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.

3.2.2 Administrative Staff

3.2.3 MES Club Members

3.2.4 MES Student Developers

3.3 Indirect Stakeholders

3.3.1 Faculty Advisors

3.4 Personas

3.5 Priorities Assigned to Stakeholders

3.6 Stakeholder Participation

4 Mandated Constraints

4.1 Solution Constraints

1. The system must be capable of managing expense reimbursement requests for up to 61 clubs or 7000 students.
2. The system must track and display the real-time status of each reimbursement request (e.g., submitted, under review, approved, rejected, reimbursed) to ensure transparency for submitters, reviewers, and administrators.
3. The system must allow users to attach digital receipts in PDF or image formats to their reimbursement requests.
4. The system must comply with data privacy regulations, ensuring that all personal and financial information is securely stored and transmitted.
5. The system must be able to import existing financial data from the current MES finance tracker to ensure continuity and data integrity.
6. The system must maintain consistent in accordance with the overall MES platform availability to support uninterrupted access.

4.2 Current MES Finance Tracker

- **MC X:**
    - **description:** The current MES finance tracker is not scalable and lacks proper traceability features.
    - **rationale:** The new system must be designed to integrate with existing MES infrastructure and address current limitations.

4.3 Cost Constraints

- **MC X:**
  - **description:** The amount spent by a team should not exceed $500.
  - **rationale:** The project must meet capstone budget constraints and be economically feasible for MES to support and maintain.

4.4 Off-the-Shelf Software

- **MC X:**
  - **description:** There are other financial tracking options (e.g., Quickbooks), but they may not match MES look-and-feel or budget.
  - **rationale:** Ensures the chosen solution integrates seamlessly with MES UX and licensing constraints.

4.5 Schedule Constraints

- **MC X:**
  - **description:** The project must be completed by March 2026.
  - **rationale:** Allows deployment one month before term end to avoid an end-of-term submission surge.

4.6 Workflow Constraints

- **MC X:**
  - **description:** Capstone group will collaborate with MES developers using staging branches and incremental merges.
  - **rationale:** Minimizes disruption and enables gradual adoption.

4.7 Enterprise Constraints

- **MC X:**
  - **description:** Development must use technologies with favourable licensing terms for non-profits.
  - **rationale:** Avoids licensing costs or obligations that could hinder MES future plans.

5 Terminology, Acronyms, and Technologies

5.1 Terminology

- **Audit Log**: A chronological record of all changes made to the system, including who made the change and when it was made.
- **Budget Allocation**: The total amount of funds allocated to a club or team for a specific period, typically a fiscal year.
- **Expense Claim**: A request for reimbursement submitted by a club or team member for expenses incurred on behalf of the club or team.
- **Receipt**: A digital or physical document that serves as proof of purchase for an expense.
- **Reimbursement**: The process of repaying a club or team member for approved expenses.
- **Reviewer**: An MES staff member or administrator responsible for reviewing and approving or rejecting expense claims.
- **Submitter**: A club or team member who submits an expense claim for reimbursement.
- **MES Administrator**: An MES staff member with elevated permissions to manage the finance tracking platform, including user roles and system settings.

5.2 Acronyms

- **MES:** McMaster Engineering Society.
  - **description:** The student-led organization representing engineering students at McMaster University.
- **SLA:** Service Level Agreement.
  - **description:** A formal agreement between a service provider and a client that outlines the expected level of service.
- **CI:** Content Integration.
  - **description:** Content integration refers to validating the correctness of the software changes, typically running unit or integration tests.
- **CD:** Content Deployment.
  - **description:** Content deployment refers to deploying those changes to your product / platform (e.g., a website) for users to interact with.

- **CTA:** Call-to-Action.
    - **description:** An element of a web-page (e.g., a button or link) that encourages users to take a specific action.
- **UX:** User Experience.
    - **description:** The overall experience a user has when interacting with a product, including usability and design.
- **UI:** User Interface.
    - **description:** The visual and interactive elements users interact with (buttons, menus, forms).
- **OCR:** Optical Character Recognition.
    - **description:** Technology that converts images of text into editable and searchable data.
- **WCAG:** Web Content Accessibility Guidelines.
    - **description:** Guidelines from W3C to make web content more accessible to people with disabilities.

5.3 Technologies

- **Front-end:** The client-side part of the application that users interact with directly, typically built using HTML, CSS, and JavaScript frameworks.
- **Back-end:** The server-side part of the application that handles business logic, database interactions, and authentication, often built using languages like Python, Ruby, or Node.js.
- **Full-Stack:** A combination of both front-end and back-end development, where a developer is proficient in working on all layers of the application.
- **Testing:** The process of evaluating the functionality and performance of the application to ensure it meets the specified requirements and is free of defects.
- **Database:** A structured collection of data that is stored and managed to facilitate efficient retrieval and manipulation.
- **Cloud Services:** Online platforms that provide computing resources, storage, and services over the internet, such as AWS, Azure, or Google Cloud.
- **Version Control:** A system that tracks changes to code and allows multiple developers to collaborate on a project, with Git being the most popular version control system.

6 Relevant Facts And Assumptions

6.1 Relevant Facts

6.2 Business Rules

6.3 Assumptions

7 The Scope of the Work

7.1 The Current Situation

7.2 The Context of the Work

7.3 Work Partitioning

7.4 Specifying a Business Use Case (BUC)

8 Business Data Model and Data Dictionary

8.1 Business Data Model

8.2 Data Dictionary

9 Functional Requirements

9.1 Functional Requirements

- The system shall allow MES clubs and teams to submit expense claims.
- The system shall provide MES reviewers with tools to efficiently review, approve, or reject the reimbursement requests.
- The system shall track the status of each expense claim (e.g., submitted, under review, approved, rejected, reimbursed).
- The system shall permanently store and retain digital receipt submissions.
- The system shall maintain an audit trail that records who submitted, reviewed, approved, or denied each expense claim.
- The system shall enable access to club expense submissions to the submitters, and other club members with a role greater or equal to that of the submitter, or MES administrators and approvers.

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 and Learning Requirements

- A straightforward optional overview or tutorial should be provided to first-time users.

12.2 Personalization and Internationalization Requirements

- The platform will be in English only.

12.3 Understandability and Politeness Requirements

- The platform will not use offensive language, imagery, symbols, or media of any kind.
- All errors and warnings will be communicated in a clear, concise, and polite manner.

12.4 Accessibility Requirements

- Front-end navigation and interactions must comply with WCAG standards to ensure accessibility for users with disabilities.
- The platform will allow users to select between light and dark mode themes.
- The front-end will use semantic tags (¡header¿, ¡main¿, ¡nav¿, ¡footer¿, ¡article¿, ¡button¿, etc.) for structure.
- All images must have meaningful alternative text (alt).
- All functionality must be navigable from a keyboard (i.e., tab to move through content).

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

- Third-party OCR software specializing in receipt scanning can be integrated. For example, Amazon's Textract.
- There exist reusable React.js components for the front-end that are proided by the MES to conform to thier look and feel requirements.

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
  - TODO.
- Justin Ho
  - TODO.
- Thomas Wang
  - TODO.
- Michael Shi
  - TODO.
- Johnny Qu
  - TODO.

26.2 How many of your requirements were inspired by speaking to your clients or their proxies?

- TODO.

26.3 What knowledge and skills will the team collectively need to acquire to successfully complete this capstone project?

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

26.4 What parts of this deliverable has each team member contributed to?

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

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?