Software Requirements Specification

for

RouteGuesser

Version 1.0 approved

Prepared by Group 9:

CJ Reitter, Anna Marini, Nico Ramos-Fernadez, Laith Agbaria, Mateusz Wilk, & Alex Lopez-Ruiz

10/12/24

Table of Contents

1. Introduction

1.1 Purpose

RouteGuesser is a web-based game where the user guesses the best possible route between two points in Leiden. The user is given an interactive map of Leiden with a starting and ending point, and they have to connect them to create the most efficient route possible. The program evaluates the result and updates a final score. The game never stops: the last destination becomes the start of the new round.

This is revision 1.0 of the project, released on 13/12/24. This will be the final revision of this product for the purposes of the Software Development class. It describes the entirety of the web-based game.

1.2 Document Conventions

This SRS is divided into 5 sections, with their own labeled subsections, bolded and sized accordingly. There is no particular relevance to any of the fonts or colors used, formatted using the default Google Docs settings.

1.3 Intended Audience and Reading Suggestions

This document is intended to be read mainly by Group 9 and the Software Development professors/TAs. Other casual users (players) of RouteGuesser also have access to this document, though none of it is needed to run the game.

Software Development professors/TAs should read the entire document to fully understand the scope of the project. There is no set order for which this document should be read. Players need only read section 1, 4, and 5 if they deem it necessary.

1.4 Project Scope

RouteGuesser is a web-based game intended to be given to incoming First Year students of Leiden University. It aims to give those students a fun and engaging way to better understand the city of their university by turning exploring the city into a browser game. It is made to be both helpful to onboarding students who can play the game while exploring the city, as well as prospective students who can’t physically be in the city.

1.5 References

[1] Ecma International. *ECMAScript Language Specification*. Ecma International, June 2023, https://262.ecma-international.org/14.0/. Accessed 17 Nov. 2024.

[2] Python Software Foundation. *The Python Standard Library.* Python.org, 2024,

https://docs.python.org/3/library/index.html. Accessed 17 Nov. 2024.

2. Overall Description

2.1 Product Perspective

RouteGuesser is a standalone web-based game made purely for use on the site it hosts, with no relation to any other software, programs, apps, etc. that may exist right now or in the future. RouteGuesser was made to be a game played by Leiden University First Years, as we looked to find a way for the First Years to get to know their host city better.

2.2 Product Features

RouteGuesser is a simple game with really only one function: running said game. After the difficulty of the game is selected, the user iterates through rounds of the game, where they move through a map of Leiden, clicking on nodes to advance, reaching a given final destination and receiving a score based on how accurate their path is against the fastest possible path calculated by the program subtracting points accordingly based on if and when the user backtracks on their path. After each iteration, the user is able to stop playing or continue playing, with no end point unless the user quits the game.

2.3 User Classes and Characteristics

There are two main user classes, differentiated by their access to the backend of the program. Software development teachers and TAs will be able to access RouteGuesser and the databases and Python and JavaScripts programs that run the game and UI respectively.

All other users, ideally students at Leiden University, specifically First Years, will only have access to the actual game, without any access to the backend that allows the game to run.

Both user classes are able to freely play the game, with no limit on how long they can play the game for. They can change the difficulty and have no different abilities or rules while playing the game.

2.4 Operating Environment

RouteGuesser operates on a private web server which can be accessed by anyone given the link to the game, and therefore, ideally operates on all web browsers and therefore, on all platforms that support web-based search.

An internet connection is required to maintain RouteGuesser’s ability to run on a web browser and connect to its backend server.

2.5 Design and Implementation Constraints

A device able to run a web browser and hold an internet connection is needed to run RouteGuesser. The game is point and click so the only controls needed are a touch screen/mouse(pad) depending on the device the game is run on. The game is run on a private server and access to that server is needed to run it.

All rules and documentation for RouteGuesser are written in English so users are limited based on their ability to read English.

2.6 User Documentation

Standard README.txt with default formatting will be attached, acting as a user manual for running RouteGuesser and a rulebook for the game.

2.7 Assumptions and Dependencies

A stable internet connection is assumed to be needed to run RouteGuesser consistently as there is no save feature implemented so it's assumed that if the internet cuts out, the program will restart. Furthermore, refreshing the browser the game runs on will likely restart it as well.

For a comfortable experience with the game, it is assumed that the user should be able to read a map and have basic spatial intelligence.

3. System Features

3.1 Running RouteGuesser

3.1.1 Description and Priority

This is the highest priority task for this project. RouteGuesser running as the most simple form of the game as it was designed can be is the basic goal of this project.

3.1.2 Stimulus/Response Sequences

User presses "start" or "play again": System loads RouteGuesser map, all possible nodes, starting node, adjacent nodes and destination goal.
User clicks adjacent node: Perspective of map changes to that node, loads new adjacent nodes, and adds node to the path user has made
User clicks previous node: Perspective of map changes to that node, loads new adjacent nodes, removes points for the possible score, and removes node from the path user has made
User reaches destination node: Score based on the path's closeness to the ideal path is graded, score shown to user, asked if user wants to play again

3.1.3 Functional Requirements

REQ-1: Front-end UI implemented to properly show user menus, the map, and nodes

REQ-2: Algorithm implemented to take map JSON data of Leiden and convert it into a list of nodes and visual the map
REQ-3: Algorithm implemented to track the path the user creates while playing the game and calculates the time it takes to traverse said path
REQ-4: Algorithm implemented to find the optimal path between the source and destination node
REQ-5: Scoring algorithm implemented that compares the two paths, generating a ideal score, subtracting

3.2 Difficulty Adjustment

3.1.1 Description and Priority

This is the lowest priority task for this project, only done if all above requirements are implemented. This creates three different difficulty options for the user, which adjusts how the map and nodes are displayed accordingly

3.1.2 Stimulus/Response Sequences

User choses “easy” difficulty: TBD
User choses “medium” difficulty: TBD
User choses “hard” difficulty: TBD

3.1.3 Functional Requirements

REQ-1: TBD
REQ-2: TBD
REQ-3: TBD

4. External Interface Requirements

4.1 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.>
TBA

4.2 Software Interfaces

RouteGuesser’s program is mainly written in Python, with it’s UI designed and programmed using HTML and JavaScript. All standard language protocol is followed for each of these languages while writing the program [1][2].

RouteGuesser's program has a built-in JSON database that it pulls from to create the map. The database is stagnant so there is no communication in this step. From there, the web browser sends requests to the program which the program relays back to the browser, allowing the game to update and allow the user to progress with the game.

4.3 Communications Interfaces

RouteGuesser's program is hosted on a private web server that ideally runs on any standard web browser. A simple web-based game, there is very little communication aside from between the program and the web server. The user therefore doesn't need to follow any standard communication protocol aside from having a stable internet connection to allow the web browser to run.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

As this is a web-based game, RouteGuesser doesn't have much in the way of performance requirements aside from the at least 2GB of RAM and a stable internet connection of at least 100 MBps that is needed to run any web browser comfortably.

5.2 Software Quality Attributes

RouteGuesser is a very rudimentary web-based game in early development, and therefore, has not been tested as extensively as generally necessary for a public web-based game. So README.txt should be read before running the game.

Furthermore, RouteGuesser has zero save functionality, so all progress in the game is lost if the webpage is closed or refreshed. So reusing this program will always restart a user's progress.

Appendix A: Analysis Models

<Optionally, include any pertinent analysis models, such as data flow diagrams, class diagrams, state-transition diagrams, or entity-relationship diagrams.>

Appendix B: Issues List

For the purposes of this class, this issue list is completely resolved, with no new updates being resolved for RouteGuesser in this submitted state. However, the following problems are/may still be present:

- Testing Errors: Testers found bugs in program that need to be resolved
- Communication Error: Despite initially working, the communication between backend and frontend has issues