

Project Specification Document

Learner's Academy

End of Phase 2-Become a back-end expert
Project

Student: Thirumavalaven

vthiru97@gmail.com

Full Stack Java Developer

Master's Program

Project Objective

As a Full Stack Developer, design and develop an airline booking portal named as Learner's Academy

Developer Details

The project is developed by Thirumavalaven. I worked as a Service Delivery Specialist at IBM India Pvt. Ltd.

The code for this project is hosted at
https://github.com/Thiru97/LearnersAcademy_Simplilearn.git

Project Details

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. 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 artefacts.

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.

Sprint Planning and Tasks Achieved

The project is planned to be completed in 1 sprint. Tasks assumed to be completed in the sprint are:

- Creating the flow of the application
- Initializing git repository to track changes as development progresses.
- Writing the Java program to fulfill the requirements of the project.
- Testing the Java program with different kinds of User input
- Pushing code to GitHub.
- Creating this specification document highlighting application capabilities, appearance, and user interactions.

Project Overview

The main objectives of this Projects are

- To gain an understanding of core concepts of the Java Programming Language (abstraction, polymorphism, inheritance, and encapsulation),
- Embrace the Eclipse Integrated Development Environment (IDE),
- Understand the Agile software development life cycle
- Gain familiarity with Java data structures for object-oriented applications.
- Familiarize with concepts of Servlets , Servlet Filter, HTTP Methods
- Familiarize with concepts of Web Application Development using Servlets
- Learn Maven, A build Automation tool and handle dependencies
- To learn SQL and do CRUD Operation(Create, Read, Update, Delete)
- To learn Hibernate and design a Web Application with Servlets and perform CRUD Operation (Create, Read, Update, Delete).
- To learn Hibernate's Association and perform @OnetoOne , @OnetoMany @ManytoMany Associations with different tables of a database
- Learn Cookie and HTTP Session Management and Perform User's Session Management in a Web Application
- Learn MVC Architecture and build a Web Application using Servlets, JSP, Hibernate and perform CRUD Operations

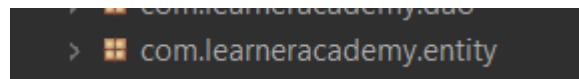
Implemented Java Concepts

This section will highlight the Java concepts used to create the Learner's Academy Airline website. HTML, CSS, JSP, JSTL, Servlets, Servlet Filters, HTTP Methods, SQL, JDBC Connection for SQL, Java Persistence API, Hibernate, HTTP Sessions and Cookie, Servlet MVC Architecture, Collections framework, Flow Control, Exception Handling, are the core concepts used in this program.

The entire Application was built in JAVA 19

Packages

I chose to create a package dedicated to the Learner's Academy Airlines as per the naming standards.

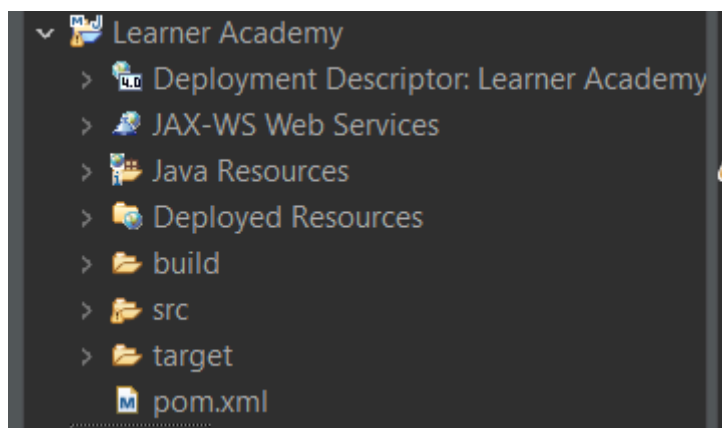


Maven

Maven is an open-source build tool from Apache Group. It is a widely-used and popular tool to build, publish, and deploy software development projects. It is used to build and manage diverse Java-based projects. A powerful project management tool, it is based on the project object model (POM) concept.

STEP 1:

Create a new Maven project via **File > New > Other > Maven > Maven Project**. Project name as Learner Academy. Once Done Maven will build the project and include a pom.xml file



STEP 2:

Add the Following dependencies in pom.xml

```
<dependencies>
  <dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>javax.servlet-api</artifactId>
    <version>3.0.1</version>
    <scope>provided</scope>
  </dependency>
  <dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>8.0.33</version>
  </dependency>
  <!-- https://mvnrepository.com/artifact/org.hibernate/hibernate-core -->
  <dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-core</artifactId>
    <version>5.6.14.Final</version>
  </dependency>
  <dependency>
    <groupId>jstl</groupId>
    <artifactId>jstl</artifactId>
    <version>1.2</version>
  </dependency>
</dependencies>
```

Servlet MVC Architecture

We will create our web application that implements the Model View Controller (MVC) design pattern, using basic Servlets and JSPs.

Model-View-Controller (MVC) is a pattern used in software engineering to separate the application logic from the user interface. As the name implies, the MVC pattern has three layers.

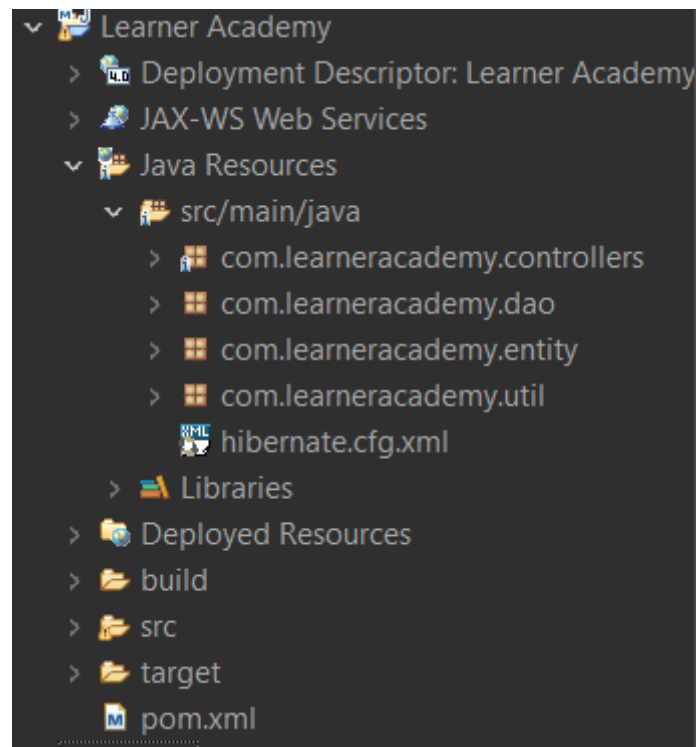
The Model defines the business layer of the application, the Controller manages the flow of the application, and the View defines the presentation layer of the application.

Although the MVC pattern isn't specific to web applications, it fits very well in this type of applications. In a Java context, the Model consists of simple Java classes, the Controller consists of servlets and the View consists of JSP pages.

Here're some key features of the pattern:

- It separates the presentation layer from the business layer
- The Controller performs the action of invoking the Model and sending data to View
- The Model is not even aware that it is used by some web application or a desktop application

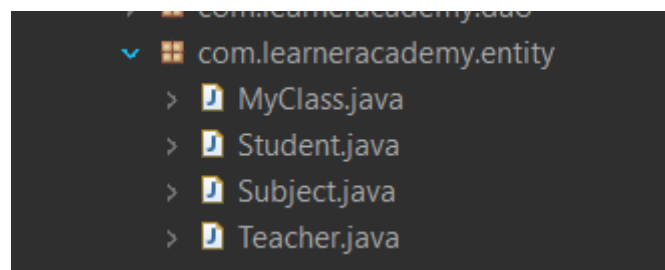
Below is the Maven and MVC Architecture based package Structure of Learner's Academy Web Application



Let's have a look at each layer.

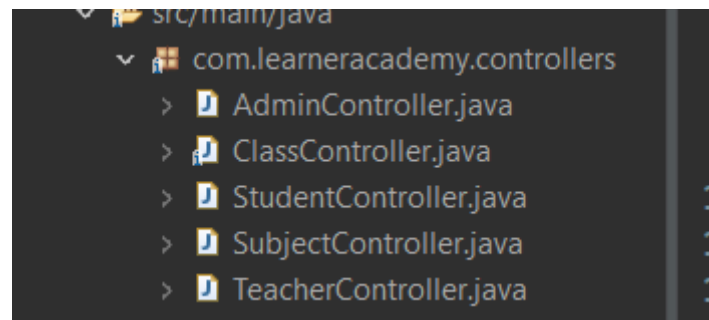
The Model Layer

This is the data layer which contains business logic of the system, and also represents the state of the application. It's independent of the presentation layer, the controller fetches the data from the Model layer and sends it to the View layer.



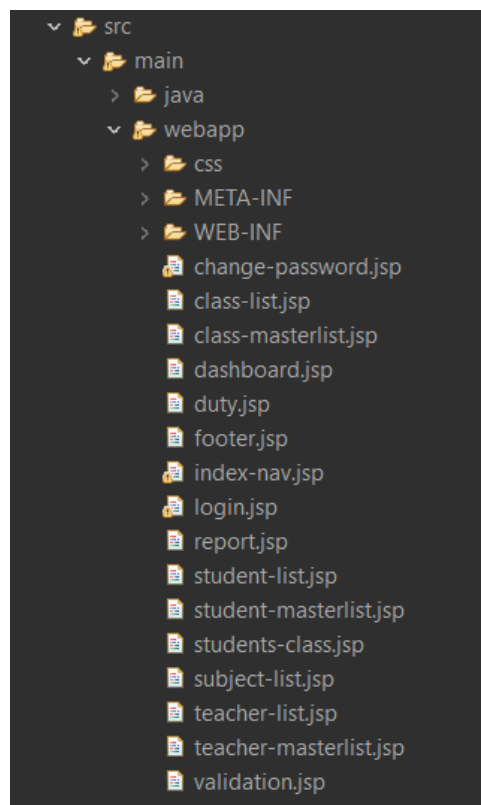
The Controller Layer

Controller layer acts as an interface between View and Model. It receives requests from the View layer and processes them, including the necessary validations. The requests are further sent to Model layer for data processing, and once they are processed, the data is sent back to the Controller and then displayed on the View.



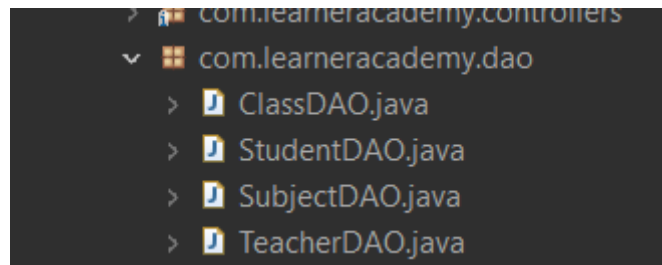
The View Layer

This layer represents the output of the application, usually some form of UI. The presentation layer is used to display the Model data fetched by the Controller.



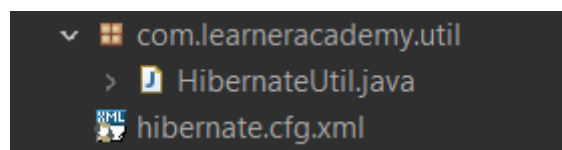
The DAO Layer

DAO stands for Data Access Object. DAO Design Pattern is used to separate the data persistence logic in a separate layer. This way, the service remains completely in dark about how the low-level operations to access the database is done. This is known as the principle of **Separation of Logic**.



The Util Package

The Util package consists of helper methods that our web application needs. We have a HibernateUtil.java files which creates a session factory with the help of hibernate.cfg.xml. The xml file hibernate.cfg.xml contains the necessary configuration needed for our hibernate to run.



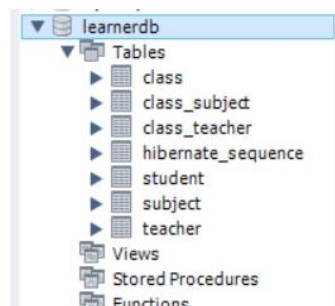
Database

I have used MySQL database. I also used an ORM (Object Relational Mapping) tool like Hibernate. Hibernate makes our project loosely coupled with respect to database. In future we can easily change from MySQL database to other database like Postgres Database. All we need to do is change the configuration file

MySQL

MySQL is a widely used relational database management system (RDBMS). MySQL is free and open-source. MySQL is ideal for both small and large applications.

I have used MySQL 8.0.33 in this application and below is the schema



Hibernate

Hibernate is an open source Object-Relational Persistence and Query service for any Java Application. Hibernate maps Java classes to database tables and from Java data types to SQL data types and relieves the developer from most common data persistence related programming tasks. Hibernate sits between traditional Java objects and database server to handle all the works in persisting those objects based on the appropriate O/R mechanisms and patterns.

Hibernate reduces lines of code by maintaining object-table mapping itself and returns result to application in form of Java objects. It relieves programmer from manual handling of persistent data, hence reducing the development time and maintenance cost.

In order to connect our database with Hibernate we need to define the configurations which includes the database port number, username and password. We define these configuration in hibernatecfg.xml file

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE hibernate-configuration PUBLIC
3   "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
4   "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
5
6 <hibernate-configuration>
7   <session-factory>
8     <property name="hibernate.connection.driver_class">com.mysql.cj.jdbc.Driver</property>
9     <property name="hibernate.connection.url">jdbc:mysql://localhost:3306/learnerdb</property>
10    <property name="hibernate.connection.username">root</property>
11    <property name="hibernate.connection.password">root@thiru1234</property>
12    <property name="dialect">org.hibernate.dialect.MySQL8Dialect</property>
13    <property name="hibernate.show_sql">true</property>
14    <!-- <property name="hibernate.enable_lazy_load_no_trans" >true</property> -->
15    <property name="hibernate.format_sql">true</property>
16    <property name="hbm2ddl.auto">update</property>
17
18   </session-factory>
19 </hibernate-configuration>
```

Hibernate Association

Association in hibernate tells the relationship between the objects of POJO classes i.e. how the entities are related to each other. Association or entities relationship can be unidirectional or bidirectional.

Below is the associations used by Learner's Academy project

@ManyToMany Association

Two items are said to be in Many-to-Many relationship if many occurrence of item are belong to the many occurrences of other item and vice versa

@OneToMany Association

A one-to-many relationship exists in a relational database when one row in table A is linked to many rows in table B, but only one row in table B is linked to one row in table A. It's vital to remember that a one-to-many relationship is the quality of the relationship, not the data.

@ManytoOne Association

When one or more entries in one table may be linked to each record in the other table, this is known as a Many-to-One relationship. This is also one of the most common types of relationship found along with one-to-many relationships. Depending on how we look at it, a Many-to-One relationship can also be described as a one-to-many relationship.

```
@Table(name = "class")
public class MyClass {

    @Id
    @GeneratedValue(strategy = GenerationType.SEQUENCE)
    private int id;

    @Column(name = "classname")
    private String className;

    @ManyToMany(mappedBy = "myClass", cascade = CascadeType.DETACH)
    @LazyCollection(LazyCollectionOption.FALSE)
    List<Subject> subject = new ArrayList<Subject>();

    @ManyToMany(mappedBy = "myClass", fetch = FetchType.EAGER)
    List<Teacher> teacher = new ArrayList<Teacher>();

    @OneToMany(mappedBy = "myClass", fetch = FetchType.LAZY)
    List<Student> student = new ArrayList<>();
}
```

Servlets

Servlets are the Java programs that run on the Java-enabled web server or application server. They are used to handle the request obtained from the webserver, process the request, produce the response, and then send a response back to the webserver.

Properties of Servlets are as follows:

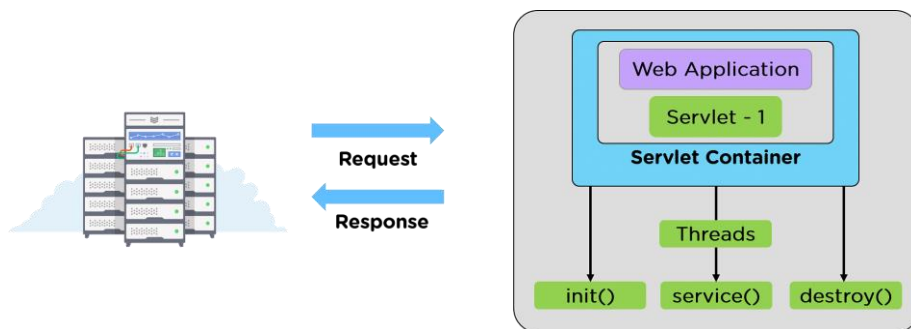
- Servlets work on the server-side.
- Servlets are capable of handling complex requests obtained from the webserver.

Execution of Servlets basically involves six basic steps:

- The clients send the request to the webserver.
- The web server receives the request.
- The web server passes the request to the corresponding servlet.
- The servlet processes the request and generates the response in the form of output.
- The servlet sends the response back to the webserver.
- The web server sends the response back to the client and the client browser displays it on the screen.

There are several varieties of interfaces and classes available in the Servlet API. Some of them are as follows:

- HTTP Servlet
- Generic Servlet
- Servlet Request
- Servlet Response



To write a Servlet, the user needs first to implement the Servlet Interface, directly or indirectly, using the following import command.

```
import javax.servlet.*;
```

Once the Servlet interface is imported, and we inherit the HTTP Class, we begin with the Java Servlet's life cycle.

In the life cycle of a servlet, we have mainly three stages, which are mentioned below.

- `init()`
- `service()`

- `destroy()`

We call these methods at their respective stages. The methods are resolved by generating the essential threads for the process to get executed.

The `service()` method is the heart of the life cycle of a Java Servlet. Right after the Servlet's initialization, it encounters the service requests from the client end.

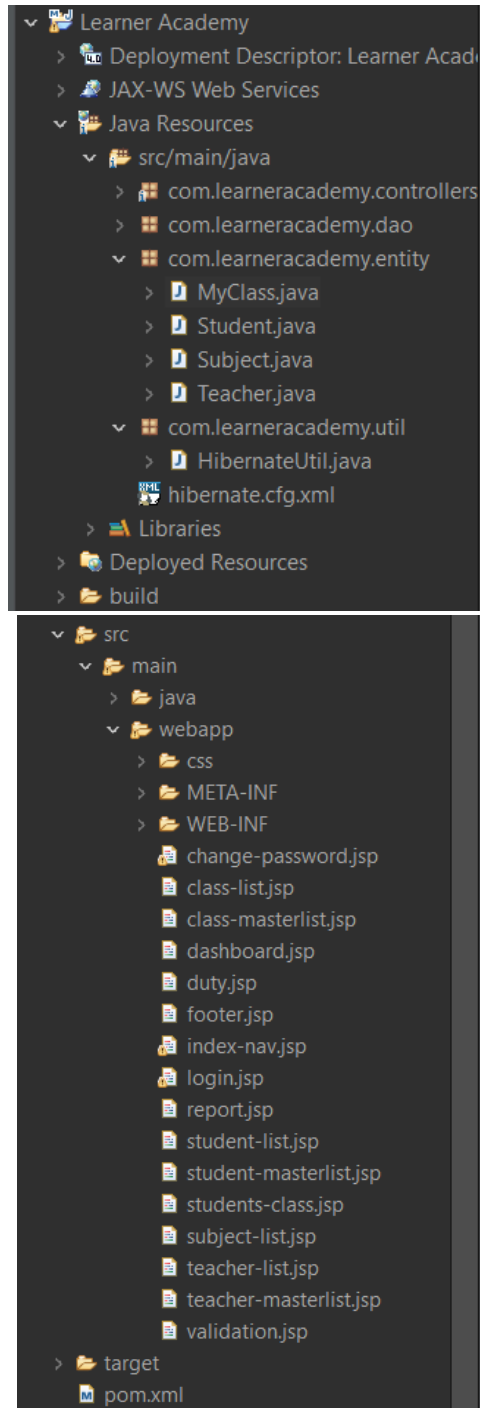
The client may request various services like:

- GET
- PUT
- UPDATE
- DELETE

The `service()` method takes responsibility to check the type of request received from the client and respond accordingly by generating a new thread or a set of threads per the requirement and implementing the operation through the following methods.

- `doGet()` for GET
- `doPut()` for PUT
- `doUpdate()` for UPDATE
- `doDelete()` for DELETE

Project Structure

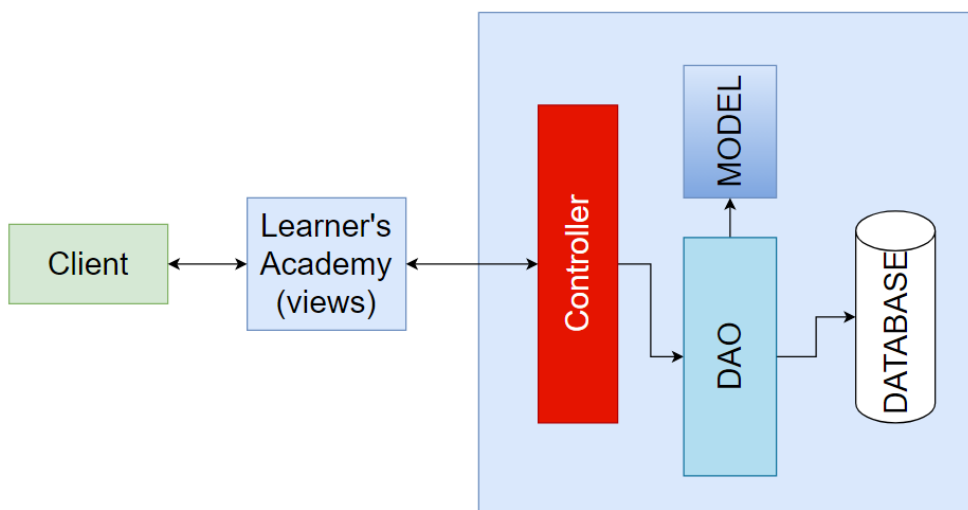


Control Statements

The program utilizes the following control statements to direct the desired logic:

- *while* loop – Controls the program flow by prompting the User for main menu and the business options sub-menu, performing the desired operations, and terminates when the User wishes to quite the program.
- *Switch* statement Executes the desired code statements associated with the main menu and the business level options sub-menu based on the value entered by the user.

Flow Chart



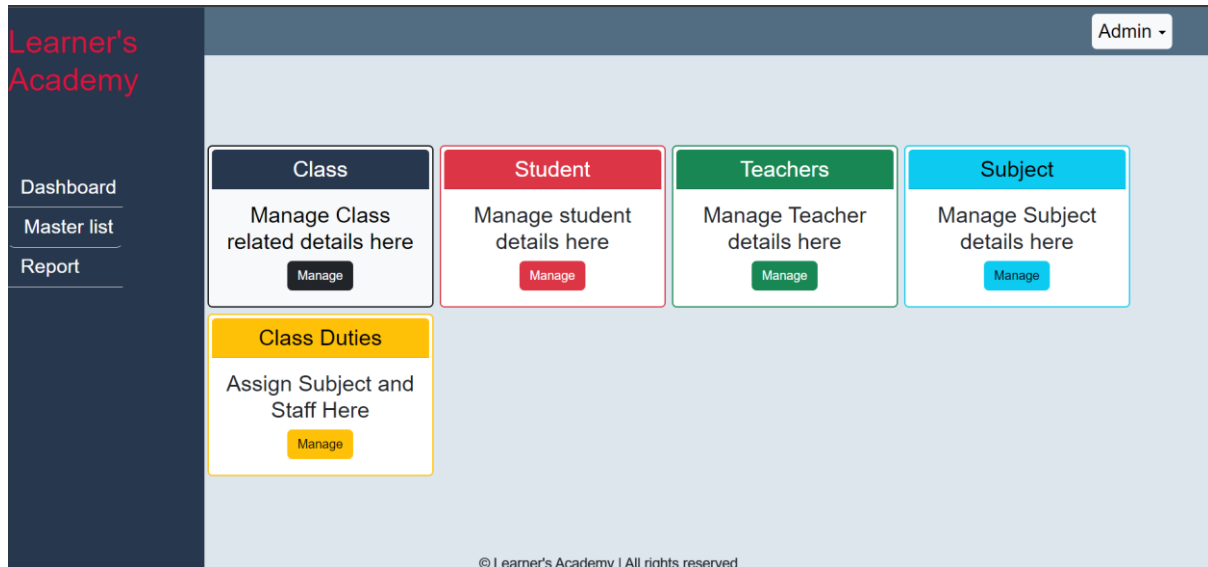
GitHub Repository

I have pushed my code and associated documentation to the following GitHub repository:

https://github.com/Thiru97/LearnersAcademy_Simplilearn.git

Web Application Screenshots

Home Page



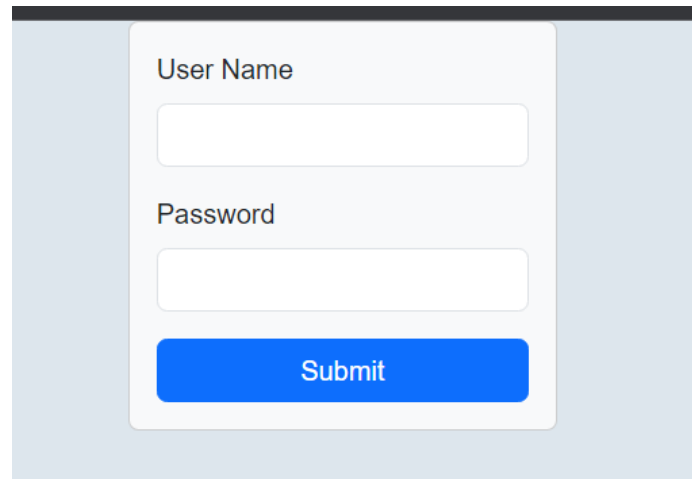
Admin can login by changing the url as localhost:8080/Learner's Academy/AdminController

Where he has to enter his credentials

Default Email Id: admin@gmail.com

Default Password: admin

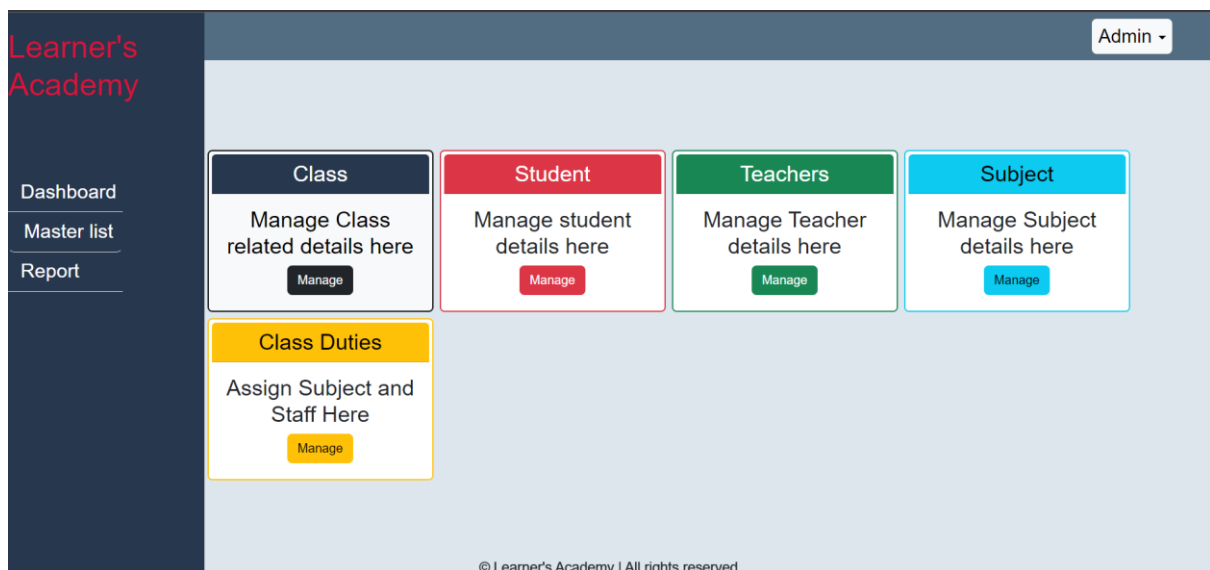
However admin should change his password after login



A login form with a light blue background. It contains two input fields: 'User Name' and 'Password'. Below the password field is a blue 'Submit' button.

Admin Dashboard

NOTE: If session is inactive for more than 5 minutes admin will be logged out



The Admin Dashboard interface features a dark blue sidebar on the left with the text 'Learner's Academy' in red. The sidebar contains three links: 'Dashboard', 'Master list', and 'Report'. The main content area has a top bar with 'Admin' and a dropdown arrow. Below this, there are four colored boxes: 'Class' (dark blue), 'Student' (red), 'Teachers' (green), and 'Subject' (light blue). Each box contains the text 'Manage [Category] related details here' and a 'Manage' button. Below these is a yellow box labeled 'Class Duties' with the text 'Assign Subject and Staff Here' and a 'Manage' button. At the bottom, there is a copyright notice: '© Learner's Academy | All rights reserved.'

Add Duty

Learner's Academy

Learner's Academy

Admin ▾

Duty Details

Assign Teacher

Class form ✕

Select Class:
Select

Select Subject:
Select

Select Teacher:
Select

Assign Cancel

Class	Teacher
Jane Doe	Jane Doe
Jane Doe	Jane Doe
Jane Doe	Jane Doe

© Learner's Academy | All rights reserved.

Master-list of all the subjects

Learner's Academy

Admin ▾

Subject Details

Add Subject

S No	Subject Name	Subject Code	Action
1	English	Eng	Delete
2	Maths	Mat	Delete

© Learner's Academy | All rights reserved.

Master-list of all the teachers

Learner's Academy

Learner's Academy Dashboard Master list • Class list • Teachers list • Student list Report	Admin ▾			
	Teacher's Details			
	S No	First Name	Last Name	Qualification
	1	Jane	Doe	M.A Physics
	2	John	Doe	M.A English
© Learner's Academy All rights reserved.				

Master-list of all the classes

Learner's Academy Dashboard Master list • Class list • Teachers list • Student list Report	Admin ▾	
	Class Details	
	S No	Class Name
	1	Class-I
	2	Class-II
© Learner's Academy All rights reserved.		

Master-list of all the students

Learner's Academy

Learner's Academy

Dashboard

Master list

Class list

Teachers list

Student list

Report

Admin ▾

Student Details

S No	Student First Name	Student Last Name
1	John	Doe
2	Harry	Potter
3	Ramesh	Simplilearn

© Learner's Academy | All rights reserved.

Class Report

Learner's Academy

Dashboard

Master list

Report

Admin ▾

Class Report

Class Name	Subject	Teacher	Student List
Class-I	English	Jane Doe	List
Class-I	Maths	Jane Doe	List
Class-II	English	Jane Doe	List

© Learner's Academy | All rights reserved.

Add/delete Class

Learner's Academy

Dashboard

Master list

Report

Admin ▾

Class Details

Add Class

S No	Class Name	Action
1	Class-I	Delete
2	Class-II	Delete

© Learner's Academy | All rights reserved.

Add/delete Student

Learner's Academy

Dashboard

Master list

Report

Admin ▾

Student Details

Add Student Assign Class

S No	Student First Name	Student Last Name	Class	Action
1	John	Doe	Class-I	Delete
2	Harry	Potter	Class-I	Delete
3	Ramesh	Simplilearn	Class-II	Delete

© Learner's Academy | All rights reserved.

Add/delete Teacher

Learner's Academy

[Dashboard](#)

[Master list](#)

[Report](#)

Admin ▾

Teacher's Details

Add Teacher

S No	First Name	Last Name	Qualification	Action
1	Jane	Doe	M.A Physics	Delete
2	John	Doe	M.A English	Delete

localhost:9000/LearnerAcademy/AdminController?action=dashboard

© Learner's Academy | All rights reserved.

Add/delete Subject

Learner's Academy

[Dashboard](#)

[Master list](#)

[Report](#)

Admin ▾

Subject Details

Add Subject

S No	Subject Name	Subject Code	Action
1	English	Eng	Delete
2	Maths	Mat	Delete

© Learner's Academy | All rights reserved.