Glen Anderson
glen_anderson@student.uml.edu
Database II COMP.3100
4/28/2016

Database II Term Project: A Web-based Student Information System

Overview

This project is a web based Student Information System that is built on top of a MySQL database. The database contains data such as student information, class information, enrollment information, teacher information, course section information, etc. On top of the database there are two sets of PHP code.

One set of PHP code consists of multiple PHP and HTML files that support the browser-based front end of my project. These files contain static HTML and HTML that is dynamically generated by PHP. The PHP code also manages the database connection/access, the database queries, and the handling of the returned data by integrating it with dynamically generated HTML to display information to the user. The browser based version of the project has multiple functionalities, such as the ability to add/remove a class, add/remove a student, check graduation requirements, view student information, and edit a class grade.

The second set of PHP code consists of just one PHP file that supports the Android app front end of my project. This PHP file manages the database connection and handles HTTP requests sent by the Android app by querying the database using the value sent in the request. Then it organizes the returned data into JSON format and sends it back to the Android app in an HTTP response so the app can parse the data so it can be displayed to the user. The Android app is a .java file that prompts the user to enter a student ID # and press submit. If a valid ID # is entered, the app displays every graduation requirement, along with a “Yes” or “No” depending on whether the requirement was fulfilled or not, and the reason that the requirement was fulfilled or not. The last line that is displayed to the user is a “Yes” or “No” that shows if the user can graduate or not.

Code Listing and Implementation/Application Details (Browser-based version)

File: index.html

This file is a simple landing page that displays a welcome message to the user and has a navigation bar on the top of the page so the user can perform the desired functionality by clicking on the corresponding link. This page does not contain any PHP code or access any PHP files.

File: view.html

This file is an HTML file that contains a form where the user can enter the name and ID # of a student. The user must enter the correct name and ID # of a valid student. The name must be non-numeric and the ID # must be numeric. Once the user enters the values and presses the submit button, the name and ID #values are passed to the file: view_process.html.

File: view_process.php

This file displays class history for a given student. This includes course name, section, teacher, year, semester, grade, credits, and group # for every class the student has taken. It also displays student information such as the student name, ID #, career, degree, major, advisor name, and number of total credits.

The file takes in the values passed to it from the form in view.html via a HTTP POST request. It performs checks on the input to make sure ID # is entered, the name is not numeric, and the ID # is numeric. If any of these checks fail, the user is prompted to go back and enter the correct information. The file then connects to the MySQL data base and displays a failure or success message in the status box accordingly. Another check is performed that uses a query to make sure that the ID # and name entered match a student in the database. If they do not a red message is displayed in the status box, otherwise a green ‘student was found’ message is displayed.

*NOTE: the code that performs the initializations and input validations mentioned above is almost identical in every PHP file that processes a form, so for the rest of the descriptions of the *_process.php files I will omit the above description. It can be assumed that the same process mentioned above occurs, unless mentioned otherwise.*

Then a query is defined that will retrieve enrollment information for the given student. An HTML table and table headers are created in the class history panel with static HTML, and PHP is used to go through each row returned by the query and generate a new row in the table for every class that has been taken.

Next two queries are initialized retrieve various information about the student. The information returned by this query is parsed with PHP and added to the student information panel on the page.

File: new.html

This file displays a static HTML form that allows a user to input information for a new student that will be passed to new_process.php to add the new student to the database. The form has input fields for the student’s name, ID #, career, degree held, adviser, and major. The career, degree held, and major fields initialize to default values, but can still be changed if needed.

File: new_process.php

This file takes in the values through the HTTP POST that was passed by new.html. It performs checks to make sure the student ID # and adviser ID # are numeric and to make sure the name, ID, career, degree, major, and adviser values are not empty. If any of these checks fail, the user is prompted with a link to go back and correct the mistakes. Then, the PHP file displays the entered values under the status panel. Then it attempts to connect to the database, and displays the status of the database connection under the status panel. Then an 'insert' query is executed to add the new student into the students table, and the status of the 'insert' query is displayed under the status panel and student information panel

File: update.php

This file has two sections. One section contains a form that is used to add a class to a students' schedule, and the other section contains a form to remove a class from a students' schedule. These forms both use PHP to dynamically generate dropdown lists to allow the user to choose from values that are in the database as opposed to being required to write in the value. Each of the dropdown lists is generated by querying the database for the desired values and looping through each of the rows and printing each value as an option in the list. For all of the dropdowns each option is the same as the value, except in the course dropdowns. Because there are two versions of Operating Systems I (the graduate and undergraduate versions), I needed to have the course dropdowns display the name of the course, but use the course ID # as the value for each course. If the course name acted as the dropdown name and value, the update_process_*.php file that processes this form would not be able to differentiate between the graduate and undergraduate versions of Operating Systems I. The 'Add a Class' form sends its values to update_process_add.php and the 'Remove a Class' form sends its values to update_process_delete.php.

File: update_process_add.php

This file takes in the values from the 'Add a Class' form update.php when a user presses the submit button at the bottom of the form. This file connects to the database, and then makes sure the section and year fields were not left blank or filled in with non-numeric characters. It also checks to make sure that the ID # and student name match a student in the students table. Then it executes a query to make sure the class can be found in the database. If it can be found, a query is executed on the prerequisite table to check if there any prerequisites for the class. If there are any prerequisites, the enrollment table is queried to make sure the required class was taken. Then if the above checks pass, the class is added into the enrollment table for the student.

File: update_process_delete.php

This file performs the same initial checks as update_process_add.php described above. When the checks are completed, it executes a 'delete from where exists' query to delete the class specified in the 'Remove a Class' form. After it attempts to remove a class, it verifies to the user that the class was removed. If the class was not removed an error message is displayed to the user.

File: delete.php

This file is used for the form that is used to delete a student from the students table in the database. This form uses dropdown menus that are dynamically generated from the database tables using 'select'/'select distinct' queries. PHP is used to populate the dropdowns using the values returned from the queries. The form passes the input values to delete_process.php to be processed so the student can be removed.

File: delete_process.php

This file takes as input the values passed by delete.php that specify a student to be deleted from the system. First, it displays the values entered by the user and the database connection status under the status panel. Then it sees if the input ID # and student name match a student in the database. If they do match a student, a message is displayed under the status panel saying that the student was found. Then the 'delete' query is executed to delete the student from the students table. A message is displayed under the student removal panel that states whether the student removal query executed successfully or not. Next, a query is executed that removes all of the classes from the enrollment table that the student was enrolled in. a Message is also displayed that reports whether or not this query was executed successfully as well.

File: grade.php

This page displays the form that is used to edit a grade for a class that the given user is already enrolled in. The new grade and semester dropdowns are implemented statically, and the ID#, name, and course dropdowns are generated dynamically using values in the database. The year field takes a value input by the user. In the course dropdown, the course names are displayed in the list but the corresponding course ID# is passed to the processing form (grade_process.php) because of the ambiguity of the graduate and undergraduate versions of Operating Systems I. In the grade dropdown, a blank value is purposely given as an option in case of a situation where a letter grade is not appropriate.

File: grade_process.php

This file takes in the 6 values passed by the form in grade.php, and reports them to the user under the status panel for verification. Then it connects to the database and reports the database connection status in the return status section of the status panel. It also makes sure the specified student is found in the database and reports the status of the student lookup in the return status section of the status panel. It also checks to make sure the input year is not empty and is numeric, since the year field is specified by the user through a text box instead of a dropdown menu like the other fields. Then a query is executed to check if the new grade is the same as the grade that already exists in the enrollment table for the specified class. If the grades are the same, a message is reported to the user under the grade change panel, and the script exits because nothing else needs to be done. If the grades are different, an 'update' query is executed to change the grade, and the status of the grade change is reported to the user under the grade change panel.

File: graduate.php

This file contains a form that is used to check the graduation status of a given student. The form is composed of two dropdown menus that are populated with values returned by querying the students table in the database. The two dropdown menus allow the user to specify the name and ID # of the student that they want to check the graduation status of. These two values are passed to graduate_process.php in a HTTP POST.

File: graduate_process.php

This file takes in the name and ID # passed by the form on graduate.php and checks that the specified student satisfied all of the graduation requirements. First, it defines a function called db_connect() that returns a static database connection object, and can be called by functions throughout the file to gain access to the database. Then the values entered are displayed under the status panel. Then it attempts to connect to the database, and reports the status of the database connection to the user under the status panel on the page. Then it performs a check to make sure the input ID # and name match a student in the database. It reports the validity of the entered student in the status panel on the page. Then global constants are defined that are used to map latter grades to their corresponding grade point values. A global variable, canGraduate, is defined that is initialized to 1, but is set to 0 if any graduation requirement is unsatisfied. A global variable, reason, is initialized to an empty string. If any graduation requirement is unsatisfied, the reason for the incompletion of that requirement is appended to the reason string. At the end of the file's execution the canGraduate and reason global are used to report whether or not the student can graduate and explanations of any unsatisfied requirements.

Next is the definition of the conditions() function that checks to see if the student fulfilled their condition requirements. First it queries the condition table for the specified student, and for every row (class) returned by the query, it queries the enrollment table to make sure the student was enrolled in that class. If there is a condition class that the student has not been enrolled in, the canGraduate global is set to 0, the class name/reason is appended to the reason global, and a message is displayed udner the 'conditions' section of the 'graduation status' panel. If all of the conditions were enrolled, then a message is displayed under the 'conditions' section of the 'graduation status' panel. This function does not have a return value because its only purpose is to alter the canGraduate and reason globals if there is an unsatisfied condition.

Next is the definition of the numCreditsEarned() function that is used to calculate the total number of credits that a student has earned. A 'select sum' query is executed to calculate the total. The 'where' clause is designed to only include credits of classes that are graduate classes (group ID > 0), and classes that were passed (grade != F). Classes with a blank grade were purposely interpreted as passed so the student can qualify for graduation during his/her last semester. This function returns the number of credits taken by the specified student.

Next is the definition of the gpa() function. This function returns the cumulative GPA for the given student. This function starts by executing a query to get all the letter grades that are not in group 0 and that have a grade higher than F because only graduate courses with passing grades should count. Then it enters a loop to calculate the cumulative GPA. In each iteration of the loop, the counter is incremented, the next letter grade is extracted from the next result row, the letter grade is converted to its numeric equivalent using the global constants defined earlier, and the grade point value is added on to a running total. After a check to prevent division by zero, the running total is divided by the counter to obtain the cumulative GPA. The function then returns this value.

Next is the definition of the numBelowB() function, which calculates the number of grades below B. The function executes a ‘select count’ query to calculate this number. The Where field is designed to count B- and grades that start with F, C, and D. This is because all ‘+’ and ‘-‘ for grades under B- should be counted in the calculation. The number returned by this query is the number of grades below B, so the function returns this value.

Next is the definition of the CoreAndElectives() function. This function checks to make sure the student has satisfied the core class requirement (Algorithms, and a class from group 2, 3, and 4.) and the elective requirement (6 courses that are different than the ones used to satisfy the core class requirement). First, a query is executed to check if the student has taken algorithms. If algorithms was not taken, the canGraduate global is set to 0, and a message is appended to the reason global. If algorithms was taken, a message is displayed under the graduation status panel. Then 3 other queries are set up that check if the student has taken the rest of the core classes (a class from group 2, 3, and 4). If a group is unsatisfied, the canGraduate global is set to 0, and a message is appended to the reason global. If the student has taken a class from groups 2, 3, and 4, a success message is displayed under ‘core classes’ section of the ‘graduation status’ panel. Then a ‘select sum’ query is set up to check if the student has taken 18 or more elective credits. The ‘where’ clause of this query is designed so none of the classes counted as a core class are counted as an elective class, and that algorithms (915030) and group 0 courses are not counted as core classes either. If the number of elective credits is greater than 18, a message is displayed under the elective section of the ‘graduation status’ panel. If the number of elective credits is less than 18, the canGraduate global is set to 0, a message is appended to the reason global, and a message is displayed under the elective section of the ‘graduation status’ panel. This function does not return a value.

After all of these function definitions, the functions are called. The conditions( ) function is called to set the canGraduate and reason globals accordingly. Then the gpa( ) function is called and a check is performed to see if it is 3 or higher. If it is higher than 3, a message is displayed under the gpa section of the ‘graduation status’ panel. If it is not, the canGraduate global is set to 0 and a message is appended to the reason global. Then the numCreditsEarned( ) function is called and a check is performed to see if the student has 30 or more credits. If there are more than 30, a message is displayed under the credits section of the ‘graduation status’ panel. If it is not, the canGraduate global is set to 0 and a message is appended to the reason global. Then the numBelowB ( ) function is called and a check is performed to see if the student has more than 2 grades below B. If there are more than 2, the canGraduate global is set to 0 and a message is appended to the reason global. If there are 2 or less, a message is displayed under the ‘Grades below B’ section of the ‘graduation status’ panel. Then the CoreAndElectives function is called to check the student’s core and elective requirement completion and set the reason and canGraduate globals accordingly. Finally, the canGraduate global is checked. If the canGraduate global is equal to 1, a message saying the user can graduate is displayed under the ‘graduation status’ panel. If the canGraduate global is equal to 0, a message is displayed under the ‘graduation status’ panel saying that the user cannot graduate, and the reason global string is appended to this message. The reason string will display every reason why the student cannot graduate.

Code Listing and Implementation/Application Details (Android app version)

File: MainActivity.java

This is the main file for the Java app portion of my project. After all of the necessary import statements is the definition for the MainActivity class, which acts as a driver class for the app. In the MainActivity class, the myURL string is set to "http://10.0.2.2/project/connector.php" because 10.0.2.2 functions as localhost in the context of an emulator and project/connector.php is the path of my server-side php file that receives the request from the app and returns a JSON file to it. A function called setSoftInputMode( ) is called so that the keyboard doesn’t appear until the editText component is selected. Then the window components are stored into local objects so that they can be used, and setMovementMethod( ) is called to make the result view scrollable (where the parsed JSON result will be displayed to the user). Next is the definition of the OnClickListener which binds a callback function to the OnClick event of my submit button. The callback function, OnClick puts the student ID # from the text input field into a local variable when the submit button is pressed. Then it defines and starts a new MyTask, which is an AsyncTask, and passes the input Student ID # to this new async task. The doInBackground function in the AsyncTask class first appends the input Student ID # to the myURL variable and uses this new link to create a URL object. Then the URL object’s openConenction method is called and returned into a url connection object. Then the request is prepared by setting the timeout values, setting the content-type to JSON, and setting the request method to GET. Then the open method if the url connection object is called, which starts the connection. Then a buffered reader is opened and set to read from the url connection’s input stream. Once the string from the input stream is assigned to a local variable, the input stream is closed and the url connection is closed. The doInBackground function returns this string. The onPostExecute function receives the string as input and passes it into a JSONObject type, which is defined in the org.json package. Then the value in the “student” key is checked to make sure that the ID # entered was a valid ID # and that the student was found in the database by the server-side php file. If the student was invalid, “invalid student” will be displayed on the screen and the JSON parsing will halt there. If the student is valid, then the reqsArray JSONArray is parsed to display to the user 3 fields for each graduation requirement. These 3 fields are: Name (the name of the requirement), Pass (whether or not the requirement was satisfied), and Reason (the reason for passing or failing the requirement). Then the value for the Can Graduate key is displayed to the user to tell the user whether or not the student can graduate.

File: connector.php

This file is the server side PHP file that handles the requests made by the Android app front end of my project. This file receives a student ID # through at HTTP GET request, finds the specified student in the database, and creates a JSON string that contains information about the graduation requirement status for the given student. Then it sends this JSON string back to the Android app client through the json_encode function. This file is similar to graduate_process.php of the browser-based version of my project.

This file starts by defining global constants that define letter grade constants as their numeric GPA values. These constants will be used during the GPA calculation. A global called canGraduate is also declared and initialized to 1. This global is used to keep track of the student's graduation status by setting it to 0 as soon as the script sees that a graduation requirement is not satisfied. Response is defined as an array that will be used to hold the outer JSON object, and response["reqs"] will be an array that will hold the array of graduation requirements (the array that is later read into the reqsArray variable in the Android app). Then the db_connect ( ) function is defined to manage the MySQL database connection by returning a static connection variable. After the variable is read from the GET request and the database connection is set up, a check is performed to make sure that the input student ID # is numeric and that it is not empty. If it is empty or non-numeric the value of the student key is set to invalid and json_encode is called before exiting the PHP script so that the android app will report the student as invalid before trying to parse the JSON string for student requirements. Another check is performed by querying the database for the student's name. If a name cannot be found, the student is reported as invalid in the same manner described previously. Each requirement is checked in a manner similar to the graduate_process.php, except there is no global reason variable that holds all of the reasons a student cannot graduate. Instead, the reason value is only displayed/stored in the Reason key for a requirement.

The first requirement check in the file is for the student's conditions. The conditions check is performed the same way as in the browser version of my project (using the same MySQL query). If the student satisfies all of their conditions then the value of the Name key is set to 'Conditions' , the value of the Pass key is set to 'Yes', and the value of the Reason key is set to 'Has taken condition class <condition>'. If the student does not satisfy all of their conditions then the canGradute global is set to 0, the value of the Name key is set to 'Conditions', the value of the Pass key is set to 'No', and the value of the Reason key is set to 'Has not taken condition class <condition>'. This JSON object is then pushed into the response["reqs"] array using the array_push function.

Then a check for the number of credits requirement is performed. This check uses the same query as in graduate_process.php, and if the student has 30 or more credits a new JSON object is created with the value of the Name key set to ‘Credits’, the value of the Pass key set to ‘Yes’, and the value of the Reason key set to ‘You have 30 or more credits (<credits>).’. If the student has less than 30 credits, the canGraduate global is set to 0, and a new JSON object is created with the value of the Name key set to ‘Credits’, the value of the Pass key set to ‘No’, and the value of the Reason key set to ‘You have less than 30 credits (<credits>).’. This JSON object is then pushed into the response[“reqs”] array using the array_push function

Next is the check for the GPA graduation requirement. This check uses the same query as in graduate_process.php. If the student has a GPA greater than or equal to 3.0, a new JSON object is created with the value of the Name key set to ‘GPA’, the value of the Pass key set to ‘Yes’, and the value of the Reason key set to ‘GPA greater than or equal to B (<GPA>).’. If the student has less than 30 credits, the canGraduate global is set to 0, and a new JSON object is created with the value of the Name key set to ‘GPA’, the value of the Pass key set to ‘No’, and the value of the Reason key set to ‘GPA less than B (<GPA>).’ This JSON object is then pushed into the response[“reqs”] array using the array_push function.

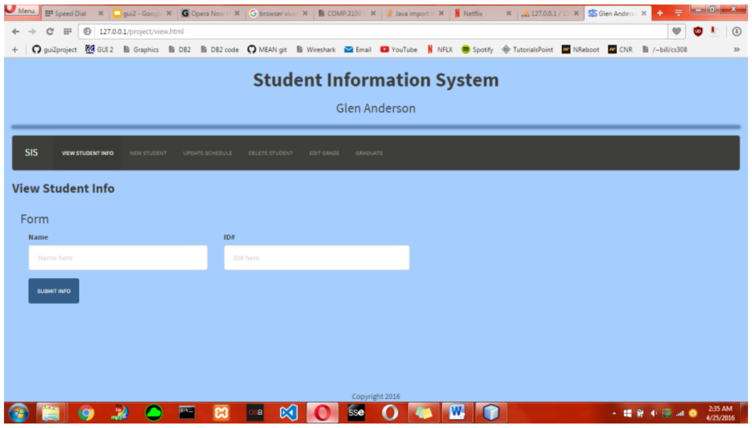
The next check is for the number of grades below B. This check uses the same query as in graduate_process.php. If the student has 2 or less grades below a B, a new JSON object is created with the value of the Name key set to ‘# of grades below B’, the value of the Pass key set to ‘Yes’, and the value of the Reason key set to ‘No more than 2 grades below B <# of grades below B>’. If the student has less more than 2 grades below B, the canGraduate global is set to 0, and a new JSON object is created with the value of the Name key set to ‘# of grades below B’, the value of the Pass key set to ‘No’, and the value of the Reason key set to ‘More than 2 grades below B (<# of grades below B>).’ This JSON object is then pushed into the response[“reqs”] array using the array_push function.

The last check is for the core and elective requirements. This check is completed in the same manner as the checks above using the same algorithm and queries as the core and elective requirement check in graduate_process.php (of the browser-based version of the project). For brevity and to avoid unnecessary repetition, the explanation of this check is omitted.

Then the canGraduate global is checked. If the canGraduate global is equal to 1, the value of the response[“Can Graduate”] key is set to “Yes”, else it is set to “No”. Then the JSON object response is sent back to the Android app using the json_encode( ) function.

Sample Outputs for Browser-based Version of Project

File: view.html



127.0.0.1/project/view.html

Student Information System

Glen Anderson

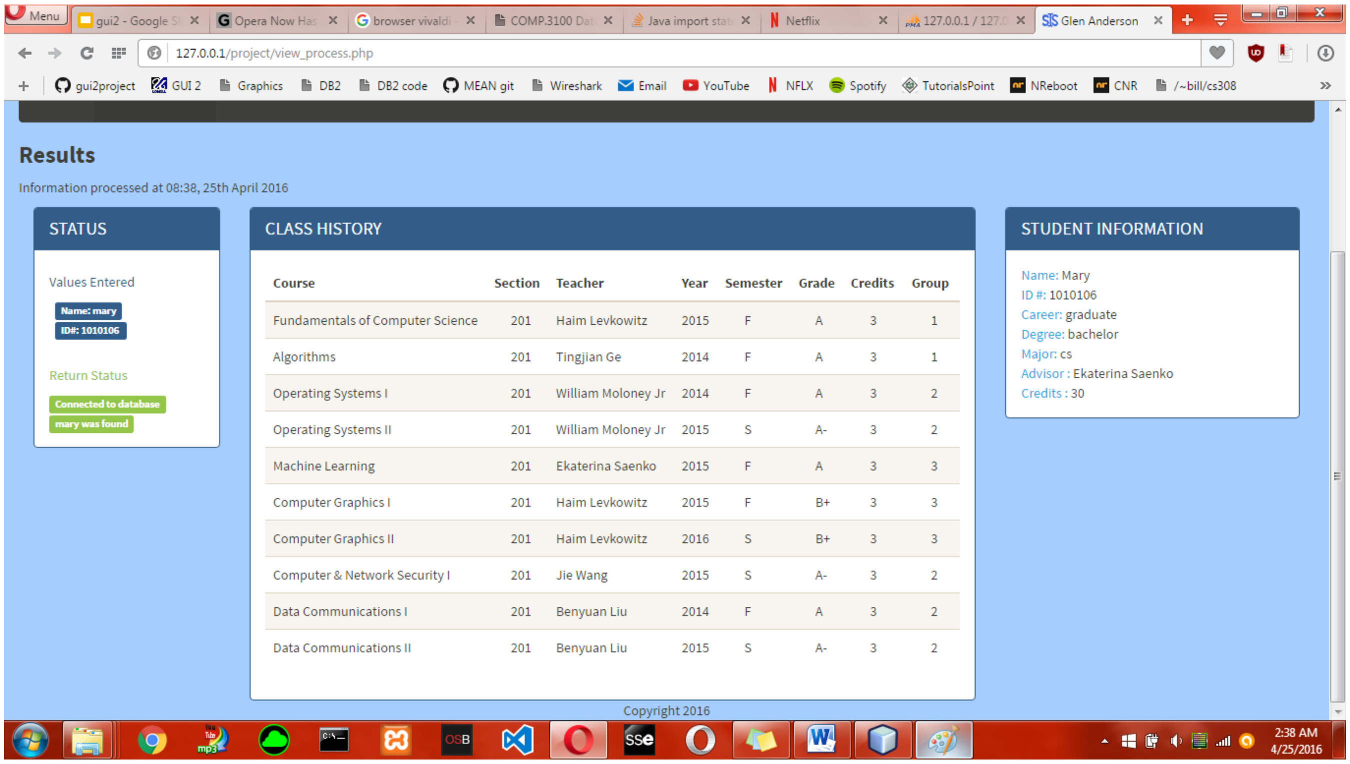
View Student Info

Form

Name	ID#
Name here	ID# here

Copyright 2016

File: view_process.php



127.0.0.1/project/view_process.php

Results

Information processed at 08:38, 25th April 2016

STATUS

Values Entered

Name: mary
ID#: 1010106

Return Status

Connected to database
mary was found

CLASS HISTORY

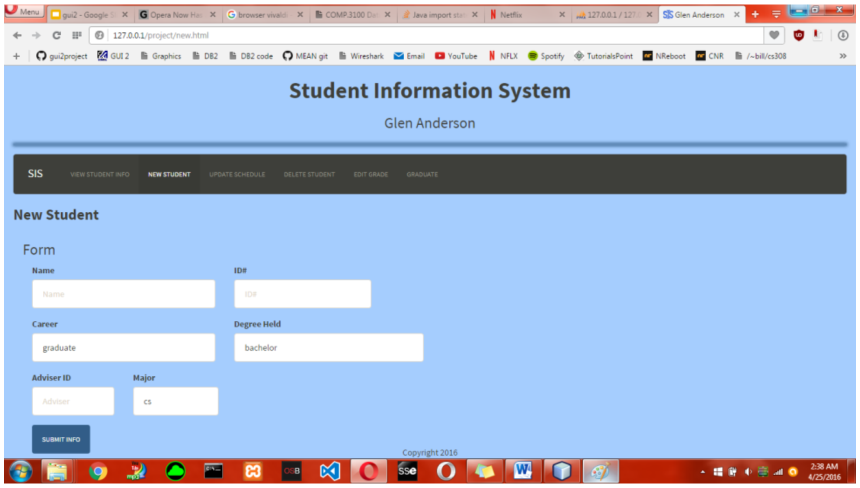
Course	Section	Teacher	Year	Semester	Grade	Credits	Group
Fundamentals of Computer Science	201	Haim Levkowitz	2015	F	A	3	1
Algorithms	201	Tingjian Ge	2014	F	A	3	1
Operating Systems I	201	William Moloney Jr	2014	F	A	3	2
Operating Systems II	201	William Moloney Jr	2015	S	A-	3	2
Machine Learning	201	Ekaterina Saenko	2015	F	A	3	3
Computer Graphics I	201	Haim Levkowitz	2015	F	B+	3	3
Computer Graphics II	201	Haim Levkowitz	2016	S	B+	3	3
Computer & Network Security I	201	Jie Wang	2015	S	A-	3	2
Data Communications I	201	Benyuan Liu	2014	F	A	3	2
Data Communications II	201	Benyuan Liu	2015	S	A-	3	2

STUDENT INFORMATION

Name: Mary
ID #: 1010106
Career: graduate
Degree: bachelor
Major: cs
Advisor : Ekaterina Saenko
Credits : 30

Copyright 2016

File: new.html



127.0.0.1/project/new.html

Student Information System

Glen Anderson

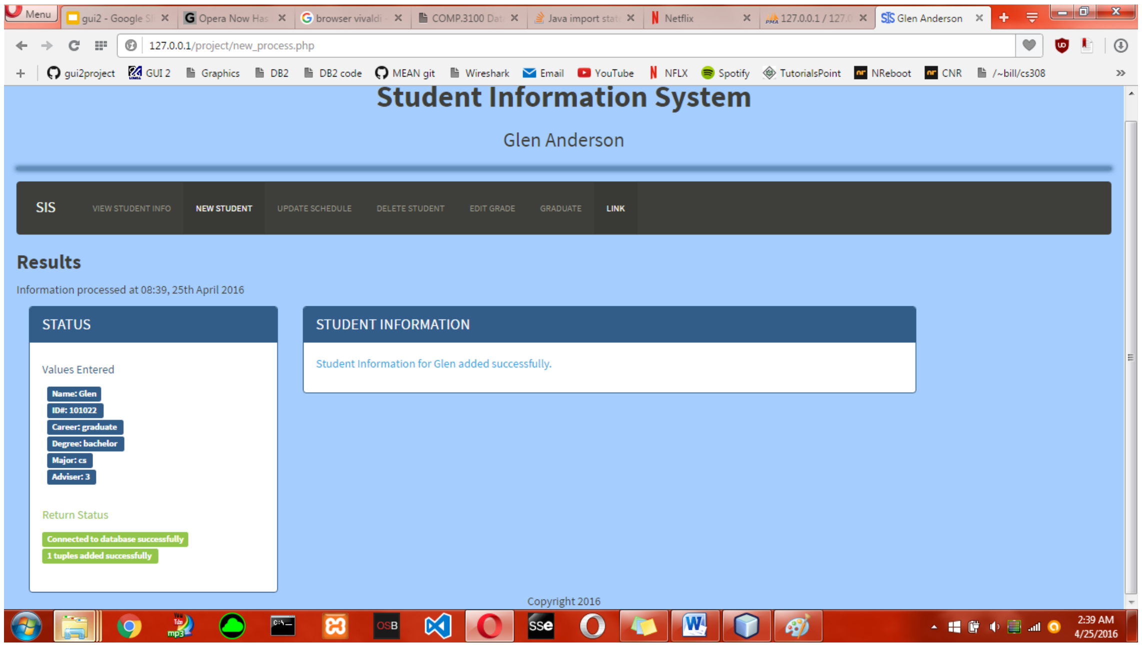
New Student

Form

Name: Name	ID#: ID#
Career: graduate	Degree Held: bachelor
Adviser ID: Adviser	Major: cs

Copyright 2016

File: new_process.php



127.0.0.1/project/new_process.php

Student Information System

Glen Anderson

Results

Information processed at 08:39, 25th April 2016

STATUS

Values Entered

Name: Glen
ID#: 101022
Career: graduate
Degree: bachelor
Major: cs
Adviser: 3

Return Status

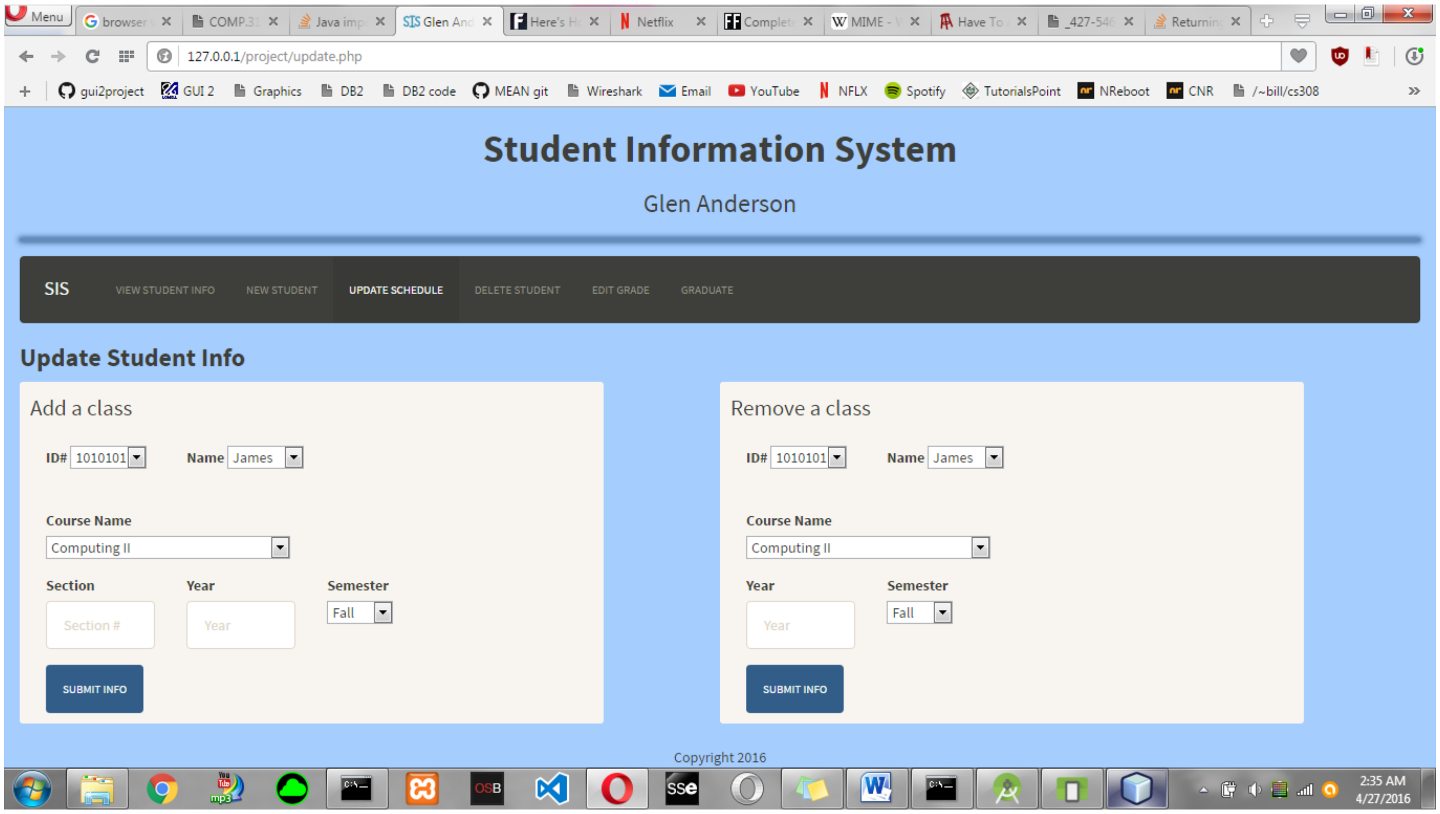
Connected to database successfully
1 tuples added successfully

STUDENT INFORMATION

Student Information for Glen added successfully.

Copyright 2016

File: update.php



Student Information System

Glen Anderson

SIS VIEW STUDENT INFO NEW STUDENT UPDATE SCHEDULE DELETE STUDENT EDIT GRADE GRADUATE

Update Student Info

Add a class

ID# 1010101 Name James

Course Name: Computing II

Section: Section # Year: Year Semester: Fall

SUBMIT INFO

Remove a class

ID# 1010101 Name James

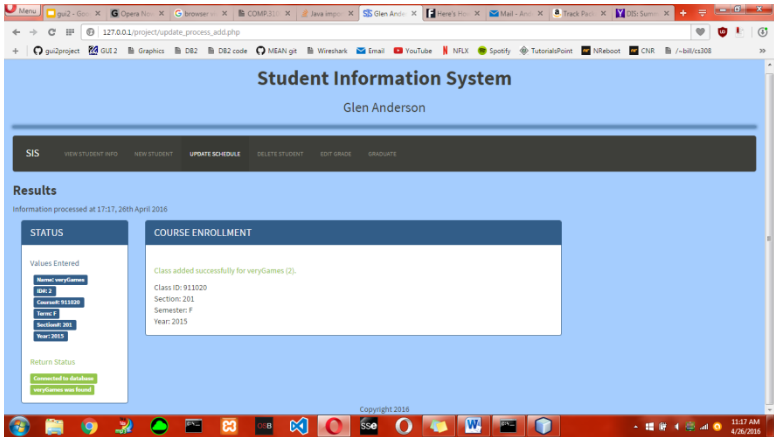
Course Name: Computing II

Year: Year Semester: Fall

SUBMIT INFO

Copyright 2016

File:update_process_add.php



Student Information System

Glen Anderson

SIS VIEW STUDENT INFO NEW STUDENT UPDATE SCHEDULE DELETE STUDENT EDIT GRADE GRADUATE

Results

Information processed at 17:17, 26th April 2016

STATUS

Values Entered

Name: veryGames
ID#: 2
Course#: 911020
Term: F
Section#: 201
Year: 2015

Return Status

Connected to database
veryGames was found

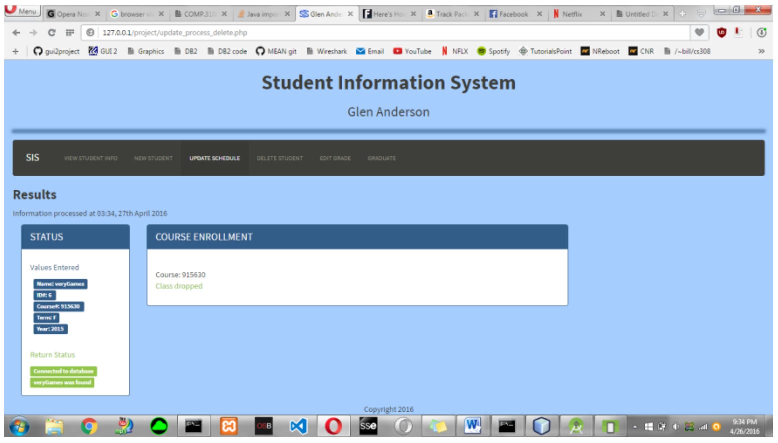
COURSE ENROLLMENT

Class added successfully for veryGames (2).

Class ID: 911020
Section: 201
Semester: F
Year: 2015

Copyright 2016

File:update_process_delete.php



Student Information System

Glen Anderson

SIS VIEW STUDENT INFO NEW STUDENT UPDATE SCHEDULE DELETE STUDENT EDIT GRADE GRADUATE

Results

Information processed at 03:34, 27th April 2016

STATUS

Values Entered

Name: veryGames
ID#: 6
Course#: 915630
Term: F
Year: 2015

Return Status

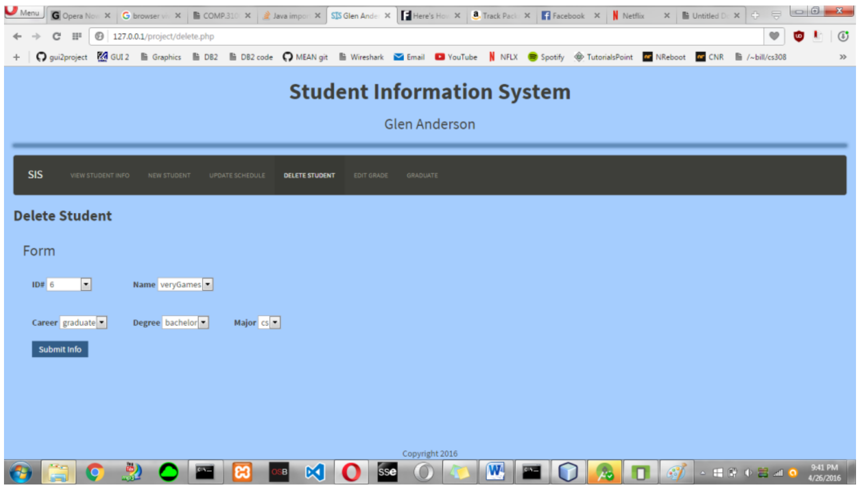
Connected to database
veryGames was found

COURSE ENROLLMENT

Course: 915630
Class dropped

Copyright 2016

File: delete.php



127.0.0.1/project/delete.php

Student Information System

Glen Anderson

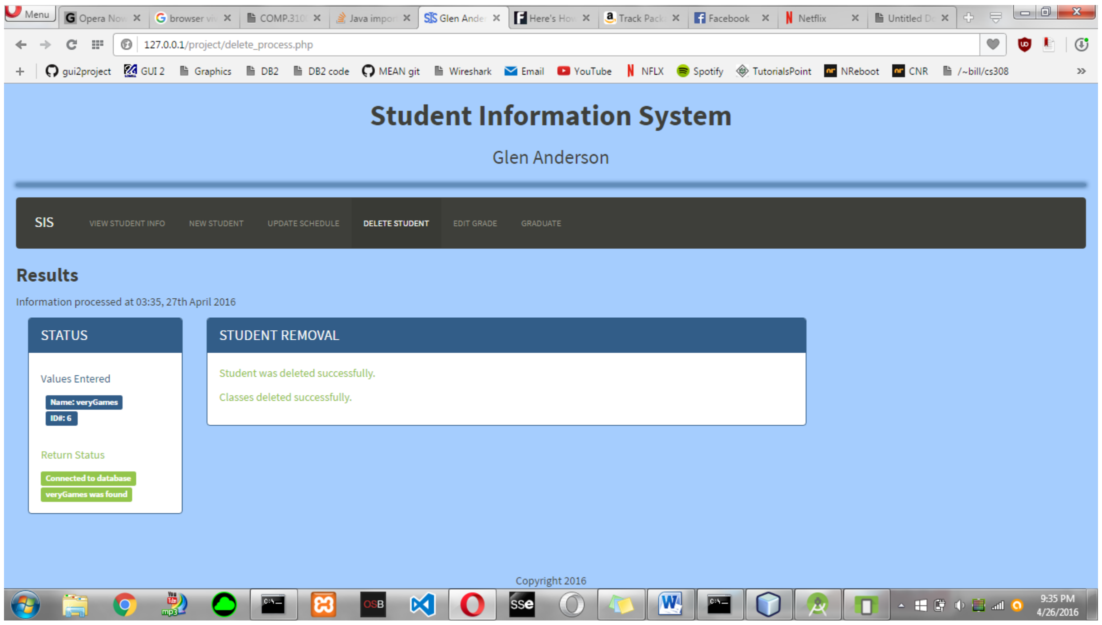
Delete Student

Form

ID# 6	Name veryGames	
Career graduate	Degree bachelor	Major cs

Copyright 2016

File: delete_process.php



127.0.0.1/project/delete_process.php

Student Information System

Glen Anderson

Results

Information processed at 03:35, 27th April 2016

STATUS

Values Entered

Name: veryGames

ID#: 6

Return Status

Connected to database

veryGames was found

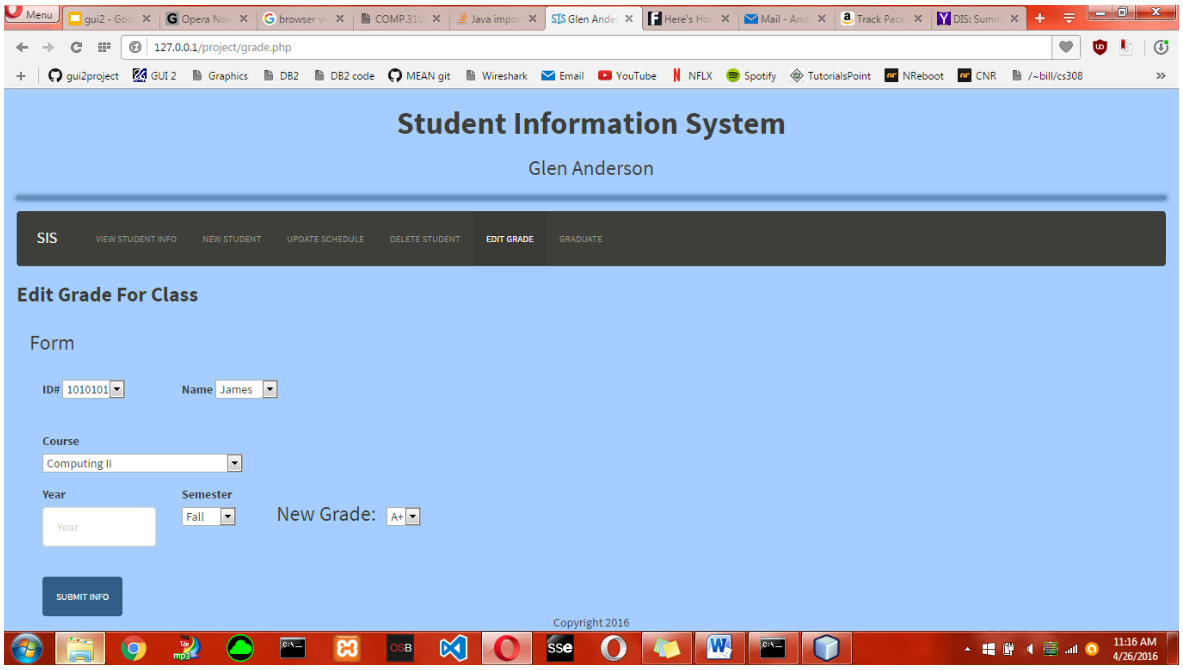
STUDENT REMOVAL

Student was deleted successfully.

Classes deleted successfully.

Copyright 2016

File: grade.php



127.0.0.1/project/grade.php

Student Information System

Glen Anderson

SIS | VIEW STUDENT INFO | NEW STUDENT | UPDATE SCHEDULE | DELETE STUDENT | EDIT GRADE | GRADUATE

Edit Grade For Class

Form

ID# 1010101 Name James

Course
Computing II

Year
Year

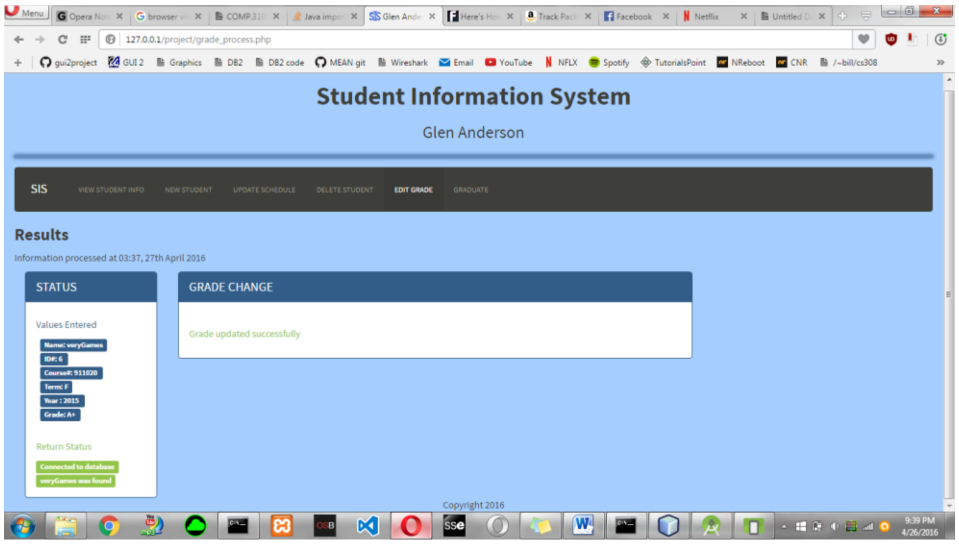
Semester
Fall

New Grade: A+

SUBMIT INFO

Copyright 2016

File: grade_process.php



127.0.0.1/project/grade_process.php

Student Information System

Glen Anderson

SIS | VIEW STUDENT INFO | NEW STUDENT | UPDATE SCHEDULE | DELETE STUDENT | EDIT GRADE | GRADUATE

Results

Information processed at 03:37, 27th April 2016

STATUS

Values Entered

Name: veryGames
ID#: 6
Course#: 911020
Term: F
Year : 2015
Grade: A+

Return Status

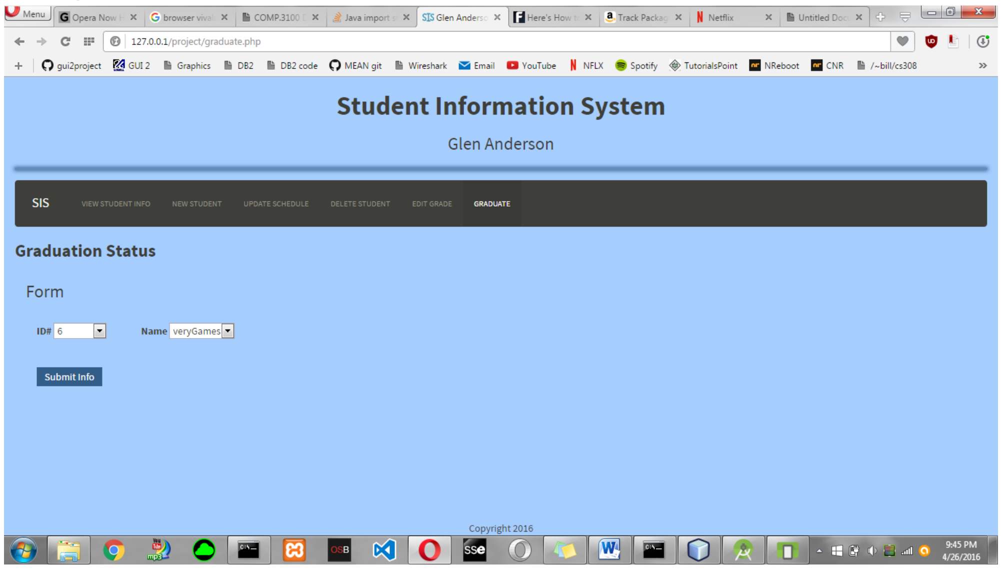
Connected to database
veryGames was found

GRADE CHANGE

Grade updated successfully

Copyright 2016

File: graduate.php



Student Information System

Glen Anderson

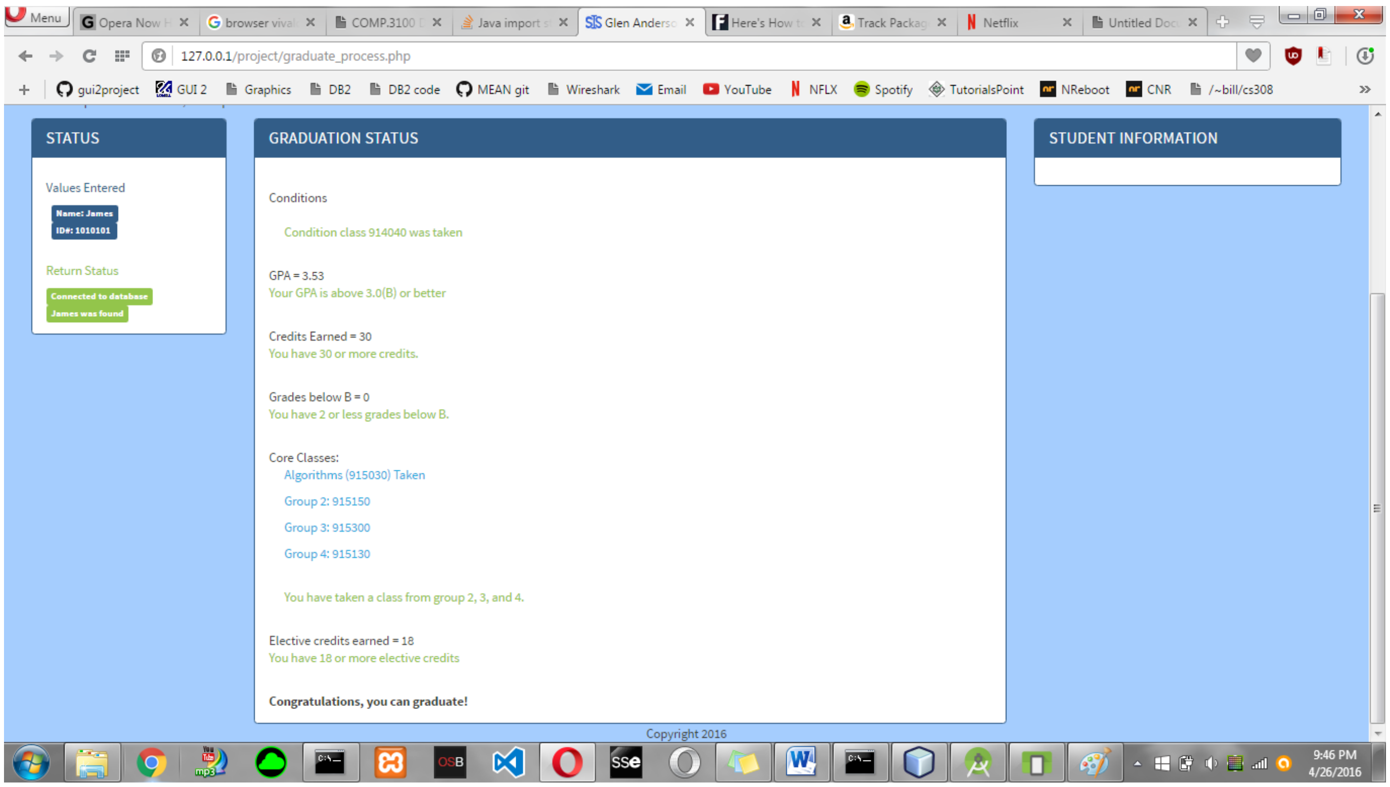
Graduation Status

Form

ID# 6   Name veryGames

Copyright 2016

File: graduate_process.php



STATUS

Values Entered

Name: James
ID#: 1010101

Return Status

Connected to database
James was found

GRADUATION STATUS

Conditions

Condition class 914040 was taken

GPA = 3.53
Your GPA is above 3.0(B) or better

Credits Earned = 30
You have 30 or more credits.

Grades below B = 0
You have 2 or less grades below B.

Core Classes:
Algorithms (915030) Taken
Group 2: 915150
Group 3: 915300
Group 4: 915130

You have taken a class from group 2, 3, and 4.

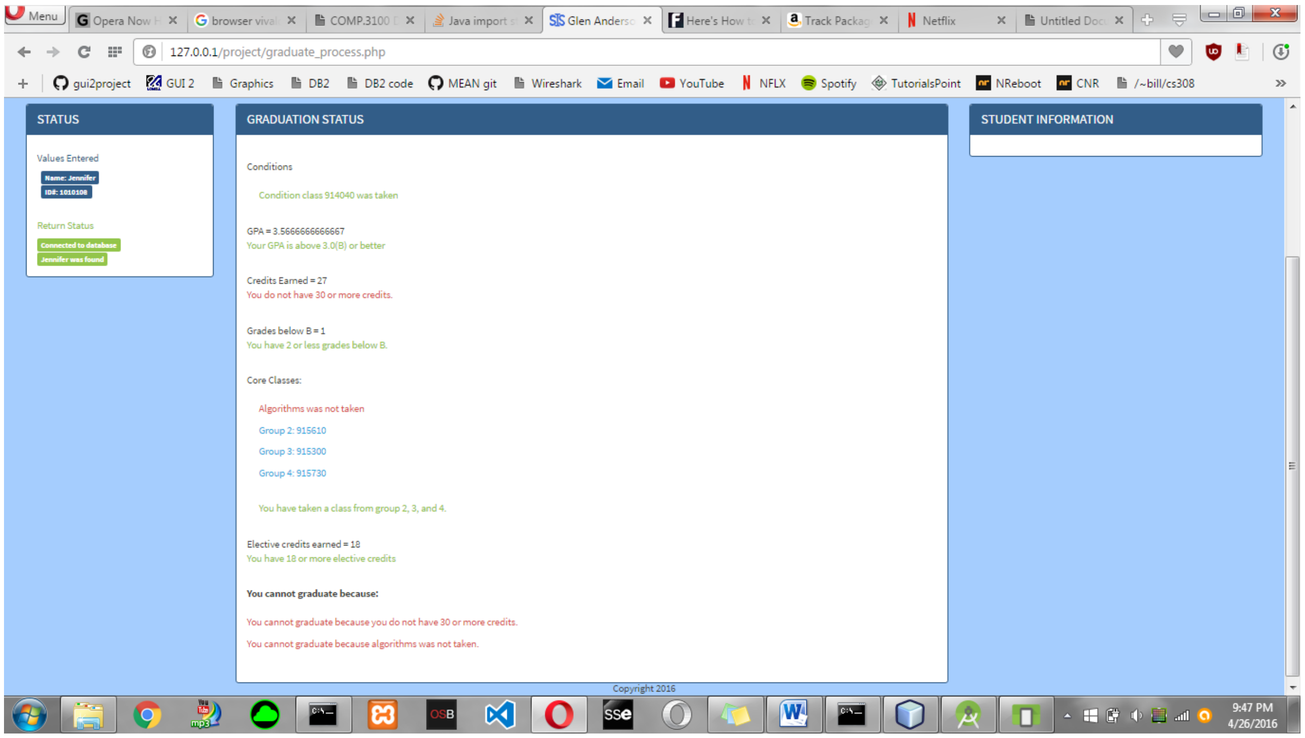
Elective credits earned = 18
You have 18 or more elective credits

Congratulations, you can graduate!

STUDENT INFORMATION

Copyright 2016

Example 1


127.0.0.1/project/graduate_process.php

STATUS

Values Entered

Name: Jennifer
ID#: 1010108

Return Status

Connected to database
Jennifer was found

GRADUATION STATUS

Conditions

Condition class 914040 was taken

GPA = 3.5666666666667
Your GPA is above 3.0(B) or better

Credits Earned = 27
You do not have 30 or more credits.

Grades below B = 1
You have 2 or less grades below B.

Core Classes:

Algorithms was not taken

Group 2: 915610

Group 3: 915300

Group 4: 915730

You have taken a class from group 2, 3, and 4.

Elective credits earned = 18
You have 18 or more elective credits

You cannot graduate because:

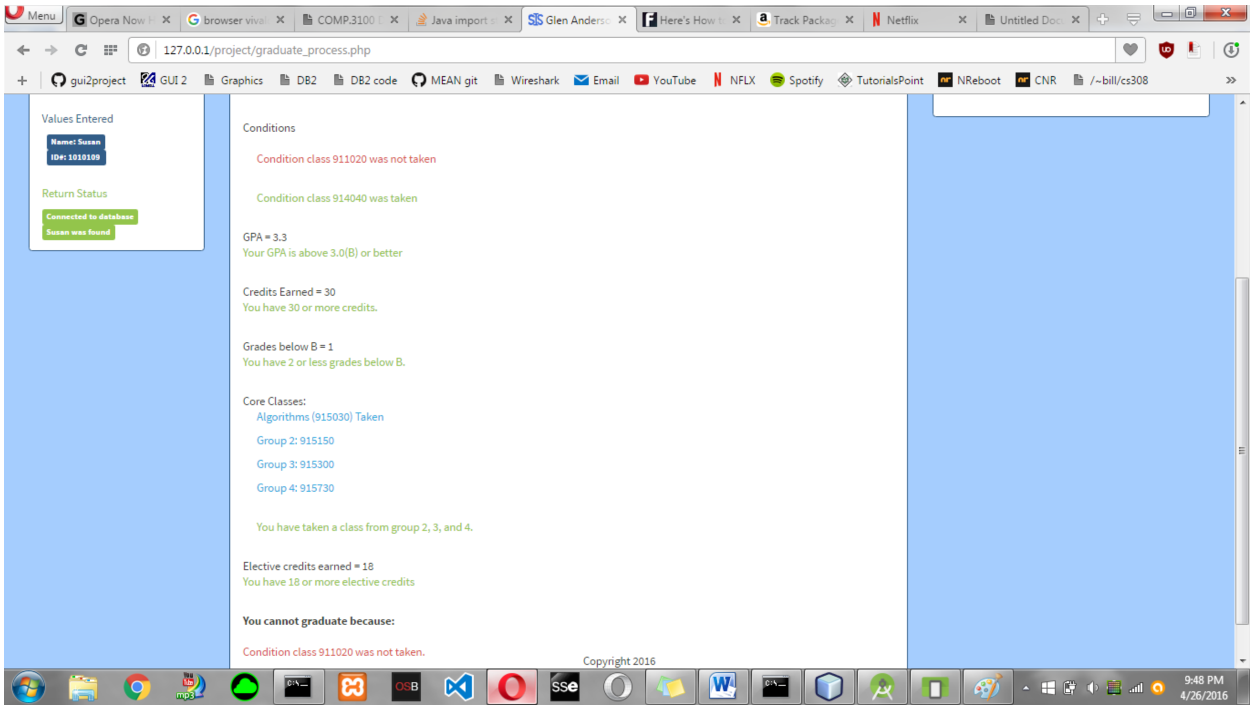
You cannot graduate because you do not have 30 or more credits.

You cannot graduate because algorithms was not taken.

STUDENT INFORMATION

Copyright 2016

Example 2


127.0.0.1/project/graduate_process.php

Values Entered

Name: Susan
ID#: 1010109

Return Status

Connected to database
Susan was found

Conditions

Condition class 911020 was not taken

Condition class 914040 was taken

GPA = 3.3
Your GPA is above 3.0(B) or better

Credits Earned = 30
You have 30 or more credits.

Grades below B = 1
You have 2 or less grades below B.

Core Classes:
Algorithms (915030) Taken

Group 2: 915150

Group 3: 915300

Group 4: 915730

You have taken a class from group 2, 3, and 4.

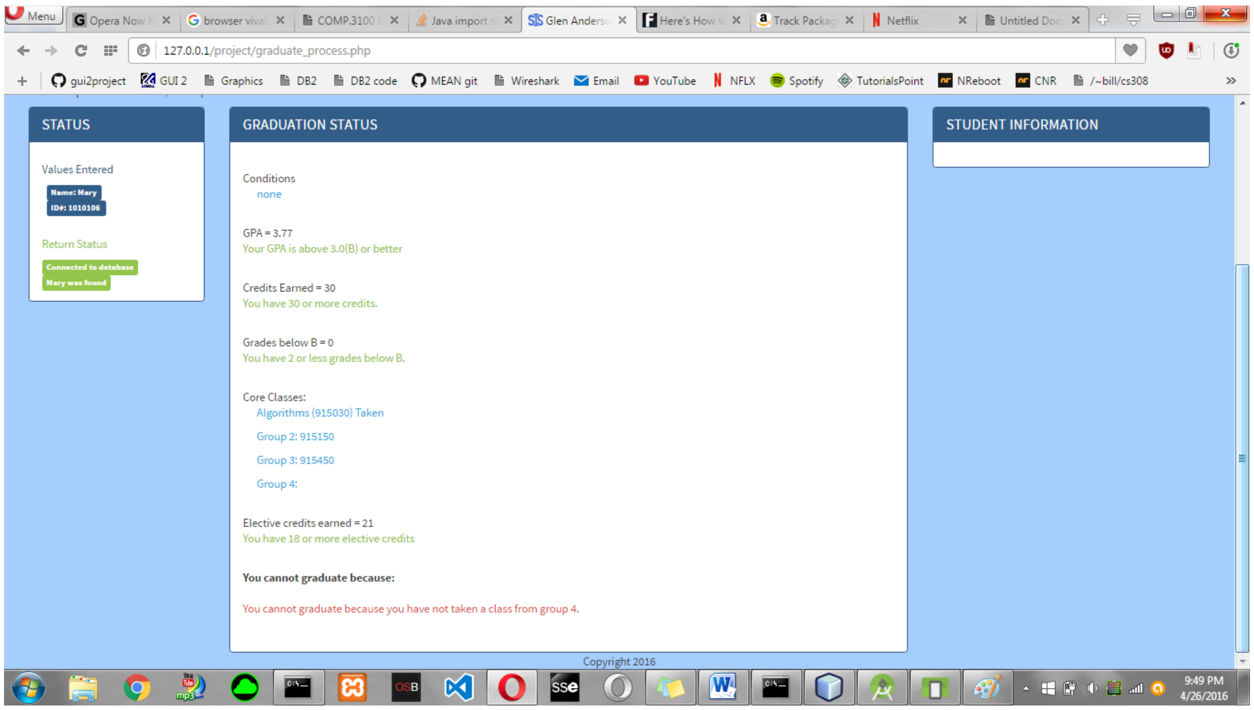
Elective credits earned = 18
You have 18 or more elective credits

You cannot graduate because:

Condition class 911020 was not taken.

Copyright 2016

Example 3


127.0.0.1/project/graduate_process.php

STATUS

Values Entered

Name: Mary

ID#: 1010106

Return Status

Connected to database

Mary was found

GRADUATION STATUS

Conditions

none

GPA = 3.77

Your GPA is above 3.0(B) or better

Credits Earned = 30

You have 30 or more credits.

Grades below B = 0

You have 2 or less grades below B.

Core Classes:

Algorithms (915030) Taken

Group 2: 915150

Group 3: 915450

Group 4:

Elective credits earned = 21

You have 18 or more elective credits

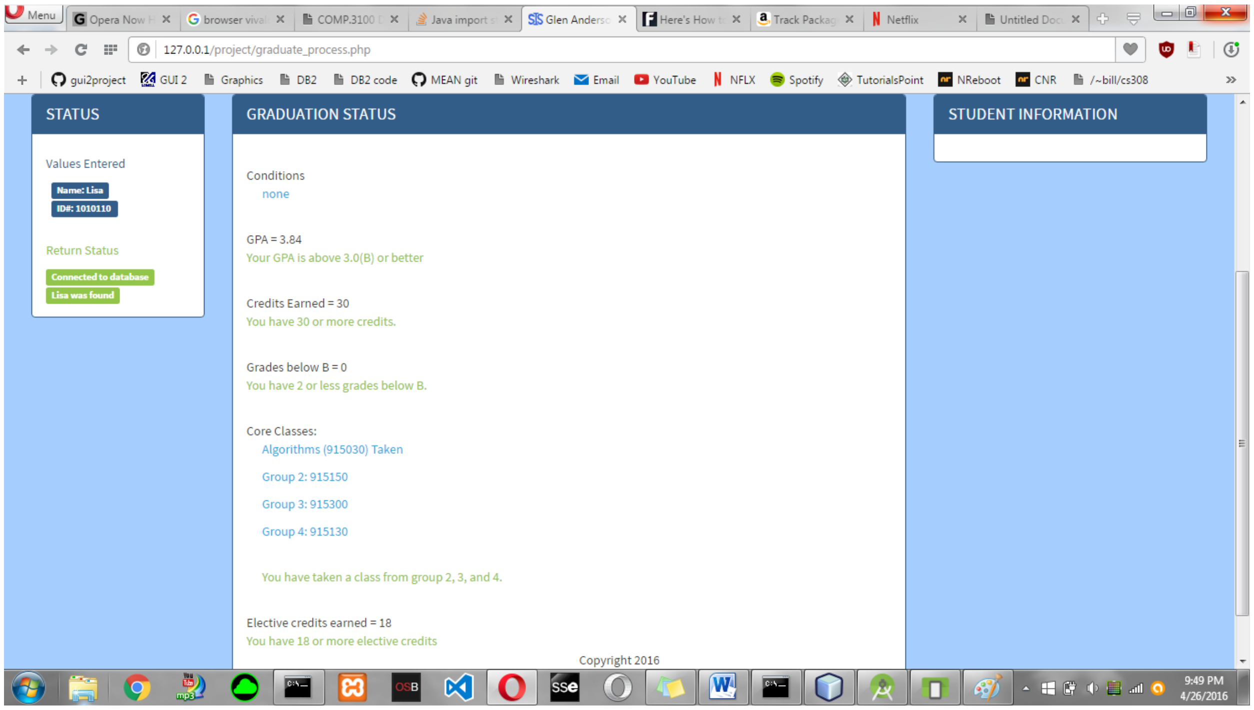
You cannot graduate because:

You cannot graduate because you have not taken a class from group 4.

STUDENT INFORMATION

Copyright 2016

Example 4


127.0.0.1/project/graduate_process.php

STATUS

Values Entered

Name: Lisa

ID#: 1010110

Return Status

Connected to database

Lisa was found

GRADUATION STATUS

Conditions

none

GPA = 3.84

Your GPA is above 3.0(B) or better

Credits Earned = 30

You have 30 or more credits.

Grades below B = 0

You have 2 or less grades below B.

Core Classes:

Algorithms (915030) Taken

Group 2: 915150

Group 3: 915300

Group 4: 915130

You have taken a class from group 2, 3, and 4.

Elective credits earned = 18

You have 18 or more elective credits

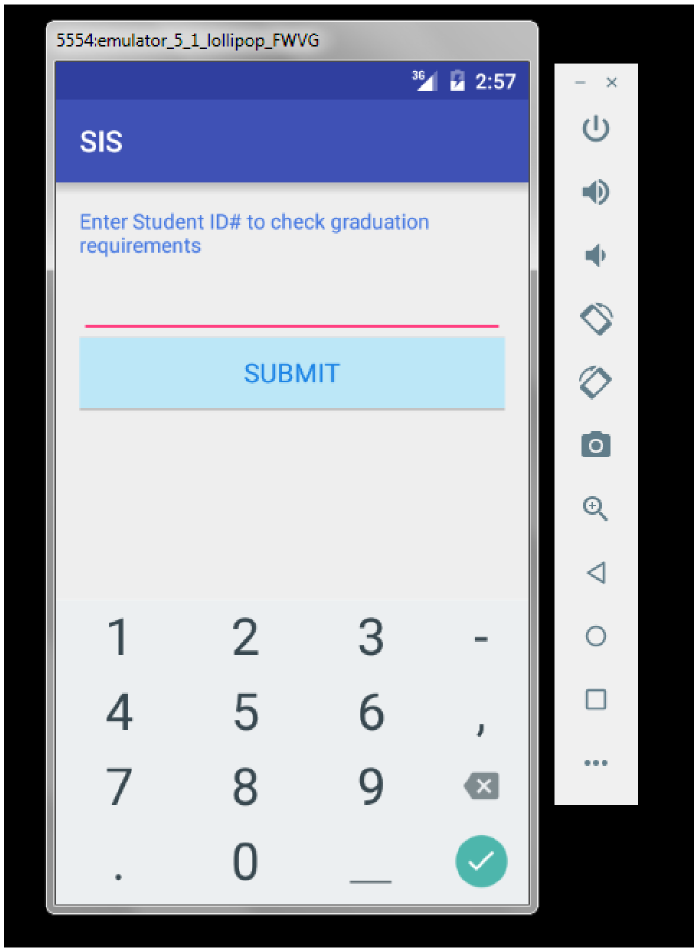
STUDENT INFORMATION

Copyright 2016

Example 5

Sample Outputs for Android-based Version of Project

First screen



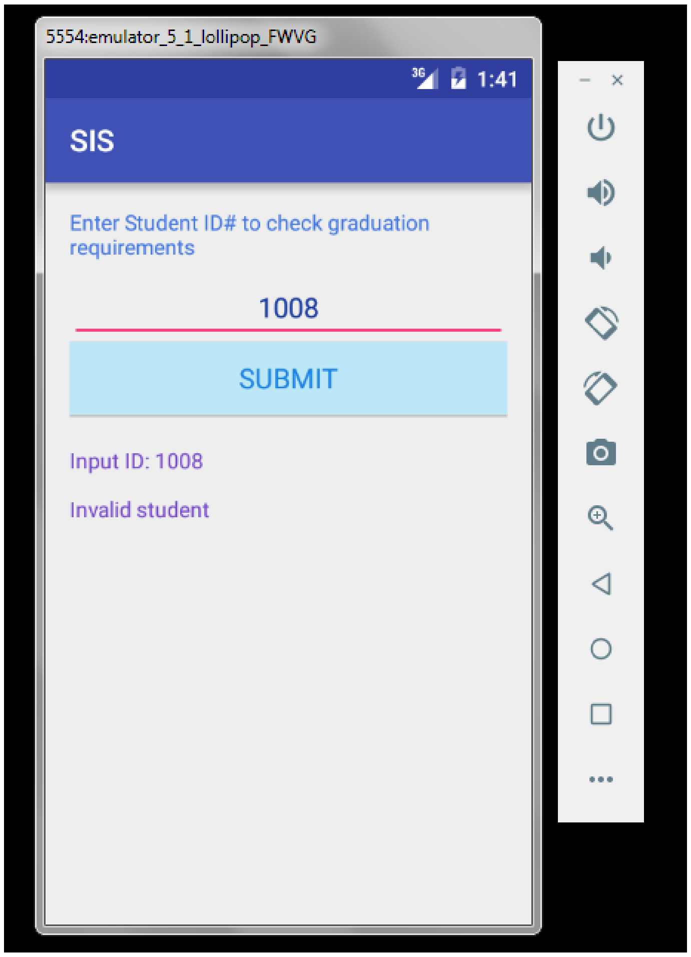
5554:emulator_5_1_lollipop_FWVG

2:57

SIS

Enter Student ID# to check graduation requirements

Invalid input



5554:emulator_5_1_lollipop_FWVG

1:41

SIS

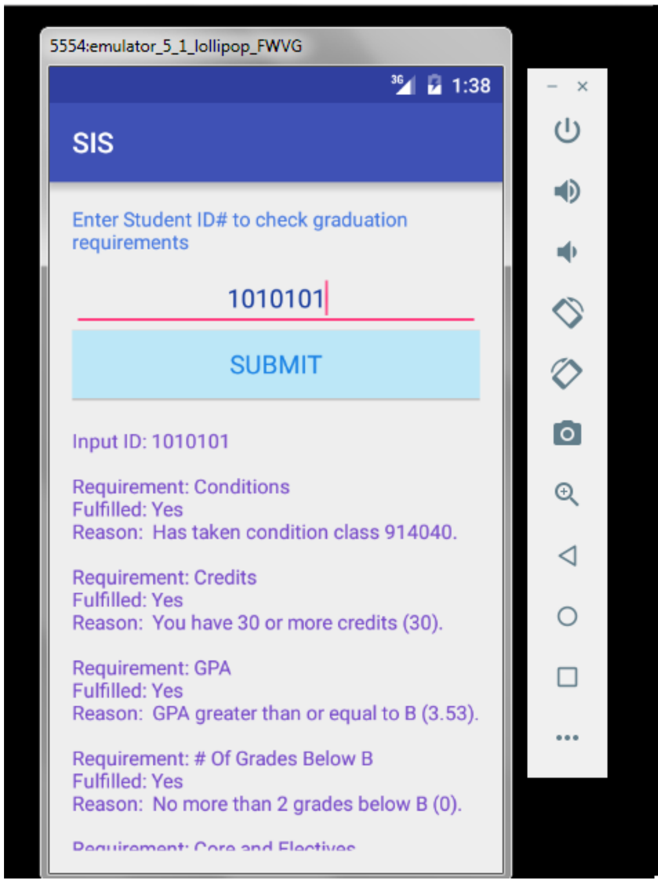
Enter Student ID# to check graduation requirements

1008

Input ID: 1008

Invalid student

Valid input (scrolled up)



5554:emulator_5_1_lollipop_FWVG

1:38

SIS

Enter Student ID# to check graduation requirements

1010101

SUBMIT

Input ID: 1010101

Requirement: Conditions
Fulfilled: Yes
Reason: Has taken condition class 914040.

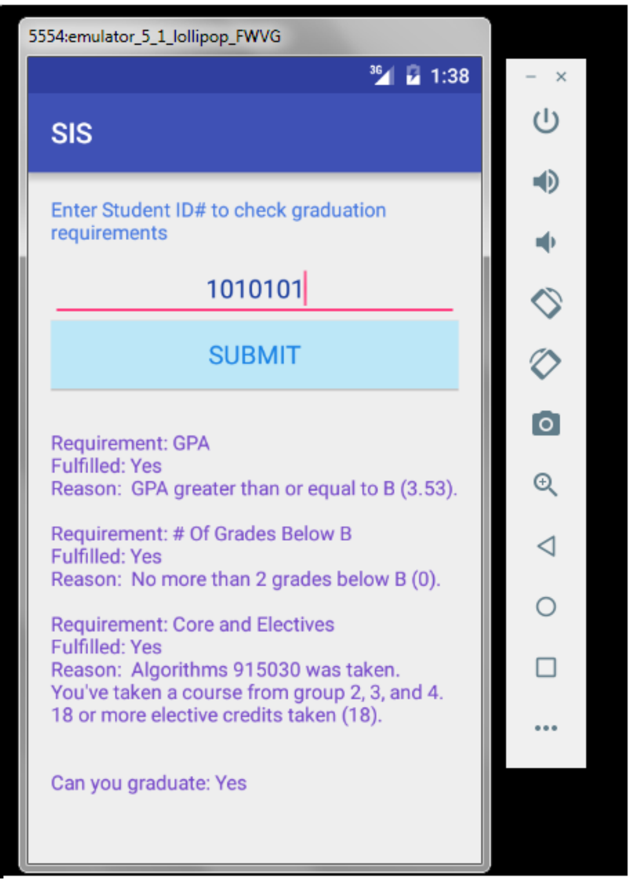
Requirement: Credits
Fulfilled: Yes
Reason: You have 30 or more credits (30).

Requirement: GPA
Fulfilled: Yes
Reason: GPA greater than or equal to B (3.53).

Requirement: # Of Grades Below B
Fulfilled: Yes
Reason: No more than 2 grades below B (0).

Requirement: Core and Electives

Valid input (scrolled down)



5554:emulator_5_1_lollipop_FWVG

1:38

SIS

Enter Student ID# to check graduation requirements

1010101

SUBMIT

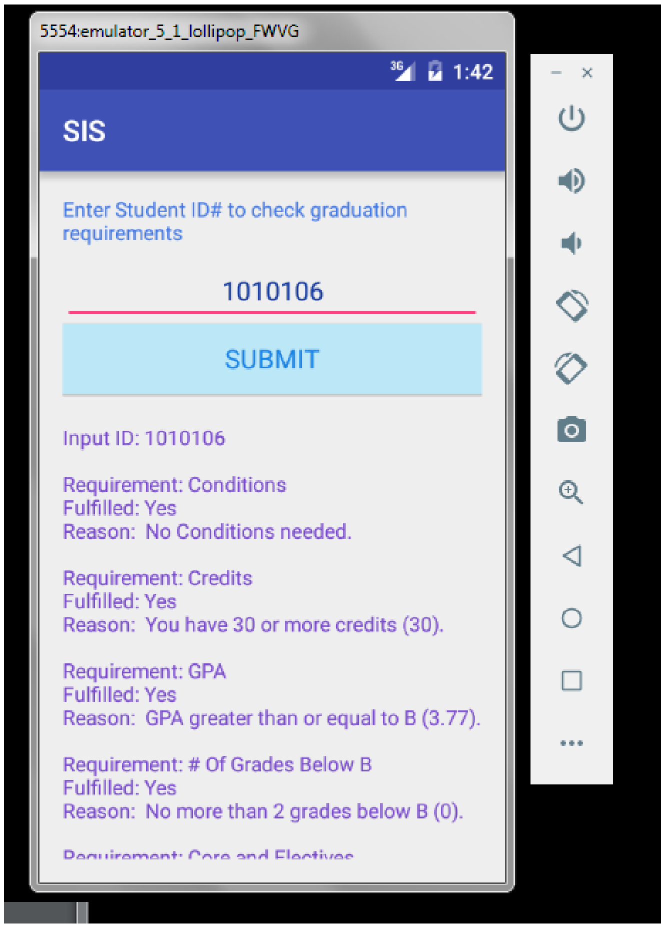
Requirement: GPA
Fulfilled: Yes
Reason: GPA greater than or equal to B (3.53).

Requirement: # Of Grades Below B
Fulfilled: Yes
Reason: No more than 2 grades below B (0).

Requirement: Core and Electives
Fulfilled: Yes
Reason: Algorithms 915030 was taken.
You've taken a course from group 2, 3, and 4.
18 or more elective credits taken (18).

Can you graduate: Yes

Valid input (scrolled up)



5554:emulator_5_1_lollipop_FWVG

1:42

SIS

Enter Student ID# to check graduation requirements

1010106

SUBMIT

Input ID: 1010106

Requirement: Conditions
Fulfilled: Yes
Reason: No Conditions needed.

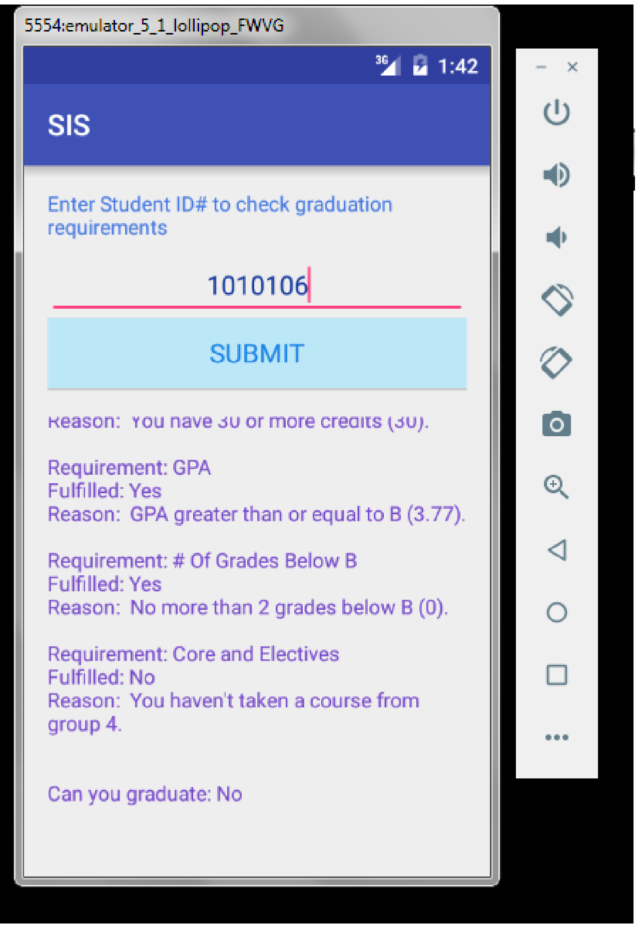
Requirement: Credits
Fulfilled: Yes
Reason: You have 30 or more credits (30).

Requirement: GPA
Fulfilled: Yes
Reason: GPA greater than or equal to B (3.77).

Requirement: # Of Grades Below B
Fulfilled: Yes
Reason: No more than 2 grades below B (0).

Requirement: Core and Electives

Valid input (scrolled down)



5554:emulator_5_1_lollipop_FWVG

1:42

SIS

Enter Student ID# to check graduation requirements

1010106

SUBMIT

Reason: You have 30 or more credits (30).

Requirement: GPA
Fulfilled: Yes
Reason: GPA greater than or equal to B (3.77).

Requirement: # Of Grades Below B
Fulfilled: Yes
Reason: No more than 2 grades below B (0).

Requirement: Core and Electives
Fulfilled: No
Reason: You haven't taken a course from group 4.

Can you graduate: No

Manual / Information on How to Run Code

For both versions of my project, make sure the folder titled ‘project’ is in the htdocs folder of the xampp folder. Also make sure that the MySQL and Apache server are currently running on the system.

<u>Browser-based version of project</u>

To run this version of my project, you will need the XAMPP framework. You will also need an internet connection in order for the bootstrap, jQuery (for bootstrap), and google font to be loaded from the CDN. Place the folder titled ‘project’ into the XAMPP htdocs folder. Then enter http://127.0.0.1/project/ in your browser’s address bar. The MySQL database that I used was set up with the username set to ‘root’, the password set to ‘’, and the database name set to ‘info’, so the database should be set up with those same parameters in order for the PHP to be able to connect to the database.

Technologies Used:

1. XAMPP framework
2. NetBeans IDE
3. Opera web browser
4. PHP, HTML, MySQL
5. Bootstrap CSS framework

<u>Android-based version of project</u>

To run this version of my project, the SIS2 folder will first need to be unzipped. This is because due to its large file size I needed to zip it in order to send it through the ‘submit’ system. Once it is unzipped, open the Android studio project titled ‘SIS2’ in Android Studio. Also, place the folder titled ‘project’ into the XAMPP htdocs folder as mentioned in the steps above. The MySQL database that I used was set up with the username set to ‘root’, the password set to ‘’, and the database name set to ‘info’, so the database should be set up with those same parameters in order for the PHP to be able to connect to the database Then, in Android Studio, open the AVD Manager, and start the Android Virtual Device (AVD). The specifications of AVD emulator that I used are as follows:

Phone: 3.7” FWVGA slider
Release Name: Lollipop
API Level: 22
Target: Android 5.1 (with Google APIs)

Once this AVD is created and started, the app can be run in the emulator. To begin, click the empty text field to type in a Student ID #.

Technologies Used:

1. Android Studio
2. Java
3. XAMPP framework, PHP, MySQL
4. org.json JSON parser