Table of Contents

[Flow Diagram: 2](#_Toc48234368)

[Components: 2](#_Toc48234369)

[a) Controller: 2](#_Toc48234370)

[b) Services: 3](#_Toc48234371)

[c) Service interface : 3](#_Toc48234372)

[d) Dao: 3](#_Toc48234373)

[e) Dao Interface: 3](#_Toc48234374)

[f) Data Model: 3](#_Toc48234375)

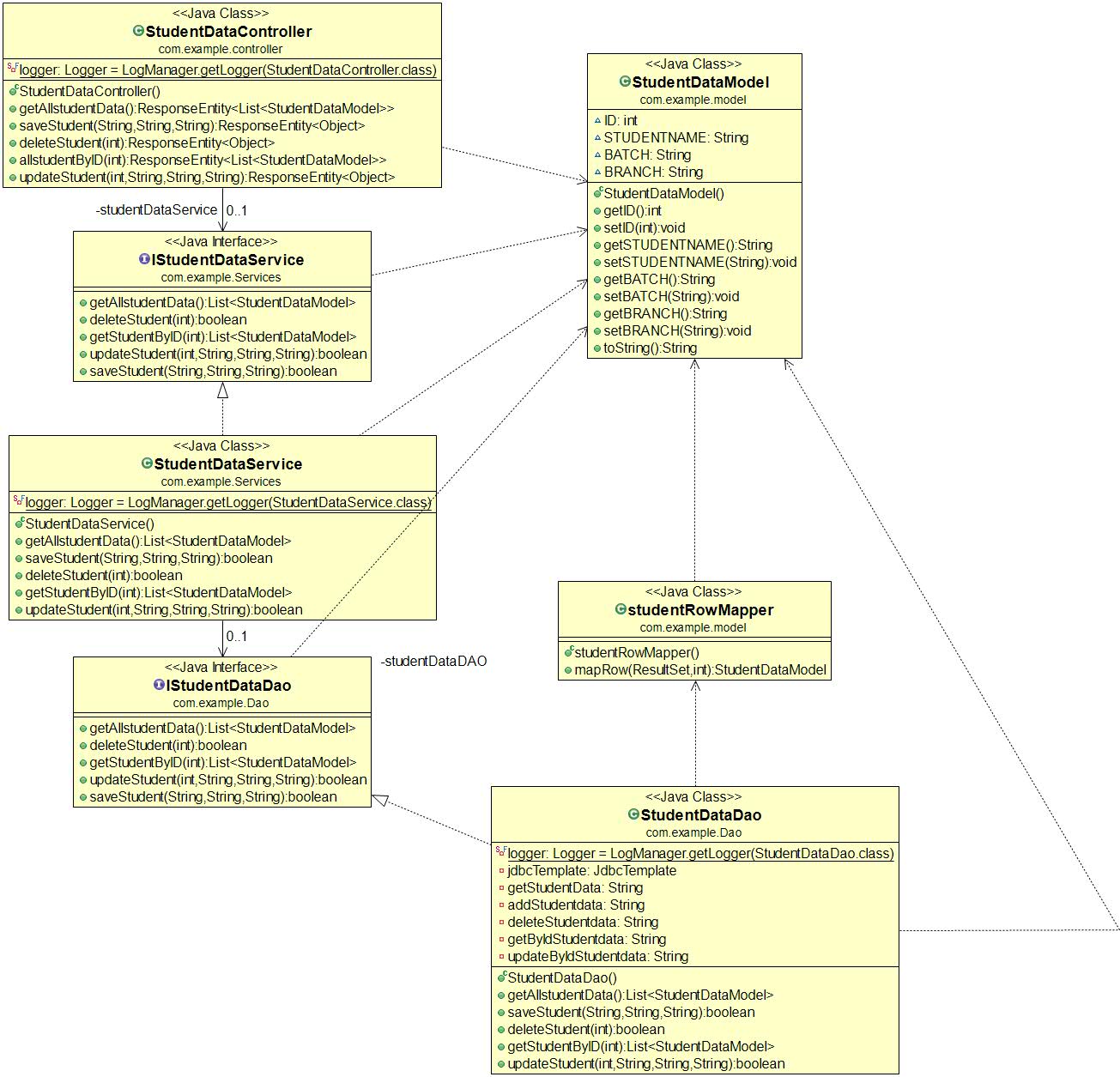
[g) Data Model Mapper: 3](#_Toc48234376)

[Approach towards Solution: 4](#_Toc48234377)

[Enhancement suggestion: 4](#_Toc48234378)

[Project details 6](#_Toc48234379)

# Flow Diagram:



## Components:

The components present in the microservice are:

* 1. Controller
  2. Service
  3. Service Interface
  4. Dao
  5. Dao Interface
  6. Data Model
  7. Data Model Mapper

### Controller:

The SutdentDataController consists of total of five microservices

* user/getstudentdata : This microservice fetches all rows from the table studentdata.
* user/save-student/{studentname}/{branch}/{batch} : This microservice inserts student name, branch and batch into the table student data where the id is auto incremented.
* user/delete-student/{id} : This microservice deletes the row where the specific student id is provided
* user/student/{id} : This microservice fetches specific user data for the given student id.
* user/update-student/{id}/{studentname}/{branch}/{batch} : This microservice updates the student data of the specific student id.

### Services:

The corresponding services are called interfaced with IServices to call the dao layer to fetch,insert,delete or update student data table.

### Service interface :

This is a simple interface for the services.

### Dao:

The dao layer for the corresponding microservices call the sqls configured exterally in application.properties through annotations.

* user/getstudentdata : *SELECT ID,STUDENTNAME,BATCH,BRANCH FROM Student*
* user/save-student/{studentname}/{branch}/{batch} : *INSERT INTO Student(studentname,branch,batch) VALUES (?,?,?)*
* user/delete-student/{id} : *Delete from Student where id = ?*
* user/student/{id} : *SELECT ID,STUDENTNAME,BRANCH,BATCH FROM Student where id = ?*
* user/update-student/{id}/{studentname}/{branch}/{batch} : *UPDATE Student SET STUDENTNAME = ?, BRANCH = ?,BATCH = ? WHERE ID = ?*

### Dao Interface:

This is a simple interface for the dao layer

### Data Model:

The data model consists of total of four entities:

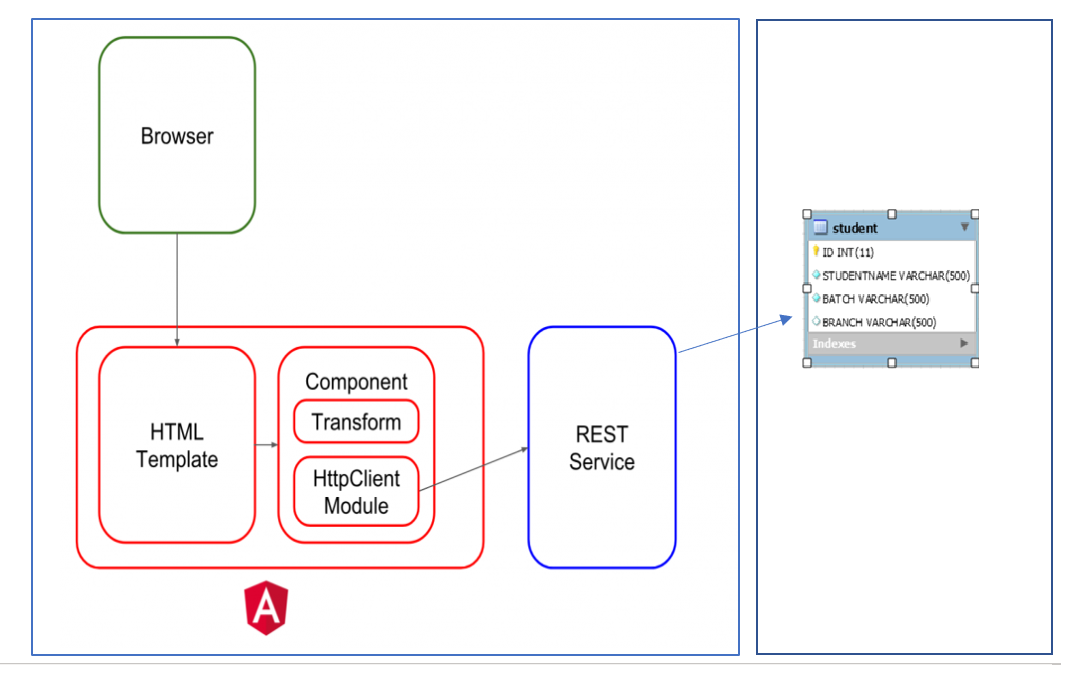
* ID which is of type integer
* STUDENTNAME which is of type String
* BATCH which is of type String
* BRANCH which is of type String

### Data Model Mapper:

This component gets and sets the data fetched into the model.

# Approach towards Solution:

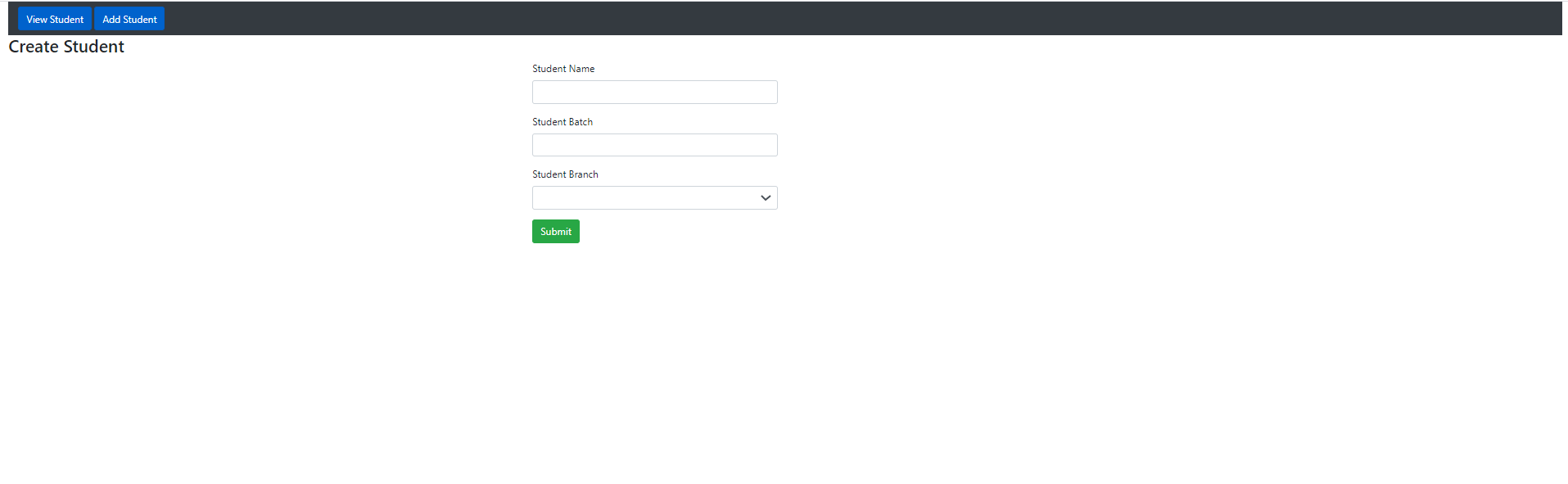
The approach taken towards the solution is Spring boot rest API microservices and MySql as data base. And for the cloud native solution, docker-ce and docker-compose is used to deploy the final jar.

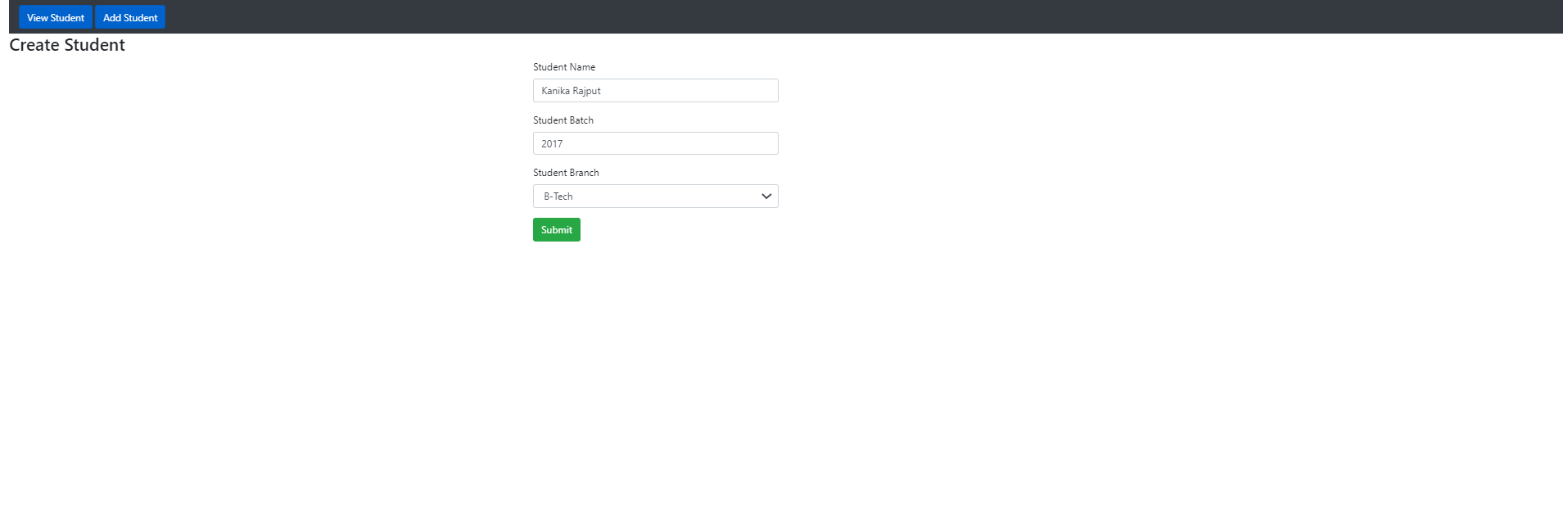


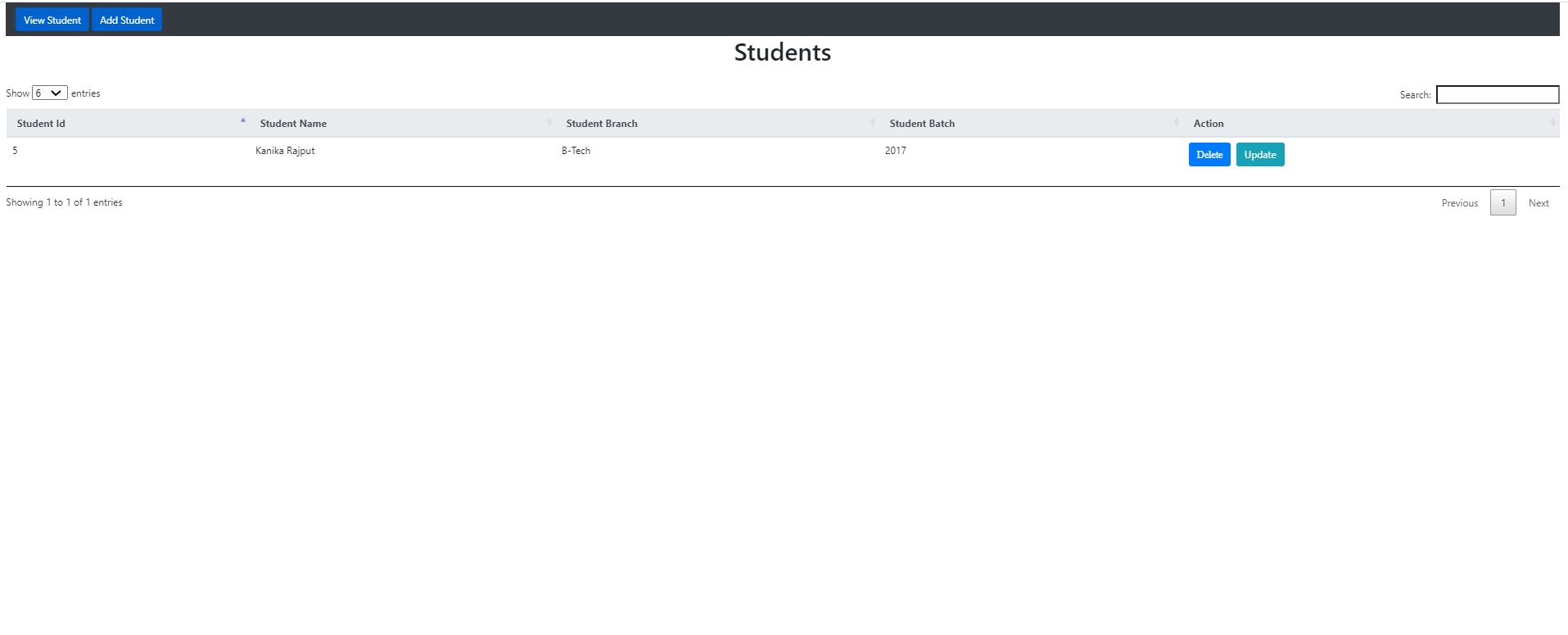
Docker image for Spring Boot Microservices Docker image of MySql and Angular as front end

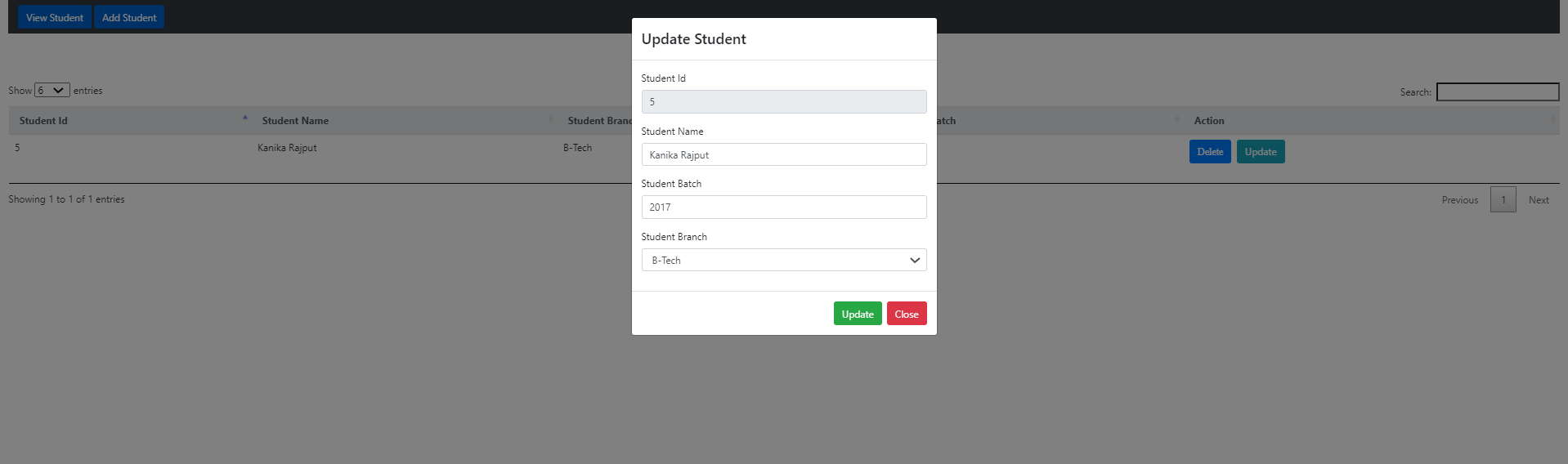
# Enhancement suggestion:

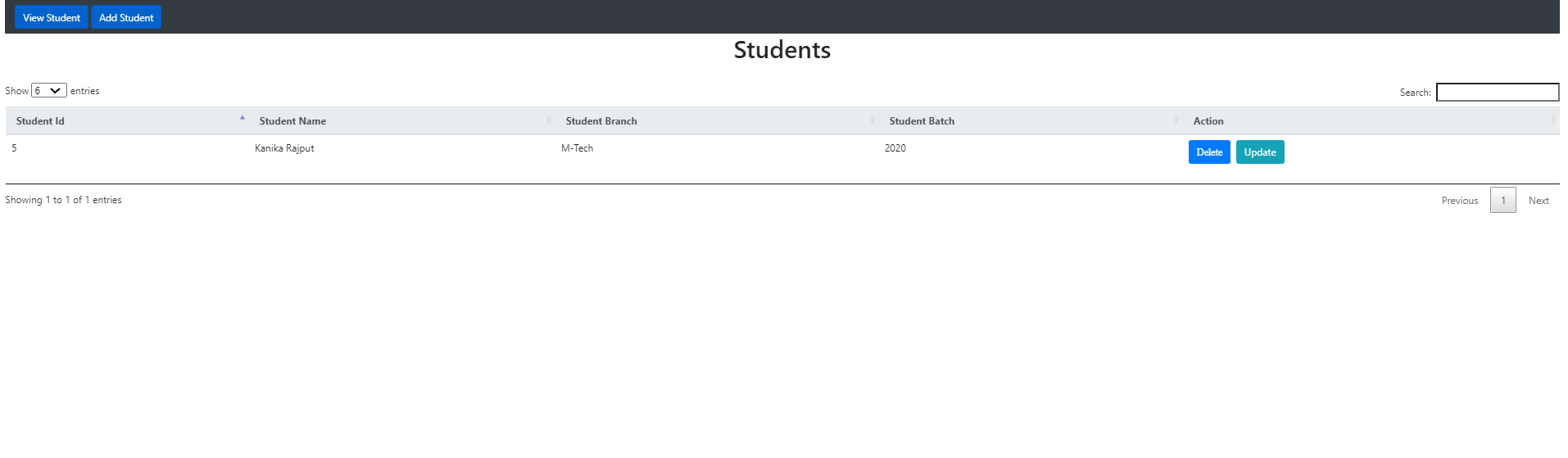
1. The solution can have a front end for the backend microservices , So a two page angular front end is created to interact with the backend services also it is compatible with desktop and mobile view. Some screen shots of the web application is attached.











1. The cloud native solution could be done through dockerfile keeping the jar in the location and with docker-compose.yaml. the solution that is chosen is to push the created microservice application image in docker hub and deploy the overall solution with only one docker compose file below is the yaml file

version: '3'

services:

mysql:

restart: always

container\_name: docker-mysql

image: mysql

environment:

MYSQL\_DATABASE: studentdata

MYSQL\_ROOT\_PASSWORD: root

volumes:

- ./sql:/docker-entrypoint-initdb.d

ports:

- "3306:3306"

healthcheck:

test: "/usr/bin/mysql --user=root --password=root --execute \"SHOW DATABASES;\""

interval: 2s

timeout: 20s

retries: 1

spring\_app:

restart: on-failure

container\_name: docker-student\_register

image: loveforever123/springapp:v1

expose:

- "2703"

ports:

- 2703:2703

environment:

WAIT\_HOSTS: mysql:3306

depends\_on:

- mysql

With this only two containers spring app and mysql is created with schema student data and is run in port 3306 and the application is run in port 2703.

# Project details

Created by: Md Dilshad Ali

Cloud used: AWS

College: Calcutta Institute Of Engineering & Management

TCS\_ION Batch\_id: Batch1

Email\_id: mddilshadali2410@gmail.com

Mobile no. : +918420376461