**Developing a Backend Admin for Learner’s Academy.**

PHASE2-FINAL-Project

DESCRIPTION

**Project objective:**

As a Full Stack Developer, design and develop a backend administrative portal for the Learner’s Academy. Use the GitHub repository to manage the project artifacts.

**Background of the problem statement:**

Learner’s Academy is a school that has an online management system. The system keeps track of its classes, subjects, students, and teachers. It has a back-office application with a single administrator login.

**The administrator can:**

● Set up a master list of all the subjects for all the classes  
● Set up a master list of all the teachers  
● Set up a master list of all the classes  
● Assign classes for subjects from the master list  
● Assign teachers to a class for a subject (A teacher can be assigned to different classes for different subjects)  
● Get a master list of students (Each student must be assigned to a single class)

There will be an option to view a Class Report which will show all the information about the class, such as the list of students, subjects, and teachers  
       
The goal of the company is to deliver a high-end quality product as early as possible.

**The flow and features of the application:**

● Plan more than two sprints to complete the application  
● Document the flow of the application and prepare a flow chart   
● List the core concepts and algorithms being used to complete this application  
● Implement the appropriate concepts, such as exceptions, collections, and sorting techniques for source code optimization and increased performance

**You must use the following:**

● Eclipse/IntelliJ: An IDE to code for the application   
● Java: A programming language to develop the web pages, databases, and others  
● SQL: To create tables for admin, classes, students, and other specifics  
● Git: To connect and push files from the local system to GitHub   
● GitHub: To store the application code and track its versions   
● Scrum: An efficient agile framework to deliver the product incrementally   
● Search and Sort techniques: Data structures used for the project   
● Specification document: Any open-source document or Google Docs

**The following requirements should be met:**

● The source code should be pushed to your GitHub repository. You need to document the steps and write the algorithms in it.  
● The submission of your GitHub repository link is mandatory. In order to track your task, you need to share the link of the repository. You can add a section in your document.   
● Document the process step-by-step starting from sprint planning to the product release.   
● The application should not close, exit, or throw an exception if the user specifies an invalid input.  
● You need to submit the final specification document which will include:   
● Project and developer details   
● Sprints planned and the tasks achieved in them   
● Algorithms and flowcharts of the application   
● Core concepts used in the project   
● Links to the GitHub repository to verify the project completion

**Class.java**

**package** com.simplilearn.models;

**public** **class** Class {

**private** **int** id;

**private** **int** section;

**private** String teacher;

**private** String subject;

**private** String time;

**public** Class(**int** id, **int** section, String teacher, String subject, String time) {

**super**();

**this**.id = id;

**this**.section = section;

**this**.teacher = teacher;

**this**.subject = subject;

**this**.time = time;

}

**public** **int** getId() {

**return** id;

}

**public** **void** setId(**int** id) {

**this**.id = id;

}

**public** **int** getSection() {

**return** section;

}

**public** **void** setSection(**int** section) {

**this**.section = section;

}

**public** String getTeacher() {

**return** teacher;

}

**public** **void** setTeacher(String teacher) {

**this**.teacher = teacher;

}

**public** String getSubject() {

**return** subject;

}

**public** **void** setSubject(String subject) {

**this**.subject = subject;

}

**public** String getTime() {

**return** time;

}

**public** **void** setTime(String time) {

**this**.time = time;

}

}

**Subject.java**

**package** com.simplilearn.models;

**public** **class** Subject {

**private** **int** id;

**private** String name;

**private** String shortcut;

**public** Subject(**int** id, String name, String shortcut ) {

**super**();

**this**.id = id;

**this**.name = name;

**this**.shortcut = shortcut;

}

**public** **int** getId() {

**return** id;

}

**public** **void** setId(**int** id) {

**this**.id = id;

}

**public** String getShortcut() {

**return** shortcut;

}

**public** **void** setShortcut(String shortcut) {

**this**.shortcut = shortcut;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

}

**Teacher.java**

**public** **class** Teacher {

**private** **int** id;

**private** String fname;

**private** String lname;

**private** **int** age;

**public** Teacher(**int** id, String fname, String lname, **int** age) {

**super**();

**this**.id = id;

**this**.fname = fname;

**this**.lname = lname;

**this**.age = age;

}

**public** **int** getId() {

**return** id;

}

**public** **void** setId(**int** id) {

**this**.id = id;

}

**public** String getFname() {

**return** fname;

}

**public** **void** setFname(String fname) {

**this**.fname = fname;

}

**public** String getLname() {

**return** lname;

}

**public** **void** setLname(String lname) {

**this**.lname = lname;

}

**public** **int** getAge() {

**return** age;

}

**public** **void** setAge(**int** age) {

**this**.age = age;

}

}

**AdminControllerServlet:**

**package** com.simplilearn.admin;

**import** java.io.IOException;

**import** java.util.List;

**import** javax.annotation.Resource;

**import** javax.servlet.RequestDispatcher;

**import** javax.servlet.ServletException;

**import** javax.servlet.annotation.WebServlet;

**import** javax.servlet.http.Cookie;

**import** javax.servlet.http.HttpServlet;

**import** javax.servlet.http.HttpServletRequest;

**import** javax.servlet.http.HttpServletResponse;

**import** javax.sql.DataSource;

**import** com.simplilearn.models.Student;

**import** com.simplilearn.models.Subject;

**import** com.simplilearn.models.Teacher;

**import** com.simplilearn.models.Class;

/\*\*

\* Servlet implementation class AdminControllerServlet

\*/

@WebServlet("/AdminControllerServlet")

**public** **class** AdminControllerServlet **extends** HttpServlet {

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

**private** DbRetrieve dbRetrieve;

@Resource(name = "jdbc\_database")

**private** DataSource datasource;

@Override

**public** **void** init() **throws** ServletException {

**super**.init();

// create instance of db util, to pass in conn pool object

**try** {

dbRetrieve = **new** DbRetrieve(datasource);

} **catch** (Exception e) {

**throw** **new** ServletException(e);

}

}

/\*\*

\* **@see** HttpServlet#HttpServlet()

\*/

**public** AdminControllerServlet() {

**super**();

// **TODO** Auto-generated constructor stub

}

@Override

**protected** **void** doPost(HttpServletRequest req, HttpServletResponse resp) **throws** ServletException, IOException {

doGet(req, resp);

}

/\*\*

\* **@see** HttpServlet#doGet(HttpServletRequest request, HttpServletResponse

\* response)

\*/

**protected** **void** doGet(HttpServletRequest request, HttpServletResponse response)

**throws** ServletException, IOException {

// **TODO** Auto-generated method stub

**try** {

// read the "command" parameter

String command = request.getParameter("command");

**if** (command == **null**) {

command = "CLASSES";

}

// if no cookeies

**if** (!getCookies(request, response) && (!command.equals("LOGIN"))) {

response.sendRedirect("/PHASE2-FINAL-PROJECT/login.jsp");

}

**else** {

// if there is no command, how to handle

// route the data to the appropriate method

**switch** (command) {

**case** "STUDENTS":

studentsList(request, response);

**break**;

**case** "TEACHERS":

teachersList(request, response);

**break**;

**case** "SUBJECTS":

subjectList(request, response);

**break**;

**case** "CLASSES":

classestList(request, response);

**break**;

**case** "ST\_LIST":

classStudentsList(request, response);

**break**;

**case** "LOGIN":

login(request, response);

**break**;

**default**:

classestList(request, response);

}

}

} **catch** (Exception e) {

**throw** **new** ServletException(e);

}

// response.getWriter().append("Served at: ").append(request.getContextPath());

}

**private** **void** studentsList(HttpServletRequest request, HttpServletResponse response) **throws** Exception {

// get students from db util

List<Student> students = dbRetrieve.getStudents();

// add students to the request

request.setAttribute("STUDENT\_LIST", students);

// send it to the jsp view page

RequestDispatcher dispatcher = request.getRequestDispatcher("/list-students.jsp");

dispatcher.forward(request, response);

}

**private** **void** teachersList(HttpServletRequest request, HttpServletResponse response) **throws** Exception {

// get students from db util

List<Teacher> teachers = dbRetrieve.getTeachers();

// add students to the request

request.setAttribute("TEACHERS\_LIST", teachers);

// send it to the jSP view page

RequestDispatcher dispatcher = request.getRequestDispatcher("/teachers-list.jsp");

dispatcher.forward(request, response);

}

**private** **void** subjectList(HttpServletRequest request, HttpServletResponse response) **throws** Exception {

// get subjects from db util

List<Subject> subjects = dbRetrieve.getSubjects();

// add subjects to the request

request.setAttribute("SUBJECTS\_LIST", subjects);

// send it to the jSP view page

RequestDispatcher dispatcher = request.getRequestDispatcher("/subjects-list.jsp");

dispatcher.forward(request, response);

}

**private** **void** classestList(HttpServletRequest request, HttpServletResponse response) **throws** Exception {

// get subjects from db util

List<Class> classes = dbRetrieve.getClasses();

// add subjects to the request

request.setAttribute("CLASSES\_LIST", classes);

// send it to the jSP view page

RequestDispatcher dispatcher = request.getRequestDispatcher("/classes-list.jsp");

dispatcher.forward(request, response);

}

**private** **void** login(HttpServletRequest request, HttpServletResponse response) **throws** Exception {

String username = request.getParameter("username");

String password = request.getParameter("password");

**if** (username.toLowerCase().equals("admin") && password.toLowerCase().equals("admin")) {

Cookie cookie = **new** Cookie(username, password);

// Setting the maximum age to 1 day

cookie.setMaxAge(86400); // 86400 seconds in a day

// Send the cookie to the client

response.addCookie(cookie);

classestList(request, response);

} **else** {

RequestDispatcher dispatcher = request.getRequestDispatcher("/login.jsp");

dispatcher.forward(request, response);

}

}

**private** **void** classStudentsList(HttpServletRequest request, HttpServletResponse response) **throws** Exception {

**int** classId = Integer.*parseInt*(request.getParameter("classId"));

String section = request.getParameter("section");

String subject = request.getParameter("subject");

// get subjects from db util

List<Student> students = dbRetrieve.loadClassStudents(classId);

// add subjects to the request

request.setAttribute("STUDENTS\_LIST", students);

request.setAttribute("SECTION", section);

request.setAttribute("SUBJECT", subject);

// send it to the jSP view page

RequestDispatcher dispatcher = request.getRequestDispatcher("/class-students.jsp");

dispatcher.forward(request, response);

}

**private** **boolean** getCookies(HttpServletRequest request, HttpServletResponse response) **throws** Exception {

**boolean** check = **false**;

Cookie[] cookies = request.getCookies();

// Find the cookie of interest in arrays of cookies

**for** (Cookie cookie : cookies) {

**if** (cookie.getName().equals("admin") && cookie.getValue().equals("admin")) {

check = **true**;

**break**;

}

}

**return** check;

}

}

**TestServlet:**

**package** com.simplilearn.admin;

**import** java.io.IOException;

**import** java.io.PrintWriter;

**import** java.sql.Connection;

**import** java.sql.ResultSet;

**import** java.sql.Statement;

**import** javax.annotation.Resource;

**import** javax.servlet.ServletException;

**import** javax.servlet.annotation.WebServlet;

**import** javax.servlet.http.HttpServlet;

**import** javax.servlet.http.HttpServletRequest;

**import** javax.servlet.http.HttpServletResponse;

**import** javax.sql.DataSource;

/\*\*

\* Servlet implementation class TestServlet

\*/

@WebServlet("/TestServlet")

**public** **class** TestServlet **extends** HttpServlet {

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

//Define datasource/connection pool for reference

@Resource(name="jdbc\_database")

**private** DataSource dataSource;

/\*\*

\* **@see** HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)

\*/

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

// Set the printwriter

PrintWriter out = response.getWriter();

response.setContentType("text/plain");

// establish connection to the DB

Connection myConn = **null**;

Statement myStmt = **null**;

ResultSet myRs = **null**;

**try** {

myConn = dataSource.getConnection();

//create a sql statement

String sql = "select \* from students";

myStmt = myConn.createStatement();

//execute the sql statement

myRs = myStmt.executeQuery(sql);

//process the resultset

**while**(myRs.next()) {

String fname = myRs.getString("fname");

out.println(fname);

}

}

**catch**(Exception e) {

e.printStackTrace();

}

}

}

**login.jsp**

<%@ page language=*"java"* contentType=*"text/html; charset=UTF-8"*

pageEncoding=*"UTF-8"*%>

<!DOCTYPE html>

<html>

<head>

<meta charset=*"ISO-8859-1"*>

<title>Login</title>

</head>

<body>

<div>

<h3>Developer Name: Posi Papa Rohini Eli</h3>

<h3>Developer Email: mailtorohini@gmail.com</h3>

<h3>SimpliLearn Full Stack Development Course Phase 2</h3>

</div>

<form action=*"AdminControllerServlet"* method=*"POST"*>

<div class=*"container"*>

<input type=*"hidden"* name=*"command"* value=*"LOGIN"* />

<label>Username : </label>

<br/>

<input type=*"text"* placeholder=*"Enter Username"* name=*"username"* required>

<br/>

<label>Password : </label>

<br/>

<input type=*"password"* placeholder=*"Enter Password"* name=*"password"* required>

<br/>

<button type=*"submit"*>Login</button>

<br/>

<input type=*"checkbox"* checked=*"checked"*> Remember me

</div>

</form>

</body>

</html>

**web.xml**

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<web-app xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xmlns=*"http://java.sun.com/xml/ns/javaee"* xsi:schemaLocation=*"http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd"* id=*"WebApp\_ID"* version=*"2.5"*>

<display-name>PHASE2-FINAL-PROJECT1</display-name>

<welcome-file-list>

<welcome-file>login.jsp</welcome-file>

<description></description>

<display-name>AdminControllerServlet</display-name>

<servlet-name>AdminControllerServlet</servlet-name>

<servlet-class>com.simplilearn.admin.AdminControllerServlet</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>AdminControllerServlet</servlet-name>

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

</servlet-mapping>

<servlet>

<description></description>

<display-name>TestServlet</display-name>

<servlet-name>TestServlet</servlet-name>

<servlet-class>com.simplilearn.admin.TestServlet</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>TestServlet</servlet-name>

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

</servlet-mapping>

</web-app>

**SQL CODE:**

create database WebAdmin;

Use WebAdmin;

CREATE TABLE `classes` (

`id` int(11) NOT NULL,

`section` int(55) NOT NULL,

`teacher` int(11) NOT NULL,

`subject` int(11) NOT NULL,

`time` varchar(44) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

--

-- Dumping data for table `classes`

--

INSERT INTO `classes` (`id`, `section`, `teacher`, `subject`, `time`) VALUES

(1, 1, 1, 1, '9:00'),

(2, 3, 2, 2, '11:30');

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

--

-- Table structure for table `students`

--

CREATE TABLE `students` (

`id` int(11) NOT NULL,

`fname` varchar(55) NOT NULL,

`lname` varchar(55) NOT NULL,

`age` int(11) DEFAULT NULL,

`class` int(11) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

--

-- Dumping data for table `students`

--

INSERT INTO `students` (`id`, `fname`, `lname`, `age`, `class`) VALUES

(1, 'Ali', 'Ahsan', 21, 1),

(2, 'Hassan', 'Ahmed', 23, 2),

(4, 'Gazi', 'Dani', 21, 1),

(5, 'Tony', 'Fadel', 18, 2),

(6, 'Lami', 'Saro', 24, 1),

(7, 'Yazen', 'Rawn', 24, 2);

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

--

-- Table structure for table `subjects`

--

CREATE TABLE `subjects` (

`id` int(11) NOT NULL,

`name` varchar(55) NOT NULL,

`shortcut` varchar(50) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

--

-- Dumping data for table `subjects`

--

INSERT INTO `subjects` (`id`, `name`, `shortcut`) VALUES

(1, 'English', 'ENG'),

(2, 'Mathematics', 'MATH');

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

--

-- Table structure for table `teachers`

--

CREATE TABLE `teachers` (

`id` int(11) NOT NULL,

`fname` varchar(55) NOT NULL,

`lname` varchar(55) NOT NULL,

`age` varchar(11) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

--

-- Dumping data for table `teachers`

--

INSERT INTO `teachers` (`id`, `fname`, `lname`, `age`) VALUES

(1, 'Sami', 'Rashed', '55'),

(2, 'Rami', 'Sari', '66');

--

-- Indexes for dumped tables

--

--

-- Indexes for table `classes`

--

ALTER TABLE `classes`

ADD PRIMARY KEY (`id`),

ADD KEY `subject\_id` (`subject`),

ADD KEY `teacher\_id` (`teacher`);

--

-- Indexes for table `students`

--

ALTER TABLE `students`

ADD PRIMARY KEY (`id`),

ADD KEY `class\_id` (`class`);

--

-- Indexes for table `subjects`

--

ALTER TABLE `subjects`

ADD PRIMARY KEY (`id`);

--

-- Indexes for table `teachers`

--

ALTER TABLE `teachers`

ADD PRIMARY KEY (`id`);

--

-- AUTO\_INCREMENT for dumped tables

--

--

-- AUTO\_INCREMENT for table `classes`

--

ALTER TABLE `classes`

MODIFY `id` int(11) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=3;

--

-- AUTO\_INCREMENT for table `students`

--

ALTER TABLE `students`

MODIFY `id` int(11) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=8;

--

-- AUTO\_INCREMENT for table `subjects`

--

ALTER TABLE `subjects`

MODIFY `id` int(11) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=3;

--

-- AUTO\_INCREMENT for table `teachers`

--

ALTER TABLE `teachers`

MODIFY `id` int(11) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=3;

--

-- Constraints for dumped tables

--

--

-- Constraints for table `classes`

--

ALTER TABLE `classes`

ADD CONSTRAINT `subject\_id` FOREIGN KEY (`subject`) REFERENCES `subjects` (`id`),

ADD CONSTRAINT `teacher\_id` FOREIGN KEY (`teacher`) REFERENCES `teachers` (`id`);

--

-- Constraints for table `students`

--

ALTER TABLE `students`

ADD CONSTRAINT `class\_id` FOREIGN KEY (`class`) REFERENCES `classes` (`id`);

COMMIT;

select \* from students;

select \* from classes;

select \* from subjects;

select \* from teachers;

**Pushing the code to your GitHub repositories**

* Open your command prompt and navigate to the folder where you have created your files.

**cd <folder path>**

* Initialize your repository using the following command:

**git init**

* Add all the files to your git repository using the following command:

**git add .**

* Commit the changes using the following command:

**git commit . -m “Changes have been committed.”**

* Push the files to the folder you initially created using the following command:

**git push -u origin master**