**Table of Contents**

[**1.0Introduction3**](bookmark://_Toc157835546)

[**1.1Purpose of this document3**](bookmark://_Toc157835547)

[**1.2Scope3**](bookmark://_Toc157835548)

[**1.3Intended Audience3**](bookmark://_Toc157835549)

[**2.0Conventions and Standards Followed4**](bookmark://_Toc157835552)

[**4.0Use Case Realization6**](bookmark://_Toc157835554)

[**5.0Package & Sub-system Design8**](bookmark://_Toc157835557)

[**6.0Database Design**](bookmark://_Toc157835566)

1. Page Break**Introduction**

Learn-online web Restful web services software is an assignment for Performance Improvement Plan here as referred as PIP.

1. **Purpose of this document**

The purpose of this document is to provide architecture, design and technologies that are used in the development of this web application.

1. **Breif introduction of restful web with respect to technology and domain**

Students can sign-up, update his profile, purchase courses and cancelled purchased courses search the online courses or get information of all available courses. searched information will be presented by sort order grouped by domain and rating.

The medium of communication will be JSON object. We have provided basic JSP pages for communication with application but this application is decoupled with any particular view technology. It is restful web service that expose HTTP restful endpoint that interact in JSON mediatype. Any application can be used to interact with it such as JSP pages, Postman, SoupUI, Swagger and etc.

I have used JDK 8, Spring Boot, JUNIT5, Mockito, Log4j2, JPA and MySQL for developing this application. In following PDF provide generalized application diagram in pdf please find it

Happy Path



Exception Occurred



1. **Scope**

This documents covers following things of project   
1. Architecture and designing of rest full web service.

2. Briefing on Business requirement.  
3. What Technologies are used in development of this project.

1. **Intended Audience**

This document is written to introduce Learn-Online restful web application to PIP team with respect its business functionality, technologies that are used to develop application and design and architecture of restful web application.

1. **Conventions and Standards Followed**

We have used meaningful name for component, packages, interfaces, classes, properties, methods, Utilities library, variables, constants and rest endpoint URLS. We followed camel case naming convention for all component other URLS. We have standardized request and response classes and defined our own standardized runtime exceptions. We have used Controller advice as standard to handle exception raised anywhere in restful web application such as in controller, service layer, dao layer or any of utilities classes. Controller advice handles these exceptions, create proper response and send it back to client.

1. **Use Case Realization**

In the following sequence diagrams, we provided sequence diagrams for ervices that are welcome data, student signup, update operation, purchasing courses and cancelling purchased courses operation. The medium of communication is JSON data format. The client can communicate with different mediums like JSP pages, Postman, SoapUI, Swagger and etc. In the following pdf files of sequence diagrams are provided.

**A:-** **Welcome Data (HTTP GET request JSON Format)**  
Student get all data that of courses grouped and sorted by domain and Rating.



**B:-** **Successful Student Creation (HTTP POST Request JSON Format)**  
 Student send request.



* 1. Input data is validated using JSR 303 validation annotation and custom defined annotation.
  2. Service layer uses DAO layer and checks if this sign-up is duplicate.   
     Email will be criteria.
  3. Service layer calls DAO layer to persist student detail.
  4. On success Student-Mgmt-Controller send response

**C:- Student creation failed. Student with same email already exists**   
 (**HTTP POST Request JSON Format**)



* 1. Input data is validated using JSR 303 validation annotation and custom defined annotation.
  2. Service layer uses DAO layer and checks if this sign-up is duplicate.
  3. Email will be criteria.
  4. Service layer throws app defined runtime application due to duplicate sign up.
  5. On failure Student Advice controller send response by handling Application runtime exception.

**D:- Successful Student Update request.** (**HTTP PUT Request JSON Format**)



* 1. Input data is validated using JSR 303 validation annotation and custom defined annotation.
  2. Service layer uses DAO layer and checks if this student details exists.   
     Email will be criteria.
  3. Service layer calls DAO layer to persist updated student detail.
  4. On success Student-Mgmt-Controller send response

**E. Student Update Request failed because student by email not found.** (**HTTP PUT Request JSON Format**)

****

* 1. Input data is validated using JSR 303 validation annotation and custom defined annotation.
  2. Service layer uses DAO layer to check if this student details exist in DB.   
     Email will be criteria.
  3. Service layer uses DAO layer and does not find student details in DB hence its throw application defined runtime exception.
  4. On failure Student Advice controller send response by handling Application runtime exception.

**F:- Student successfully Purchased Courses**(**HTTP POST Request JSON Format**)



* 1. Input data is validated using JSR 303 validation annotation and custom defined annotation.
  2. Service layer uses DAO layer and checks if this student details exists.   
     Student key-codes will be criteria.
  3. Service layer uses DAO layer and checks if these courses exist in DB to purchase. Courses key-codes will be criteria.
  4. Service layer checks whether student is buying duplicate courses by mistake.
  5. Service layer perform courses purchase operation and after that it calls DAO layer to persist updated student detail.
  6. On success Student-Mgmt-Controller send response

**G:- Student is failed to purchase courses (HTTP POST Request JSON Format)**

****

* 1. Input data is validated using JSR 303 validation annotation and custom defined annotation.
  2. Service layer uses DAO layer and checks if this student details exists.   
     Student key-codes will be criteria.
  3. Service layer uses DAO layer and checks if these courses exist in DB to purchase. Courses key-codes will be criteria. Service layer uses DAO layer and finds that courses details are not available in DB.
  4. Service layer throws application defined runtime exception because requested courses to be purchased are not available in DB.
  5. On failure Student Advice controller send response by handling Application runtime exception.

**H. Student successfully cancelled the purchased courses (HTTP POST Request JSON Format)**

****

* 1. Input data is validated using JSR 303 validation annotation and custom defined annotation.
  2. Service layer uses DAO layer and checks if this student details exists.   
     Student key-codes will be criteria.
  3. Service layer uses DAO layer and checks if these purchased courses going to be cancelled exist in DB for given student. Courses key-codes will be criteria.   
     Then Service layer also checks whether these courses   
     Date of purchasing exceeds 30 days from current date.   
     If no then Student cancellation request for courses is valid.
  4. Service layer checks whether student is cancelling already cancelled courses by mistake.
  5. Service layer perform courses cancellation operation and after that it calls DAO layer to persist updated student detail.
  6. On success Student-Mgmt-Controller send response

**I:- Student failed to cancel the purchased courses (HTTP POST Request JSON Format)**

****

* 1. Input data is validated using JSR 303 validation annotation and custom defined annotation.
  2. Service layer uses DAO layer and checks if this student details exists.   
     Student key-codes will be criteria.
  3. Service layer uses DAO layer and checks if these purchased courses going to be cancelled exist in DB for given student. Courses key-codes will be criteria.   
     Then Service layer also checks whether these courses   
     Date of purchasing exceeds 30 days from current date.   
     If ye then Student cancellation request for courses is invalid.
  4. Service layer checks whether student is cancelling already cancelled courses by mistake.
  5. Service layer throws application defined runtime exception because

Either date of purchasing of courses has crossed 30 days from current date or student provided invalid courses entries for cancellation.

* 1. On failure Student Advice controller send response by handling Application runtime exception.

1. **Package & Sub-system Design**

In the following project skeleton, packages, classes, interfaces, app defined expception are given,utilities and enums.

**Folder Name: src/man/java**

Packge Name:- com.learn.online

Desciption:- Spring boot startup application

Package Name:- com.learn.online.beans

Description:- It contains JPA POJO persistence entity classes

Contents:-

* + 1. CourseEntity.java
    2. CourseOrderEntity.java
    3. StudentEntity.java

Package Name:- com.learn.online.controllers

Description:- It contains Student Mgmt controller that handles incoming request for students

Conentents:-

StudentMgmtController.java  
 welcome()

searchByEmail()

createStudent()

updateStudent()

buyCourse()

deleteCourses()

searchCoursesByDomainAndRating()

searchCoursesByDomain()

Package Name:- com.learn.online.custom.validation.annotations

Description:- It contains annotation based validation for cross field value checking

FieldMatch.java

FieldMatchValidator.java

PasswordConstraintValidator.java

ValidPassword.java

Package Name:- com.learn.online.daos

Description:- Contains JPA dao interfaces extends from JpaRepository

CourseEntityDao.java

CourseOrderEntityDao.java

StudentEntityDao.java

Package Name:- com.learn.online.dtos

Description:- Data transfer objects for respective JPA entities classes

CourseDto.java

CourseOrderDto.java

StudentDto.java

Package Name:- com.learn.online.enums

Description:- Enums for constants values and messages

CourseNotFoundtException.java

CourseServiceException.java

LearnOnLineException.java

StudentServiceException.java

Package Name:- com.learn.online.exceptions

Description:- Application Defined Runtime application

InputValidationHandler.java

LearnOnlineExceptionHandler.java

Package Name:- com.learn.online.exceptions.handlers

Description:- Controller Advice class for handling exception and HTTP sending response

InputValidationHandler.java

LearnOnlineExceptionHandler.java

Package Name:- com.learn.online.requests

Description:- Controller handler methods receive incoming request JSON input converted these class object.

BuyOrCancelCouresesRequest.java

StudentSignupRequest.java

StudentUpdateRequest.java

Package Name:- com.learn.online.responses

Description:- Controller handler methods send response into these object.

CourseOrderResponse.java

CoursesResponse.java

ErrorMessageResponse.java

LearnOnlineResponse.java

StudentDetailResponse.java

StudentResponse.java

Package Name:- com.learn.online.services and com.learn.online.services.impls

Description:- Service layers business interfaces

CourseService.java

StudentService.java

Package Name:- com.learn.online.services.impls

Description:- Service implementation layers containing business and business logic

CoursesServiceImpl.java

StudentServiceImpl.java

Package Name:- com.learn.online.utils

Description:-Classes that contains utilities methods used in this application

CustomUtils.java

URLConstants.java

**Directory**

**src/main/resources**

application.properties

messages.properties

**Direectory**

**src/test/java**

Package Name:- com.learn.online.controllers

Description:- UNIT test case 5 with mockito for controller.

* + 1. StudentMgmtControllerKURDTest.java

@Test

testCreateStudent()

@Test

updateStudentTest()

@Test

purchaseCourseTest()

@Test

cancelPurchaseCourseTest()

* + 1. StudentMgmtControllerSearchTest.java @Test

testWelcome()

@Test

searchCoursesByDomainAndRating()

@Test

searchCoursesByDomain()

@Test

testSearchByEmail()

Package Name:- com.learn.online.services.impls

Description:- UNIT test case 5 with mockito for Service layer

* + 1. StudentServiceKURDImplTest.java

@Test

signupTest()

@Test

signupFailedTest()

@Test

updateTest()

@Test

updateFailedTest()

@Test

RequestedCoursestoBuyDoNotExistsTest()

@Test

duplicatePurchaseCoursesTest()

@Test

purchaseCoursesTest()

@Test

cancePurcahsedCoursesNotFoundForDeletionTest()

@Test

purchaseCoursesStudentNotFoundTest()

@Test

cancelPurchaseCoursesTest()

@Test

canNotcancelPurchaseCoursesDatesExccedTest()

@Test

cancelPurchaseCoursesStudentNotFoundTest()

@Test

cancelPurchaseEmptyCourseListTest()

@Test

searchStudentByEmailId()

@Test

searchStudentByEmailIdNotFound()

Package Name:- com.learn.online.dummies

Description:- Dummy data and helper methods for test cases

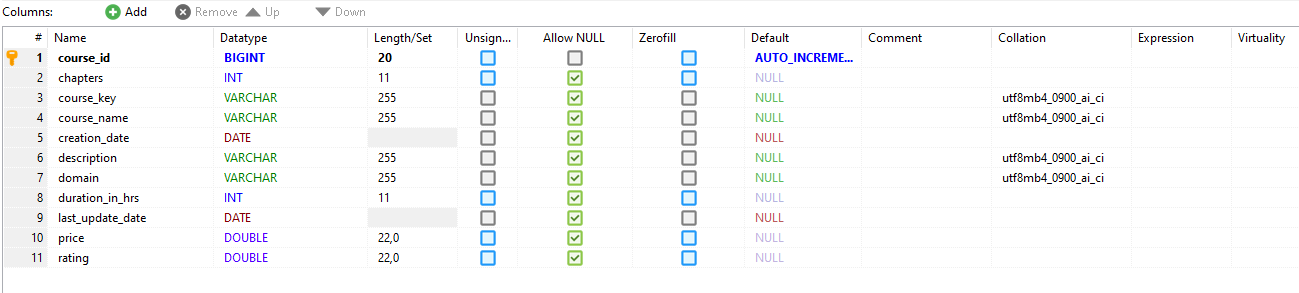
1. **Database Design**

There are 3 tables. Courses, Students and course\_orders. Courses will contains details of courses offered by application. Student table contains details of signed up students and courses\_orders table contains the transaction details of student regarding purchased courses. If student cancel the course then those records will be deleted from this table. cours\_orders has associate with courses and student table.  
Please find the ERP diagram of DB in the following



Student and course\_orders have one to many relationship with cascading of save/update and delete.

courses table



course table create script

CREATE TABLE `courses` (

`course\_id` BIGINT(20) NOT NULL AUTO\_INCREMENT,

`chapters` INT(11) NULL DEFAULT NULL,

`course\_key` VARCHAR(255) NULL DEFAULT NULL COLLATE 'utf8mb4\_0900\_ai\_ci',

`course\_name` VARCHAR(255) NULL DEFAULT NULL COLLATE 'utf8mb4\_0900\_ai\_ci',

`creation\_date` DATE NULL DEFAULT NULL,

`description` VARCHAR(255) NULL DEFAULT NULL COLLATE 'utf8mb4\_0900\_ai\_ci',

`domain` VARCHAR(255) NULL DEFAULT NULL COLLATE 'utf8mb4\_0900\_ai\_ci',

`duration\_in\_hrs` INT(11) NULL DEFAULT NULL,

`last\_update\_date` DATE NULL DEFAULT NULL,

`price` DOUBLE(22,0) NULL DEFAULT NULL,

`rating` DOUBLE(22,0) NULL DEFAULT NULL,

PRIMARY KEY (`course\_id`) USING BTREE

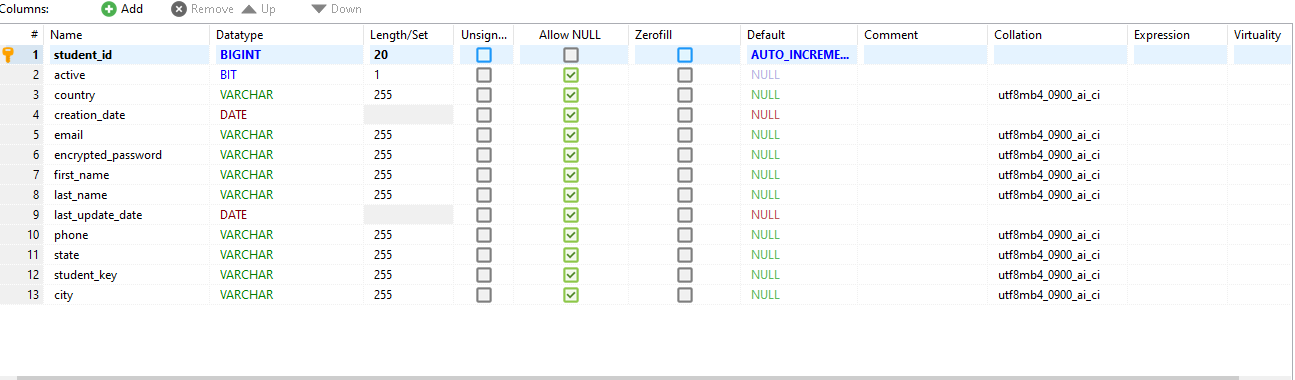
)

COLLATE='utf8mb4\_0900\_ai\_ci'

ENGINE=MyISAM

AUTO\_INCREMENT=9;

Student Table:



Student Table Create script

CREATE TABLE `students` (

`student\_id` BIGINT(20) NOT NULL AUTO\_INCREMENT,

`active` BIT(1) NULL DEFAULT NULL,

`country` VARCHAR(255) NULL DEFAULT NULL COLLATE 'utf8mb4\_0900\_ai\_ci',

`creation\_date` DATE NULL DEFAULT NULL,

`email` VARCHAR(255) NULL DEFAULT NULL COLLATE 'utf8mb4\_0900\_ai\_ci',

`encrypted\_password` VARCHAR(255) NULL DEFAULT NULL COLLATE 'utf8mb4\_0900\_ai\_ci',

`first\_name` VARCHAR(255) NULL DEFAULT NULL COLLATE 'utf8mb4\_0900\_ai\_ci',

`last\_name` VARCHAR(255) NULL DEFAULT NULL COLLATE 'utf8mb4\_0900\_ai\_ci',

`last\_update\_date` DATE NULL DEFAULT NULL,

`phone` VARCHAR(255) NULL DEFAULT NULL COLLATE 'utf8mb4\_0900\_ai\_ci',

`state` VARCHAR(255) NULL DEFAULT NULL COLLATE 'utf8mb4\_0900\_ai\_ci',

`student\_key` VARCHAR(255) NULL DEFAULT NULL COLLATE 'utf8mb4\_0900\_ai\_ci',

`city` VARCHAR(255) NULL DEFAULT NULL COLLATE 'utf8mb4\_0900\_ai\_ci',

PRIMARY KEY (`student\_id`) USING BTREE

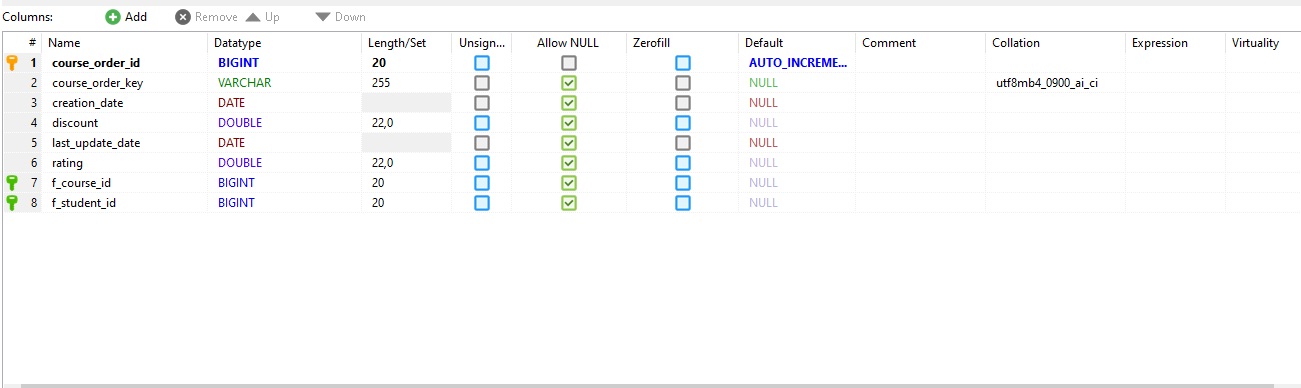
)

COLLATE='utf8mb4\_0900\_ai\_ci'

ENGINE=MyISAM

AUTO\_INCREMENT=27;

course\_orders



Course\_orders table create script

CREATE TABLE `course\_orders` (

`course\_order\_id` BIGINT(20) NOT NULL AUTO\_INCREMENT,

`course\_order\_key` VARCHAR(255) NULL DEFAULT NULL COLLATE 'utf8mb4\_0900\_ai\_ci',

`creation\_date` DATE NULL DEFAULT NULL,

`discount` DOUBLE(22,0) NULL DEFAULT NULL,

`last\_update\_date` DATE NULL DEFAULT NULL,

`rating` DOUBLE(22,0) NULL DEFAULT NULL,

`f\_course\_id` BIGINT(20) NULL DEFAULT NULL,

`f\_student\_id` BIGINT(20) NULL DEFAULT NULL,

PRIMARY KEY (`course\_order\_id`) USING BTREE,

INDEX `FK3640bo8wnwrxe5egh6kriu5yo` (`f\_course\_id`) USING BTREE,

INDEX `FK1m1dpg8d7vduvquhrbqwdhnm0` (`f\_student\_id`) USING BTREE

)

COLLATE='utf8mb4\_0900\_ai\_ci'

ENGINE=MyISAM

AUTO\_INCREMENT=30;