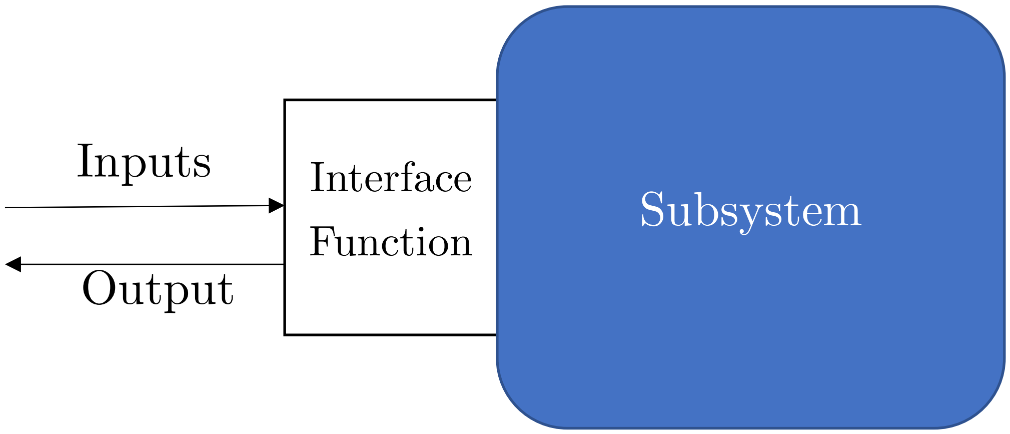
How to write the interface specification for your subsystem/part of the project?

1) For Dactar and Jeevan's part (cuz it's simpler)
The interface should be as depicted below:



Things to note about the specification function:

i) There shall be one and only one interface function to the subsystem. Design your entire subsystem as a high-level object with exactly one public method: the interface function.

ii) The input to the subsystem shall be of the form: subsystem_name.input(a,b,c...) where a, b, c are the parameters. For each parameter p, define the data type(s) accepted and if any IO is involved, also denote the type of file and file extension(s) accepted.

iii) The output of the subsystem shall be the sole return of the output function. The returned value must be immutable and shall be of a predefined data type (not subject to change). All unexpected control flows must throw exceptions and shall not return values. No assumptions shall be made of how the output shall be modified or used by other subsystems.

iv) The immutability condition of iii. requires you to cast any data structure allocated on the heap (eg. list, matrices) to a static data structure (eg. tuple or a tuple of tuples). In case of object return, (for eg. a dog object), make sure that its attributes are not allocated on the heap. (eg. if an object contains a list, the list should be cast to a tuple before returning).

v) The types and causes of exceptions thrown by the function shall be enumerated in the document. The causes shall be labeled as follows:

a. Due to error in an input parameter: PARAM_ERR[n] where n represents either the position or the keyword that refers to the parameter that is under consideration.
b. Due to some internal problem which can be seen even if all the parameters are of the correct format. Some unexpected exception. (eg. buffer overflow, disk space full etc.): INTERNAL_ERR.

*The difference between (a) and (b) is that (a) is the caller's problem whereas (b) needs to be dealt with by the subsystem.

A format of this document is depicted on the next page.

Subsystem Name: ....

Subsystem Engineer: ....

Input Parameters:

S.N.	Parameter name	Data Type(s)	Description
1	vid_path	String	The complete path to the input video file
2	vid_codec	string	The codec string used to compress the video file. Must be one of ‘h.264’, ‘mpeg-4’, ‘h.265’ or ‘hevc’.
...	...	...	...
...	...	...	...

Output:

Name: detected_vehicles

Data type: tuple(VehicleImage)

Class VehicleImage{data: byteArray, vehicleType: string, colorcode: integer }

*vehicleType must be one of ‘car’, ‘bus’, ‘truck’, ‘bike’, ‘scooter’ or ‘tempo’

Exceptions:

S.N.	Exception Name	Description	Probable Cause	Error Code
1.	IOException	Read/write exception on disk.	Caused if either the vid_path parameter is wrong or if permissions are not available.	PARAM_ERR[0]
2.	DecodingException	Error in decoding the video file.	Caused if the file read doesn’t match the codec entered.	PARAM_ERR[1]
3.	NullReferenceExceptio n	The file handle is not available.	Caused by some internal IO error.	INTERNAL_ERR

With this specification in hand, any other subsystem can call the subsystem with the function:

detected_vehicles: tuple(VehicleImage) = *subsystem_name*.input(“c:/program files/video.mp4”, “mpeg-4”).