|  |  |
| --- | --- |
| BlueT_nobackgroundcmyk_sml | logo |
|  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  | | | | | | | Shortest Path Design Document | | | | | | |  | | |  |  | | | | Status: | Draft | | | | Version: | 0.1 | | | | Date: | **16 Nov 2018** | | | | | |
|  |  |  | |

Table of Contents

[1. Document Purpose 3](#_Toc530150696)

[2. Scope & Out of Scope 3](#_Toc530150697)

[2.1 Scope 3](#_Toc530150698)

[2.2 Out of Scope 3](#_Toc530150699)

[3. Requirement Reference 4](#_Toc530150700)

[4. Assumptions and Dependencies 4](#_Toc530150701)

[4.1 Assumptions 4](#_Toc530150704)

[4.2 Dependencies 5](#_Toc530150705)

[5. Key Architecture Decisions 5](#_Toc530150706)

[6. Solution Architecture 6](#_Toc530150707)

[6.1 Technology Stack 6](#_Toc530150710)

[7. References 6](#_Toc530150711)

# Document Purpose

This document provides Design for Path Optimization. Solution will find the shortest possible route that visits each city and return to the origin city, for a given list of cities and the distance between them. The program uses The Held Karp algorithm. It is as follows:

* Dynamic programming which gives real time solution for any valid input entered by the user.

# Scope & Out of Scope

# Scope

* The scope of is to implement a Path Optimization solution to find the shortest possible route that visits each city and return to the origin.
* Docker Containerization of the solution

# Out of Scope

The activities/deliverables which are out of scope of this change are:

* Solution is not bound to find optimal solution for directed graphs(path to be travelled among the cities is not defined.)

**Road Map for Path Optimization Solution**

* Minimum Viable Product developed initially will comprise of the below listed components
  + - Optimized path for given cities

# Requirement Reference

|  |  |
| --- | --- |
| **Requirements** | |
| US001 | User will provide a number of cities and distance between each city. Application should give optimised path as solution. |
| US002 | User should be able to containerize the solution and run the solution in docker |
| **Version Control through Bitbucket** | |
| US003 | User can download solution from gitHub public repository. |

# Assumptions and Dependencies



# Assumptions

* + - Current solution does not cater to ensure additional requirements like finding optimal path for cities connected in particular design patterns. Shortest path can be calculated considering distance between cities in any direction.
    - Environment specific configurations and management to be done by individual teams.
    - Public GitHub repository can be used for version control
    - Docker container setup will be used from available public sources under trail run

# Dependencies

* Availability of public gitHub repositories for version control
* Availability of Docker setup

# Key Architecture Decisions

|  |  |  |  |
| --- | --- | --- | --- |
| Sl. No | Dependency Description | Impacted Systems | Priority |
|  |  |  |  |

Table #1

# Solution Architecture

* Centralized Source Control system to be followed (GitHub in this case).
* Dynamic programming for various inputs to be considered



# Technology Stack

1. Java Maven Project
2. Dockers

# References

1. <https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-greedy-algo-7/>
2. <http://benalexkeen.com/implementing-djikstras-shortest-path-algorithm-with-python/>