Software Requirements Specification for Software Engineering: MES Finance Tracking Platform

Team #5, Money Making Mauraders
Zhenia Sigayev
Justin Ho
Thomas Wang
Michael Shi
Johnny Qu

October 8, 2025

Contents

Revision History

Date	Version	Notes
Date 1	1.0	Notes
Date 2	1.1	Notes

1 Purpose of the Project

1.1 User Business

1.2 Goals of the Project

- **Goal 1:** Brief explanation of goal 1. Add more goals.

2 System Overview

2.1 Architecture Diagram

Figure 1: TODO: ADD CAPTION FOR ARCHITECTURE DIAGRAM. This should be a general overview of the flow of information and how features interact with each other. No technologies etc should be included here.

2.2 Technology Stack Overview

Figure 2: TODO: ADD CAPTION FOR TECH STACK OVERVIEW. This should be a general overview of the technologies used in the system and how they interact with each other. This should show how the frontend and backend interact with each other and any external services used.

3 Stakeholders

3.1 McMaster Engineering Society (MES)

3.2 Direct Stakeholders

3.2.1 MES Club Leadership

Describe in more detail and bullet points.

10 Non-Functional Requirements

10.1 Non-Functional Requirements

- Example

11 Look and Feel Requirements

11.1 Appearance Requirements

- Example

11.2 Style Requirements

- Example

12 Usability and Humanity Requirements

12.1 Ease of Use Requirements

- Example

12.2 Personalization and Internationalization Requirements

- Example

12.3 Learning Requirements

- Example

12.4 Understandability and Politeness Requirements

- Example

12.5 Accessibility Requirements

- Example

13 Performance Requirements

13.1 Speed and Latency Requirements

- Example

13.2 Safety-Critical Requirements

- Example

13.3 Precision or Accuracy Requirements

- Example

13.4 Robustness or Fault-Tolerance Requirements

- Example

13.5 Capacity Requirements

- Example

13.6 Scalability or Extensibility Requirements

- Example

13.7 Longevity Requirements

- Example

14 Operational and Environmental Requirements

14.1 Expected Physical Environment

- Example

14.2 Wider Environment Requirements

- Example

14.3 Requirements for Interfacing with Adjacent Systems

- Example

14.4 Productization Requirements

- Example

14.5 Release Requirements

- Example

15 Maintainability and Support Requirements

15.1 Maintenance Requirements

- Example

15.2 Supportability Requirements

- Example

15.3 Adaptability Requirements

- Example

16 Security Requirements

16.1 Access Requirements

- Example

16.2 Integrity Requirements

- Example

16.3 Privacy Requirements

- Example

16.4 Audit Requirements

- Example

16.5 Immunity Requirements

- Example

17 Cultural Requirements

17.1 Cultural Requirements

- Example

18 Compliance Requirements

18.1 Legal Requirements

- Example

18.2 Standards Compliance Requirements

- Example

19 Open Issues

- Example

20 Off-the-Shelf Solutions

20.1 Ready-Made Products

- Example

20.2 Reusable Components

- Example

20.3 Products That Can Be Copied

- Example

21 Likely Changes

22 Unlikely Changes

23 Ideas for Solution

24 Requirements Traceability

The purpose of the traceability matrices is to provide easy references on what has to be additionally modified if a certain component is changed. Every time a component is changed, the items in the column of that component that are marked with an "X" may have to be modified as well. Table 1 shows the dependencies of theoretical models, general definitions, data definitions, and instance models with each other. Table 2 shows the dependencies of instance models, requirements, and data constraints on each other. Table 3 shows the dependencies of theoretical models, general definitions, data definitions, instance models, and likely changes on the assumptions.

[You will have to modify these tables for your problem. —TPLT]

[The traceability matrix is not generally symmetric. If GD1 uses A1, that means that GD1's derivation or presentation requires invocation of A1. A1 does not use GD1. A1 is "used by" GD1. —TPLT]

[The traceability matrix is challenging to maintain manually. Please do your best. In the future tools (like Drasil) will make this much easier. —TPLT]

	TM??	TM??	TM??	GD??	GD??	DD??	DD??	DD??	DD??	IM??	IM??	IM?
TM??												
TM??			X									
TM??												
GD??												
GD??	X											
DD??				X								
DD??				X								
DD??												
DD??								X				
IM??					X	X	X				X	
IM??					X		X		X	X		
IM??		X										
IM??		X	X				X	X	X		X	

Table 1: Traceability Matrix Showing the Connections Between Items of Different Sections

The purpose of the traceability graphs is also to provide easy references on what has to be additionally modified if a certain component is changed. The arrows in the graphs represent

	IM??	IM??	IM??	IM??	??	R??	R??
IM??		X				X	X
IM??	X			X		X	X
IM??						X	X
IM??		X				X	X
R??							
R??						X	
R??					X		
R??	X	X				X	X
R??	X						
R??		X					
R??			X				
R??				X			
R??			X	X			
R??		X					
R??		X					

Table 2: Traceability Matrix Showing the Connections Between Requirements and Instance Models

	A??	A??	A??	A??	A??	A??	A??	A??	A??	A??	A??	A??	A??	A??	A??	A??	A??	A??	A??
TM??	X																		
TM??																			
TM??																			
GD??		X																	
GD??			X	X	X	X													
DD??							X	X	X										
DD??			X	X						X									
DD??																			
DD??																			
IM??											X	X		X	X	X			X
IM??												X	X			X	X	X	
IM??														X					X
IM??													X					X	
LC??				X															
LC??								X											
LC??									X										
LC??											X								
LC??												X							
LC??															X				

Table 3: Traceability Matrix Showing the Connections Between Assumptions and Other Items

dependencies. The component at the tail of an arrow is depended on by the component at the head of that arrow. Therefore, if a component is changed, the components that it points to should also be changed. Figure **??** shows the dependencies of theoretical models, general definitions, data definitions, instance models, likely changes, and assumptions on each other. Figure **??** shows the dependencies of instance models, requirements, and data constraints on each other.

25 Development Plan

[This section is optional. It is used to explain the plan for developing the software. In particular, this section gives a list of the order in which the requirements will be implemented. In the context of a course this is where you can indicate which requirements will be implemented as part of the course, and which will be "faked" as future work. This section can be organized as a prioritized list of requirements, or it could should the requirements that will be implemented for "phase 1", "phase 2", etc. —TPLT]

26 Values of Auxiliary Constants

[Show the values of the symbolic parameters introduced in the report. —TPLT]

[The definition of the requirements will likely call for SYMBOLIC_CONSTANTS. Their values are defined in this section for easy maintenance. —TPLT]

[The value of FRACTION, for the Maintainability NFR would be given here. —TPLT]

References

Appendix — Reflection

26.1 What went well while writing this deliverable?

- Zhenia Sigayev
- Justin Ho
- Thomas Wang
- Michael Shi
- Johnny Qu

26.2 How many of your requirements were inspired by speaking to your client(s) or their proxies (e.g. your peers, stakeholders, potential users)?

- Zhenia Sigayev
- Justin Ho
- Thomas Wang
- Michael Shi
- Johnny Qu

26.3 What knowledge and skills will the team collectively need to acquire to successfully complete this capstone project? This includes domain specific knowledge or software engineering and/or computer science knowledge. Skills may be related to technology, or writing, or presentation, or team management, etc. You should look to identify at least one item for each team member.

- Zhenia Sigayev
- Justin Ho
- Thomas Wang
- Michael Shi
- Johnny Qu

Appendix — References

[1] Example," Example Institution, https://example.ca/doc/ (accessed Mon. day, year).

The purpose of reflection questions is to give you a chance to assess your own learning and that of your group as a whole, and to find ways to improve in the future. Reflection is an important part of the learning process. Reflection is also an essential component of a successful software development process.

Reflections are most interesting and useful when they're honest, even if the stories they tell are imperfect. You will be marked based on your depth of thought and analysis, and not based on the content of the reflections themselves. Thus, for full marks we encourage you to answer openly and honestly and to avoid simply writing "what you think the evaluator wants to hear."

Please answer the following questions. Some questions can be answered on the team level, but where appropriate, each team member should write their own response:

1. What went well while writing this deliverable?

2. What pain points did you experience during this deliverable, and how did you resolve them?

3. How many of your requirements were inspired by speaking to your client(s) or their proxies (e.g. your peers, stakeholders, potential users)?

4. Which of the courses you have taken, or are currently taking, will help your team to be successful with your capstone project.

5. What knowledge and skills will the team collectively need to acquire to successfully complete this capstone project? Examples of possible knowledge to acquire include domain specific knowledge from the domain of your application, or software engineering knowledge, mechatronics knowledge or computer science knowledge. Skills may be related to technology, or writing, or presentation, or team management, etc. You should look to identify at least one item for each team member.

6. For each of the knowledge areas and skills identified in the previous question, what are at least two approaches to acquiring the knowledge or mastering the skill? Of the identified approaches, which will each team member pursue, and why did they make this choice?