Requirement

What is good requirement?

Based on IEEE's Guide for Developing System Requirements Specifications in 1998, a requirement is "a description of what the system's customers expect it to do for them, the system's expected environment, the system's usage profile, its performance parameters, and its expected quality and effectiveness" (IEEE, 1998). IBM's booklet, Get It Right The First Time also suggest that requirement should be a communication line between the user and the developer. (IBM, n.d.)

Based on the 2 resource, I have the following requirement for my set of requirement:

- Unique
    - Each requirement must only be stated once.
- Normalised
    - Requirement should not be interdependent on each other.
- Unambigious
    - There should be no room for misinterpretation for each requirement
- Short and precise
    - Each requirement should only be 1 or 2 sentence that encapsule only 1 specification
- Consistent
    - Each requirement must not contradict each other

Eliciting the requirement

I based my work on the 9 elicitation technique BABOK have identified (BABOK, 2015):

1. ~~Interview~~
2. ~~Workshop~~
3. ~~Survey / Questionnaire~~
4. **Interface analysis**
5. ~~Focus group~~
6. ~~Observation or ethnography~~
7. **Brainstorming**
8. Prototyping
9. **Analysing documentation**

After considering the ethic requirement from UCL, some methods(1,2,3,5,6) are excluded from consideration because they involve gathering information directly from people or contacts which I do not have. Looking at my project, I have decided that I should start with a document analysis to establish a sense of "as if". It is important understand what I should create. To ensure the ideas are relevant to my target audience, an interface analysis would also be useful.

Document analysis

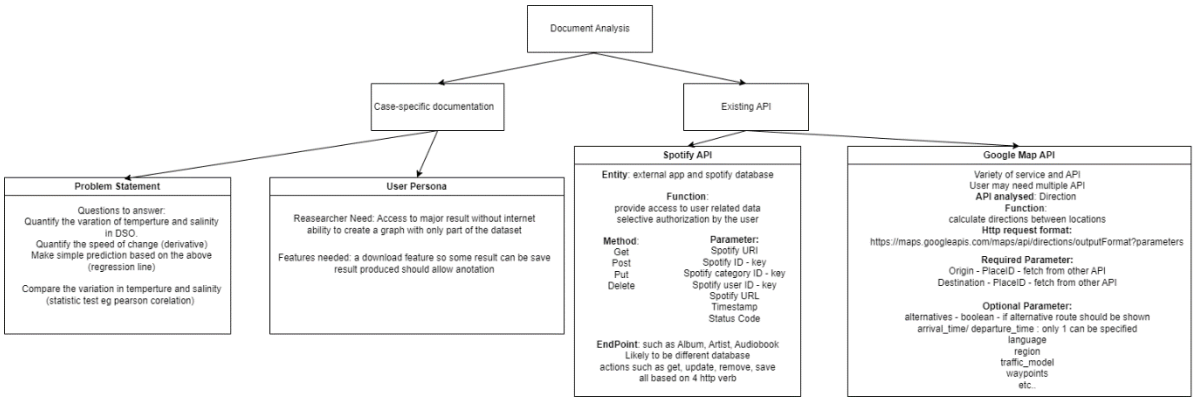
I have started to with document analysis. BABOK included 3 elements in this method:

1. Preparation – identification and preparation of suitable document
2. Document review – studying and documenting relevant details
3. Wrap-up – reviewing and organizing the information

In preparation, I have identified the problem statement from coursework 1 and API documentation from various source, such as Spotify and Google Map, to be a suitable source.

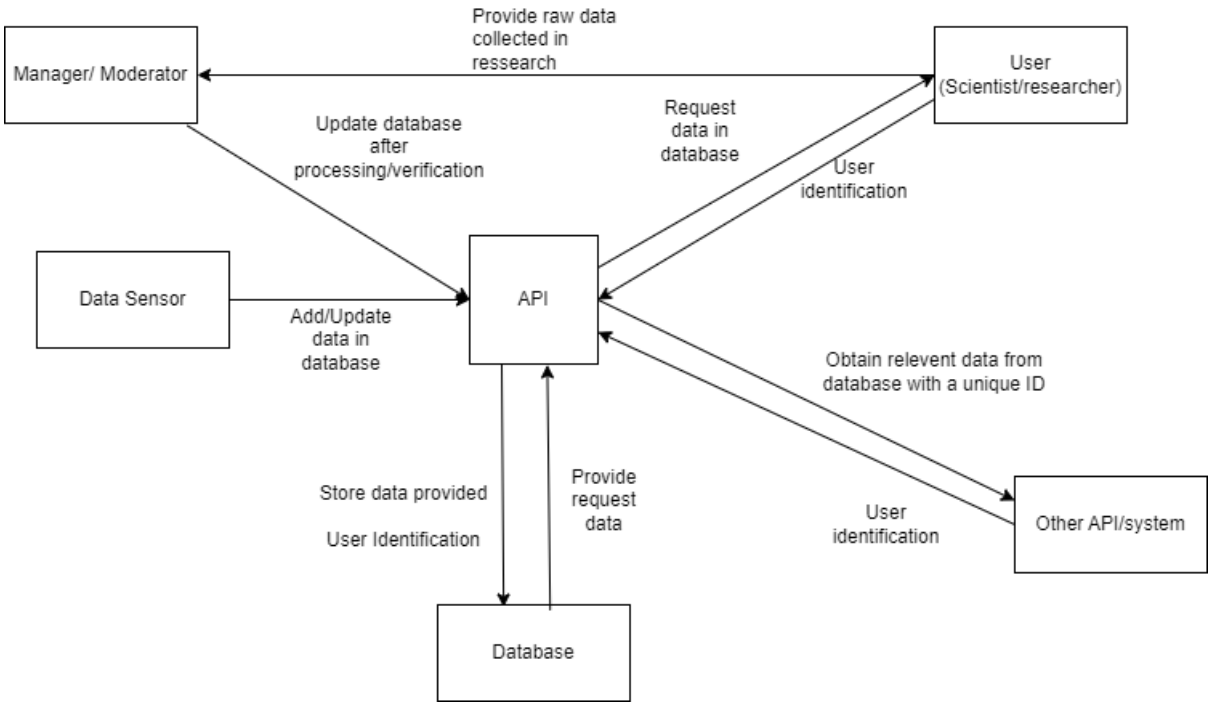
Both analysed API are chosen because I use their service regularly and they have excellent documentation to aid my understanding.

Below is a diagram from the analysis: (Spotify, 2019) (Spotify, n.d.) (Google, n.d.)



Interface analysis

Based on the document analysis, I have drawn a context diagram to describe how my users and data can interact.



Note that although not in the original problem statement, I have decided to allow my API to integrated with other API or system so that some data can be obtain with their unique data ID. Ideally, this is only available if the requester is authorised.

User Story

Based on the analysis done above, I have written the following user stories:

ID	As a …	I want to …	So that I can …
1	Scientist	have my username and password	access the data securely
2	Scientist	access a chosen range of data	perform analysis
3	Scientist	be provided documentation	learn how to command the API efficiently
4	Scientist	see simple statistical analysis done automatically	draw insight easily
5	Scientist	access some of the data offline	continue to work on field trips with no internet access
6	Scientist	search for specific values or date	access relevent data quickly
7	Scientist	provide new entry into the database	collabarate with other researchers
8	Developer	Have all new entry cleaned	provide clean data for my users
9	Developer	be able to intergrate this API into other apps or API	provide extra features in my own app
10	Developer	verify users when they access the data	prevent misuse of my database
11	Developer	update the API regularly	it support any platform the user is on

All these stories are based on the document and interface analysis. The stories are stated in a way that they do not overlap but can be expanded further to give unique and unambiguous requirement. The ID is assigned so the requirements written can be traced back to the user stories.

Table of requirement

A table of requirement have been written with unique IDs to they are traceable.

ID	Requirement	Type	MoSCow
1	**Means to login**		
1.1	Have a username for each individual user	Funtional	
1.1.1	Username is customisable	Funtional	Must Have
1.1.2	Username is unique	Funtional	Must Have
1.1.3	Username is comprehensible for human	Non-functional	Must Have
1.1.4	Each username is tied to a unique primary key	Funtional	Must Have
1.2	Organisation username	Funtional	
1.2.1	Organisation can generate unique username for each user	Funtional	Won't Have
1.2.2	The username generated can be recognised to be under the same organisation	Funtional	Won't Have
1.2.3	The organisation admin can modify usernames	Funtional	Won't Have
1.3	Password	Funtional	
1.3.1	Password can be customised by users	Funtional	Must Have
1.3.2	Check password is longer than specific length	Funtional	Must Have
1.3.3	Password can be reset	Funtional	Must Have
1.4	Alternative mean of log in	Funtional	
1.4.1	Security questions can be used to reset password	Funtional	Can Have
1.4.2	Email address can be used to retrieve password	Funtional	Can Have
2	**Access to information from database**		
2.1	Information on data avalible in database	Funtional	
2.1.1	Show the range of dates with data avalible	Funtional	Should Have
2.1.2	Show date in a format that is comprehensible to human	Funtional	Should Have
2.1.3	Show number of data entry avalible for each location at specified date range	Funtional	Should Have
2.1.4	location shown as an ID or longtitude and latitude	Funtional	Should Have
2.2	Provide data on request	Funtional	
2.2.1	Allow date range to be specified	Funtional	Must Have
2.2.2	Allow location to be specified	Funtional	Must Have
2.2.3	Location can be specified as a ID	Funtional	Must Have
2.2.3	Allow the data column to be specified	Funtional	Must Have
2.2.4	At least 1 of the parameter must be specified	Funtional	Must Have
2.2.4.1	If date is not specified, all data on the specified location/column will be returned	Funtional	Must Have
2.2.4.2	If location is not specified, all data on the specified date/column will be returned	Funtional	Must Have
2.2.4.3	If column is not specified, all data on the specified date/location will be returned	Funtional	Must Have
2.2.5	Return status code after request have been processed	Funtional	Must Have
2.2.6	Return ways to contact support if error have occurred.	Funtional	Must Have
2.2.7	Allow returned data to be saved	Funtional	Must Have
2.2.8	All saved data in req 2.2.7 will have unique ID	Funtional	Must Have
2.2.9	The result should be returned within 60 second	Non-functional	Should Have

3	**Documentation**		
3.1	Provide simple documentation online for general user	Funtional	
3.1.1	documentation on action verb that can be used	Funtional	Must Have
3.1.2	documentation on essential parameter	Funtional	Must Have
3.1.3	documentation on optional parameter	Funtional	Must Have
3.1.4	Provide examples in the documentation	Funtional	Should Have
3.1.5	Provide a standard format for requests	Funtional	Should Have
3.2	Provide detailed documentation for other developer	Funtional	
3.2.1	Example of simple app that make use of API	Funtional	Can Have
3.2.2	Documentation on client libraries	Funtional	Can Have
3.2.3	support different langauges	Funtional	Won't Have

4	**Do simple statistical analysis**		
4.1	Verify if data range specified is sutable for analysis	Funtional	
4.1.1	Dataset can be identified as their unique ID	Funtional	Should Have
4.1.2	Only 2 variable/column should be specified	Funtional	Should Have
4.1.3	User must specifiy which variable is dependent	Funtional	Should Have
4.1.4	If data specified is not suitable, an error code is returned	Funtional	Should Have
4.2	Plot simple graph of specified data range if requested as a parameter	Funtional	
4.2.1	Plot scatter graph by default	Funtional	Should Have
4.2.2	Plot line graph if requested	Funtional	Should Have
4.2.3	Plot bar chart if requested	Funtional	Should Have
4.3	Calculate simple statistical measure for specified data range	Funtional	
4.3.1	calculate mean	Funtional	Should Have
4.3.2	calculate variance	Funtional	Should Have
4.3.3	calculate standard deviation	Funtional	Should Have
4.3.4	calculate median	Funtional	Should Have
4.3.5	Statistical measure can be returned as a number	Funtional	Should Have
4.3.6	Statistical measure can be shown on a boxplot if requested	Funtional	Should Have
4.4	Perform regression analysis on specified data range	Funtional	
4.4.1	Perform linear regression calculate on specified data range	Funtional	Should Have
4.4.1.1	Use least square estimation by default	Funtional	Should Have
4.4.1.2	Use Random sample consensus (RANSAC) if requested	Funtional	Can Have
4.4.1.3	Default parameter value for RANSAC is used unless specified	Funtional	Can Have
4.4.2	Perform non-linear analysis if requested	Funtional	Should Have
4.4.2.1	Use Polynomial regression of 2 degree by default	Funtional	Should Have
4.4.2.2	degree can be specified as parameter	Funtional	Can Have
4.4.2.3	Cross-validation can be used to decide optimal degree	Funtional	Won't Have
4.5	Provide confidence level on prediction	Funtional	
4.5.1	return Coefficient of determination along with prediction	Funtional	Can Have
4.5.2	return warning when values are extrapolated.	Funtional	Can Have
4.6	Compare relations of data	Funtional	
4.6.1	Require 2 range of data to be provide	Funtional	Can Have
4.6.2	Calculate Pearson's correlation coefficient for the data specified	Funtional	Can Have

5	**Offline access to data**		
5.1	Allow download to personal device	Funtional	Should Have
5.1.1	Device must be registered before downloading is allowed	Funtional	Can Have
5.1.2	Dataset as created in req 2.2.8 and req 6.2.3 can be downloaded	Funtional	Should Have
5.1.3	Graphs can be downloaded	Funtional	Should Have
5.1.4	Result returned can be downloaded	Funtional	Can Have
6	**Search for specific values**		
6.1	Search Parameter must be used to narrow search range	Funtional	
6.1.1	The value searched must be specified	Funtional	Should Have
6.1.2	The data type/range searched can be specified	Funtional	Should Have
6.1.3	The data that should be returned can be specified	Funtional	Should Have
6.2	All relevent data need to be returned	Funtional	
6.2.1	The entry ID of all matched result must be returned	Funtional	Should Have
6.2.2	The result of a query can be saved	Funtional	Should Have
6.2.3	A saved query will have a unique ID	Funtional	Should Have
6.2.4	The result of a query can be used to plot graphs in req 4	Funtional	Should Have
7	**New entry into database**		
7.1	A request can be made to add data into database	Funtional	Can Have
7.1.1	The value of data added must be specified	Funtional	Can Have
7.1.2	The column data should be added to must be specified	Funtional	Can Have
7.1.3	The date of which the data is collected must be specified	Funtional	Can Have
7.1.4	The exact location of which the data is collected must be specified	Funtional	Can Have
7.1.5	The userID of the user who made the request must be specified	Funtional	Can Have
7.2	The new data must be verified before entered into database	Funtional	Won't Have
7.2.1	The new data is stored into a another database	Funtional	Won't Have
7.2.2	The new entry is to be verified before merged into the main database	Funtional	Won't Have
7.2.3	User must be authorised befored making such request	Funtional	Won't Have

8	**All data in the database must be clean**		
8.1	The API should reject data that are not sensible	Funtional	
8.1.1	If all values in a column is numerical, any non-numerical new entry is rejected	Funtional	Won't Have
8.1.2	If the data is is over 2 magnitude off from the mean, the new entry is rejected	Funtional	Won't Have
8.1.3	If the date entered is later than the date the new entry is entered, the entry is reject	Funtional	Won't Have
8.1.4	If the location entered is over 1° off from the closet entry, the entry is rejected	Funtional	Won't Have
8.2	The API should provide interface for admins to accept or reject entry	Funtional	
8.2.1	Each admin must have an unique ID	Funtional	Won't Have
8.2.2	The admin which approved an entry should be recorded	Funtional	Won't Have
8.2.3	The interface must not be accessible to general user	Funtional	Won't Have
9	**Ability to be intergrated into an external apps**		
9.1	Generate unique API keys so that external developer can use it	Funtional	Should Have
9.2	Follow and support existing format	Funtional	
9.2.1	Follow HTTP (The Hypertext Transfer Protocol)	Funtional	Must Have
9.2.2	Support HTML	Funtional	Should Have
9.2.3	Support XML	Funtional	Should Have
9.2.4	Support JSON	Funtional	Should Have

10	**User verification**		
10.1	User must enter username and password to be verified	Funtional	Must Have
10.1.1	The username must exist in database	Funtional	Must Have
10.1.2	The password must match the username	Funtional	Must Have
10.2	An API key can be entered to access the database	Funtional	Should Have
10.2.1	The API key entered must match ones that have been generated	Funtional	Should Have
10.2.2	API key can be nullified if there is proof of misuse	Funtional	Should Have
11	**Regular update**		
11.1	The API must follow the latest relevent protoccol	Non-functional	Can Have
11.2	API must be updated regularly to fix bug	Non-functional	
11.2.1	Minor bug should be fixed within 1 month	Non-functional	Can Have
11.2.2	Major bug should be fixed within 1 week	Non-functional	Can Have
11.3	The API should be functional 99.9% of the time	Non-functional	Should Have
11.4	At least 5 older version of the API should be avaliable to users	Funtional	Can Have
11.5	Each version must have a unique version number	Funtional	Can Have
11.6	Maintain a change log so changes can be traced	Funtional	Can Have

Prioritisation of requirement

To start with the technique, I must first know what the bare minimum is. I have defined the essential function of my API to be "allowing authorised users to access numerical data of a specified range". If I am working in a team, this will be discussed with the group. As I am working as an individual now, this is based solely on my problem statement, and APIs I have analysed.

Then, the features that complement my essential feature were categorised as should have. This is most of my requirements. The difference between should have and can have is that the API can stand out without the can haves but will not be very useful without the should have.

Both can have and won't have would be nice features to have. Won't have are the feature that are not time-efficient or cost-efficient. However, I do not have the experience to estimate the difficulty in implementation at this stage, therefore, the differentiation between the 2 were mainly the amount of knowledge I lack in the implementation. Some won't have function are in that category because it is useful to only a minority of my users.

The limitation of this method is that there is no subjective method of scoring, the items are placed in category through objective judgement. This could be mitigated by addition of another method, such as priority poker, or discussion within a group. My options are limited since I have a very long list of requirements and I am working alone.

Therefore, MoSCoW is the only technique I have used. To verify if it was a sensible decision, I have counted the number of requirements in each category. It was summarized in the table on the right. I found the numbers to be sensible, with most requirement being should haves. I did not see the need to supplement with another technique.

MoSCoW	Count
Must Have	26
Should Have	42
Can Have	25
Won't Have	16
Total	109

Reference list

BABOK (2015). *Elicitation*. [online] BABOK Page. Available at: https://babokpage.wordpress.com/elicitation/.

Google (n.d.). *Getting directions through the Directions API*. [online] Google Developers. Available at: https://developers.google.com/maps/documentation/directions/get-directions.

IBM (n.d.). *https://www.ibm.com/docs/en/SSYQBZ_9.7.0/com.ibm.doors.requirements.doc/topics/get_it_right_the_first_time.pdf*. [online] Available at: https://www.ibm.com/docs/en/SSYQBZ_9.7.0/com.ibm.doors.requirements.doc/topics/get_it_right_the_first_time.pdf [Accessed 29 Nov. 2022].

IEEE (1998). *1233-1998 IEEE Guide for Developing System Requirements Specifications*. IEEE.

Spotify (2019). *Web API | Spotify for Developers*. [online] Spotify.com. Available at: https://developer.spotify.com/documentation/web-api/.

Spotify (n.d.). *Web API Reference | Spotify for Developers*. [online] developer.spotify.com. Available at: https://developer.spotify.com/documentation/web-api/reference/#/.