

JwtAuthenticationFilter.java

```
package io.agileintelligence.fullstack.security;
import io.agileintelligence.fullstack.domain.User;
import io.agileintelligence.fullstack.services.CustomUserDetailsService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.authentication.
UsernamePasswordAuthenticationToken;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.security.web.authentication.
WebAuthenticationDetailsSource;
import org.springframework.util.StringUtils;
import org.springframework.web.filter.OncePerRequestFilter;
import javax.servlet.FilterChain;
import javax.servlet.ServletException;
```

```
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.util.Collections;
import java.util.logging.Logger;
import static io.agileintelllligence.fullstack.security.SecurityConstants.

HEADER_STRING;

import static io.agileintelllligence.fullstack.security.SecurityConstants.TOKEN_
PREFIX;

public class JwtAuthenticationFilter extends OncePerRequestFilter {

    @Autowired

    private JwtTokenProviderjwtTokenProvider;

    @Autowired

    private CustomUserDetailsServicecustomUserDetailsService;

    @Override

    protected void doFilterInternal(HttpServletRequesthttpServletrequest,
    HttpServletResponsehttpServletresponse, FilterChainfilterChain) throws
    ServletException, IOException {

        try {
            String jwt = getJWTFromRequest(httpServletrequest);
            if(StringUtils.hasText(jwt) &&jwtTokenProvider.validateToken(jwt) ) {

                Long userId = jwtTokenProvider.getUserIdFromJWT(jwt);

                User userDetails = customUserDetailsService.loadUserById(userId);

                UsernamePasswordAuthenticationToken authentication =new
                UsernamePasswordAuthenticationToken(
                userDetails,null,(Collections.emptyList()));
                authentication.setDetails(new WebAuthenticationDetailsSource().
                buildDetails(httpServletrequest));

                SecurityContextHolder.getContext().setAuthentication(authentication);

            }

        } catch (Exception ex) {

            logger.error("Could not Set user authentication in security context",ex);

        }
    }
}
```

```
filterChain.doFilter(httpServletRequest,httpServletResponse);
    }
    private String getJWTFromRequest(HttpServletRequest request)
    {
        private String getJWTFromRequest(HttpServletRequest request)
        {
            return bearerToken.substring(7,bearerToken.length());
        }
        return null;
    }
}
```

FRONTEND SERVICE

App.js

```
import React, { Component } from "react";
import "./App.css";
import Dashboard from "./components/Dashboard";
import "bootstrap/dist/css/bootstrap.min.css";
import Header from "./components/Layout/Header";
import { BrowserRouter as Router, Route, Switch } from "react-router-dom";
import AddProject from "./components/Project/AddProject";
import { Provider } from "react-redux";
import store from "./store";
import UpdateProject from "./components/Project/UpdateProject";
import ProjectBoard from "./components/ProjectBoard/ProjectBoard";
import AddProjectTask from "./components/ProjectBoard/ProjectTasks/
AddProjectTask";
import UpdateProjectTask from "./components/ProjectBoard/ProjectTasks/
UpdateProjectTask";
import Landing from "./components/Layout/Landing";
import Register from "./components/UserManagement/Register";
import Login from "./components/UserManagement/Login";
import jwt_decode from "jwt-decode";
import setJWTToken from "./securityUtils/setJWTToken";
```



```
import { SET_CURRENT_USER } from “./actions/type”;
import { logout } from “./actions/securityActions”;
import SecuredRoute from “./securityUtils/secureRoute”;
const jwtToken = localStorage.jwtToken;
if (jwtToken) {
  setJWTToken(jwtToken);

  const decoded_jwtToken = jwt_decode(jwtToken);

  store.dispatch({
    type: SET_CURRENT_USER,
    payload: decoded_jwtToken,
  });

  const currentTime = Date.now() / 1000;

  if (decoded_jwtToken.exp < currentTime) {
store.dispatch(logout());
window.location.href = “/”;
  }
}

class App extends Component {
  render() {
    return (
      <Provider store={store}>
        <Router>
          <div className=“App”>
            <Header></Header>
            {
              //PUBLIC ROUTES
            }
            <Route exact path=“/” component={Landing} />
            <Route exact path=“/register” component={Register} />
            <Route exact path=“/login” component={Login} />
```



```
{  
  //PRIVATE ROUTES  
}  
  
<Switch>  
<SecuredRoute exact path="/dashboard" component={Dashboard} />  
<SecuredRoute exact path="/addProject" component={AddProject} />  
<SecuredRoute  
  exact  
  path="/updateProject/:id"  
  component={UpdateProject}  
/>  
<SecuredRoute  
  exact  
  path="/projectBoard/:id"  
  component={ProjectBoard}  
/>  
<SecuredRoute  
  exact  
  path="/addProjectTask/:id"  
  component={AddProjectTask}  
/>  
<SecuredRoute  
  exact  
  path="/addProjectTask/projectBoard/:id"  
  component={ProjectBoard}  
/>  
<SecuredRoute  
  exact  
  path="/updateProjectTask/:backlog_id/:pt_id"  
  component={UpdateProjectTask}  
/>  
</Switch>  
</div>
```

```
</Router>
</Provider>
  );
}
}
export default App;
Index.js
import React from 'react';
import ReactDOM from 'react-dom'; import './index.css';
import App from './App';
import reportWebVitals from './reportWebVitals';
ReactDOM.render(
  <React.StrictMode>
  <App />
  </React.StrictMode>,
  document.getElementById('root')
);

// If you want to start measuring performance in your app, pass a function
// to log results (for example: reportWebVitals(console.log))
// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals
reportWebVitals();
Header.js
import React, { Component } from "react";
import { Link } from "react-router-dom";
import PropTypes from "prop-types";
import { connect } from "react-redux";
import { logout } from "../../actions/securityActions";
class Header extends Component {
  logout() {
    this.props.logout();
    window.location.href = "/";
  }
}
```

```
render() {

  const { validToken, user } = this.props.security;

  const userIsAuthenticated = (

    <div className="collapse navbar-collapse" id="mobile-nav">
      <ul className="navbar-nav mr-auto">
        <li className="nav-item">
          <Link className="nav-link" to="/dashboard">
            Dashboard
          </Link>
        </li>
      </ul>
      <ul className="navbar-nav ml-auto">
        <li className="nav-item">
          <Link className="nav-link" to="/dashboard">
            <i className="fas fa-user-circle mr-1" />
              {user.fullName}
            </Link>
          </li>
          <li
            className="nav-item">
            <Link className="nav-link"
              to="/logout"
              onClick={this.logout.bind(this)}
            >
              Logout
            </Link>
          </li>
        </ul>
      </div>
    );
```

```

const userIsNotAuthenticated = (
  <div className="collapse navbar-collapse" id="mobile-nav">
    <ul className="navbar-nav ml-auto">
      <li className="nav-item">
        <Link className="nav-link" to="/register">
          Sign Up
        </Link>
      </li>
      <li className="nav-item">
        <Link className="nav-link" to="/login">
          Login
        </Link>
      </li>
    </ul>
  </div>

  );

let headerLinks;

if (validToken&& user) {
  headerLinks =
    userIsAuthenticated;
  } else {
    headerLinks = userIsNotAuthenticated;
  }
  return (
    <nav className="navbar navbar-expand-sm navbar-dark bg-primary mb-4">
      <div className="container">
        <Link className="navbar-brand" to="/">
          Amar's Project Management Tool
        </Link>
        <button

```

```

        className="navbar-toggler"
        type="button"
        data-toggle="collapse"
        data-target="#mobile-
        nav"
    >
    <span className="navbar-toggler-icon" />
</button>
        {headerLinks}
    </div>
</nav>
    );
}
}
Header.propTypes = {
    logout: PropTypes.func.isRequired,
    security: PropTypes.object.isRequired,
};
const mapStateToProps = (state) => ({
    security: state.security,
});
export default connect(mapStateToProps, { logout })(Header);

```

Landing.js

```

import React, { Component } from "react";
import { Link } from "react-router-dom";
import { connect } from "react-redux";
import PropTypes from "prop-types";
class Landing extends Component {
    componentDidMount() {
        if (this.props.security.validToken) {
            this.props.history.push("/dashboard");
        }
    }
}

```

```

}

render() {
  return (
    <div className="landing">

    <div className="light-overlay landing-inner text-dark">

    <div className="container">

    <div className="row">

    <div className="col-md-12 text-center">
    <h1 className="display-3 mb-4">
      Amar's Project Management Tool
    </h1>
    <p className="lead">
      Create your account to join active projects or start you own
    </p>
    <hr />
    <Link className="btn btn-lg btn-primary mr-2" to="/register">
      Sign Up
    </Link>
    <Link className="btn btn-lg btn-primary mr-2" to="/login">
      Login
    </Link>
    </div>
    </div>
    </div>
    </div>
    </div>
    );
  }
}

Landing.propTypes = {
  security: PropTypes.object.isRequired,

```

```

};
const mapStateToProps = (state) => ({
  security: state.security,
});
export default connect(mapStateToProps)(Landing);
AddProject.js
import React, { Component } from "react";

import PropTypes from "prop-types";
import { connect } from "react-redux";
import { createProject } from "../../actions/projectAction.js";
import classNames from "classnames";

class AddProject extends Component { constructor() {
  super();
  this.state = {
    projectName: "",
    projectIdentifier: "",
    description: "",
    startDate: "",
    endDate: "",
    errors: "",
  };
  this.onChange = this.onChange.bind(this);
  this.onSubmit = this.onSubmit.bind(this);
}

componentWillReceiveProps(nextProps) {
  if (nextProps.errors) {
    this.setState({ errors: nextProps.errors });
  }
}

onChange(e) {
  this.setState({ [e.target.name]: e.target.value });
}

```




```
onSubmit(e) {
  e.preventDefault();
  const newProject = {
    projectName: this.state.projectName,
    projectIdentifier: this.state.projectIdentifier,
    description: this.state.description,
    startDate: this.state.startDate,
    endDate: this.state.endDate,
  };
  this.props.createProject(newProject, this.props.history);
}

render() {
  const { errors } = this.state;

  return (
    <div>
      <div className="project">
        <div className="container">
          <div className="row">
            <div className="col-md-8 m-auto">
              <h5 className="display-4 text-center">Create Project form</h5>
              <hr />
              <form onSubmit={this.onSubmit}>
                <div className="form-group">
                  <input
                    type="text"
                    className={classnames("form-control form-control-lg", {
                      "is-invalid": errors.projectName,
                    })}
                    placeholder="Project Name"
                    name="projectName"

```



```

        value={this.state.projectName}
        onChange={this.onChange.bind(this)}
      />
      {errors.projectName&& (
<div className="invalid-feedback">
        {errors.projectName}
</div>
      )}
</div>

<div className="form-group">
  <input
    type="text"
    className={classnames("form-control form-control-lg", {
      "is-invalid": errors.projectIdentifier,
    })}
    placeholder="Unique Project ID"
    name="projectIdentifier"
    value={this.state.projectIdentifier}
    onChange={this.onChange.bind(this)}
  />
  {errors.projectIdentifier&& (
<div className="invalid-feedback">
    {errors.projectIdentifier}
</div>
  )}
</div>

<div className="form-group">
  <textarea
    className={classnames("form-control form-control-lg", {
      "is-invalid": errors.description,

```

```

    }}}
    placeholder="Project Description"
    name="description" value={this.state.description}
onChange={this.onChange.bind(this)}
</textarea>
    {errors.description}&& (
<div className="invalid-feedback">
    {errors.description}
</div>
    )}
</div>
<h6>Start Date</h6>
<div className="form-group">
<input
    type="date"
    className="form-control form-control-lg"
    name="startDate"
    value={this.state.startDate}
    onChange={this.onChange.bind(this)}
/>
</div>
<h6>Estimated End Date</h6>
<div className="form-group">
<input
    type="date"
    className="form-control form-control-lg"
    name="endDate"
    value={this.state.endDate}
    onChange={this.onChange.bind(this)}
/>
</div>
<input

```

```

        type="submit"
        className="btn btn-primary btn-block mt-4"
      />
    </form>
  </div>
</div>
</div>
</div>
</div>
  );
}
}
AddProject.propTypes = {
  createProject: PropTypes.func.isRequired,
  errors: PropTypes.object.isRequired,
};
const mapStateToProps = (state) => ({
  errors: state.errors,
});
export default connect(mapStateToProps, {
  createProject,
})(AddProject);
CreateProject.js
import React from "react";
import { Link } from "react-router-dom";
const CreateProjectButton = () => {
  return (
    <React.Fragment>
    <Link to="/addProject" className="btn btn-lg btn-info">
      Create a Project
    </Link>
  )
}

```

```

</Link>

</React.Fragment>

);

};

export default CreateProjectButton;

ProjectItem.js
import React, { Component } from "react";
import { Link } from "react-router-dom";
import PropTypes from "prop-types";
import { connect } from "react-redux";
import { deleteProject } from "../../actions/projectAction";

class ProjectItem extends Component { onDeleteClick = (id) => {
  this.props.deleteProject(id);
};

render() {
  const { project } = this.props; return (
    <div className="container">

      <div className="card card-body bg-light mb-3">

        <div className="row">
          <div className="col-2">
            <span className="mx-auto">{project.projectIdentifier}</span>
          </div>
          <div className="col-lg-6 col-md-4 col-8">
            <h3>{project.projectName}</h3>
            <p>{project.description}</p>
          </div>
          <div className="col-md-4 d-none d-lg-block">
            <ul className="list-group">
              <Link to={`projectBoard/${project.projectIdentifier}`}>
                <li className="list-group-item board">

```

```

<i className="fa fa-flag-checkered pr-1">Project Board </i>
</li>
</Link>
<Link to={` /updateProject/${project.projectIdentifier}`} >
<li className="list-group-item update">
<i className="fa fa-edit pr-1">Update Project Info</i>
</li>
</Link>
<li
className="list-group-item delete"
onClick={this.onDeleteClick.bind(
    this,
    project.projectIdentifier
    )}
>
<i className="fa fa-minus-circle pr-1">Delete Project</i>
</li>
</ul>
</div>
</div>
</div>
</div>
);
}
}
ProjectItem.propTypes = {
deleteProject: PropTypes.func.isRequired,
};
export default connect(null, { deleteProject })(ProjectItem);

```



```
UpdateProject.js
import React, { Component } from "react";
import { getProject, createProject } from "../../actions/projectAction";
import PropTypes from "prop-types";
import { connect } from "react-redux";
import classNames from "classnames";
import { GET_ERRORS } from "../../actions/type";
class UpdateProject extends Component {
  constructor() {
    super();
    this.state = {
      id: "",
      projectName: "",
      projectIdentifier: "",
      description: "",
      startDate: "",
      endDate: "",
      errors: {},
    };
    this.onChange = this.onChange.bind(this);
    this.onSubmit = this.onSubmit.bind(this);
  }
  componentWillReceiveProps(nextProps) {
    if (nextProps.errors) {
      this.setState({ errors: nextProps.errors });
    }
    const {
      id,
      projectName,
      projectIdentifier,
```



```
        description,
        startDate,
        endDate,
        } = nextProps.project;
        this.setState({
            id,
            projectName,
            projectIdentifier,
            description,
            startDate,
            endDate,
        });
    }

    componentDidMount() {
        const { id } = this.props.match.params;
        this.props.getProject(id, this.props.history);
    }
    onChange(e) {
        this.setState({ [e.target.name]: e.target.value });
    }
    onSubmit(e) {
        e.preventDefault();
        const updateProject = {
            id: this.state.id,
            projectName: this.state.projectName,
            projectIdentifier: this.state.projectIdentifier,
            description: this.state.description,
            startDate: this.state.startDate,
            endDate: this.state.endDate,
```



```
};

this.props.createProject(updateProject, this.props.history);

}

render() {

  const { errors } = this.state;

  return (
    <div className="project">

      <div className="container">

        <div className="row">

          <div className="col-md-8 m-auto">

            <h5 className="display-4 text-center">

              Create / Edit Project form

            </h5>

            <hr />

            <form onSubmit={this.onSubmit}>
            <div className="form-group">
            <input

              type="text"

              className={classnames("form-control form-control-lg", {
                "is-invalid": errors.projectName,
              })}

              placeholder="Project Name"
              name="projectName"
              value={this.state.projectName}

              onChange={this.onChange}

            />

            {errors.projectName && (
            <div className="invalid-feedback">{errors.projectName}</div>

            )}


```

```

</div>

<div className="form-group">
<input
    type="text"

    className={classnames("form-control form-control-lg", {
        "is-invalid": errors.projectIdentifier,
    })}

    placeholder="Unique Project ID"
    name="projectIdentifier"
    value={this.state.projectIdentifi
er} disabled
/>

    {errors.projectIdentifier&& (
<div className="invalid-
    feedback">
        {errors.projectIdentifier}
    </div>

    )}

</div>

<div className="form-group">
<textarea
    className={classnames("form-control form-
        control-lg", { "is-invalid":
            errors.description,
        })}
    placeholder="Project Description"
    name="description"
    value={this.state.description}
    onChange={this.onChange}
    ></textarea>

    {errors.description&& (

```

```

<div className="invalid-feedback">{errors.description}</div>
    )}
</div>
<h6>Start Date</h6>
<div className="form-group">
  <input
    type="date"
    className="form-control form-control-lg"
    name="startDate"
    value={this.state.startDate}
    onChange={this.onChange}
  />
</div>
<h6>Estimated End Date</h6>
<div className="form-group">
  <input
    type="date"
    className="form-control form-control-lg"
    name="endDate"
    value={this.state.endDate}
    onChange={this.onChange}
  />
</div>
  <input
    type="submit"
    className="btn btn-primary btn-block mt-4"
  />
</form>
</div>
</div>
</div>

```

</div>

);

}

}

UpdateProject.propTypes = {

getProject: PropTypes.func.isRequired,

createProject: PropTypes.func.isRequired,

project: PropTypes.object.isRequired,

errors: PropTypes.object.isRequired,

};

const mapStateToProps = (state) => ({

project: state.project.project,

errors: state.errors,

});

export default connect(mapStateToProps, { getProject, createProject })(

UpdateProject

);

Login.js

import React, { Component } from

“react”; import PropTypes from “prop-

types”; import { connect } from “react-

redux”; import classNames from

“classnames”;

import { login } from “../actions/securityActions”;

class Login extends Component {

constructor() {

super();

this.state = {

username: “”,

password: “”,

errors: {},



```
};

this.onChange = this.onChange.bind(this);

this.onSubmit = this.onSubmit.bind(this);
}

componentDidMount() {

  if (this.props.security.validToken) {
this.props.history.push("/dashboard");
  }

}

componentWillReceiveProps(nextProps) {

  if (nextProps.security.validToken) {
this.props.history.push("/dashboard");
  }

  if (nextProps.errors) {

this.setState({ errors: nextProps.errors });
  }

}

onSubmit(e) {
e.preventDefault();

const LoginRequest = {
  username: this.state.username,
  password: this.state.password,
};

this.props.login(LoginRequest);
}

onChange(e) {
this.setState({ [e.target.name]: e.target.value });
}

render() {
```

```

    const { errors } = this.state;
    return (
<div className="login">
<div className="container">
<div className="row">
<div className="col-md-8 m-auto">
<h1 className="display-4 text-center">Log In</h1>
<form onSubmit={this.onSubmit}>
<div className="form-group">
<input
      type="text"
      className={classnames("form-control form-control-lg", {
        "is-invalid": errors.username,
      })}
      placeholder="Email
      Address"
      name="username"
      value={this.state.username
    }
    onChange={this.onChange}
  />
      {errors.username && (
<div className="invalid-feedback">{errors.username}</div>
      )}
    </div>
<div className="form-group">
<input
      type="password"
      className={classnames("form-control form-control-lg", {
        "is-invalid": errors.password,
      })}

```

```

        placeholder="Password"
        name="password"
        value={this.state.password}
onChange={this.onChange}
    />

    {errors.password && (
<div className="invalid-feedback">{errors.password}</div>

    )}

</div>

<input type="submit" className="btn btn-info btn-block mt-4" />

</form>

</div>

</div>

</div>

</div>

    );
}
}

Login.propTypes = {
    login: PropTypes.func.isRequired,
    errors: PropTypes.object.isRequired,
    security: PropTypes.object.isRequired,
};

const mapStateToProps = (state) => ({
    security: state.security,
    errors: state.errors,

```

```

    });

export default connect(mapStateToProps, { login })(Login);

Register.js

import React, { Component } from "react";

import { createUser } from "../../actions/securityActions";

import PropTypes from "prop-types";
import { connect } from "react-redux";
import classNames from "classnames";

class Register extends Component {
  constructor() {
    super();
    this.state = {
      username: "",
      fullName: "",
      password: "",
      confirmPassword: "",
      errors: {},
    };
  }

  this.onChange = this.onChange.bind(this);

  this.onSubmit = this.onSubmit.bind(this);
  }

  componentDidMount() {
    if (this.props.security.validToken) {
      this.props.history.push("/dashboard");
    }
  }

  componentWillReceiveProps(nextProps) {
    if (nextProps.errors)
      this.setState({ errors: nextProps.errors });
  }

```



```

    }
    onSubmit(e) {
      e.preventDefault();
      const newUser = {
        username: this.state.username,
        fullName: this.state.fullName,
        password: this.state.password,
        confirmPassword: this.state.confirmPassword,
      };
      this.props.createNewUser(newUser, this.props.history);
    }
    onChange(e) {
      this.setState({ [e.target.name]: e.target.value });
    }
    render() {
      const { errors } = this.state;
      return (
        <div className="register">
          <div className="container">
            <div className="row">
              <div className="col-md-8 m-auto">
                <h1 className="display-4 text-center">Sign Up</h1>
                <p className="lead text-center">Create your Account</p>
                <form onSubmit={this.onSubmit}>
                  <div className="form-group">
                    <input
                      type="text"
                      className={classnames("form-control form-control-lg", {
                        "is-invalid": errors.fullName,
                      })}
                      placeholder="Full Name"

```

```

        name="fullName"
        value={this.state.fullName}
        onChange={this.onChange}
      />
      {errors.fullName && (
<div className="invalid-feedback">{errors.fullName}</div>
      )}
    </div>
    <div className="form-group">
      <input
        type="text"
        className={classnames("form-control form-control-lg", {
          "is-invalid": errors.username,
        })}
        placeholder="Email Address (Username)"
        name="username"
        value={this.state.username}
        onChange={this.onChange}
      />
      {errors.username && (
<div className="invalid-feedback">{errors.username}</div>
      )}
    </div>
    <div className="form-group">
      <input
        type="password"
        className={classnames("form-control form-control-lg", {
          "is-invalid": errors.password,
        })}
        placeholder="Password"
        name="password"
        value={this.state.password}
        onChange={this.onChange}

```

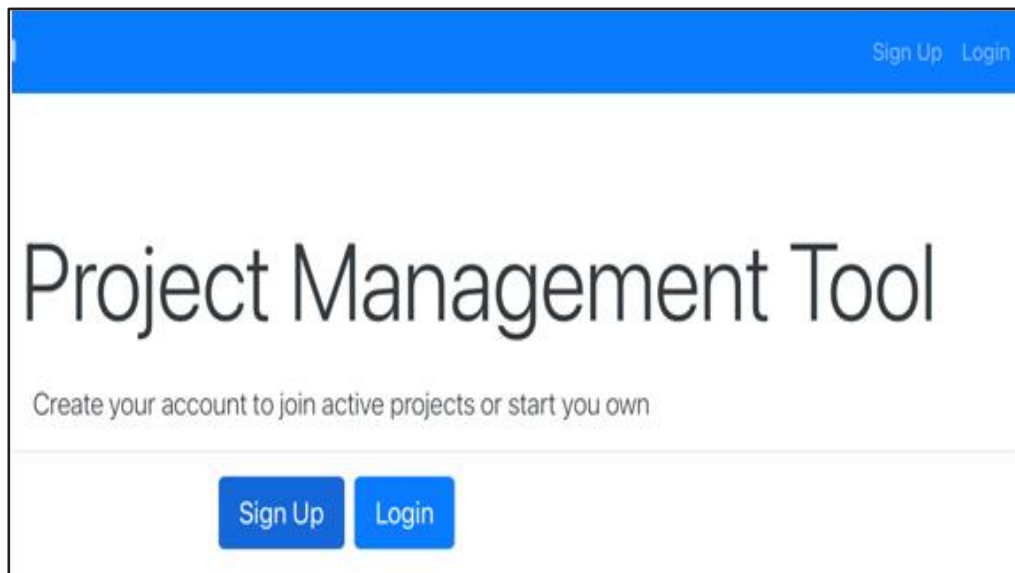
```

        />
        {errors.password&& (
<div className="invalid-feedback">{errors.password}</div>
        )}
</div>
<div className="form-group">
<input
        type="password"
        className={classnames("form-control form-control-lg", {
            "is-invalid": errors.confirmPassword,
        })}
        placeholder="Confirm Password"
        name="confirmPassword"
        value={this.state.confirmPassword}
        onChange={this.onChange}
        />
        {errors.confirmPassword&& (
<div className="invalid-feedback">
            {errors.confirmPassword}
</div>
        )}
</div>
<input type="submit" className="btn btn-info btn-block mt-4" />
</form>
</div>
</div>
</div>
</div>
);
}

```

```
}  
Register.propTypes = {  
  createNewUser: PropTypes.func.isRequired,  
  errors: PropTypes.object.isRequired,  
  security: PropTypes.object.isRequired,  
};  
const mapStateToProps = (state) => ({  
  errors: state.errors,  
  security: state.security,  
});  
export default connect(mapStateToProps, { createNewUser })(Register);
```

OUTPUT





Sign Up

Create your Account

Log In

Projects

(ONCE YOU CREATE A PROJECT GOTO =>  Project Board IN PROJECTS)

Create a Project

RCC12	React React is of-course easy to use	Project Board Update Project Info Delete Project
AMAR	C++ C++ is difficult to learn	Project Board Update Project Info Delete Project

Create Project Task

TO DO	In Progress	Done
ID: RCC12-4 -- Priority: LOW HELLO THERE amar View / Update Delete	ID: RCC12-5 -- Priority: MEDIUM HEKLO first View / Update Delete	ID: RCC12-3 -- Priority: LOW TASK_3 EASY TO BUILD View / Update Delete

RESULT:

The program to implement full-stack application using React and Spring, Spring REST, Spring Security, Spring Data JPA, Hibernate, Spring Boot, Gradle and React's higher order component has been implemented successfully.