Software Requirements Specification

for

Campus Mapomatic

Version 1.1 approved

Prepared by Eliss Glasper

Pink Australian Pigeons

3/21/2021

Table of Contents

Table of Contents ii

Revision History ii

1. Introduction 1

1.1 Purpose and Scope 1

1.2 Definitions, Acronyms and Abbreviations 1

1.3 References 1

1.4 Overview 1

2. Overall Description 1

2.1 Product Functions 1

2.2 User Classes and Characteristics 1

2.3 Constraints 2

2.4 Assumptions and Dependencies 2

3. Specific requirements 2

3.1 External Interface Requirements 2

3.1.1 User Interfaces 2

3.1.2 Hardware Interfaces 2

3.1.3 Software Interfaces 2

3.1.4 Communications Interfaces 3

3.2 Functional requirements 3

3.2.1 Functional requirement 1 3

3.2.2 Functional requirement 2 3

3.3 Performance Requirements 3

3.4 Software System Attributes 4

3.4.1 Availability 4

3.4.2 Security 4

3.5 Design constraints 4

3.5.1 Standards Compliance 4

3.5.2 Hardware Limitations 4

Appendix A: Issues List 4

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason for Changes** | **Version** |
| Eliss Glasper | 03/25/2021 | Fixing mistakes that professor brought to light. | 1.0 |
| Cody Colburn | 03/25/2021 | Fixing mistakes that professor brought to light. | 1.0 |
| Jose Ramirez | 03/25/2021 | Fixing mistakes that professor brought to light. | 1.0 |
| Eliss Glasper | 05/2/2021 | Addng wxPython as our user interface | 1.1 |

# Introduction

## Purpose and Scope

<Identify the product whose software requirements are specified in this document, including the revision or release number. Describe the scope of the product that is covered by this SRS (what does it do and what does it not do), particularly if this SRS describes only part of the system or a single subsystem.>

This product is a navigation system that will help students and faculty get from one building to another, with the ability to display routes for students and faculty with disabilities.

## Definitions, Acronyms and Abbreviations

<Describe definitions of all the terms, acronyms and abbreviations used in the document. Special attention should be paid to the clarification of terms and concepts from the domain of application.>

MCOM = Mass Communications

Nigh = Nigh University building

LIB = Chamber’s library

API = Application programming interface

UI = User Interface

## References

<List any other documents or Web addresses to which this SRS refers. These may include user interface style guides, contracts, standards, system requirements specifications, use case documents, or a vision and scope document. Provide enough information so that the reader could access a copy of each reference, including title, author, version number, date, and source or location.>

Google API documentation

<https://developers.google.com/docs/api>

Python

https://www.python.org/doc/

Software Engineering Course project description

<https://learn.uco.edu/d2l/le/content/323235/viewContent/5836963/View>

wxPython

<https://www.wxpython.org/>

cefpython

https://pypi.org/project/cefpython3/#:~:text=CEF%20Python%20is%20an%20open,%2C%20PyGame%2FPyOpenGL%20and%20PyWin32.

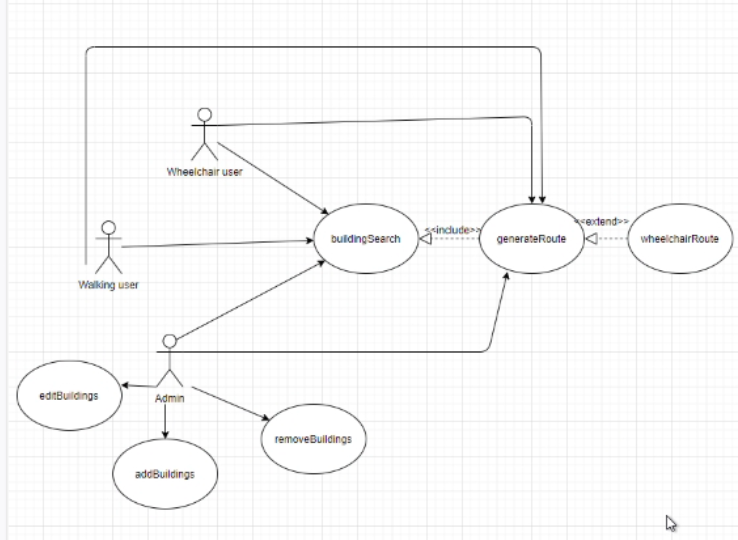
## Overview

<Contain an outline of the remainder of the document.>

This document provides information regarding the Campus Mapomatic system. In the first section we discuss the overall description of the system. We go into detail with a use case diagram assumptions and dependencies. Next, we talk about the specific requirements then move onto Ui details and external interface requirements. We go into detail regarding function requirements, software requirements, and system attributes and constraints. At the end it explains any issues or problems we have run into or fixed.

# Overall Description

## Product Functions

<Summarize the major features the product contains or the significant functions that it performs or lets the user perform. Details will be provided in Section 3, so only a high-level summary is needed here. Organize the functions to make them understandable to any reader of the SRS. A picture of the major groups of related requirements and how they relate, i.e., a use case diagram, is required.>

## User Classes and Characteristics

<Identify the various user classes that you anticipate will use this product. User classes may be differentiated based on frequency of use, subset of product functions used, technical expertise, security or privilege levels, educational level, or experience. Describe the pertinent characteristics of each user class. Certain requirements may pertain only to certain user classes. Distinguish the favored user classes from those who are less important to satisfy.>

* Faculty
  + With disability – would have access to wheelchair accessible routes
* Students
  + With disability - would have access to wheelchair accessible routes
* School visitors
  + With disabilities - would have access to wheelchair accessible routes
* Admin
  + Has access to edit building, add building, remove building

## Assumptions and Dependencies

<List any assumed factors (as opposed to known facts) that could affect the requirements stated in the SRS. These could include third-party or commercial components that you plan to use, issues around the development or operating environment, or constraints. The project could be affected if these assumptions are incorrect, are not shared, or change. Also identify any dependencies the project has on external factors, such as software components that you intend to reuse from another project, unless they are already documented elsewhere (for example, in the vision and scope document or the project plan).>

* We assume that there will be no major construction on campus that will hinder travel route.
* We assume that there will be no campus events that will block routes.

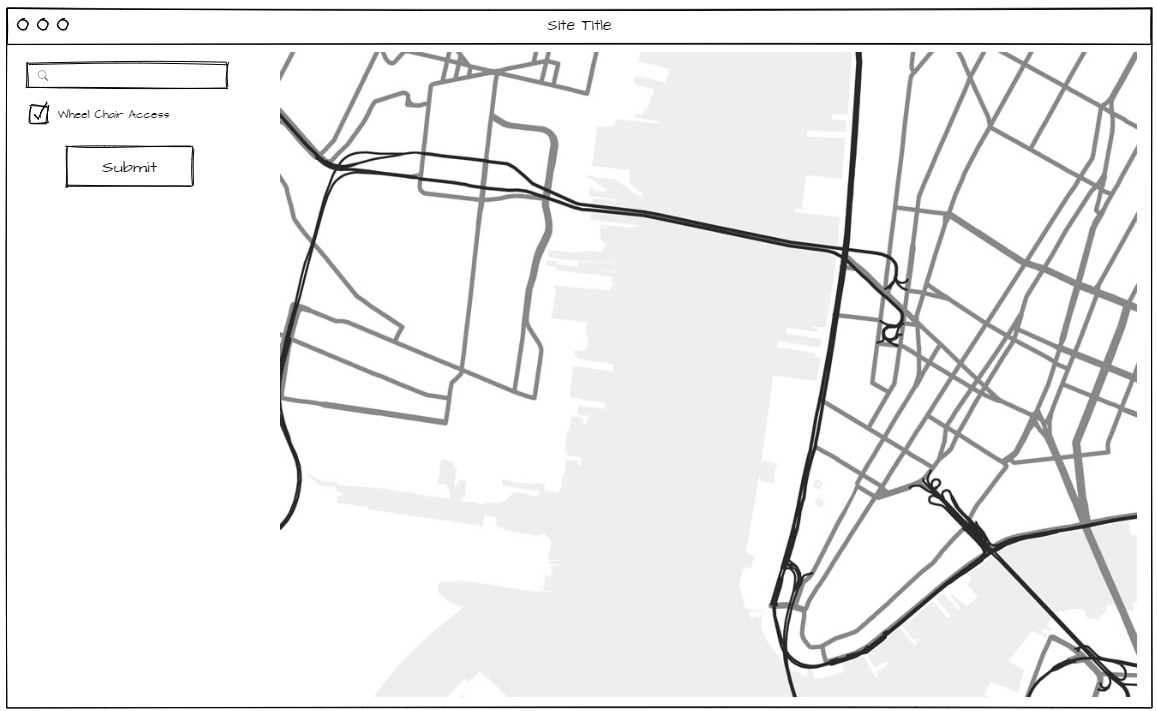
# Specific requirements

<This template illustrates organizing the functional requirements for the system-by-system features, the major services provided by the system. You may prefer to organize this section by use case, mode of operation, user class, object class, functional hierarchy, or combinations of these, whatever makes the most logical sense for your system.>

## External Interface Requirements

### User Interfaces

<Describe the logical characteristics of each interface between the software product and the users. This may include sample screen images, any GUI standards or product family style guides that are to be followed, screen layout constraints, standard buttons and functions (e.g., help) that will appear on every screen, keyboard shortcuts, error message display standards, and so on. Define the software components for which a user interface is needed. Details of the user interface design should be documented in a separate user interface specification.>



### Hardware Interfaces

<Describe the logical and physical characteristics of each interface between the software product and the hardware components of the system. This may include the supported device types, the nature of the data and control interactions between the software and the hardware, and communication protocols to be used.>

* Our application is very lightweight and can run on any pc or laptop.
* Mac and PC will both be supported.

### Software Interfaces

<Describe the connections between this product and other specific software components (name and version), including databases, operating systems, tools, libraries, and integrated commercial components. Identify the data items or messages coming into the system and going out and describe the purpose of each. Describe the services needed and the nature of communications. Refer to documents that describe detailed application programming interface protocols. Identify data that will be shared across software components. If the data sharing mechanism must be implemented in a specific way (for example, use of a global data area in a multitasking operating system), specify this as an implementation constraint.>

We will use Kivy, which is a Python UI module, that will then interact with the Google Maps API.

The Google Maps API produces an html link of the map generated which will then be displayed inside of our application using Kivys integrated html embedding

### Communications Interfaces

<Describe the requirements associated with any communications functions required by this product, including e-mail, web browser, network server communications protocols, electronic forms, and so on. Define any pertinent message formatting. Identify any communication standards that will be used, such as FTP or HTTP. Specify any communication security or encryption issues, data transfer rates, and synchronization mechanisms.>

* The Maps APIs web services use HTTP requests to specific URLs, passing URL parameters as arguments to the services. Generally, these services return data in the HTTP request as either JSON or XML for parsing and/or processing by your application.
  + https://developers.google.com/maps/documentation/directions/web-service-best-practices
* A typical Directions API web service request is generally of the following form:
  + [https://maps.googleapis.com/maps/api/*directions*/*output*?*parameters*](https://maps.googleapis.com/maps/api/directions/output?parameters)
* Requires a valid URL
* Large numbers of synchronized requests to Google's APIs can look like a Distributed Denial of Service (DDoS) attack on Google's infrastructure, and be treated accordingly. To avoid this, you should make sure that API requests are not synchronized between clients.

## Functional requirements

<Describe how the transformation of inputs to outputs is achieved. The description is given for each class of functions, and sometimes for each individual function. To a certain extent, this description can be seen as a solution to the user. This component of the requirement specification is the main starting point for the design phase.>

### buildingSearch

#### Introduction

<Describe the purpose of this function and the approaches and techniques used. The introduction should include information to clarify the intent of the function.>

This function allows the user to search for buildings

Inputs

<*Describe the function’s inputs (source, quantities, range of acceptable values and the like).>*

* *Characters would be acceptable values*
* *Name of building*

#### Processing

<A definition of the operations that must be performed, such as checking for acceptable values, reaction to abnormal situations, or a description of algorithms to be used.>

* If building name does not exist error will display
* Building name will be entered then system will search for building in building list.

#### Outputs

<*Describes the outputs (destination, quantities, error messages, and the like).>*

* *Errors message if building does not exist*
* *Will display a pin on the buildings location if found.*

### wheelChairRoute

#### Introduction

<Describe the purpose of this function and the approaches and techniques used. The introduction should include information to clarify the intent of the function.>

This function displays the route for the wheel chair user

Inputs

<*Describe the function’s inputs (source, quantities, range of acceptable values and the like).>*

* *Coordinates for both buildings*
* *Latitude and longitude are the acceptable values*
* *Integer values only*
* *Check the box for whether user is using a wheelchair*

#### Processing

<A definition of the operations that must be performed, such as checking for acceptable values, reaction to abnormal situations, or a description of algorithms to be used.>

* Send the coordinates to Google Maps API

#### Outputs

<*Describes the outputs (destination, quantities, error messages, and the like).>*

* Display Route from coordinates that shows wheelchair access route

### generateRoute

#### Introduction

<Describe the purpose of this function and the approaches and techniques used. The introduction should include information to clarify the intent of the function.>

This function displays the route for the user

Inputs

<*Describe the function’s inputs (source, quantities, range of acceptable values and the like).>*

* *Coordinates for both buildings*
* *Latitude and longitude are the acceptable values*
* *Integer values only*

#### Processing

<A definition of the operations that must be performed, such as checking for acceptable values, reaction to abnormal situations, or a description of algorithms to be used.>

* Send the coordinates to Google Maps API

#### Outputs

<*Describes the outputs (destination, quantities, error messages, and the like).>*

* Display Route from coordinates

### addBuilding

#### Introduction

<Describe the purpose of this function and the approaches and techniques used. The introduction should include information to clarify the intent of the function.>

This function allows the admin to add buildings to the program

Inputs

<*Describe the function’s inputs (source, quantities, range of acceptable values and the like).>*

* *Coordinates for buildings*
* *Latitude and longitude are the acceptable values*
* *Integer values only*
* *Building Name*

#### Processing

<A definition of the operations that must be performed, such as checking for acceptable values, reaction to abnormal situations, or a description of algorithms to be used.>

* Add to the List of buildings in our code

#### Outputs

<*Describes the outputs (destination, quantities, error messages, and the like).>*

* Building(s) is added to the list

### removeBuilding

#### Introduction

<Describe the purpose of this function and the approaches and techniques used. The introduction should include information to clarify the intent of the function.>

This function allows the admin to remove buildings to the program

Inputs

<*Describe the function’s inputs (source, quantities, range of acceptable values and the like).>*

* *Coordinates for buildings*
* *Latitude and longitude are the acceptable values*
* *Integer values only*
* *Building(s) Name*

#### Processing

<A definition of the operations that must be performed, such as checking for acceptable values, reaction to abnormal situations, or a description of algorithms to be used.>

* remove the building(s) from the List

#### Outputs

<*Describes the outputs (destination, quantities, error messages, and the like).>*

* Building(s) is removed from the List

### editBuilding

#### Introduction

<Describe the purpose of this function and the approaches and techniques used. The introduction should include information to clarify the intent of the function.>

This function allows the admin to add buildings to the program

Inputs

<*Describe the function’s inputs (source, quantities, range of acceptable values and the like).>*

* *Coordinates for buildings*
* *Latitude and longitude are the acceptable values*
* *Integer values only*
* *Building Name*

#### Processing

<A definition of the operations that must be performed, such as checking for acceptable values, reaction to abnormal situations, or a description of algorithms to be used.>

* Editing the List of buildings in our code

#### Outputs

<*Describes the outputs (destination, quantities, error messages, and the like).>*

* Building(s) is edited

… …

## Performance Requirements

<If there are performance requirements for the product under various circumstances, state them here and explain their rationale, to help the developers understand the intent and make suitable design choices. Specify the timing relationships for real time systems. Make such requirements as specific as possible. You may need to state performance requirements for individual functional requirements or features.>

* Program should load no longer than 10 secs
* Should not need more than 5gb of Storage
* Should not need more than 8gb of ram
* API requests shouldn’t take longer than 10 secs

## Software System Attributes

<Particular attention is paid to quality aspects. These requirements must be measurable and verifiable. They must be stated in objective terms.>

### Availability

<Factors that guarantee a certain level of availability such as restart procedures. In this subsection we may also enlist requirements regarding fault tolerance (with respect to both hardware failures and software failures.>

Our system will be available 95% of the time

### Security

<Requirements regarding unauthorized access and other forms of misuse. Certain cryptographic techniques may be prescribed, and we may put constraints on the communication between different parts of the system.>

* Open System will not require login information

## Constraints

<Design constraints may result from such things as the prescribed use of certain standards or hardware.>

* Must have autofill function
* Must have wheelchair accessibility routes
* Project due 04/30/2021

### Standards Compliance

<Which existing standards or regulations must be followed, and what requirements result from these. For example, certain report formats or audit procedures may be prescribed>

### Hardware Limitations

<A description of the characteristics of the hardware environment, in as far as they lead to software requirements. An example of this might be the amount of memory available.>

* Must have a computer
* Wont work on mobile devices

Appendix A: Issues List

< This is a dynamic list of the open requirements issues that remain to be resolved, including TBDs, pending decisions, information that is needed, conflicts awaiting resolution, and the like.>

To be determined