**DEPARTMENT OF COMPUTER ENGINEERING**

**CMPE 273-02**

**SPRING 2013**



**Submitted to**

Prof. Simon Shim

**Submitted by**

Padmaja Adhyam Nagarajan<anpadmaja@gmail.com>

Dinesh Reddy Tathekalva<dineshreddy.in@gmail.com

Hari Vardhan Pyaram<p.hari.vrdhn@gmail.com>

Jaismeen Kaur <jaismeensudan@gmail.com>

Sandeep Rao Boinpally<b.sandeeprao@gmail.com>

# INDEX

1. Introduction
2. System Overview
   1. Functional Requirements
   2. Non Functional Requirements
3. Connection Pooling
4. Evaluating Performance of System
5. Database ER Model
6. Screen Shots
7. Conclusion

# INTRODUCTION

The project is about the University Database as a web service application which stores all the data pertaining to students, courses, instructors and the admin. This web service application is mainly used by the students and the instructors. The students can search for the classes available and enroll in the interested classes. The instructor can make changes in the subjects he teach and can select the hours he can teach.

* Firstly, all the classes and functions defining the interface have been specified.
* For each of the entity used in the system the Data Access Object Classes have been created.
* In order to test the functionalities of the web service application the Test Cases have also been created.
* To develop these test cases, the JUnit is used.
* Various validation techniques were utilized by the system to validate the details of the users.

# SYSTEM OVERVIEW

This system performs student, instructor and administrator functionalities. Here student can search for the classes, enroll for the interested classes and can modify the classes enrolled. The instructor can change his classes to be taught and timing by sending a request to the administrator. The administrator does the changes based on the requests.

While designing the web service application system, the design considerations are assumed to successfully utilize the functionalities are as follows

## Functional Requirements

The system should be able to do the following:

## Student Portal

* Allow the students to search for the classes required.
* Allow the students to enroll for the classes.
* Allow the students to drop the classes.
* Allow the students to edit their profiles and update their details.
* Allow the students to view all their details.
* Allow the student to change the password.

Instructor Portal

* Allow the instructor to change the timings and room number of the classes.
* Allow the instructor to submit a change request to the admin for adding or dropping a class.
* Allow the instructor to edit their details.
* Allow the instructor to view all their details.
* Allow the instructor to change the password.

Administrator Portal

* Allow the admin to list all the details of the users.
* Allow the admin to make the changes as per the requests of the instructor.
* Allow the admin to auto generate the email ids of the every user created.
* Allow the admin to edit database.
* Allow the admin to assign courses.
* Allow the admin to change the password.

Display Success or Error information on any type of Delete, Update, Issue, Return operation.

## 

## Non - Functional Requirements

Client Side Validations

* All the Java servlet Pages are validated for any operation performed and for any input given with incorrect values on a particular page.
* All user IDs for all members in SSN Format.

Server Side Validations

* The server checks all the incoming values from the client side & validates the values.
* It is enhanced for certain process carried out directly from the JSP i.e. the data is sent through JSP rather than URL.
* Handles all the exceptions in the functionality.

Availability

* The web service application system should be available at any point of time.

Accuracy

* The web service application system should be accurate on the calculations and operations performed.

Accessibility

* The web service application system should be accessible from any location.

Compatibility

* The web service application system should be compatible with all the web browsers.

Reliability

* The web service application system should be reliable.

Scalability

* The web service application system should be able to take greater loads.

Usability

* The web service application system should be easy to use and

## CONNECTION POOLING

In order to accommodate maximum number of database connections possible, the concept of connection pooling was used in this project. The following code was used in the constructor of the service and the connection pool was created before using the web services. Each time a method needs a connection, it fetches the available connection from the pool of connections.

**public** UniversityService(){

BasicDataSource ds=**new** BasicDataSource();

ds.setDriverClassName("com.mysql.jdbc.Driver");

ds.setUsername("root");

ds.setPassword("root");

ds.setUrl("jdbc:mysql://localhost/universitydatabase");

dataSource = ds;

}

For each method, following code was used to retrieve the connection from the pool.

con.dataSource.getConnection();

## EVALUATING THE PERFORMANCE OF THE SYSTEM

The performance of the system was improved by following a step by step process. Below are the various steps that were undertaken in order to improve performance.

* Pooling of db connections.
* JMeter scripts to identify response time.
* JavaScript for loading, auto complete and front-end validations.
* JUnit for Test Harness.
* Updations done when required.

**JMeter Runs**

Following runs were taken on the web application in order to gauge the application at each step.

1. **Baseline Run:** This is the base run taken when the application didn’t implement Connection pooling and validation checks.

*Baseline = Base Code*

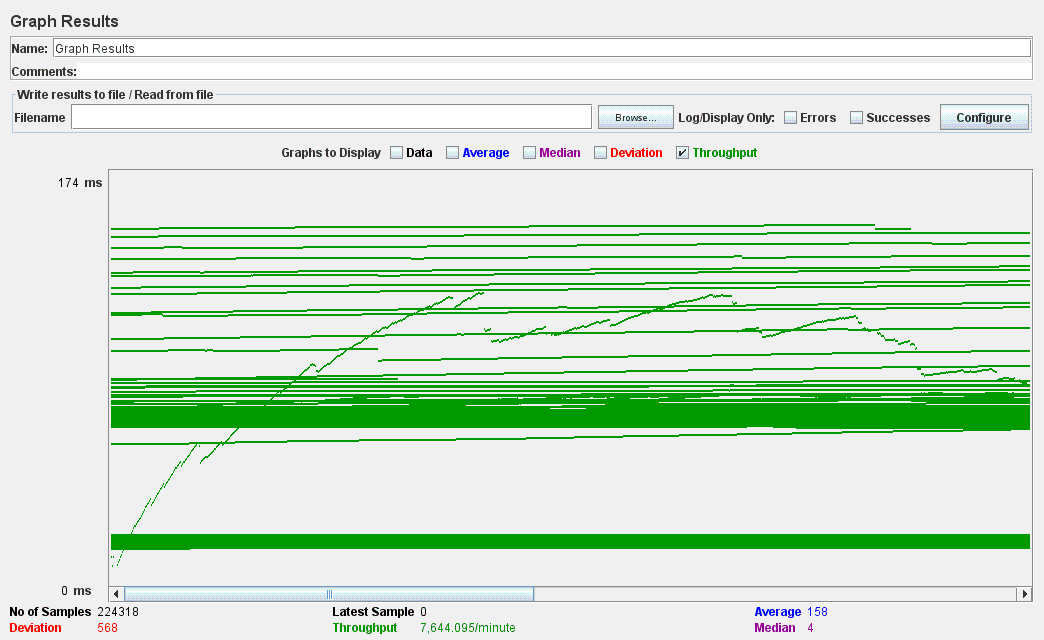
1. **Run 1:** This run was taken on the application with the connection pooling in place.

*Run 1 = Base Code + Connection Pooling*

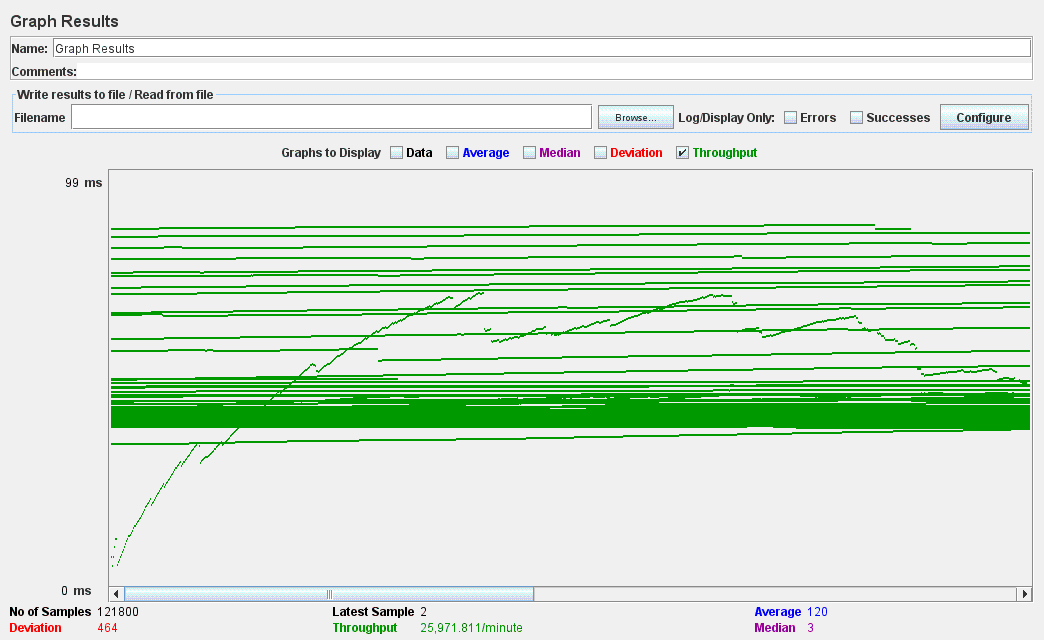
1. **Run 2:** This run was taken on the application with the connection pooling and validations check.

*Run 2 = Base Code + Connection Pooling + Validations*

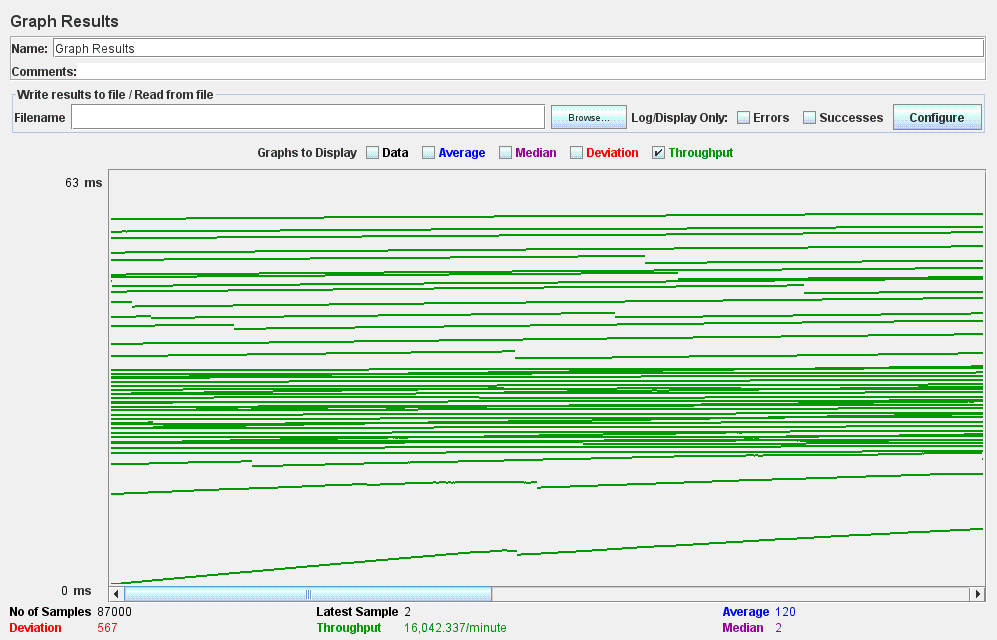
Below are the screen shots of throughput variable of each of the three runs.



*Figure: Baseline Run- throughput*



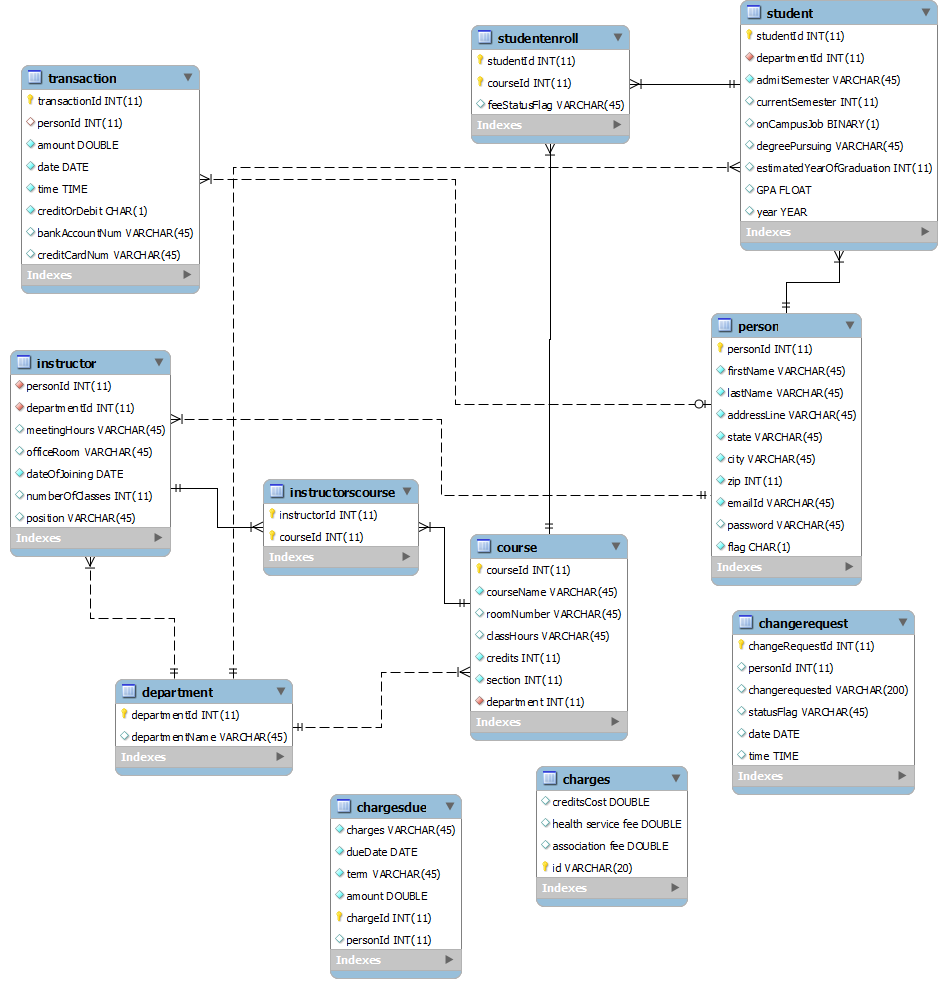
*Figure: Run 1- throughput*



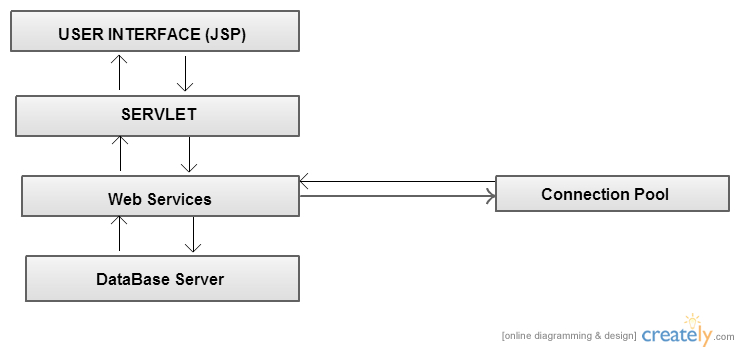
*Figure: Run 2 - throughput*

Performance related statistics were generated by the JMeter after the three runs were taken. These statistics were converted into useful information by drawing graphs. Below are the few graphs that convey the different aspects of the performance of the system.

**DATA BASE ENTITY RELATIONAL MODEL:**

****

**SYSTEM ARCHITECTURE:**

****

**SCREEN SHOTS OF THE SYSTEM**

**Screen shots for all the three modules ( student, instructor and admin )are attached in the UniversityRecordSystem\_screenshots folder.**

**CONCLUSION**

**The University Record System as a project improved the team’s understanding of the concept of Web services and how it is implemented with the services client. The team learnt how to design database in MySql and its connectivity to the system. A very essential aspect while designing database that was learnt was Connection Pooling. The team was successfully able to implement the pooling feature. Validation checks were done at the frontend to reduce the possibility of system breakdown during its execution.**

**Hence, the team learnt several new concepts during the designing of this system and opens the future prospect of learning and improvements.**