CSC 415-01

Course Lion

Brian Worts

11/4/2019

Table of Contents

Link to All Relevant LucidChart Diagrams:
https://www.lucidchart.com/invitations/accept/5346d2b1-559a-4741-8b5a-5eaafff86b10

Link to Github: https://github.com/brianworts/CourseLion

Pathname on VM (server24): local/home/sysadmin/CourseLion

Link to Wikipage:

Detailed Use Cases:

1:

Use Case:	Student makes their selections for courses
Iteration:	First
Primary Actor/s:	Student
Goal In Context:	The student will input their desired courses for next semester
Preconditions:	Student has a valid account; Student has not yet entered their desired courses;
Postconditions:	Student's inputs will be entered into the course selection database;
Trigger:	The student wants to enroll in courses for next semester
Scenario:	1. The student accesses the CourseLion website. 2. The student enters their email 3. The student enters their password 4. The system displays the homepage with links to pages containing major function buttons 5. The student selects the "Sign-up for courses" button 6. The system displays the "Sign-up" page 7. The student selects the number of courses they would like to be enrolled in (0 to 2) 8. The student selects up to 5 of their desired course codes from a dropdown of valid course codes 9. The student hits the "submit" button 10. The system reads the input from the input boxes as long as there is input in them

	11. The system saves this information to the student preferences database 12. The system takes the user to a “submission successful” page
Exceptions:	1. The ID or password combination is incorrect/invalid - see use case **Validate ID and password.** 2. The student has already inputted their desired courses- They will first be directed to a warning saying there is already input. If they continue than their choices will be overwritten in the database.
Priority:	High- this is one of basic functionalities
Chanel to Actor:	Via PC based browser with internet connection
Secondary Actors:	System administrator
Chanel to Secondary Actors:	System administrator- Via PC based browser with internet connection
Frequency of Use:	Once
Open Issues:	1. Is transmission of information to and from databases secure (information cannot be intercepted by third party)? 2. Are databases secure (hash passwords, strong admin passwords)?

2:

Use Case:	Student edits their selections for courses
Iteration:	First
Primary Actors:	Student
Goal In Context:	The student will view and edit their chosen courses for next semester

Preconditions:	Student has a valid account; Student has entered their desired courses;
Postconditions:	None
Trigger:	The student wants to view and edit their course selections for next semester
Scenario:	1. The student accesses the CourseLion website. 2. The student enters their email 3. The student enters their password 4. The system displays the homepage with links to pages containing major function buttons 5. The student selects the “View selections” button 6. The system displays the “View selections” page 7. The system accesses the database and displays the inputted choices made previously by the student 8. The student hits the “Edit selections” button 9. The student is directed to the “Sign-up” with their previously chosen selections filled in already 10. Follow steps 7-12 in **Student makes their selections for courses** use case
Exceptions:	1. The ID or password combination is incorrect/invalid - see use case **Validate ID and password.** 2. The student has not inputted their desired courses- The “View selections” page will display a message saying they have not inputted courses yet with a link to the “Sign-up” page
Priority:	Medium- Should be implemented after basic functionality
Chanel to Actor:	Via PC based browser with internet connection
Secondary Actors:	System administrator
Chanel to Secondary Actors:	System administrator- Via PC based browser with internet connection
Frequency of Use:	Infrequent

Open Issues:	1. Is transmission of information to and from databases secure (information cannot be intercepted by third party)? 2. Are databases secure (hash passwords, strong admin passwords)?

3:

Use Case:	Administrator runs the enrollment algorithm
Iteration:	First
Primary Actors:	Administrator
Goal In Context:	The administrator will run the enrollment algorithm to enroll the students in courses
Preconditions:	Admin has a valid account; Student preferences database is populated (will still function without this but won't do anything); Course constraints database is populated;
Postconditions:	Students database is populated with each course a student is enrolled in; Courses database is populated with each student for each course;
Trigger:	The administrator wants to enroll all students
Scenario:	1. The admin accesses the CourseLion website. 2. The admin enters their email 3. The admin enters their password 4. The system displays the admin homepage with links to pages containing major function buttons 5. The admin selects the "run enrollment" button 6. The system displays a warning saying that this action cannot be reversed 7. The admin selects the "acknowledge" button 8. The system runs the enrollment algorithm

	9. The system populates the Students and Courses databases with the results of the algorithm 10. The system directs the admin to the “enrollment success” page
Exceptions:	1. The ID or password combination is incorrect/invalid - see use case **Validate ID and password.** 2. The student has not inputted their desired courses- The “View selections” page will display a message saying they have not inputted courses yet with a link to the “Sign-up” page 3. The system cannot access the databases- display appropriate message 4. The enrollment algorithm has already run- No error, will overwrite previous data
Priority:	High- this is one of basic functionalities
Chanel to Actor:	Via PC based browser with internet connection
Secondary Actors:	Students
Chanel to Secondary Actors:	System administrator- Via PC based browser with internet connection
Frequency of Use:	Once
Open Issues:	1. Is transmission of information to and from databases secure (information cannot be intercepted by third party)? 2. Are databases secure (hash passwords, strong admin passwords)? 3. Is algorithm efficient? 4. Is the algorithm working properly?

4.

Use Case:	Student views what courses they were enrolled in
Iteration:	First

Primary Actors:	Student
Goal In Context:	The student will view what courses they were enrolled in
Preconditions:	Student has a valid account; Student has entered their desired courses before algorithm ran; Enrollment Algorithm ran by admin;
Postconditions:	None
Trigger:	The student wants to view what courses they were enrolled in
Scenario:	1. The student accesses the CourseLion website. 2. The student enters their email 3. The student enters their password 4. The system displays the homepage with links to pages containing major function buttons 5. The student selects the “View Enrollment” button 6. The system displays the “View Enrollment” page 7. The system accesses the appropriate databases and displays the student’s choices, whether they were enrolled or not, and if not, why they were not enrolled
Exceptions:	1. The ID or password combination is incorrect/invalid - see use case **Validate ID and password.** 2. The student did not input their desired courses before the algorithm ran- System will display appropriate message on “View enrollment” page 3. The enrollment algorithm has not run yet (Students database is empty)- System will display appropriate message on “View enrollment” page
Priority:	High- this is one of basic functionalities
Chanel to Actor:	Via PC based browser with internet connection
Secondary Actors:	System administrator
Chanel to Secondary Actors:	System administrator- Via PC based browser with internet connection

Frequency of Use:	Infrequent
Open Issues:	1. Is transmission of information to and from databases secure (information cannot be intercepted by third party)? 2. Are databases secure (hash passwords, strong admin passwords)? 3. Is algorithm efficient? 4. Is the algorithm working properly?

5.

Use Case:	Administrator views results of enrollment algorithm
Iteration:	First
Primary Actors:	Administrator
Goal In Context:	The administrator will view the results of the enrollment algorithm
Preconditions:	Admin has a valid account; Enrollment algorithm has successfully ran
Postconditions:	None
Trigger:	The administrator wants to view the results of the enrollment algorithm
Scenario:	1. The admin accesses the CourseLion website. 2. The admin enters their email 3. The admin enters their password 4. The system displays the admin homepage with links to pages containing major function buttons 5. The admin selects the “view enrollment results” button 6. The system directs the admin to the “View results” page 7. The admin selects whether they want to view course results or student results 8. The system accesses the appropriate database and displays a table showing the data

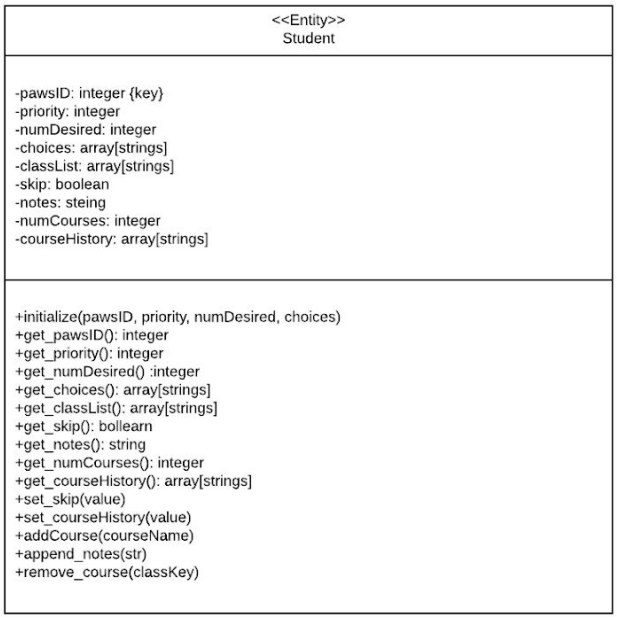
Exceptions:	1. The ID or password combination is incorrect/invalid - see use case **Validate ID and password.** 2. The enrollment algorithm has not run yet (Students database is empty)- System will display appropriate message on “View results” page
Priority:	Medium- This should be implemented after basic functionality since there are already ways for the admin to access the database directly
Chanel to Actor:	Via PC based browser with internet connection
Secondary Actors:	Students
Chanel to Secondary Actors:	System administrator- Via PC based browser with internet connection
Frequency of Use:	Infrequent
Open Issues:	1. Is transmission of information to and from databases secure (information cannot be intercepted by third party)? 2. Are databases secure (hash passwords, strong admin passwords)?

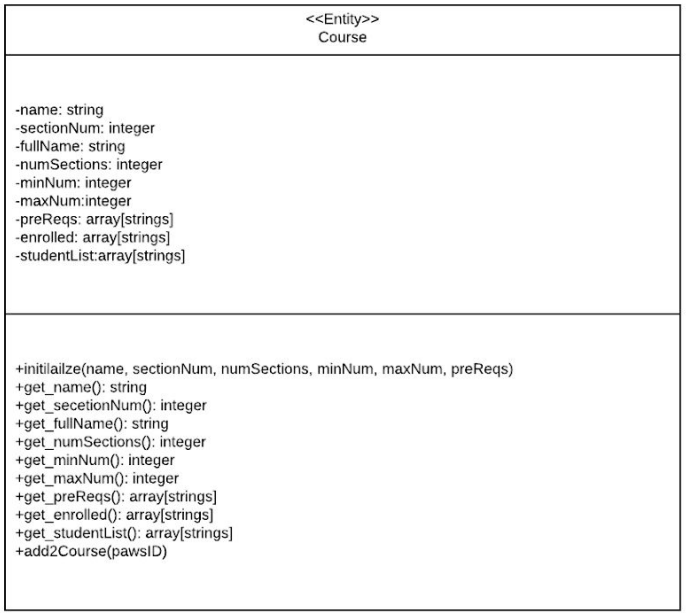
6.

Use Case:	Validate ID and password.
Iteration:	First
Primary Actors:	Student; Admin
Goal In Context:	The system will validate the username/password combination
Preconditions:	User has a valid account in the database
Postconditions:	Valid user is directed to the homepage; Invalid user is given error message

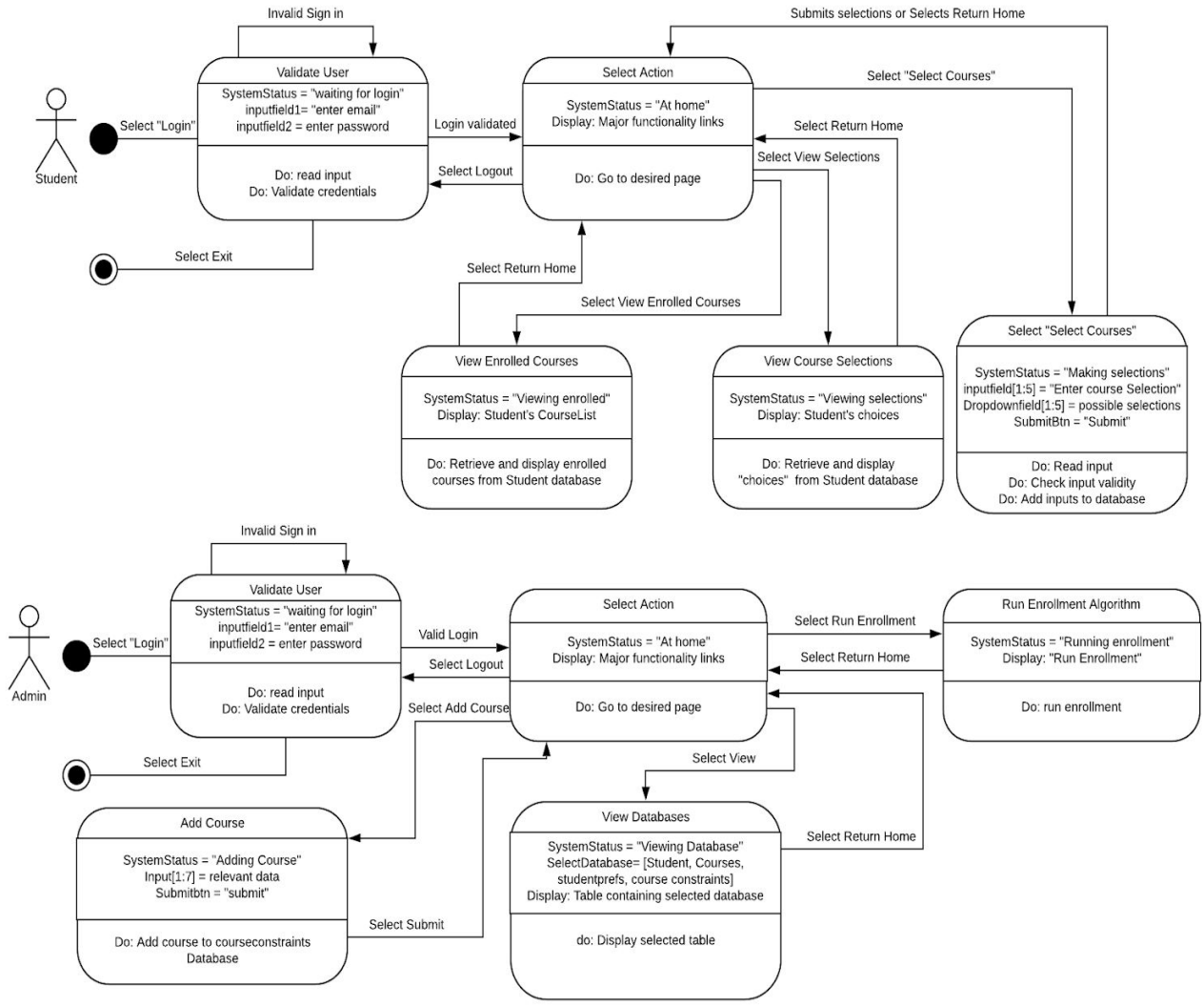
Trigger:	The user wants to access the website
Scenario:	1. User inputs their credentials into the appropriate areas. 2. System reads the inputted information 3. System cross references databases to check if email is valid 4. System encrypts password and checks whether matches encrypted password in database 5. User is directed to the home page if match or given error message if failure
Exceptions:	1. User hits submit without inputting anything- system gives invalid login message
Priority:	High- this is part of the basic functionality
Chanel to Actor:	Via PC based browser with internet connection
Secondary Actors:	None
Chanel to Secondary Actors:	NA
Frequency of Use:	Frequent
Open Issues:	1. Is transmission of information to and from databases secure (information cannot be intercepted by third party)? 2. Are databases secure (hash passwords, strong admin passwords)?

Detailed Design Class Diagram

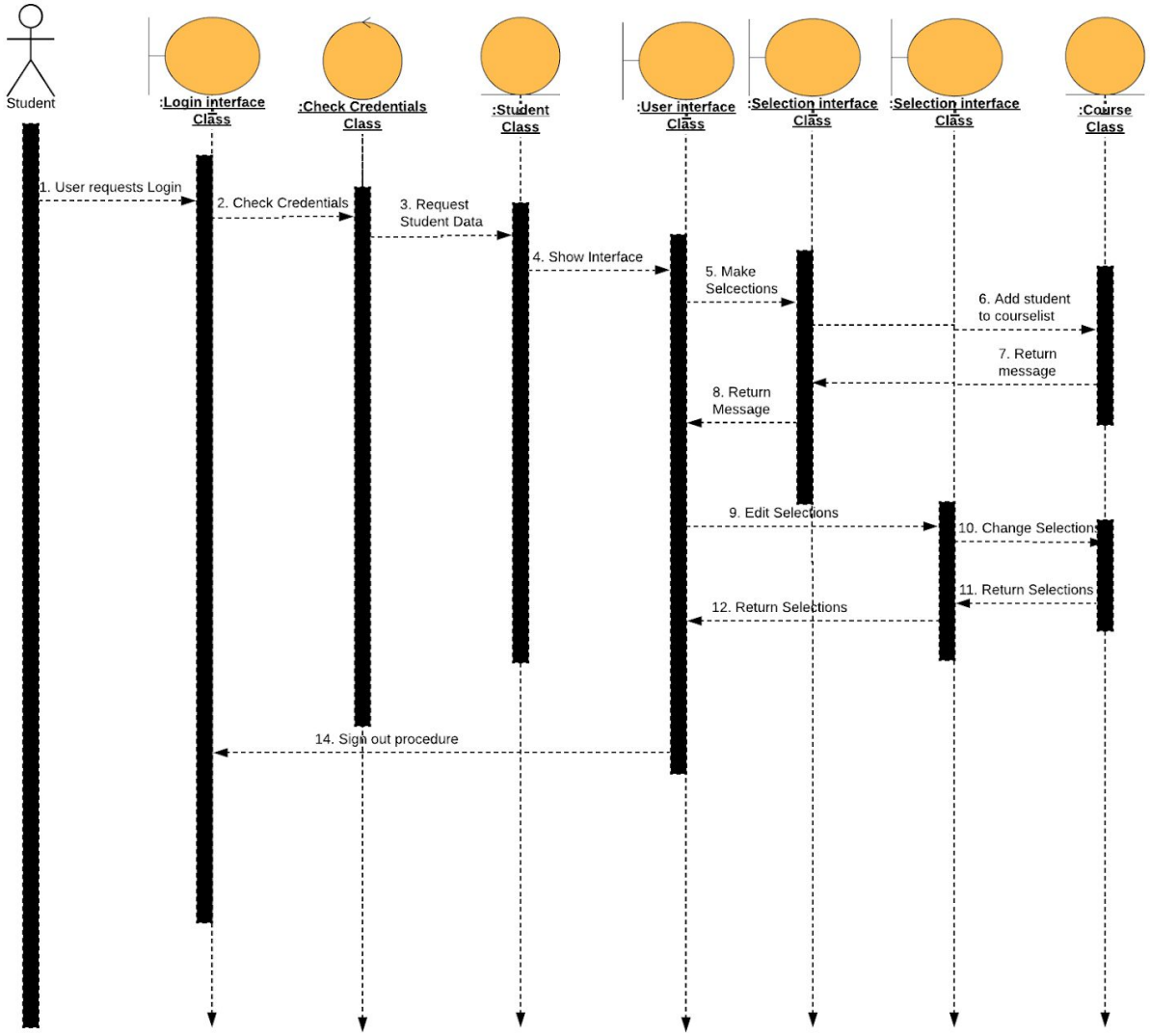


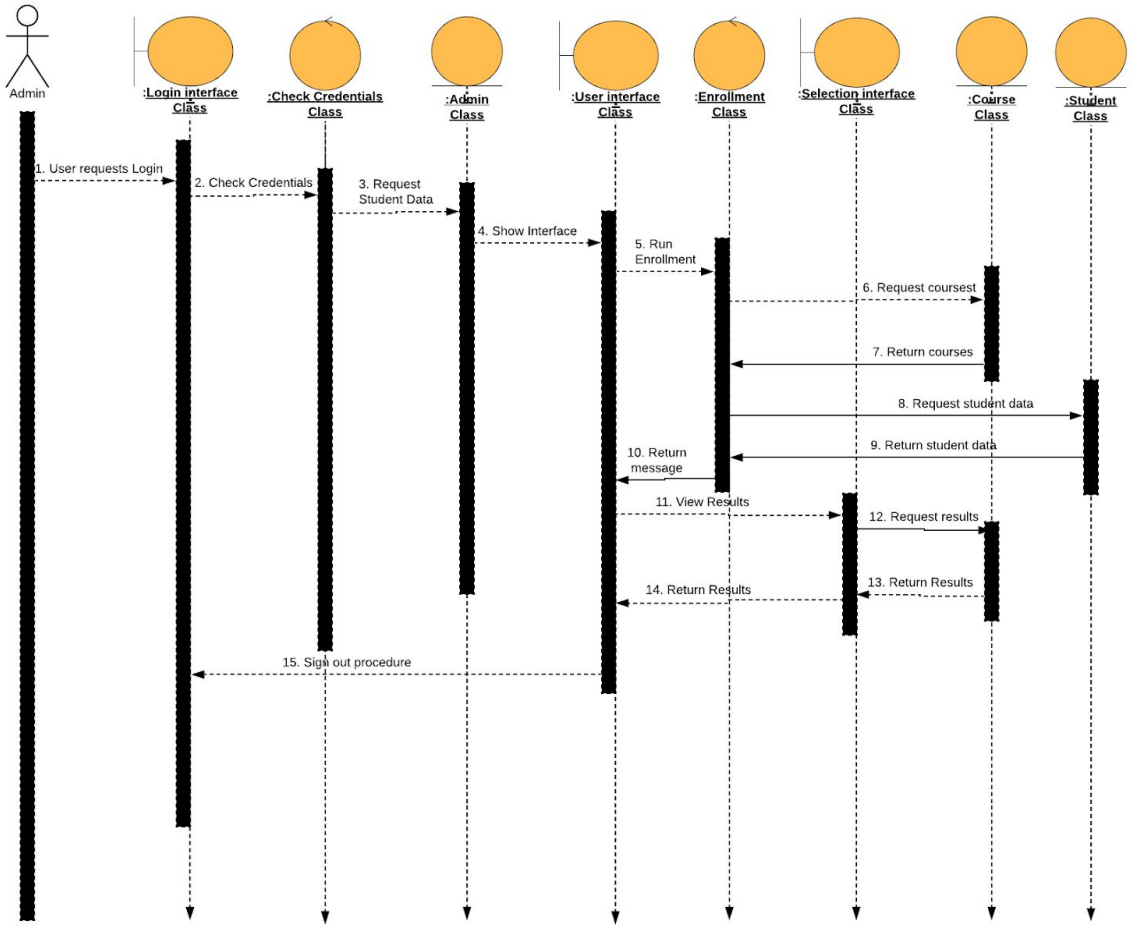


Statecharts

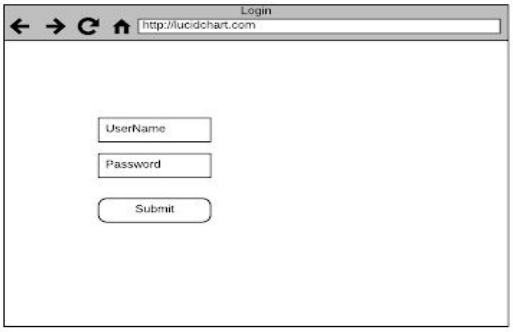


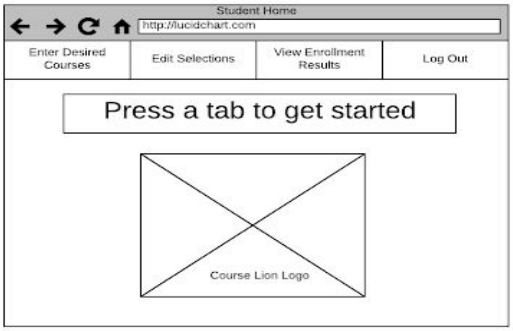
System Sequence Diagrams

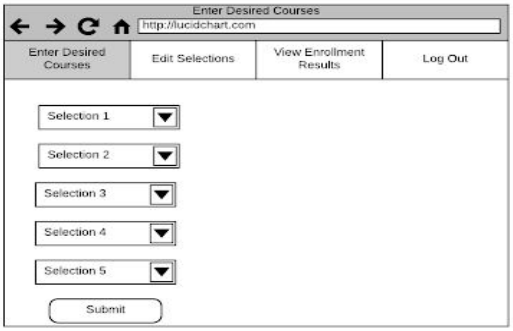


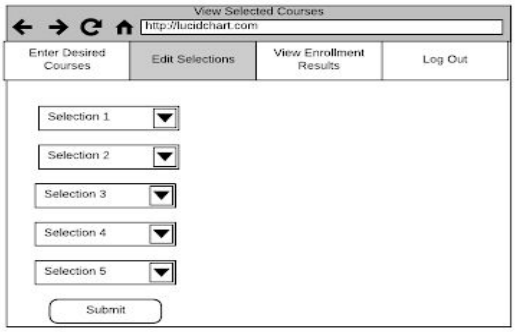

Admin
:Login interface Class
:Check Credentials Class
:Admin Class
:User interface Class
:Enrollment Class
:Selection interface Class
:Course Class
:Student Class
1. User requests Login
2. Check Credentials
3. Request Student Data
4. Show Interface
5. Run Enrollment
6. Request coursest
7. Return courses
8. Request student data
9. Return student data
10. Return message
11. View Results
12. Request results
13. Return Results
14. Return Results
15. Sign out procedure

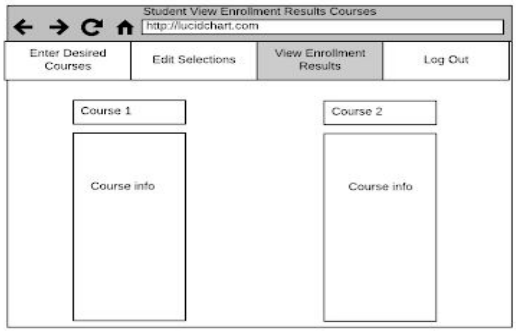
2. UI Mockups

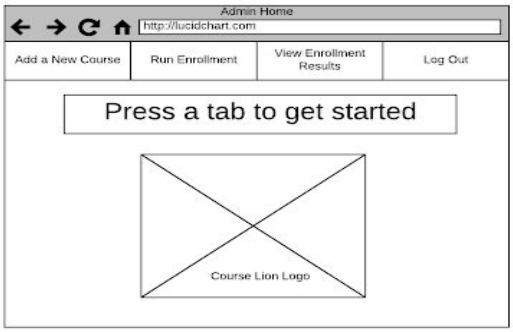

Login
http://lucidchart.com
UserName
Password
Submit

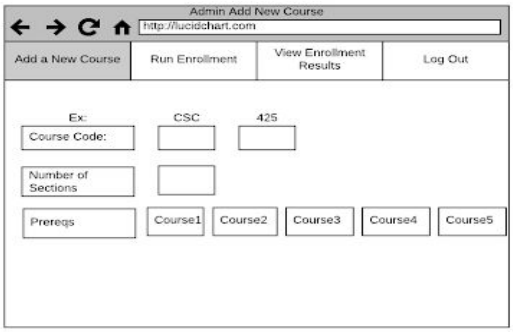

Student Home
http://lucidchart.com
Enter Desired Courses
Edit Selections
View Enrollment Results
Log Out
Press a tab to get started
Course Lion Logo

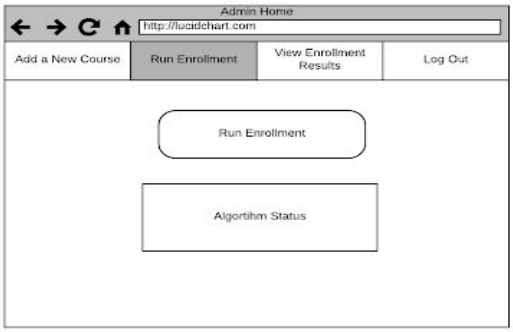

Enter Desired Courses
http://lucidchart.com
Enter Desired Courses
Edit Selections
View Enrollment Results
Log Out
Selection 1
Selection 2
Selection 3
Selection 4
Selection 5
Submit

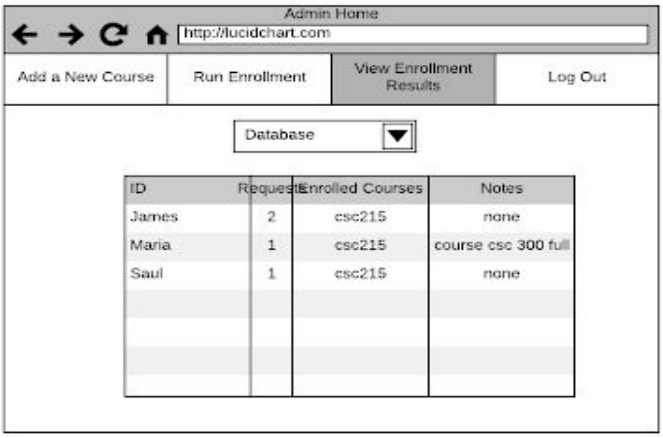

View Selected Courses
http://lucidchart.com
Enter Desired Courses
Edit Selections
View Enrollment Results
Log Out
Selection 1
Selection 2
Selection 3
Selection 4
Selection 5
Submit


Student View Enrollment Results Courses
http://lucidchart.com
Enter Desired Courses
Edit Selections
View Enrollment Results
Log Out
Course 1
Course info
Course 2
Course info


Admin Home
http://lucidchart.com
Add a New Course
Run Enrollment
View Enrollment Results
Log Out
Press a tab to get started
Course Lion Logo


Admin Add New Course
http://lucidchart.com
Add a New Course
Run Enrollment
View Enrollment Results
Log Out
Ex:
CSC
425
Course Code:
Number of Sections
Prereqs
Course1
Course2
Course3
Course4
Course5


Admin Home
http://lucidchart.com
Add a New Course
Run Enrollment
View Enrollment Results
Log Out
Run Enrollment
Algorithm Status



The UI's were designed to be as simple and easy for the user to understand as possible. In order to follow the principle of "visibility," the UI was designed to give all the functionality a user would need. In order to follow the affordance principle, the controls are as self explained as possible through their labels and locations.

In order to follow the eight golden rules...

1. Strive for consistency: The top bar of the UI will not change no matter what page you are on. All forms have a similar format.
2. Enable Frequent User Shortcuts: The top bar of the UI gives users a constant link to any functionality they may need
3. Offer Informative Feedback: UI will give all necessary information to user
4. Design Dialogs to Yield Closure: UI will tell a user when a task is complete
5. Offer Simple Error Handling: The UI will use drop downs to prevent user input errors while also performing checks on inputs such as length requirements
6. Permit Easy Reversal of Actions: All commands can be changed via the UI and the ones that can't have warnings telling the user so
7. Support Internal Locus of Control: Flow of pages makes sense for a user and there is no confusing mapping of pages.
8. Reduce Short-Term Memory Load: Information is easily accessible by the user and there is little to nothing for a user to memorize

3. How is program meeting requirements?

1) Modularity and encapsulation to facilitate information hiding and reuse
    a) So far, a significant portion of the project time has been dedicated to refactoring the project. Mostly, the original project contained functions that were far too large. These functions have been split up into smaller functions that are now called individually and in new places, to allow the enrollment algorithm to be more functional. Classes for the Student and the Course have been revised to encapsulate more functionality and more data in one data structure. All the data within these classes are accessible only through getters and setters so that objects of one class cannot call the functions of other classes.
2) Elegance and efficiency of the algorithms
    a) The algorithms used are heavily based on the order and type of the data. This allows them to be as efficient and elegant as possible. For example, the database containing student choices is numerically ordered, so a binary search is used here. The database containing the courses does not have a numeric order, so a linear search is used. Thus, the algorithms are effective, efficient, and simple while also doing their jobs.
3) Appropriateness of the data structures used for the problem
    a) In the original project, there were many points where strings were used to store data when an array of strings would have been much more effective since it allows for significantly faster searching so this has been changed. The primary structure of the program will remain as a hash table. This structure is used for when the whole database needs to be loaded into a structure, and then can be quickly searched via the hash keys with a O(1) search time which is the primary operation done to it.

Test Cases

Unit- In order to test the individual units, I will isolate each unit and manually pass values in that may work or may cause errors, and make sure the output is as expected.

Integration- I will start with isolating one unit and slowly add the other units while passing values into the original. Once completed, I will perform this process again but start with a different unit.

System- Once completed, put as many incorrect inputs as I can into the source databases as possible and make sure the system handles each one correctly.

Debugging- I plan on using the gdb debugger to debug and step through my Ruby program to test my test cases. This won't cover all the debugging I need to do though, since some of the processes are too large to be able to review manually such as testing the reading and writing to the data files. However, this can be easily done by viewing the number of inputs and outputs of the program.

Test Cases:

Functionality Tested	Inputs	Expected Output	Actual Output
Retrieving student input	Course selections	System should read the inputted courses, valid or not, into the students database	
Student edits their choices	Course selections	System should show which inputs have already been entered by that student. System should read new values and overwrite old values in the students database	
Student views the courses they were enrolled in (before enrollment)	N/A	System should tell student that enrollment has not performed enrollment yet	
Admin adds a new course	New course information	System should read all information passed in related to the course, perform checks to test for consistency with other courses, warn admin if not consistent, and allow the admin to make the ultimate decision to add the course.	
Admin runs enrollment	N/A	Upon pressing the enrollment button, system should read in from student and courses databases, and write to the proper databases.	
Student views the courses they were enrolled in (after enrollment)	N/A	System should pull from students database and display the classes the student was enrolled in, and why they were not enrolled in the others.	
Admin views results of algorithm	Selection of which database to view	The admin will select which database to view and based on their input, that database will display all necessary info (not confidential info like passwords).	

Open Source Maintenance and Communication

Please review documentation before asking questions

Thorough documentation is required for all changes made to the project

All contributions will be reviewed by maintainers before being accepted, and maintainers will need to unanimously agree on whether to accept it or not. Contributors will be notified of the final decision or any changes they may need to make to get their contribution accepted.

Report bugs or questions on the "issues" page

Contributions regarding higher priority issues generally will be looked at and voted on before lower priority issues

Open Source Licensing

	Apache License 2.0	GNU GPLv3	MIT License
Source Code	Required to release all unmodified parts of the software. Must list changes to source code. Derivative works do not need to be under the same license.	Must release all modified and source code of software developed using this license if releasing a version	No requirements other than adding the original copyright notice to all future developments
Distributing altered versions	Anyone is allowed to profit off of a project developed from this source	All released improved versions must be free	Anything can be done with altered versions including selling or releasing it as open source
Best use	Best used for projects where the original developer does not care what others do with it, but also wants their code to be visible in the altered projects	Best used if developing a project and don’t want anyone to profit off of the original work and want to ensure that future iterations stay open	Best used if the original developer want others to build on their work and doesn’t care what others do with it

The Apache License will work best for my project since I want a hands-off approach where I will get credit for what I did, but also does not restrict someone from innovating and improving my idea. I think using the GNU GPL v3 might stop someone else from innovating off my project since they would have to release their project under this license as well. I also want my source code in the finished project if it was not changed, which is why I did not choose the MIT license.