System Requirements Specification

Team EVAL

October 29, 2018

College Course Evaluation System

System Requirements Specification

Contents

1 Introduction

1.1 Purpose of this Document

This system requirements specification details what our course evaluation system does and what tests we will make to ensure the system is complete. It includes why we are creating the system, the scope of the product, diagrams that illustrate the system, what we will deliver to the customer, and currently pending issues about the software.

This document is intended for the development team, the product client, Dr. Harlan Onsrud, and potential users of the system. Team EVAL needs this document to ensure that the product works as intended. Dr. Onsrud needs it to know that he will receive the program that he desires. The document also helps the software's users in that they can learn more about the functions of the evaluation system.

1.2 References

LimeSurvey: The online survey tool - open source surveys. (n.d.). Retrieved from https://www.limesurvey.org/

Fowler, M. (2004). UML Distilled: A Brief Guide to the Standard Object Modeling Language. Boston: Addison-Wesley.

1.3 Purpose of the Product

The University of Maine gives out course evaluation surveys to students at the end of each course. The survey is filled on a bubble sheet and is then scanned. Harlan Onsrud finds it inconvenient for the school administrators to manually scan and compile the survey results. Current campus experiments with electronic evaluations are not fully automated and do not seem to fulfill the desires of students and faculty. He desires an online, automated evaluation system that is responsive to the University's community needs and improves productivity.

1.4 Product Scope

Team EVAL will create a product which interfaces with an already existing survey software, LimeSurvey, to provide both individual teachers and administrators the ability to create and administer evaluations. This product will be usable even by those without technical backgrounds, and it will have an intuitive interface for setting up and administering teacher evaluations.

It will allow users to create one or more courses with predefined survey data and rosters of e-mail addresses. For each class, users will be able to choose from provided questions or enter their own custom questions. The question sets that they make may be saved to their account and applied to future evaluations they create. Upon request of the user, the product will create a LimeSurvey with the supplied information. It then sends an invitation to complete the survey to the students on the class' roster via e-mail, along with reminders as appropriate. When the survey is terminated by the instructor/administrator or by a certain data and time, the product will allow users to view and download a statistical analysis and clear visualization of the data collected for one or more of their courses.

The software will support the reporting of accumulated data and the appropriate statistics and graphics derived from it for each academic period. A report will be generated for the following levels: course section, all sections of the same course, all courses of each instructor, all courses with the same designator (e.g. "COS"), all courses in the same department/school, all courses in the same college, and all courses in the same university.

The product will be completed in time to administer teacher evaluations for the spring of 2019 at the University of Maine, whether evaluation forms are created by instructors or administrators. A UML diagram (Figure 1) shows the scope as a dotted rectangle.

LimeSurvey is an existing software that has the ability to create and publish surveys. This product will simply interface with LimeSurvey and interact with its components. LimeSurvey lies outside of the scope of this project and will be treated as a third party actor.

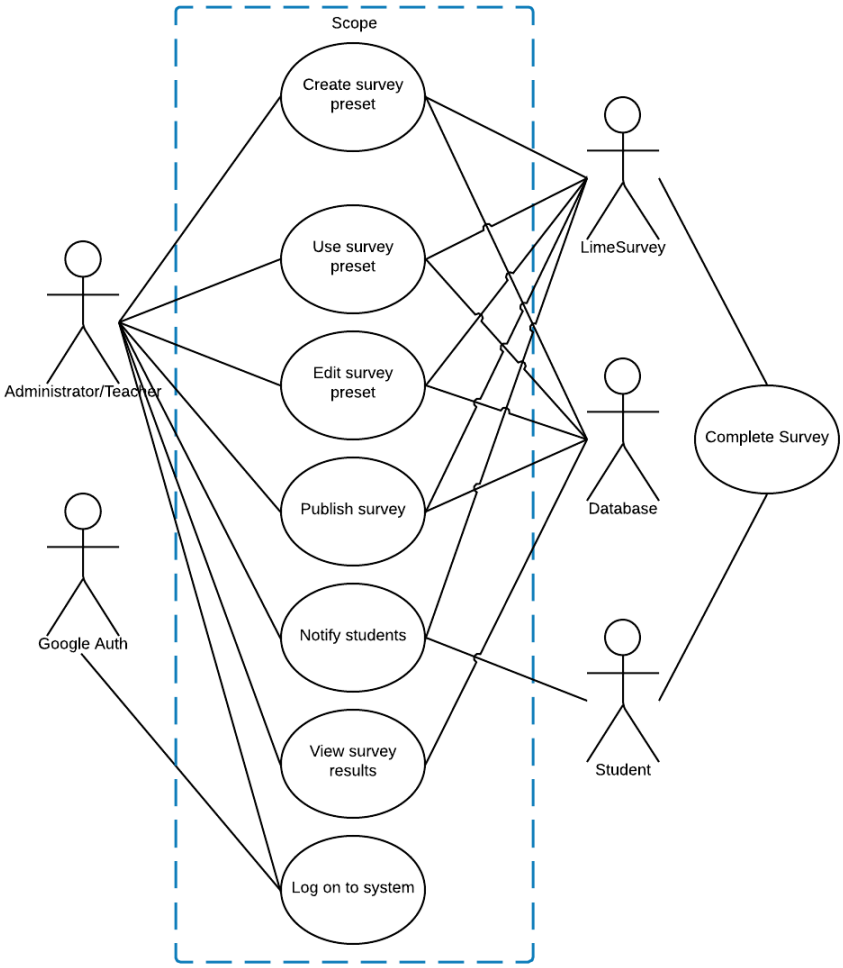

Figure 1: Use Case diagram of the system scope.

2 Functional Requirements

The functional requirements specify what actions the program will perform. Each requirement is represented as a use case. The UML diagram showing the use cases is on the next page.

Table 1

Number	1
Name	Log on to system
Summary	A teacher or administrator logs in with a username and password
Priority	5
Preconditions	The user has an Google account (or can otherwise sign in with Google)
Postconditions	A session is started and an authentication token is sent to the program
Primary Actors	Administrator, Teacher
Secondary Actors	Google Authentication
Trigger	A user attempts to log in
Main Scenario	**Step** | **Action** 1 | Open the log-in page 2 | Enter Google log-in information 3 | User is authenticated 4 | User gains access to the system
Extensions	**Step** | **Branching Action** 2a | Incorrect log-in info : Do not proceed and alert user
Open Issues	What method to use for authentication of non-anonymous comments by students

Table 2

Number	2
Name	Create survey preset
Summary	An administrator or teacher adds a preset list of questions to the database
Priority	5
Preconditions	A course's evaluation survey questions are unchosen or unwritten
Postconditions	The course's evaluation survey is written
Primary Actors	Administrator, Teacher
Secondary Actors	Database, LimeSurvey
Trigger	A new course needs to be evaluated
Main Scenario	**Step** | **Action** 1 | Select option to add a preset 2 | Enter the survey data on LimeSurvey 3 | Database updated with survey info
Extensions	**Step** | **Branching Action** 2a | Invalid data is entered : Notify user of incorrect data
Open Issues	What information about the survey shall be stored, what format should the data be in

Table 3

Number	3
Name	Use survey preset
Summary	An administrator or teacher retrieves a preset for a survey developed previously for a course
Priority	4
Preconditions	The survey preset exists in the database
Postconditions	A preset list is available to create a survey
Primary Actors	Administrator, Teacher
Secondary Actors	Database, LimeSurvey
Trigger	A user wants the same questions as on a previous survey
Main Scenario	**Step** / **Action** 1 / Select option to use a preset list of questions 2 / Choose which preset to use 3 / Preset data from database converted to LimeSurvey form 4 / LimeSurvey updated with preset
Extensions	**Step** / **Branching Action** 2a / Data in preset is invalid : Notify user of invalid data
Open Issues	Which presets are visible to each teacher

Table 4

Number	4
Name	Edit survey preset
Summary	An administrator or teacher edits the survey questions for a course
Priority	5
Preconditions	A course for the survey questions exists in the database
Postconditions	A course includes appropriate survey data
Primary Actors	Administrator, Teacher
Secondary Actors	Database, LimeSurvey
Trigger	A college course survey has incorrect data or preferences
Main Scenario	**Step** / **Action** 1 / Choose the course whose questions are to be edited 2 / Select questions preset if applicable 3 / Modify the data to be appropriate for the course 4 / Database updated with new edits
Extensions	**Step** / **Branching Action** 3a / Invalid data is entered : Notify user of incorrect data
Open Issues	None

Table 5

Number	5
Name	Publish survey
Summary	LimeSurvey or the evaluation software is used to send out a survey to students
Priority	5
Preconditions	The chosen class has questions entered
Postconditions	A survey is created using LimeSurvey and sent to the students on the class roster
Primary Actors	Administrator, Teacher
Secondary Actors	Database, LimeSurvey
Trigger	A teacher or administrator wants to send out an evaluation
Main Scenario	**Step** / **Action** 1 / Select option to publish survey 2 / Select course(s) 3 / A survey is created using LimeSurvey 4 / The survey is sent to all students on the class roster
Open Issues	None

Table 6

Number	6
Name	Notify students
Summary	LimeSurvey or the evaluation software is used to e-mail students to complete the survey, giving up to 3 automatic reminders as appropriate
Priority	5
Preconditions	A course evaluation survey is completed
Postconditions	All students are finished with their survey, or a deadline is reached
Primary Actors	Teacher
Secondary Actors	Student, LimeSurvey
Trigger	The teacher releases the survey to students
Main Scenario	**Step** / **Action** 1 / Send initial reminder to students about survey using course roster 2 / Students complete survey 3 / Copy LimeSurvey results to database 4 / Database deletes data that is 60 days old
Extensions	**Step** / **Branching Action** 2a / A student does not complete survey : Send another reminder to student
Open Issues	How often to remind students, how to handle signed comments

Table 7

Number	7
Name	View survey results
Summary	An administrator, teacher, or both receive an automated e-mail, which has a link to view or download the survey results (i.e. averages and charts)
Priority	4
Preconditions	Survey responses are entered in database
Postconditions	All appropriate averages are computed and displayed
Primary Actors	Administrator, Teacher
Secondary Actors	Database
Trigger	A user seeks information about the responses to a group of surveys
Main Scenario	**Step** / **Action** 1 / Click link to view results 2 / Compute average score for each question 3 / Display results by instructor, department, and university 4 / Select option to download results
Open Issues	Which survey questions are averaged for display

2.1 Tests

These are the tests that will verify the functional requirements:

1. (Use Case 1) The tester attempts to log on to the system using four different sets of credentials: a valid e-mail address and password of an existing account (with courses already created), a valid address and password of a nonexistent account, an address with an invalid password for an existing account, and an invalid e-mail address and password. The test is successful if the first set passes and the last three sets fail. In the second case, a user will be prompted to register for the system.

2. (Use Case 2) The tester selects the option to add a course preset, then submits evaluation questions, preferences, the course roll, and e-mail texts. The test is successful only if all the data entered is reflected in the database. The data should still be there after logging out of the system. The team will create several courses/sections of the same course to ensure that the data loaded into LimeSurvey remains unchanged from when the original course was loaded. We will test adding students' full names and e-mails to the list of enrolled students.

3. (Use Case 3) The tester creates a course survey. He or she then tries to select that same survey to be used again. The new preset should still be present on LimeSurvey. The test is successful only if the preset data is loaded correctly to LimeSurvey from the database. The tester will exit the evaluation system and log in again to ensure that the data is saved.

4. (Use Case 4) The tester selects the option to create a survey. Next, he or she chooses a questions preset for the survey. The tester then adds one question and edits an existing question from the preset. After submitting, the database should include the new survey, containing the preset list, added question, and edited question. The team will run the program several times, each time selecting different questions to be included in the survey. We will ensure that some questions are mandatory and must be included no matter what.

5. (Use Case 5) The tester creates a course survey, does not enter any questions, and attempts to send the survey. The tester then enters some questions and sends the survey again. The team verifies that the survey is created with all questions entered and that the e-mail addresses on the course roster receive a prompt to complete it.

6. (Use Case 6) The tester messages a group of ten unique people about a mock survey. In a successful test, all ten people receive an e-mail with each containing a link to the survey. Most respondents promptly complete the survey, but one does not. The database must contain the survey responses of the nine people who finished, and the program must send another reminder e-mail to the unfinished person about the survey. Finally, the database must delete old survey data. The team will set the time periods before the reminder e-mail and data deletion to be very short to test whether the two functions work.

7. (Use Case 7) The tester selects the option to view the survey results. There are survey results using mock data for (a) each course section, (b) all sections of the same course, (c) all courses of each instructor, (d) all courses with the same designator, (e) all courses in the same department/school, (f) all courses in the same college, and (g) all courses in the same university. The team will add several surveys, each with different questions. If the test is successful, the correct average scores for every question in the course evaluation survey are shown in the application. Bar graphs for each course and section are supplied as well. Also, the software must show the appropriate results for all seven levels listed above.

3 Non-Functional Requirements

The non-functional requirements state the qualities of the program that are unrelated to its function.

Table 8

Number	1
Priority	3
Description	The software should be supported by the latest versions of Windows, Mac, Linux, iOS, and Android.
Test Number(s)	1

Table 9

Number	2
Priority	4
Description	The software should be accessible by the latest versions of Safari, Chrome, Firefox, and Edge.
Test Number(s)	2

Table 10

Number	3
Priority	5
Description	All questions entered by the teacher or administrator shall appear on the output survey.
Test Number(s)	3

Table 11

Number	4
Priority	5
Description	All data stored in the program's database shall be valid.
Test Number(s)	4

Table 12

Number	5
Priority	5
Description	All collected survey data shall not be alterable.
Test Number(s)	5

Table 13

Number	6
Priority	4
Description	Teachers shall not be able to access data of courses other than their own.
Test Number(s)	6

Table 14

Number	7
Priority	3
Description	The mean time between failures should be at least 60 minutes.
Test Number(s)	7

Table 15

Number	8
Priority	5
Description	Students shall have no access to any data stored by the program.
Test Number(s)	8

Table 16

Number	9
Priority	5
Description	All survey responses (except signed comments) shall be anonymous.
Test Number(s)	9

Table 17

Number	10
Priority	2
Description	The software should scale to at least three universites, 1000 courses per semester, 1000 teachers per university, and 500 students per course.
Test Number(s)	10

Table 18

Number	11
Priority	1
Description	The software should not exceed 500 MB in size.
Test Number(s)	11

Table 19

Number	12
Priority	4
Description	The software's source code shall be open-source and shall use a GPLv2 license.
Test Number(s)	12

Table 20

Number	13
Priority	4
Description	The licensing requirements of any non-original code shall be met.
Test Number(s)	13

Table 21

Number	14
Priority	4
Description	The software shall meet UMaine AFUM requirements.
Test Number(s)	13

3.1 Tests

These are the tests that will verify the non-functional requirements:

1. The tester attempts to run the program on the last versions of Windows, Mac, Linux, iOS, and Android. The test is successful only if all five use cases execute properly on each operating system.

2. The tester attempts to run the program on the last versions of Safari, Chrome, Firefox, and Edge. The test is successful only if all five use cases execute properly on each browser.

3. The tester inputs five diverse questions into a mock survey and sends the survey to a mock student. The test is successful only if the mock student can follow the link to a survey showing those five questions.

4. The tester inputs a course, questions preset, and survey. The test is successful only if all data for the course, preset, and survey are present in the database, and if that data correctly follows the database schema.

5. The tester inputs the same course, preset, and survey as in test 4. The test is successful only if any function in the software does not change the survey responses stored in the database.

6. The tester creates three users for the evaluation program. Each user inputs a different course, questions preset, and survey. The test is successful if the users cannot view nor modify each other's data.

7. The tester runs an automatic script that simulates several hours of using the software. The test is successful only if the average time between two errors in the software is at least 60 simulated minutes.

8. The tester inputs a course, present, and survey as a teacher, then poses as someone unauthorized to use the software. In a successful test, the tester cannot view nor modify any data present in the database.

9. Five testers complete a survey sent by the software, and a sixth tester uses the software afterwards. The test is successful only if the sixth tester cannot find the identities of anyone who gave a response meant to be anonymous.

10. The tester runs an automatic script that sets up the software with three colleges, 1000 courses per college, 1000 teachers per university, and 500 students per course. The test is successful if all five use cases execute properly for all courses.

11. The tester sees the info for the folder containing the code. The folder size is at most 500 megabytes if the test passes.

12. Ten computer users attempt to access the source code on GitHub. The test is successful if all users can do so.

13. The tester reads the GNU General Public License and University of Maine AFUM requirements. The test is successful if the software meets all terms in both the license and AFUM requirements.

4 User Interface

See "User Interface Design Document for the College Course Evaluation System."

5 Deliverables

The following lists the estimated date and format that each submission will be delivered:

Table 22

Submission	**Date of Delivery**	**Format**
System Requirements Specification	10/29/2018	Hard Copy
System Design Document	11/16/2018	Hard Copy
User Interface Design Document	11/30/2018	Hard Copy
Administrator Manual	April 2019	Hard Copy
User Manual	May 2019	Hard Copy
System Requirements Specification	May 2019	Electronic
System Design Document	May 2019	Electronic
User Interface Design Document	May 2019	Electronic
Administrator Manual	May 2019	Electronic
User Manual	May 2019	Electronic
Source Code	May 2019	Electronic
Web Link to Program	May 2019	Electronic

6 Open Issues

The most significant issue our team has relates to the non-anonymous student responses. In a course evaluation survey, a student has the option to add a signed comment to be stored in an instructor's file. However, the LimeSurvey software does not store the identities of survey respondents. We need to find a way to collect the signed comment along with the survey, while ensuring that the signature is authentic.

There are several more minor issues that need to be resolved. We have not decided what information about each course should be stored in the database. We do not know what format to use for storing the survey data. We should also know which question presets and which surveys' results are visible to each user. Another issue is how often to notify students about their survey. Finally, we need to know how to authenticate a student who wants to submit a non-anonymous response.

The team projects that all of the above issues will be resolved by the end of November.

A Agreement Between Customer and Contractor

This page shows that all members of Team EVAL and the client, Harlan Onsrud, have agreed on all the information in the system requirements specification. By signing this document, Team EVAL and Dr. Onsrud agree on the goals and scope of the project, each use case and requirement, the tests for each requirement, and how the deliverables will be sent.

The team will follow a process in the case that the requirements specification is changed after we sign it. First, the team writes a rough draft of the changes to be made to the document. Second, all team members and Harlan Onsrud will sign the document agreeing to the changes. Finally, the changes are made to the final copy of the specification.

Name	***Signature***	***Date***
Jovon Craig	_____	_____
Sam Elliott	_____	_____
Robert Judkins	_____	_____
Stanley Small	_____	_____
Harlan Onsrud	_____	_____

Customer Comments: _____

_____

B Team Review Sign-off

This page shows that all members of Team EVAL have reviewed the system requirements specification and agreed on its content. By signing this document, the team members agree on the goals and scope of the project, each use case and requirement, the tests for each requirement, and how the deliverables will be sent. There is nothing in the document that is a source of contention.

Name	***Signature***	***Date***
Jovon Craig	_____	_____
Comments:	_____	
Sam Elliott	_____	_____
Comments:	_____	
Robert Judkins	_____	_____
Comments:	_____	
Stanley Small	_____	_____
Comments:	_____	

C Document Contributions

Stanley Small created the document, added the system requirements specification template to the document, and formatted the text to be compatible with LaTeX. He added the log-in use case to the requirements and the team logo to the table of contents. He was also an active participant in discussing and writing down requirements during meetings with the client. Stan contributed approximately 20 percent of the document.

Jovon Craig wrote five of the use cases in the functional requirements, all non-functional requirements, and a draft of the requirement tests. He also wrote the purpose of the document, the purpose of the product, the deliverables section, and the three appendices. Jovon contributed about 55 percent of the document.

Sam Elliott worded the non-functional requirements, wrote the section explaining the product scope, and created the use case diagram that describes the system. He also added the "publish survey" use case and two tests for the functional requirements. Sam contributed about 20 percent of the document.

Robert Judkins made revisions to the tests for the functional requirements. He contributed about 5 percent of the document.