Software Development Plan

1. Scope.

This section shall be divided into the following paragraphs.

1.1 Identification.

Title: University of Maryland Baltimore County Automated Grading System

Abbreviations:

UMBC - University of Maryland Baltimore County

CSEE - Computer Science and Electrical Engineering

FERPA - Family Educational Right and Privacy Act

1.2 System overview.

The purpose of this software is to automatically run and grade assignment submitted by students over the course of semester, and provide a mechanism for instructors to review this process for their classes. The sponsor of this project is Maksym Morawski. The the acquirer of this system will be the CSEE department of UMBC. The users include both students and instructors. The developers are the team known as “Graders”. The main support agency for the ongoing use of this software will be the acquirer. Operating sites will be a physical domain hosted by a dedicated server of the acquirer.

1.3 Document overview.

Not applicable. This project is not protected by an NDA and does not require any financial investment on the part of the sponsor or acquirer.

1.4 Relationship to other plans.

[Not applicable.]

2. Referenced Documents

All relevant development documents can be found here:

http://datahole.ddns.net/cmsc447/main/documents.html

http://www2.ed.gov/policy/gen/guid/fpco/ferpa/index.html

Web Server Domain - http://datahole.ddns.net/cmsc447/

3. Overview of required work.

A.Requirements and constraints on the system and software to be developed

All user of the software must be either students or instructors at umbc with a valid account for login through google. Students will only have access to their associated assignments within their associated classes. Instructors will only have access to assignments within their associated classes and the evaluation of students in those classes with respect to those assignments. The system must provide meaningful feedback on all student submissions with up to three attempts per assignment per student. The system must be able to accept, manage, and securely store documents and other input provided by all users. The system must also include the functionality as enumerated in the use case document.

B. Requirements and constraints on project documentation

Instructions for use by administrators, instructors, and students will be provided. The user cases for each group of users is described in the use case document.

C. Position of the project in the system life cycle

The project is currently in the inception phase and no functionality has been implemented.

D. The selected program/acquisition strategy or any requirements or constraints on it
[Not applicable.]
E. Requirements and constraints on project schedules and resources
Alternative obligations for the software development team will place time constraints on what can be accomplished within the limited time frame of one school semester. Every member of the team will require access to a development environment and a synchronized repository where development will progress.
F. Other requirements and constraints, such as on project security, privacy, methods, standards, interdependencies in hardware and software development, etc.
Student grades are to remain confidential between student and instructor as listed in the referenced document outlining FERPA. The system must use an interface that enables the use of UMBC student and instructor accounts. The software must be deployed on a dedicated server with internet access and web services enabled.
4. Plans for performing general software development activities.
4.1 Software development process.
The Rational Unified Process will be followed as part of the software development process. Requirements will be modified and created throughout the project according to the amount of progress made and feedback from Maksym Morawski. Task scheduling will be needed at all points to maximize output by the team. Coding and testing will primarily take place during Construction and Deployment phases, since elaboration of software requirements and the architectural design will be necessary beforehand.

Components for the various use cases will be implemented sequentially in the order of priority, and tested and deployed into the stable version.
4.2 General plans for software development.
4.2.1 Software development methods.
The team will use development environments at remote locations and coordinate efforts via voice over IP services. Version control software will be used to facilitate ongoing development and the stable project will be modified only through testing of additional components. Testing environments will be accessed remotely by development team members via secure shell, file transfer protocol, and over a web domain.
4.2.2 Standards for software products.
[Not applicable.]
4.2.3 Reusable software products.
4.2.3.1 Incorporating reusable software products.
- PHP5
- MySQL
- Apache (Web Server Hosting)
- Google API for UMBC user account logins

MySQL will be used to create a structured database for establishing and executing queries on information related to student coding assignments. Apache will assist in creating a secure HTTP web server. The Google API provides functionality for connecting to google services, including UMBC logins that are run through Gmail.

Any other aspect of the software will be implemented by hand without reusable software products.

4.2.3.2 Developing reusable software products.

Possible opportunities exist to extrapolate the basic automatic grading system onto other programming courses. Currently the automatic grading system will be designed with consideration for introductory Computer Science courses incorporating Python scripts. If the software remains modular, the staff will be able to identify opportunities to reuse the software by abstracting as much of the functionality from python as possible.

4.2.4 Handling of critical requirements.

This paragraph shall be divided into the following subparagraphs to describe the approach to be followed for handling requirements designated critical. The planning in each subparagraph shall cover all contractual clauses concerning the identified topic.

4.2.4.1 Safety assurance

Not applicable. There are no inherent safety risks with the use of this software. Information is shared strictly between student and instructor. Introductory programming assignments do not include sensitive data, and any other concerns fall under security or privacy.

4.2.4.2 Security assurance

The system will need to mitigate the risks of server tampering, malware infection, etc. that could occur through the execution of uploaded code by students.

4.2.4.3 Privacy assurance

As mentioned, privacy of student grading information is required by FERPA. Privacy of submitted programs is also of concern. All user information must be inaccessible by other users, unless permitted, and must be properly stored. The security assurance will also mitigate these issues.

4.2.4.4 Assurance of other critical requirements

Not applicable because there are no contractual clauses concerning any critical requirements for this project.

4.2.5 Computer hardware resource utilization.

A dedicated server with internet access and web services enabled must have a system administrator to ensure continual operation.

4.2.6 Recording rationale.

[Not applicable.]

4.2.7 Access for acquirer review.

[Not applicable.]

5. Plans for performing detailed software development activities.

This section shall be divided into the following paragraphs. Provisions corresponding to non-required activities may be satisfied by the words "Not applicable." If different builds or different software on the project require different planning, these differences shall be noted in the paragraphs. The discussion of each activity shall include the approach (methods/procedures/tools) to be applied to: 1) the analysis or other technical tasks involved, 2) the recording of results, and 3) the preparation of associated deliverables, if applicable. The discussion shall also identify applicable risks/uncertainties and plans for dealing with them. Reference may be made to 4.2.1 if applicable methods are described there.

5.1 Project planning and oversight.

This paragraph shall be divided into the following subparagraphs to describe the approach to be followed for project planning and oversight. The planning in each subparagraph shall cover all contractual clauses regarding the identified topic.

5.1.1 Software development planning (covering updates to this plan)

5.1.2 CSCI test planning

Each component as defined in the use cases will be thoroughly tested to cover all possible types of input by users.
http://stackoverflow.com/questions/4312760/official-definition-of-csci-computer-software-configuration-item

5.1.3 System test planning

The overall architecture will be evaluated for throughput and responsiveness. Unit tests will be run periodically to generate requests and ensure the system delivers intended responses. Information on peak traffic, maximum throughput, and any responsiveness issues will be recorded and updated per each test.

5.1.4 Software installation planning

All reusable software including PHP5, MySQL, and Apache must be deployed on the system used for installation of this project's software. Copies of scripts, markup documents, style documents, and database architecture will also be required. This can be encapsulated into a single repository and downloaded for deployment. A series of commands will need to be issued on the intended server to facilitate installation of the packages mentioned.

5.1.5 Software transition planning

All deliverables will be retained in a repository that can be downloaded over the internet.

5.1.6 Following and updating plans, including the intervals for management review

Plans will be followed and executed with respect to the documents outlined in Section 2. Plans will be updated according to meetings with our sponsor, Maksym Morawski, and also may vary due to time constraints and technical constraints. All revisions to plans will be reflected in the aforementioned documents and confirmed through a consensus by the development team over IP communications.

5.2 Establishing a software development environment.

This paragraph shall be divided into the following subparagraphs to describe the approach to be followed for establishing, controlling, and maintaining a software development environment. The planning in each subparagraph shall cover all contractual clauses regarding the identified topic.

5.2.1 Software engineering environment

Environments will differ for each development team member. A text editor will be necessary for the construction of essential code. IDEs may also be used depending on applicability and convenience for the immediate problem. Each member will operate remotely using personal computers and devices.

5.2.2 Software test environment

One of the development team member will provide access to a personal web server with secure shell, and file transfer protocol capabilities. The server will support PHP5 and MySQL, and operate using Apache. Version control can be synced with the server for testing purposes.

5.2.3 Software development library

Only the default libraries for the associated packages will be required.

5.2.4 Software development files

Version control software will be used to develop and maintain the repository containing the software. Google drive will be used for all other pertinent files to be shared among the development team.

5.2.5 Non-deliverable software

Unit testing of the various components will not be delivered for final deployment. Previous builds before the most recent stable version will be deprecated and will not be delivered for final deployment.

5.3 System requirements analysis.

This paragraph shall be divided into the following subparagraphs to describe the approach to be followed for participating in system requirements analysis. The planning in each subparagraph shall cover all contractual clauses regarding the identified topic.

5.3.1 Analysis of user input

Every possible user input will be mapped out and organized within a flow chart to examine the possible paths of input in order to detail how a functionality of the software can be executed.

5.3.2 Operational concept

The operational concepts are outlined in the use case document for each intended user.

5.3.3 System requirements

Minimum system requirements include a device and corresponding storage unit to transfer files and a functional internet connection to connect to the web server.

5.4 System design.

5.4.1 System-wide design decisions

The system will be run using a web interface and all information will be processed through a server.

5.4.2 System architectural design

The system will function on a web server which will process user requests using PHP5 and store information in a database using MySQL.

5.5 Software requirements analysis.

Software requirements will be analyzed by reviewing the overlying web architecture and tailoring the design accordingly for each use case. The interface for each case will be implemented using HTML and CSS, while any processing will require PHP5 and information storage and retrieval will use MySQL.

5.6 Software design.

This paragraph shall be divided into the following subparagraphs to describe the approach to be followed for software design. The planning in each subparagraph shall cover all contractual clauses regarding the identified topic.

5.6.1 CSCI-wide design decisions

User interface and actual processing must be separated. Pertinent information for each user linking back to an associated google account must be stored.

5.6.2 CSCI architectural design

Each component of the design must work towards fulfilling all of the enumerated use cases. Components may be computer systems, function calls, etc. There will be a logical and coherent transition between each component. The design will specify what information is necessary for a given component, how that information will be modified, and where it will go next according to a component layout. Diagrams will be made to display these components and how they are interconnected.

5.6.3 CSCI detailed design

Each component will be designed according to corresponding use cases and the details of the design will be based on the main success scenario of that use case as well as the defined extensions for that scenario.

5.7 Software implementation and unit testing.

This paragraph shall be divided into the following subparagraphs to describe the approach to be followed for software implementation and unit testing. The planning in each subparagraph shall cover all contractual clauses regarding the identified topic.

5.7.1 Software implementation

The web server that processes HTTP requests will be run using Apache. This server will route traffic to the software's domain where PHP scripts will run to process user requests, create database entries, run database queries, and perform the functions pertaining to the UML diagrams. The database will be created using MySQL and all webpages will use HTML5 and CSS3.

5.7.2 Preparing for unit testing

As mentioned in 5.2.2, a personal server will be established to process submissions and perform the outlined software tasks.

5.7.3 Performing unit testing

Unit testing will be done on all aspects of the software. Server performance will be tested using high-traffic scenarios. Grade submission accuracy will be tested by automatically submitting assignments and comparing received grades to expected grades. User access and functionality will be tested manually by walking through the software and performing standard use scenarios (Use Case Document).

5.7.4 Revision and retesting

Edge cases must be considered for submitted programs to the server. Server side processing must be modified to produce some reliable output that does not result in failure given any possible input. Upon deployment the software will require retesting to consider response time, stability, and throughput on the intended hardware which is to be determined.

5.7.5 Analyzing and recording unit test results

Accuracy will be weighed by comparison to manually created rubrics that outline criteria for a certain grade on programming assignments. The unit tests will verify that the correct grade has been given to student submissions. All other backend functionality (e.g. submissions, 3 submission limit, error checking) will also be run through a script and then output from debugging messages inside PHP Scripts will be outputted within corresponding log files and examined for correctness.

5.8 Unit integration and testing.

This paragraph shall be divided into the following subparagraphs to describe the approach to be followed for unit integration and testing. The planning in each subparagraph shall cover all contractual clauses regarding the identified topic.

5.8.1 Preparing for unit integration and testing

When all the units have been thoroughly tested, the unit integration will take place. The team will adopt an integration procedure based on the unit testing results. The units will be combined to slowly shape the actual software. All the units/components will be combined into the overall architecture of the designed and proposed plan for overall software. From here on, the next procedure will involve performing initial tests on the unit integration into the system.

5.8.2 Performing unit integration and testing

When the programs are ready for integration, the actual procedure of the unit integration will occur. In this phase, the software 'pieces' will be gradually combined together to form the desired output. After the integration of all the units occurs, the testing phase will come. Thorough testing will occur to test out every single outcome of the software and to see if the integration of software does not produce additional issues. The testing of the integration of the overall system (PHP,MYSQL,APACHE) will take place at this phase of the project.

5.8.3 Revision and retesting

After the initial tests, the next phase of the project will require the team to make any necessary changes or revisions in the project to solve the issues that occurred during the initial test phase. All the team members will be assigned the revisions (distributed evenly) in order to minimize the time being spent in this phase. After the revisions have been made, the whole software will be run against different testing procedures. The retest will then allow the team to gather further results and record them in the next phase of the project.

5.8.4 Analyzing and recording unit integration and test results

The revision tests will provide the team with resulting data. This will be used to analyze the results by all the team members. Based on the analyzation, the team will try to create strategies to solve any arising issues. All the results will be recorded and thoroughly analyzed in this phase.

5.9 CSCI qualification testing.

5.9.1 Independence in CSCI qualification testing

Each part of the software will be tested individually

5.9.2 Testing on the target computer system

Development and testing is done on a similar unix based system and ported to gl when needed

5.9.3 Preparing for CSCI qualification testing

Not applicable because qualification testing for this software is of a minimal nature.

5.9.4 Dry run of CSCI qualification testing

Not applicable because qualification testing for this software is of a minimal nature.

5.9.5 Performing CSCI qualification testing

Unit tests for each component will be performed as stated, and user testing of the integrated components will also be performed

5.9.6 Revision and retesting

Revisions to the system will be made based on unit testing and integration testing results. and retested until all test pass.

5.9.7 Analyzing and recording CSCI qualification test results

Results will be recorded on the web server and analyzed on a pass fail basis.

5.10 CSCI/HWCI integration and testing.

5.10.1 We will come up with unit test scenarios before each piece of the system is finished and then apply the unit tests to the individual parts of the system

5.10.2 Testing will be performed as each part of the system is finished from the predetermined unit tests

5.10.3 We will retest the parts of the software after we finished a new parts of the software as well as integrate the parts of the software (login with uploading etc.)

5.10.4 We will have a checklist of our unit tests on the group webpage

5.11 System qualification testing.

[Not applicable.]

5.12 Preparing for software use.

This paragraph shall be divided into the following subparagraphs to describe the approach to be followed for preparing for software use. The planning in each subparagraph shall cover all contractual clauses regarding the identified topic.

5.12.1 Preparing the executable software

Not applicable because the software is not executable and will be deployed as is on a web server with a PHP5 script interpreter.

5.12.2 Preparing version descriptions for user sites

Descriptions will be written for each user site.

5.12.3 Preparing user manuals

[Not applicable.]

5.12.4 Installation at user sites

[Not applicable.]

5.13 Preparing for software transition.

This paragraph shall be divided into the following subparagraphs to describe the approach to be followed for preparing for software transition. The planning in each subparagraph shall cover all contractual clauses regarding the identified topic.

5.13.1 Preparing the executable software

This phase is not applicable. We are developing a web-based software. Hence, there will be no exe file.

5.13.2 Preparing source files

The Project Version Control System that we will be using for this project will be Git. Git will combine all the source files in an organized manner. We can prepare the source files for final delivery int this phase.

5.13.3 Preparing version descriptions for the support site

Not applicable because there will not be a support site for this software.

5.13.4 Preparing the "as built" CSCI design and other software support information

Readme of server setup for administrators, and useage for instructors, as well as information for instructors to inform students of usage.

5.13.5 Updating the system design description

Once the system has been developed and thoroughly tested, the team will update the system design with any changes that were made. The original designed will be updated to include all the specifics and design description.

5.13.6 Preparing support manuals

[Not applicable.]

5.13.7 Transition to the designated support site

[Not applicable.]

5.14 Software configuration management.

The software configuration management will be through a private github repository. A clone of the master branch will be kept up to date in a directory on the development server.

5.14.1 Configuration identification

Repository Name - cmsc447

https://github.com/vkonen1/cmsc447.git

5.14.2 Configuration control

The web interface provided on github and git will be used to control the configuration management software.

5.14.3 Configuration status accounting

Separate branches will be used for individual components of the software as will be detailed in the architectural design. The state of the master branch shall be kept in a functioning and fully tested capacity.

5.14.4 Configuration audits

Reviews of separate component branches will be conducted and functionality will be tested before it is merged into the master branch.

5.14.5 Packaging, storage, handling, and delivery

The contents of the repository will be delivered to the customer along with instructions for setup of the required third party server software.

5.15 Software product evaluation.

5.15.1 In-process and final software product evaluations

The final evaluation report will be created by the team which will include all the changes made to the original design and if the changes contributed to the desired output of the overall project. It will also contain an overall evaluation that whether or not the project meets the requirement of the customer and gives out the desired results. The team will also evaluate itself on its working throughout the different phases of the software development process.

5.15.2 Software product evaluation records, including items to be recorded

Records will be kept on the web server.

5.15.3 Independence in software product evaluation

Our customer will be the source of evaluation separate from unit testing and evaluation of the software by the development team to ensure the product meets their requirements.

5.16 Software quality assurance.

5.16.1 Software quality assurance evaluations

In this part of the process, the team will evaluate all the testing procedures and measures taken for quality assurance purposes. This will help us understand which type of testing procedures can be applied to each and every scenario to ensure quality assurance.

5.16.2 Software quality assurance records, including items to be recorded

Quality assurance records will be kept on the Web Server demonstrating whether components passed particular unit tests.

5.16.3 Independence in software quality assurance

The quality of the software is contingent upon meeting of use cases in the referenced document.

5.17 Corrective action.

5.17.1 Problem/change reports, including items to be recorded (candidate items include project name, originator, problem number, problem name, software element or document affected, origination date, category and priority, description, analyst assigned to the problem, date assigned, date completed, analysis time, recommended solution, impacts, problem status, approval of solution, follow-up actions, corrector, correction date, version where corrected, correction time, description of solution implemented)

Each commit to the Github repository will include a log of changes. More detailed accounts of changes and errors will be added as text files to the repository. Errors and problems can be assigned team members and a priority number, and will be flagged on Github for team visibility using given Github site functionality.

5.17.2 Corrective action system

QA reports will be put onto the web server as a checklist of things to fix / get done as well as commit logs to the server or the repository

5.18 Joint technical and management reviews.

5.18.1 Joint technical reviews, including a proposed set of reviews

Analysis of the software and current status on progress will be discussed weekly between members of the group based on the schedule created in 6.1. Efforts will be made to create and execute unit tests to ensure quality of the code and correct functionality. Any issues will be flagged and ticketed accordingly. Meetings with sponsor and adviser, Maksym Morawski, will be held as necessary when technical difficulties arise.

5.18.2 Joint management reviews, including a proposed set of reviews

[Not applicable.]

5.19 Other software development activities.

5.19.1 Risk management, including known risks and corresponding strategies

[See Risk Assessment Document]

5.19.2 Software management indicators, including indicators to be used

[Not applicable.]

5.19.3 Security and privacy

All the software development processes will be conducted in a secure and private environments. The Version Control System (Git) will further help secure the data and source code.

5.19.4 Subcontractor management

[Not applicable.]

5.19.5 Interface with software independent verification and validation (IV&V) agents

[Not applicable.]

5.19.6 Coordination with associate developers

[Not applicable.]

5.19.7 Improvement of project processes

[Not applicable.]

5.19.8 Other activities not covered elsewhere in the plan

6. Schedules and activity network.

1. Schedule(s) identifying the activities in each build and showing initiation of each activity, availability of draft and final deliverables and other milestones, and completion of each activity

Each activity is sequential one following the next, refer to the use case document for deliverables and what is needed to be done in each step

In-order

Server setup - 1 week

[completed]

UC-1 login and upload - 2-3 weeks

UC-5 Admin access - 1-2 weeks

UC-3 Instructor assignment setup - 1 week

UC-4 Instructor review - 1 week

UC-2 Grading of assignment - 3 weeks

Final testing - 1 week

Final Delivery

2. An activity network, depicting sequential relationships and dependencies among activities and identifying those activities that impose the greatest time restrictions on the project

Login into an authorized account is required for all subsequent activities. Authorized accounts populate relations between user and student and provide a secure platform. Grading cannot occur unless assignments are uploaded, requiring there be a system in place to access files uploaded by students or instructors. Admin access is needed to design the instructor level of access and functionality, and in order to assign access levels to users. Grading assignments will take the most time. It is the most sensitive job in the software and will be the main source of load on the server, requiring numerous unit tests.

7. Project organization and resources.

7.1 Project organization.

7.2 Project resources.

a. Personnel resources, including:

1. The estimated staff-loading for the project (number of personnel over time)

There are five members on the development team that must cover all aspects of the project.

2. The breakdown of the staff-loading numbers by responsibility (for example, management, software engineering, software testing, software configuration management, software product evaluation, software quality assurance)

As of now, each of the five staff members will be involved in all aspects of development described above.

3. A breakdown of the skill levels, geographic locations, and security clearances of personnel performing each responsibility

Not Applicable

b. Overview of developer facilities to be used, including geographic locations in which the work will be performed, facilities to be used, and secure areas and other features of the facilities as applicable to the contracted effort.

Most work will be performed remotely on personal machines. Some coordination effort will take place on the UMBC campus.

c. Acquirer-furnished equipment, software, services, documentation, data, and facilities required for the contracted effort. A schedule detailing when these items will be needed shall also be included.

Not Applicable

d. Other required resources, including a plan for obtaining the resources, dates needed, and availability of each resource item.

A dedicated web server to development and testing will be hosted by one of the members of the development team.

8. Notes.

Not Applicable - see use cases for documentation of what is needed to be done and the terms used in this document do not need a glossary.

A. Appendixes.

Not Applicable - definitions and abbreviations are covered in the notes section of the document, refer to section 2 for these