# 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 4, 2022

# Contents

# 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 → 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 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: [Requirements for the inputs that are supplied by the user. This information has to be explicit. —TPLT]

R2: [It isn't always required, but often echoing the inputs as part of the output is a good idea. —TPLT]

R3: [Calculation related requirements. —TPLT]

R4: [Verification related requirements. —TPLT]

R5: [Output related requirements. —TPLT]

[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 now 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.

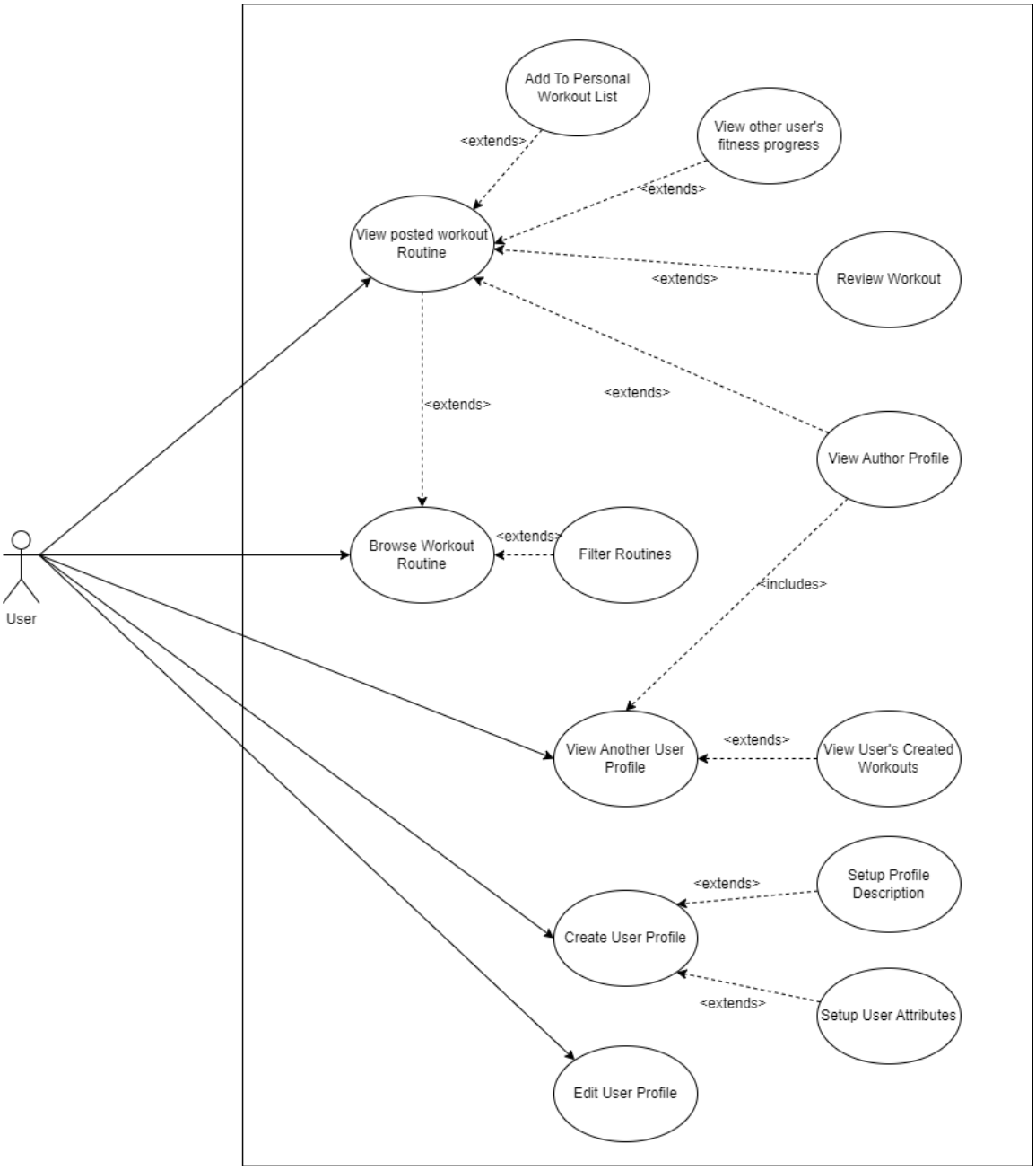
# 7 Use cases

- View posted workout routine
    1. 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

## 7.1 Use case Diagram




# 8 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 | | | 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

# 9 Project Issues

## 9.1 Open Issues

## 9.2 Off the Shelf Solutions

## 9.3 New Problems

## 9.4 Tasks

## 9.5 Migration to the New Product

## 9.6 Risks

## 9.7 Costs

## 9.8 User Documentation and Training

## 9.9 Waiting Room

## 9.10 Ideas for Solutions

# 10 Reference Material

This section records information for easy reference.

## 10.1 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]

# References

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

- For Mac users `*.DS_Store` should be in `.gitignore`
- LaTeX 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?