Software Requirements Specification for Greenway: subtitle describing software

Team #11, Roadkill
Priyansh Shah, shahp36
Utsharga Rozario, rozariou
Jash Mehta, mehtaj8
Bilal Shaikh, shaikb2
Pranay Kotian, kotianp
Sharjil Mohsin, mohsis2

October 4, 2022

Contents

Revision History

Date	Version	Notes
Date 1	1.0	Notes
Date 2	1.1	Notes

1   Introduction

Navigation applications are commonly used applications while driving to get directions from point A to B, but these applications never tell you how much it costs you to get there or how much gas was used on the trip. When carpooling with friends, the driver of the vehicle always asks everyone in the group for gas money and often times these calculations are mere estimates that are not always very accurate. People often wonder how much it costs to get to a destination before starting your journey, having an application perform these cost and fuel calculations based on real time gas pricing information ensures you save the most money on your journey while minimizing gas usage to encourage a more sustainable lifestyle. The introduction section focuses on the" Purpose of the Document", the "Scope of Requirements", the "Characteristics of the Intended Reader", and the "Organization of Document".

1.1   Purpose of Document

The purpose of this document is to outline the overall description and target functionality of the application to stakeholders. It will communicate, without any ambiguity, the system requirements for Greenway and the different use cases and contexts of use. These design aspects will be communicated using a combination of diagrams and written descriptions. In addition, it will also serve as a reference for verifying correctness and validation of the product during the testing phases.

1.2   Scope of Requirements

The scope of the proposed software, Greenway, is a mapping software that not only gives directions to the intended destination but provides the user with fuel cost calculations. The intention is to calculate fuel costs using gas price data, car mileage information and terrain information to show how much money it will take to get to the intended destination using the most fuel efficient route.

1.3   Characteristics of Intended Reader

The intended readers characteristics include having knowledge in the following domains, the understanding of mapping software, the knowledge of how data is stored and will be used in the context of the google maps API, which algorithms need to be used to calculate the most fuel efficient route, the understanding of how terrain information will be used to alter car mileage calculations, and the understanding of how information will be retrieved for all types of data being used. The intended reader needs this knowledge as it is required to understand the rest of the SRS document; this ensures the reader understands what the information is that is being conveyed and how the technology is going to be used to put out the final product.

1.4 Organization of Document

The document is organized into sections with smaller subsection that will aid in conveying information. The intention is that these sections will work together to provide a more informed understanding of the product. The document is seperated into sections that include, "General System Description", "Specific System Description", "Requirements", "Likely Changes", "Unlikely Changes", "Traceability Matrices and Graphs", "Development Plan", and the "Values of Auxiliary Constants".

2 Project Drivers

2.1 The Purpose of the Project

2.2 Stakeholders

2.2.1 Product Users

The product user is anyone who can drive and hence will be using the product. The product users have the most influence on the requirements of the application and its overall development since the purpose of the application is help these users choose an cost effective eco-friendly route for a journey.

2.2.2 Development Team

The development team is the group of people who are developing the application. They are the software developers who are developing the application and hence are an important stakeholder.

2.2.3 Teaching Staff/Instructors

The teaching staff and instructors are also important stake holders as they will be directly interacting with our product and assessing its usability. They are similar to investors and are required to be shown updates and hold influence on the requirements of the application and its development.

3 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]

3.1 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]
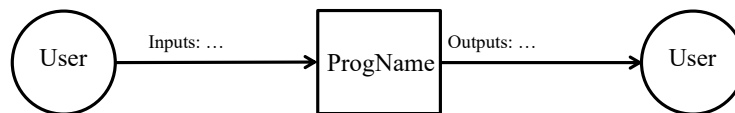

Figure 1: System Context

[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 responsibilites 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:

    –

- Greenway Responsibilities:

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

–

3.2 User Characteristics

3.3 System Constraints

4 Specific System Description

This section first presents the problem description, which gives a high-level view of the problem to be solved. This is followed by the solution characteristics specification, which presents the assumptions, theories, definitions and finally the instance models.

4.1 Problem Description

Greenway is intended to solve ...

4.1.1 Terminology and Definitions

This subsection provides a list of terms that are used in the subsequent sections and their meaning, with the purpose of reducing ambiguity and making it easier to correctly understand the requirements:

-

4.1.2 Physical System Description

The physical system of Greenway, as shown in Figure ?, includes the following elements:

PS1:

PS2: ...

4.1.3 Goal Statements

Given the [inputs —TPLT], the goal statements are:

GS1:

4.2 Solution Characteristics Specification

(GDs). All of these refinements can potentially use Assumptions (A) and Data Definitions (DD). TMs are refined to create new models, that are called GMs or IMs. DDs are not refined; they are just used. GDs and IMs are derived, or refined, from other models. DDs are not derived; they are just given. TMs are also just given, but they are refined, not used. If a potential DD includes a derivation, then that means it is refining other models, which would make it a GD or an IM. —TPLT]

[The above makes a distinction between "refined" and "used." A model is refined to another model if it is changed by the refinement. When we change a general 3D equation to a 2D equation, we are making a refinement, by applying the assumption that the third dimension does not matter. If we use a definition, like the definition of density, we aren't refining, or changing that definition, we are just using it. —TPLT]

[The same information can be a TM in one problem and a DD in another. It is about how the information is used. In one problem the definition of acceleration can be a TM, in another it would be a DD. —TPLT]

[There is repetition between the information given in the different chunks (TM, GDs etc) with other information in the document. For instance, the meaning of the symbols, the units etc are repeated. This is so that the chunks can stand on their own when being read by a reviewer/user. It also facilitates reuse of the models in a different context. —TPLT]

[The relationships between the parts of the document are show in the following figure. In this diagram "may ref" has the same role as "uses" above. The figure adds "Likely Changes," which are able to reference (use) Assumptions. —TPLT]
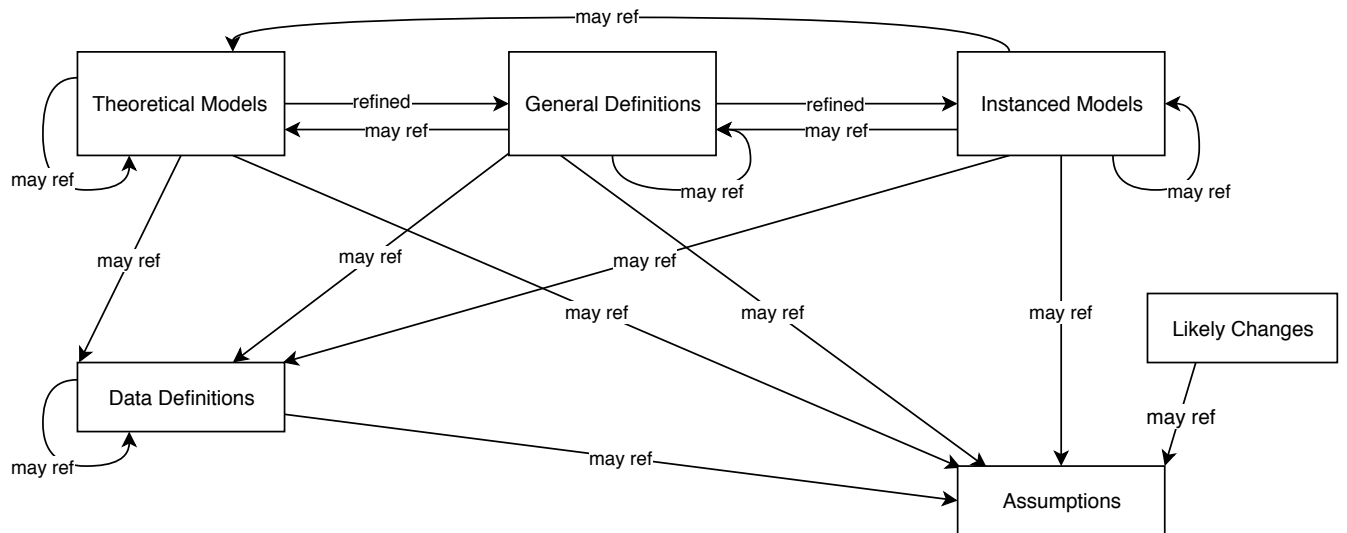
The instance models that govern Greenway are presented in Subsection 4.2.6. The information to understand the meaning of the instance models and their derivation is also presented, so that the instance models can be verified.

4.2.1 Assumptions

[The assumptions are a refinement of the scope. The scope is general, where the assumptions are specific. All assumptions should be listed, even those that domain experts know so well that they are rarely (if ever) written down. —TPLT] [The document should not take for granted that the reader knows which assumptions have been made. In the case of unusual assumptions, it is recommended that the documentation either include, or point to, an explanation and justification for the assumption. —TPLT]

This section simplifies the original problem and helps in developing the theoretical model by filling in the missing information for the physical system. The numbers given in the square brackets refer to the theoretical model [T], general definition [GD], data definition [DD], instance model [IM], or likely change [LC], in which the respective assumption is used.

A1: [Short description of each assumption. Each assumption should have a meaningful label. Use cross-references to identify the appropriate traceability to T, GD, DD etc., using commands like dref, ddref etc. Each assumption should be atomic - that is, there should not be an explicit (or implicit) "and" in the text of an assumption. —TPLT]

4.2.2 Theoretical Models

[Theoretical models are sets of abstract mathematical equations or axioms for solving the problem described in Section "Physical System Description" (Section 4.1.2). Examples of theoretical models are physical laws, constitutive equations, relevant conversion factors, etc. —TPLT]

This section focuses on the general equations and laws that Greenway is based on. [Modify the examples below for your problem, and add additional models as appropriate. —TPLT]

RefName: **T:COE**

Label: Conservation of thermal energy

Equation: $-\nabla \cdot \mathbf{q} + g = \rho C \frac{\partial T}{\partial t}$

Description: The above equation gives the conservation of energy for transient heat transfer in a material of specific heat capacity C ($\mathrm{J\,kg^{-1}\,{}^\circ C^{-1}}$) and density ρ ($\mathrm{kg\,m^{-3}}$), where $\mathbf{q}$ is the thermal flux vector ($\mathrm{W\,m^{-2}}$), g is the volumetric heat generation ($\mathrm{W\,m^{-3}}$), T is the temperature ($^\circ$C), t is time (s), and ∇ is the gradient operator. For this equation to apply, other forms of energy, such as mechanical energy, are assumed to be negligible in the system (A**??**). In general, the material properties (ρ and C) depend on temperature.

Notes: None.

Source: http://www.efunda.com/formulae/heat_transfer/conduction/overview_cond.cfm

Ref. By: GD**??**

Preconditions for T:COE: None

Derivation for T:COE: Not Applicable

[“Ref. By” is used repeatedly with the different types of information. This stands for Referenced By. It means that the models, definitions and assumptions listed reference the current model, definition or assumption. This information is given for traceability. Ref. By provides a pointer in the opposite direction to what we commonly do. You still need to have a reference in the other direction pointing to the current model, definition or assumption. As an example, if T1 is referenced by G2, that means that G2 will explicitly include a reference to T1. —TPLT]

4.2.3   General Definitions

[General Definitions (GDs) are a refinement of one or more TMs, and/or of other GDs. The GDs are less abstract than the TMs. Generally the reduction in abstraction is possible through invoking (using/referencing) Assumptions. For instance, the TM could be Newton's

This section collects the laws and equations that will be used in building the instance models.

Number	GD1
Label	**Newton's law of cooling**
SI Units	$\mathrm{W\,m^{-2}}$
Equation	$q(t) = h\Delta T(t)$
Description	Newton's law of cooling describes convective cooling from a surface. The law is stated as: the rate of heat loss from a body is proportional to the difference in temperatures between the body and its surroundings.
	$q(t)$ is the thermal flux ($\mathrm{W\,m^{-2}}$).
	h is the heat transfer coefficient, assumed independent of T (A**??**) ($\mathrm{W\,m^{-2}\,{}^{\circ}C^{-1}}$).
	$\Delta T(t) = T(t) - T_{\mathrm{env}}(t)$ is the time-dependent thermal gradient between the environment and the object (°C).
Source	Citation here
Ref. By	DD1, DD**??**

Detailed derivation of simplified rate of change of temperature

4.2.4 Data Definitions

This section collects and defines all the data needed to build the instance models. The dimension of each quantity is also given. [Modify the examples below for your problem, and add additional definitions as appropriate. —TPLT]

Number	DD1
Label	**Heat flux out of coil**
Symbol	q_C
SI Units	$\mathrm{W\,m^{-2}}$
Equation	$q_C(t) = h_C(T_C - T_W(t))$, over area A_C
Description	T_C is the temperature of the coil (°C). T_W is the temperature of the water (°C). The heat flux out of the coil, q_C ($\mathrm{W\,m^{-2}}$), is found by assuming that Newton's Law of Cooling applies (A**??**). This law (GD1) is used on the surface of the coil, which has area A_C ($\mathrm{m^2}$) and heat transfer coefficient h_C ($\mathrm{W\,m^{-2}\,°C^{-1}}$). This equation assumes that the temperature of the coil is constant over time (A**??**) and that it does not vary along the length of the coil (A**??**).
Sources	Citation here
Ref. By	IM1

4.2.5 Data Types

[This section is optional. In many scientific computing programs it isn't necessary, since the inputs and outpus are straightforward types, like reals, integers, and sequences of reals and integers. However, for some problems it is very helpful to capture the type information. —TPLT]

[The data types are not derived; they are simply stated and used by other models. —TPLT]

[All data types must be used by at least one of the models. —TPLT]

[For the mathematical notation for expressing types, the recommendation is to use the notation of **?**. —TPLT]

This section collects and defines all the data types needed to document the models. [Modify the examples below for your problem, and add additional definitions as appropriate. —TPLT]

Type Name	Name for Type
Type Def	mathematical definition of the type
Description	description here
Sources	Citation here, if the type is borrowed from another source

4.2.6 Instance Models

[The motivation for this section is to reduce the problem defined in "Physical System Description" (Section 4.1.2) to one expressed in mathematical terms. The IMs are built by refining the TMs and/or GDs. This section should remain abstract. The SRS should specify the requirements without considering the implementation. —TPLT]

This section transforms the problem defined in Section 4.1 into one which is expressed in mathematical terms. It uses concrete symbols defined in Section 4.2.4 to replace the abstract symbols in the models identified in Sections 4.2.2 and 4.2.3.

The goals [reference your goals —TPLT] are solved by [reference your instance models —TPLT]. [other details, with cross-references where appropriate. —TPLT] [Modify the examples below for your problem, and add additional models as appropriate. —TPLT]

Number	IM1
Label	**Energy balance on water to find T_W**
Input	m_W, C_W, h_C, A_C, h_P, A_P, t_{final}, T_C, T_{init}, $T_P(t)$ from IM**??**
	The input is constrained so that $T_{\text{init}} \leq T_C$ (A**??**)
Output	$T_W(t)$, $0 \leq t \leq t_{\text{final}}$, such that
	$\frac{dT_W}{dt} = \frac{1}{\tau_W}[(T_C - T_W(t)) + \eta(T_P(t) - T_W(t))]$,
	$T_W(0) = T_P(0) = T_{\text{init}}$ (A**??**) and $T_P(t)$ from IM**??**
Description	T_W is the water temperature (°C).
	T_P is the PCM temperature (°C).
	T_C is the coil temperature (°C).
	$\tau_W = \frac{m_W C_W}{h_C A_C}$ is a constant (s).
	$\eta = \frac{h_P A_P}{h_C A_C}$ is a constant (dimensionless).
	The above equation applies as long as the water is in liquid form, $0 < T_W < 100^o$C, where 0^oC and 100^oC are the melting and boiling points of water, respectively (A**??**, A**??**).
Sources	Citation here
Ref. By	IM**??**

Derivation of ...

[The derivation shows how the IM is derived from the TMs/GDs. In cases where the derivation cannot be described under the Description field, it will be necessary to include this subsection. —TPLT]

4.2.7 Input Data Constraints

Table 1 shows the data constraints on the input output variables. The column for physical constraints gives the physical limitations on the range of values that can be taken by the variable. The column for software constraints restricts the range of inputs to reasonable values. The software constraints will be helpful in the design stage for picking suitable algorithms. The constraints are conservative, to give the user of the model the flexibility to experiment with unusual situations. The column of typical values is intended to provide a feel for a common scenario. The uncertainty column provides an estimate of the confidence with which the physical quantities can be measured. This information would be part of the input if one were performing an uncertainty quantification exercise.

The specification parameters in Table 1 are listed in Table 2.

Table 1: Input Variables

Var	Physical Constraints	Software Constraints	Typical Value	Uncertainty
L	$L > 0$	$L_{\min} \leq L \leq L_{\max}$	1.5 m	10%

(*) [you might need to add some notes or clarifications —TPLT]

Table 2: Specification Parameter Values

Var	Value
$L_{\min}$	0.1 m

4.2.8 Properties of a Correct Solution

A correct solution must exhibit [fill in the details —TPLT]. [These properties are in addition to the stated requirements. There is no need to repeat the requirements here. These additional properties may not exist for every problem. Examples include conservation laws (like conservation of energy or mass) and known constraints on outputs, which are usually summarized in tabular form. A sample table is shown in Table 3 —TPLT]

Table 3: Output Variables

Var	Physical Constraints
T_W	$T_{\text{init}} \leq T_W \leq T_C$ (by A**??**)

[This section is not for test cases or techniques for verification and validation. Those topics will be addressed in the Verification and Validation plan. —TPLT]

5   Requirements

[The requirements refine the goal statement. They will make heavy use of references to the instance models. —TPLT]

This section provides the functional requirements, the business tasks that the software is expected to complete, and the nonfunctional requirements, the qualities that the software is expected to exhibit.

5.1 Functional Requirements

R1 The system shall allow the user a way to input start location.

R2 The system shall allow the user a way to input final destination.

R3 The system shall allow the user a way to select car models.

R4 The system shall allow the user a way to input the custom details of their car.

R5 The system shall have a way to update available car models from either user input or external sources.

R6 The system shall have a map to show journey from start to final destination.

R7 The system shall display the gas stations along the route.

R8 The system shall display charging stations along the route for electric vehicles.

R9 The system shall have a way to collect gas prices on route to destination.

R10 The system shall have a way to calculate the most fuel efficient route.

R11 The system shall have a way to collect elevation data along the suggested route.

R12 The system shall have a way to calculate fuel consumption on different terrain elevations.

R13 The system shall have a way to calculate the total cost of the journey.

5.2 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]

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 Greenway 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 Greenway 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 Greenway 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.

6 Likely Changes

LC1: The addition of the application being able to use mileage data as well as gas tank size to calculate the most efficient route on a long journey, allowing for the least amount of stops to fill up gas.

LC2: The addition of the application taking data of electric cars to allow the use of not only gas station but charging stations for electric car users.

LC3: The application will be able to be used in all of Ontario with elevation and gas data for all areas.

LC4: Future changes should allow the application to be used with other apps other than Google Maps, such as Wayz or Apple Maps.

7 Unlikely Changes

ULC1: The application will not support other provinces or countries.

ULC2: The application will not be able to update car catalog information on its own and will require an update to the application when new cars are introduced.

ULC3: The application will not support hybrid vehicles as they require complex calculations to determine when the electric component of the vehicle is used and when the gas component is.

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 4 shows the dependencies of theoretical models, general definitions, data definitions, and instance models with each other. Table 5 shows the dependencies of instance models, requirements, and data constraints on each other. Table 6 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]

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.

9 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

16

	T??	T??	T??	GD1	GD??	DD1	DD??	DD??	DD??	IM1	IM??	IM??	IM??
T??													
T??			X										
T??													
GD1													
GD??	X												
DD1				X									
DD??				X									
DD??													
DD??								X					
IM1					X	X	X				X		
IM??					X		X		X	X			X
IM??		X											
IM??		X	X				X	X	X		X		

Table 4: Traceability Matrix Showing the Connections Between Items of Different Sections

	IM1	IM??	IM??	IM??	4.2.7	R??	R??
IM1		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 5: 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??
T??	X																	
T??																		
T??																		
GD1		X																
GD??			X	X	X	X												
DD1							X	X	X									
DD??			X	X							X							
DD??																		
DD??																		
IM1											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 6: Traceability Matrix Showing the Connections Between Assumptions and Other Items

organized as a prioritized list of requirements, or it could should the requirements that will be implemented for "phase 1", "phase 2", etc. —TPLT]

10 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]

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

[Grammar, flow and LaTeX advice:

- 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

There are many skills needed to accomplish the task, these skills however need to be distributed amongst the team as it makes it easier to complete the goals that have been set to acquire a finished product. The domain specific knowledge required ranges from understanding APIs to collecting, storing, and using data. The team requires the knowledge of how to collect the different types of data required, gas price data, elevation data of a specified area, car catalog data that contains information about gas mileage, and google maps data for fuel efficient route calculations. The intention is to use this data as a collective to perform calculations which would allow the user to find the cost of travel from one destination to another in the simplest manner; for this, an understanding of how the Google Maps API works, how the application will be developed, how the interface will be designed, and how everything will work simultaneously to provide the user with the best experience. These tasks are split amongst the team.

Jash — Collection of gas price data and how it will be stored and how the fuel calculation will be performed.

Priyansh — Collection of elevation specific data and how it will be stored and how the fuel calculation will be performed.

Utsharga — Collection of car catalog data and how it will be stored and how the fuel calculation will be performed.

Sharjil — Understanding of how the interface will be designed and how the fuel-efficient route will be calculated.

Bilal — Understanding of how the Google Maps API works and how the fuel-efficient route will be calculated.

Pranay — Understanding How the interface will be designed and how everything will work simultaneously.

There are many approaches to acquiring this knowledge; one approach is researching in the specific domain through forums and google searches. Another approach is getting hands on with the entity you would like to learn, this allows one to experience and interact with it, giving a greater understanding of how everything works.

Jash — The approach for collecting gas price data will be through the research domain, this allows him to find information about the gas prices as well as learn how to collect and store them in a specific way. The same approach applies to the fuel calculations as understanding how its done and factoring in the elevation data is going to be used.

Priyansh — The approach for collecting elevation data will be done through the research domain, this makes sure the data being collected is relevant to the information required

to perform the fuel calculations. The same approach applies to the fuel calculations as understanding how its done and factoring in the elevation data is going to be used.

Utsharga — The approach for collecting car catalog data will be done through the research domain, this allows him to collect all necessary data on all existing cars to be used in production. The same approach applies to the fuel calculations as understanding how its done and factoring in the elevation data is going to be used.

Sharjil — The design of the user interface will be done through the hand-on domain, this allows him to interact with the product and test to see what works and what doesn't. The calculations for the fuel-efficient route will be done through the research domain, this allows him to create and work on the algorithm required to perform the calculation.

Bilal — The Google Maps API will require the hand-on domain to be used to explore how it works and how it will be used for the purposes of this application, this will allow him to learn what to do and how to use it for our needs. The calculations for the fuel-efficient route will be done through the research domain, this allows him to create and work on the algorithm required to perform the calculation.

Pranay — The design of the user interface will be done through the hand-on domain, this allows him to interact with the product and test to see what works and what doesn't. To get everything working together, he will have to look into the hand-on domain to be able to test and see how all the different parts will need to be integrated together to allow for the product to function as intended.