Lab 1 - CORSICA Product Description

Red Team

Latimer Gerle

CS411W

Professor Janet Brunelle

September 11, 2014

Version 1

Table of Contents

List of Figures

List of Tables

1 INTRODUCTION

At the start of each academic term at institutions of higher learning across the country, students attempt to register for courses that are already filled to capacity. This causes headaches for both students and faculty; the students pester the instructors and advisors to be allowed into closed classes, and the faculty is barraged by emails, phone calls, and visitations from worried students. This cycle, which continues until the students are either let into their desired class or give up, leads to wasted time and frustration for both parties. And as the number of people seeking bachelor's degrees continues to increase, the aforementioned situation will occur more often. Between 2001 and 2011 alone, students enrolled in post-secondary institutions increased by thirty-two percent ("Postsecondary Education", 2012). Meanwhile, the number of spots available for enrollment is not keeping pace, with the number of new post-secondary institutions increasing by about ten percent during the same time period (National Center for Education Statistics, Table 5).

Waitlists are one solution for decreasing the frustration and excessive communication that results from students wishing to register for closed courses. This is an option that numerous universities offer, and while the details differ depending on the specific institution, the basic implementation is the same; if a class is full, a student can place him or herself on the waitlist, and he or she will be notified when a seat opens up. However, this is not a perfect solution for those students wishing to register for a closed course. First of all, these waitlists often have certain restrictions; for example, a student has to be in a particular grade level or have a certain number of credits. Secondly, not all universities have course waitlists, and those that do typically offer them only within certain departments or colleges. And finally, the waitlists lack basic features, such as the ability to view your position or be notified via text.

CORSICA, the COmputeR ScIence Course wAitlist, was designed to be an enhanced and more efficient waitlist. The hallmarks of CORSICA, a graphical user interface (GUI), ability to handle labs and recitations, and improved notification system, combine to make a simple and streamlined process for both students and instructors. Initially developed for the Computer Science department of Old Dominion University, it will be available for use by any post-secondary institution.

(This space intentionally left blank.)

2 PRODUCT DESCRIPTION

CORSICA is a course waitlist system through which students can wait for an enrollment spot to become available in a full course. When a spot does become available, all of the students on the waitlist are notified, and the student who occupies the highest position on the waitlist is allowed to register for the course. CORSICA is utilized by way of a GUI, through which students can add themselves to the waitlist for a particular course if it has reached the total enrollment capacity. Via the GUI, students can also remove themselves from a waitlist and view their current position in relation to number one on the list.

2.1 Key Product Features and Capabilities

CORSICA is comprised of three principal features: a notification system, an ability to handle labs and recitations, and a GUI. The notification system is the most significant feature of CORSICA. The majority of course waitlists currently in use at postsecondary institutions function the same way: when a registration spot becomes available, the student at the top of the waitlist is notified via email and given twenty-four hours to register in that class. CORSICA, however, presents an original and efficient way of notifying students. When a registration spot opens up, all of the students on the waitlist are notified and given twenty-four hours to respond; they can choose to be notified via email, text, or both. Those students who respond are kept on the waitlist, while those that do not are removed, and the remaining students are moved up the list. Of those students who respond, the student who holds the highest position on the waitlist is allowed to register for the course.

Another important aspect of CORSICA is its GUI. Dependent upon the current user, the CORSICA GUI has numerous capabilities. The supported users are Visitors, Students, Schedulers, Advisors, and Administrators. Visitors can only view basic information pertaining to

CORSICA, such as facts and contact information, while students can add or drop themselves from one or multiple waitlists, view their position on the waitlists they are currently on, and see which courses offer waitlists. Schedulers have the ability to start or end a waitlist for a course, as well as view the statistics for any given waitlist. Advisors can add or remove students from waitlists, view waitlists, and change students positions on waitlists. Finally, Administrators can exercise all of the above abilities; in addition, they can add or delete users as needed.

A final key feature of CORSICA is the ability to handle course lectures, labs, and recitations. If a course has more than a single lecture component, a student can either add himself to waitlists for all available sections, or he can add himself to the waitlist for a specific Course Registration Number. If a course includes a lab and/or recitation component, a student cannot add himself to the lecture component alone; trying to do so will result in an error. He must add himself to the waitlist of at least one section of each course component.

2.2 Major Components (Hardware/Software)

CORSICA is a software service comprised of four main components: a database, various back-end algorithms, a GUI, and a notification system. The database is a vital part of the system; without it, there is no information regarding the waitlist. It will utilize MySQL, and will contain all of the information relevant to CORSICA, such as administrator names, course numbers, and student ID numbers. The GUI, which would be driven by the algorithms, is where users would perform different actions based upon their login credentials. And the notification system, also driven by the algorithms, would alert students of an opening in a class.

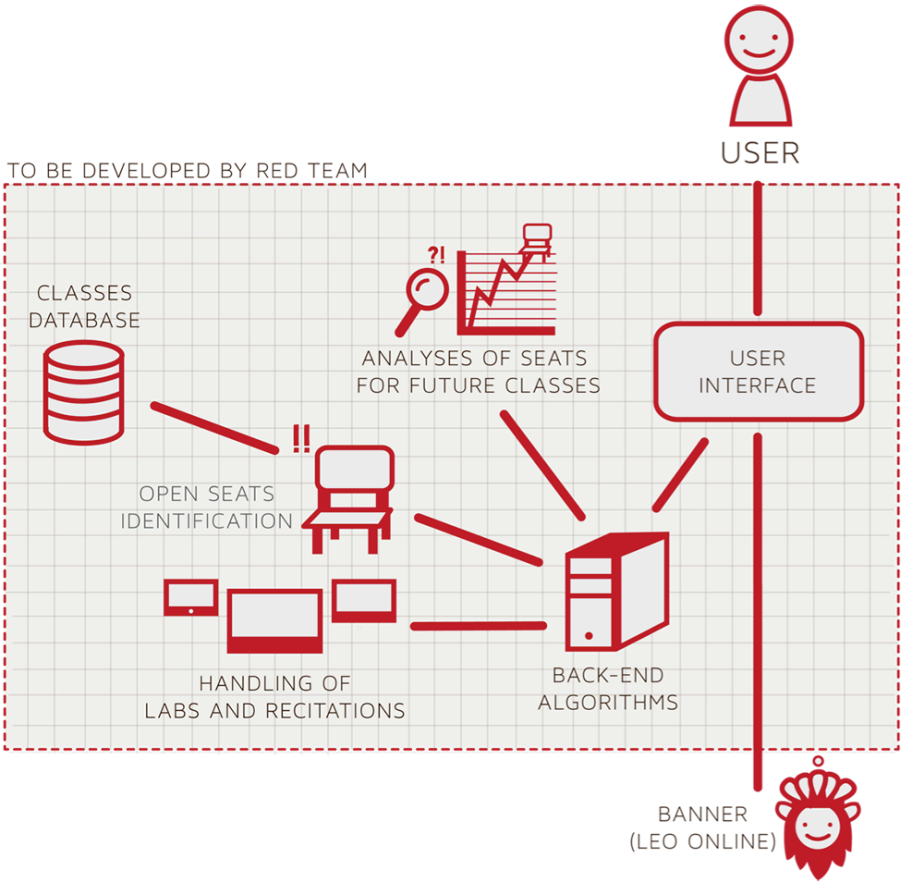

Figure 1. CORSICA Major Components

In addition to the software components, CORSICA requires three hardware components: a computer, an Internet connection, and a server. The computer and Internet connection are required in order to view and connect to the Internet, respectively. The server will consist of both a web and database server. The web server will be used to display and deliver web pages, while the database server will hold the database.

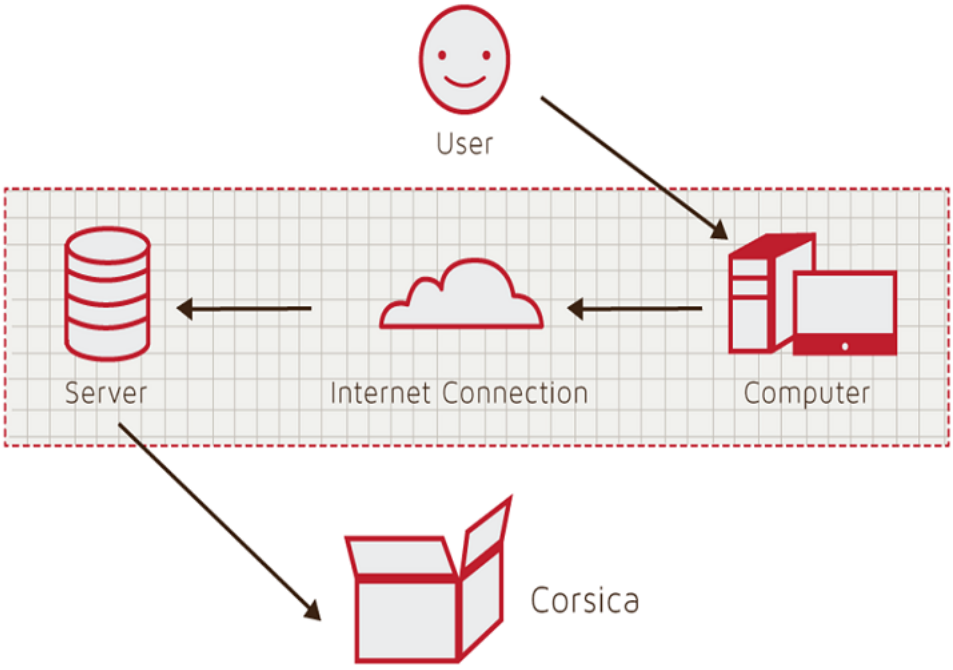

Figure 2. CORSICA Hardware Components

(This space intentionally left blank.)

3 IDENTIFICATION OF CASE STUDY

Professor Janet Brunelle, the Assistant Chair and Chief Advisor of the ODU Computer Science department, initially suggested the idea for CORSICA, and recommended developing it in conjunction with Dr. Irwin Levinstein. Dr. Levinstein is an Associate Professor in Computer Science at ODU, as well as the course scheduler for the department. Professor Brunelle explained that the current registration process is quite flawed. The Computer Science courses typically fill up quickly; therefore, any student who is delayed in registering may find him or herself unable to sign up for all of his or her desired classes. This in turn leads to the students consistently checking the course enrollment figures and emailing advisors and faculty with requests to be allowed into the class or classes; the advisors and faculty must then take time away from other tasks to respond to these requests. This time-consuming cycle, illustrated in Figure 3, repeats itself until the registration period is over.

(This space intentionally left blank.)

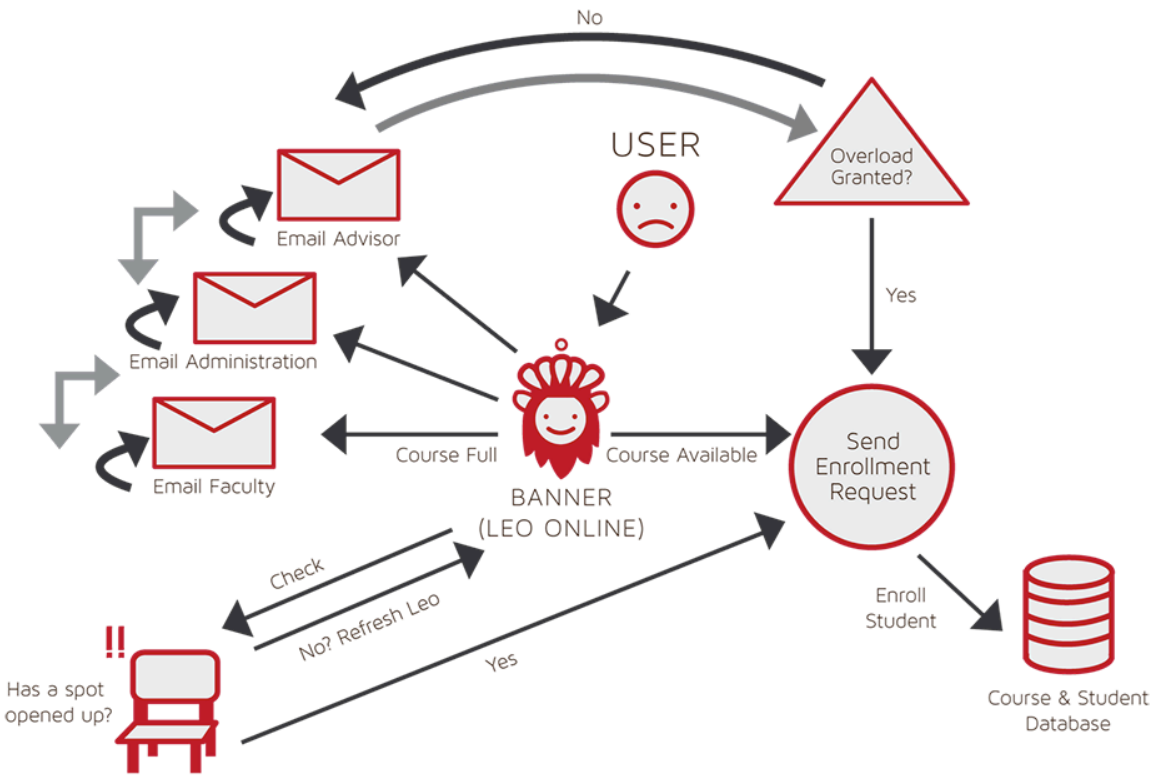

Figure 3. Current Registration Process

In order to resolve this issue, Professor Brunelle recommended developing a waitlist for ODU Computer Science courses. As shown in Figure 4, CORISCA would stop students from continually checking enrollment numbers, and it would also cease the constant emails to advisors and faculty requesting course overrides. Thus, the cycle from Figure 3 would transform into a simplified and less stressful process. In addition, the analysis component of CORSICA will be of assistance to Dr. Levinstein in that it will determine how many seats should be added or removed from future Computer Science courses. Upon successful testing and demonstration, CORSICA will be available for use by any post-secondary institution wishing to utilize it.

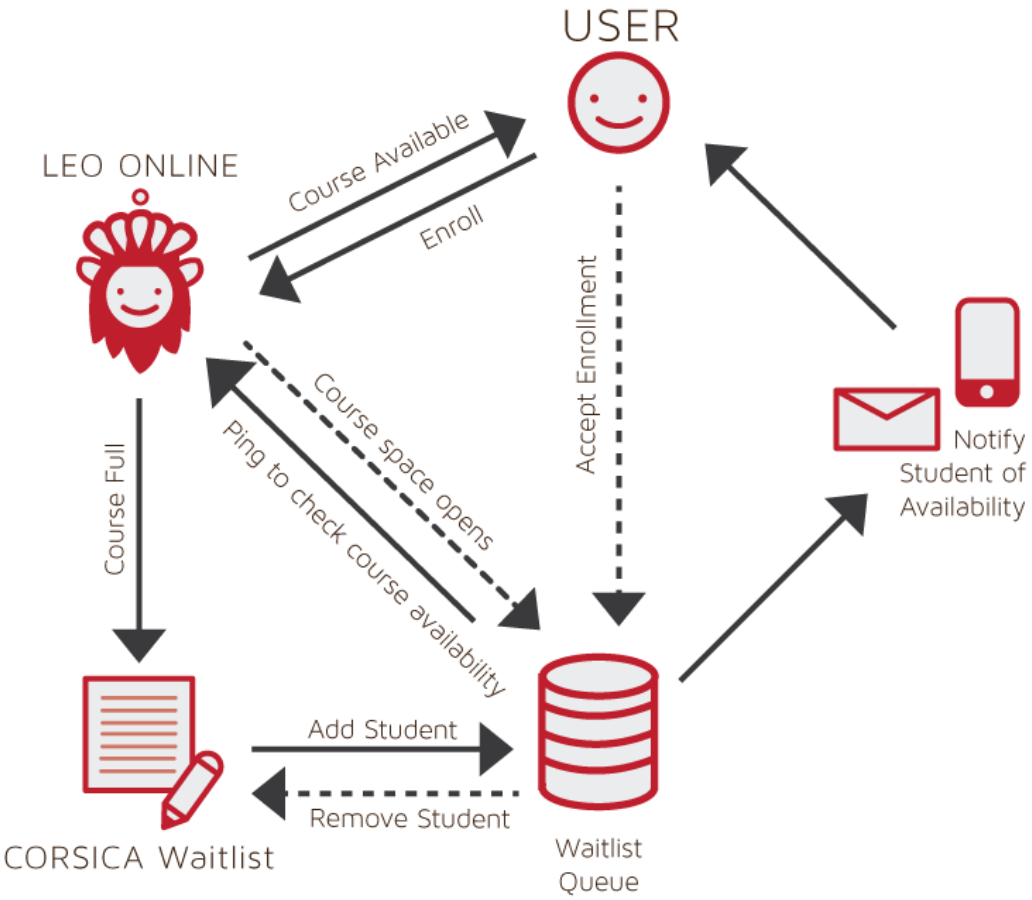

Figure 4. Improved Registration Process Utilizing CORSICA

(This space intentionally left blank.)

4 PRODUCT PROTOTYPE DESCRIPTION

The CORSICA prototype is designed to illustrate the concept that its' improved and efficient waitlist system will save both time and frustration for students and faculty. The prototype will accomplish this by showing that the CORSICA waitlist process will render it unnecessary for students to stay in constant communication with faculty and advisors when they wish to register for a closed course. Thus, the time that would be spent on these communications will now be free for other purposes.

One objective of the prototype is to successfully demonstrate that students will be notified when a registration spot opens in a class. However, the prototype will not display the end product notification system; rather, it will be simulated by integrating with that of ODU. The ODU notification system works by alerting students of emergencies and school closures via text and email.

Another objective of the CORSICA prototype is to exhibit the waitlist functionality. This will be achieved by filling the database with data files containing dummy student information. Once the database is populated, the prototype will demonstrate a simplified version of CORISCA's capabilities in a rudimentary version of the real-world GUI. However, functionality will only be available for Student, Scheduler, and Administrator users. This will allow the basic functions of the waitlist to be displayed, such as a student adding or dropping herself from a waitlist, or an administrator changing course information.

	Real World Product	Prototype
Environments for all Users:	Yes	No Will demonstrate student, admin, and scheduler users
Notification System	Yes	No Will be simulated with text box
Check for available seats	Yes	Yes
Add Student to Waitlist	Yes	Yes
Drop Student from Waitlist	Yes	Yes
Fair process	Yes	Yes
Alert System	Yes	No Will be simulated with text box
Mostly automated	Yes	No Will rely heavily on user interaction
Link to Banner	Yes	No Will be loaded with data.txt files instead
Link to Leo-Online	Yes	No Will be simulated with command box menu
GUI	Yes	Very Basic (Text System)
Seat Analysis System	Yes	No

Table 1. Comparison of CORSICA Real World Product to Prototype

4.1 Prototype Architecture (Hardware/Software)

The CORSICA prototype architecture will be virtually the same as the real-world product, consisting of a database, algorithms, GUI, and notification system. The database will be created with MySQL, and queries will be written using SQL. These queries will add information

to, delete information from, and search the database. Connection to the database will be established using both PHP and Javascript.

Multiple algorithms will be required for the prototype. They are as follows: Increase Course Capacity, Close Course, Notification, Add Student to Waitlist, Drop Student from Waitlist, Check for Open Seats, and Load Enrollment Data Files. These algorithms will be coded using C++, as will the notification system.

The GUI, although it will be a simplified version, will still be coded using HTML, CSS, and Javascript.

Algorithm	Functionality
Load Enrollment Data Files	Course data files are loaded into CORSICA Files contain: Course Capacity, Number of Enrollments, and Available seats
Open Course	An Administrator or Scheduler user logs into Banner and opens a course for students to enroll in Banner database is updated CORSICA database is notified of change and is updated
Check for Open Seats	Once a course becomes full, a waitlist is activated for it by CORSICA CORSICA will continually reference the current course capacity and amount of students enrolled If the amount of students enrolled is less than course capacity, a seat has become available CORSICA database updates Calls notification algorithm

Add Student to Waitlist	Student X wishes to enroll in Course Y's waitlist CORSICA receives this request and adds Students X to waitlist queue Course Y's waitlist is updated
Notification	The check for open seats algorithm completes and returns true for an available seat All students on the waitlist queue are notified of opening Students respond
Drop Student from Waitlist	Student X wishes to be dropped from Course Y's waitlist or the time window for that student has expired CORSICA receives this request and removes Student X from the waitlist queue Course Y's waitlist is updated
Increase Course Capacity	Administrator logs into Banner and increases course capacity for Course Y Banner database is updated CORSICA database is notified of change and is updated
Close Course	An Administrator or Scheduler user logs into Banner and closes a course as an available option for students to enroll in Banner database is updated CORSICA database is notified of change is updated

Table 2. Explanation of CORSICA's Algorithms

4.2 Prototype Features and Capabilities

The CORSICA prototype will first and foremost exhibit basic waitlist functionality. Students will be able to add or drop themselves from course waitlists, as well as view their positions on whatever waitlists they are currently on.

Another feature the CORSICA prototype will display is a simplified version of its notification system. Even though the prototype system will be simplified, however, it will still achieve the same goal as the real-world system; that is, when a registration spot opens up in a previously closed course, all students currently on the waitlist for that course will be notified. This notification will be carried out through email, text, or both.

A final feature the prototype will demonstrate is its automation. The majority of CORSICA will be automated, making for an efficient and easy to operate system. User actions will trigger the back-end algorithms, which will in turn power the automation process.

4.3 Prototype Developmental Challenges

One challenge the CORSICA prototype will face is syncing exactly with ODU's notification system. If the prototype's clock does not align exactly with ODU's clock, the notification will not be timed correctly. In addition, if the two clocks are not synced, a difference of a few seconds could be compounded and end up as a larger difference due to the various algorithms that are dependent on the time.

Another challenge will be guaranteeing that the algorithms work smoothly and correctly. Multiple algorithms power the back-end of CORSICA; not only must these algorithms function correctly on their own, they must also be successfully integrated with the GUI, notification system, and database. Thus, it is essential that all of the algorithms be properly tested and debugged in order to ensure proper and ideal performance.

A third challenge facing the CORSICA prototype will be that of security, specifically concerning the database. With banking and retail websites increasingly being hacked and having confidential data stolen, it is essential that the CORSICA database remain secure. The database will hold sensitive data concerning both students and faculty, so it will be necessary to develop proper security measure to protect this data.

A final developmental challenge will be ensuring that proper operating instructions are provided so that users know how CORSICA functions; if it is difficult to understand or run, people will be less likely to use it.

(This space intentionally left blank.)

GLOSSARY

Algorithm: A set of instructions that is carried out in order to solve a problem; typically used in the context of Computer Science.

Banner: The centralized academic and administrative records system used by Old Dominion University.

C++: An object-oriented computer programming language.

CSS: Cascading Style Sheets. A style sheet language used for formatting a document; typically used in conjunction with HTML for websites.

Computer: An electronic device that can store information and carry out programmed instructions.

CORSICA: Computer Science Class Waitlist. A piece of software developed for the Old Dominion University Computer Science department.

Database: A collection of (usually related) information.

Email: A method of sending messages from one computer to another on a network.

GUI: Graphical User Interface. A program that allows a user to interact with a computer by means of graphical icons such as buttons and text boxes.

HTML: Hypertext Markup Language. A markup language that is used for creating websites.

Internet: A worldwide system of interconnected computer networks.

JavaScript: A dynamic computer programming language, typically used in web browsers.

Lab: A section of a college class that commonly involves hands-on work.

Lecture: An oral presentation given by a college professor within a university setting.

MySQL: A relational database management system.

Notification: The act of informing someone of something.

ODU: Old Dominion University. A public post-secondary institution in Norfolk, Virginia.

PHP: A scripting language commonly used for web development.

Prototype: A preliminary model of something.

Recitation: A college class section where small groups of students from the corresponding lecture meet to review weekly material.

Server: A computer that provides information and services to other computers.

SQL: Structured Query Language. A programming language designed for managing data held in a relational database management system.

Text Message: An electronic message that is sent between cell phones.

UIN: University Identification Number. A unique identification number provided to students at Old Dominion University.

Waitlist: A list of people who are waiting for something.

REFERENCES

National Center for Education Statistics, U.S. Department of Education. (2001). Postsecondary Education. In *Digest of* education statistics 2001 (chap. 3). Retrieved September 6, 2014, from http://nces.ed.gov/programs/digest/d12/ch_3.asp.

National Center for Education Statistics, U.S. Department of Education. (2001). Table 5: Number of education Institutions, by level and control of institution: Selected years, 1980-81 through 2010-11. In *Digest of* education statistics 2001 (chap. 3). Retrieved September 6, 2014, from http://nces.ed.gov/programs/digest/d12/tables/dt12_005.asp.