

Software Requirements Specification for Software Eng

4G06: subtitle describing software

Team 2, Parnas' Pals

William Lee

Jared Bentvelsen

Bassel Rezkalla

Yuvraj Randhawa

Dimitri Tsampiras

Matthew McCracken

October 5, 2022

Contents

1	Introduction	iv
1.1	Purpose of Document	iv
1.2	Scope of Requirements	iv
2	General System Description	iv
2.1	System Context	v
2.2	System Constraints	vi
2.3	System Scope	vi
3	Stakeholders	vi
4	Functional Requirements	vi
5	Nonfunctional Requirements	vi
5.1	Look and Feel Requirements	vii
5.2	Usability and Humanity Requirements	vii
5.3	Performance Requirements	vii
5.4	Operational and Environmental Requirements	vii
5.5	Maintainability and Support Requirements	vii
5.6	Security Requirements	vii
5.7	Cultural Requirements	vii
5.8	Compliance Requirements	vii
6	Use cases	viii
6.1	Use case Diagram	viii
7	Likely Changes	viii
8	Unlikely Changes	viii
9	Traceability Matrices and Graphs	viii
10	Off the Shelf Solutions	xi
11	Risks	xi
12	Reference Material	xi
12.1	Abbreviations and Acronyms	xi

Revision History

Date	Version	Notes
Date 1	1.0	Notes
Date 2	1.1	Notes

1 The Purpose of the Project

1.1 The User Business or Background of the Project Effort

1.2 Goals of the Project

2 Project Drivers

2.1 The Purpose of the Project

2.2 Stakeholders

1. Fitness Enthusiasts - Anyone interested in exploring other fitness routines, creating their own routines, and tracking their own personal progression towards goals.
2. Personal Trainers - Olympian provides the ideal platform for trainers to share routines and goals with their clients.
3. Fitness Advertisers - One avenue of monetization that Olympian could take is running advertisements. Although these advertisements could fall into any category, the largest stakeholders will be Fitness Advertisers, as the users of Olympian will be heavily involved with fitness, and thus most likely to buy fitness products.

3 Project Constraints

3.1 Mandated Constraints

3.2 Naming Conventions and Terminology

3.3 Relevant Facts and Assumptions

4 General System Description

This section provides general information about the system. It identifies the interfaces between the system and its environment, describes the user characteristics and lists the system constraints. [This text can likely be borrowed verbatim. —TPLT]

[The purpose of this section is to provide general information about the system so the specific requirements in the next section will be easier to understand. The general system description section is designed to be changeable independent of changes to the functional requirements documented in the specific system description. The general system description provides a context for a family of related models. The general description can stay the same, while specific details are changed between family members. —TPLT]

4.1 System Context

[Your system context will include a figure that shows the abstract view of the software. Often in a scientific context, the program can be viewed abstractly following the design pattern of Inputs \rightarrow Calculations \rightarrow Outputs. The system context will therefore often follow this pattern. The user provides inputs, the system does the calculations, and then provides the outputs to the user. The figure should not show all of the inputs, just an abstract view of the main categories of inputs (like material properties, geometry, etc.). Likewise, the outputs should be presented from an abstract point of view. In some cases the diagram will show other external entities, besides the user. For instance, when the software product is a library, the user will be another software program, not an actual end user. If there are system constraints that the software must work with external libraries, these libraries can also be shown on the System Context diagram. They should only be named with a specific library name if this is required by the system constraint. —TPLT]

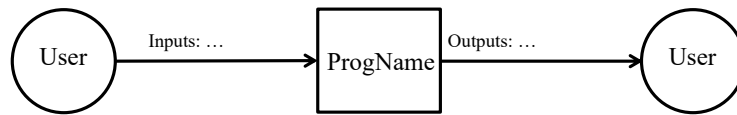


Figure 1: System Context

[For each of the entities in the system context diagram its responsibilities should be listed. Whenever possible the system should check for data quality, but for some cases the user will need to assume that responsibility. The list of responsibilities should be about the inputs and outputs only, and they should be abstract. Details should not be presented here. However, the information should not be so abstract as to just say “inputs” and “outputs”. A summarizing phrase can be used to characterize the inputs. For instance, saying “material properties” provides some information, but it stays away from the detail of listing every required properties. —TPLT]

- User Responsibilities:

-

- Software Eng 4G06 Responsibilities:

- Detect data type mismatch, such as a string of characters instead of a floating point number

-

4.2 System Constraints

[System constraints differ from other type of requirements because they limit the developers' options in the system design and they identify how the eventual system must fit into the world. This is the only place in the SRS where design decisions can be specified. That is, the quality requirement for abstraction is relaxed here. However, system constraints should only be included if they are truly required. —TPLT]

4.3 System Scope

5 Functional Requirements

5.1 The Scope of the Work

5.2 Business Data Model and Data Directory

5.3 The Scope of the Product

5.4 Functional Requirements

- R1: Description: The system shall allow the user to create a workout routine.
Rational: To allow the user to save their own workout routines.
Fit Criterion: The user created routines are accessible after creation.
- R2: Description: The system shall allow the user to add and remove individual exercises in order to a created workout routine, with a maximum of 20 exercises per workout routine.
Rational: A workout routine is composed of a sequence of exercises performed in order, which the user should be able to add and remove as they wish to create the desired routine.
Fit Criterion: The user is able to add and remove exercises from a created workout routine.
- R3: Description: The system shall allow the user to add or remove quantifiers to a given exercise.
Rational: A single exercise requires quantifiers including but not limited to sets, reps, weight (light, medium, heavy), time, rest time to adequately specify how it is to be performed within a given routine.
Fit Criterion: The user is able to add quantifiers to a given exercise.
- R4: Description: The system shall allow the user to add a “hint” section to a given exercise. A “hint” is a strictly text less than 500 characters.
Rational: The user should be able to add hints, tips, tricks, and finer workout descriptions to a given exercise.
Fit Criterion: The user is able to attach a hint to each exercise in a workout routine.

- R5: Description: The system shall store a workout routine created by a user.
Rational: To be able to store a routine to potentially share the routine with other users.
Fit Criterion: On creation of a workout routine, the data for the routine should be accessible by the system.
- R6: Description: The system shall allow the user to publicly post a workout routine.
Rational: To be able to display their workout routine to other users.
Fit Criterion: On publication of a routine, another user should be able to access and view the routine.
- R7: Description: The product shall allow a user to update a workout routine capable as making the same changes as during routine creation.
Rational: For users to be able to change quantifiers or routine properties after the creation time. Users may change their mind or make mistakes, so making updates to existing routines is an important requirement.
Fit Criterion: The user is able to update pre-existing workout routines.
- R8: Description: The system shall allow a user to save and view a performed workout.
Rational: To allow a user to keep track of their current workout, and review their previous workouts when doing them again. This is especially helpful for ensuring progressive overload. That is, doing a little bit more than last time.
Fit Criterion: The user is able to save a performed workout and review that data in the future.
- R9: Description: The product shall allow a user to browse and search for workout routines.
Rational: To make publicly posted workout routines discoverable and for users to find routines that cater their fitness goals.
Fit Criterion: The user is able to browse and search for workout routines. The returned routines contain words that the user searched for.
- R10: Description: The product should allow a user to save another user's public workout routine, such that the other user's routine is accessible from the saving user's profile.
Rational: To allow a user to save a workout routine for later use.
Fit Criterion: The user is able to save another user's workout routine to their profile.
- R11: Description: The system should allow a user to create a profile, with a username between 1 and 25 characters.
Rational: To display social, informational content to other users.
Fit Criterion: The user is able to create a profile.
- R12: Description: The system shall allow a user to edit their profile.
Rational: To update any information regarding the user profile.
Fit Criterion: The user is able to make changes to their profile after it has been created.

- R13: Description: The system shall allow a user to search for and view another user's profile.
Rational: To allow a user to determine if another user has similar fitness goals or similar workout routines.
Fit Criterion: The user is able to view the profile of another user after searching for the target profile username.
- R14: Description: The system shall allow the user to create and view a goal in the form of an exercise and a metric pair. For example, "Bench press - 100kg".
Rational: This allows the user to set and witness progression towards their fitness goals
Fit Criterion: If the user is able to create and view goals.
- R15: Description: The system shall allow the user to create progress points towards a specific goal.
Rational: Being able to track progress towards set goals can help encourage more progression until the goal is reached.
Fit Criterion: The user is able to create progress points towards affiliated with a specific goal.
- R16: Description: A progress point must be associated with a one specific goal, and must exist as a date metric pair. For example, "09/04/22 - 96kg" under a "Bench Press - 100kg" goal. The metric type must match the goal metric type, for example kg.
Rational: Progress points must be simple to enter so that the resultant data can be easily visualized.
Fit Criterion: Progress points can only exist as a date-metric pair with the metric type matching the associated goal metric type.
- R17: Description: The product shall be able to visually display fitness progress towards set fitness goals.
Rational: To help the user determine progress towards fitness goals.
Fit Criterion: The user is able to view progress points toward a set fitness goal.

[Every IM should map to at least one requirement, but not every requirement has to map to a corresponding IM. —TPLT]

6 Nonfunctional Requirements

[List your nonfunctional requirements. You may consider using a fit criterion to make them verifiable. —TPLT] [The goal is for the nonfunctional requirements to be unambiguous, abstract and verifiable. This isn't easy to show succinctly, so a good strategy may be to give a "high level" view of the requirement, but allow for the details to be covered in the Verification and Validation document. —TPLT] [An absolute requirement on a quality of the system is rarely needed. For instance, an accuracy of 0.0101 % is likely fine, even if the requirement is for 0.01 % accuracy. Therefore, the emphasis will often be more on describing how well the quality is achieved, through experimentation, and possibly theory,

rather than meeting some bar that was defined a priori. —TPLT] [You do not need an entry for correctness in your NFRs. The purpose of the SRS is to record the requirements that need to be satisfied for correctness. Any statement of correctness would just be redundant. Rather than discuss correctness, you can characterize how far away from the correct (true) solution you are allowed to be. This is discussed under accuracy. —TPLT]

6.1 Look and Feel Requirements

6.2 Usability and Humanity Requirements

6.3 Performance Requirements

6.4 Operational and Environmental Requirements

6.5 Maintainability and Support Requirements

6.6 Security Requirements

6.7 Cultural Requirements

6.8 Compliance Requirements

- NFR1: **Accuracy** [Characterize the accuracy by giving the context/use for the software. Maybe something like, “The accuracy of the computed solutions should meet the level needed for <engineering or scientific application>. The level of accuracy achieved by Software Eng 4G06 shall be described following the procedure given in Section X of the Verification and Validation Plan.” A link to the VnV plan would be a nice extra. —TPLT]
- NFR2: **Usability** [Characterize the usability by giving the context/use for the software. You should likely reference the user characteristics section. The level of usability achieved by the software shall be described following the procedure given in Section X of the Verification and Validation Plan. A link to the VnV plan would be a nice extra. —TPLT]
- NFR3: **Maintainability** [The effort required to make any of the likely changes listed for Software Eng 4G06 should be less than FRACTION of the original development time. FRACTION is then a symbolic constant that can be defined at the end of the report. —TPLT]
- NFR4: **Portability** [This NFR is easier to write than the others. The systems that Software Eng 4G06 should run on should be listed here. When possible the specific versions of the potential operating environments should be given. To make the NFR verifiable a statement could be made that the tests from a given section of the VnV plan can be successfully run on all of the possible operating environments. —TPLT]

- Other NFRs that might be discussed include verifiability, understandability and reusability.

7 Use cases

- View posted workout routine
 1. Fitness Enthusiasts - Anyone interested in exploring other fitness routines, creating their own routines, and tracking their own personal progression towards goals.
 2. Personal Trainers - Olympian provides the ideal platform for trainers to share routines and goals with their clients.
 3. Fitness Advertisers - One avenue of monetization that Olympian could take is running advertisements. Although these advertisements could fall into any category, the largest stakeholders will be Fitness Advertisers, as the users of Olympian will be heavily involved with fitness, and thus most likely to buy fitness products.

8 Project Constraints

8.1 Mandated Constraints

8.2 Naming Conventions and Terminology

8.3 Relevant Facts and Assumptions

9 General System Description

This section provides general information about the system. It identifies the interfaces between the system and its environment, describes the user characteristics and lists the system constraints. [This text can likely be borrowed verbatim. —TPLT]

[The purpose of this section is to provide general information about the system so the specific requirements in the next section will be easier to understand. The general system description section is designed to be changeable independent of changes to the functional requirements documented in the specific system description. The general system description provides a context for a family of related models. The general description can stay the same, while specific details are changed between family members. —TPLT]

9.1 System Context

[Your system context will include a figure that shows the abstract view of the software. Often in a scientific context, the program can be viewed abstractly following the design pattern of Inputs → Calculations → Outputs. The system context will therefore often follow this pattern. The user provides inputs, the system does the calculations, and then provides the outputs to the user. The figure should not show all of the inputs, just an abstract view of the main categories of inputs (like material properties, geometry, etc.). Likewise, the outputs should be presented from an abstract point of view. In some cases the diagram will show other external entities, besides the user. For instance, when the software product is a library, the user will be another software program, not an actual end user. If there are system constraints that the software must work with external libraries, these libraries can also be shown on the System Context diagram. They should only be named with a specific library name if this is required by the system constraint. —TPLT]

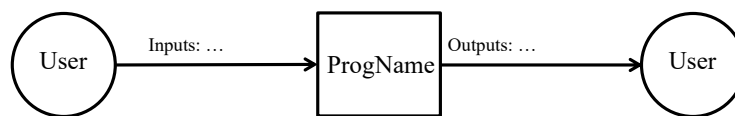


Figure 2: System Context

[For each of the entities in the system context diagram its responsibilities should be listed. Whenever possible the system should check for data quality, but for some cases the user will need to assume that responsibility. The list of responsibilities should be about the inputs and outputs only, and they should be abstract. Details should not be presented here. However, the information should not be so abstract as to just say “inputs” and “outputs”. A summarizing phrase can be used to characterize the inputs. For instance, saying “material properties” provides some information, but it stays away from the detail of listing every required properties. —TPLT]

- User Responsibilities:

-

- Software Eng 4G06 Responsibilities:

- Detect data type mismatch, such as a string of characters instead of a floating point number

-

9.2 System Constraints

[System constraints differ from other type of requirements because they limit the developers' options in the system design and they identify how the eventual system must fit into the world. This is the only place in the SRS where design decisions can be specified. That is, the quality requirement for abstraction is relaxed here. However, system constraints should only be included if they are truly required. —TPLT]

9.3 System Scope

10 Functional Requirements

10.1 The Scope of the Work

10.1.1 The Current Situation

N/A

10.1.2 The Context of the Work

N/A

10.1.3 Work Partitioning

N/A

10.2 The Scope of the Product

10.2.1 Use cases

- View posted workout routine
 1. View other user's fitness progress
 2. Add Personal Workout List
 3. View workout Author
 4. Review workout
- Browse Workout routines
 1. Filter routines
- View Another User's Profile
 1. View user's created routines
- Create User Profile

1. Setup profile description
 2. Setup attributes
- Edit User Profile
 - Start workout routine
 1. Track exercises in-progress
 2. Track personal Quantifiers
 3. Update current routine
 - Create workout routine
 1. Post workout routine
 2. Categorize routine
 3. Add workout length details
 4. Add exercise
 - (a) Add Quantifier
 - (b) Add Workout Descriptions
 - Edit Routine
 - Remove Routine
 - View Workout List

10.2.2 Use case Diagram





10.3 Functional Requirements

R18: Description: Template
 Rational:
 Fit Criterion

R19: Description: The system shall allow the user to create a workout routine.
 Rational: To allow the user to save their own workout routines.

Fit Criterion: The user created routines are accessible after creation.

- R20: Description: The system shall record a workout routine created by a user.
Rational: To be able to store a routine to potentially share the routine with other users.
Fit Criterion: On creation of a workout routine, the data for the routine should be accessible by the system.
- R21: Description: The system shall allow the user to publically post a workout routine.
Rational: To be able to display their workout routine to other users.
Fit Criterion: On publication of a routine, another user should be able to access and view the routine.
- R22: Description: The product shall allow a user to update a workout routine.
Rational: For users to be able to change quantifiers or routine properties.
Fit Criterion: Workout routine data should be changed to new values after a user updates a workout routine.
- R23: Description: The system shall allow a user to track their current workout progress.
Rational: To allow a user to keep track of their current workout.
Fit Criterion: Current workout data is updated after a user adds inputted data regarding their exercise.
- R24: Description: The product shall allow a user to browse workout routines.
Rational: To make publicly posted workout routines discoverable and for users to find routines that cater their fitness goals.
Fit Criterion: Public workout routines should be accessible for user viewing such that users can view posted routines.
- R25: Description: The product should allow a user to save another user's public workout routine.
Rational: To allow a user to save a workout routine for later use.
Fit Criterion: Upon saving a public workout routine, the user shall be able to access it as their own workout to use.
- R26: Description: The system should allow a user to create a profile.
Rational: To display social, informational content to other users.
Fit Criterion: On creating a profile, other users should be able to view any publicly displayed data.
- R27: Description: The system shall allow a user to edit their profile.
Rational: To update any information regarding the user profile.
Fit Criterion: After a user edits their profile any profile data should be changed to the new values set by the user.

- R28: Description: The system shall allow a user to view another user's profile.
Rational: To allow a user to determine if another user has similar fitness goals or similar workout routines.
Fit Criterion: On selecting a user profile, all publicly displayed data should be accessible by the user.
- R29: Description: The product shall record the previous workouts of a user.
Rational: To store all user input data for future display of fitness progress for the user.
Fit Criterion: On completion of a user workout, the data for the workout should be accessible by the system and the user.
- R30: Description: The product shall display fitness progress regarding various exercises.
Rational: To help the user determine progress towards fitness goals.
Fit Criterion: Previous fitness progress completed by a user should be accessible up to a minimum of the previous 30 days.

[Every IM should map to at least one requirement, but not every requirement has to map to a corresponding IM. —TPLT]

11 Nonfunctional Requirements

[List your nonfunctional requirements. You may consider using a fit criterion to make them verifiable. —TPLT] [The goal is for the nonfunctional requirements to be unambiguous, abstract and verifiable. This isn't easy to show succinctly, so a good strategy may be to give a "high level" view of the requirement, but allow for the details to be covered in the Verification and Validation document. —TPLT] [An absolute requirement on a quality of the system is rarely needed. For instance, an accuracy of 0.0101 % is likely fine, even if the requirement is for 0.01 % accuracy. Therefore, the emphasis will often be more on describing how well the quality is achieved, through experimentation, and possibly theory, rather than meeting some bar that was defined a priori. —TPLT] [You do not need an entry for correctness in your NFRs. The purpose of the SRS is to record the requirements that need to be satisfied for correctness. Any statement of correctness would just be redundant. Rather than discuss correctness, you can characterize how far away from the correct (true) solution you are allowed to be. This is discussed under accuracy. —TPLT]

- 11.1 Look and Feel Requirements
- 11.2 Usability and Humanity Requirements
- 11.3 Performance Requirements
- 11.4 Operational and Environmental Requirements
- 11.5 Maintainability and Support Requirements
- 11.6 Security Requirements
- 11.7 Cultural Requirements
- 11.8 Compliance Requirements

- NFR5: **Accuracy** [Characterize the accuracy by giving the context/use for the software. Maybe something like, “The accuracy of the computed solutions should meet the level needed for <engineering or scientific application>. The level of accuracy achieved by Software Eng 4G06 shall be described following the procedure given in Section X of the Verification and Validation Plan.” A link to the VnV plan would be a nice extra. —TPLT]
- NFR6: **Usability** [Characterize the usability by giving the context/use for the software. You should likely reference the user characteristics section. The level of usability achieved by the software shall be described following the procedure given in Section X of the Verification and Validation Plan. A link to the VnV plan would be a nice extra. —TPLT]
- NFR7: **Maintainability** [The effort required to make any of the likely changes listed for Software Eng 4G06 should be less than FRACTION of the original development time. FRACTION is then a symbolic constant that can be defined at the end of the report. —TPLT]
- NFR8: **Portability** [This NFR is easier to write than the others. The systems that Software Eng 4G06 should run on should be listed here. When possible the specific versions of the potential operating environments should be given. To make the NFR verifiable a statement could be made that the tests from a given section of the VnV plan can be successfully run on all of the possible operating environments. —TPLT]
- Other NFRs that might be discussed include verifiability, understandability and reusability.

12 Traceability Matrices and Graphs

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]

	T??	T??	T??	GD??	GD??	DD??	DD??	DD??	DD??	IM??	IM??	IM??	IM??
T??													
T??			X										
T??													
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 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.

	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		
R2	X	X				X	X
R??	X						
R??		X					
R??			X				
R??				X			
R4			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??
T??	X																		
T??																			
T??																			
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

13 Project Issues

13.1 Open Issues

13.2 Off the Shelf Solutions

13.3 New Problems

13.4 Tasks

13.5 Migration to the New Product

13.6 Risks

13.7 Costs

13.8 User Documentation and Training

13.9 Waiting Room

13.10 Ideas for Solutions

14 Reference Material

This section records information for easy reference.

14.1 Abbreviations and Acronyms

symbol	description
A	Assumption
DD	Data Definition
GD	General Definition
GS	Goal Statement
IM	Instance Model
LC	Likely Change
PS	Physical System Description
R	Requirement
SRS	Software Requirements Specification
Software Eng 4G06	[put an expanded version of your program name here (as appropriate) —TPLT]
T	Theoretical Model

[Add any other abbreviations or acronyms that you add —TPLT]

[The following is not part of the template, just some things to consider when filing in the template. —TPLT]

[Grammar, flow and L^AT_EX advice:

- For Mac users *.DS_Store should be in .gitignore
- L^AT_EX and formatting rules
 - Variables are italic, everything else not, includes subscripts ([link to document](#))
 - * [Conventions](#)
 - * Watch out for implied multiplication
 - Use BibTeX
 - Use cross-referencing
- Grammar and writing rules
 - Acronyms expanded on first usage (not just in table of acronyms)
 - “In order to” should be “to”

—TPLT]

[Advice on using the template:

- Difference between physical and software constraints
- Properties of a correct solution means *additional* properties, not a restating of the requirements (may be “not applicable” for your problem). If you have a table of output constraints, then these are properties of a correct solution.
- Assumptions have to be invoked somewhere
- “Referenced by” implies that there is an explicit reference
- Think of traceability matrix, list of assumption invocations and list of reference by fields as automatically generatable
- If you say the format of the output (plot, table etc), then your requirement could be more abstract

—TPLT]

Appendix — Reflection

The information in this section will be used to evaluate the team members on the graduate attribute of Lifelong Learning. Please answer the following questions:

1. 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.
2. 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?