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 10, 2025

Contents

Revision History

Date	Version	Notes
Date 1	1.0	Notes
Date 2	1.1	Notes

1 Purpose of the Project - Thomas

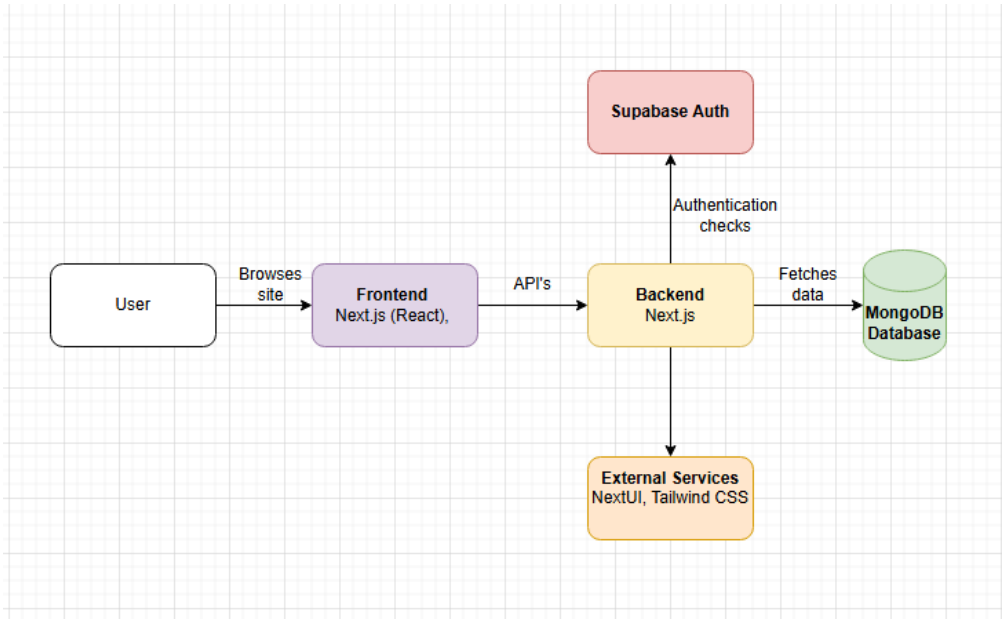
1.1 User Business - Thomas

1.2 Goals of the Project - Thomas

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

2 System Overview

2.1 Architecture Diagram



2.2 Technology Stack Overview

The system is a web-based application built with modern JavaScript frameworks and libraries, designed to support the needs of the MacEng Society website. The application provides user-facing web pages and administrative features for content management. It is structured as a client-server application where the frontend runs in a web browser and the backend interacts with a database.

2.3 Frontend (Client-Side)

- Developed using **React/Next.js**.
- Runs in modern web browsers such as Google Chrome, Microsoft Edge, Firefox, and Safari.

- Provides the user interface for interacting with content such as council information, events, and bookings.
- Fetches and displays data from the backend via RESTful or GraphQL APIs.

2.4 Backend (Server-Side)

- Executes on the **Node.js** runtime environment.
- Manages application logic, routing, and communication with the database.
- Uses **MongoDB Community Server** as the primary data storage system.
- Exposes APIs consumed by the frontend for dynamic content delivery.

2.5 Database

- Implemented using **MongoDB Community Server**.
- Stores member information, content, and other structured data required for the application.

2.6 Development Environment

- Source code is managed using **Git** and hosted in a repository such as GitHub.
- Developers use **Visual Studio Code (VSCode)** with mandatory extensions:
  - ESLint
  - Prettier
- Project dependencies are managed with **pnpm** or **yarn**.
- **Husky** and linting hooks enforce branch naming conventions, commit message standards, and code consistency.

2.7 External Dependencies

- **GitHub Actions** (via `.github/workflows`) for CI/CD and automated testing.
- **NextUI** and **TailwindCSS** for UI components and styling.
- Additional **Node.js** ecosystem libraries and packages installed via pnpm or yarn.

3 Stakeholders - Thomas

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 - Thomas

3.5 Priorities Assigned to Stakeholders - Thomas

3.6 Stakeholder Participation - Thomas

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

There exist a set of relevant facts related to the project that are not considered hard constraints or goals. They have been collected below.

- The MES has existing information about clubs stored within the MES monorepo. This data exists in a MongoDB database instance in non-relational, document form.
  - Their current applications restrict access based on role according to the data stored in this database.
- Internal documentation is stored in a seperate repo @ https://github.com/McMaster-Engineering-Society/Documentation (Private Repository)
- The MES has existing development practices. For the purposes of this project, they are generally considered *suggestions*.
  - A Kanban board has been setup for MES stakeholders to track our progress.
  - Current testing standards consist of informally performing manual end-to-end tests. If proof is required, video captures of testing are standard.
  - Work will be based on a staging branch in the monorepo specifically for our project.
  - Pre-existing styled react front-end components exist, and we may use them. We are additionally allowed to add functionality, though may not remove or change existing functionality.

6.2 Business rules

6.3 Assumptions

7 The Scope of the Work

7.1 The Current Situation

The MES currently uses spreadsheets and manual processes to track club finances. This approach results in a large amount of work for MES financial staff due to lost receipts, duplicated claims, or other manual errors. A centralized financial tracking platform is needed to streamline and automate this process for the MES.

7.2 The Context of the Work

The MES Information Technology (IT) team currently has systems and data related to clubs associated with their organization. The project however, will not directly interface with most of the other systems. The work will be conducted within the MES's monorepo, on an individual staging branch, and production databases and code will be owned by the MES.

7.3 Work Partitioning

Work on the project is to be partitioned such that development can proceed in an efficient manner while maintaining traceability and accountability for project supervisors. As the project involves a client body (the MES), requirement gathering and communications are included as part of the work.

- **Initial Requirements Gathering and Communications**
  - **Collection of initial information:** Directly elicit initial requirements, situation context, and problem context from the client body. Details about desired project output, existing systems, as well as expectations in terms of client supervision/input during project work.
  - **Formal requirement and planning documentation:** Create various documents, including this one, detailing the formal requirements, situation information, and development plan for the project.
- **Proof of Concept (PoC) Development**
  - **Local Wireframe Full-stack App Development:** Create a local implementation including all major system components to be required for the project. Capability of developing the full system will be demonstrated.

  - **Receipt Scanning Proof of Concept:** Develop a robust system for extracting finacial data from receipts as a PoC for other physical mediums of financial data. Testing and related documentation will be required to prove system effectiveness.
  - **User Authentication Proof of Concept:** Develop a system to authenticate users and restrict access to system data depending on user role.
  - **Integration with MES Monorepo:** Integration of minimum PoC with existing MES monorepo to prove ability to fulfill project under ownership of the MES.
  - **PoC Demonstration:** Demonstrate PoC to client and supervisors. Collect feedback and iterate development plans for fulfillment of the rest of project goals and functional requirements.

- **Integration and Requirement Fulfillment**
  - **Full Development Plan:** Create plan and work delegation for fulfillment of rest of functional requirements and goals of the project.
  - **Fulfillment Development:** Fulfill requirements according to plan through software development and iteration, including maintaining iterative communication with client and supervisors.

In addition to the mentioned work, communication with the client and testing of the application will be maintained during the entire development process.

7.4 Specifying a Business Use Case (BUC) - Michael

8 Business Data Model and Data Dictionary - Michael

8.1 Business Data Model - Michael

8.2 Data Dictionary - Michael

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

The following 7 sections outline the non-functional requirements for the MES Finance Tracking Platform.

11 Look and Feel Requirements

11.1 Appearance Requirements

- The interface should use the McMaster Engineering Society's branding colors and logo.
- The design should maintain a clean, modern layout using NextUI and TailwindCSS for consistency.
- The application should use responsive design principles to adapt to desktop, tablet, and mobile displays.
- Buttons, icons, and navigation components should follow accessible contrast ratios (WCAG 2.1 AA).

11.2 Style Requirements

- Typography should match MES branding guidelines, using sans-serif fonts for readability.
- UI components must adhere to consistent padding, spacing, and rounded-corner styles.
- Notification banners and status indicators (e.g., "Submitted," "Pending," "Approved") should use color cues for clarity.
- The application should provide a visually appealing balance between white space and content density.

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

- API response times must average below 1 second for standard database queries and data submissions.
- Page load times should not exceed 3 seconds under normal operating conditions on modern browsers.
- File uploads (e.g., receipts or invoices) must complete within 5 seconds for files under 10 MB.

13.2 Safety-Critical Requirements

- The system must ensure that no reimbursement or financial transaction is executed without MES reviewer approval.
- All monetary data must be validated before submission to prevent corruption or duplication.

13.3 Precision or Accuracy Requirements

- Financial transactions must maintain accuracy up to two decimal places.
- All calculations involving funding allocations or reimbursements must be precise and consistent with MES accounting standards.

13.4 Robustness or Fault-Tolerance Requirements

- The system should handle unexpected backend or network failures gracefully, displaying meaningful error messages to users.
- Data must be saved incrementally to prevent loss during unexpected session timeouts or browser crashes.
- In the event of a failed API request, the system should automatically retry up to three times before reporting failure.

13.5 Capacity Requirements

- The system must support at least 500 concurrent users without noticeable degradation in performance.
- The database must handle storage for at least 10,000 user records and 100,000 expense claims.

13.6 Scalability or Extensibility Requirements

- The system must be designed to allow horizontal scaling (e.g., deploying additional backend instances during high usage).
- New features such as additional modules (e.g., club budgeting or analytics dashboards) should be easily integrable without major architectural changes.

13.7 Longevity Requirements

- The system is expected to remain operational and maintainable for at least 5 years with regular updates.
- Future technology stack upgrades (React, Node.js, MongoDB versions) should not require a full system rewrite.

14 Operational and Environmental Requirements

14.1 Expected Physical Environment

- The system will be accessed through desktop and laptop computers using modern browsers (Chrome, Edge, Firefox, Safari).
- Users may also access the site via tablets or smartphones with stable internet connectivity.

14.2 Wider Environment Requirements

- The application will be hosted on cloud infrastructure (e.g., Vercel or AWS) with automatic scaling and uptime monitoring.
- The system must operate effectively under typical university network conditions, including campus Wi-Fi and VPN connections.

14.3 Requirements for Interfacing with Adjacent Systems

- The system may integrate with McMaster's Single Sign-On (SSO) or authentication system in the future.
- Potential interfaces include email APIs (e.g., SendGrid) for notifications and payment APIs for reimbursement processing.

14.4 Productization Requirements

- The application should be packaged for deployment using Docker containers to simplify installation and environment configuration.
- Versioning of releases must follow semantic versioning conventions (e.g., v1.0.0).

14.5 Release Requirements

- New versions of the application must be deployed through GitHub Actions CI/CD pipelines.
- Each release must undergo code review and automated testing before deployment to production.

15 Maintainability and Support Requirements

15.1 Maintenance Requirements

- The codebase must adhere to consistent formatting and linting rules enforced by ESLint and Prettier.
- All major system dependencies must be updated at least once per academic year to maintain compatibility and security.

15.2 Supportability Requirements

- The system should include detailed developer documentation covering setup, deployment, and troubleshooting steps.
- MES technical staff or designated maintainers should have access to logs and monitoring dashboards for issue diagnosis.

15.3 Adaptability Requirements

- The application should support easy modification to accommodate policy or procedural changes in MES financial processes.
- The UI should allow for additional modules or new forms without major rework to existing code.

16 Security Requirements

16.1 Access Requirements

- Only authenticated users may access the system.
- Role-based access control (RBAC) must be implemented to distinguish between club members, reviewers, and administrators.

16.2 Integrity Requirements

- All form submissions must be validated both client-side and server-side to prevent tampering.
- Database operations must be protected against injection and cross-site scripting (XSS) attacks.

16.3 Privacy Requirements

- Personal information such as names and emails must be stored securely in compliance with McMaster University privacy policies.
- Sensitive financial data must be encrypted at rest and in transit using TLS 1.2 or higher.

16.4 Audit Requirements

- Every key action (claim submission, approval, rejection, fund disbursement) must be logged with timestamp, user ID, and action details.
- Audit logs must be retained for a minimum of 3 years for accountability and financial review.

16.5 Immunity Requirements

- The system must be protected from common web vulnerabilities (e.g., SQL injection, CSRF, XSS).
- Rate limiting must be enforced to prevent denial-of-service (DoS) attacks.

17 Cultural Requirements

17.1 Cultural Requirements

- The system must use inclusive and gender-neutral language throughout all interfaces.
- The design should align with the professional and academic tone of McMaster University and the Engineering Society.

18 Compliance Requirements

18.1 Legal Requirements

- The system must comply with Canadian privacy laws (PIPEDA) and McMaster University's data governance standards.
- All financial data must adhere to internal MES and university auditing regulations.

18.2 Standards Compliance Requirements

- The software must conform to standard web development best practices (HTML5, CSS3, ECMAScript 6+).
- Accessibility and usability must meet WCAG 2.1 AA compliance.
- Version control, branching, and commit message standards must comply with MES development guidelines.

19 Open Issues - Zhenia

- Example

20 Off-the-Shelf Solutions

20.1 Ready-Made Products - Zhenia

- 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 - Zhenia

- Example

21 Likely Changes - Zhenia

22 Unlikely Changes - Zhenia

23 Ideas for Solution - Zhenia

24 Requirements Traceability - Michael (both matrix tables)

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

	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

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 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.

Appendix — Reflection

24.1 What went well while writing this deliverable?

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

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

- TODO.

24.3 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.

24.4 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?