

---

*Leaners Academy*

---

## **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, The administrator can:

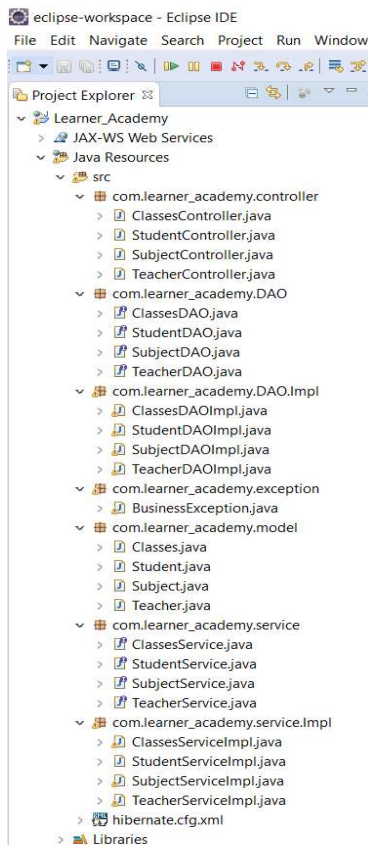
- Set up a master list of all the subjects for all the classes
  - Set up a master list of all the teachers
  - Set up a master list of all the classes
  - Assign classes for subjects from the master list
  - Assign teachers to a class for a subject (A teacher can be assigned to different classes for 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.

## **Tools used in the Project:**

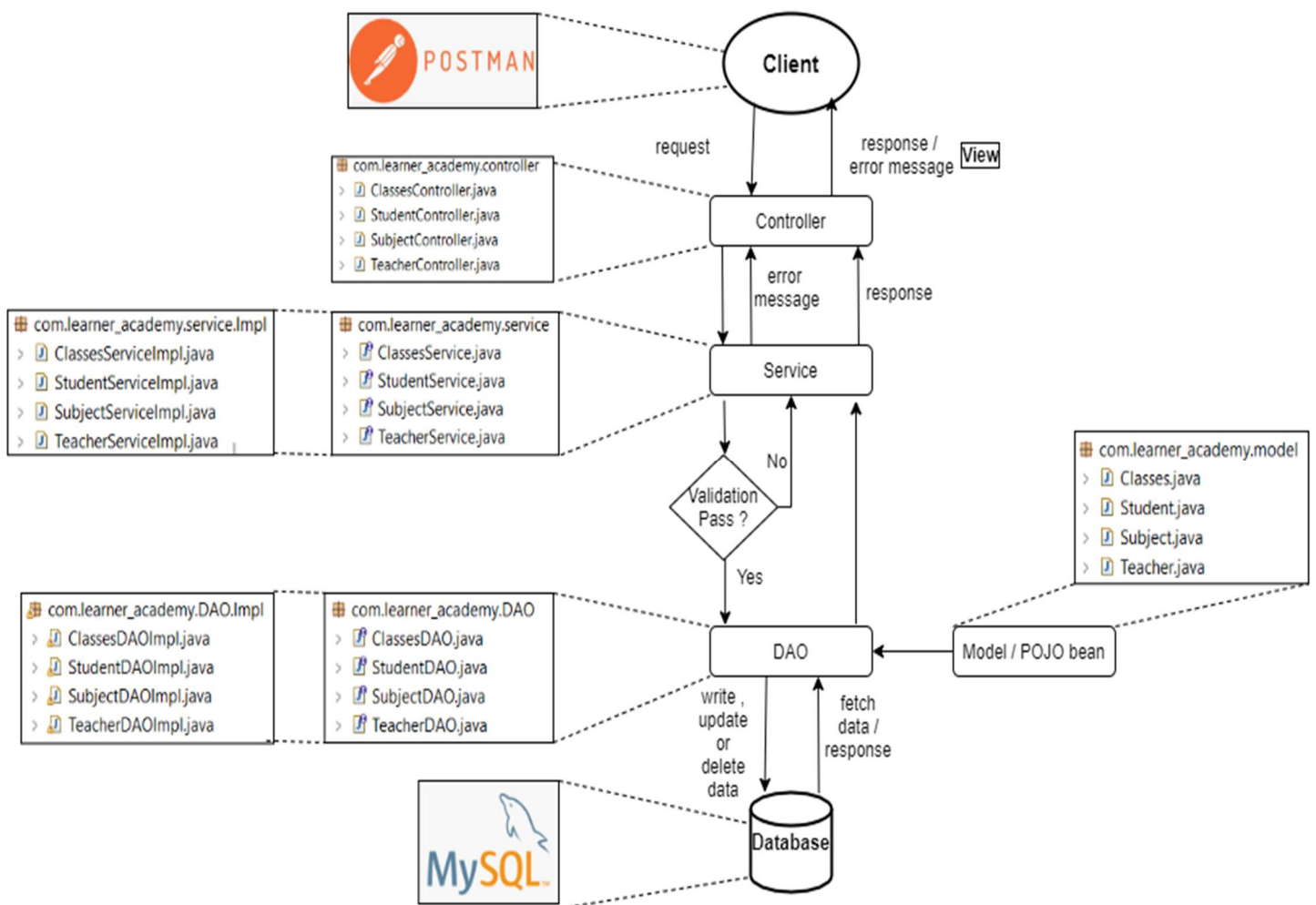
<b>Programming Language and related</b>	<b>Java, Hibernate, Jersey</b>
<b>IDE</b>	<b>Eclipse</b>
<b>SQL</b>	<b>MySQL</b>
<b>Version Control System</b>	<b>Git &amp; GitHub</b>
<b>Software Development Methodology</b>	<b>Agile - Scrum - Jira</b>
<b>API Client</b>	<b>POSTMAN</b>

## **Hierarchy of codes developed:**



- > JavaScript Resources
- > Referenced Libraries
- > build
- > WebContent
  - > META-INF
  - > WEB-INF
    - > lib
    - > web.xml
- > README.md

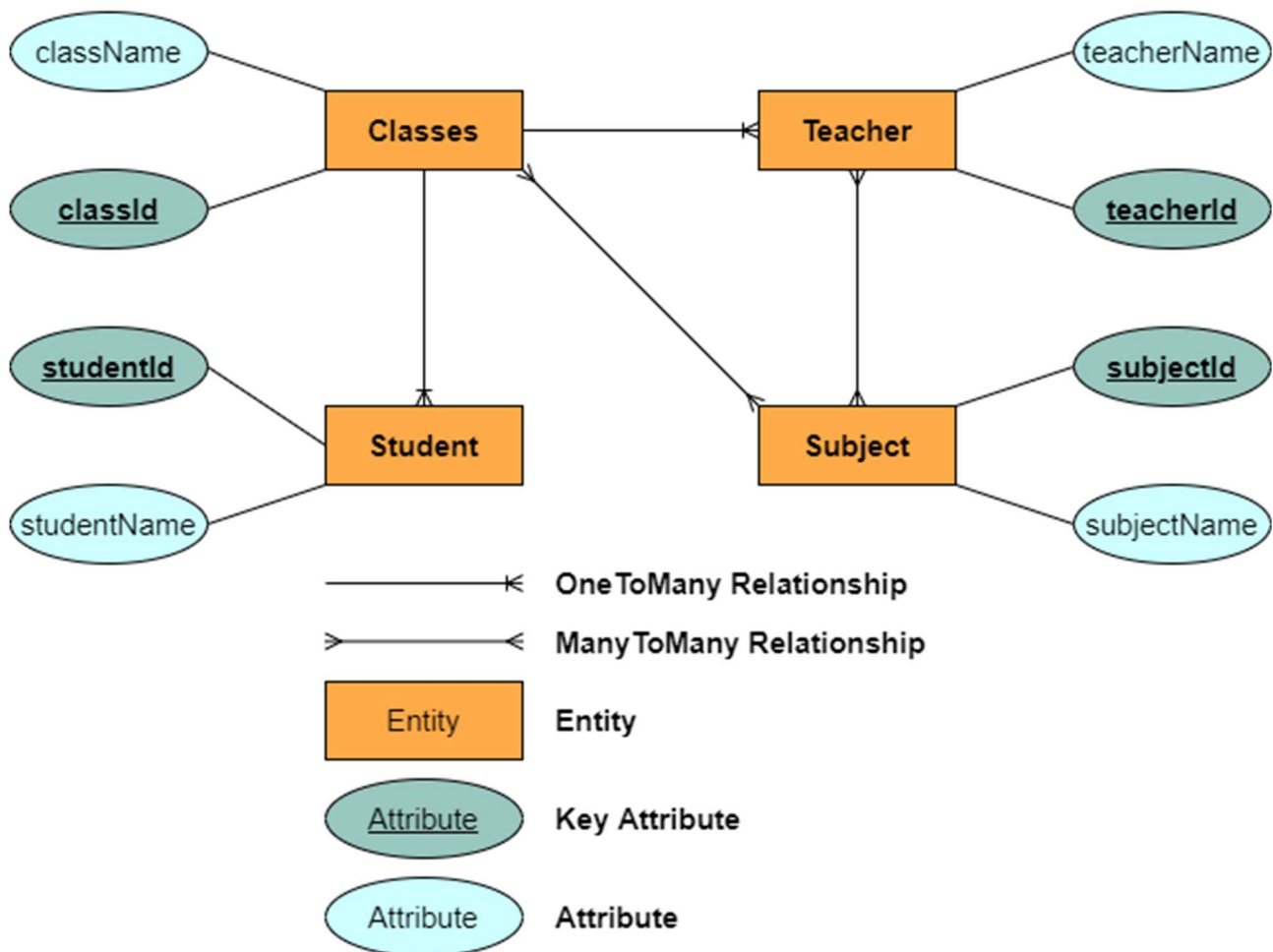
## Application Architecture and Flow Chart:



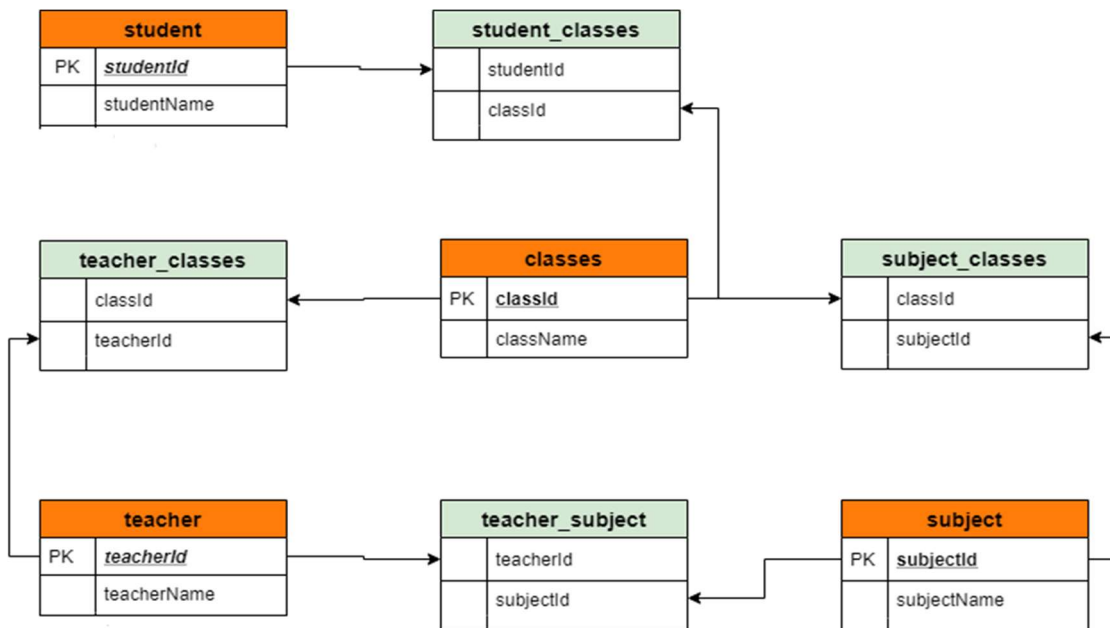
## Exception Handling:

Some of the exceptions and errors are handled, so that the application won't close, exit or throw an exception.

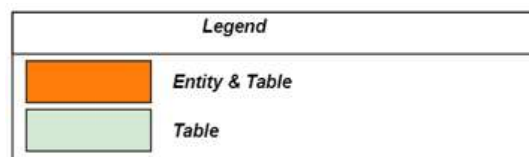
### Entity Relationship Diagram



## Entity and Tables:



Database Tables and Mapping



## Tables created in Database:

```
mysql> show tables;
+-----+
| Tables_in_db1 |
+-----+
| classes        |
| student        |
| student_classes|
| subject        |
| subject_classes|
| teacher        |
| teacher_classes|
| teacher_subject|
+-----+
8 rows in set (0.00 sec)
```

## ***Project Requirement and Verification in POSTMAN Tool:***

Project Requirement		Verifying in POSTMAN tool	
S.No.	Requirement	HTTP Method	Request URL
1	Set up a master list of all the subjects	GET	<a href="http://localhost:8080/Learner_Academy/Subject">http://localhost:8080/Learner_Academy/Subject</a>
2	Set up a master list of all the teachers	GET	<a href="http://localhost:8080/Learner_Academy/Teacher">http://localhost:8080/Learner_Academy/Teacher</a>
3	Set up a master list of all the classes	GET	<a href="http://localhost:8080/Learner_Academy/Classes">http://localhost:8080/Learner_Academy/Classes</a>
4	Assign subject for class	GET	<a href="http://localhost:8080/Learner_Academy/Classes">http://localhost:8080/Learner_Academy/Classes</a>
5	Assign teacher to subject	GET	<a href="http://localhost:8080/Learner_Academy/Classes">http://localhost:8080/Learner_Academy/Classes</a>
6	Get a master list of students	GET	<a href="http://localhost:8080/Learner_Academy/Student">http://localhost:8080/Learner_Academy/Student</a>
7	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	GET	<a href="http://localhost:8080/Learner_Academy/Classes">http://localhost:8080/Learner_Academy/Classes</a>

### ***Screenshots taken while executing project requirements and their verification:***

#### ***1) Set up a master list of all the subjects:***

The left screenshot shows a POST request in Postman to the URL `http://localhost:8080/Learner_Academy/Subject`. The request body is a JSON object: `{ "subjectName": "sub4" }`. The right screenshot shows a GET request to the same URL. The response body is a JSON array of four subjects: `[ { "subjectId": 1, "subjectName": "sub1" }, { "subjectId": 2, "subjectName": "sub2" }, { "subjectId": 3, "subjectName": "sub3" }, { "subjectId": 4, "subjectName": "sub4" } ]`.

## 2) Set up a master list of all the teachers:

POST http://localhost:8080/Learner\_Academy/Teacher

Params Authorization Headers (8) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2   "teacherName": "tea4"
3 }
```

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2   "subject": [],
3   "teacherId": 4,
4   "teacherName": "tea4"
5 }
```

GET http://localhost:8080/Learner\_Academy/Teacher

Params Authorization Headers (6) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL

This request

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON

```
1 [
2   {
3     "subject": [],
4     "teacherId": 1,
5     "teacherName": "tea1"
6   },
7   {
8     "subject": [],
9     "teacherId": 2,
10    "teacherName": "tea2"
11  },
12  {
13    "subject": [],
14    "teacherId": 3,
15    "teacherName": "tea3"
16  },
17  {
18    "subject": [],
19    "teacherId": 4,
20    "teacherName": "tea4"
21  }
22 ]
```

## 3) Set up a master list of all the classes:

POST http://localhost:8080/Learner\_Academy/Classes

Params Authorization Headers (8) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2   "className": "clas4"
3 }
```

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2   "classId": 4,
3   "className": "clas4",
4   "student": [],
5   "subject": [],
6   "teacher": []
7 }
```

GET http://localhost:8080/Learner\_Academy/Classes

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON

```
1 [
2   {
3     "classId": 1,
4     "className": "clas1",
5     "student": [],
6     "subject": [],
7     "teacher": []
8   },
9   {
10    "classId": 2,
11    "className": "clas2",
12    "student": [],
13    "subject": [],
14    "teacher": []
15  },
16  {
17    "classId": 3,
18    "className": "clas3",
19    "student": [],
20    "subject": [],
21    "teacher": []
22  },
23  {
24    "classId": 4,
25    "className": "clas4",
26    "student": [],
27    "subject": [],
28    "teacher": []
29  }
30 ]
```

## 4) Assign subject for class:

GET http://localhost:8080/Learner\_Academy/Classes/1

Params Authorization Headers (6) **Body** Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL

This request

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON

```

1 {
2   "classId": 1,
3   "className": "clas1",
4   "student": [],
5   "subject": [],
6   "teacher": []
7 }

```

PATCH http://localhost:8080/Learner\_Academy/Classes

Params Authorization Headers (8) **Body** Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```

1 {
2   "classId": 1,
3   "className": "clas1",
4   "student": [],
5   "subject": [
6     {
7       "subjectId": 1,
8       "subjectName": "sub1"
9     }
10  ],
11   "teacher": []
12 }

```

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON

```

1 {
2   "classId": 1,
3   "className": "clas1",
4   "student": [],
5   "subject": [
6     {
7       "subjectId": 1,
8       "subjectName": "sub1"
9     }
10  ],
11   "teacher": []
12 }

```

### 5) Assign teacher to subject:

GET http://localhost:8080/Learner\_Academy/Teacher/1

Params Authorization Headers (6) **Body** Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL

This request c

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON

```

1 {
2   "subject": [],
3   "teacherId": 1,
4   "teacherName": "teal"
5 }

```

PATCH http://localhost:8080/Learner\_Academy/Teacher

Params Authorization Headers (8) **Body** Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```

1 {
2   "subject": [
3     {
4       "subjectId": 1,
5       "subjectName": "sub1"
6     }
7   ],
8   "teacherId": 1,
9   "teacherName": "teal"
10 }

```

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON

```

1 {
2   "subject": [
3     {
4       "subjectId": 1,
5       "subjectName": "sub1"
6     }
7   ],
8   "teacherId": 1,
9   "teacherName": "teal"
10 }

```

### 6) Get a master list of students



POST
http://localhost:8080/Learner\_Academy/Student

Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings

none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON

```

1 {
2   "studentName": "stu4"
3 }

```

Body
Cookies
Headers (5)
Test Results

Pretty
Raw
Preview
Visualize
JSON

```

1 {
2   "studentId": 5,
3   "studentName": "stu4"
4 }

```

GET
http://localhost:8080/Learner\_Academy/Student

Params
Authorization
Headers (6)
Body
Pre-request Script
Tests
Settings

none
form-data
x-www-form-urlencoded
raw
binary
GraphQL

This request

Body
Cookies
Headers (5)
Test Results

Pretty
Raw
Preview
Visualize
JSON

```

1 [
2   {
3     "studentId": 1,
4     "studentName": "stu1"
5   },
6   {
7     "studentId": 2,
8     "studentName": "stu2"
9   },
10  {
11    "studentId": 3,
12    "studentName": "stu3"
13  },
14  {
15    "studentId": 5,
16    "studentName": "stu4"
17  }
18 ]

```

**7) 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:**

GET
http://localhost:8080/Learner\_Academy/Classes

Body
Cookies
Headers (5)
Test Results

Pretty
Raw
Preview
Visualize
JSON

```

1 [
2   {
3     "classId": 1,
4     "className": "clas1",
5     "student": [
6       {
7         "studentId": 1,
8         "studentName": "stu1"
9       }
10    ],
11    "subject": [
12      {
13        "subjectId": 1,
14        "subjectName": "sub1"
15      }
16    ],
17    "teacher": [
18      {
19        "subject": [],
20        "teacherId": 1,
21        "teacherName": "tea1"
22      }
23    ]
24  },
25  {
26    "classId": 2,
27    "className": "clas2",
28    "student": [
29      {
30        "studentId": 2,
31        "studentName": "stu2"

```

**Since the response Body for above requirement is lengthier, the output is copied to notepad and embedded here (also uploaded to the project's GitHub Repository)**



Class Report  
response JSON.txt

**Converted the above JSON response of Class Master Report to CSV for ease of viewing**

classId	className	student__studentId	student__studentName	subject__subjectId	subject__subjectName	teacher__subject__subjectId	teacher__subject__subjectName	teacher__teacherId	teacher__teacherName
1	clas1	1	stu1	1	sub1	1	sub1	1	tea1
						2	sub2		
						4	sub4		
2	clas2	2	stu2	2	sub2			2	tea2
3	clas3	3	stu3	3	sub3	3	sub3	3	tea3
4	clas4	5	stu4	4	sub4			4	tea4