================================
02- Mini Project Requirement
================================

Develop Front Office Team Portal for Managing Student Enquiries

Project Mock Up Screens : http://13.232.229.99/

==================
Functionalities
==================

1) Home Page with caurosel

2) Sign Up

3) Unlock Account

4) Login (Only unlocked account users can login)

5) Dashboard (Logged in user performance report)

6) Add Enquriry

7) View Enquiries (with Dynamic Filters)

8) Edit and Update Enquriy Status

9) Filter Enqueries

10) Logout

Note -1 : When user logged in, only that logged in user performance report should display in dashboard and only that user added enquries should display in View Enquiries screen.

===================
Technology Stack
===================

Database : MySQL

Backend : Java + Spring Boot + Data JPA + Web MVC

Frontend : HTML + CSS + JQuery + BootStrap + Ajax + Thymeleaf

Front Ofc Team : registration/unlock/login//forgotpwd/dashboard

student enquiries : Add Enquiry / View Enqury / Edit & Update

=================
Execution Flow

================

1) User will open website home page

-> Logo Display with Nav bar
-> Carousel (banners sliding)
-> Course Images Display
-> Login Button

2) User Registration

-> Unique Email Validation
-> Temporary Pwd should be generated by app
-> Send Registration Email with Temp Pwd and Unlock Link

3) Unlock Account

-> Temporary Pwd validation
-> Set New Password
-> Unlock Account

Note: user should unlock account only one time. If user try to unlock account secondtime application should display message as "Your account already unlocked"

4) Login Functionality :

-> Unlocked account users only can login
-> Invalid Credentials Check (error : Invalid Credentials)
-> Account Status Check (error : Your account locked)
-> Login Success : Display Dasboard with Logged in User Performance report

5) Forgot Pwd :

-> Invalid Email : No Account Found With Given Email
-> Valid Email : Get Pwd from DB and Send User Pwd to user email

6) Dashboard :

-> Display logged in user performance report

7) Add Enquiry :

-> Add new Enquirty with status as 'New'

8) View Enquries :

-> Display enquries which are added by logged in user
-> Edit Enquiry to update Enquiry Status

-> Filter Enquiries based on given criteria (Async)

9) Logout :

-> When user click on logout display home page

===========

Db_Tables

============

1 :   AIT_USER_DTLS

USER_ID	INTEGER	PK AUTO_INCREMENT
NAME	VARCHAR	
EMAIL	VARCHAR	UNIQUE
PHNO	INTEGER	
PWD	VARCHAR	
ACC_STATUS	VARCHAR	(DEFAULT : LOCKED)

2 :   AIT_STUDENT_ENQURIES

ENQUIRY_ID	INTEGER	PK AUTO_INCREMENT
STUDENT_NAME	VARCHAR	
PHNO	INTEGER	
CLASS_MODE	VARCHAR	
COURSE_NAME	VARCHAR	
ENQUIRY_STATUS	VARCHAR	(Default : NEW)
CREATED_DATE	DATE	
UPDATED_DATE	DATE	
USER_ID	INTEGER	FK REF : AIT_USER_DTLS

3 :   AIT_COURSES

COURSE_ID	INTEGER	PK	AUTO_INCREMENT
COURSE_NAME	VARCHAR		

4 :   AIT_ENQURIRY_STATUS

STATUS_ID	INTEGER	PK AUTO_INCREMENT
STATUS_NAME	VARCHAR	

======================== Components Analysis ================

-------------

Entities

--------------

UserDtlsEntity.java
StudentEnqEntity.java
CourseEntity.java
EnqStatusEntity.java

--------------

Repositories

UserDtlsRepo.java
StudentEnqRepo.java
CourseRepo.java
EnqStatusRepo.java

Binding Classes

LoginForm.java
SignUpForm.java
UnlockForm.java

DashboardResponse.java
EnquiryForm.java
EnquirySearchCriteria.java

Services

UserService.java & UserServiceImpl.java
EnquiryService.java & EnquiryServiceImpl.java

Helper Classes (Utility)

PwdUtils.java
EmailUtils.java
AppExceptionHadler.java

Controllers

IndexController.java
UserController.java
EnquiryController.java

----
pages
----
index.html
login.html
singup.html
unlock.html
forgotPwd.html
dashboard.html
addEnquiry.html
viewEnquiry.html
errorPage.html

Runner

DataLoader.java (insert course names and enquriy status)

===

==================
Types of DB tables
==================

1) Transactional Table (INSERT / UPDATE / DELETE / RETRIEVE)

2) Non-Transactional / Static table (retrieve)

Note: We can insert data into static tables using insert queries directley (we can use runner also)

==============
Project Setup
===============

1) Create Project with required dependencies

a) web-starter
b) data-jpa-starter
c) mysql-driver
d) mail-starter
e) thymeleaf-starter
f) validation-starter
g) project-lombok
h) devtools

2) Configure Datasource & SMTP properties in application.properties file

3) Create Entity classes & Repository interfaces

4) Create Form Binding classes

5) Create Helper classes (Utility classes)

6) Create Service Interfaces with Impl classes

7) Create Controller classes with required methods

8) Create View Pages with presentation logic

9) Run the application and test it.

================== Download Resources (images + view pages)
========================

https://www.mediafire.com/file/vswc2h4nswu3ncn/02_mini_project_resources.zip/file

===
=====

Module-1 : User management
----
1) Index Page
2) Sign Up with Unique Email validation
3) Unlock Account
4) Login
5) Recover Password

Module-2 : Enquriy management

-> Enq Management functionality is depends on logged in user

-> As it is user specific functionality, our application should remember logged in user details

-> To remember logged in user details we will use Session in our application.

Note: When user logged in - session should be created. When user logout then session should be destroyed.

1) Dashboard
2) Add Enquriy
3) View Enquiries
4) Enquriy Filter
5) Logout

=======
AJAX
=======

2 Types of requests

1) Synchronus Request (whole web page will be reloaded)
2) Asynchronus Request (DOM will be updated without reloading page)

Note: To send Async request to server we can use AJAX

-> Ajax stands for Asynchronus Java Script and XML

1) We can reload only data without loading whole web page
2) We can send/read data to/from server in async mode
3) Semless experience for end user

1) Add jquery dependency in pom.xml

```
<dependency>
<groupId>org.webjars</groupId>
<artifactId>jquery</artifactId>
<version>3.6.4</version>
</dependency>
```

2) Include jquery.min.js file in our html page in <head/> tag

```
<script src="/webjars/jquery/3.6.4/jquery.min.js"></script>
```

3) write ajax logic to send async request to server

```
<Script>
$(document).ready(function() {

$("#cname").on("change", function(e) {
$.ajax({
type : "GET",
url : "cmsg",
data : {
cname : $("#cname").val()
},
success : function(data) {
$("#dropDownDiv").html(data);
}
});
});

$("#submitBtn").click(function(e) {
$.ajax({
type : "GET",
url : "msg",
data : {
name : $("#username").val()
},
success : function(data) {
$("#msgDiv").html(data);
},
error : function(result) {
alert('error');
}
});
});
});
</Script>
```

4) Update response in web page

```
<!DOCTYPE html>
<html xmlns:th="www.thymeleaf.org">
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>

<script src="/webjars/jquery/3.6.4/jquery.min.js"></script>

<Script>
$(document).ready(function() {

$("#cname").on("change", function(e) {
$.ajax({
type : "GET",
url : "cmsg",
data : {
cname : $("#cname").val()
},
success : function(data) {
$("#dropDownDiv").html(data);
}
});
});

$("#submitBtn").click(function(e) {
$.ajax({
type : "GET",
url : "msg",
data : {
name : $("#username").val()
},
success : function(data) {
$("#msgDiv").html(data);
},
error : function(result) {
alert('error');
}
});
});
});
</Script>

</head>
<body>

Enter Name :
```

```
<input type="text" name="name" id="username" />

<input type="button" value="Submit" id="submitBtn" />

<hr />

<div id="msgDiv"></div>

<hr />

<select id="cname">
<option>-Select-</option>
<option>India</option>
<option>USA</option>
<option>UK</option>
<option>Canada</option>
</select>

<hr />

<div id="dropDownDiv"></div>

</body>
</html>
```

==============================

```
package in.ashokit.rest;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.ResponseBody;

@Controller
public class UserRestController {

@GetMapping("/")
public String load() {
return "index";
}

@GetMapping("/cmsg")
@ResponseBody
public String getDropdownMsg(@RequestParam("cname") String cname) {
String msg = "I am going to "+ cname+" in next month";
return msg;
}

@GetMapping("/msg")
@ResponseBody
public String getMsg(@RequestParam("name") String name) {
String msg = "Hello, " + name;
return msg;
}
}
```
==