Software Requirements Specification

1. Scope.

This section shall be divided into the following paragraphs.

1.1 Identification.

The system and software contained within this document belong to the Automatic Grading System. No previous versions or forms of identification exist.

1.2 System overview.

The purpose of the software is to provide a method by which instructors can evaluate students through assignments and provide feedback remotely. The software will be accessed by all users over the WWW through a web page interface. The project sponsor is Maksym Morawski. The acquirer of the software is UMBC. The users are instructors, students, and administrative staff. The developers are Team 1 of Dr. Biranne's Software Engineering I class for the Spring 2016 semester. The software will be hosted on a web server provided by one of the developers during construction. The software will then be deployed on a server of the acquirer's choosing.

1.3 Document overview.

The software, when completed, will be able to facilitate the grading process for CMSC 201 and hopefully make it easy for students to submit and get their grades for the class. The software will respect the privacy of all the students in the class as each student will have a unique account and privacy settings associated with it.

2. Referenced documents.

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

3. Requirements.

3.1 Required states and modes.

Different access levels will be given to users of the system dependent on their roles inside classes where automatic grading applies. Students will be given lower level access, such that they can review their classes enlisted in the automatic grading system. They will not be able to view other students' assignments. The view will be restricted such that they can look at all submissions assigned to their student id and functionality will be limited to viewing and submitting assignments. Instructors will be given higher level access. They can view all student submissions pertinent to the classes they are instructing, submit grading criteria per assignment, upload new assignments, or modify existing assignments. Admins exist as another mode of access to the system, and have the highest level of access. They will create the aforementioned student and instructor classes, allowing them the necessary views and access levels for relevant courses.

3.2 CSCI capability requirements.

3.2.1 Uploading programming assignment

- Uploaded assignments must be associated with the student's UMBC user account.
- Only properly named files will be accepted into the system.
- The assignment must be uploaded before the specified due date.

3.2.2 Grading of student assignment

- Student grades must be kept confidential.

3.2.3 Instructor adding assignment

- Instructor can add assignments to courses they are enrolled as instructor.
- Instructor must include a due date.
- Instructor must include a submission name and extension.

3.2.4 Instructor reviewing assignment

- Student grades must be accessible by the instructor.
- Student uploads must be accessible by the instructor.

3.2.5 Administrator adding classes and instructors

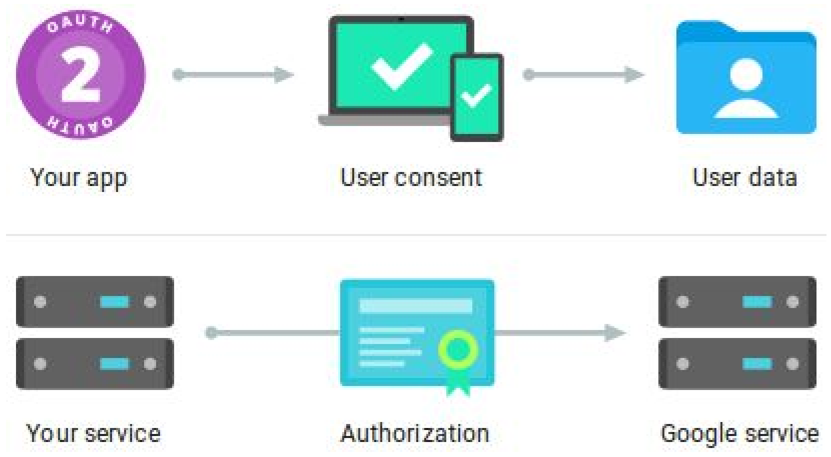
- Class creation must be possible by administrators.
- Instructor assignment must be possible by administrators.

3.3 CSCI external interface requirements.

3.3.1 Interface identification and diagrams.

https://developers.google.com/+/web/api/rest/#api

This interface will be referred to as the Google+ API for the project.



3.3.2 Google+ API

1. Priority that the CSCI must assign the interface: The highest priority would be at log-in; it would switch to low priority after that.

2. Requirements on the type of interface (such as real-time data transfer, storage-and-retrieval of data, etc.) to be implemented: The user will be directed to Google, where they will login, and Google will redirect the user back to the web interface with a provided authentication token and relevant user data.

3. Required characteristics of individual data elements that the CSCI must provide, store, send, access, receive, etc., such as:

1. **Names/identifiers**
    1. **Project-unique identifier:** User Authentication Response
    2. **Non-technical (natural language):** User Authentication
    3. **DoD standard data element name:** Not applicable
    4. **Technical name (e.g., record or data structure name in code or database):** User OAuth 2.0 Token HTTP Response
    5. **Abbreviations or synonymous names:** Not applicable
2. **Data type (alphanumeric, integer, etc.):** JSON String

  3. **Size and format (such as length and punctuation of a character string):** Size relevant to JSON format of user data.
  4. **Units of measurement (such as meters, dollars, nanoseconds):** Not applicable
  5. **Range or enumeration of possible values (such as 0-99):** Not applicable
  6. **Accuracy (how correct) and precision (number of significant digits):** Not applicable
  7. **Priority, timing, frequency, volume, sequencing, and other constraints, such as whether the data element may be updated and whether business rules apply:** The volume Google can supply is far above our needs for this project.
  8. **Security and privacy constraints:** All security and privacy is on Google and the HTTPs request.
  9. **Sources (setting/sending entities) and recipients (using/receiving entities):** Server sends request to google for user authentication. Server receives authentication response token from google for user validation.

4. **Required characteristics of data element assemblies (records, messages, files, arrays, displays, reports, etc.) that the CSCI must provide, store, send, access, receive, etc., such as:**
   1. **Names/identifiers**
      1. **Project-unique identifier:** JSON String
      2. **Non-technical (natural language) name:** User Authentication Data Element Assembly
      3. **Technical name (e.g., record or data structure name in code or database):** JSON String
      4. **Abbreviations or synonymous names:** Not applicable
   2. **Data elements in the assembly and their structure (number, order, grouping):** Username and authentication token within an associative array.
   3. **Medium (such as disk) and structure of data elements/assemblies on the medium:** HTTP Response - JSON formatted string
   4. **Visual and auditory characteristics of displays and other outputs (such as colors, layouts, fonts, icons and other display elements, beeps, lights):** Not applicable
   5. **Relationships among assemblies, such as sorting/access characteristics:** Not applicable, only one assembly
   6. **Priority, timing, frequency, volume, sequencing, and other constraints, such as whether the assembly may be updated and whether business rules apply:** Not applicable
   7. **Security and privacy constraints:** Not applicable, constraints handled by Google

5. **Required characteristics of communication methods that the CSCI must use for the interface, such as:**
   1. **Project-unique identifier(s)**: HTTP/HTTPS
   2. **Communication links/bands/frequencies/media and their characteristics:** Use of established network protocols: HTTP/HTTPS
   3. **Message formatting:** JSON

4. Flow control (such as sequence numbering and buffer allocation): Handled by the protocol

5. Data transfer rate, whether periodic/aperiodic, and interval between transfers: Handled by the protocol

6. Routing, addressing, and naming conventions: Handled by the protocol

7. Transmission services, including priority and grade: Handled by the protocol

8. Safety/security/privacy considerations, such as encryption, user authentication, compartmentalization, and auditing: Handled by the protocol

6. Required characteristics of protocols the CSCI must use for the interface, such as:

1. Project-unique identifier(s): HTTP/HTTPS

2. Priority/layer of the protocol: Networking layer

3. Packeting, including fragmentation and reassembly, routing, and addressing: Handled by the protocol

4. Legality checks, error control, and recovery procedures: Handled by the protocol

5. Synchronization, including connection establishment, maintenance, termination: Handled by the protocol

6. Status, identification, and any other reporting features: Handled by the protocol

7. Other required characteristics, such as physical compatibility of the interfacing entities (dimensions, tolerances, loads, plug compatibility, etc.), voltages, etc.: Not applicable

3.4 CSCI internal interface requirements.

Internal interface to configure database server and layout of database.

3.5 CSCI internal data requirements.

A database server will be configured to manage user data. Decisions about this will be left to design.

3.6 Adaptation requirements.

Not applicable, this project is not location dependant

3.7 Safety requirements.

Not applicable, our software can not endanger lives in any way shape or form

3.8 Security and privacy requirements.

Security on login falls into Google's authentication territory. After that, we must make sure that only the student and instructor can see their own grades.

3.9 CSCI environment requirements.

The computer hardware and operating system must be capable of hosting a web server and processing traffic at a reasonable rate.

3.10 Computer resource requirements.

3.10.1 Computer hardware requirements.

The software would be hosted on a server running a PHP server along with a database. The server does not have to have a lot of memory storage for this project. The server we have currently arranged meets the memory requirements.

3.10.2 Computer hardware resource utilization requirements.

The CPU, RAM, and Disk shall not exceed 90% capacity as to prevent a system lockup

3.10.3 Computer software requirements.

The CSCI will run on a server capable of hosting a web server and will utilize a database server.

3.10.4 Computer communications requirements.
Not applicable, since there are no communication requirements that require transmitting.
3.11 Software quality factors.
The CSCI will have a high priority on being reliable and available so the students and/or instructor can always access it and obtain accurate results. Below that is functionality; it does not need every planned feature, just a majority of them. Maintainability is also on the same level as functionality; it needs to be able to be corrected in case the reliableness fails. The rest are lower priority as the software will not likely be needed to be reused and the useability is not nearly as important as the functionality. The software has no need to be portable.
3.12 Design and implementation constraints.
1. Use of a particular CSCI architecture or requirements on the architecture, such as required databases or other software units; use of standard, military, or existing components; or use of Government/acquirer-furnished property (equipment, information, or software): A web server, database server, and the Google+ API.
2. Use of particular design or implementation standards; use of particular data standards; use of a particular programming language: PHP5 will be used for server side processing. Standard HTML5, CSS, and JavaScript will be used for the web interface. There are no constraints for the operating system for the server, or for the database server.
3. Flexibility and expandability that must be provided to support anticipated areas of growth or changes in technology, threat, or mission: Not Applicable
3.13 Personnel-related requirements.
Administrators, instructors, and students will be using the software. Administrators must follow guidelines of instructor and class setup. Instructors must follow guidelines of assignment setup. All other activities will be facilitated by the system, and errors will be handled for all other cases automatically.
3.14 Training-related requirements.
We will provide a README.txt for the instructors. Youtube videos can be supplied to teach students how to use the software.
3.15 Logistics-related requirements.
System maintenance will initially be on the designers but will transition over to the UMBC system after the CSCI is more widely used and accepted.
3.16 Other requirements.
Not applicable, since all software requirements were covered in this document.
3.17 Packaging requirements.
Not applicable, since there is no physical delivery to be packaged.
3.18 Precedence and criticality of requirements.
Priority Scale - 1 (Very Low) 2 (Low) 3 (Average) 4 (High) 5 (Very High)

Google+ API - 5
Database Server Setup - 5
Uploading programming assignment - 4
Grading of student assignment - 3
Instructor adding assignment - 4

Instructor reviewing assignment - 4
Administrator adding classes and instructors - 4

4. Qualification provisions.

Google+ API - A demonstration will be performed to ensure that the login prompt validates a known UMBC email and authorizes use of the account

Database Server Setup - Analysis of the database structure will be performed to ensure all pertinent data is being recorded and the format of the data is optimal. A demonstration will be done to show that data can be retrieved from the database, and can be stored in the database.

Uploading programming assignment - Verify that assignment was uploaded and accessible by the server.
Grading of student assignment - Tests and analysis will be performed to determine the accuracy and timeliness of the grading system. It must closely approximate a desired grade as outlined by the instructor, and should provide a grade within a reasonable measure of time.

Instructor adding assignment - A demonstration will be performed to show successful adding of an assignment and class access to that assignment.

Instructor reviewing assignment - A demonstration will be performed to show the capability of an instructor to review assignments and access data provided by students and by the system.

Administrator adding classes and instructors - A demonstration will be performed to show that classes and instructors can be added, and instructors will have access to add assignments within those classes.

5. Requirements traceability.

To access UMBC accounts, the Google+ API will be used, allowing secure authentication.

Administrators will be given access to the system manually and through other administrators who will then have the ability to add instructors and classes for those instructors to be assigned to by the administrators.

Instructors will add assignments to their classes and be able to review student submissions and the grades provided by the system on those submissions.

Students will add classes and be able to access the assignments provided by the instructors. They will be able to upload submissions and have them graded by the system, as well as have them reviewed by the instructor. Student submissions will be confidential between the student and instructor only.

Grading of the assignments will require higher throughput and lower latency. This can be traced to computer requirements. The computer must have sufficient RAM and processing power to execute grading quickly and to avoid system lock-ups in the scenario where there is a high level of traffic.

6. Notes.

Not Applicable.

A. Appendixes.

Not Applicable.