Student-Course Management TLDS

Student-Course Management Version: 1.0

Document Version: 1.0

NOTE: The hard copy version of this document is FOR REFERENCE ONLY. The online version is the master.
It is the responsibility of the user to ensure that they have the current version.
Any outdated hard copy is invalid and must be removed from possible use.
It is also the responsibility of the user to ensure the completeness of this document prior to use.

OWNER: The owner of a Top Level Design Specification is expected to be in a role as Chief Architect and/or Lead Programmer.

	E6998 Microservice Application and API Dev. **45 Tiemann Place, Apt 1G**
???	**929-287-8549**
???	**somdeep.dev@columbia.edu**

Issue Date:	10-08-2015
Owner:	Somdeep Dey
Authors:	Somdeep Dey(sd2988) Ranjith Kumar Shanmuga Visvanathan(rs3579) Harshit Saxena(hs2873) Harish Karthikeyan(hk2854)

Table of Contents

1. Document Control

1.1 Document Master Location

Filename:	**TLDS 1.0**
Document Location	

1.2 Document Revision History

The table below contains the summary of changes:

Version	Date Changed	Completed By	Description of changes
1.0	10-08-2015	Somdeep Dey	Initial Version

1.3 Approvers

The table below contains the record of approver, or delegate, signoff:

Approver Name	Approver Title	Version	Date Approved

1.4 Reviewers

The table below contains the record of reviewers:

Reviewer Name	Reviewer Title	Version	Date Reviewed

2. General Overview

2.1 Summary of Capabilities

- Maintain Student records, with CRUD features
- Maintain Course records, with CRUD features
- Maintain association between students and courses
- Setup a request router that will map public to private URIs (the corresponding microservice).
- Micro service to maintain referential integrity between student record and course record.
- Micro service to auto partition or repartition the students' data.
- Configuration APIs for the micro services created.

2.3 User Stories

Title	Edit Student Details
Description	As a student, I should be able to edit my details.
Actors/Roles Involved	Student, StudentDB
Pre-conditions	Student should be registered in the system.
Flow of Events	
Post-conditions	
Assumptions	
Limitations	

Title	Add or drop Courses
Description	As a student, I should be able to add or drop courses.
Actors/Roles Involved	Student, StudentDB, CourseDb
Pre-conditions	Student should be registered in the system.
Flow of Events	
Post-conditions	A minimum number of courses is maintained.
Assumptions	Student is eligible for making modifications to courses.
Limitations	

Title	Add Student Details
Description	As an administrator, I should be able to add student details to the system.
Actors/Roles Involved	Adminstrator, StudentDB, CourseDB.
Pre-conditions	
Flow of Events	
Post-conditions	Student will be registered/updated in the system.
Assumptions	
Limitations	

Title	Remove Student
Description	As an administrator, I should be able to remove a student from the system.
Actors/Roles Involved	Administrator, Student DB, Course DB
Pre-conditions	
Flow of Events	
Post-conditions	Student will no longer be in the system.
Assumptions	
Limitations	

Title	Check enrollment

Description	As an administrator, I should be able to verify enrollment in a particular course.
Actors/Roles Involved	Administrator, Student DB, Course DB
Pre-conditions	
Flow of Events	
Post-conditions	
Assumptions	
Limitations	

Title	**Modify enrollment**
Description	As an administrator, I should be able to modify enrollment in a particular course.
Actors/Roles Involved	Administrator, Student DB, Course DB
Pre-conditions	
Flow of Events	
Post-conditions	
Assumptions	
Limitations	

3. Solution Integrations

3.1 Summary of Products

- Node.Js : Server-side
- MongoDB/REDIS : Back-end Database Store
- Zookeeper : Registry Service
- Swagger/API Docs/Bootstrap : API documentation for the project
- LucidChart : UML diagram constructions

3.2 Solution Use-Cases

Title	Student Management
Description	To main the records of students, add/remove and updating their records
Products Involved	Student Micro-service
Types of Integration	No Integration Required
Functions	CRUD API of Students

Title	Course Management
Description	To main the records of students, add/remove and updating their records
Products Involved	Course Micro-service
Types of Integration	No Integration Required
Functions	CRUD API of Courses

Title	Enrollment
Description	To main the records of which student enrolled in which course. Also maintained at course level, the list of students enrolled in each course
Products Involved	Student Micro-service, Course Micro-service and Referential Integrity Micro-service
Types of Integration	Integration is thorough the Referential Integrity Micro Service
Functions	Student Changes Made API, Courses Changes Made API and CRUD of Courses & Students

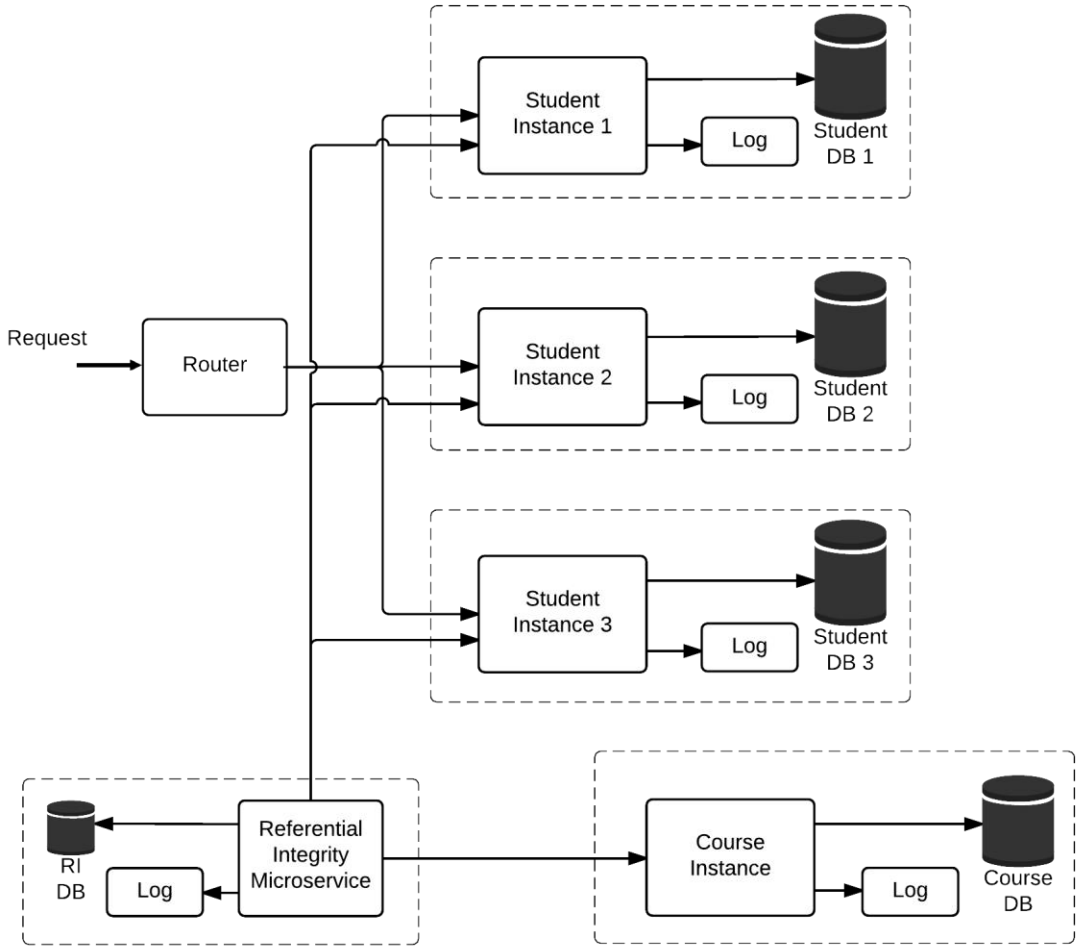
4. Architecture Design

4.1 Overview

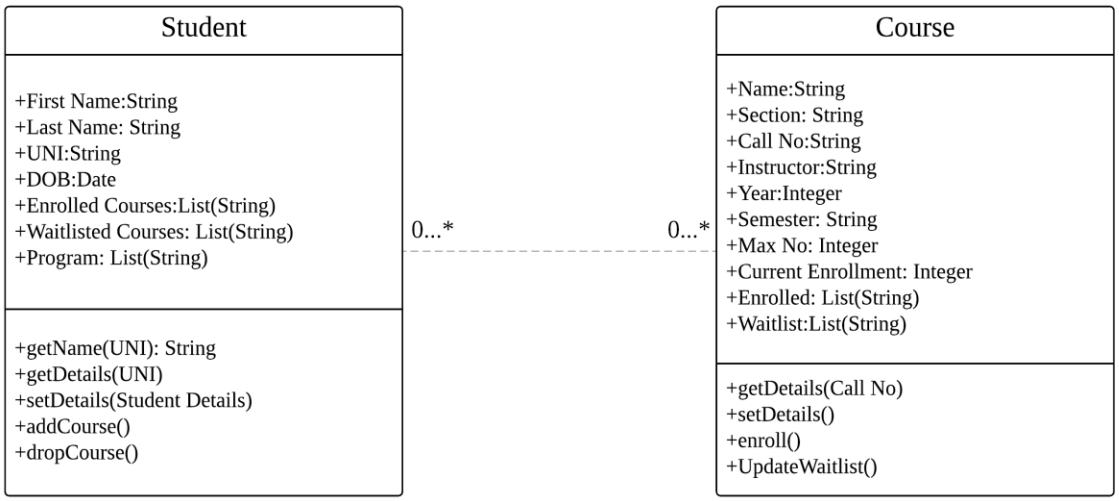
All the micro services are developed in such a way so as to maintain referential integrity and to configure them using APIs.

4.2 System Design

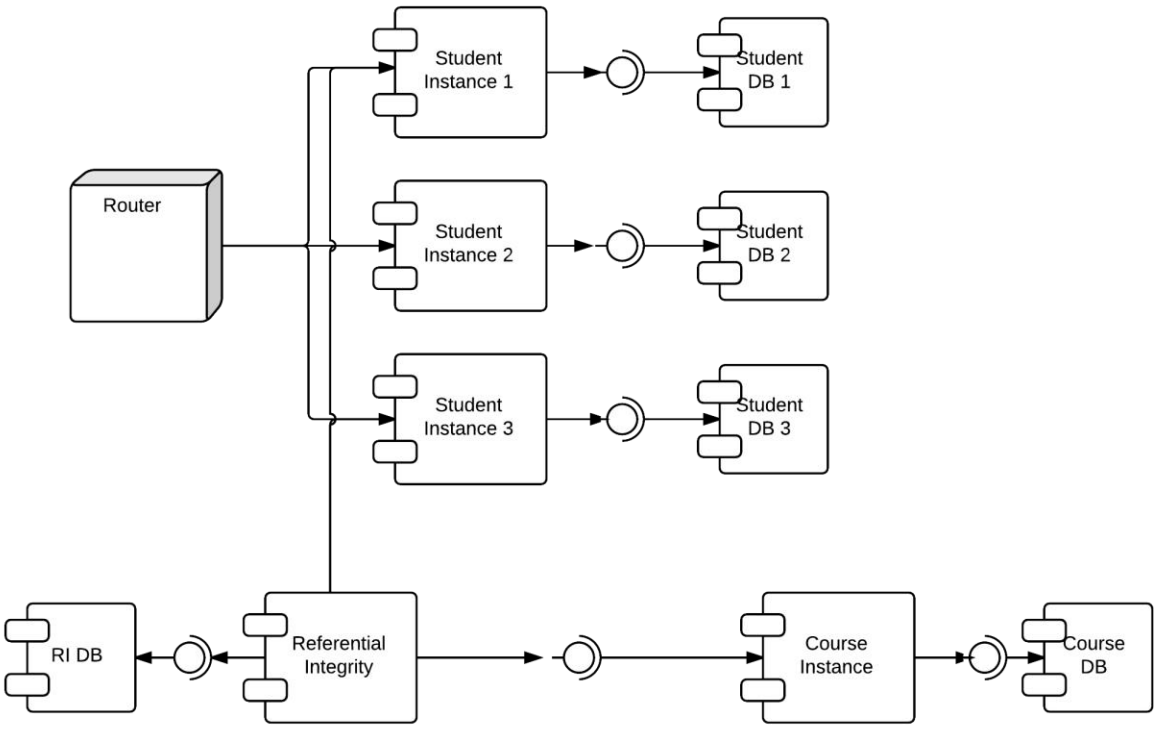
Architecture Diagram



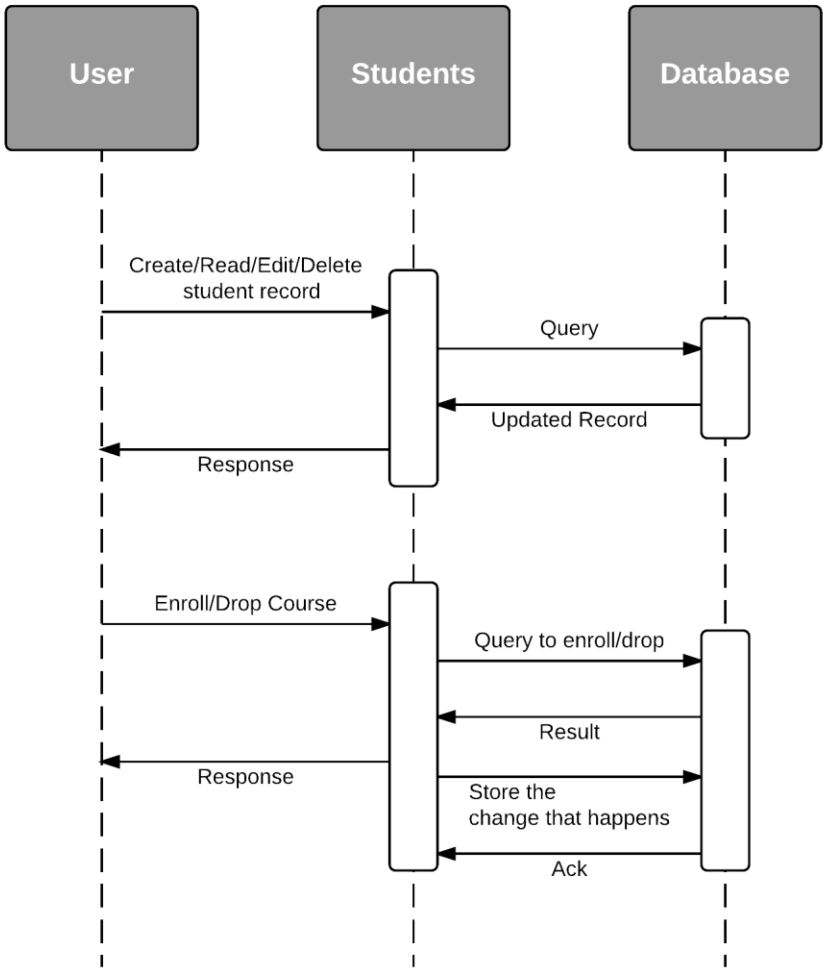
Class Diagram



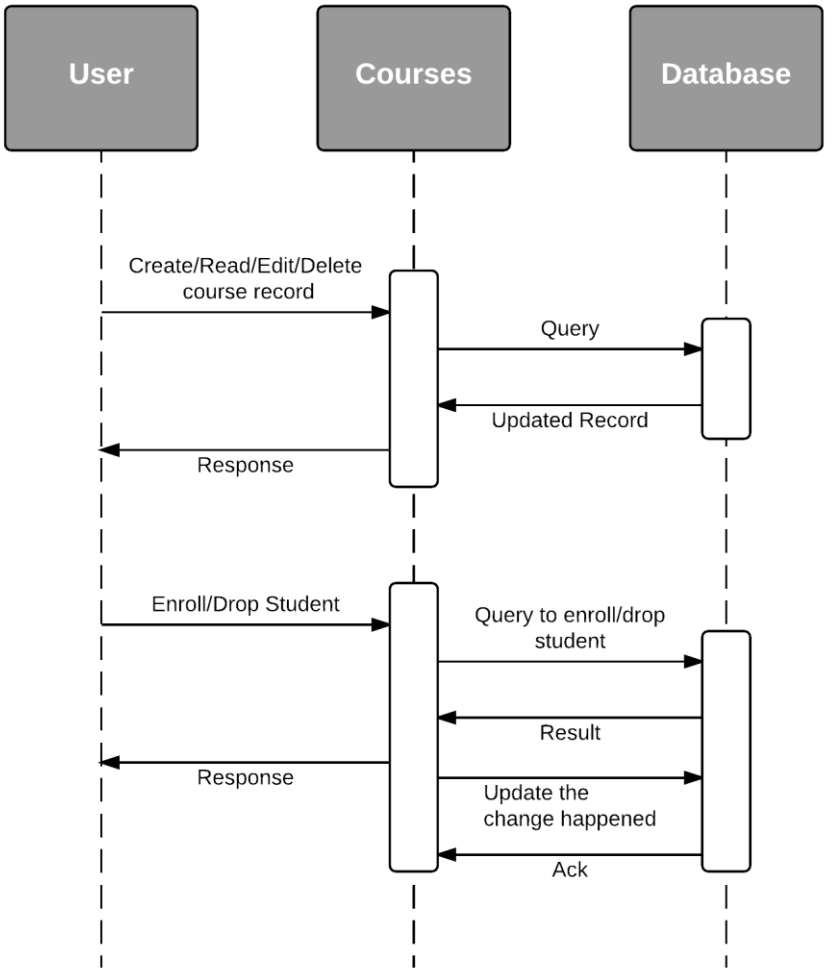
Component Diagram



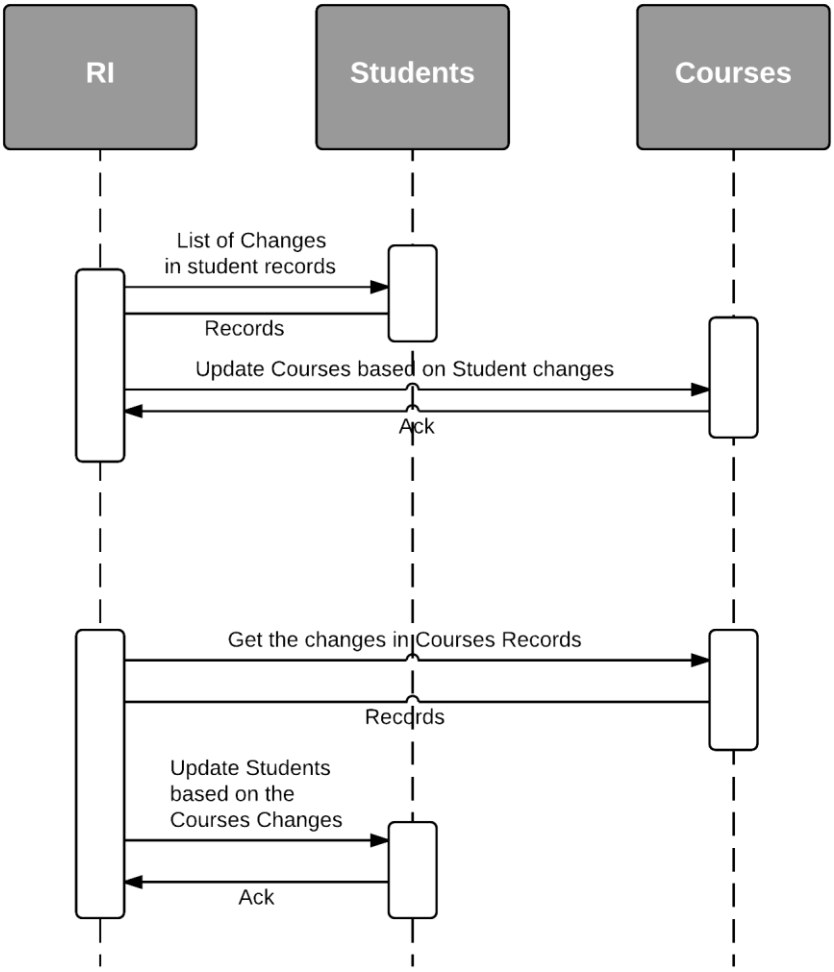
Student Sequence Diagram



Course Sequence Diagram



Referential Integrity Micro Service Sequence Diagram



4.3 Dependencies

Node.Js : Server-side
MongoDB/REDIS : Back-end Database Store
Zookeeper : Registry Service

4.4 Environment

Server: Offline mode
Database: Local Database Store
Node.Js: V4.1.1
MongoDB: MongoDB 3.0

4.5 User Interfaces

No such requirements or interaction, all the interaction is channeled entirely through the APIs.

4.6 Integration/Services Interfaces

The new APIs to be defined include:

- API for CRUD features on student record.
- CRUD features on Course records.
- Configuration APIs for the micro services created.

4.7 Data Models

Student:

First Name	String
Last Name	String
UNI	String
DOB	Date
Enrolled Courses	List (String)
Waitlisted Courses	List (String)
Program	List (String)

Course

Name	String
Section	String
Call No	String
Instructor	String
Year	Integer
Semester	String
Max No	Integer
Current Enrollment	Integer

Enrolled	List (String)
Waitlist	List (String)

4.8 Security

- Only public APIs made available to general users.
- Students only exposed to APIs connecting to student DB.
- Only administrator has privileges to configure the micro service instances.

4.9 Extensibility

Build API's

API's can be consumed by any application

Each Microservice with all required DB store

Possible future extension includes migration to Zookeeper and API Docs.

4.10 Concurrency

- Inbuilt database concurrency features with multi-user edits. (MongoDB)
- MongoDB uses multi-granularity locking which enables operations to lock at multiple levels, such as the global, database or collection level.

4.11 Performance

- Load Balancing
- Routing based on partitioning of Student Data

4.12 Scale

Scaling assumptions include:

- Re-partitioning of student data based on load or increased requirement
- We will take into consideration a scale factor of 1000 entries, for each DB store, which when exceeded, will trigger the creation of a new Db instance.

4.13 High Availability

- Exceptions will be handled properly and properly logged in the application logs/ corresponding logs

- In case of Failures corresponding Failure messages will be sent to the Users. The info about various messages can be found in API docs

4.14 Installation/Deployment/Distribution

- Will be deployed on local system, no special requirements or SaaS components.
- Database also hosted locally, for now.
- We intend to utilize Zookeeper to handle service registry and make them accessible to all users who require our services.

4.15 Configuration and Administration

- Config APIS for administration of each micro service will be provided.
- This enables changes to the data model of students and courses.
- Numerous APIs for instance, for repartitioning of the student micro service.

5. Challenges

- Maintaining referential integrity between the student and course db.
- Protecting the course db access from students.
- Handling data partitioning, particularly when scaling up from an initial number of students is required.

6. References

E6998 – Microservice Application and API Development – Lectures and Presentations by Donald Ferguson.

7. TLDS Template Revision History