Requirements Specification

for

Mantis

Version 1.1

**Hakeem Madahey(5854526), Ismail Abumathkour (5921267), Ajay Patel (5660055), Maninder Singh (5738042), Xinan Wang (5535802),**

**Corey Wehr (4283123)**

COSC4F00 Mantis Analysis & Design

Tuesday, March 26, 2019

Table of Contents

Table of Contents ii

Revision History ii

1. Introduction 1

1.1 Purpose 1

1.2 Document Conventions 1

1.3 Intended Audience and Reading Suggestions 1

1.4 Project Scope 1

1.5 References 2

2. Overall Description 3

2.1 Product Perspective 3

2.2 Product Features 3

2.3 User Classes and Characteristics 3

2.4 Operating Environment 3

2.5 User Documentation 3

2.6 Assumptions and Dependencies 4

3. System Features 5

3.1 File Transfer and loading 5

3.2 Unzip file and modify tokenization 5

3.3 Tokenize all files……………………………………………………………………………………6

3.4 Checking the tokenized file………………………………………………………………………....9

3.5 Generate report based on checked results………………………………………………………….10

4. External Interface Requirements 12

4.1 User Interfaces 12

4.2 Hardware Interfaces 18

4.3 Software Interfaces 19

4.4 Communications Interfaces 20

5. Other Nonfunctional Requirements 21

5.1 Performance Requirements 21

5.2 Safety Requirements 21

5.3 Security Requirements 21

5.4 Software Quality Attributes 22

6. Other Requirements 23

Appendix A: Glossary 23

Appendix B: Analysis Models 24

Appendix C: Issues List 36

**Appendix D: Diagrams…………………………………………………………………………37**

**Appendix E: Tests of System Features**………………………………………………………..**43**

**Appendix F: Log History………………………………………………………………………45**

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
|  | March 24 | * Grammar/Structure update, as well as correcting comments * Tests run on the different system features in new appendix E * Updated system features * New appendix F for log history | 1.1 |
|  |  |  |  |

1. **Introduction**

**1.1 Purpose**

The purpose of this document is to describe and outline the details of the Mantis plagiarism detection system. The features of the Mantis system, how it works and what its purpose is will be explained, as well as it application and limitations. The interface Mantis will use, how it is used and who will be using it will also be explored. The overall architecture and how Mantis will interact with its environment and other external applications will also be made clear to the reader.

**1.2**     **Document Conventions**

Sections is represented as Times 14pt bold

Subsection is represented as Times 12pt bold

Text Times is represented as 12pt regular

“Figure X” and “Table X” is represented as Times 12pt bold; title is Times 12pt regular.

**1.3**     **Intended Audience and Reading Suggestions**

This document is meant for a variety of readers. Mainly, the document is intended for instructors and those who would want a further understanding of the Mantis system. An instructor using this system may want to know the underlying principles about it, and whether it will be suitable for them to use for their classes. A student or an aspiring developer may want to view this document to get an idea of how an SRS document is supposed to look like. It is possible that project managers or companies searching for a new plagiarism system will want to view this document.

**1.4**     **Project Scope**

The Mantis is a new take on detecting plagiarism within code. There are many systems out there, mainly MOSS that is used or detecting plagiarism within academia, which are effective but are not necessarily user friendly or transparent to students. The goal of this system is to allow instructors to set up their class, allow students to submit their work, and allow the instructor and student to view possible plagiarism of their work on their web browser.

The Mantis system will be able to detect various levels of severity, which is up to the instructor or TA, for checking the amount of plagiarism and information copied. With code, at times you are not able to make absolute unique code, programmers use the same principles and at face value, two similar code samples are the same, but they are the standard method of applying an algorithm. Having severity levels allow certain levels of identical code, which is necessary in building a project. The system will allow the student (and instructor) to view, on their web browser, a side-by-side comparison of their code and the code that is marked as being copied with statistics to analyze.

The Mantis will have capability of detecting plagiarism between up to 3 different programming languages. Meaning, converting a java code to a C code will be considered plagiarism as well.

The Mantis detection system will allow transparent and effective plagiarism tools for instructors and students to view specific code that is marked as plagiarized in a manner that was not available before.

**1.5**     **References**

(2010, July). Retrieved February 4, 2019, from <https://theory.stanford.edu/~aiken/moss/>

P., Clough. (2000). Plagiarism in Natural and Programming Languages: An Overview of Current Tools and Technologies.

J., Hage. (2010, June). A comparison of plagiarism detection tools. Retrieved February 4, 2019,from <https://pdfs.semanticscholar.org/81de/497e2a2a58dfb1bd4ee9dced5f0c0e009e52.pdf>

Arwin, C. (2006). Plagiarism detection across programming languages.

Bakker, T. (n.d.). Plagiarism Detection in Source Code.

Bowyer, K. (n.d.). Experience Using ”MOSS” to Detect Cheating On Programming Assignments. Retrieved from <https://www3.nd.edu/~kwb/nsf-ufe/1110.pdf>

M., Mozgovy. (n.d.). Fast Plagiarism Detection System. Retrieved February 4, 2019, from<https://pdfs.semanticscholar.org/6454/885b2e0c103b6dc1febe102a7fe7466e0b48.pdf>

Sidorov, G. (2013). Syntactic Dependency Based N-grams in Rule Based Automatic English as Second Language Grammar Correction.

S., Mann. (2006). Similarity and Originality in Code: Plagiarism and Normal Variation in Student Assignments.

S., Schliemer. (2003). Winnowing: Local Algorithms for Document Fingerprinting.

Writing a Software Requirements Specification Document. (2016, November 09). Retrieved from <https://www.onedesk.com/writing-a-software-requirements-specification-document/>

# 2. Overall Description

## 2.1 Product Perspective

Mantis is a self-contained product to detect software structure similarity. Mantis runs part on the university server and part on its own servers. Mantis provides the university with a program that will act as the local server. Mantis will run on a sub-domain of the university (Mantis.University.com). Student will use this website to upload their projects and all the files will be saved and converted on the local server. Once, all projects have been submitted and with an authorization of the professor the converted files will be sent to Mantis for processing. This is done by converting all projects submitted to an intermediate language then the intermediate language is compared to each other on Mantis servers. Mantis servers will only receive converted project and will only save converted projects in the intermediate language format. This will ensure the plagiarism is detected across multiple languages. Additionally, professors/instructors will be able to check for plagiarism across projects submitted and projects stored on the database in multiple languages.

## 2.2 Product Features

Mantis will accept files submitted using the UI on the university website by a professor/instructor. Then will remove all unnecessary comments and information in a program, then convert all the files to an intermediate “language”, which is more of a tokenized text file. The intermediate language is a form of tokenizing all the language reserved words, input and output. This will ensure that the detection of patterns within the structures among the variables are found.

## 2.3 User Classes and Characteristics

Mantis will be available to all professors and faculty. Students are not allowed to purchase a copy of the software, this is done so the students are not able to submit their assignment multiple times to try to fool the algorithm. Confirmation that the buyer is a professor/instructor will be done through the university/college.

## 2.4 Operating Environment

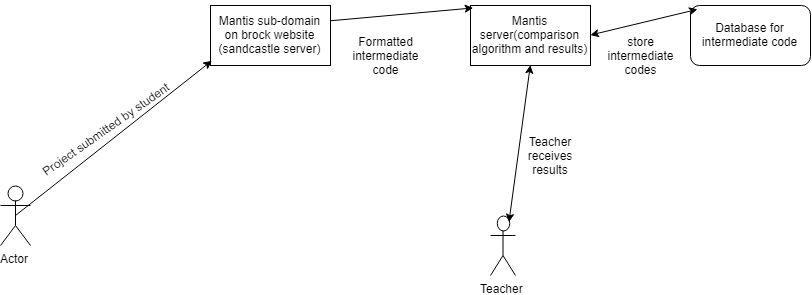
Any computer with internet access will be able to use the site and purchase Mantis. The user will be able to upload the files through any browser or FTP after the user authentication. Using HTML, the browser will accept zip(compressed) files and this will be verified on the server before processing. In order to ensure no incorrect files being uploaded, like images, or videos and other files that are not necessary.

## 2.5 User Documentation

Mantis site will have a tutorial for students, professor and faculty. This tutorial will guide them through every process and how to submit their projects and review results.

## 2.6 Assumptions and Dependencies

Student must have a zip program to compress the folders into one file. This is a must since uploading one file at a time would be cumbersome. The student must have the latest browser to upload the files. The university/college must have a local server to host the site and run the first part of the program. The university/college must be able to use FTP to connect with Mantis’s servers.



Student

Instructor

**Figure 1:** Overall view of Mantis the service

**3. System Features**

**3.1 File Transfer and loading**

**3.1.1    Description and Priority**

Using the Mantis subdomain of the brock website, users (students) can upload to website, in the form of a zip file. This is a lower priority because there is no real complexity here.

**3.1.2    Response sequence**

None, this is the first step to load a file.

**3.1.3    Functional Requirements**

There will be a window or UI control for the user to let them upload their files into the system, Files will be uploaded if the user is ready to select and submit.

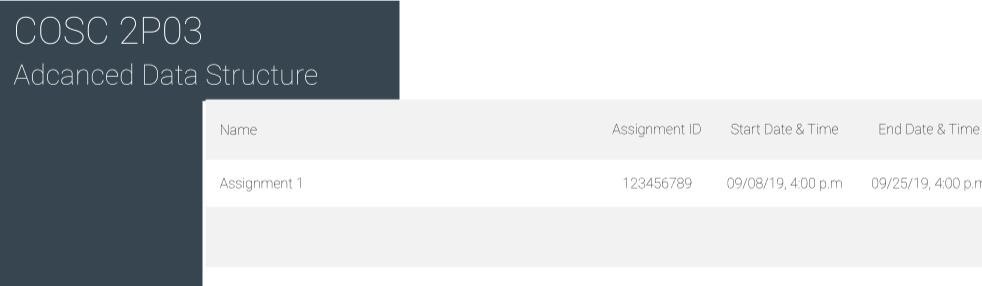
REQ-1: Indicate file path

REQ-2: Able to upload file (zip file)

**3.2 Unzip file and modify to tokenizable file**

**3.2.1    Description and Priority**

The Mantis subdomain will unzip the file that the student uploads, and will detect the coding source files (Java, C, C++). The packages and files within the student upload will be merged into one. Comments and identifiers are removed at this stage. This merged file, will be linked to the student ID of the student. At this point the Mantis’ provided program to the school converts the file to an intermediate tokenizable string we call a .lang file. All these files of

**Figure 2:** Student assignment page

assignments will be zipped and sent to the Mantis server. This is a medium level priority, because specific formats must be achieved by using the specific mantis program before sending it to the mantis server for comparison.

**3.2.2   Response sequence**

Need Feature 1 to upload file or zip file.

**3.2.3 Functional Requirements**

It will detect the code file (Java, C, C++) and disallow other types of files (pictures, music, anything other than the essential coding source files), then find out the valid assignment id as a marker after converting it to a tokenizable file.

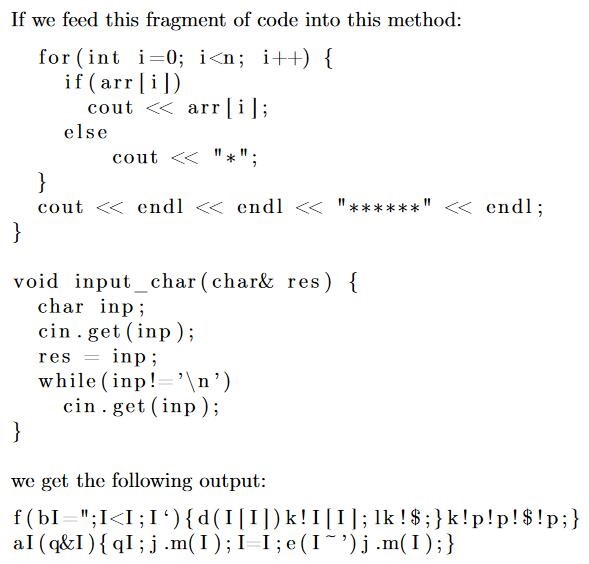
REQ-1: Define code file (create tokenizable file).

REQ-2: Obtain valid assignment id for matching file.

**3.3 Tokenization of the files**

**3.3.1    Description and Priority**

The intermediate tokenizable file is then sent to the Mantis server, where it will be converted into tokens for comparison. The intermediate will be comparable between the three programming languages and the tokenized strings that are created are used to draw conclusions regarding what was copied. There is high priority here, the specific methods for tokenization will determine how effective the overall system will work.



**Figure 3:** Tokenization process.

**3.3.2 Response sequence**

Need Feature 1 to upload file or zip file.

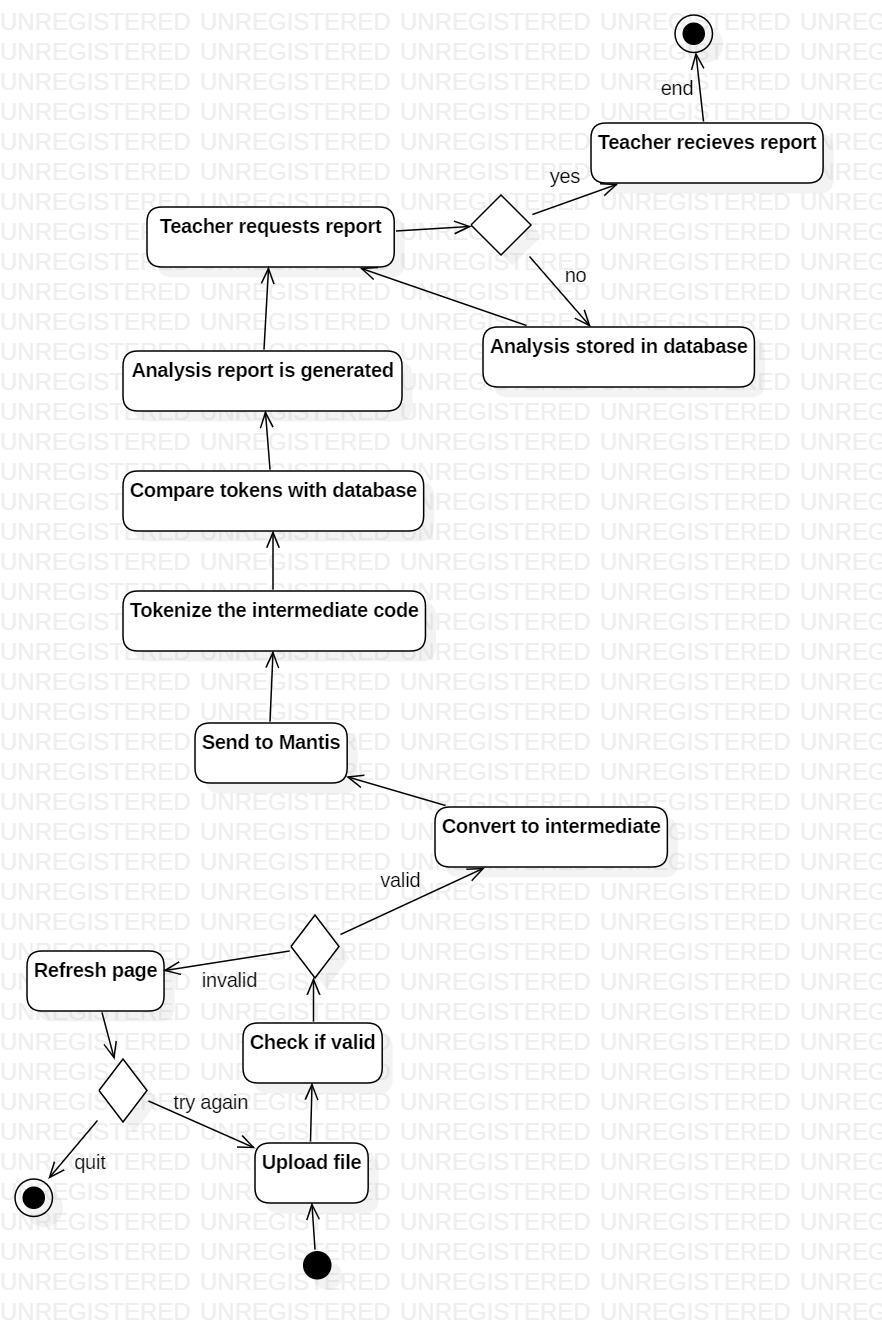
**3.3.3 Functional Requirements**

Tokenization of the strings creates the tokenized intermediate file to compare tokens with each of the other assignments, and when the system runs, the intermediate file requires an id as a marker to track, to compare with other intermediates and to keep metadata. Tokenization will be followed by translation of the intermediate code file into strings, and convert these strings as some type, the type is called “token”. The tokens will be variables/methods, anything that can be indicated and possibly replicated.

REQ-1: Enough storage for tokenized file, we need to have the correct storage within our database, so we can receive and send the assignments, so we are able to scale up in the case of additional assignments, courses and other usage of the system.

REQ-2: Need valid assignment id for matching intermediate files, we will require the assignment id and other identifier in the text files title, so the system can link it back after the algorithm is run, because identifiers will be broken within the file.

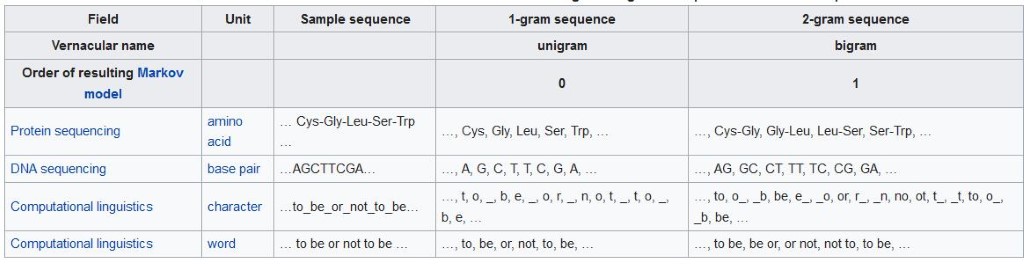
REQ-3: Require the tokenizable .lang formatted text file in order to tokenize

**Figure 4:** Activity diagram in respect to uploaded file.

**3.4 Checking the tokenized file**

**3.4.1    Description and Priority**

The tokenized intermediate file will then compare to other intermediates on the Mantis server. When similar tokens are found they will be highlighted and marked. The system will be using the N-grams method to detect and separate the string and to find common segments within the code intermediate. We have tested for 2-3 grams, it seems as though 3 grams is optimal to achieve accurate comparisons percentages. This will be costly because there will be many intermediate codes that will be analyzed and compared to, due to the volume of possible assignments that are needed to be checked.

Priority will be high, based on database that accumulated and with time cost it is also increasing, as well as the fact that this is where the actual detection occurs.

**Table 1:** N-gram examples; 1-gram sequence, 2-gram sequence.

**3.4.2 Response sequence**

Requires the interface on the website to send the files appropriately to allow this algorithm have access to the original files in the form of tokens.

**3.4.3 Functional Requirements**

The system must have the specific .lang file to be sent, to be able compare and contrast with the other intermediate tokenized code that is within the realm of the assignment.

REQ-1: Perform N-gram string comparison

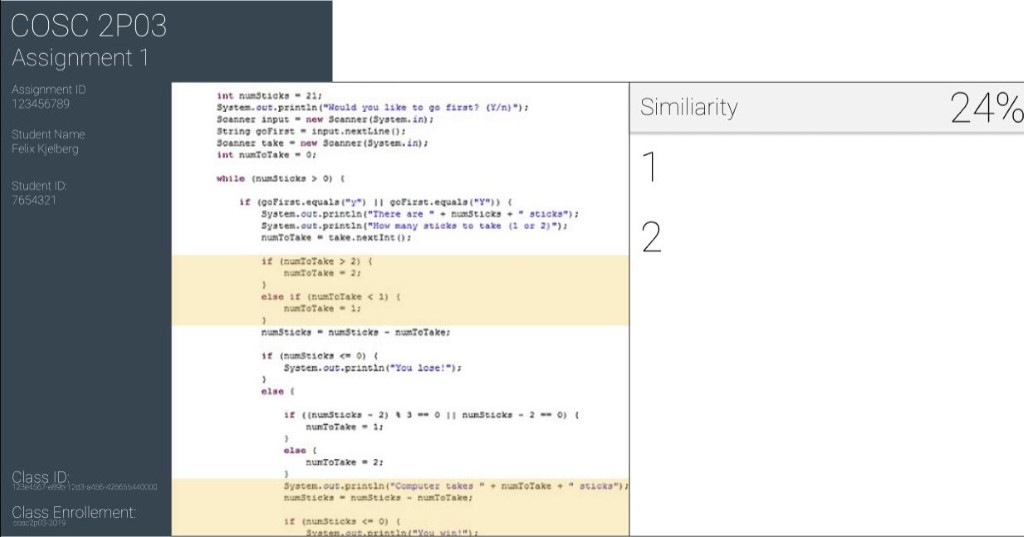
REQ-2: Able to pass highlight information for later viewing.

REQ-3: Able to save total compare time cost

**3.5 Generate file based on checking result**

**3.5.1    Description and Priority**

The mantis server then generates a browser viewable pdf file to view matched code to each student, and their percentage for the clone, this pdf will also be able to highlight the cloned segment. There will also be a total percentage with all students and their student number. This pdf will be the only output for this system and is sent back to the teacher. If the user sets a percentage as a number, it will show which student is higher than the clone percentage. If the user is admin(teacher), they can search single student by their name or student number to show.



**Figure 6:** Similarity report.

The statement of this student. When the result is finished it will send to database to store and send a signal back to structure interface to let the user know job is finished.

Priority will be high because the validation is based on the way the user (TA) chooses for the severity. Also, some common code is calculated inside, for example “get package” or given same assignment value's name. Therefore, the priority is essential and sensitive in this part because there will be similar code that will be impossible to alter.

**3.5.2 Response sequence**

To generate a pdf file, it needs highlighted segments and original code from feature 3.

**3.5.3    Functional Requirements**

Based on the highlighted segment from feature 3, the Mantis generates the output file that will be able to recognize the highlighted section and translate that into output, then this output file will be able to be interacted with by the user to show the clone section, be able to search by student name or number, and be able to distinguish the percentage rate.

REQ-1: Output package will be able to read and handle the pdf file.

REQ-2: Validify result from feature 3 and be recognizable.

# External Interface Requirements

## User Interfaces

**4.1.1** **Student/Instructor User Interface Overview**

Mantis’ user interface comprises a collection of active/inactive courses, and user profile.



2

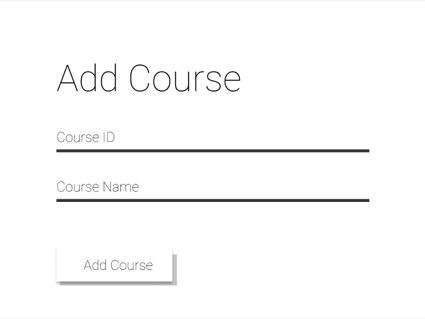
1

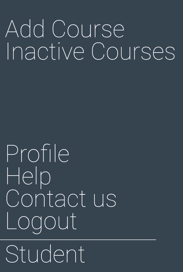
|  |  |  |
| --- | --- | --- |
| No. | Name | Description |
| 1 | Active/Inactive course card | An active course card allows you to access the ability to view all your upcoming assignments. The inactive course card looks similar to that of the active course card however you are unable to access the assignment list. |
| 2 | User profile | The user profile (drop down menu) you to add a course to your collection. It allows you to cycle through the active and inactive courses, that you have registered to. It also allows the user to logout once completing desired task. |

## 

**Figure 7:** Course portal as viewed by student.

**Table 2:** Interactive components of the course portal.



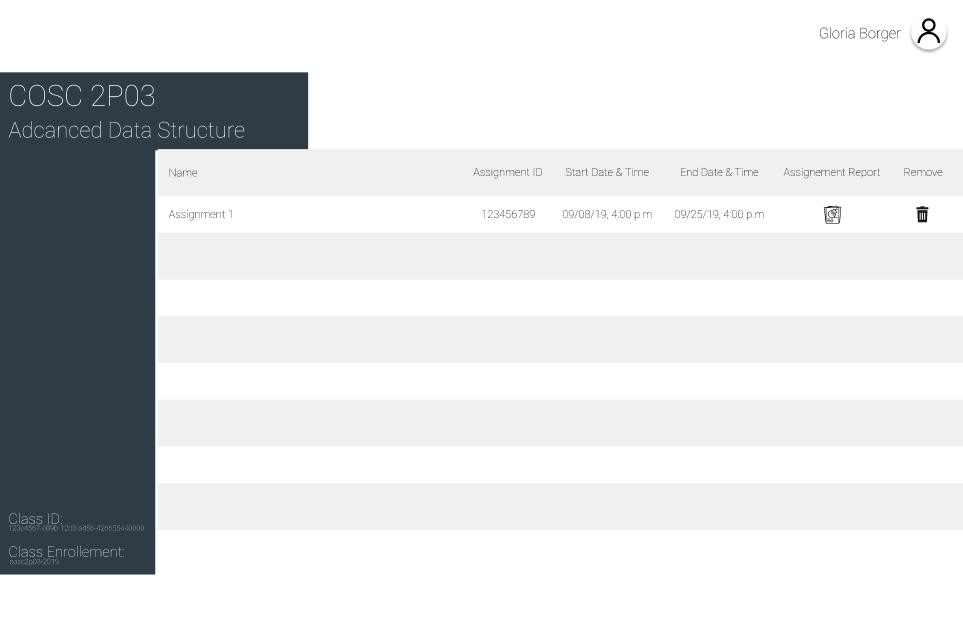


**Figure 8: (a) Profile drop down menu (b) add course to course portal.**

* The instructor has the ability to add course. The course produces a class ID and a class enrollment key.

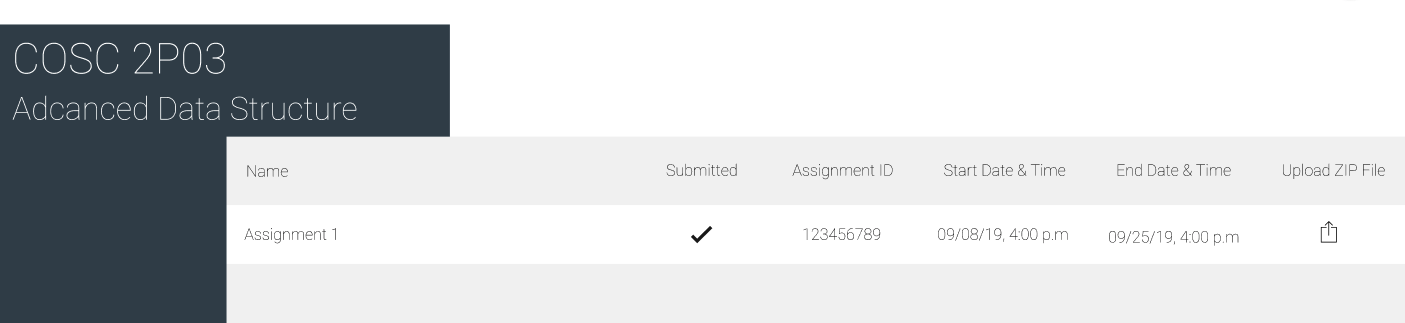
**4.1.2 Student/Instructor Course page**

* Class ID and class enrollment key is displayed in the bottom left.
* The list of active assignments is displayed to the screen page. Each assignment is given a unique assignment ID.
* The assignment start/end & time is displayed to screen. This time & date is set by your instructor.



**Figure 9:** Course page (as seen by instructor)

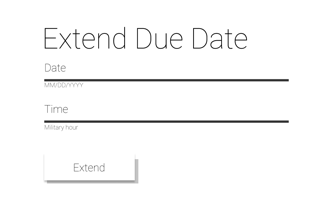
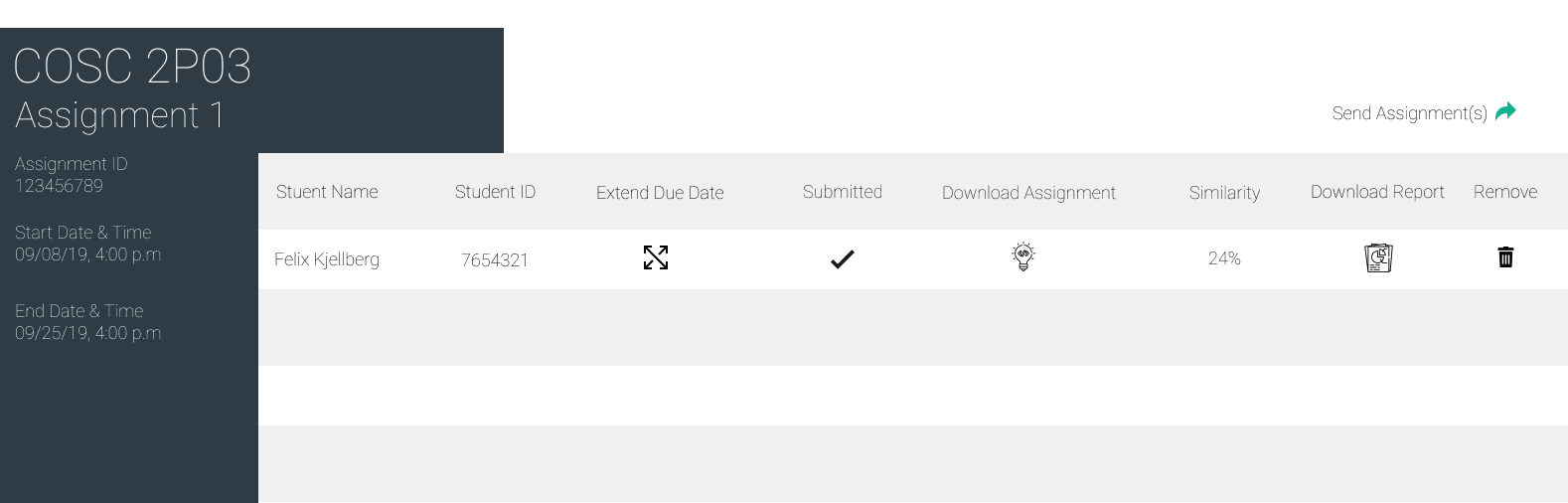
* The professor has the ability to remove the assignment form the course page. This will then propagate through the system and remove all instances of assignment 1 for all students registered to the course.
* The TA course page is similar to that of the instructor page (above), however the TA is unable to delete the assignment form the course page.
* The assignment report button (located next to the remove button) allows the instructor to download the report generated after the Mantis algorithm is run, it is otherwise inactive.
* The student course page looks quite similar to that of the instructor course page. The student is able to upload the assignment as a zip file, as seen in figure 9.



**Figure 10: Course page (as seen by student)**

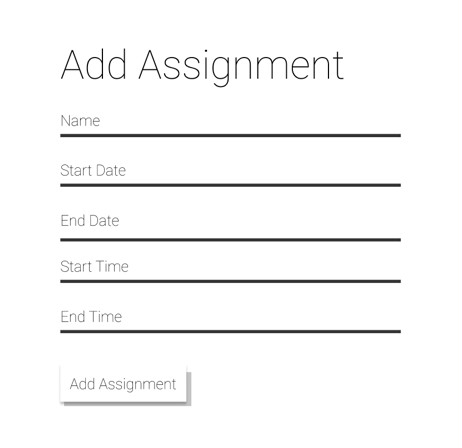
* Once the assignment is submitted the system populates the submitted column with an indication verifying that fact that it has been submitted (check mark).

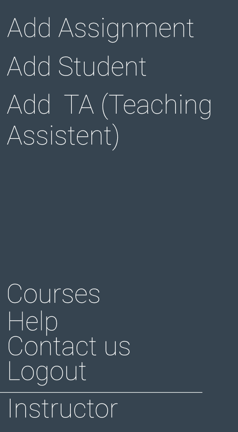
**4.1.3 Instructor Assignment page**

* The instructor can see all those that have registered to the course. Each student is given a unique student id (this is primarily your school student id).
* The Instructor has the ability to see if an individual student has submitted the assignment, this is represented with a check mark.
* The instructor has the ability to extend the due date for the individual student for circumstantial reasons. The system will give the instructor the ability to explicitly add a new end date & time.

**Figure 11: Assignment page (as seen by TA/instructor)**

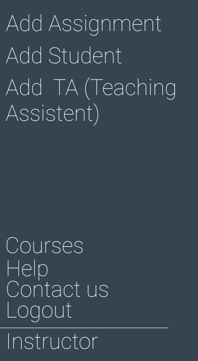
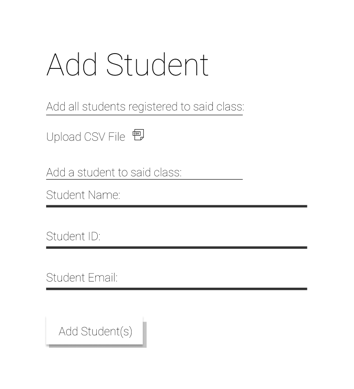
* The instructor has the ability to download the assignment to your computer.
* The instructor is able to add a new active assignment via instructor profile.

**



**Figure 12:** Instructor drop down menu; add assignment.

* The instructor has the ability to send the entire collection of assignments to the Mantis server in order to check for assignment similarity. The assignments can only be sent to the server once the assignment end date & time has passed.
* The instructor is able to remove student submission from the assignment page using the remove button.
* The assignment similarity algorithm returns a similarity percentage. The percentage is displayed to the similarity column. This is only seen by the instructor and TA (teaching assistant).
* The instructor is able to add a TA via instructor profile.
* The instructor is also able to add a student via instructor profile.





**Figure 13:** Instructor drop down menu; add TA/add student.

* The ability to add a student/TA can be done individually or via csv file. The csv file is supposed to be if more than one student is to be added to the system.
* The TA is unable to remove student from the assignments page. This must only be done by the instructor.
* TAs are also not able to add students/other TAs to the system. This must only be done by the instructor

**4.1.4 Similarity Display (Student Report Page)**

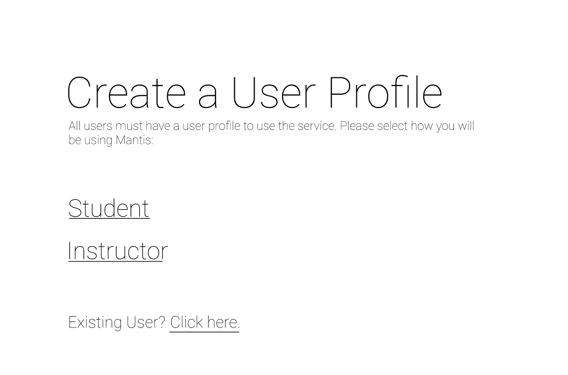
* The instructor and TA will be able to see the similarities percentage (this is presented in the top right-hand corner). This reflects the amount of code that may be plagiarized (further investigation may be required).
* The Student Report page has two parts; the left-hand side display, displays the students submitted assignment. The plagiarized code is reflected by yellow highlight.
* The yellow highlighted code is then matched to the individual (right-hand side display) most likely where the code was copied from.



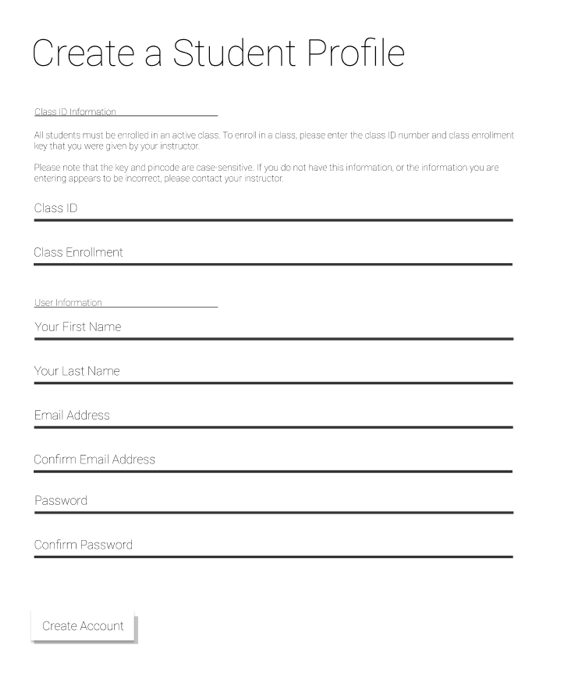
**Figure 14: Similarity report (only seen by TA/instructor)**

**4.1.5 Create Profile**

* Only students and instructors are able to create accounts through our system. TA accounts must be created by the instructor.
* The instructor is required to provide a join key while creating an instructor account. The join key is provided to the instructor by Mantis.
* The student/instructor is asked to provide the university name (exclusive to instructor) at which they attend. Their first and last name, email address and password.

****

**Figure 15:** Create user profile.

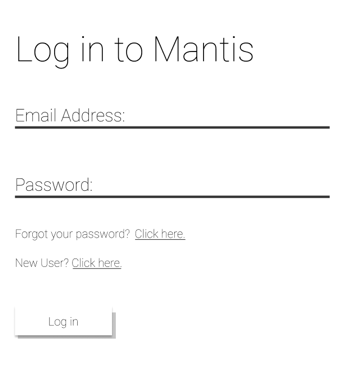


**Figure 16:** Create instructor profile; Create student profile.

* The student is required to provide a Class ID and Class Enrollment Key when creating a student account. The Class ID and Class Enrollment Key must be provided by the instructor.

**4.1.6 Login Portal**

* Student/Instructor/TA is asked to provide email and password.
* If user forgets mantis password a password change email is used to change user password.



## 

## Figure 17: Log in to Mantis

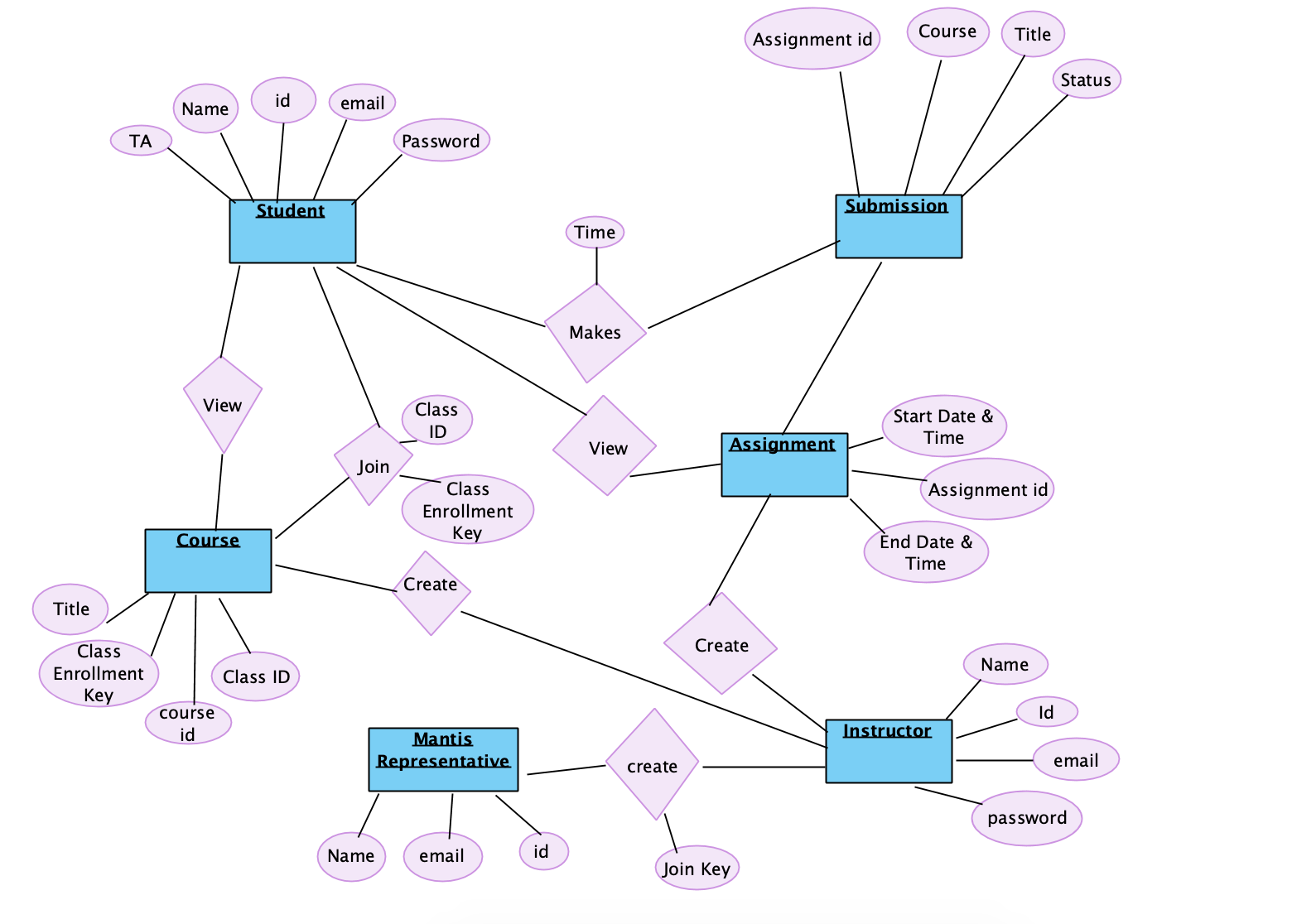
## Hardware Interfaces

* This application is supported on all devices connected to the internet. The Mantis service is presented in HTML form (website).
* The nature of data download is packaged in pdf format.
  + Data download referring to the similarity report for each individual student.
  + An Assignment similarity report produces a list of all students and its associated similarity score (in percentage).
* Uploading files is made available on the website, and can be zipped files, and specifically programming source code only.

## Software Interfaces

* User (Student/Instructor/TA) information is stored in a database.
  + Passwords are hashed using bcrypt (password hashing function)
  + Bcrypt version 3.0.3.
  + Bcrypt module uses node-gyp to build and install. Stable version of Node is required; Node version 10.
* Students are instructed to provide assignments in ZIP file.
  + Our system will check both the file extension and mime type to ensure it is the desired file.

**Figure 18:** ER diagram; describing the key entities and attributes of the system.

* Mantis uses a relational database SQL, as seen in figure 17.
* The most common library used for our frontend design:
  + React JavaScript Library.
  + Materialize CSS framework version 1.0.0.
    - The latest NPM is required to install Materialize.
* Mantis provides similarity reports; assignment and individual reports
  + We use a PHP library TCPDF.
* Mantis will send confirmation emails upon creating account, and when the Mantis algorithm has completed its detection process (this is sent to the instructor and TA).
  + We will be using Action Mailer – Ruby on Rails API

## Communications Interfaces

* This service requires internet access/permission.
* Web browser permission
* User login authentication requires authentication mechanism SCRAM.
* User (Student/Instructor/TA) is in direct contact with our website server (a subdomain of the university).
  + This is where user information, course creation, assignment upload and report download occurs.
  + The Mantis algorithm used for plagiarism detection is on a completely different server (student users do not have access to the Mantis server).
  + The assignments are sent by the instructor to the Mantis server in order to run the algorithm.
* Client is in direct contact to the Mantis service which is a subdomain of the university.
* The student is required to upload assignments via ZIP file. In order to upload files to our website server we must use a standard ftp protocol.
* The files are cached after the user has uploaded them to our website server. Using the ‘Send Assignment(s)’ button on the top right-hand corner (assignments page) the instructor is able to send multiple assignments to the Mantis server. This is done by creating an XMLHttpRequest (ajax) which opens a connection to the server. The multiple files are then sent to the server in parallel stream.

1. **Other Nonfunctional Requirements**
   1. **Performance Requirements**

* This system is based on the Java programming language and elements of HTML, CSS, SQL and will analyze code from Java, C, C++.
* The online system must have a Server and Client
* The online system must have a Database to keep a repository of previous documents
* The documents submitted must not exceed 100mb, due to possible malicious uploading of other files, typically a document should not exceed 100mb in our system for precaution
* The analysis will take a few minutes and the results will appear on the screen for the user
* The number of times an assignment can be submitted must be indicated by the Professor/Instructor
* Assignments will not be accepted after a date outlined by the Professor/Instructor
* The Professor will have the ability to extend the accepted date for either the individual student or the entire class.
  1. **Safety Requirements**
* To uphold intellectual property rights the Server (which holds the main system) will be located locally at Brock University
* To ensure user protection and confidentiality when submitted, the assignments are stripped
* Only when the submission is returned with result may the information of the user be presented
* The user must agree to allow their submission to be placed with others in a repository located at your local university to safeguard against future plagiarism.
* The user will have the choice to accept these terms when submitting the assignment

* 1. **Security Requirements**
* All that use this system must go through an authentication process, this is required for both logging in and signing up
* The Professor/Instructor will be given a unique Join Key to create a course page, which will be provided by our Mantis representative (some 24/7 tech support assistant that will allow teachers to allocate their classes on our system)
* The Professor/Instructor will be given a Class ID and Class Enrollment Key(randomly generated, to be distributed to students for access to course) which must send to those students who are enrolled in the course and have signed up
* Anonymity must be granted to the student who is submitting the work
* The student’s identification can only be accessed by the instructors and professors

* 1. **Software Quality Attributes**
* The system is completely online based and can be accessed by any device that can connect to the internet
* Professor/Instructor will be given a unique Join Key to register and create a course page
* Admin accessibility will be granted to the professor or instructor of the course and may use the services for submission purposes.

**Appendix A: Glossary**

**Assignment page:**The assignment page gives user assignment information such as; start time & date, end time & date.

**Client:**User workstation able to retrieve information from a server (in a network).

**Class ID:**The class ID distinguishes one class instance from another.

**Class Enrollment Key:**The enrollment key is distributed to the class by the instructor. The key is used to register to the course Mantis service. This is the only way to access the assignment page.

**Course portal:**The course portal displays active courses (and inactive courses)

**Database:**Organized collection of data stored on a computer system.

**Join Key:**The join key is given to the instructor by a Mantis representative. The key allows the instructor to create an account.

**Mantis representative:** One of our many dedicated tech support assistants that assist the professors the professors that utilize our site. Will dictate class/course setup with the professors.

**N-Grams:**is a contiguous sequence of *n* items, used in detecting similar sequences.

**Project Mantis:**Is the plagiarism detection service. The project name pays homage to the Mantis Shrimp.

**Plagiarism:**Is the act of using other people's ideas and or words without crediting the source.

**Repository:**A collection of all student assignments will be locally saved to your system.

**Server:**A centralized resource of data in a network.

**Token:**tokenization is the act of substituting sensitive data with an equivalent token. These tokens don’t mean anything if someone was to retrieve this information.

**ZIP:**Archive file format that supports lossless data compression. Students are required to submit assignment projects using the ZIP format.

**Appendix B:**

**User Story**

* **U1**: As a user I want to be able to log in.
* **U2**: As a student I want to be able to upload my completed assignment.
* **U3**: As a student I want to be able to register my account.
* **U4**: As an instructor I want to be able to register my account.
* **U5**: As an instructor I want to be able to manage all files in the database.
* **U6**: As an instructor I want to be able to generate a plagiarism check report (pdf) on any group of programs I chose.
* **U7**: As an instructor I want to be able to add all students in my class so they can gain access to the system.
* **U8**: As an instructor I want to be able to see and manage a list of all my student and TA accounts that I have added.
* **U9**: As an instructor I want to be able to add assignments and manage their deadlines.
* **U10**: As an instructor I want to be able to add courses and manage their status.
* **U11**: As a TA I want to be able to download all the files in my course.

**Use Cases**

**U1: User Logs In**

*Actors:*

* Student/TA/Instructor User (Has an existing account)
* Locally run program to upload files
* Account Storing Database

*Triggers:*

* The user opens the program for the first time this session.

*Preconditions:*

* User has installed the program and clicks on it.

*Post-conditions:*

* The user will have access to the functions of the program.

*Normal Flow:*

1. The user opens the program.
2. The program displays the log in screen.
3. The user enters the email and password.
4. The program sends this to the database.
5. The database checks if this is valid account info.
6. The database sends confirmation back to the program.
7. The program confirms the log in and opens the user profile page.

*Alternate Flows:*

5a: If the information does not match any of the accounts, the database will send an error to the program.

5b: The program will display an error and gives the user a chance to change log in info and try again.

6a: If the database never returns confirmation, the program will display an ‘unable to connect to server’ error.

**U2: Student Uploads**

*Actors:*

* Student User (Has an existing account)
* Locally run program to upload files
* File Storing Database

*Triggers:*

* The user indicates that he/she wants to upload a file they have selected.

*Preconditions:*

* User has selected the file to be uploaded.

*Post-conditions:*

* The file and metadata on who uploaded it will be recorded on the server.
* The user will have a visual confirmation the file was successfully uploaded.
* The user will be emailed confirmation the file was uploaded.

*Normal Flow:*

1. The user will open the program and log in.
2. The program will request a list of courses from the database.
3. The database will send the list of courses.
4. The program will display the list of courses.
5. The user will select a course.
6. The program will request a list of assignments from the database.
7. The database will send the list of assignments.
8. The program will display the list of assignments.
9. The user will select the assignment number.
10. The program will open an interactive file directory of the computer.
11. The user will browse to the location of the program file and select it.
12. The system will ask the user to confirm the file is correct and ask for the password.
13. The user will confirm that it is correct.
14. The program will upload the file to database.
15. The database will record the file, and metadata on who uploaded it and when, and which assignment it is for.
16. The database will send a confirmation message to the student’s email.
17. The database will send confirmation back to the program that the file was received.
18. The program will display to the user confirmation that the file was received.
19. The user will exit the program.

*Alternate Flows:*

11a: The user selects multiple files or an entire folder to upload

11b: The files are automatically put into a zip file before uploading

15a: The database sees that the user has already uploaded a file for this assignment

       15b: The database will override the old file with the new one.

16a: The program never receives confirmation from the database.

16b: The program will display an error and give the user the option of selecting a new file or   trying again.

**U3: Student Registers**

*Actors:*

* Student User
* Locally run program to register
* Account Storing Database

*Triggers:*

* The user indicates that he/she wants to register.

*Preconditions:*

* Instructor has added this student to a course.
* Student has a class ID and enrollment key from the instructor.

*Post-conditions:*

* The database will contain an active account for this student.

*Normal Flow:*

1. The user will open the program.
2. The user will select to create a student profile.
3. The program will display the create a student profile screen.
4. The user will enter the class ID, class enrollment, name, email, and password.
5. The program will check that duplicate fields for email and password match.
6. The program will send the info to the database.
7. The database will check that the info is correct, and then create the student account.
8. The database will send conformation back to the program.
9. The program will confirm that the account has been created and open the log in screen.

*Alternate Flows:*

5a: The duplicate fields do not match.

5b: The program will display an error that they do not match and not send the data until fixed.

7a: The database does not contain a record for this course, or the professor has not added this student to the course.

7b: The database will not save the account and will send back an error.

7c: The program will display this error to the student.

7d: The user will be given a chance to change the registration information and try again.

8a: If the program never receives confirmation back from the database, it will display a connection error and give the user the option to send the info again.

**U4: Instructor Registers**

*Actors:*

* Instructor User
* Locally run program to register
* Account Storing Database
* Mantis server

*Triggers:*

* The user indicates that he/she wants to register.

*Preconditions:*

* Instructor has gotten a key from Mantis

*Post-conditions:*

* The database will contain an active account for this instructor.

*Normal Flow:*

1. The user will open the program.
2. The user will select to create an instructor profile.
3. The program will display the create an instructor profile screen.
4. The user will enter the key, university, name, email, and password.
5. The program will check that duplicate fields for email and password match.
6. The program will send the info to the database.
7. The database will send this info to the Mantis server.
8. The Mantis server will check that the key is correct.
9. The Mantis server will send conformation back to the database.
10. The database will save the professor account.
11. The database will send conformation back to the program.
12. The program will confirm that the account has been created and open the log in screen.

*Alternate Flows:*

5a: The duplicate fields do not match.

5b: The program will display an error that they do not match and will not send the data until fixed.

8a: The Mantis server finds that the key is not correct.

8b: The Mantis server will send an error to the database.

8c: The database will send an error to the program to display.

8d: User will be given a chance to change the registration information and try again.

11a: If the database never gets confirmation back from the Mantis server, it will try sending again.

11b: If it still does not get confirmation back after a few tries, it will send an error to the     program.

11c: The program will display a connection to server error and give the user the option to send the info again.

**U5: Instructor Looks at Files**

*Actors:*

* Instructor User (Has an existing account)
* Locally run program to look at files
* File Storing database

*Triggers:*

* The user indicates that he/she wants to see a list of all the files on the server.

*Preconditions:*

* User has logged in.

*Post-conditions:*

* The user will have an interactive list of all the files that were uploaded.

*Normal Flow:*

1. The user will open the program and log in.
2. The user will navigate to a page to look at all the files on the database.
3. The program will send a request to the database for a list of all files it has.
4. The server will send this data back to the program.
5. The program will display an interactive list of all the files and the metadata on who uploaded it, when it was uploaded, what course and assignment it was for, and if it was late.
6. The user may select different options to change how the list is sorted or displayed.
7. The program will correctly respond to the sort and display options.
8. The user may select a file he/she wants to download and look at.
9. The program will send a request to the server for that file.
10. The server will send that file.
11. The program will save the file to the computer and open that folder.

*Alternate Flows:*

4a: If the server does not send any data back, or the current list of files is empty, then the program will not display any files.

5a: The program will display an option to upload more files to the database.

5b: The user choses this option.

5c: The program uploads an interactive file directory for this computer.

5d: The user navigates to the files and selects them.

5e: The program zips these files and sends them to the database.

5f: The database will save these files not attached to any course, student, or assignment.

5g: The database will send confirmation back to the program that the files have been saved.

5h: The program will update the displayed file list to include these new files.

8a: The user will select multiple files to download at once.

8b: The program will correctly save all the files to the folder.

10a: If the program never receives the file from the server it will display a connection to database error message.

**U6: Instructor Checks for Plagiarism**

*Actors:*

* Instructor User (Has an existing account)
* Locally run program to look at files
* File Storing database
* Plagiarism Checking Server

*Triggers:*

* The user indicates that he/she wants to check some files for Plagiarism.

*Preconditions:*

* User has logged in.

*Post-conditions:*

* The user will have a final report of what the plagiarism algorithm found.

*Normal Flow:*

1. The user will open the program and log in.
2. The user will navigate to the page that shows all the files for a chosen assignment.
3. The program will request a list of files for this assignment from the database.
4. The database will send the list of files to the program.
5. The program will display the list of files and their metadata.
6. The program will give the user the option to generate a plagiarism checking report for all files listed under this assignment.
7. The user will click the option to get a plagiarism report for these files.
8. The program will tell the database to generate a report for those files.
9. The program will display a ‘working, please wait’ message until the report is received.
10. The database will send all saved files of code it has to the plagiarism checking server, marking which ones are the ones to make a report on.
11. The plagiarism checking server will send confirmation to the database that it has received these files.
12. The plagiarism checking server will run the Mantis algorithm on the report files, comparing them to all other files.
13. The plagiarism checking server will generate a report in PDF format.
14. The plagiarism checking server will delete all the files it received.
15. The plagiarism checking server will send this report to the database.
16. The database will email the user that the report is done.
17. The database will tell the locally run program that the report is done.
18. The program will update the assignment page with the option to download the report.
19. The user will select this option.
20. The program will request the report from the database.
21. The database will send the report.
22. The program will save the file to the local computer and open the report in the default PDF viewer.

*Alternate Flows:*

4a: If the program never receives the list of files, or the list is currently empty, then it will not display any files.

7a: Instead of choosing to generate a report on all files under this assignment, the user may manually select any number of these files and request a report for only those files.

     7b: The program will correctly tell the database which files need a report.

     7c: The database will correctly tell the server which files need a report.

11a: If the database does not receive confirmation from the server, the database will try sending the files again.

     11b: If the retry fails multiple times the database will send an error to the program.

     11c: The program will display this error and give the user the option to try again.

**U7: Instructor Adds Student Accounts**

*Actors:*

* Instructor User (Has an existing account, possibly with an ID number)
* Locally run program.
* File Storing Database

*Triggers:*

* The user indicates that he/she wants to add some new student accounts.

*Preconditions:*

* User has logged in.
* User has generated a CSV file of students in some other program.
* User has navigated to the page for the course to add students to.

*Post-conditions:*

* The user will receive confirmation that the student accounts have been activated.

*Normal Flow:*

1. The user will open the program and log in.
2. The program will give the user the option to add student accounts from the course page.
3. The user will confirm that they want to add student accounts from a CSV file.
4. The program will open a menu with the file directory of the computer.
5. The user will navigate to the correct folder and select the CSV file
6. The program will check that this is a CSV file in the correct format, and then ask the user for confirmation
7. The user will confirm that they want to do this.
8. The program will contact the database and send it the data.
9. The database will add the student accounts to its internal database, and record that they are under this instructor user.
10. The program will send an email to each student with a link to the registration page.
11. The database server will send confirmation back to the program.
12. The program will show confirmation to the instructor user that the student accounts have been added and emailed.

*Alternate Flows:*

2a: The user may select to add TA account(s) to this course instead of student account(s).

2b: The rest of the process is the same except it will result in TA accounts being created.

3a: The user will select that they want to add an individual student account instead of using a csv file.

3b: The program will show a menu to enter the student’s username, ID, and email

3c: The user will enter the information

3d: The program will send the information to the database.

6a: The check of the CSV file shows it is in the incorrect format or does not have all the required information for every account.

6b: The program will display an error to the user and ask to choose another CSV file.

11a: If the program never receives confirmation, it will display a connection error and give the option to try sending it again.

**U8: Instructor Manages Student Accounts**

*Actors:*

* Instructor User (Has an existing account, possibly with an ID number)
* Locally run program.
* Database

*Triggers:*

* The user indicates that he/she wants to manage student accounts

*Preconditions:*

* User has logged in.

*Post-conditions:*

* The user will finish looking at or managing the student accounts

*Normal Flow:*

1. The user will open the program and log in.
2. The program will give the user the option to manage student accounts.
3. The user will confirm that he or she wants manage student accounts
4. The program will request a list of student accounts and TA accounts from the database for this instructor user.
5. The database will reply with a list of student accounts and TA accounts that are recorded as being under this instructor user.
6. The program will display an interactive list of student accounts and TA accounts
7. The user may choose different sorting or display options
8. The program will correctly respond to the user’s choice.
9. The user may select individual student accounts.
10. The program will show the option of looking at all files uploaded by this student.
11. The user may select this option.
12. The program will contact the database with the request.
13. The server will respond with the correct list of files
14. The program will display this list of files.
15. The user will eventually finish looking at this list of files and close this list
16. The user will eventually finish looking at the list of students and close this list.

*Alternate Flows:*

5a: If the database does not send the list, or the list is empty, then the program will not display any students on this page.

9a: After selecting a student account or TA account, the program will also offer the option to delete this account.

9b: The user will select this option.

9c: The program will ask for confirmation to delete this student account

9d: The user will confirm.

9e: The program will again ask for confirmation to delete this student account

9f: The user will confirm.

9g: The program will ask if it should also delete all files uploaded by this student account.

9h: The user will choose yes or no.

9i: The program will send a delete request to the server

9j: The server will delete the correct accounts and/or files

9k: The server will send confirmation back to the program.

9l: The program will show confirmation to the user.

14a: The user will select any number of these files to download.

14b: The program will ask for confirmation.

14c: The user will confirm to download these files

14d: The program will send a request to the database.

14e: The server will send the correct files to the program

14f: The program will save the files to the computer.

14g: The program will open the file directory it saved the files to.

**U9: Instructor Manages Assignments**

*Actors:*

* Instructor User (Has an existing account)
* Locally run program.
* Database

*Triggers:*

* The user indicates that he/she wants to add an assignment or change assignment options.

*Preconditions:*

* User has logged in.

*Post-conditions:*

* The user will finish managing the assignments

*Normal Flow:*

1. The user will open the program and log in.
2. The user will navigate to a course page to view the current list of assignments for that course
3. The program will ask the database for that list.
4. The database will send that list to the program.
5. The program will display that list.
6. The program will show the option to add a new assignment.
7. The user will choose that option.
8. The program will open a menu to enter the assignment number, due date and time, final upload date and time.
9. The user will enter that information.
10. The program will check that there isn’t already an assignment with that number, and that the due time is in the future, and that the late time is after the due time.
11. The program will send this info to the database.
12. The database will save the new assignment.
13. The database will send confirmation back to the program.
14. The program will display confirmation.
15. The user may select any assignment in the list to go to the assignment page
16. The program will ask the database for a list of files uploaded for this assignment.
17. The database will send the list back to the program.
18. The program will display the list of files for this assignment.
19. The program will give the option to edit the due date and/or final upload date
20. The user will select that option
21. The program will show the current dates in an editable format.
22. The user will change the dates and hit confirm.
23. If a date or time is getting moved earlier, the program will check if any files have already been uploaded after that date.
24. The program will send the info to the database.
25. The database will update the due date.
26. The database will send confirmation back to the program.
27. The program will confirm that the due date has been changed.
28. The user will finish managing the assignments and close this menu.

*Alternate Flow:*

4a: If there is a connection error to the database, or the list is currently empty, then the program will not display any assignments on this page.

10a: If any of those checks fail the program will not send to the database and tell the user what the problem is.

13a: If the program does not receive confirmation it will display a connection error.

17a: If the program does not receive a list from the database, or the list is empty, it will not display any files on this page.

18a: The program will offer the option to delete that assignment.

18b: The user will select that option.

18c: The program will ask the user to confirm they really want to delete it.

18d: The user will confirm they want to.

18e: The program will send a delete request to the database.

18f:  The database will delete the records of this assignment and all files uploaded under it.

18g: The database will send confirmation to the program that it deleted the files.

18h: The program will go back to the list of assignments, with this assignment removed   from that list.

23a: If any files have been uploaded after the time that the new due date would be set to, the program will display an error and not send the information to the database.

26a: If the program does not receive any confirmation from the database it will display a connection error and give the user the option to try sending it again.

**U10: Instructor Manages Courses**

*Actors:*

* Instructor User (Has an existing account)
* Locally run program.
* Database

*Triggers:*

* The user indicates that he/she wants to add a course or change course options.

*Preconditions:*

* User has logged in.

*Post-conditions:*

* The user will finish managing their courses

*Normal Flow:*

1. The user will open the program and log in.
2. The user will navigate to their user profile to view a list of their courses
3. The program will ask the database for that list.
4. The database will send that list to the program.
5. The program will display that list.
6. The program will show the option to add a new course.
7. The user will choose that option.
8. The program will open a menu to enter the course ID and course name.
9. The user will enter that information.
10. The program will check that there isn’t already a course with that ID.
11. The program will send this info to the database.
12. The database will save the new course.
13. The database will send confirmation back to the program.
14. The program will display confirmation and add the course to the displayed list.
15. The user may select any course in the list to go to the course page
16. The program will ask the database for a list of assignments created for this course
17. The database will send the list back to the program.
18. The program will display the list of assignments for this course.
19. The program will give the option to change the active or inactive status of the course.
20. The user will change this status.
21. The program will send the info to the database.
22. The database will update the course status.
23. The database will send confirmation back to the program.
24. The program will confirm that the status.
25. The user will finish managing courses and close this menu.

*Alternate Flow:*

4a: If there is a connection error to the database, or the list is currently empty, then the program will not display any courses on this page.

10a: If this check fails the program will not send to the database and tell the user to choose a different ID.

13a: If the program does not receive confirmation it will display a connection error.

17a: If the program does not receive a list from the database, or the list is empty, it will not display any assignments on this page.

18a: The program will offer the option to delete that course.

18b: The user will select that option.

18c: The program will ask the user to confirm they really want to delete it.

18d: The user will confirm they want to.

18e: The program will send a delete request to the database.

18f:  The database will delete the records of this course and all files uploaded under it.

18g: The database will send confirmation to the program that it has been deleted.

18h: The program will go back to the list of courses, with the course removed from the list.

23a: If the program does not receive any confirmation from the database it will display a connection error and give the user the option to try sending it again.

**U11: TA Downloads Files**

*Actors:*

* TA User (Has an existing account)
* Locally run program to look at files
* File Storing database

*Triggers:*

* The user indicates that he/she wants to download files.

*Preconditions:*

* User has logged in.

*Post-conditions:*

* The user will have the requested files on their computer.

*Normal Flow:*

1. The user will open the program and log in.
2. The program will send the database a request for the list of courses the TA has been added to by a professor.
3. The database will send the list of courses.
4. The program will display the list of courses.
5. The TA will select one of the courses.
6. The program will send the database a request for the list of assignments in this course.
7. The database will send the list of assignments.
8. The program will display the list of assignments.
9. The user will select one of the assignments.
10. The program will send the database a request for a list of files uploaded for this assignment.
11. The database will send the list of files.
12. The program will display the list of files.
13. The program will give the option to download all files.
14. The user choses that option.
15. The program sends a request to the database to download these files.
16. The database will send these files to the program.
17. The program will save these files to the computer and open the folder they were saved to.
18. The TA will close the program and start looking at the files to mark.

*Alternate Flows:*

3a: If the program never receives the list from the database, or the list is empty, it will not show any courses on this page.

7a: If the program never receives the list from the database, or the list is empty, it will not show any assignments on this page.

11a: If the program never receives the list from the database, or the list is empty, it will not show any files on this page.

12a: If the professor has created a plagiarism report for this assignment, then the program will show the option to download it to the user.

12b: The user selects the option

12c: The program sends a request to the database for that file.

12d: The database sends the report to the program.

12e: The program saves the report to the computer and opens it in the default PDF viewer.

13a: The user can select any number of files in the list and chose to only download those files.

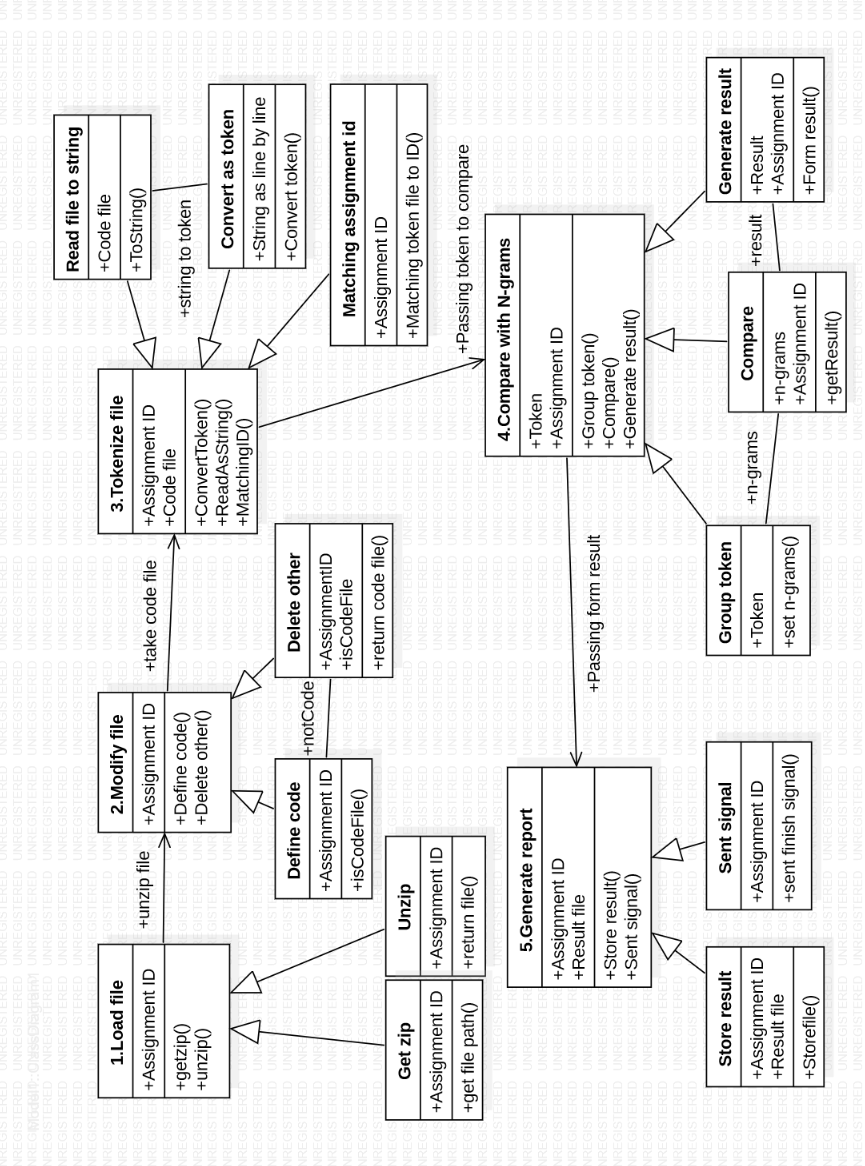
13b: The program will send a request to the database to download the correct files.

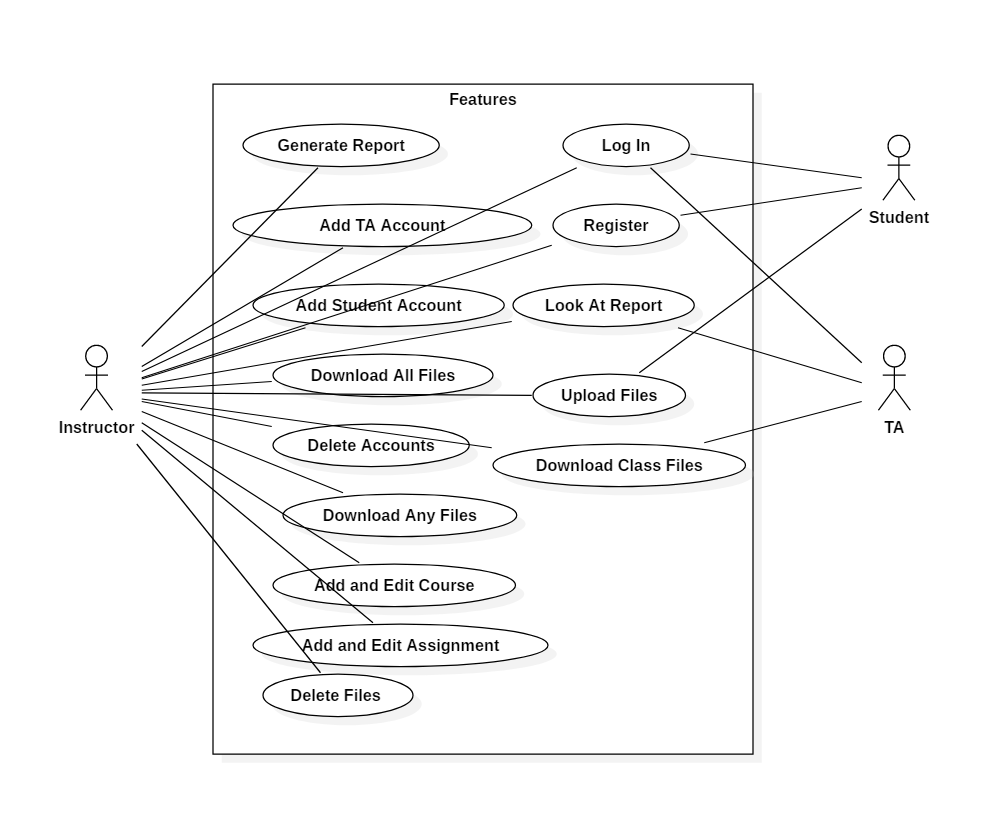
16a: If the program never receives the files, it shows a connection error and gives the user the option to try again.

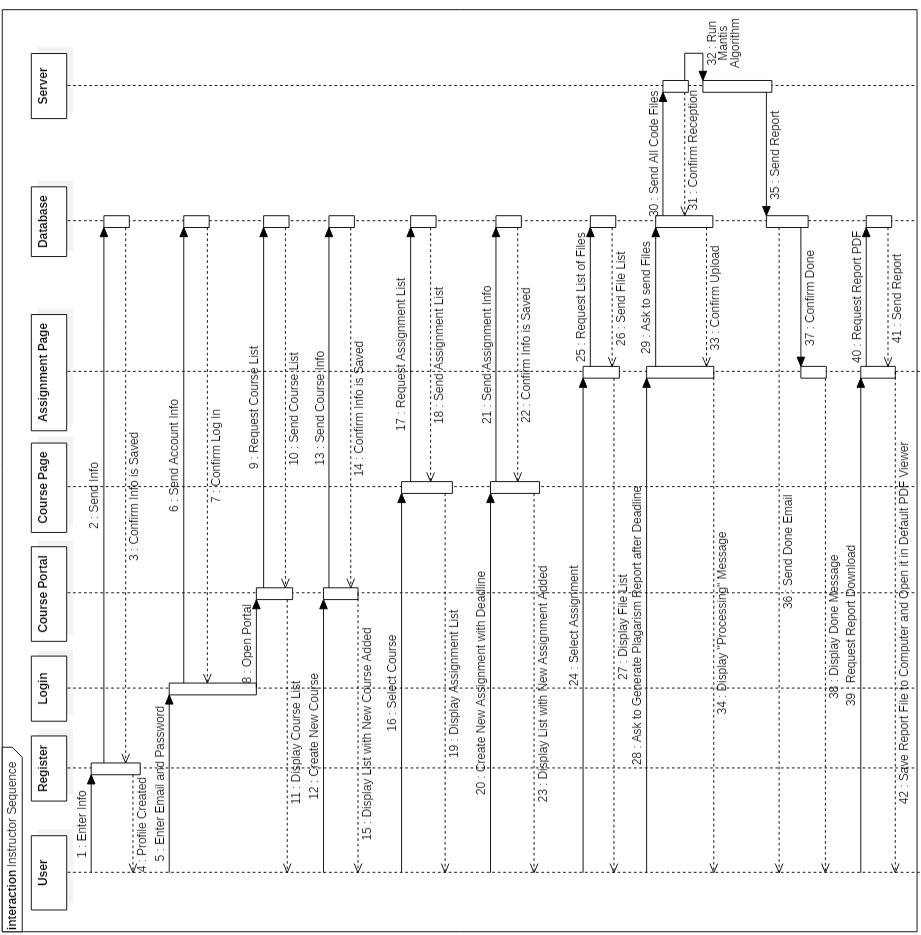
Appendix C: Issues List

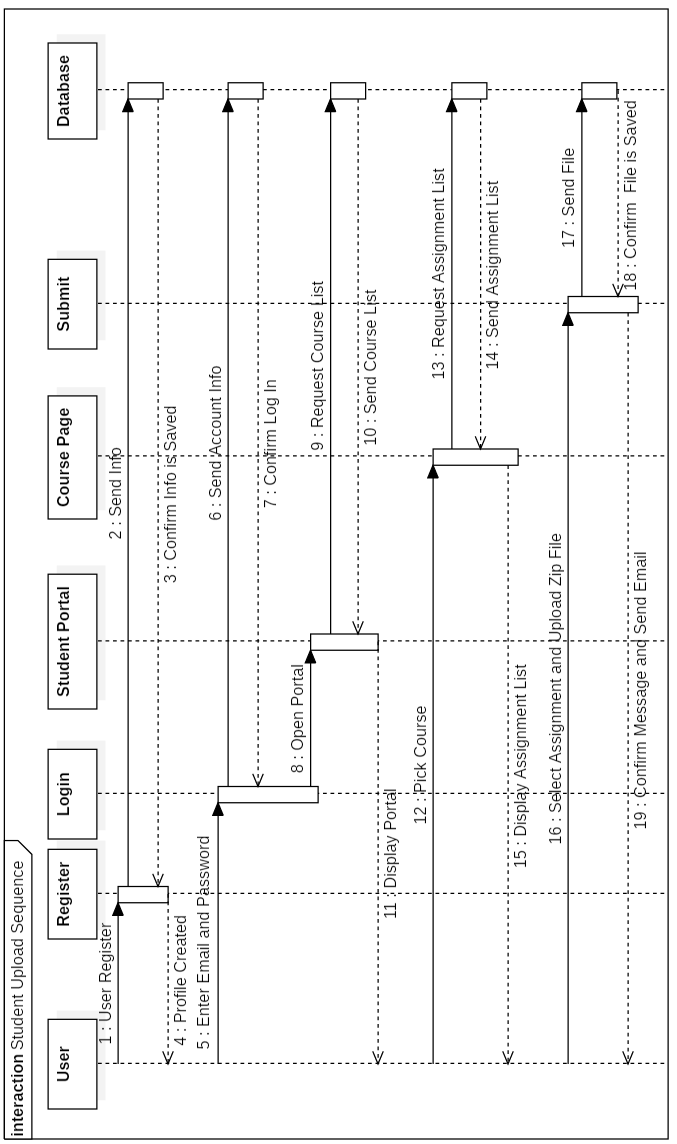
* The way the output assignment file is displayed with yellow highlights once it is detected as plagiarized code.
* The way we decide to store the hashed intermediate code. We will require the files that are being sent from subsystem to subsystem to be consistent and could be retraced back to the original user. Since it appears anonymity for our test data is not necessary, the code does not need to be hashed, but the code similarities will be able to go back to the assignment ID, so we can see which assignment is copied, without potentially viewing the other details of the student. The system may anonymously send a message to the professor, stating that this assignment id has a possibility of plagiarism, but our system won’t be able to connect it back to the student, which provides a level of anonymity.

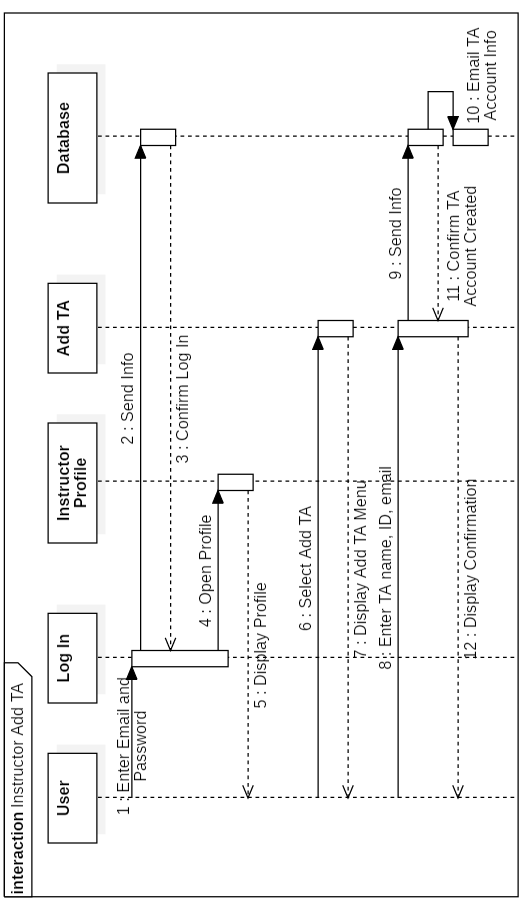
Appendix D: Diagrams

**Figure 5:** UML class diagram; Plagiarism detection algorithm.

**Figure 19:** Use case diagram; Mantis service features.

Figure 20: Sequence Diagram; instructor interactions.

Figure 21: Sequence Diagram; Student interactions.

Figure 22: Sequence Diagram; TA interactions.

Appendix D: Tests

Appendix E: Log History