**Student Registration System**

Technical Design

Version 1.0

UC Irvine: Java Programming I

Team E: Travis Ly, Derek Tam, Sasi Vallikkat, Chris Verhoff

**Table of Contents**

Contents

[Overview 2](#_Toc412200266)

[Requirements 2](#_Toc412200267)

[system features: 2](#_Toc412200268)

[business rules: 2](#_Toc412200269)

[assumptions: 2](#_Toc412200270)

[System Processes 3](#_Toc412200271)

[System Flow 4](#_Toc412200272)

[Object Model 5](#_Toc412200273)

[User Interface 6](#_Toc412200274)

[Data Structure 8](#_Toc412200275)

[Student file structure 8](#_Toc412200276)

[course file structure 8](#_Toc412200277)

[Test Cases 9](#_Toc412200278)

# Overview

Our Team (**Team E**) has an assignment to write a console application that will be used to register students for a course. The storage for the data for this registration will be store in text file(s).

Github will be used to store the code which will also serve as the source for the version control sheet.

# Requirements

The student registration system will display the list of courses available for registration. The program should show current registration numbers and available seats for each course. After a student successfully registers, the registration information should be stored in a file.

## system features:

The system should provide for students the ability:

* to view an alphabetical list of courses that includes the course identification number, course dates, name, brief summary, the enrollment limit, and the number of students already enrolled.
* to register or un-register for a course.
* to see a list of classes for which he/she is currently registered.

## business rules:

The system should **not** allow:

* a student to register for a full course.
* one student to view the registration information for another student.
* a student to un-register from a course for which he/she is not registered.

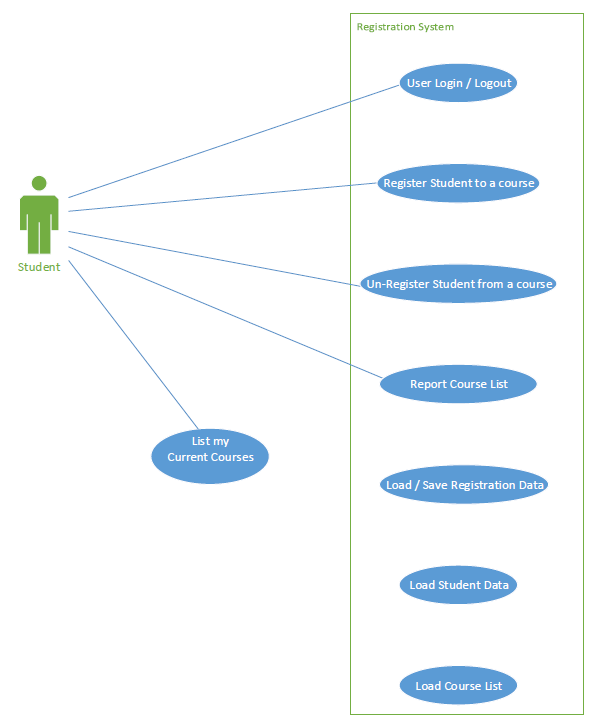
## assumptions:

* The Use Case diagram can be created using any tool and will highlight all Use Cases for the system.
* You must submit all of your data files as a part of your assignment.
* Your source code must be submitted in a single zip file containing the group name.
* At a minimum, the version control sheet must include the name of the updated file, the date the change was made, the name of the person making the change, and a high-level description of the change.
* Have one person submit your final project to me.
* Your program will be designed using object oriented techniques.

# System Processes

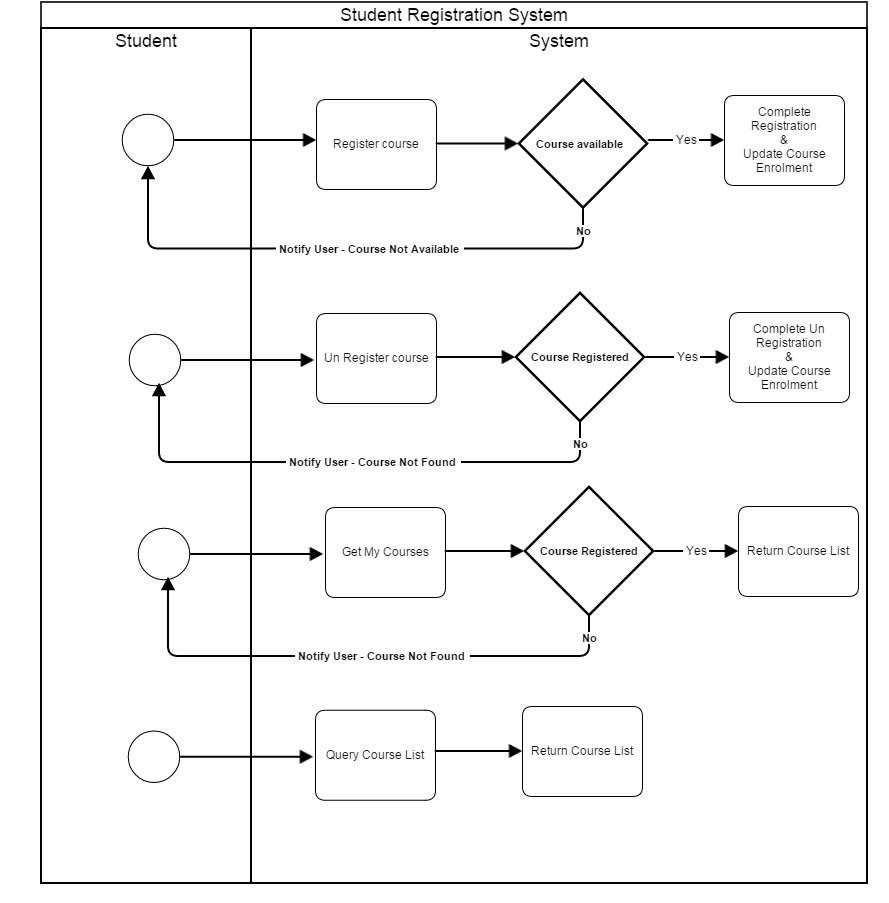
The following use case diagram illustrates interaction of the student with the system.

As defined in the requirements, a student will login and be able to list their current courses, list the current courses list, register for a course, or unregister from a course.



# System Flow

The following activity diagram illustrates interaction, via modules, of the student with the system.



# Object Model

Model

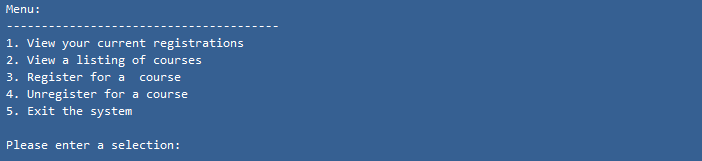


# User Interface

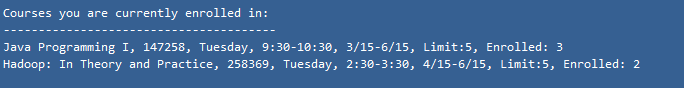
An example of the login screen:



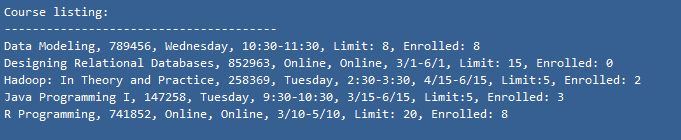
An example of the menu screen:



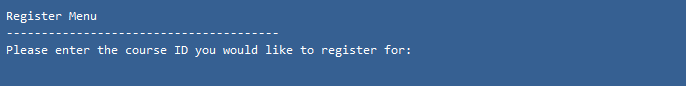
An example of the currently enrolled course listing screen:



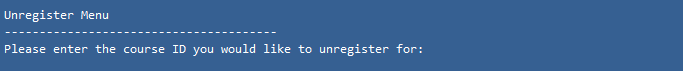
An example of the course listing screen:



An example of the screen to register for a class:



An example of the menu to unregister for a class:



An example of an error if trying to register for a full class:



# Data Structure

The data stored for this system will be in text files. The proposed file format is that of a csv (comma separate values). This format is easy to parse, transportable, and read/write to for this project.

## Student file structure

An example format for a student would be as follows:

**studentFirstname, studentLastname, [courseId, courseId,,,…]**

studentId: integer Student ID

studentFirstName: text The first name of the student.

studentLastName: text The last name of the student.

courses: ArrayList [int] A list of the course IDs for which the student is currently enrolled.

## course file structure

An example structure for a course would be as follows:

**courseName, courseId, courseStart, courseEnd, courseSummary, courseEnrollmentLimit, [student Id, student Id, student Id,…]**

courseName: text The name of the course.

courseId: integer The unique id associated with the course.

courseStart: date The start date for the course.

courseEnd: date The end date for the course.

courseSummary: text A brief summary for the course.

courseEnrollmentLimit: integer The limit of the number of students that can enroll in the course.

courseStudents: ArrayList[int] List of student IDs enrolled.

# Test Cases

**Team E** will provide text files that will provide example data for a number of students and a number of courses. This test will show students that are currently enrolled in the system.

A simple test program will also be provided that will walk through the program features testing them for correctness.