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 Reference Material

This section records information for easy reference.

1.1 Table of Units

Throughout this document SI (Système International d'Unités) is employed as the unit system. In addition to the basic units, several derived units are used as described below. For each unit, the symbol is given followed by a description of the unit and the SI name.

symbol	unit	SI
m	length	metre
kg	mass	kilogram
s	time	second
°C	temperature	centigrade
J	energy	joule
W	power	watt ($\mathrm{W} = \mathrm{J\,s^{-1}}$)

[Only include the units that your SRS actually uses. —TPLT]

[Derived units, like newtons, pascal, etc, should show their derivation (the units they are derived from) if their constituent units are in the table of units (that is, if the units they are derived from are used in the document). For instance, the derivation of pascals as $\mathrm{Pa} = \mathrm{N\,m^{-2}}$ is shown if newtons and m are both in the table. The derivations of newtons would not be shown if kg and s are not both in the table. —TPLT]

[The symbol for units named after people use capital letters, but the name of the unit itself uses lower case. For instance, pascals use the symbol Pa, watts use the symbol W, teslas use the symbol T, newtons use the symbol N, etc. The one exception to this is degree Celsius. Details on writing metric units can be found on the NIST web-page. —TPLT]

1.2 Table of Symbols

The table that follows summarizes the symbols used in this document along with their units. The choice of symbols was made to be consistent with the heat transfer literature and with existing documentation for solar water heating systems. The symbols are listed in alphabetical order.

symbol	unit	description
A_C	$\mathrm{m^2}$	coil surface area
A_{in}	$\mathrm{m^2}$	surface area over which heat is transferred in

[Use your problems actual symbols. The si package is a good idea to use for units.

—TPLT]

1.3 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
Greenway	[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]

1.4 Mathematical Notation

[This section is optional, but should be included for projects that make use of notation to convey mathematical information. For instance, if typographic conventions (like bold face font) are used to distinguish matrices, this should be stated here. If symbols are used to show mathematical operations, these should be summarized here. In some cases the easiest way to summarize the notation is to point to a text or other source that explains the notation. —TPLT]

[This section was added to the template because some students use very domain specific notation. This notation will not be readily understandable to people outside of your domain. It should be explained. —TPLT]

[This SRS template is based on **??**. It will get you started. You should not modify the section headings, without first discussing the change with the course instructor. Modification means you are not following the template, which loses some of the advantage of a template, especially standardization. Although the bits shown below do not include type information, you may need to add this information for your problem. If you are unsure, please can ask the instructor. —TPLT]

[Feel free to change the appearance of the report by modifying the LaTeX commands. —TPLT]

[This template document assumes that a single program is being documented. If you are documenting a family of models, you should start with a commonality analysis. A separate template is provided for this. For program families you should look at **??**. Single family member programs are often programs based on a single physical model. General purpose tools are usually documented as a family. Families of physical models also come up. —TPLT]

[The SRS is not generally written, or read, sequentially. The SRS is a reference document. It is generally read in an ad hoc order, as the need arises. For writing an SRS, and for reading one for the first time, the suggested order of sections is:

- Goal Statement
- Instance Models
- Requirements
- Introduction
- Specific System Description

—TPLT]

[Guiding principles for the SRS document:

- Do not repeat the same information at the same abstraction level. If information is repeated, the repetition should be at a different abstraction level. For instance, there will be overlap between the scope section and the assumptions, but the scope section will not go into as much detail as the assumptions section.

—TPLT]

[The template description comments should be disabled before submitting this document for grading. —TPLT]

[You can borrow any wording from the text given in the template. It is part of the template, and not considered an instance of academic integrity. Of course, you need to cite the source of the template. —TPLT]

[When the documentation is done, it should be possible to trace back to the source of every piece of information. Some information will come from external sources, like terminology. Other information will be derived, like General Definitions. —TPLT]

[An SRS document should have the following qualities: unambiguous, consistent, complete, validatable, abstract and traceable. —TPLT]

[The overall goal of the SRS is that someone that meets the Characteristics of the Intended Reader (Section 2.3) can learn, understand and verify the captured domain knowledge. They should not have to trust the authors of the SRS on any statements. They should be able to independently verify/derive every statement made. —TPLT]

2 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".

2.1 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.

2.2 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.

2.3 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.

2.4 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".

3 General System Description

This section provides general information about the system. It identifies the interfaces between the system and its environment and describes the user characteristics.

3.1 System Context

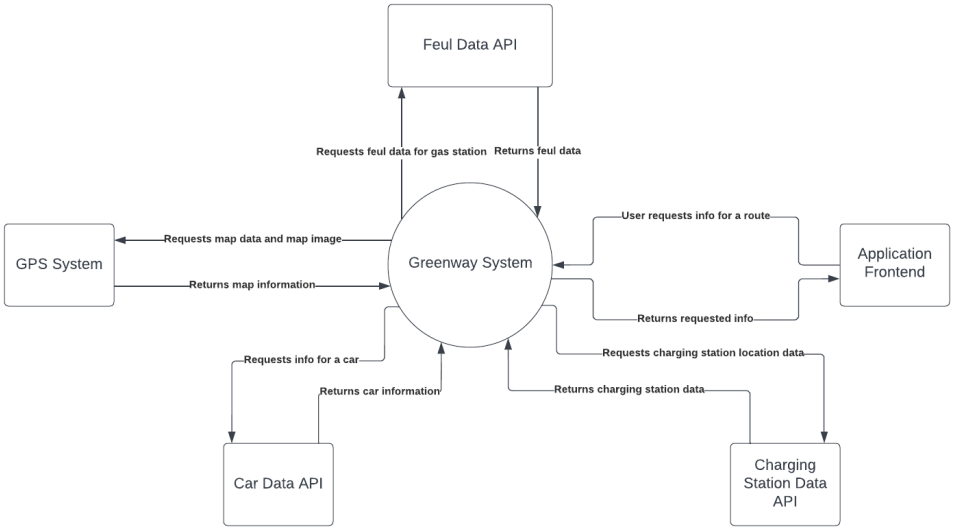

Figure 1: Context Diagram

- User Responsibilities:
  - Responsible for interacting with Application Frontend and receiving its output.
- Greenway System Responsibilities:
  - Responsible for calculating the total fuel price for trip with data from Feul Data API.
  - Responsible for interacting with GPS API to get all neccessary GPS data to present to user.
  - Responsible for interacting with Car Data API to get car information and process it.
  - Responsible for interacting with Charging station API to get charging station data.
- Application Frontend Responsibilities:
  - Responsible for allowing user to interact with application and displaying application output.
- Fuel Data API Responsibilities:
  - Responsible for sending Feul Station Data to any service requesting it.
- GPS System Responsibilities:
  - Responsible for sending any GPS data required to a service needing it.
- Car Data API Responsibilities:
  - Responsible for sending any Car data required to a service needing it.
- Charging API Responsibilities:
  - Responsible for sending any Charging Station data required to a service needing it.

3.2 User Characteristics

The users for this application will be individuals that are looking to drive electric cars, or already have them. As such this application serves individuals considering to transition to electric vehicles from gas vehicles, and will assume all users have basic familiarity with cars. This app will also support individuals in assessing the infrastructure surrounding electric cars to ease in the transition previously mentioned. Lastly, this app can also assist individuals that already own electric cars to find economic routes to reach their destination.

4 Behavior

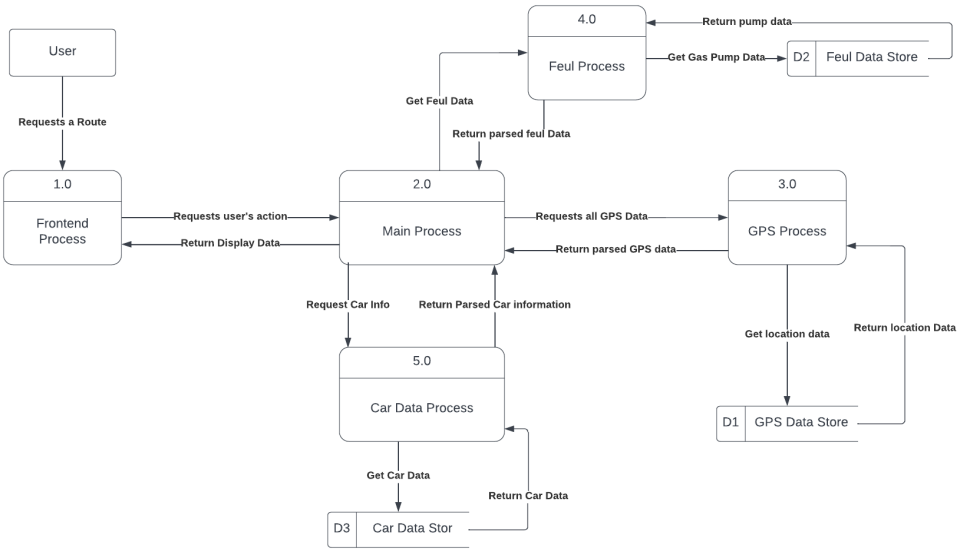

Figure 2: Application Behavior from a Data Flow Diagram

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: [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]

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 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

	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

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 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 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

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 doesnt. 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 doesnt. 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.