**High Level Design**

for

**My Social Connections App**



AUTHOR: **SANDEEP PAPUDESI**

EMAIL ID: **SANDEEP.PAPUDESI@YAHOO.COM**

**TABLE OF CONTENTS**

[1. iNTRODUCTION 4](#_Toc521878680)

[1.1 In Scope 4](#_Toc521878681)

[1.2 Out of Scope 4](#_Toc521878682)

[1.3 Assumptions 4](#_Toc521878683)

[1.4 Dependencies 4](#_Toc521878684)

[2. Realized Design decissions 5](#_Toc521878685)

[3. Solution view 6](#_Toc521878686)

[3.1 Retrieve users with highest connections 7](#_Toc521878687)

[3.2 Retrieve users with lowest connections 7](#_Toc521878688)

[3.3 Retrieve total number of connections for Lucas (id 4) 7](#_Toc521878689)

[3.4 Total number of connections between two users 8](#_Toc521878690)

[3.5 Find Path between two users 8](#_Toc521878691)

[4. Visual UI 9](#_Toc521878692)

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date of Revision** | **Description of Change** | **Author** |
| 0.1 | 08/10/2018 | Initial Draft | Sandeep Papudesi |
| FINAL | 08/12/2018 | Updated document | Sandeep Papudesi |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# iNTRODUCTION

The purpose of this document is to present a high-level design to address user stories provided by CapitalOne.

## In Scope

Below features are enabled as part of application development

1. Which user has highest connections
2. Which user has lowest connections
3. How many total connections Lucas(id=4) has?
4. How many connections are there between two users
5. Who can introduce Lucas to Adam(id=62)
6. Build a web-based application to support above capabilities (Optional)

## Out of Scope

UI Validations

All User Interface validations / Server-side validations are outside the scope of this project due to time constraints.

E2E Exception Handling

End to End exception handling is outside the scope of this project due to time constraints.

## Assumptions

* Solution will address direct connections for all user stories mentioned in Sec 1.1.
* Develop application using any opensource technology.

## Dependencies

None

# Realized Design decissions

Following table provides the set of design decisions that have been realized as part of implementation

|  |  |  |
| --- | --- | --- |
| S. No | Requirement | Realized Design Decision |
| 1 | Technology Stack | **Decision:** We will be using Spring Boot, JDK 10.0.2, Maven and Angular JS to develop this application  **Rational:** Spring Boot makes it easy to create Spring-powered, production-grade applications and services with absolute minimum efforts.  This enables to develop faster solutions and requires little efforts to setup and run the code. |
| 2 | Angular JS vs Angular 6 | **Decision:** Angular JS will be used as front-end instead of Angular 6.  **Rational:** Though Angular 6 is the latest JavaScript framework, Angular JS is chosen to develop the UI. Angular 6 uses NodeJS as a runtime environment which requires additional steps to configure. This decision is taken keeping time and efforts required to setup and run the code. |
| 3 | Read the files and construct Java object each time to process user requests | **Decision:**  The application will read and process the text file for each user request instead of caching the records.  **Rational:**  As this is prototype application and due to time constraints, I have implemented reading the files for each user request. We have other alternatives to cache the data using third party opensource caching solutions. |
| 4 | All User stories are exposed as REST services | **Decision:** All features will be exposed as REST services  **Rational:** This is to enable loose coupling of components and to enhance components modularity, abstraction and reuse. |
| 5 | Search Algorithm Used to find the path between two users | **Decision:** Breadth First Search (BFS) will be used to implement the solution against Depth First Search (DFS)  **Rational:** BFS is best suited to find the shortest path though it is not memory efficient option when compared to DFS. As our requirement is to find the shortest path, I have considered BFS solution. |

# Solution view

Below sequence diagram provides detail sequence of events to retrieve users with the highest number of connections. The flow for rest of the services is going to be similar and will not be covered as part of this document.

As part of the design, the application reads Persons.txt, Relationships.txt files, and associates the relationship in Person object. The information is stored in HashMap with the key as “id” and values as “Person” object which internally contains Id, Name, and List of connections.

All user stories are exposed as REST services



**Sample REST services output**

**"Output"**: [    
       {    
          **"id"**:"1",  
         **"name”**: “Jakson",  
          **"connections"**:"[ 3, 18, 30, 38, 49, 52, 57, 63, 70, 89]"  
 },  
       {    
          **"id"**:"4",  
         **"name”**: “Lucas",  
          **"connections"**:"[ 7, 27, 37, 38, 47, 51, 61, 79, 81, 95]"  
 }…

      ]

## Retrieve users with highest connections

## 

**URL: GET**: http://{server}:{port}/socialconnect/max

**Input Parameters**

|  |  |  |  |
| --- | --- | --- | --- |
| **Input Parameters** | **Data Type** | **Mandatory Attribute** | **Description** |
| NA | NA | NA | NA |

## Retrieve users with lowest connections

**URL:**  [**GET**: http://{server}:{port}/socialconnect/min](%20GET:%20http://%7bserver%7d:%7bport%7d/socialconnect/min)

**Input Parameters**

|  |  |  |  |
| --- | --- | --- | --- |
| **Input Parameters** | **Data Type** | **Mandatory Attribute** | **Description** |
| NA | NA | NA | NA |

## Retrieve total number of connections for Lucas (id 4)

**URL:**  [**GET**: http://{server}:{port}/socialconnect/{id}](%20GET:%20http://%7bserver%7d:%7bport%7d/socialconnect/%7bid%7d)

**Input Parameters**

|  |  |  |  |
| --- | --- | --- | --- |
| **Input Parameters** | **Data Type** | **Mandatory Attribute** | **Description** |
| Id | Integer | Yes | User Id to retrieve total connections |

## Total number of connections between two users

**URL:**  [**GET**: http://{server}:{port}/socialconnect/{fromId}/{toId](%20GET:%20http://%7bserver%7d:%7bport%7d/socialconnect/%7bfromId%7d/%7btoId)}

**Input Parameters**

|  |  |  |  |
| --- | --- | --- | --- |
| **Input Parameters** | **Data Type** | **Mandatory Attribute** | **Description** |
| fromId | Integer | Yes | From Id to retrieve common connections |
| toId | Integer | Yes | To Id to retrieve common connections |

## Find Path between two users

**URL:**  [**GET**: http://{server}:{port}/socialconnect/findpath/{source}/{destination}](%20GET:%20http://%7bserver%7d:%7bport%7d/socialconnect/findpath/%7bsource%7d/%7bdestination%7d)

**Input Parameters**

|  |  |  |  |
| --- | --- | --- | --- |
| **Input Parameters** | **Data Type** | **Mandatory Attribute** | **Description** |
| source | Integer | Yes | From source Id to traverse |
| destination | Integer | Yes | Destination Id to find the path from source id |

# Visual UI

Belowscreenshots depicts look and feel of all user stories.

This page can be accessed via web page URL : http://localhost:8080/

