

# Requirements Specifications Document

Group Number: 02

Lab Section: L01

Team Members: Kabishan Suvendran, Franklin Tian, Jiawei Yu, Bowen Zhang

The domain of our application consists of the back-end, which is comprised of the Read, Sort, Search, Graph and YouTuber modules, and the front-end, which is comprised of the Display module. By enabling low coupling and high cohesion between these modules, we will be able to achieve an application domain in which operations can be executed in a manner that does not affect other modules. In creating all of these modules and organizing them into a USES hierarchy, we wish to be able to sort, search and display YouTube channels that correspond to the channel preference of the user. The stakeholders of this application are solely those seeking entertainment from YouTube. It is also possible for the YouTube corporation to be involved in this application, as they may see some benefits in our product and may choose to modify their algorithm to resemble our solution. This is highly unlikely, however. The goal of this application is to provide a sequence or list of YouTube channels that correspond to the client's preference. Once this sequence has been generated, it is also possible to display a visual representation that orders the channels based on followers. Ordering and searching for these channels will be handled by the Read, Sort, Search and YouTuber modules. Displaying the representation will be added into the module interface specification (MIS) in the future. The Graph module will use searching algorithms to provide further channel suggestions based on the client's channel preferences. As of now, we expect for all of these functionalities to be incorporated in the final product, but we will update this document in the future. Once the client inputs their channel preference, the front-end will communicate with the back-end modules, which will do most of the heavy lifting. The product will be designed in a manner that enables information hiding and encapsulation. The client will only see a list of YouTube channels and a visual representation. The relationships between the modules can be found in the MIS below. We wish to make life easier for the stakeholders of our product by providing accurate recommendations. These recommendations will be checked against YouTube recommendations.

For the functional requirements, please see the MIS below.

In terms of the non-functional requirements, we will discuss reliability, accuracy of the results, performance, human-computer interface issues and portability issues. In terms of reliability, we will host our solution on a trusted cloud application platform like Netlify or Heroku. These domains provide HTTPS encryption, host the website 24/7/365 and do not log user information. Since there is no login or registration feature, encrypting passwords is not a problem. In terms of the accuracy of the results, we can verify the results of our program with channel recommendations from YouTube itself. To mitigate human-computer interface issues, we will design a webpage that is easy to use, as it will have a huge font size, an input box and button. Lastly, since this is a web application, we will try our best to ensure that it is compatible with most mainstream browsers, but Google Chrome and Mozilla Firefox are our primary concerns.

We will be using Git for version control and will ensure the quality of our product by occasionally conducting JUnit tests on a subset of our sample data to ensure that certain features are working properly. We give a high priority to the searching, sorting and graphing algorithms because they perform the bulk of the operations and are integral in completing the application. The front-end has a lower priority in comparison to the back-end. We do not anticipate any likely changes to the system maintenance procedures at the moment. This document will be updated regularly.

# Search Module

## Module

Search

## Uses

YouTuber, Graph

## Syntax

### Exported Constants

None

### Exported Access Programs

Routine name	In	Out	Exceptions
Search_id	String, sequence of sequences of YouTuber	sequence of YouTuber	
displayStats	String, sequence of YouTuber	String Country, String Category_name, String JoinDate, $\mathbb{N}$ Followers $\mathbb{N}$ Videos	NotFoundException

## Semantics

### State Variables

name: String

youtuberList: sequence of YouTuber

### State Invariant

None

## Access Routine Semantics

Search\_id(n, s)

- transition: name, youtuberList := n, s
- output: out := L : sequence of YouTuber where  $(x : \text{YouTuber} \mid x.\text{getCategory\_name()} = s \Rightarrow L.\text{add}(x))$
- exception: none

displayStats(s : String, l : sequence)

- output: out := <Country, Category\_name, JoinDate, Followers, Videos>
- exception:  $(s \notin l \Rightarrow \text{NotFoundException})$

# Graph Module

## Template Module

Graph( $\mathbb{N}$ )

## Uses

YouTuber, Read

## Syntax

### Exported Constants

None

### Exported Types

Graph = ?

### Exported Access Programs

Routine name	In	Out	Exceptions
new Graph	$\mathbb{N}$ , sequence of YouTuber	Graph	
Vertices		$\mathbb{N}$	
Edges		$\mathbb{N}$	
addEdge	YouTuber, YouTuber		
adj	YouTuber	YouTuber	

## Semantics

### State Variables

vertices:  $\mathbb{N}$

edges:  $\mathbb{N}$

AdjList: sequence of sequences of YouTuber

### State Invariant

None

## Assumptions

The constructor Graph should be called before calling any other methods.

## Access Routine Semantics

new Graph(V, L)

- transition: vertices, edges, AdjList := V, 0, L
- output: out := self
- exception: none

Vertices()

- output: out := vertices
- exception: none

Edges()

- output: out := edges
- exception: none

addEdge(v, w)

- transition:  $(v, w : \text{YouTuber} \mid L : \text{sequence of YouTuber} : (v \in L[0] \wedge L \in \text{AdjList} \Rightarrow L.\text{add}(w)) \vee (w \in L[0] \wedge L \in \text{AdjList} \Rightarrow L.\text{add}(v)))$
- exception: none

adj(Y)

- output: out :=  $(\text{adjacency} : \text{YouTuber} \mid Y : \text{YouTuber}, L : \text{sequence of YouTuber} : (Y = L[0] \wedge L \in \text{AdjList} \Rightarrow \text{adjacency} = L[1..|L| - 1]))$
- exception: none

# YouTuber Module

## Template Module

YouTuber

## Uses

None

## Syntax

### Exported Constants

None

### Exported Types

YouTuber = ?

### Exported Access Programs

Routine name	In	Out	Exceptions
new YouTuber	N, String, N, String, String, String, N	YouTuber	
getCategory_id		N	
getCategory_name		String	
getCountry		String	
getFollowers		N	
getJoinDate		String	
getTitle		String	
getVideos		N	

## Semantics

### State Variables

cat\_id: N

cat\_name: String

country: String

followers: N

join\_date: String

title: String

videos: String

## State Invariant

None

## Assumptions

The constructor “YouTuber” should be called before calling any other methods.

## Access Routine Semantics

new YouTuber(id, n, c, f, j, t, v)

- transition: cat\_id, cat\_name, country, followers, join\_date, title, videos := id, n, c, f, j, t, v
- output: out := self
- exception: none

getCategory\_id()

- output: out := cat\_id
- exception: none

getCategory\_name()

- output: out := cat\_name
- exception: none

getCountry()

- output: out := country
- exception: none

getFollowers()

- output: out := followers
- exception: none

getJoinDate()

- output: out := join\_date
- exception: none

getTitle()

- output: out := title



- exception: none

getVideos()

- output: out := videos
- exception: none

# Read File Module

## Module

Read

## Uses

YouTuber

## Syntax

### Exported Constants

None

### Exported Types

Read = ?

### Exported Access Programs

Routine name	In	Out	Exceptions
read	String	Read	FileNotFoundException
generateYouTuberList		sequence of YouTuber	

## Semantics

### State Variables

filename: String

list: sequence of YouTuber

### State Invariant

None

## Assumptions

The method “read” should be called before any other methods.

## Access Routine Semantics

read(f)

- transition: filename := f
- output: out := self
- exception: File does not exist

generateYouTuberList()

- output: out := sequence of YouTuber
- exception: none

# Sort Module

## Module

Sort

## Uses

YouTuber, Search

## Syntax

### Exported Constants

None

### Exported Access Programs

Routine name	In	Out	Exceptions
MergeSort_Followers		sequence of YouTuber	
MergeSort_Videos		sequence of YouTuber	
QuickSort_Followers		sequence of YouTuber	
QuickSort_Videos		sequence of YouTuber	

## Semantics

### State Variables

s: sequence of YouTuber

### State Invariant

None

### Access Routine Semantics

MergeSort\_Followers()

- output:  $\text{out} := \text{sequence of YouTuber where } \forall i : N | i \in [0..|s| - 2] : s[i].\text{getFollowers()} \geq s[i + 1].\text{getFollowers()}$
- exception: none

MergeSort\_Videos()

- output:  $\text{out} := \text{sequence of YouTuber where } \forall i : N | i \in [0..|s| - 2] : s[i].\text{getVideos()} \geq s[i + 1].\text{get Videos()}$
- exception: none

QuickSort\_Followers()

- output:  $\text{out} := \text{sequence of YouTuber where } \forall i : N | i \in [0..|s| - 2] : s[i].\text{getFollowers()} \geq s[i + 1].\text{getFollowers()}$
- exception: none

QuickSort\_Videos()

- output:  $\text{out} := \text{sequence of YouTuber where } \forall i : N | i \in [0..|s| - 2] : s[i].\text{getVideos()} \geq s[i + 1].\text{get Videos()}$
- exception: none

# Display Module

## Module

Display

## Uses

YouTuber, Search, Graph

## Syntax

### Exported Constants

None

### Exported Access Programs

Routine name	In	Out	Exceptions
User_Input	String	sequence of YouTuber	EmptyException

## Semantics

### State Variables

pref: String

s: sequence of sequences of YouTuber

### State Invariant

None

### Access Routine Semantics

User\_Input(pref)

- output:  $\text{out} := \text{Search\_id}(\text{pref}, s)$
- exception:  $(|\text{pref}| = 0) \Rightarrow \text{EmptyException}$