CS372 - Software Engineering

Project Documentation

Registration Assistance Program

R.A.P

Dr. Samira Sadaoui

March 24th, 2014

<u>Group Members:</u>

Anthony Nguyen

Chris Arnold

Nathan Cherwaty

Dawn Buttazoni

Mark Comte

Ian Hauser

Table of Contents

1. Problem Definition

Develop a web based tool for University of Regina Computer Science students capable of: tracking a student's progress through their degree, giving a list of courses that can be taken, and showing which courses are still necessary in order to graduate. The system accepts completed courses submitted by the user as well as their associated grades. Combined with a standard framework, the system will return to the user a graphical representation of their current progress through the degree as well as a list of required course that still need to be completed.

Functional Requirements:

Two types of Users: Admins and Students

Functional Requirements for Students:

- *Sign Up / Create account:* A student is prompted to sign up for an account by filling in appropriate information pertaining to themselves and their education. The information they enter will be stored and used to identify the student when they attempt to log in during succeeding sessions.
- *Log In*: If student has previously signed up and created an account, they can enter their password and username to access the information they have already supplied as well as manage course schedule. If they enter the wrong information, display a warning. If they reach a threshold for failed attempts, lock them out temporarily.
- *Course Selection:* A student selects the courses that they have taken (and adds grades) from a list of courses that the University offers, which are filtered by the user.
- *Search For Courses*: students should be able to use simple tools to search the given list of courses in order to quickly and easily find the course they are looking for.
- *Modify Taken Courses*: Students should be able to remove their previously added courses or update their marks for those courses.
- *Request Course*: If a course that is not among the list of offered courses, students

should be able request that an admin add this course.

Functional Requirements for Admins:

- ➢ *Log in*: An admin responsible for software maintenance can log in using a special admin account.
- ➢ *Add Course*: add a new course to the system easily.
- ➢ *Reset Passwords*: If a user or other admin has forgotten a password an admin could reset it.
- ➢ *Review Requested Course*: Admin should be notified in list format when users create new courses. The admin can then review the information and add a verification stamp if the course information is correct, edit the submitted course, or delete the course if the information is incorrect.
- ➢ *Promote Users*: An admin should be able to change the roles of a user from student to admin or demote an admin to a student.

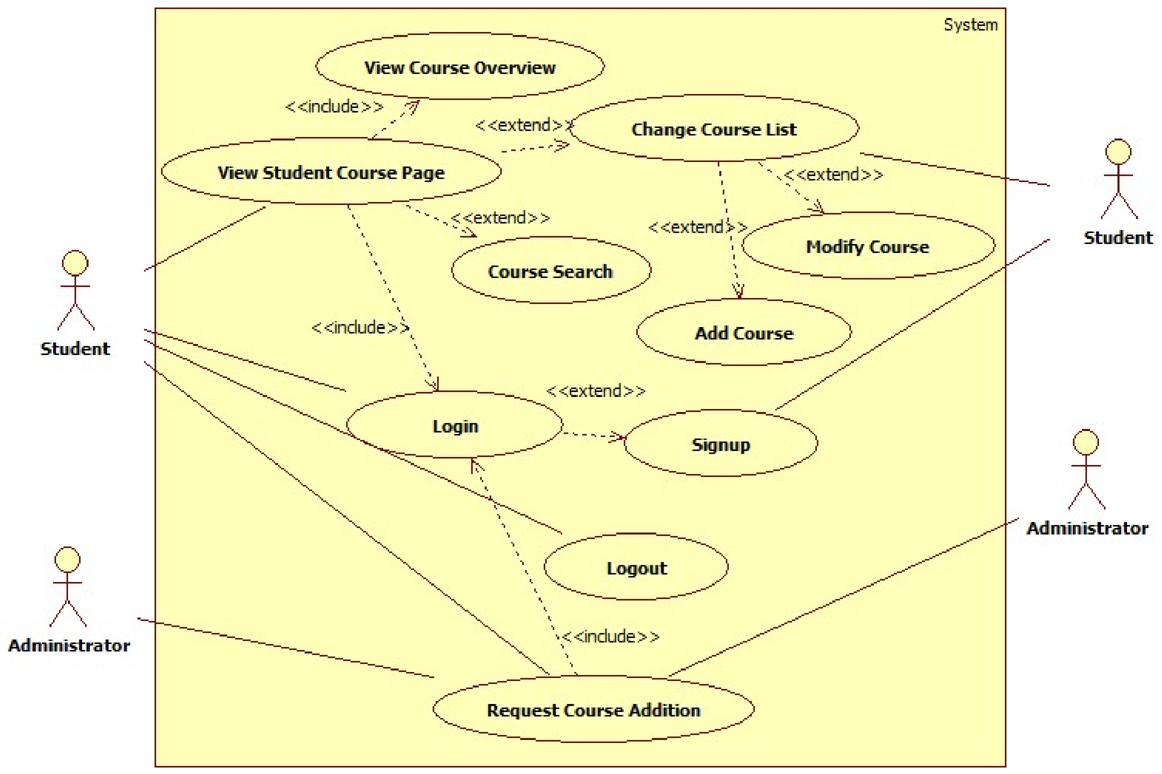
Qualities Required

- ➢ *Correctness*: Software produces the correct averages and course information pertaining to each student. Display prerequisites to the user when adding courses that require other classes.
- ➢ *User Friendly*: The software displays the information in a logical manner that makes it easy to read and manage the user's information. The software must also make it easy to view and add new courses.
- ➢ *Robustness*: Errors (such as improper login) produce messages to the user and don't cause unexpected results. Web based input should be cleaned and trimmed to accepting erroneous input as valid.
- ➢ *Scalability*: Software must be able to accommodate a large number of users concurrently.
- ➢ *Security*: User information such as login information is stored safely in a database that cannot be maliciously accessed.

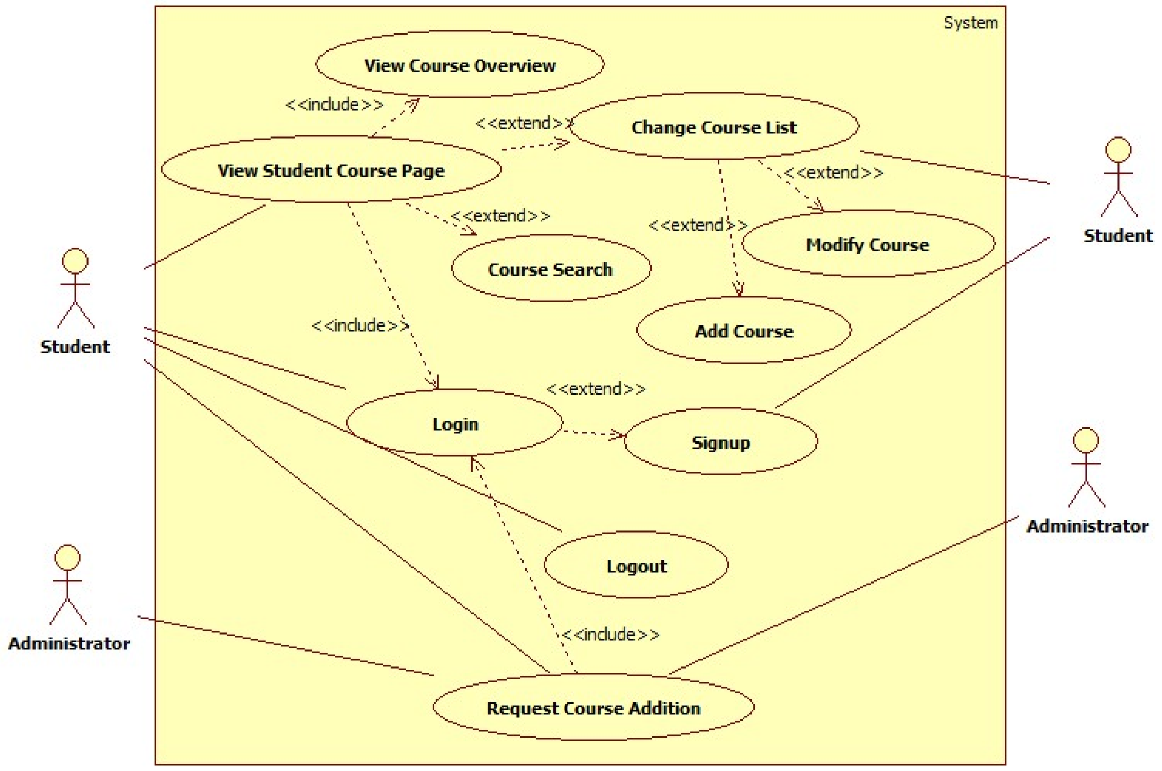
2. Software Requirement Specification Document

A. Use Cases

Student Use Case Diagram



Administrator Use Case Diagram



1. Use Case: "Modify Course"

Initiating Actor: Student wants to remove class from classes taken list or add a mark

Preconditions: User has already logged in and has previously added classes

Scenario 1: Remove Course

- Student views course page
- List of courses student had previously specified as taken is shown
- Student selects a course from the list of courses they have taken
- Specified course is removed from list of courses taken by student

Scenario 2: Edit Grade

- Student views course page
- List of courses student had previously specified as taken is shown
- Student selects a course from the list of courses they have taken and enters their grade received

- Specified course is updated to contain the grade entered

Scenario 3: Invalid Grade Entered

- Student views course page
- List of courses student had previously specified as taken is shown
- Student selects a course from the list of courses they have taken and enters an invalid grade
- Error message is displayed to student requesting appropriate grade

Scenario 4: Student session timed out

- Student is directed to login page

Scenario 5: Student has no courses taken

- No option to modify courses displayed

Postconditions:

- Student removes a course from course taken list

  OR

- Student successfully edits grade on a taken course

  OR

- Student list of taken classes remains the same

Benefiting Actor: Student; Their course taken page is updated

2. Use Case: "Add Course"

Initiating Actor: Student wants to add a course to taken course list

Preconditions: User has already logged in

Scenario 1: Add Course

- List of courses student has previously specified as taken is shown as well as courses that are still required, a course search bar is displayed.
- Student specifies search criteria from the corresponding menus
- A list of courses corresponds to the search criteria is populated in the search sidebar
- Student selects the course from the list of populated courses

- Specified course is verified and added to list of courses taken by student

Scenario 2: Desired Course does not exist

- List of courses student has previously specified as taken is shown as well as courses that are still required, a course search bar is displayed.
- Student specifies search criteria from the corresponding menus
- A list of courses corresponds to the search criteria is populated in the search sidebar
- Course that student is searching for is absent from populated list

Scenario 3: Student session timed out

- Student is directed to login page

Postconditions:

- Users view of courses taken includes newly added course

OR

- Users view of courses taken remains the same

Benefiting Actor: Student; Their course taken page is updated

3. Use Case: "Review Requested Course"

Initiating Actor: Student wanting to add a non-existing course to a list of course options

Precondition: Student has already logged in

Scenario 1: Verify Course - Accepted

- Student views Submit a course page
- Student enters course number, faculty, subject, prerequisites, and a brief description of the course
- Student presses submit and course is added to pending courses
- Admin logs on and accesses verify course page in which all pending courses are listed

- Admin selects a course to verify, all information entered by student is displayed
- Admin believes it's a reasonable course request that is missing from current list of available courses, they accept the request and pending course is added to approved course database table
- Student logs on
- Course requested becomes a searchable course

Scenario 2: Verify Course - Rejected

- Student views course request page
- Student enters course number, faculty, subject, prerequisites, and a brief description of the course
- Student presses submit and course is added to pending courses
- Admin logs on
- Admin accesses verify course page in which all pending courses are listed
- Admin selects a course to verify, all information entered by student is displayed
- Admin believes it's an unreasonable course request and presses the deny button

Scenario 3: Verify Course - Accepted - Modify Course request

- Student views course request page
- Student enters course number, faculty, subject, prerequisites, and a brief description of the course
- Student presses submit and course is added to pending courses
- Admin logs on and accesses verify course page in which all pending courses are listed
- Admin selects a course to verify, all information entered by student is displayed

- Admin changes some of the information entered by user
- Admin now believes it's a reasonable course request that is missing from current list of available courses, they accept the request and pending course is added to approved course database table
- Student logs on
- Course requested becomes a searchable course

Scenario 4: User isn't logged in

- Student attempts to view course request page
- Redirected to login page (See login use case)

Postconditions:

- Course is added to list of courses all Students can choose from
- Course table in database contains new course
- Pending course table contains record of student's request

OR

- Pending course table contains record of student's request

OR

- User must login

Benefiting Actor: Students; courses to choose from having taken is now more accurate

4. Use Case: "View Student Course Page"

- Initiating Actor: Student viewing their course page

- Precondition: Student has already logged in

Scenario 1: View Student Course Page without the Course Search box populated

- Student clicks the button to view their course page
- The web server contacts the database to load the course table of the student

- The business layer checks the students course table against the current courses required for the degree they have selected
    - if a course in the students course table matches a course in the degree then the course overview table on the students web page is shown with course as being taken
    - if all the courses are checked in the students course table against the degree and there are unmatched courses in the degree table then the course overview table on the students web page is show with the unmatched courses as not taken
- Once all the students courses have been checked the course overview table on the students web page is updated.

Scenario 2: View Student Course Page with the Course Search box populated

- Student clicks the button to view their course page
- The web server contacts the database to load the course table of the student
- The business layer checks the students course table against the current courses required for the degree they have selected
    - if a course in the students course table matches a course in the degree then the course overview table on the students web page is shown with course as being taken
    - if all the courses are checked in the students course table against the degree and there are unmatched courses in the degree table then the course overview table on the students web page is show with the unmatched courses as not taken
- Once all the students courses have been checked the course overview table on the students web page is updated.
- Student chooses the faculty and subject to search in the search field
- Based on what the student chose as the faculty and subject the web server

contacts the database and requests all the courses in the subject selected

- The database sends the information to the web server

- The web server then populates the search table with the courses received from the database

Postconditions:

- View of courses is up to date

- Student can search for classes

Benefiting Actor: Students; they have a graphical view of their degree

B. Software Qualities

Correctness: Our software demonstrates correctness by always displaying proper information. This means once the user logins in, it produces the correct average and course information pertaining to that particular student as outlined in our functional requirements. Also, when the user tries to find courses to take, no courses are suggested unless the required prerequisites are met.

User Friendliness: The graphical interface of our software has a very clean look and easy for users to navigate through. A sidebar on the left will be used to navigate between all possible pages once logged in, so the user will easily be able to see their options. Course information pertaining to each student is shown upon login, so the user has an immediate view of what they logged in to see. Adding courses to a user's degree is accessible from the same page as the degree overview, so users can see what classes they need to add as they add them. Our software also includes search criteria which can be used so that users can conveniently find a class based on a chosen faculty and subject.

Robustness: Informative error messages are produced when a user tries to submit a form with invalid information. This is checked on the submit a course, login, signup, and modify degree forms. If a user tries to go straight to a certain page without logging in they are redirected to our login page. User requested information is also checked before it is stored in a database so that no unexpected outputs occur.

Efficiency*:* Our software involves simplistic querying of data, so that no bottlenecking occurs. Also, in an attempt to remove computations, AJAX and javascript is only used in situations where it is truly needed, like when a form is submitted. AJAX and javascript is used to minimize database access when incorrect form information is entered.

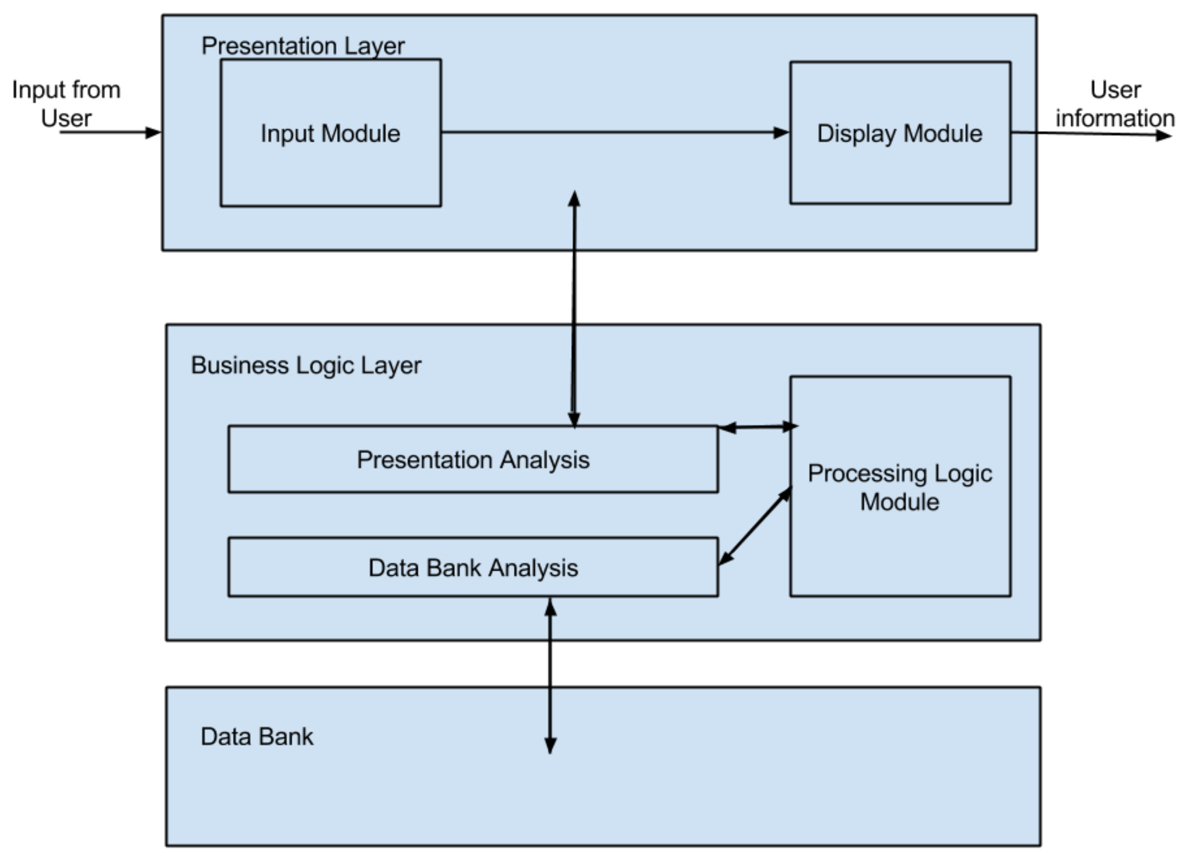
Security: User entered information such as login information and grades are stored safely in a database that cannot be maliciously accessed. Encryption is also used to make sure that password information is secure.

Scalability: A large number of users accessing our site concurrently is made possible with minimal data queries in order to reduce time spent accessing the database. Also, since interaction with the database is minimal, most of the users time is spent interacting with their own view of the site, thus not tying up server resources.

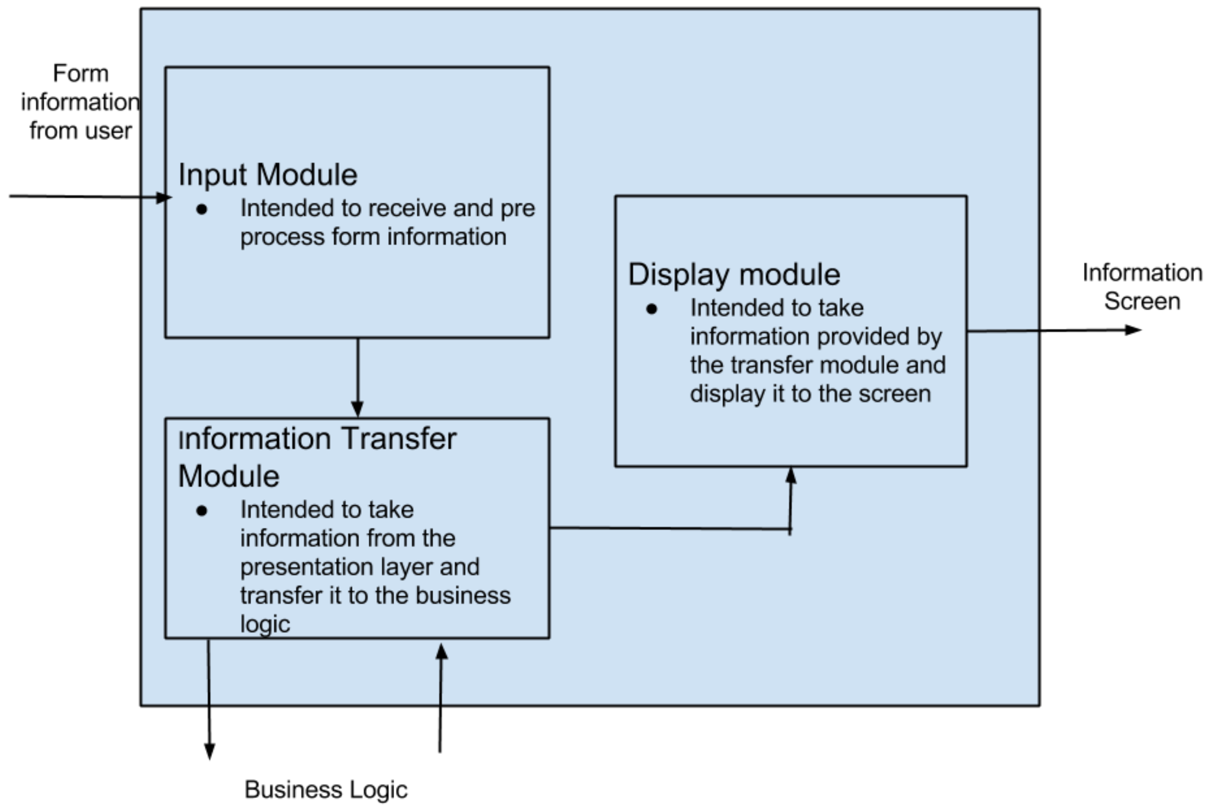
3. Design Specification Document

A. Software Architecture

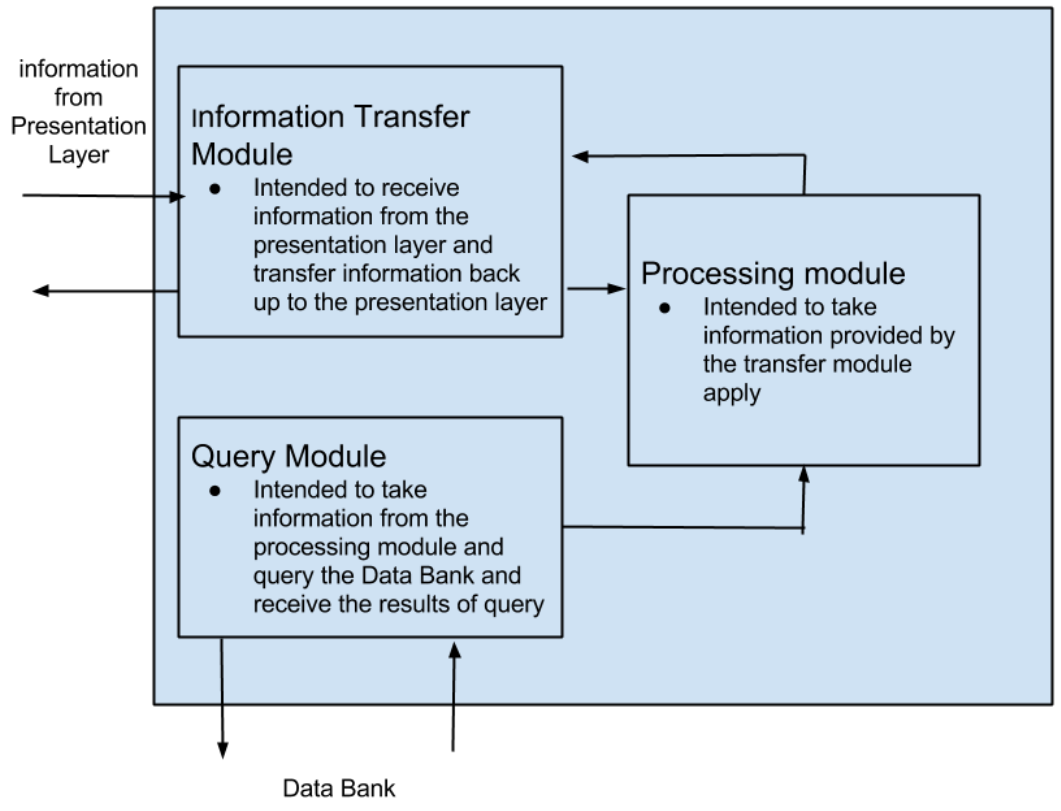
System Top Level Architecture



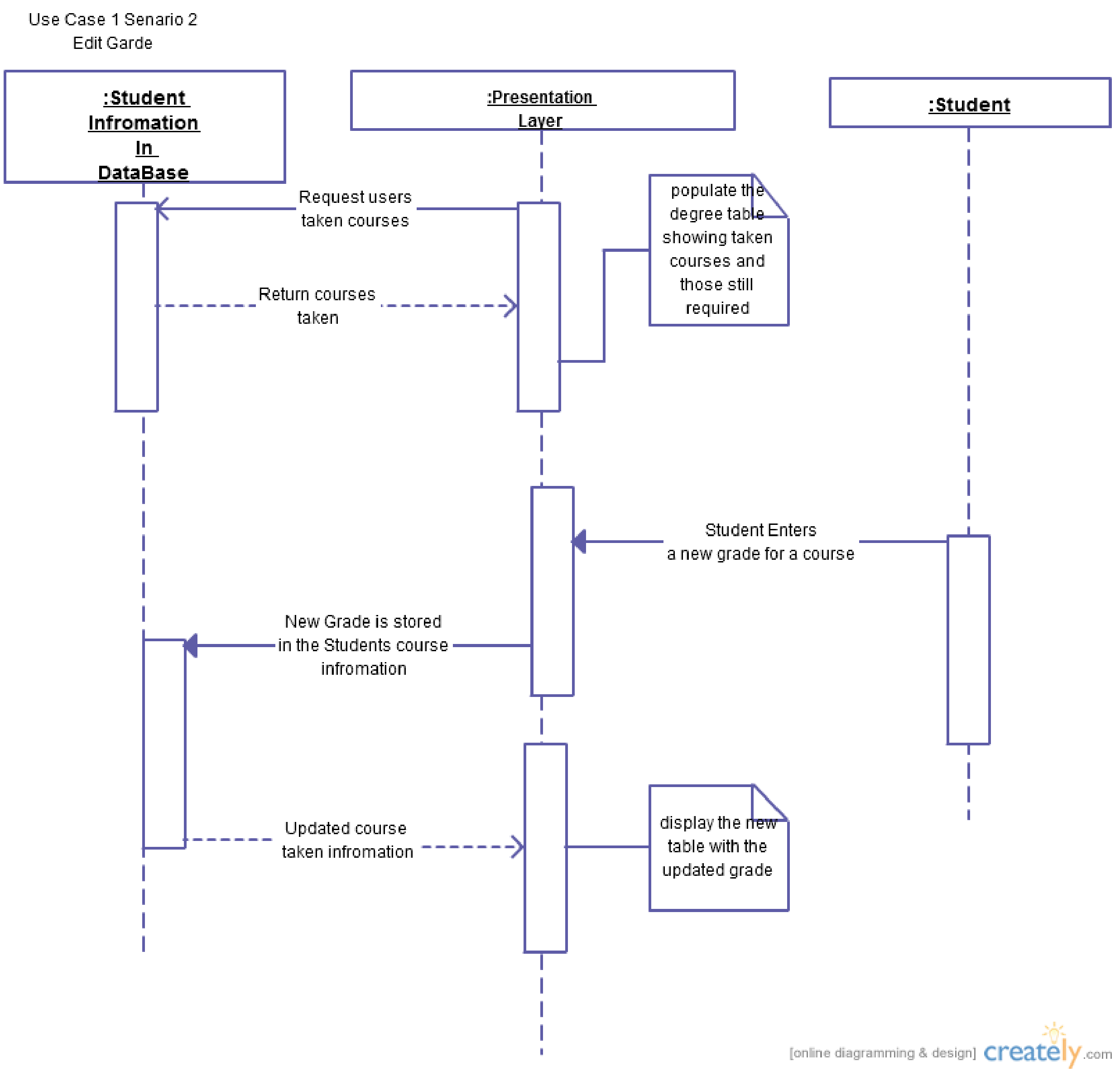
Presentation Layer Top Level Architecture

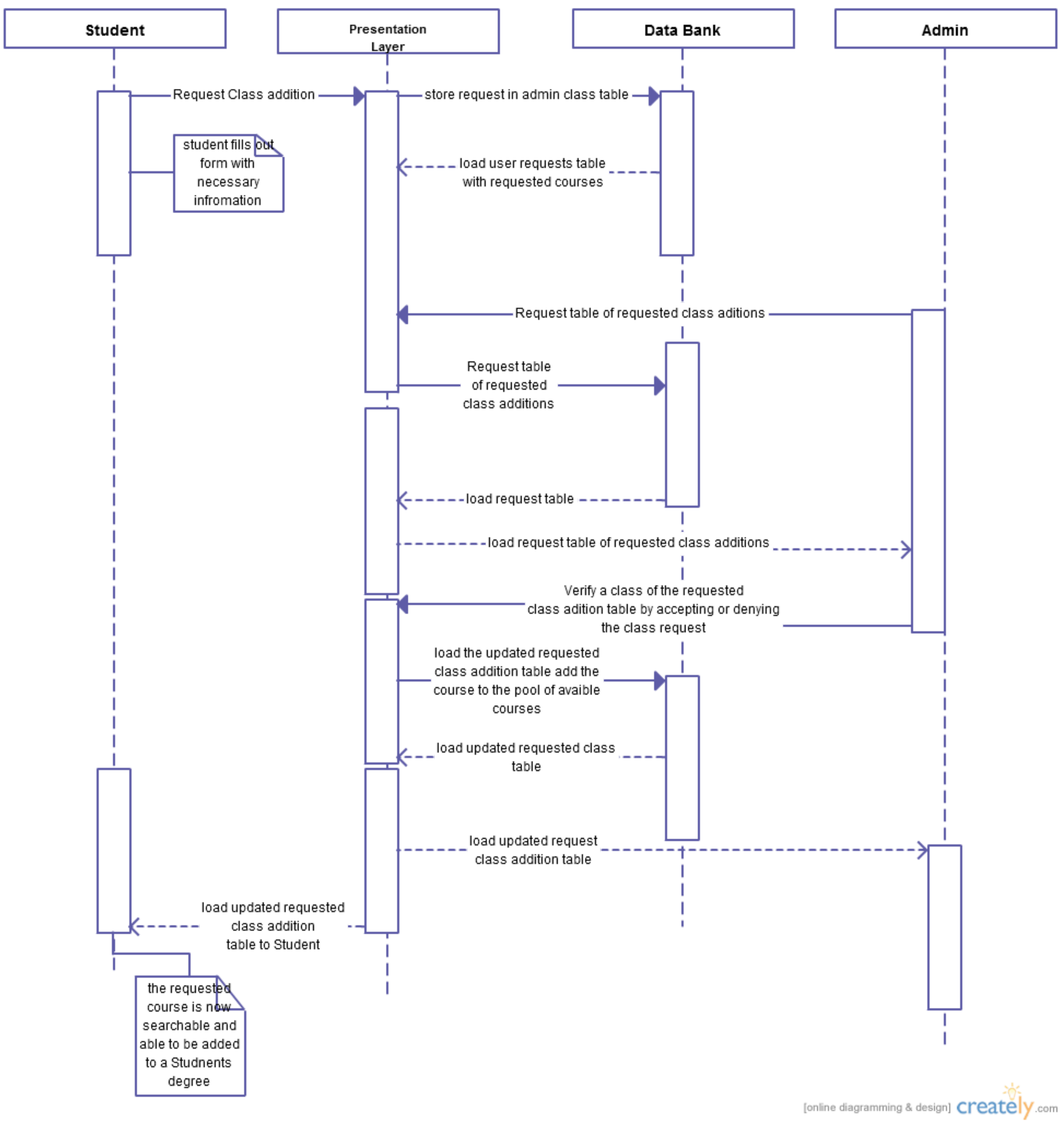

Form information from user
Input Module
• Intended to receive and pre process form information
Information Transfer Module
• Intended to take information from the presentation layer and transfer it to the business logic
Display module
• Intended to take information provided by the transfer module and display it to the screen
Information Screen
Business Logic

Business Layer Top Level Architecture

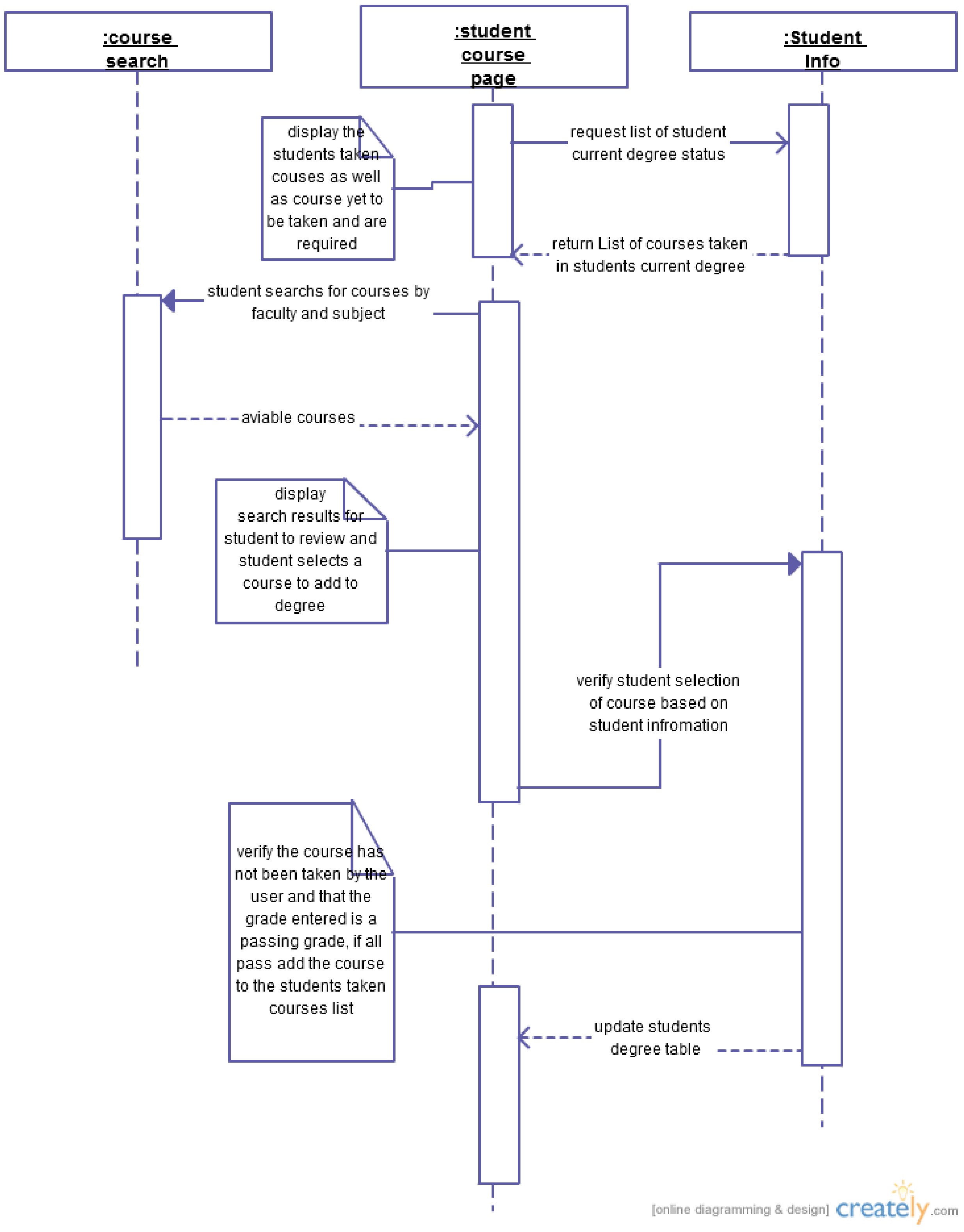

information from Presentation Layer
Information Transfer Module
• Intended to receive information from the presentation layer and transfer information back up to the presentation layer
Processing module
• Intended to take information provided by the transfer module apply
Query Module
• Intended to take information from the processing module and query the Data Bank and receive the results of query
Data Bank

B. Sequence Diagrams


Use Case 1 Senario 2
Edit Garde
:Student Infromation In DataBase
:Presentation Layer
:Student
Request users taken courses
Return courses taken
populate the degree table showing taken courses and those still required
Student Enters a new grade for a course
New Grade is stored in the Students course infromation
Updated course taken infromation
display the new table with the updated grade
[online diagramming & design] creately.com

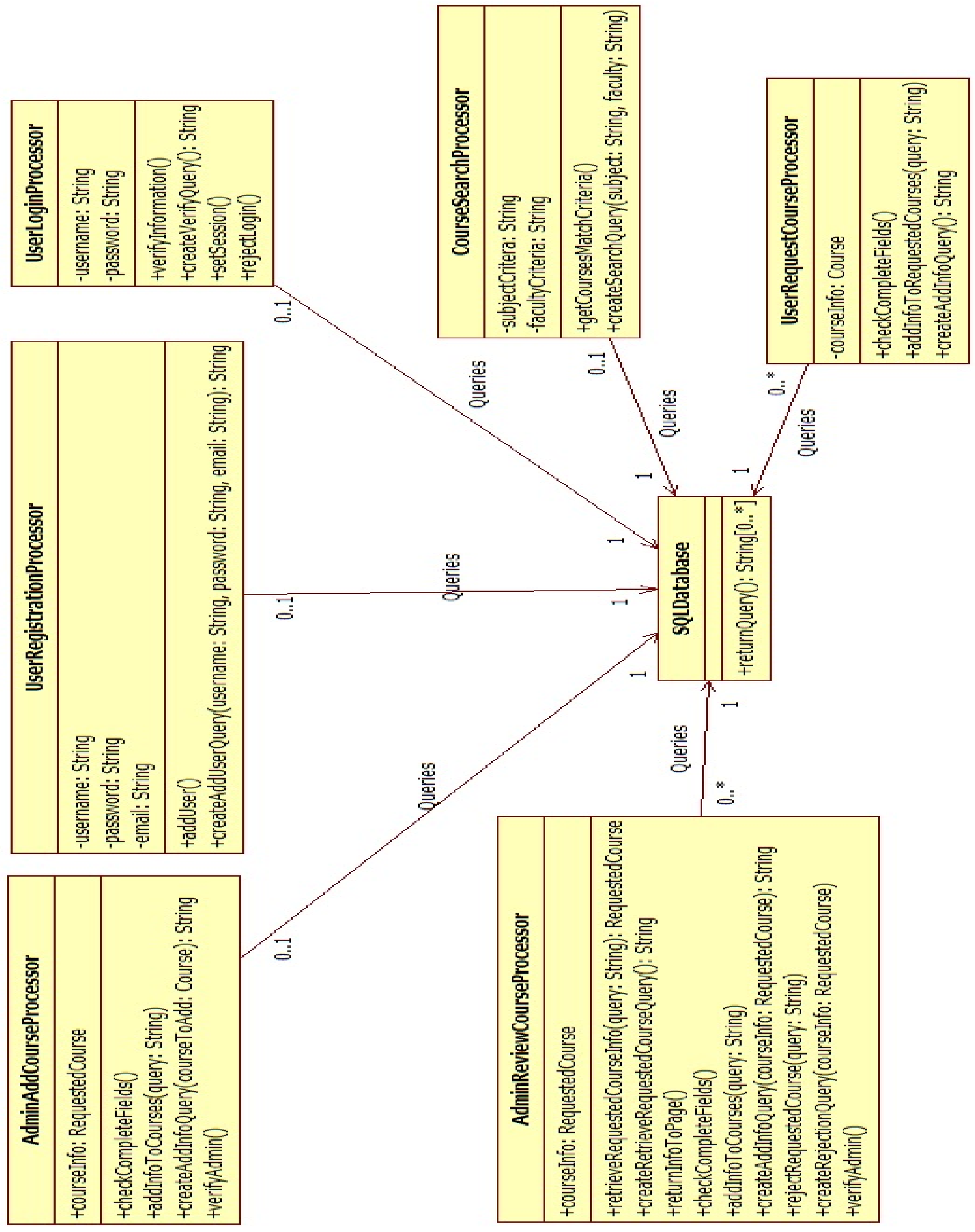

Student
Presentation Layer
Data Bank
Admin
Request Class addition
store request in admin class table
student fills out form with necessary infromation
load user requests table with requested courses
Request table of requested class aditions
Request table of requested class additions
load request table
load request table of requested class additions
Verify a class of the requested class adition table by accepting or denying the class request
load the updated requested class addition table add the course to the pool of avaible courses
load updated requested class table
load updated request class addition table
load updated requested class addition table to Student
the requested course is now searchable and able to be added to a Studnents degree
[online diagramming & design] creately.com

Use Case Change taken Course List: Scenario 2

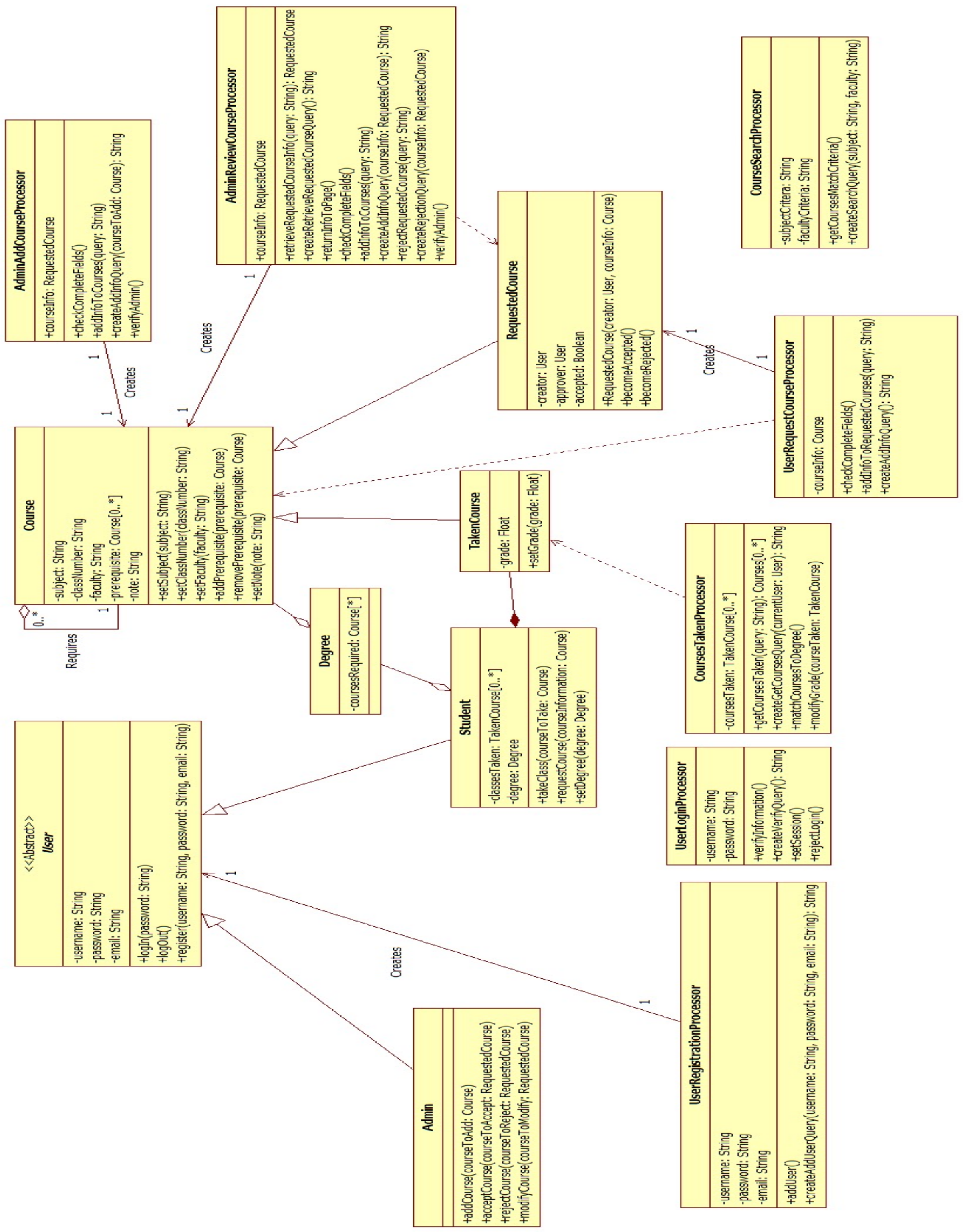

:course search
:student course page
:Student Info
display the students taken couses as well as course yet to be taken and are required
request list of student current degree status
return List of courses taken in students current degree
student searchs for courses by faculty and subject
aviable courses
display search results for student to review and student selects a course to add to degree
verify student selection of course based on student infromation
verify the course has not been taken by the user and that the grade entered is a passing grade, if all pass add the course to the students taken courses list
update students degree table
[online diagramming & design] creately.com

C. Class Diagrams:

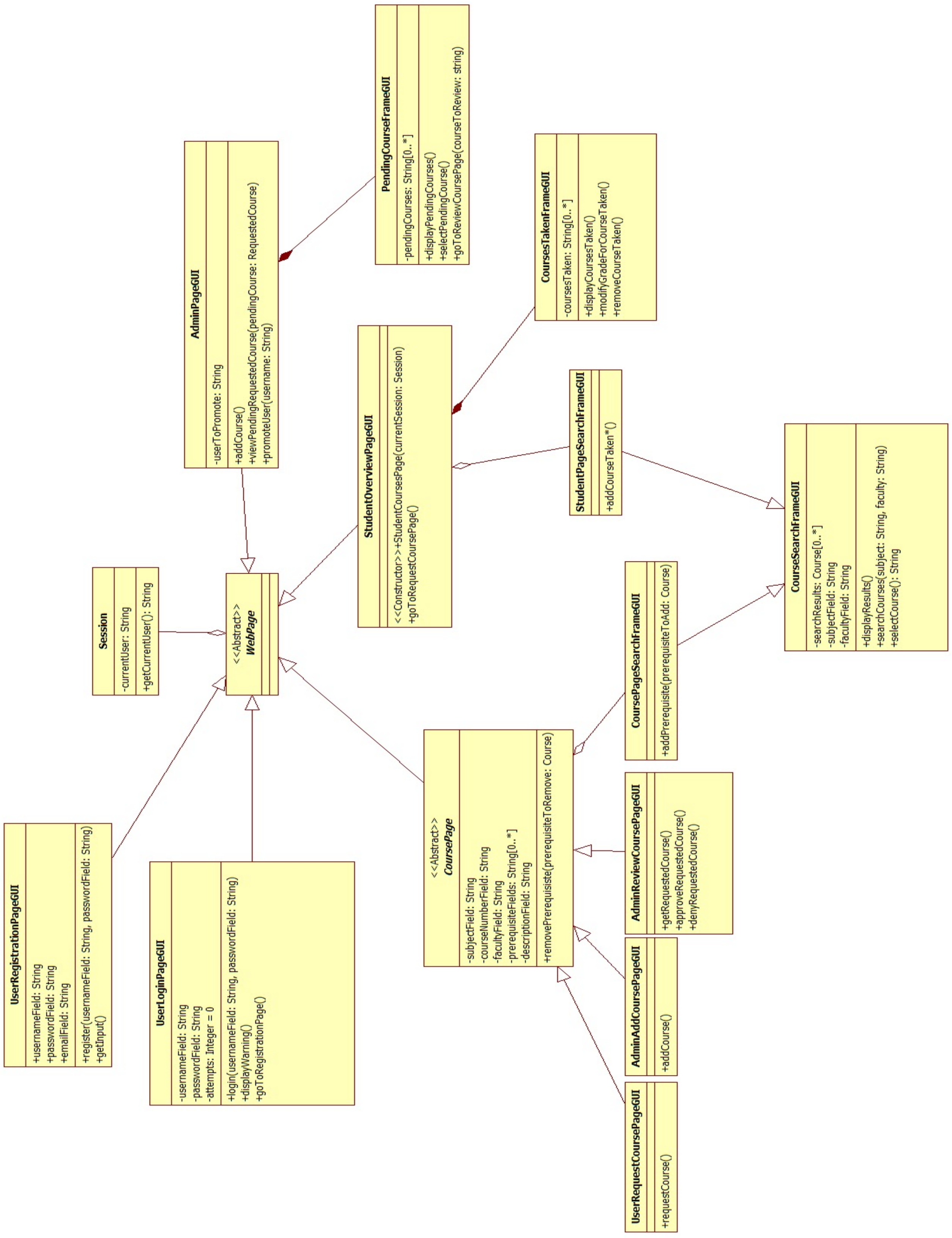
Database Analysis


UserLoginProcessor
-username: String
-password: String
+verifyInformation()
+createVerifyQuery(): String
+setSession()
+rejectLogin()
CourseSearchProcessor
-subjectCritera: String
-facultyCriteria: String
+getCoursesMatchCriteria()
+createSearchQuery(subject: String, faculty: String)
UserRequestCourseProcessor
-courseInfo: Course
+checkCompleteFields()
+addInfoToRequestedCourses(query: String)
+createAddInfoQuery(): String
UserRegistrationProcessor
-username: String
-password: String
-email: String
+addUser()
+createAddUserQuery(username: String, password: String, email: String): String
SQLDatabase
+returnQuery(): String[0..*]
AdminAddCourseProcessor
+courseInfo: RequestedCourse
+checkCompleteFields()
+addInfoToCourses(query: String)
+createAddInfoQuery(courseToAdd: Course): String
+verifyAdmin()
AdminReviewCourseProcessor
+courseInfo: RequestedCourse
+retrieveRequestedCourseInfo(query: String): RequestedCourse
+createRetrieveRequestedCourseQuery(): String
+returnInfoToPage()
+checkCompleteFields()
+addInfoToCourses(query: String)
+createAddInfoQuery(courseInfo: RequestedCourse): String
+rejectRequestedCourse(query: String)
+createRejectionQuery(courseInfo: RequestedCourse)
+verifyAdmin()
Queries
Queries
Queries
Queries
Queries
Queries
0..1
0..1
0..*
0..1
0..1
0..*
1
1
1
1
1
1

Processing

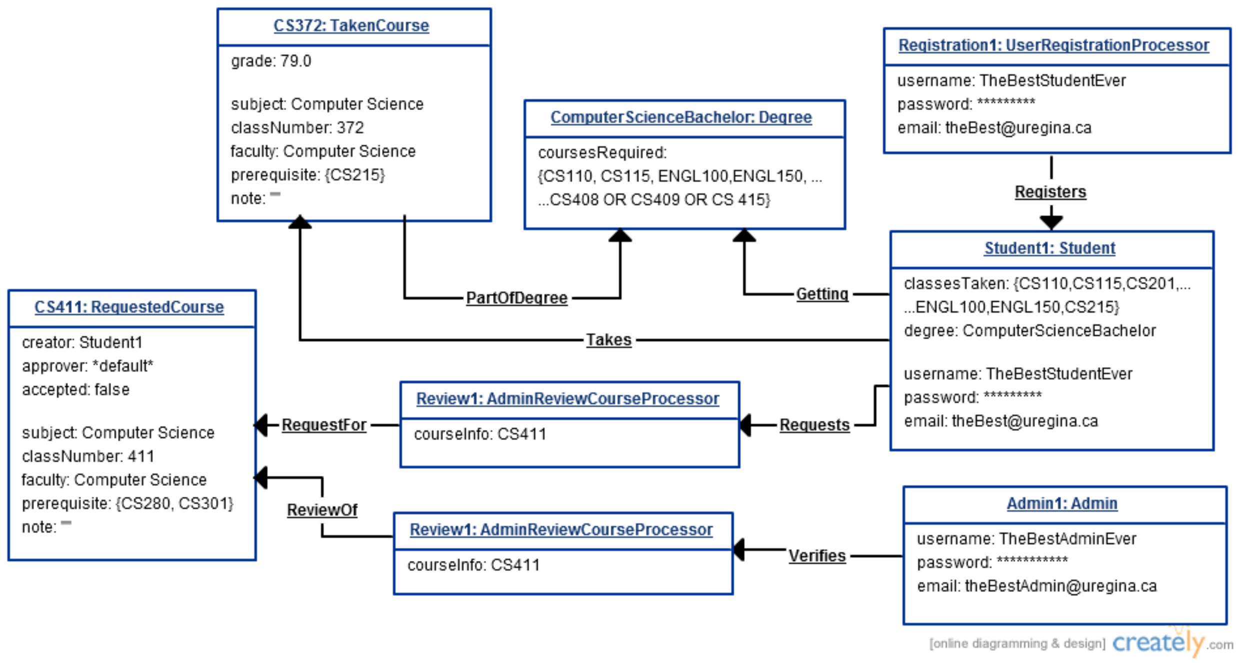

AdminAddCourseProcessor
+courseInfo: RequestedCourse
+checkCompleteFields()
+addInfoToCourses(query: String)
+createAddInfoQuery(courseToAdd: Course): String
+verifyAdmin()
AdminReviewCourseProcessor
+courseInfo: RequestedCourse
+retrieveRequestedCourseInfo(query: String): RequestedCourse
+createRetrieveRequestedCourseQuery(): String
+returnInfoToPage()
+checkCompleteFields()
+addInfoToCourses(query: String)
+createAddInfoQuery(courseInfo: RequestedCourse): String
+rejectRequestedCourse(query: String)
+createRejectionQuery(courseInfo: RequestedCourse)
+verifyAdmin()
CourseSearchProcessor
-subjectCritera: String
-facultyCriteria: String
+getCoursesMatchCriteria()
+createSearchQuery(subject: String, faculty: String)
RequestedCourse
-creator: User
-approver: User
-accepted: Boolean
+RequestedCourse(creator: User, courseInfo: Course)
+becomeAccepted()
+becomeRejected()
UserRequestCourseProcessor
-courseInfo: Course
+checkCompleteFields()
+addInfoToRequestedCourses(query: String)
+createAddInfoQuery(): String
Course
-subject: String
-classNumber: String
-faculty: String
-prerequisite: Course[0..*]
-note: String
+setSubject(subject: String)
+setClassNumber(classNumber: String)
+setFaculty(faculty: String)
+addPrerequisite(prerequisite: Course)
+removePrerequisite(prerequisite: Course)
+setNote(note: String)
TakenCourse
-grade: Float
+setGrade(grade: Float)
CoursesTakenProcessor
-coursesTaken: TakenCourse[0..*]
+getCoursesTaken(query: String): Courses[0..*]
+createGetCoursesQuery(currentUser: User): String
+matchCoursesToDegree()
+modifyGrade(courseTaken: TakenCourse)
Degree
-coursesRequired: Course[*]
Student
-classesTaken: TakenCourse[0..*]
-degree: Degree
+takeClass(courseToTake: Course)
+requestCourse(courseInformation: Course)
+setDegree(degree: Degree)
UserLoginProcessor
-username: String
-password: String
+verifyInformation()
+createVerifyQuery(): String
+setSession()
+rejectLogin()
<<Abstract>>
User
-username: String
-password: String
-email: String
+logIn(password: String)
+logOut()
+register(username: String, password: String, email: String)
Admin
+addCourse(courseToAdd: Course)
+acceptCourse(courseToAccept: RequestedCourse)
+rejectCourse(courseToReject: RequestedCourse)
+modifyCourse(courseToModify: RequestedCourse)
UserRegistrationProcessor
-username: String
-password: String
-email: String
+addUser()
+createAddUserQuery(username: String, password: String, email: String): String
Creates
Creates
Creates
Creates
Requires
0..*
1
1
1
1
1
1
1
1
1
1

Interface Analysis

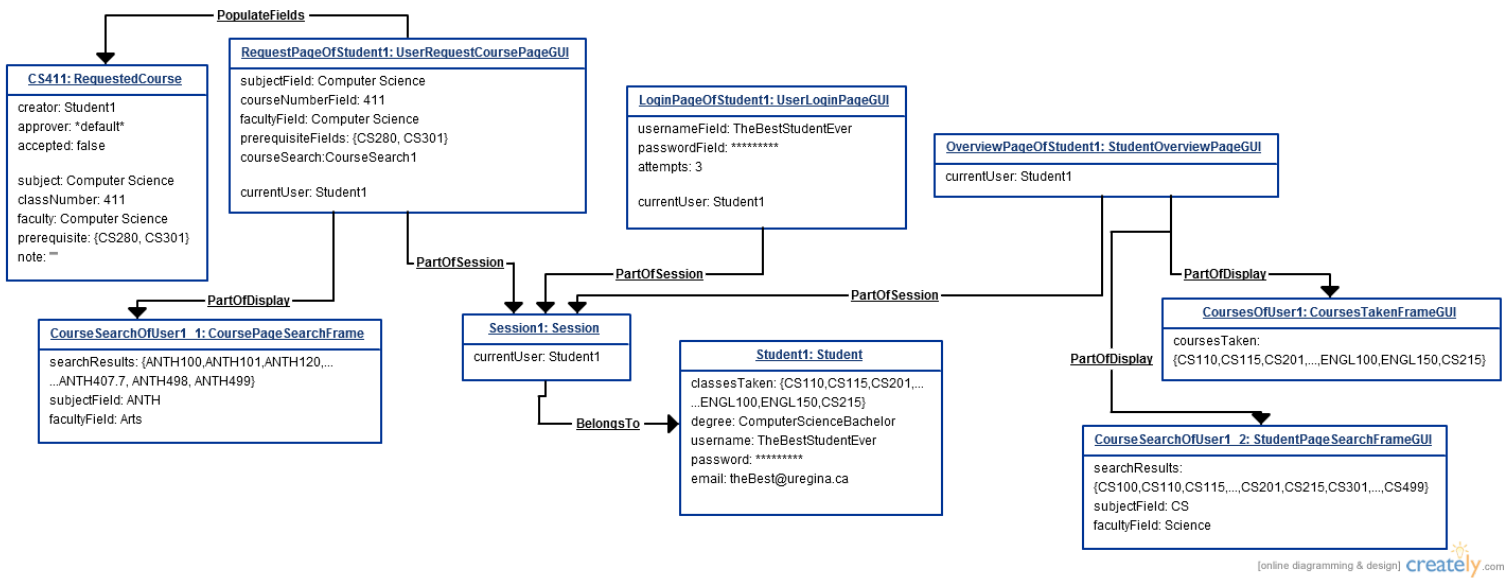


D. Object Diagrams:

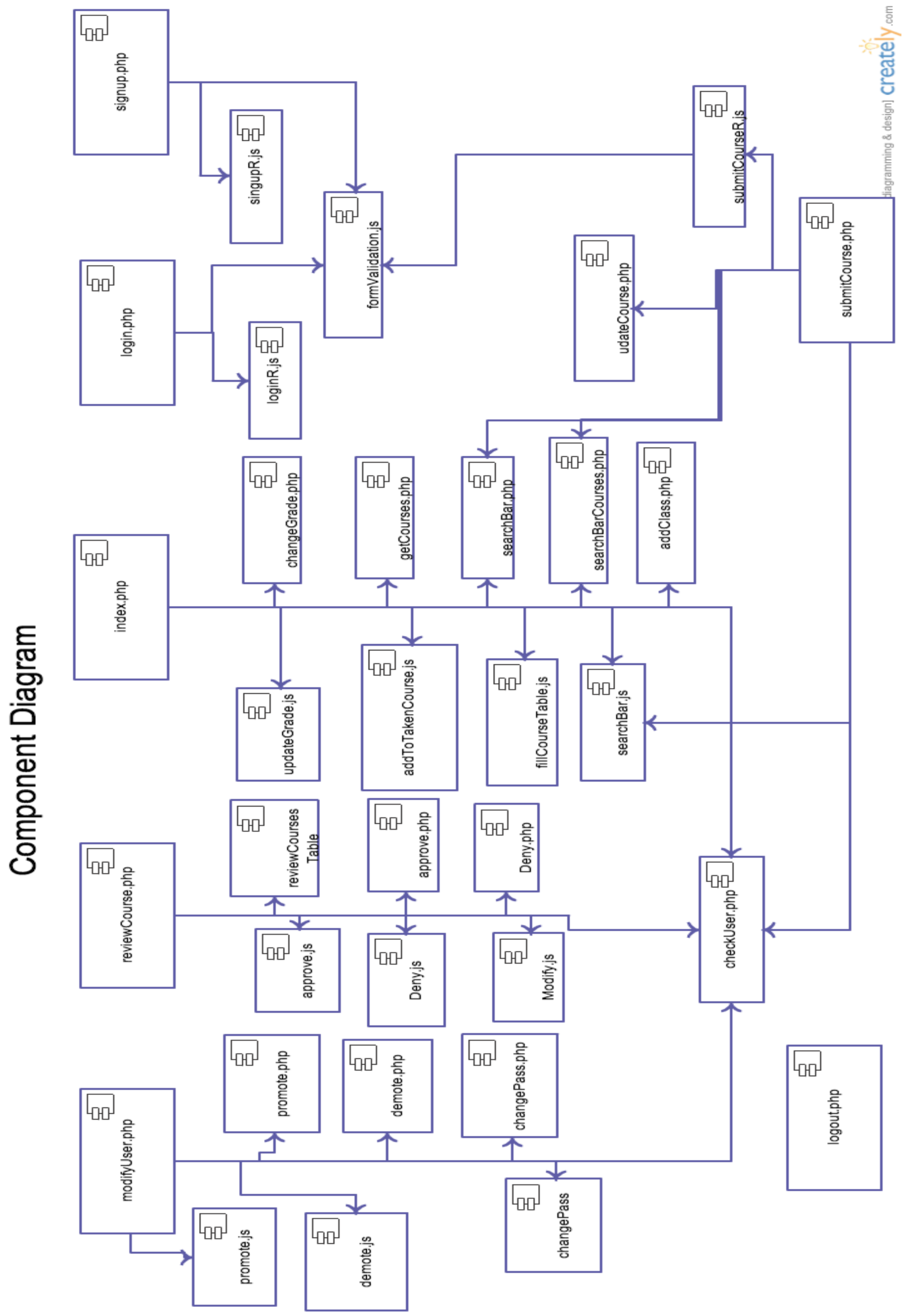
Instance of Student registration and simple Course request+verification


CS372: TakenCourse
grade: 79.0
subject: Computer Science
classNumber: 372
faculty: Computer Science
prerequisite: {CS215}
note: ""
ComputerScienceBachelor: Degree
coursesRequired:
{CS110, CS115, ENGL100,ENGL150, ...
...CS408 OR CS409 OR CS 415}
Registration1: UserRegistrationProcessor
username: TheBestStudentEver
password: *********
email: theBest@uregina.ca
Registers
Student1: Student
classesTaken: {CS110,CS115,CS201,...
...ENGL100,ENGL150,CS215}
degree: ComputerScienceBachelor
username: TheBestStudentEver
password: *********
email: theBest@uregina.ca
PartOfDegree
Getting
Takes
CS411: RequestedCourse
creator: Student1
approver: *default*
accepted: false
subject: Computer Science
classNumber: 411
faculty: Computer Science
prerequisite: {CS280, CS301}
note: ""
RequestFor
Review1: AdminReviewCourseProcessor
courseInfo: CS411
Requests
ReviewOf
Review1: AdminReviewCourseProcessor
courseInfo: CS411
Verifies
Admin1: Admin
username: TheBestAdminEver
password: ***********
email: theBestAdmin@uregina.ca
[online diagramming & design] creately.com

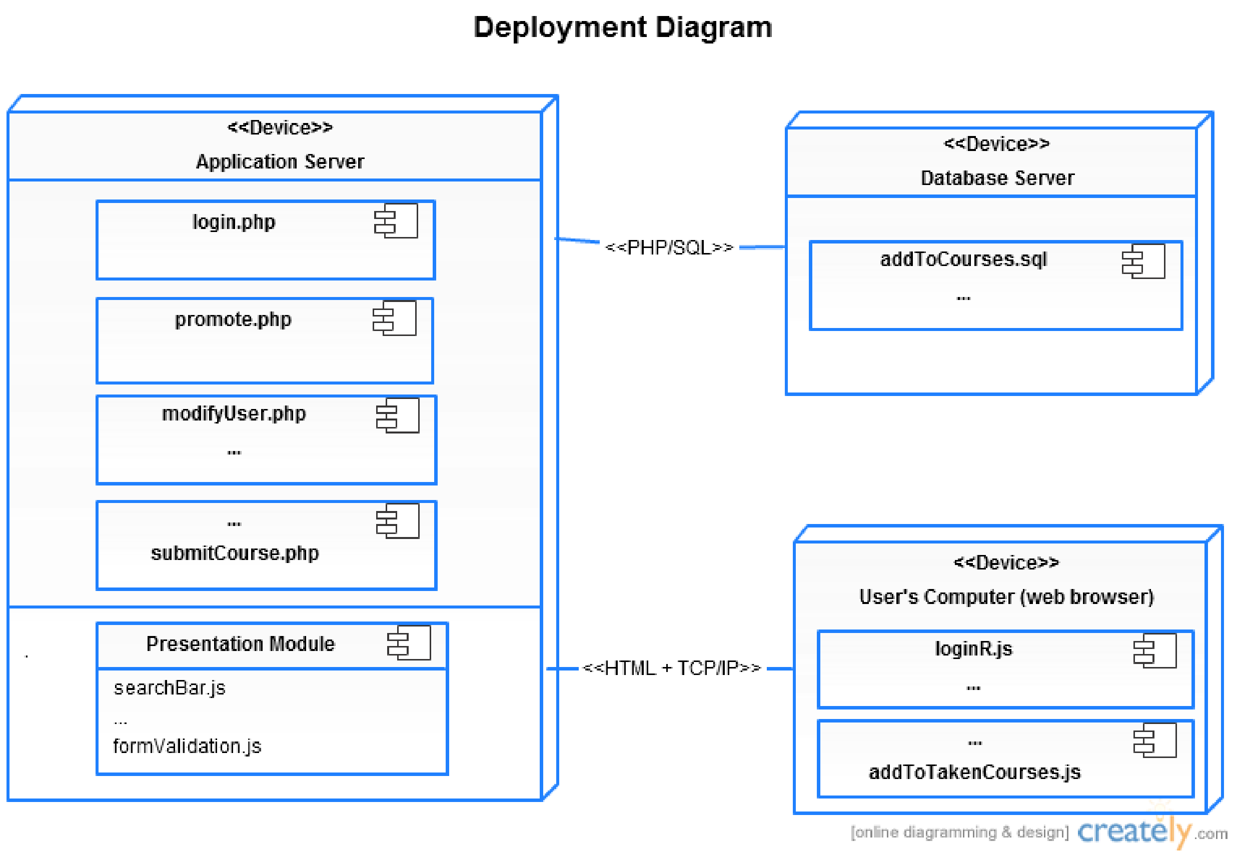
Instance of Session and *Frame classes


PopulateFields
CS411: RequestedCourse
creator: Student1
approver: *default*
accepted: false
subject: Computer Science
classNumber: 411
faculty: Computer Science
prerequisite: {CS280, CS301}
note: ""
RequestPageOfStudent1: UserRequestCoursePageGUI
subjectField: Computer Science
courseNumberField: 411
facultyField: Computer Science
prerequisiteFields: {CS280, CS301}
courseSearch:CourseSearch1
currentUser: Student1
LoginPageOfStudent1: UserLoginPageGUI
usernameField: TheBestStudentEver
passwordField: *********
attempts: 3
currentUser: Student1
OverviewPageOfStudent1: StudentOverviewPageGUI
currentUser: Student1
PartOfSession
PartOfSession
PartOfSession
PartOfDisplay
PartOfDisplay
PartOfDisplay
CourseSearchOfUser1_1: CoursePageSearchFrame
searchResults: {ANTH100,ANTH101,ANTH120,...
...ANTH407.7, ANTH498, ANTH499}
subjectField: ANTH
facultyField: Arts
Session1: Session
currentUser: Student1
BelongsTo
Student1: Student
classesTaken: {CS110,CS115,CS201,...
...ENGL100,ENGL150,CS215}
degree: ComputerScienceBachelor
username: TheBestStudentEver
password: *********
email: theBest@uregina.ca
CoursesOfUser1: CoursesTakenFrameGUI
coursesTaken:
{CS110,CS115,CS201,...,ENGL100,ENGL150,CS215}
CourseSearchOfUser1_2: StudentPageSearchFrameGUI
searchResults:
{CS100,CS110,CS115,...,CS201,CS215,CS301,...,CS499}
subjectField: CS
facultyField: Science
[online diagramming & design] creately.com

E. Component Diagram:



F. Deployment Diagram:



4. UML Tools

For our project we used two distinct UML diagram tools - StarUML and Creately.

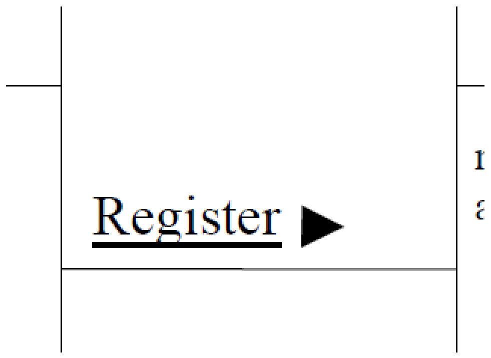
StarUML: This tool was used to create our use case diagrams, and class diagrams. StarUML is a powerful and free UML toolset that facilitates the creation of 10 different types of UML diagrams. Overall, we found it to be fairly simple to learn, and easy to use. The tool allows for extensibility in the form of plugins, and supports a Model Driven Architecture (MDA) approach natively.

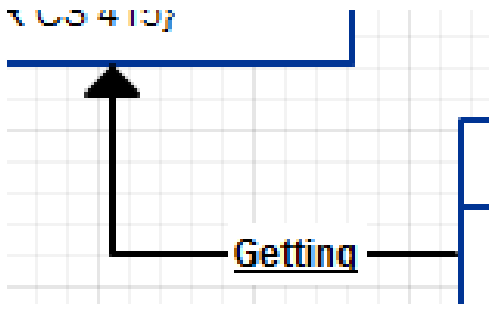
StarUML has a very large problem for our purposes however - the diagrams it generates are not at all visually appealing. It seems that emphasis is put completely on functionality of this UML tool - it has an incredible number of options and properties that can be used to create very complex diagrams. This comes at the cost of visual fidelity, since the diagrams it produces are very limited in terms of what can be done to make them look professional. They generally have a sharp, single colour, cell shaded appearance that makes the diagrams appear to have been made in a very old program. This is why, for the most part, we chose to use our second UML tool whenever possible:

Creately: This is an online diagram creation application that gives users the ability to easily create diagrams (including UML diagrams) in their web browser. It contains many templates for commonly used diagrams of different kinds, and has a very intuitive interface. It can be purchased monthly if certain more advanced features are needed, however we chose to create free accounts and utilize only the basic functionality that this provides. This tool was used to create our sequence diagrams, object diagrams, deployment diagram, and component diagram.

Creately is very simple - when compared to StarUML - in terms of sophistication. It has very limited choices of attributes that can be set for the different components of diagrams, and does absolutely no code generation with the diagrams created. It is effectively a visual tool only, but it does very well in this respect. The diagrams created in Creately can be easily modified to fit new themes, appear to be very polished and professional visually, and can be edited easily by anyone with access to a web browser. Since code generation was not part of our requirements when searching for a UML tool, we opted to use one that would produce documents of a higher aesthetic quality than that of StarUML.

Since the functionality of Creately is limited, certain parts of our diagrams to not match up with the conventions laid out in class. This is because the tool did not allow us to edit the necessary components - they were not programmed into the application to begin with. This can be seen in our object diagrams:





The requested format for links in an object diagram is the image on the left. This was not possible with Creately, and so was approximated as closely as possible (pictured on the right). Another issue we had with Creately was with the Component diagram. The graphics syntax taught in class were different from the ones used in Creately, so our components do not appear as they should.

This shows how both tools have usefulness but lacked important aspects preventing us from making all of our diagrams using a single tool.

5. Technical Documentation

A. Programming languages

1. HTML5

We used HTML5 to create our web pages that are displayed on the students machine. HTML5 is the latest adaptation of HTML making it much more versatile. It enables cleaner and neater code from being able to use the semantic HTML5 elements. Since HTML5 is being used on more and more websites it creates consistency between sites allowing developers and programmers to understand the code a lot better. Using HTML5 helped us create a more versatile website that is consistent, well formatted, and easy to understand.

2. JavaScript

We used JavaScript on our web pages to make them more functional in their responses to the students actions. Since JavaScript is on the client-side it can run functions immediately instead of having to contact the server to run functions. JavaScript is relatively simple to learn and can be easily implemented. A disadvantage of JavaScript are its lack of security since it is run on the student's computer.

3. PHP

We used PHP to interact with our MySQL database. It enabled us to pass information back and forth between the students browser, the web server and the database with relative ease. PHP provides of a variety of security functions to ensure secure transfer of information. A disadvantage of PHP is that it is not object oriented meaning that the code can't be as organized as compared to it being object oriented.

4. CSS

We used CSS on the web pages to create the look and format of them. It was used to create a professional and clean looking website that is well formatted. CSS was a good use since it has multiple libraries that can provide a variety of designs and formats to make a professional looking website. Using CSS also provides cross-browser functionality to ensure the design and format of the website is available on multiple browsers. A drawback of using CSS is that we are limited to the frameworks and designs available in it.

B. Reused algorithms and programs

1. JQuery

JQuery is a JavaScript library that simplifies the use of JavaScript code. It provides more advanced and cross browser functions that minimize any browser incompatibilities. We used JQuery on our web pages to minimize the amount of code and to enhance the functionality of the pages.

2. AJAX

AJAX enables the ability to send and retrieve data from the server without interfering the display of the current page. This means that information can be passed back and forth in the background to and from the server without inconvenience to the student. We used AJAX on our web pages to pass information back and forth on our web pages to the server to keep everything up to date as the student edits their courses and degree information.

3. PHP CRYPT

The PHP crypt function encrypts a string using DES, Blowfish, or MD5 algorithms. We used the crypt function to encrypt the passwords for students accounts when being passed back and forth from the students browser to the web server. The function enabled us to keep the website secure and safe for all users.

4. Reused Code

We reused the HTML code from the computer science web page of the bachelor of computer science degree. The code was used as the format of the class degree table on the students degree overview page.

Code was taken from: http://www.cs.uregina.ca/UndergradProgram/programs/academpro.html

- JQuery DataSheet

5. Mechanize

Mechanize is a cross-platform library that emulates a web browser. In our case we used the Python version to scrape an initial course set from the University course catalog web pages. We then used custom Python scripts to analyze the gathered data and generate SQL statements to populate our database with an initial set of course information.

C. Tools, environments, Web services, etc.

System information for the webserver and database server:

Ubuntu Server 12.04.4 LTS 64 Bit

1 GB RAM

21GB HD

Intel(R) Core(TM) i3-3220T CPU @ 2.80GHz (One Thread)

Web Server: Apache 2.2.22

Database: mysql Ver 14.14 Distrib 5.5.35

1. WinSCP

We used WinSCP to connect to the web server and edit our HTML, JavaScript and PHP documents

2. HeidiSQL

HeidiSQL was used to connect to the database and create and edit the database tables

3. Notepad++

Notepad++ was used to edit HTML, JavaScript, and PHP files.

4. Google Drive

All of the group documentations and files were uploaded or created on Google Drive to a shared folder between all the group members. Group members were able to edit, modify, and create documents. Using Google Drive ensured that only one document was created between group members and the finished document was worked on and built by multiple members.

5. Eclipse

Eclipse was used to edit HTML, JavaScript and PHP files.

6. PuTTY

PuTTY was used to connect to the webserver and database server to edit the various files and to test PHP code.

7. MySQL Workbench

MySQL Workbench was used to generate our ERD.

8. Sublime Text 2

Sublime Text 2 was used to edit HTML, JavaScript and PHP files.

9. Drupal Framework

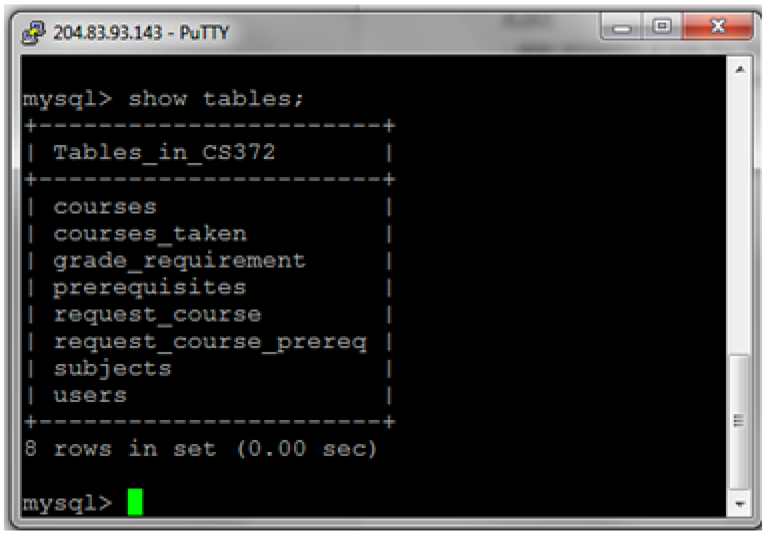
Drupal is a free open source web based content management system used as a back end framework; providing structure to common web structures such as menu and user management tools. Drupal is coded with PHP ensuring cross platform compatibility. Drupal offers many features to developers such as code generation, automatic web page theming and responsive web page structuring. To be able to take full advantage of the benefits of Drupal however, requires extensive knowledge of Drupal syntax and file structure. Our team attempted to utilize Drupal to aid our development but found the learning curve too steep and decided to abandon this tool early on.

10. Google Chrome

We used Google Chrome and the Google Chrome developer tools to view and run the test cases for our implemented website. The developer tools made it simple to check the performance and network activity of the site, and to check and debug the JavaScript and PHP code.

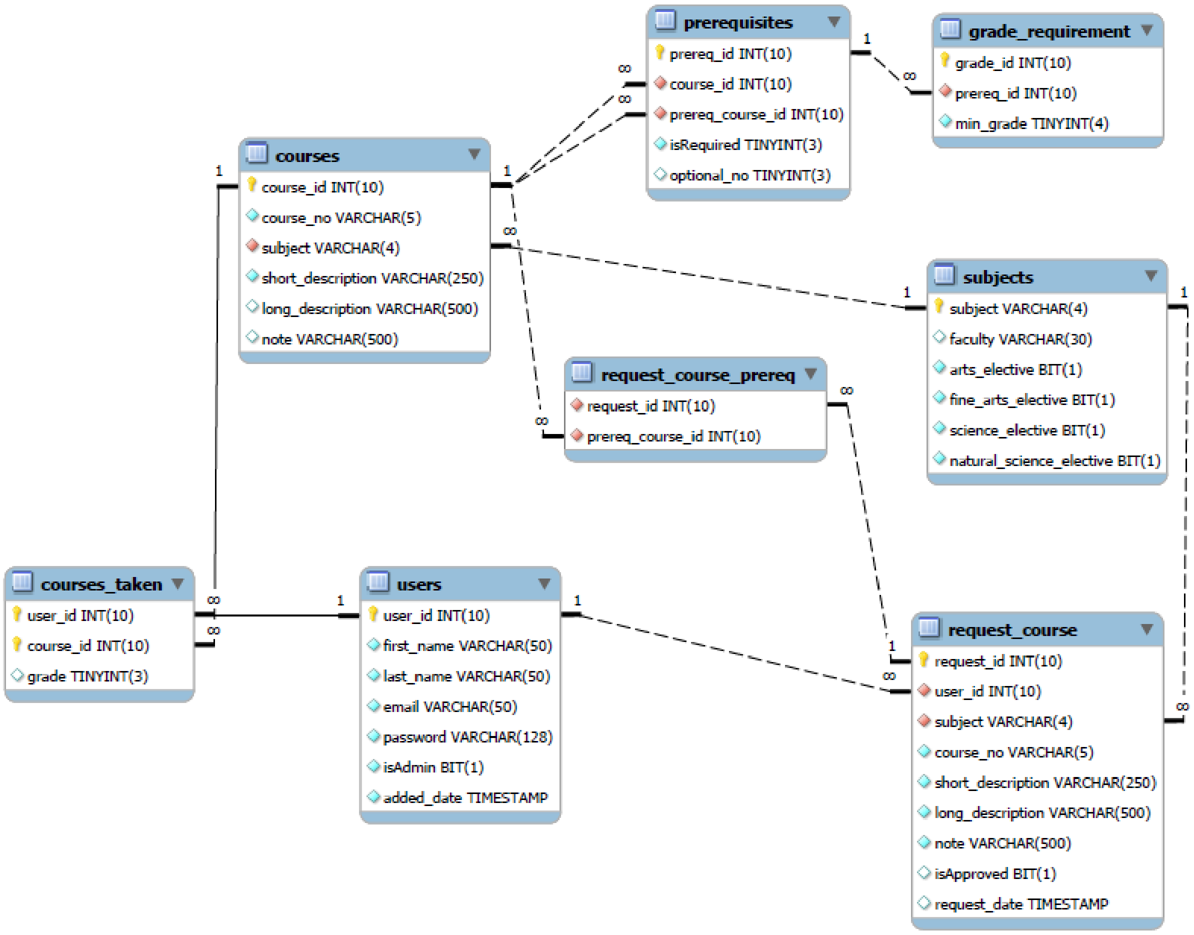
D. Database management system.

We used MySQL as our database management system. MySQL is one of the most used database management systems making it a good choice. Since it’s so widely used there are a lot of tutorials on how to use it and there are very few bugs. Another reason why we chose a MySQL database was because we already learnt how to use it from CS215.



```
mysql> show tables;
+-----------------------+
| Tables_in_CS372       |
+-----------------------+
| courses               |
| courses_taken         |
| grade_requirement     |
| prerequisites         |
| request_course        |
| request_course_prereq |
| subjects              |
| users                 |
+-----------------------+
8 rows in set (0.00 sec)

mysql>
```



6. User Documentation

R.A.P or Registration Assistance Program is a degree tracking system is a system meant to aid University of Regina students in tracking and calculating their degree. These include tracking courses taken, courses needed for completion on the degree and lastly course grades and current GPA. Users can access R.A.P at IP addresses 204.83.93.143:10080. R.A.P has three main pages in which the user can access as well as two additional pages for administrators. This user manual will describe the operations and basic actions available on each page such as

- Regular Users

A. Logging in

B. Signing up

C. Viewing your degree

D. Adding a course and/or grade to your degree

E. Request a Course to be added to Database

F. Viewing requested course table

G. Logging out

- Administrators

A. Reviewing the requested course additions submitted by users

B. Adding a course to the database

C. Giving a user administrative rights

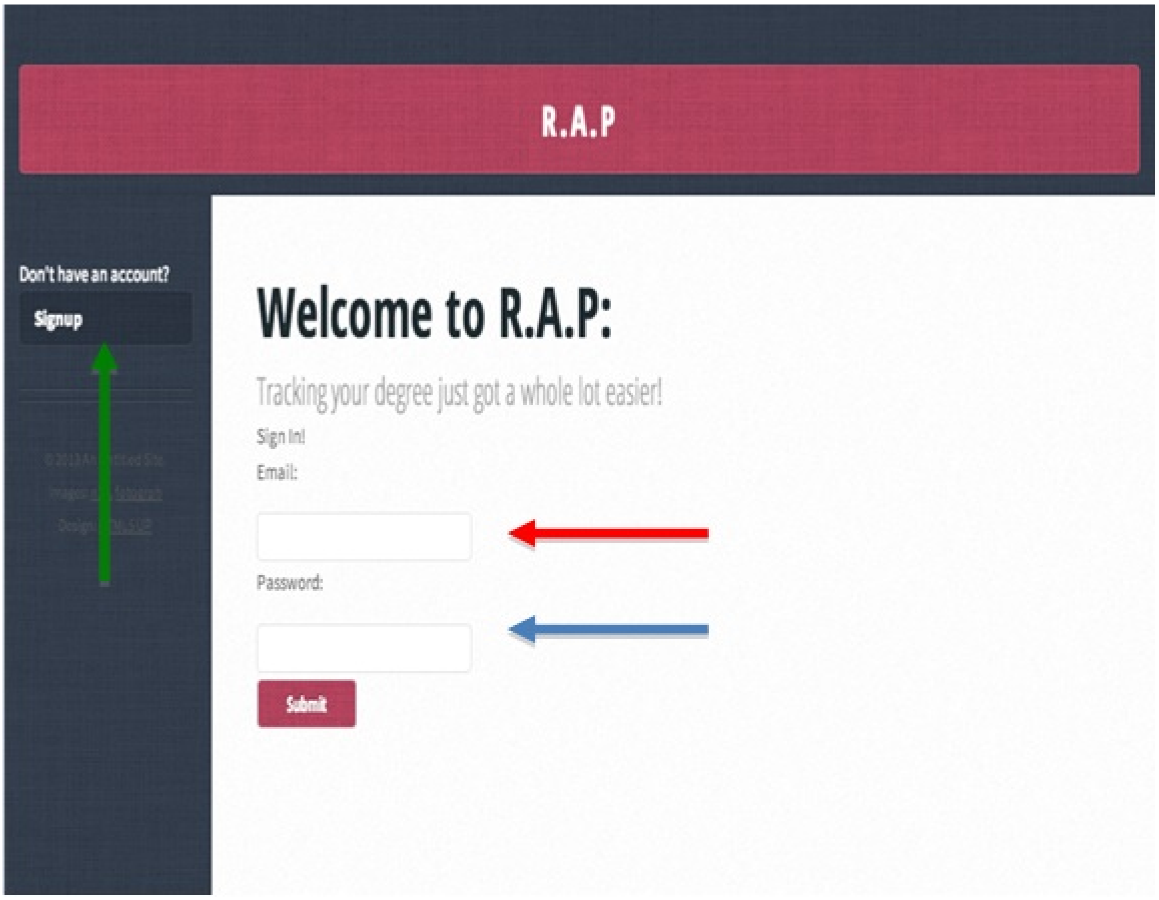
D. Changing a users password

E. Logging out

Regular Users actions:

A. Logging in

All users are able to accesses rap through the IP address of http://204.83.93.143:10080. Users will be brought to figure 1 and are able to sign in with an email indicated by the red arrow and password specified by the blue arrow. Both email and password are consistent with the indicated email and password combination specified at the time of signup. Upon ensuring correct information and clicking the button labeled "submit," the user will be taken to the home page. If the information is incorrect error messages will be displayed above the field that is incorrect. If the user is new to R.A.P., they can navigate to the sign up page indicated by the green arrow.



Welcome to R.A.P:

Tracking your degree just got a whole lot easier!

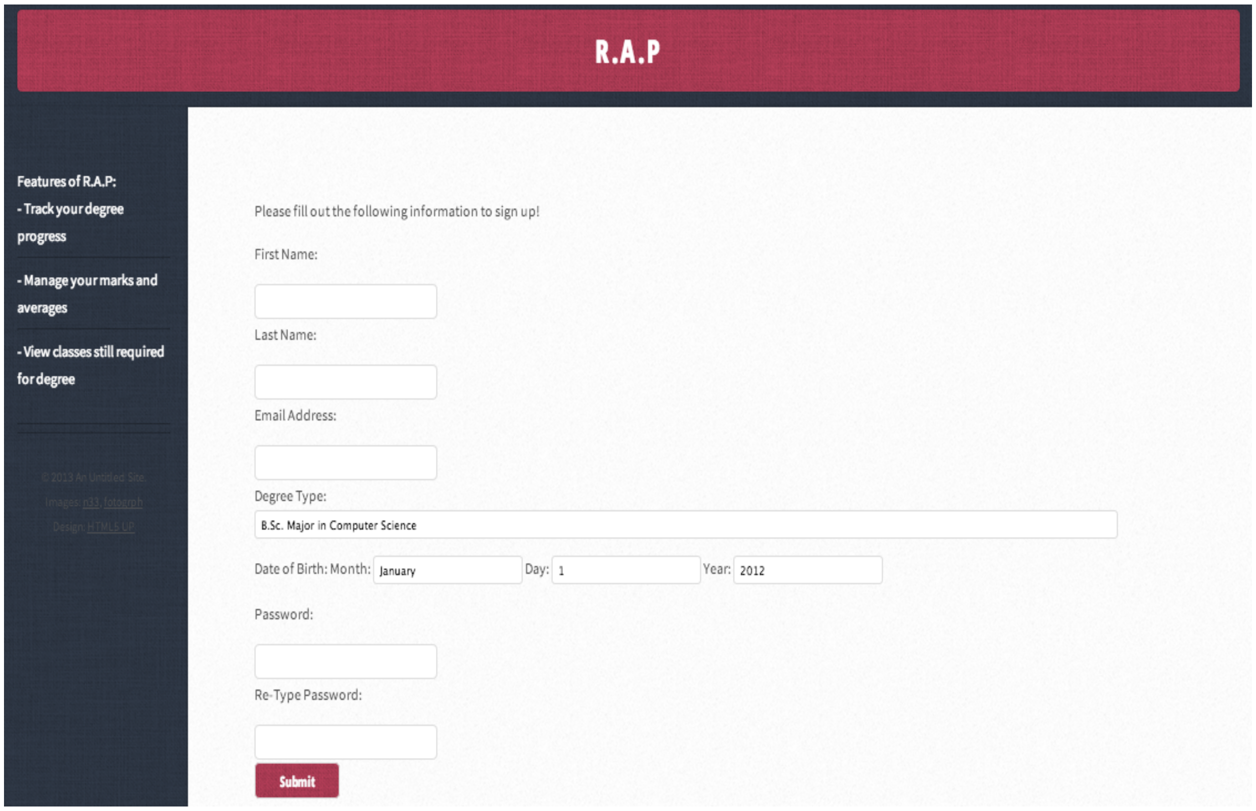
Sign In!

Email:

Password:

B. <u>Sign Up</u>

Users can sign up for R.A.P by filling in the information on the sign up page that can be accessed through the login page under the sign up tab. Users must enter a first name, last name and valid email address in the input boxes. The user must select a degree from the drop down box labeled degree type. They must also enter their date of birth by selecting a month, day and year from the drop down boxes as well as choose a password that must be at least 8 characters long and contain at least one non-letter character. Upon clicking the submit button the user will be redirected to the login page: however, if any information is rejected an error message will be displayed above the corresponding field.



R.A.P

Features of R.A.P:
- Track your degree progress
- Manage your marks and averages
- View classes still required for degree

© 2013 An Untitled Site.
Images: n33, fotogrph
Design: HTML5 UP

Please fill out the following information to sign up!

First Name:

Last Name:

Email Address:

Degree Type: B.Sc. Major in Computer Science

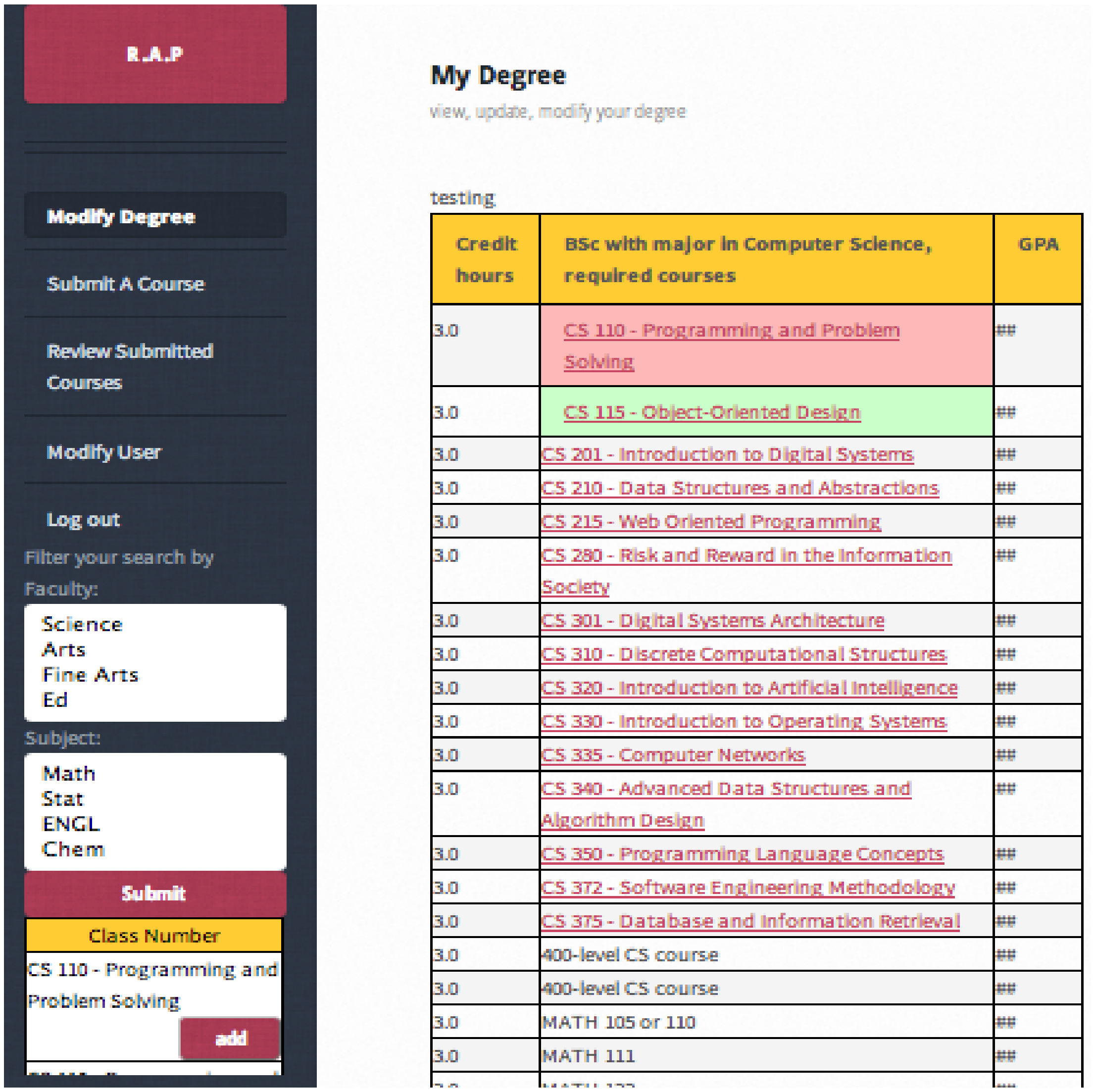
Date of Birth: Month: January Day: 1 Year: 2012

Password:

Re-Type Password:

C. Viewing your degree:

Once logged in, users will be able to view their main page. Users can begin to add courses to their degree from this page. Courses taken will be indicated by the green tint within the course cell. Courses that have yet to be taken are indicated by red tint.



My Degree

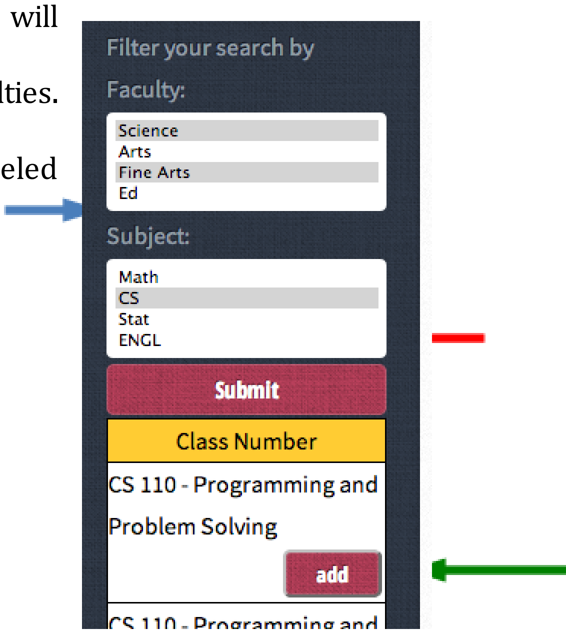
view, update, modify your degree

testing

Credit hours	BSc with major in Computer Science, required courses	GPA
3.0	CS 110 - Programming and Problem Solving	##
3.0	CS 115 - Object-Oriented Design	##
3.0	CS 201 - Introduction to Digital Systems	##
3.0	CS 210 - Data Structures and Abstractions	##
3.0	CS 215 - Web Oriented Programming	##
3.0	CS 280 - Risk and Reward in the Information Society	##
3.0	CS 301 - Digital Systems Architecture	##
3.0	CS 310 - Discrete Computational Structures	##
3.0	CS 320 - Introduction to Artificial Intelligence	##
3.0	CS 330 - Introduction to Operating Systems	##
3.0	CS 335 - Computer Networks	##
3.0	CS 340 - Advanced Data Structures and Algorithm Design	##
3.0	CS 350 - Programming Language Concepts	##
3.0	CS 372 - Software Engineering Methodology	##
3.0	CS 375 - Database and Information Retrieval	##
3.0	400-level CS course	##
3.0	400-level CS course	##
3.0	MATH 105 or 110	##
3.0	MATH 111	##

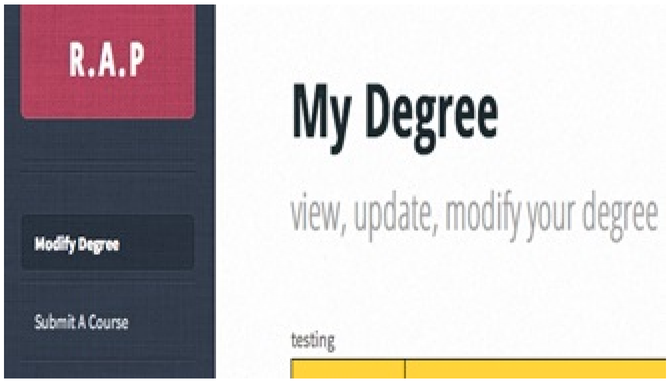
D. <u>Adding a course to a degree</u>

Users will navigate to the side bar indicated by the blue arrow and select one or more faculties by clicking and holding the command key this will populate the subject field with subjects from those faculties. Selecting one or more subjects and clicking the button labeled "submit" will then populate the course table with courses of the selected subject. From here the user can click the "add" button indicated by the green arrow and if the course has not yet been taken the course will then be added to the users taken course list.

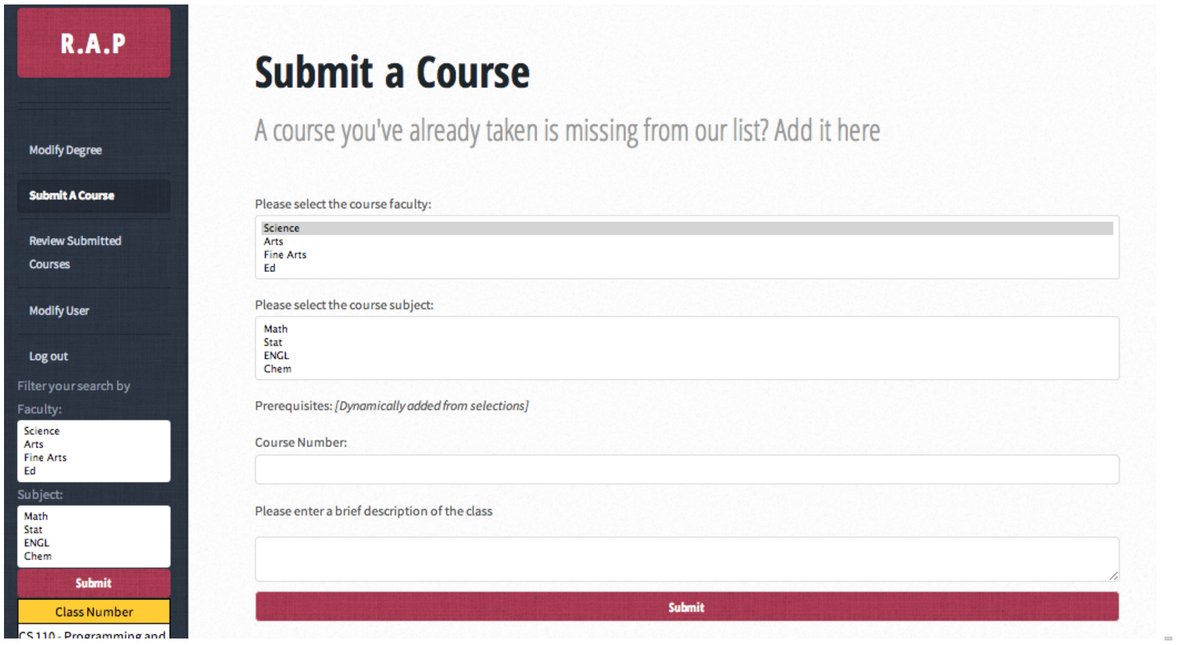


E. Request a Course to be added to Database:

Users can navigate to the "request a course" page through the main page "Submit A Course" tab. This will bring up a page that will allow the user to enter information



pertaining to each course. This information includes to which Faculty and Subject the course belongs, including the prerequisites associated with the course, the course number and a short description of the course. To add prerequisites to your submission follow the same procedure for adding a course to a degree. Upon clicking the add button the course will be shown within the prerequisite area. Upon clicking "submit," and as long as the information is correct, the course will be added to a table for an administrator to review. Users can review their submissions, as well as other users' pending requests in the "Review Submitted Courses" link in the sidebar.



Submit a Course

A course you've already taken is missing from our list? Add it here

Please select the course faculty:
Science
Arts
Fine Arts
Ed

Please select the course subject:
Math
Stat
ENGL
Chem

Prerequisites: *[Dynamically added from selections]*

Course Number:

Please enter a brief description of the class

Submit

F. Review Submitted Courses

Users can review the status of their request for course addition by navigating to the "Review Submitted Courses" tab. This page consists of a table of pending request for course additions. If a user's course is no longer listed, an admin has either accepted the course or rejected the course. In the case that a course is accepted it will be added to the pool of valid courses, and will be available to be added to a user's degree. In the case a course is rejected by the admin, the course will no longer be shown in the request course addition table and will not be available to be added to a degree.

G. Log out

Once a user is done using R.A.P they can log out of their session by navigating to the “log out” tab located in the side bar above the filter course area.

Administrative Actions:

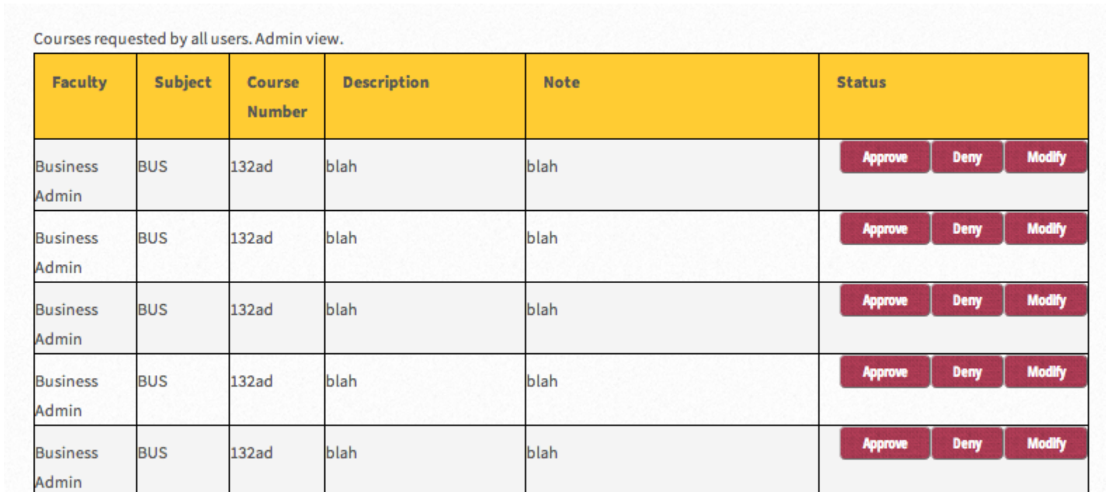
Administrators may perform all the actions of a regular user by following the same procedures. However, administrators have special rights while adding a course to the database, as well as reviewing the requested course addition table. Administrators are also able to promote and demote users to administrators. Administrators are also able to change the password of a user.

A. Adding a Course to the database

Administrators follow the same procedure as regular users when it comes to adding a course to the database on the “Request a Course Addition” page. However, instead of the course being added to the “request a course addition” table it will be added straight to the pool of available courses.

B. Reviewing requested Course Addition table

Administrators will navigate to the same “Review Submitted Courses” tab, in which all users are able to see courses submitted for review. The administrator will have an additional table column however with accept or deny buttons. If the information is correct the administrator can add the course to the pool of available courses by clicking the accept button. Conversely if the information is not correct the administrator can reject and delete the course request by clicking the deny button. Or Administrators can click modify to change the information relating to the course.

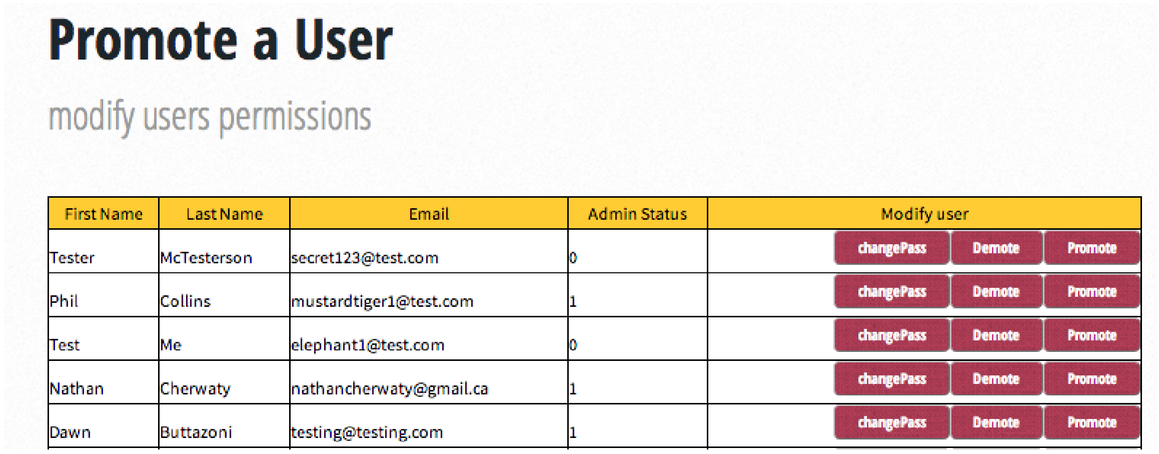


Courses requested by all users. Admin view.

Faculty	Subject	Course Number	Description	Note	Status
Business Admin	BUS	132ad	blah	blah	Approve Deny Modify
Business Admin	BUS	132ad	blah	blah	Approve Deny Modify
Business Admin	BUS	132ad	blah	blah	Approve Deny Modify
Business Admin	BUS	132ad	blah	blah	Approve Deny Modify
Business Admin	BUS	132ad	blah	blah	Approve Deny Modify

C. <u>Giving a user administrative rights:</u>

An administrator can grant user administrative rights by navigating to the "Modify User" tab in the sidebar. This will load a page that populates a table full of users. The administrator can browse all of the users and grant administrative rights to a user by clicking a "promote" button next to a user's name. Administrators can also demote users taking away administrative rights by clicking the "demote" button next to their name.



Promote a User

modify users permissions

First Name	Last Name	Email	Admin Status	Modify user
Tester	McTesterson	secret123@test.com	0	changePass Demote Promote
Phil	Collins	mustardtiger1@test.com	1	changePass Demote Promote
Test	Me	elephant1@test.com	0	changePass Demote Promote
Nathan	Cherwaty	nathancherwaty@gmail.ca	1	changePass Demote Promote
Dawn	Buttazoni	testing@testing.com	1	changePass Demote Promote

A. <u>Changing a users password</u>

Administrators can also change a users password on the same webpage “modify User” by clicking “changePass” button next to the users name and entering the a new password in the pop-up box.

A. <u>Logging out</u>

Administrators can log out the same way regular users log out, by navigating to and clicking the “log out” tab.

7. Software Testing

Software Test Cases:

(a) Correctness testing with some data tests (at least 5 test cases).

1. A required class when added should appear on main screen with correct color
2. An added elective should appear in the correct spot on the main screen
3. The same course should not appear more than once on a degree listing
4. Overall average should be correctly computed and displayed
5. Search menu should correctly filter and display courses

(b) Robustness testing with some incorrect data (at least 5 test cases).

1. Try to login with incorrect password
2. Try to login with incorrect username
3. Try to enter invalid grade
4. Request a missing course that already exists
5. Request a missing course with incorrect data (Admin Rejection)

(c) Performance testing with some benchmarks (at least 5 test cases).

1. Handle 2 number of simultaneous users.
2. Pages should load in at least 2 seconds
3. Search menu should be fast and responsive
4. Database should be able to respond and load requests from the web server in at least 2 seconds

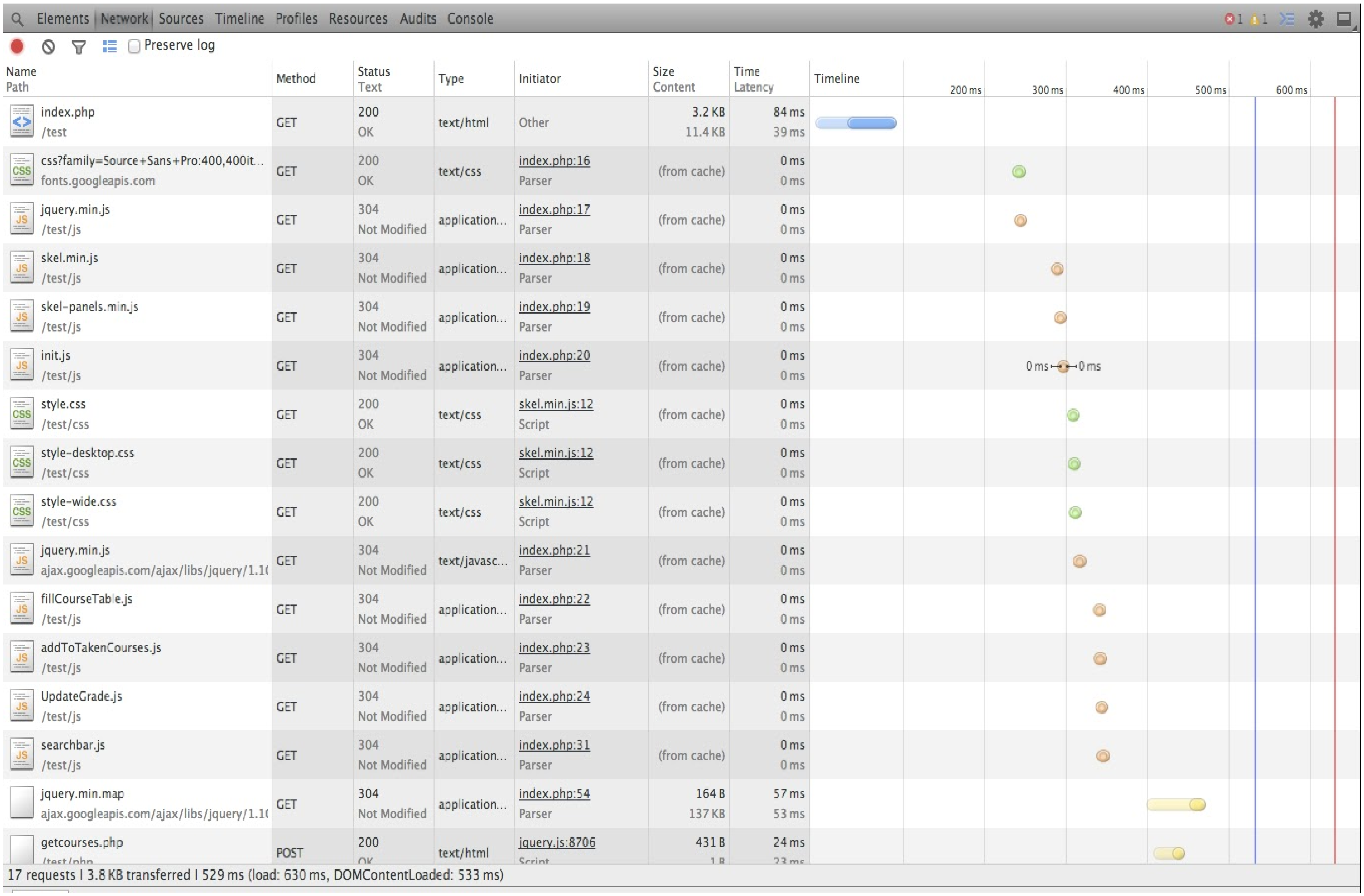
Test Case	Input Data	Output Data	Correct Behaviour
a1	Add CS 110	CS 110 Changes from red to green	Yes
a2	Add ASTR 101	ASTR 101 appears as Natural Science elective	Yes
a3	Add CS 110 twice	It only shows up as CS 110 and not as an elective as well.	Yes
a4	Input class grades	Computed average displayed	Yes
a5	Click Science faculty. Then	Class number only shows CS	Yes

	CS.	classes	

Test Case	Input Data	Output Data	Correct Behaviour
b1	Enter incorrect password and try to login	Re-direct to home page and display incorrect login message	Yes
b2	Enter incorrect username and try to login	Re-direct to home page and display incorrect login message	Yes
b3	Try to enter grade less than 50 or a string	Display incorrect grade error message	Yes
b4	Submit a missing course using the missing course page(Correct data)	Page submits and is added to the administrators add course queue. Admin then accepts valid request and it appears as an addible course	Yes
b5	Submit a missing course using the missing course page(Incorrect data)	Page submits and is added to the administrators add course queue. Admin then rejects invalid request and it is not added as an addible course	Yes

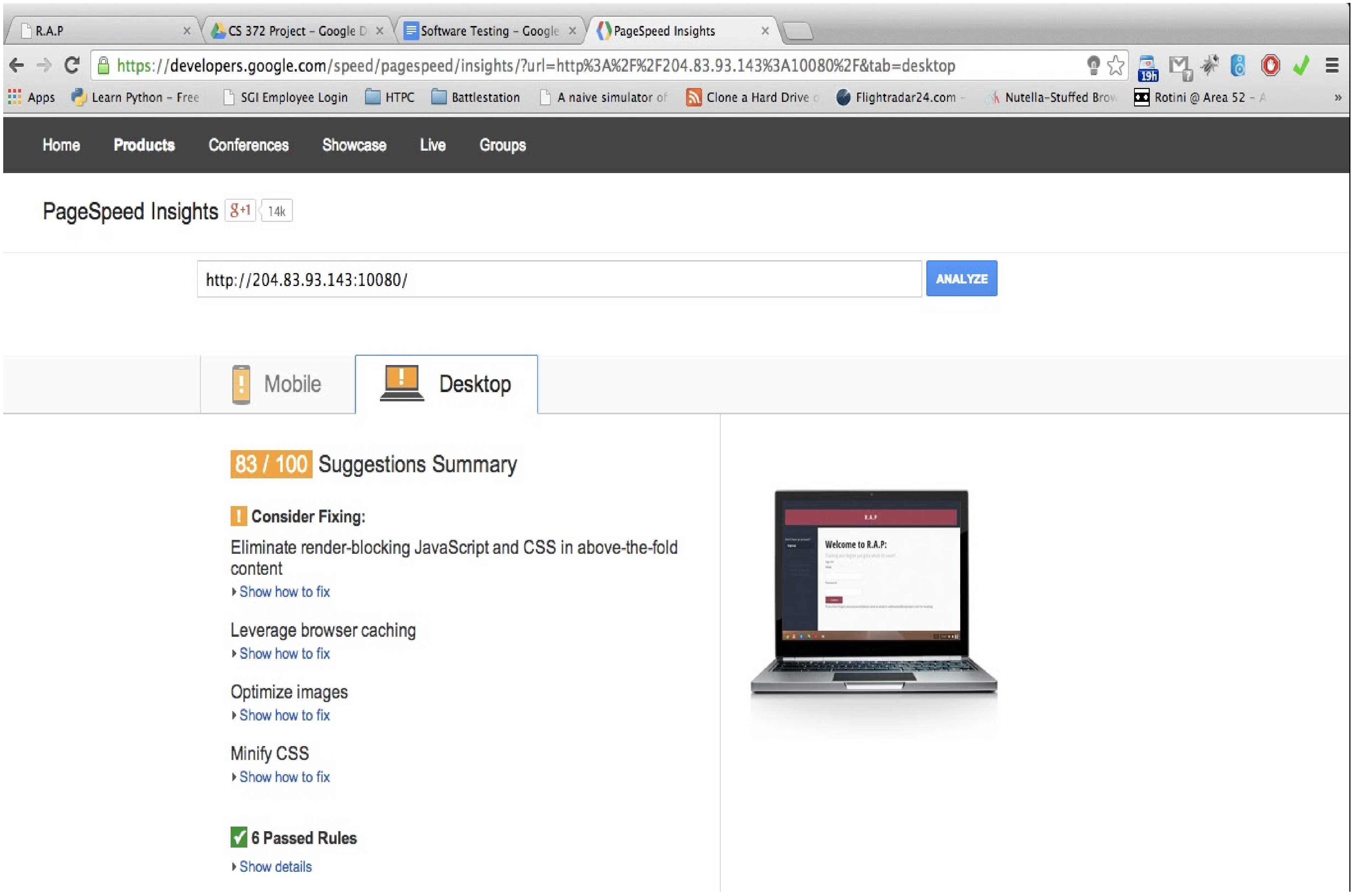
Test Case	Input Data	Output Data	Correct Behaviour
c1	2 users logged in at the same time.	Each user should have full access to the site and not notice diminished performance	Yes
c2	Web page is requested.	Page completely loads in under 2 seconds	Yes
c3	Click a faculty and subject and the appropriate menu.	The menu appropriately loads the list boxes in real time	Yes
c4	Fill in a submit a course web page	After the submit button the next page should load in under 2 seconds	Yes

An extra testing method was used for the efficiency and organization of our code. This was done through Google Developper's "PageSpeed Insights", giving us an overall rating of 83/100, and showing the access speeds for the different pages of our system:



Name Path	Method	Status Text	Type	Initiator	Size Content	Time Latency	Timeline
index.php /test	GET	200 OK	text/html	Other	3.2 KB 11.4 KB	84 ms 39 ms	
css?family=Source+Sans+Pro:400,400it... fonts.googleapis.com	GET	200 OK	text/css	index.php:16 Parser	(from cache)	0 ms 0 ms	
jquery.min.js /test/js	GET	304 Not Modified	application...	index.php:17 Parser	(from cache)	0 ms 0 ms	
skel.min.js /test/js	GET	304 Not Modified	application...	index.php:18 Parser	(from cache)	0 ms 0 ms	
skel-panels.min.js /test/js	GET	304 Not Modified	application...	index.php:19 Parser	(from cache)	0 ms 0 ms	
init.js /test/js	GET	304 Not Modified	application...	index.php:20 Parser	(from cache)	0 ms 0 ms	0 ms 0 ms
style.css /test/css	GET	200 OK	text/css	skel.min.js:12 Script	(from cache)	0 ms 0 ms	
style-desktop.css /test/css	GET	200 OK	text/css	skel.min.js:12 Script	(from cache)	0 ms 0 ms	
style-wide.css /test/css	GET	200 OK	text/css	skel.min.js:12 Script	(from cache)	0 ms 0 ms	
jquery.min.js ajax.googleapis.com/ajax/libs/jquery/1.1(	GET	304 Not Modified	text/javasc...	index.php:21 Parser	(from cache)	0 ms 0 ms	
fillCourseTable.js /test/js	GET	304 Not Modified	application...	index.php:22 Parser	(from cache)	0 ms 0 ms	
addToTakenCourses.js /test/js	GET	304 Not Modified	application...	index.php:23 Parser	(from cache)	0 ms 0 ms	
UpdateGrade.js /test/js	GET	304 Not Modified	application...	index.php:24 Parser	(from cache)	0 ms 0 ms	
searchbar.js /test/js	GET	304 Not Modified	application...	index.php:31 Parser	(from cache)	0 ms 0 ms	
jquery.min.map ajax.googleapis.com/ajax/libs/jquery/1.1(	GET	304 Not Modified	application...	index.php:54 Parser	164 B 137 KB	57 ms 53 ms	
getcourses.php /test/php	POST	200 OK	text/html	jquery.js:8706 Script	431 B 1 B	24 ms 23 ms	

17 requests I 3.8 KB transferred I 529 ms (load: 630 ms, DOMContentLoaded: 533 ms)



https://developers.google.com/speed/pagespeed/insights/?url=http%3A%2F%2F204.83.93.143%3A10080%2F&tab=desktop

PageSpeed Insights

http://204.83.93.143:10080/

83 / 100 Suggestions Summary

Consider Fixing:

Eliminate render-blocking JavaScript and CSS in above-the-fold content
- Show how to fix

Leverage browser caching
- Show how to fix

Optimize images
- Show how to fix

Minify CSS
- Show how to fix

6 Passed Rules
- Show details

8 + 9. Group Member Contributions

(8. All code should be handed in on a flash drive, see User Documentation for instructions to access the website.)

Chris: Database Management, Database Diagrams+documentation, Implementation, debugging, Technical Documentation, Project Proposal, Software Testing.

Nathan: Basically everything. Implementation, debugging, Sequence Diagrams, Technical Documentation, Project Proposal, Component Diagram, User Documentation, Software Architecture.

Mark: Sequence Diagrams, Technical Documentation, Project Proposal, Component Diagram.

Anthony: Class Diagrams, Technical Documentation, Implementation, debugging, Project Proposal, compiling final project document.

Dawn: Implementation, debugging, Use Case Diagrams, Use Case Specifications, Technical Documentation, Project Proposal, Component Diagram, Software Qualities.

Ian: Object diagrams, Use Case Diagrams, UML tools documentation, Technical Documentation, Project Proposal, Deployment Diagram, compiling final project document.