Systems Requirements Specification

Student Symposium Management Software (SSMS)

UMaine Student Symposium (UMSS)

Student Symposium Software (SSS)

Rhiannon Gould

Ryan Shanz

Sophie Walden

Nick Pease

Cole Adams

Version 1

10/11/24

Table of Contents

1. Introduction

Provide a one-paragraph statement describing the project. This is a capstone project for *whom*, in partial fulfillment of the Computer Science BS degree for the University of Maine.

Remember, every major section should have a section introduction.

Student Symposium Management Software (SSMS) is a tracking and automation product to make many of the repetitive tasks of hosting the yearly UMaine Student Symposium (UMSS) efficient and painless. The goal of SSMS includes allowing students to submit their abstracts and then be organized into 10 different categories where they will be assigned abstract numbers. Judges will also be allowed to submit their application, and then based on a set of rules they will be assigned to judge two abstract numbers. Next, during the symposium, judges must be able to submit their scores that will be used to quickly determine an undergraduate and graduate winner for each of the 10 abstract categories. The goal of the project aims to create software that will streamline this process making it a more hands off experience for the UMSS team, allowing them more time to dedicate to improving symposium experience and activities.

1.1 Purpose of This Document

This document is intended to be used by the client, UMSS, the capstone instructor Laura Gurney, and the SSS team developing this product. It serves the UMSS as a detailed and informative outline of product features that are going to be developed. This will be achieved through descriptions of requirements, including functional, non-functional, and user interface requirements alongside proposed testing procedures. It serves the SSS team as a reference for the purpose and requirements we need to fill to appease the UMSS.

1.2. References

Provide a list of all applicable and referenced documents and other media (e.g., the Somerville text, UML references, documents provided by the customer, websites). For each reference, provide the title, author, publisher (if applicable), date, and URL (for websites).

List of majors/categories**,** UMSS, 2024, https://umaine.edu/umss/list-of-majors-categories/

Book of Abstract throughout the years, UMSS, 2019-2024,
https://umaine.edu/umss/book-of-abstracts-by-year/

Judging Sign up form, UMSS, 2024

Judging Scoring form, UMSS, 2024

2025 UMaine Student symposium: Call for abstracts, UMSS, 2024

1.3. Purpose of the Product

This section provides a short description of the UMSS's work and the situation that triggered the need for the product. It describes the task(s) that the UMSS wants to accomplish with the delivered product. It is the product justification.

When the UMSS is running the student symposium they have a small team that has an exorbitant amount of time dedicated to organizing the abstract submissions and judging. They feel that much of this is repetitive and tedious processes done annually because the Book of Abstracts, judging assignments and category winners are created and calculated manually. The UMSS states that if they had these processes automated they would be able to expend their valuable time and resources towards other non-repetitive tasks that improve the symposium. They expressed that they wished they had the time to include meeting with sponsors and working on perks (e.g. headshot booth, food trucks) to expand the benefits and experience for the symposium attendees.

2. Functional Requirements

The functional requirements starts with a set of user stories in 2.1 and then include an overview of all functional requirements in Section 2.2. The following tables 2.3.1 through 2.3.11 encapsulate the use cases for each requirement.

2.1 User stories

As a Symposium employee, I want a display of the applicant demographics so that I can easily understand the demographics of the event.

As a Symposium employee, I want to determine the highest undergraduate student and graduate student score so that I can award the category winner.

As a student applicant, I want to submit my abstract through Google Forms so I can receive an abstract number.

As a judge, I want to submit my application, so that I can receive an automated email with information about the abstract I will be judging.

As a judge, I want to be able to submit scores, so that the abstracts can be fairly judged.

As a student applicant, I want to be able to pick a topic category so I can be nearby similar projects.

2.2 Use Case Diagrams

This section displays Use case diagrams that encapsulate both the user stories and the use cases themselves.

Figure 2.2.1
Use case diagrams for use case ID:4 and ID:5.

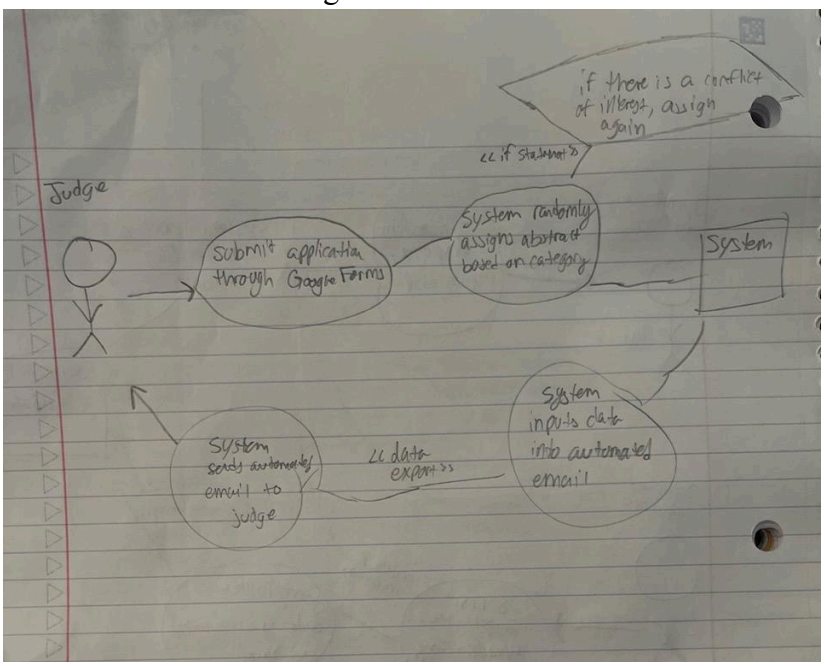


Figure 2.2.2
Use case diagram for use case ID:6 and ID:7.

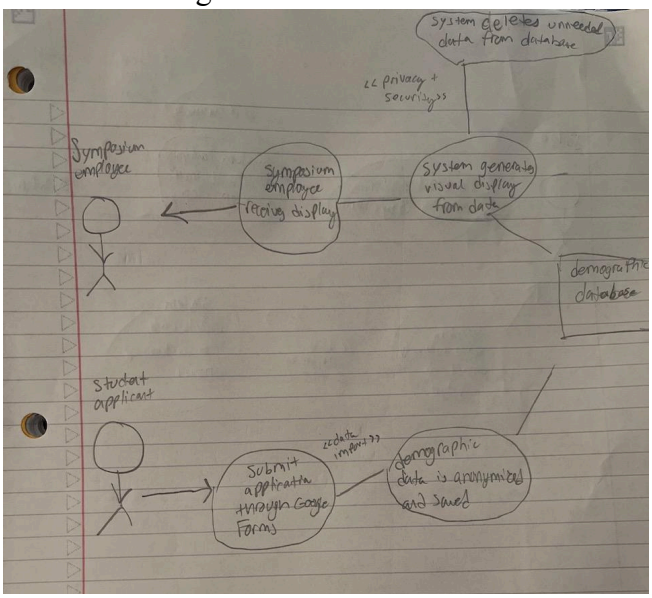
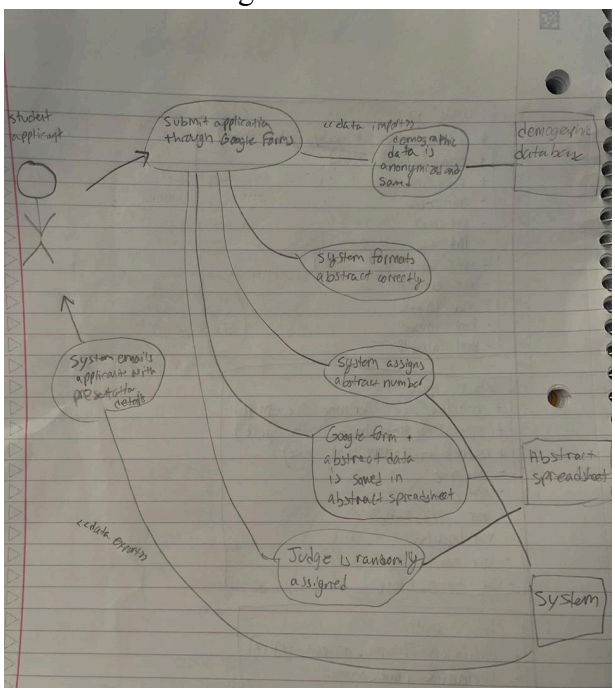


Figure 2.2.3

Use case diagram for use case ID:1.



2.2 Overview of functional requirements

A brief description of all functional requirements and ID number that is assigned to each unique requirement.

Table 2.2

Overview of functional requirements

ID	Description
1	The system must allow users to submit applications through Google Forms
2	The system must sort submissions into 10 different categories
3	The system must assign abstract numbers to each user in a category
4	The system must randomly assign 2 judges per abstract
5	The system must send an email to each judge with their assigned abstract numbers
6	The system must have a database of demographic information
7	The system must plot and display demographic information of judges, submissions and student attendees

8	The system must notify student submissions of their abstract number through email
9	The system must clean the submitted abstracts and check for formatting errors.
10	The system must collect 2 scores from each judge through Google Forms
11	The system must determine one undergraduate and one graduate with the highest score within each category

Section 2.3 Use Cases

This section covers use cases for each functional requirement.

Table 2.3.1
Functional Requirement 1

Number	#1	
Name	The system must allow users to submit applications through Google Forms	
Summary	The Student Symposium currently accepts abstract and judging applications through Google Forms. They want to maintain their current workflow as much as possible, and do not want to change the submission process for applicants. Therefore, we must accept applications in Google Form format.	
Priority	5	
Preconditions	The system shall be able to accept and save Google Forms.	
Postconditions	The Google Forms will be saved and its data will be condensed into a spreadsheet.	
Primary Actor	Applicant	
Secondary Actors	Symposium Employees	
Trigger	Application submission	
Main Scenario	Step	Action
	1	An Applicant submits a Google Form
	2	The system saves the Form
	3	The system runs a script to automate its transference to a spreadsheet
Extensions	Step	Branching Action
	1a	<Applicant data is saved from Google Form> <Data can now be saved in database for demographic records>
Open Issues	the specific details of the script that will transfer the data	

Table 2.3.2
Functional Requirement 2

Number	2	
Name	The system must sort submissions into 10 different categories.	
Summary	Abstracts are sorted into various categories depending on their subject matter. Our system must accurately sort the received abstracts into their appropriate categories. The 10 categories are Allied Health, Arts, Biomedical Sciences, Business, Education, Engineering and Information Sciences, Interdisciplinary, Natural Sciences, Physical and Mathematical Sciences, Social Sciences and Humanities.	
Priority	4	
Preconditions	The system must have received an abstract via Google Forms	
Postconditions	The submitted abstract will be properly sorted into the spreadsheet based on the chosen categories.	
Primary Actor	Applicant	
Secondary Actors	Symposium Employee	
Trigger	Submission of abstract using Google Forms	
Main Scenario	Step	Action
	1	Applicant selects a category from a predefined list of options
	2	Applicant submits Abstract via Google Forms
	3	System receives and saves Form
	4	System identifies the selected category
	5	System transfers category to appropriate cell in spreadsheet
Extensions	Step	Branching Action
	1a	< Abstracts are assigned categories > : < Abstracts can now be assigned appropriate judges >
Open Issues	Details of script used to automate data transfer, Details of spreadsheet layout	

Table 2.3.3
Functional Requirement 3

Number	3	
Name	The system must assign abstract numbers to each user in a category.	
Summary	Abstract numbers are selected based on two factors. The hundreds digit of the three digit number is selected based on the Abstract's chosen category. The tens and ones digits are chosen based on submission order. For example, a person who is the first person to submit an Abstract in the Computer Science category will have an Abstract number of 201.	
Priority	4	
Preconditions	Applicants will have chosen a specific category number based on content matter.	
Postconditions	Abstracts will be sorted and assigned Abstract numbers	
Primary Actor	Applicant	
Secondary Actors	Symposium Employee	
Trigger	submission of Abstract via Google Forms.	
Main Scenario	Step	Action

	1	Applicant selects a category on the application
	2	Applicant submits the application via Google Forms
	3	System receives and saves the application
	4	System assigns number based on predefined categories and submission order
Extensions	Step	Branching Action
	1a	Assigning Abstract number : notifying Applicant of their Abstract Number
Open Issues	Specifying the layout of the spreadsheet	

Table 2.3.4

Functional Requirement 4

Number	4	
Name	The system must randomly assign 2 judges per abstract.	
Summary	Each abstract is judged by 2 judges. The judges are assigned randomly, but have some constraints. For example, professors must not judge their students. Our system must take these requirements into account and assign judges based on them.	
Priority	5	
Preconditions	The judging Applicant has submitted their application	
Postconditions	Accepted judges will be assigned abstracts to judge without breaching the predefined judging rules.	
Primary Actor	Judging Applicant	
Secondary Actors	Spreadsheet data regarding abstract numbers and abstract categories	
Trigger	The Judging Applicant submits their application to be a judge.	
Main Scenario	Step	Action
	1	Judging Applicant submits their application through Google Forms
	2	System receives and saves the submission.
	3	System randomly assigns the judge a Presenter in the appropriate category
	4	System checks to see if there is a conflict of interest
	5a	If yes, it randomly assigns the judge a new Presenter, then checks again
	5b	If no, the judge will judge that Presenter
	6	System records the judge for the selected Abstract, and ensures each Abstract has two judges.
Extensions	Step	Branching Action
	1a	< Judges are assigned > : < automated email can be sent out to judges with information about their abstract >
Open Issues	Specifying the logic/data of the program that is used to determine if judges are allowed to judge a specific Presenter Specifying the logic used to ensure each Abstract has two and only two judges	

Table 2.3.5

Functional Requirement 5

Number	5	
Name	The system must send an email to each judge with their assigned abstract numbers.	
Summary	The system must notify judging applicants with the details of the presentation that they will judge.	
Priority	4	
Preconditions	The system will assign a judge to a specific abstract	
Postconditions	The judge will be notified of the number of the abstract they will be judging	
Primary Actor	Judging Applicant	
Secondary Actors	Spreadsheet data with abstract numbers	
Trigger	The Judge is assigned an abstract number	
Main Scenario	Step	Action
	1	The judge is assigned an abstract in an appropriate category
	2	The number of that abstract is assigned to the judge
	3	The automated email is sent to the judge with the abstract number
Extensions	Step	Branching Action
	1a	No branching action - no additional or sub use cases are derived fro, this requirement
Open Issues	The contents of the automated email	

Table 2.3.6

Functional Requirement 6

Number	6	
Name	The system must have a database of demographic information	
Summary	To ensure judges are not assigned presenters with a conflict of interest, we must gather data on judges and presenters.	
Priority	3	
Preconditions	The presenter and judging applicant must submit their applications.	
Postconditions	The system will be able to assign judges fairly	
Primary Actor	Applicant (presenter or judge)	
Secondary Actors	Submitted applications	
Trigger	< the action that starts the use case >	
Main Scenario	Step	Action
	1	Applicants submit their application
	2	Demographic information from the application is saved to a database
Extensions	Step	Branching Action
	1a	< Database saves demographic information > : < Use Case #7 can begin plotting data >
Open Issues	Setting up database infrastructure, deleting the personal data once it is no longer needed	

Table 2.3.7

Functional Requirement 7

Number	7	
Name	The system must plot and display demographic information of judges, submissions and student attendees.	
Summary	To gather information about judges, submissions, and student attendees (presenters), the system will take the demographic information discussed in Functional Requirement 6 and present it in a visual fashion	
Priority	3	
Preconditions	Demographic data must be saved in a database	
Postconditions	Demographic data will be easily accessible in displayed form	
Primary Actor	Demographic data	
Secondary Actors	Database with the Demographic data	
Trigger	The deadline for submission has passed and all possible demographic data has been collected	
Main Scenario	Step	Action
	1	The system retrieves demographic data from database
	2	The system uses graphing tools to plot and display the data
Extensions	Step	Branching Action
	1a	< Demographic data is anonymously plotted > : < Demographic data can be deleted to meet privacy and security regulations >
Open Issues	Setting up the database to retrieve demographic data, deleting the personal data once it is no longer needed	

Table 2.3.8

Functional Requirement 8

Number	8	
Name	The system must notify student submissions of their abstract number through email.	
Summary	The system must notify applicants with the details of their abstract presentation. The client is interested in doing so via an automated email.	
Priority	4	
Preconditions	The applicant must submit their application. The abstract must be assigned an abstract number.	
Postconditions	Applicants will be notified of their abstract numbers	
Primary Actor	Applicant	
Secondary Actors	Spreadsheet data with abstract numbers	
Trigger	The Abstract is assigned an abstract number	
Main Scenario	Step	Action
	1	Abstract is assigned an abstract number
	2	Automated email is composed with details of abstract number
	3	Automated email is sent.
Extensions	Step	Branching Action
	1a	< If the applicant needs to change something in the email details > : < The applicant replies to the email with necessary changes >

Open Issues	The contents of the automated email

Table 2.3.9
Functional Requirement 9

Number	9	
Name	The system must clean the submitted abstracts and check for formatting errors.	
Summary	All abstracts have to ensure that scientific names are italicized and font size is uniform.	
Priority	3	
Preconditions	Abstract has been submitted	
Postconditions	Abstract can be entered into the book of abstracts.	
Primary Actor	Applicant	
Secondary Actors	Symposium employee	
Trigger	Abstract submission	
Main Scenario	Step	Action
	1	Abstract is submitted by applicant
	2	Abstract is checked for scientific names
	3	Scientific names are italicized
	4	Font size is changed to 12pt
Extensions	Step	Branching Action
	1a	< Abstracts are formatted correctly > : < Abstracts are ready to be added to the Book of Abstracts >
Open Issues	Creating the book of abstracts	

Table 2.3.10
Functional Requirement 10

Number	10	
Name	The system must collect 2 scores from each judge through Google Forms.	
Summary	For each abstract assigned to a judge, after scoring the abstract they must submit a score.	
Priority	5	
Preconditions	Judges have applied and are assigned abstract numbers	
Postconditions	Category winners can be determined	
Primary Actor	Judge applicant	
Secondary Actors	Symposium employee	
Trigger	Judges are assigned to judge an abstract	
Main Scenario	Step	Action
	1	Judges receive their abstract numbers
	2	Judges score their abstract
	3	Judges submit their abstract through a Google Form
Extensions	Step	Branching Action
	1a	< Scores are recorded through Google Forms > :

		< System begins comparing scores to determine a winner >
Open Issues	Determining winners for each of the 10 categories	

Table 2.3.11
Functional Requirement 11

Number	11	
Name	The system must determine one undergraduate and one graduate with the highest score within each category.	
Summary	Each of the 10 categories, after judges score abstracts, will have 1 undergraduate and one graduate student with the highest score. Those with the highest scores will be deemed the winners.	
Priority	5	
Preconditions	Judges have submitted all abstract scores	
Postconditions	Winners are determined	
Primary Actor	Abstract applicant	
Secondary Actors	Judge applicant	
Trigger	Submissions of scores by judge applicant	
Main Scenario	Step	Action
	1	Judge submits abstract scores
	2	Category scores are sorted into graduate and undergraduate.
	3	Maximum undergraduate and graduate scores are found
	4	A list of winners, for each category is produced
Extensions	Step	Branching Action
	1a	< List of winners is determined > : < Winner data is recorded in spreadsheet for data retention >
Open Issues	Display of list of winners	

2.5. Testing

Lastly, write the tests that will be used during system and acceptance testing to verify that each requirement has been met. Note that a single requirement may require multiple tests, so be thorough. It is also possible that a single test verifies more than one requirement. The goal is to come up with the minimum number of test cases that thoroughly test the system. Make sure that the test numbers correspond to the use case numbers.

Test: (use case number)
Test Desc:
Preconditions:
Test Steps:
Expected Results:
Actual Results:
Pass/Fail Criteria:
Assumptions:

Test: 1
Test Desc: Verify users can submit an application through google forms
Preconditions: User must properly fill out each field of the form
Test Steps:
1. User clicks link to google form
2. Fill out each form field appropriately
3. Click submit

Expected Results: The form is expected to submit the response to an organized spreadsheet
Actual Results: TBD
Pass/Fail Criteria: It's a pass if the form organizes and submits the data to the spreadsheet
Assumptions: User has a google/maine.edu account, a computer, and internet access.

Test: 2
Test Desc: Verify the data passed from the form is organized appropriately on the spreadsheet
Preconditions: Test 1 passes
Test Steps:
1. Submit application from form
2. Check spreadsheet to verify data has been sorted properly

Expected Results: Application data is sorted in rows and columns based on requirements for application
Actual Results: TBD
Pass/Fail Criteria: Pass if all data is correctly sorted in designated row/columns
Assumptions: Users submit valid applications in the correct format

Test: 3
Test Desc: All data collected and demographic displays created should be accessible for all previous years the software has been run on the Symposium
Preconditions: A previous years symposium has run
Test Steps:
1: Navigate to the associated spreadsheet tab for the previous year
2: Navigate to the demographic display for that tab

Expected Results: All data from that year is sorted and displayed inside of the spreadsheet tab
Actual Results: TBD
Pass/Fail Criteria: Pass if all data is still stored for multiple different years
Assumptions: New spreadsheet tab is created based on the year

Test: 4
Test Desc: Verify each abstract gets two randomly assigned judges
Preconditions: There are judges
Test Steps:
1. Verify there are enough judges
2. Run script to randomly assign 2 judges to each application
3. Verify that there are 2 judges per application

Expected Results: Each application will get 2 randomly assigned judges with no conflicts of interest
Actual Results: TBD

Pass/Fail Criteria: Each submission gets 2 judges with no conflicts of interest
Assumptions: There is abstract submissions and judges

Test: (use case number) 5
Test Desc: Verify each judge receives an email confirming their role and what abstract numbers they will be judging
Preconditions: Each submission has an abstract number, Previous tests pass
Test Steps:
1. Run script to send appropriate emails
2. Verify email contents

Expected Results: Each judge receives an email with which abstracts they will be judging
Actual Results: TBD
Pass/Fail Criteria: Pass if each judge receives an email with which abstracts they will be judging
Assumptions: Judges have email, Judges check email

Test: 6
Test Desc: Verify there is a database of demographic information
Preconditions: There is demographic information
Test Steps:
1. Verify demographic information is being stored in a database

Expected Results: Demographic information is being collected and stored in a database
Actual Results: TBD
Pass/Fail Criteria: Pass is demographic information is being appropriately collected, organized, and stored in a database
Assumptions: There is a database

Test: 7
Test Desc: Verify demographic information of judges, abstract, and student attendees can be plotted and displayed
Preconditions: The user is logged into the system and there is existing demographic data for judges, submissions, and student attendees in the database.
Test Steps:
1. Navigate to the demographic database
2. Observe plotted data for judges, abstracts, and student attendees

Expected Results: The system should display clear plotted data of judges, attendees, and abstracts
Actual Results: TBD
Pass/Fail Criteria: Pass is the system displays clear plotted data of judges, attendees, and abstracts

Test: 8
Test Desc: Verify that the system sends an email notification to students with their abstract number upon submission.
Preconditions: The user is registered in the system. The student has successfully submitted their abstract.

Test Steps:
1. Submit a new abstract.
2. Monitor the email inbox associated with the student's registered email address.
3. Check for the receipt of an email notification after submission.

Expected Results: Student receives an email upon submission with their abstract number
Actual Results: TBD
Pass/Fail Criteria: Pass if the student receives an email upon submission with their abstract number
Assumptions: The students email is functional

Test: 9
Test Desc: Verify that the system cleans submitted abstracts and checks for formatting errors.
Preconditions: The submission has formatting errors
Test Steps:
1. Submit abstract with formatting errors
2. Verify errors have been cleaned and formatting errors fixed on spreadsheet

Expected Results: System cleans and properly formats submissions
Actual Results: TBD
Pass/Fail Criteria: Pass if the system cleans and properly formats submissions
Assumptions: The system has defined rules for formatting errors

Test: 10
Test Desc: Verify that the system successfully collects two scores from each judge using Google Forms.
Preconditions: Judges have access to form, form is set up correctly
Test Steps:
1. Open google form for scoring
2. Submit test scores
3. Verify scores have been submitted on spreadsheet

Expected Results: Submitted scores get record on a spreadsheet
Actual Results:TBD
Pass/Fail Criteria: Pass if submitted scores get recorded on a spreadsheet
Assumptions: Judges know how to use google forms

Test: 11
Test Desc: Verify the system determines one undergraduate and one graduate with the highest score within each category.
Preconditions: Each abstract gets scored
Test Steps:
1. Run script to return abstracts with the highest score in each category
2. Verify results are accurate

Expected Results: System returns accurate results for each category
Actual Results: TBD
Pass/Fail Criteria: Pass if system returns accurate results for each category
Assumptions: Judges give accurate and appropriate scores

3. Non-Functional Requirements

The following non-functional requirements are a list of certain metrics we want our system to hit to be ready to deliver to UMSS. Each NFR has an individual ID and the ID of the test that it is associated with.

Product Requirements:

Id: 1
Priority: 5
Description: The system shall be able to take in 500 submissions without a single submission being dropped or not added to the associated spreadsheet
Associated Test: 1

Id: 3
Priority: 4
Description: The system shall be able to assign two judges for any given submitted submission with up to 500 submissions and 100 judges.
Associated Test: 4

Id: 9
Priority: 1
Description: The system shall verify up to 500 abstracts and 100 judges for no malformed submissions affecting the demographic displays
Associated Test: 7

Id: 10
Priority: 3
Description: The system shall have an associated tab or sheet to verify steps of the symposium such as submissions in, judging assigned, and scoring finished
Associated Test: 2,4,6

Id: 11
Priority: 3
Description: The system shall be able to assign abstract numbers to up to 500 submissions accurately
Testing: 8

Id: 17
Priority: 4
Description: The system shall be able to sort scores by different criteria, such as undergraduate or graduate on up to 500 scored submissions.
Testing: 11

Id: 16
Priority: 3

Description: The system should be able to grade up to 500 submissions based on up to 1000 judge submissions and create a total ranking of all the abstracts submitted
Testing: 11

Id: 19
Priority: 2
Description: The system should have multiple checks to verify the grading of up to 500 submissions is accurate to traditional UMSS grading standards
Testing: 11

Id: 20
Priority: 2
Description: The system shall be able to find problematic abstracts in a group of 500 in less then a minute of runtime
Testing: 9

Id: 27
Priority: 2
Description: The system shall be able to assign abstract numbers to up to 100 judges adhering to judging restrictions
Testing: 5

External Requirements:

Id: 5
Priority: 1
Description: The system shall not allow access to submission data to anyone outside of the UMSS or those purposefully shared with.
Associated Test: 1

Id: 6
Priority: 4
Description: The system shall email up to 100 judges simultaneously confirming their role and abstract numbers to judge.
Associated Test: 5

Id: 12
Priority: 3
Description: The system shall be able to email up to 500 students about their abstract submissions assigning correct individual abstract numbers
Testing: 8

Id: 13
Priority: 2

Description: The system shall be able to diagnose errors in a batch of 500 submitted abstracts with up to 95% accuracy
Testing: 9

Id: 14
Priority: 1
Description: The system shall be able to email students about incorrect or incomplete abstract submissions for up to 500 submissions
Testing: 9

Id: 15
Priority: 4
Description: The system shall be able to handle up to 1000 external judging submissions, 2 for each of submitted abstracts.
Testing: 10

Id: 26
Priority: 3
Description: The system shall be able to sort and display proper score rankings in all 10 categories
Testing: 11

Id: 28
Priority: 2
Description: The system shall be able to properly verify the submission of up to 1000 judging forms to track live how many abstracts still need to be judged
Testing: 10

Id: 29
Priority: 1
Description: the system shall be able to verify that the 500 abstracts were assigned randomly among judges the right criteria by running tests on mock data
Testing: 4

Id: 30
Priority: 1
Description: The system shall be able to display demographics on any sorted feature of the 500 abstract submissions.
Testing: 7

Organizational Requirements:

Id: 2
Priority: 2
Description: The system shall be able to be access all collected data in demographic displays for every past year

Associated Test: 3

Id: 4
Priority: 2
Description: The system shall handle up to 500 submissions for future years to come without needing to manually modify any processes.
Associated Test: 1

Id: 7
Priority: 3
Description: The system should handle demographics and data displays for up to 500 submissions
Associated Test: 7

Id: 8
Priority: 2
Description: The system should format up to 500 submissions in a consistent manner to be able plug into demographic displays
Associated Test: 2

Id: 18
Priority: 1
Description: The system should have demographic data created for up to 100 judges for all years present
Testing: 3

Id: 21
Priority: 3
Description: The system shall be able to properly format up to 500 submissions for the book of abstracts
Testing: 9

Id: 22
Priority: 1
Description: The system shall be able to properly sort up to 500 submissions based on custom fields from the input form
Testing: 2

Id: 23
Priority:
Description: The system shall have a verification step for sending up to 100 emails for detailing judging of up to 500 unique abstract numbers
Testing: 5

Id: 24

Priority: 1
Description: The system must verify that there are enough judges to safely and fairly apply judging for up to 500 submissions
Testing: 4

Id: 25
Priority: 2
Description: The system should be able to fix up to 500 submissions incase of malformed abstract submission within the system.
Testing: 9

4. User Interface Requirements

This section covers the specified user requirements that were negotiated with the client. It plans to address design choices, and plans to address usability, accessibility and inclusion within SMSS.

This software requires integration with the customers pre-existing software pipeline. This requires that submission of abstracts and demographic information be via the Google Forms interface. This addresses concerns about familiarity of the user interface and ease of use for students using the program to enter the symposium. With a pre-existing process in place, we will create our program to output a book of abstracts in the same format as previous years. All logic from the Google Form until the finished product will be run entirely on the Google ecosystem, largely in the background. Users should only see front-facing Google products such as Google Sheets, Google Forms, and any generated documents in Google Drive or Gmail.

Requirements
Google Forms API, Google Sheets API, Google Drive API, Gmail API