SURM
Digital Assistant

By N Paul

V 1.0.0

< Developer guide and User manual >

List of contents

1. Basic Introduction

1.1 Full-Form:

What is SURM?

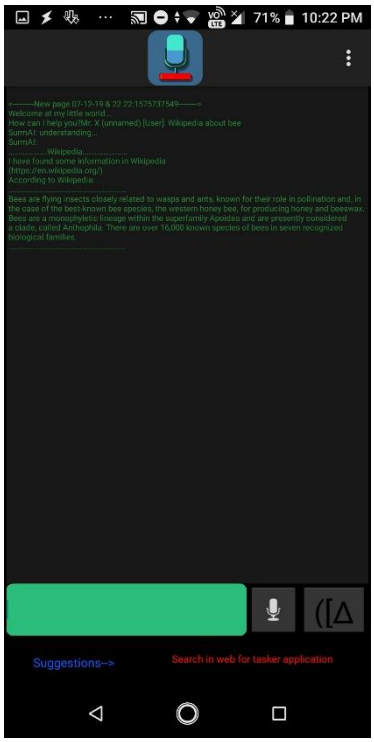
SURM is a virtual-digital assistant program. Which can understand user commands-queries, perform them as tasks or services and even can answer them accordingly with the help of multiple integrated offline executable function modules as well as some online API services based function modules, such as Wolfram Alpha, Wikipedia, OpenWeather etc.

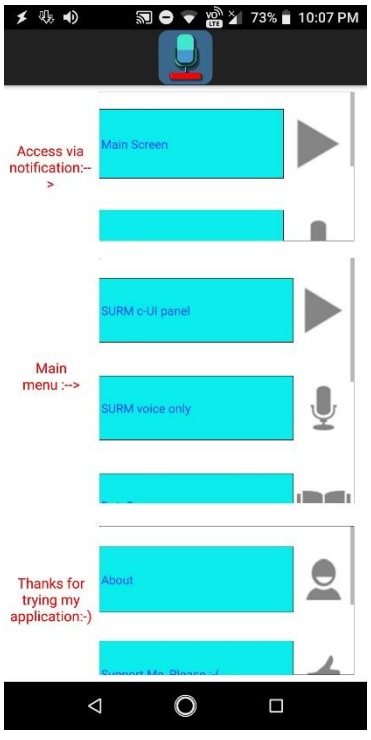
SURM is short form of the term: **S**implified **U**p-to **R**ecognizable **M**odules. We'll talk about the term later.

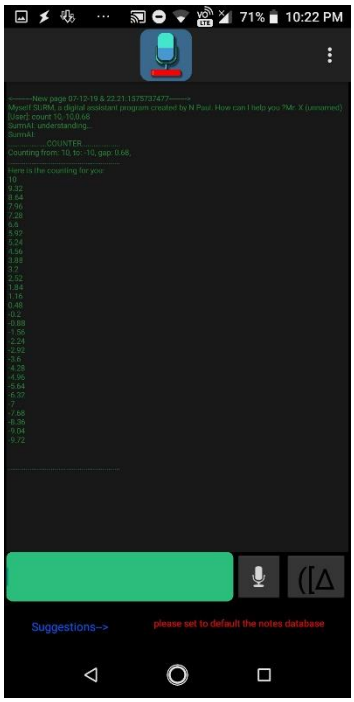
Examples Commands:

- Please select any option Ram,Sam,Zodu,Modhu
- Can you Select any number from -670 to -9850
- Can you turn on Bluetooth
- Please calculate 150 + Sin(45)
- What is the population of Kolkata
- Give me a random fact
- Give me information about Sum
- Count -50,50,9.5
- Take selfie
- Open youtube

And many more…..





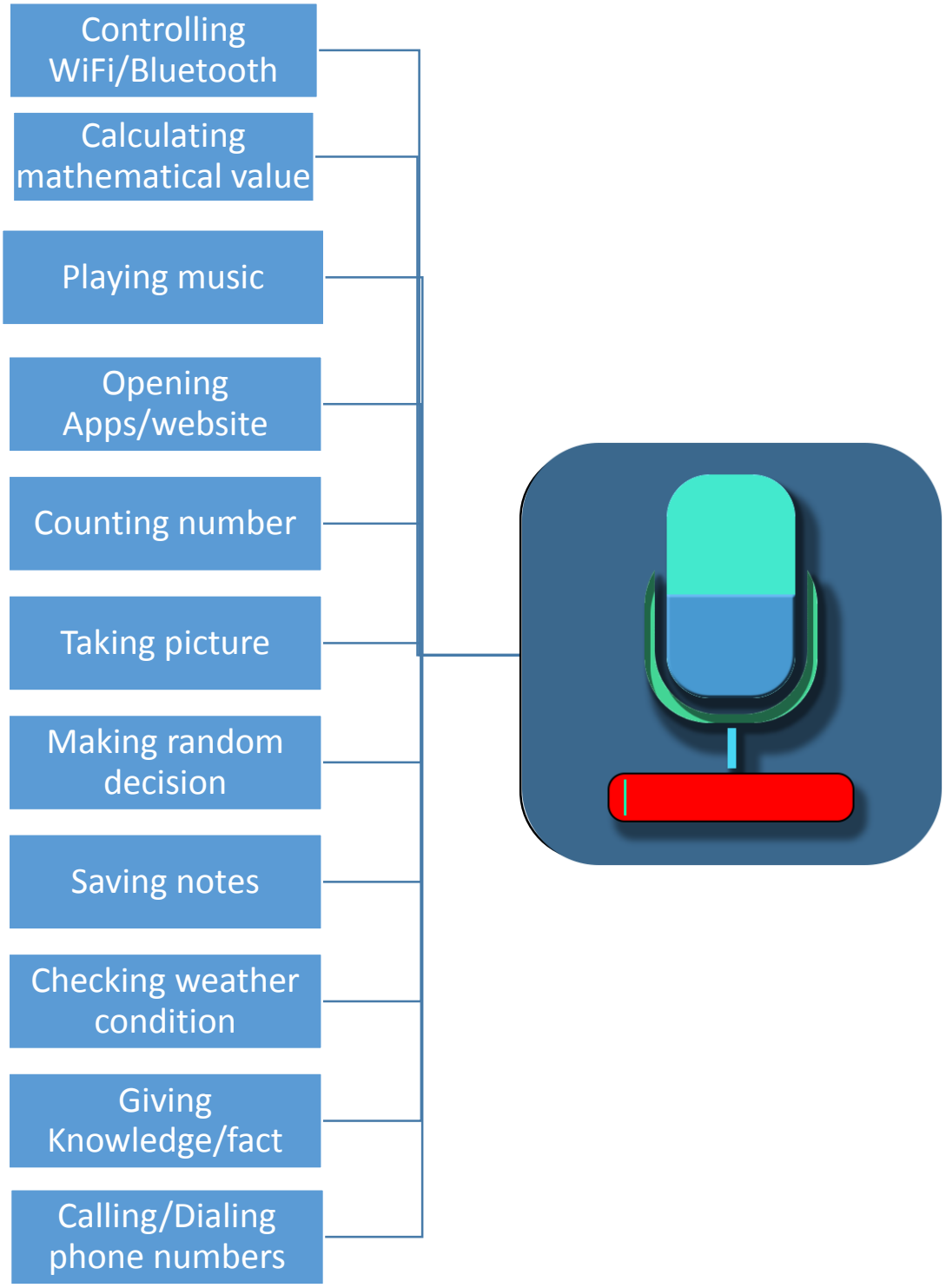


1. Basic Introduction

1.2 Basic Features List:

What it can do?

SURM can perform different types of task for end users. As long as the right module(s) and the right keywords are available, SURM can solve all types of tasks, from simple to relatively complex.

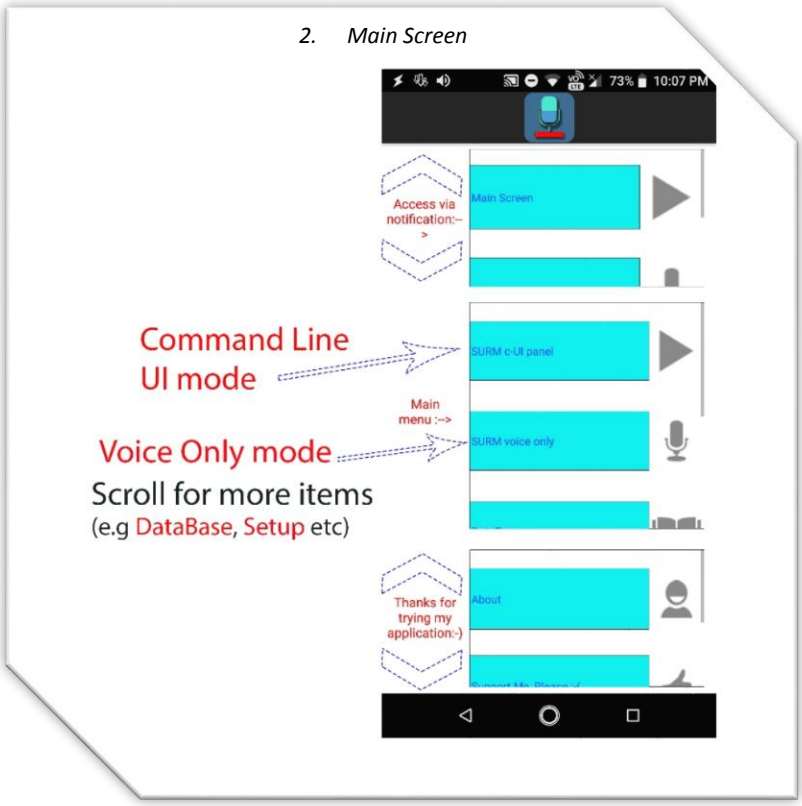


1. Basic Introduction

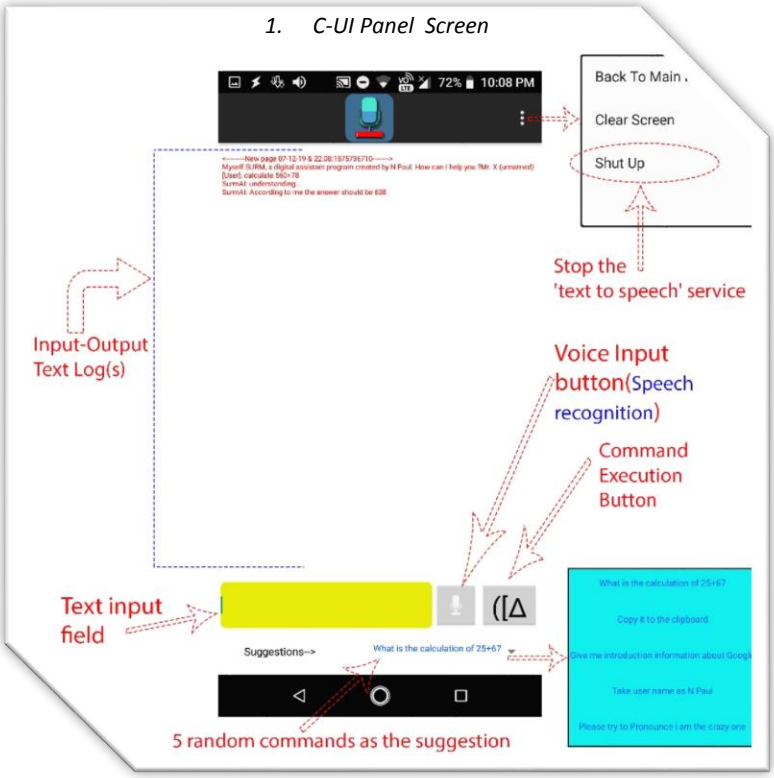
1.3 User-Guide:

How to use SURM (User-Guide):

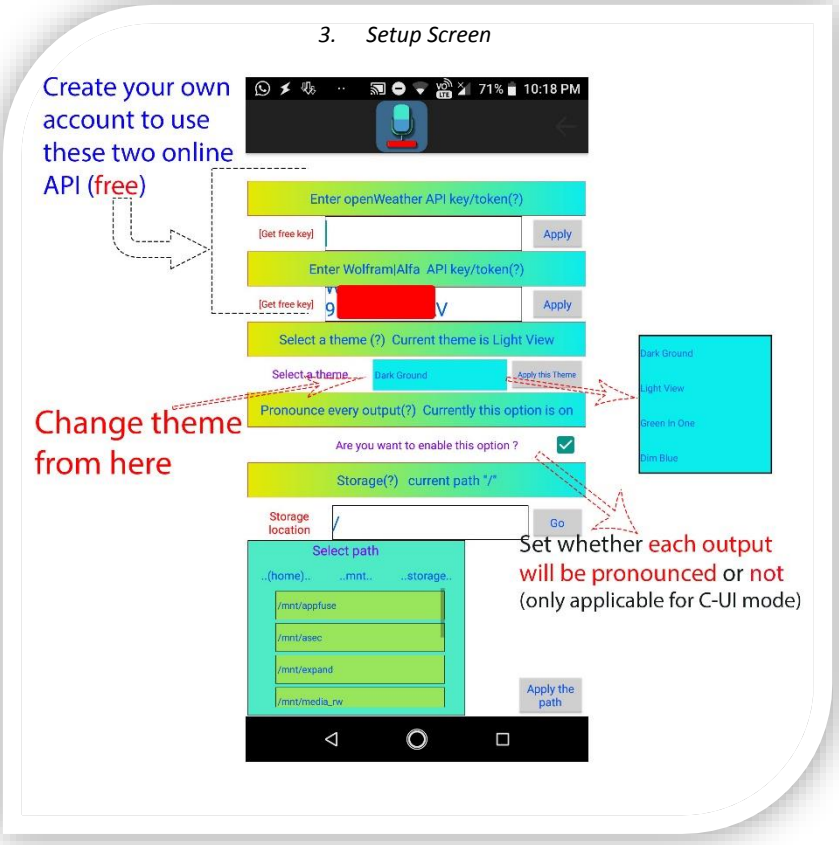
2. *Main Screen*



1. *C-UI Panel Screen*



3. *Setup Screen*



1. Basic Introduction

1.4 About Source Project files:

About app compatibility: This application's icon and contains are designed and developed by me (N Paul). This program is best suitable for stock android with 480x800 P, **HD** (720x1280 p) and **FullHD+** (1080x2160 p) resolution's display. This is an open-source project (originally made by me, N Paul) and anybody can view, modify and rebuilt it. To download the project source file visit my GitHub profile https://www.github.com/nirmalpaul383/ If you want to support me please give like to our Facebook page https://www.facebook.com/a.new.way.technical/

About source project: This application is made with Tasker (https://play.google.com/store/apps/details?id=net.dinglisch.android.taskerm) and with Tasker app factory (https://play.google.com/store/apps/details?id=net.dinglisch.android.appfactory). To install and run 'SURM' you will not need any of this application. However, if you want to 'view' or 'modify' source file you 'will need' Tasker application and if you want to 'export your own modification' then you 'will need both' Tasker and its extension app Tasker app factory.

Note: (1) Tasker app has a wide range of collections of features and it can export its features via an android package file(.apk). But Tasker is not a professional android app making tool like "Google Android SDK". In fact, purposefully Tasker is an Android automation app.

(2)This project is only compatible with Tasker and this project is not directly compatible with "Google Android SDK"

(3)The exported application from the Tasker may or may not fully compatible with all devices especially with various screen sizes, android versions, device's brand and any other various specification of devices.

1. Basic Introduction

1.5 Project Directory Guide & Thanks to:

Here is a directory and file guide:

- **Project SURM**(folder)>**Guide Folder**, **Media Folder**, **Project and Program Folder** ;
- **Guide Folder**(Contains Guides)> **Developer guide and User manual**(Folder containing user guide files), **Guide and Screenshot**(Folder containing screenshots and guide picture files) , **Keywords Books in Text files**(Folder containing all keywords in .txt files);
- **Media Folder**(Contains several media files e.g logo files, trailer video)
- **Projects files and Pre_made apks** Folder(Contains readymade apks and main source project files)>Select your preferred screen resolution folder(e.g **FullHD_Plus**)>**Apk folder**(Contains readymade application file), **Project Folder**(Contains Tasker's Project file);

Apk file name should be "SurmAI_Resolution_Screen.apk" (e.g "*SurmAI_FullHD_Screen.apk*")

Project file name should be "*SurmAI__.prj.xml*"

If you want to support please like my facebook page https://www.facebook.com/a.new.way.Technical/

Thanks to: *Some important online APIs that have been used in SURM:*

Web address:	**Used for:**	**API Guide:**
https://uselessfacts.jsph.pl/random.txt?language=en	Get Random Facts	https://uselessfacts.jsph.pl/
https://sv443.net/jokeapi/category/Any?format=xml	Get Random Jokes	https://sv443.net/jokeapi/
https://ipapi.co/json	Get Location By IP	https://ipapi.co/api/
https://api.openweathermap.org/data/2.5/weather?&units=metric	Get Weather Data	https://openweathermap.org/api
http://api.wolframalpha.com/v2/query?input=	Get answer from Wolfram Alpha	https://products.wolframalpha.com/api/
https://en.wikipedia.org/w/api.php?format=xml&action=query&prop=extracts&exsentences=3&explaintext=&titles=	Get Information From Wikipedia	https://en.wikipedia.org/w/api.php

2. Modules and Keywords Structure

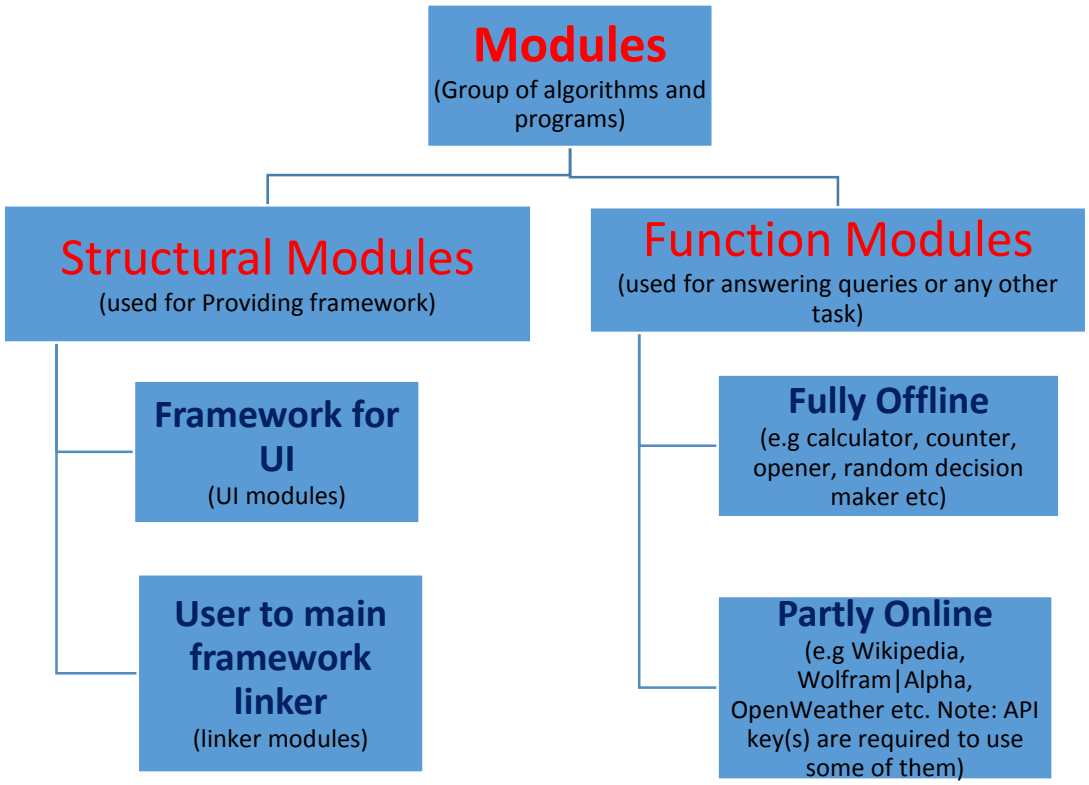
2.1 Introduction:

Modules and keywords

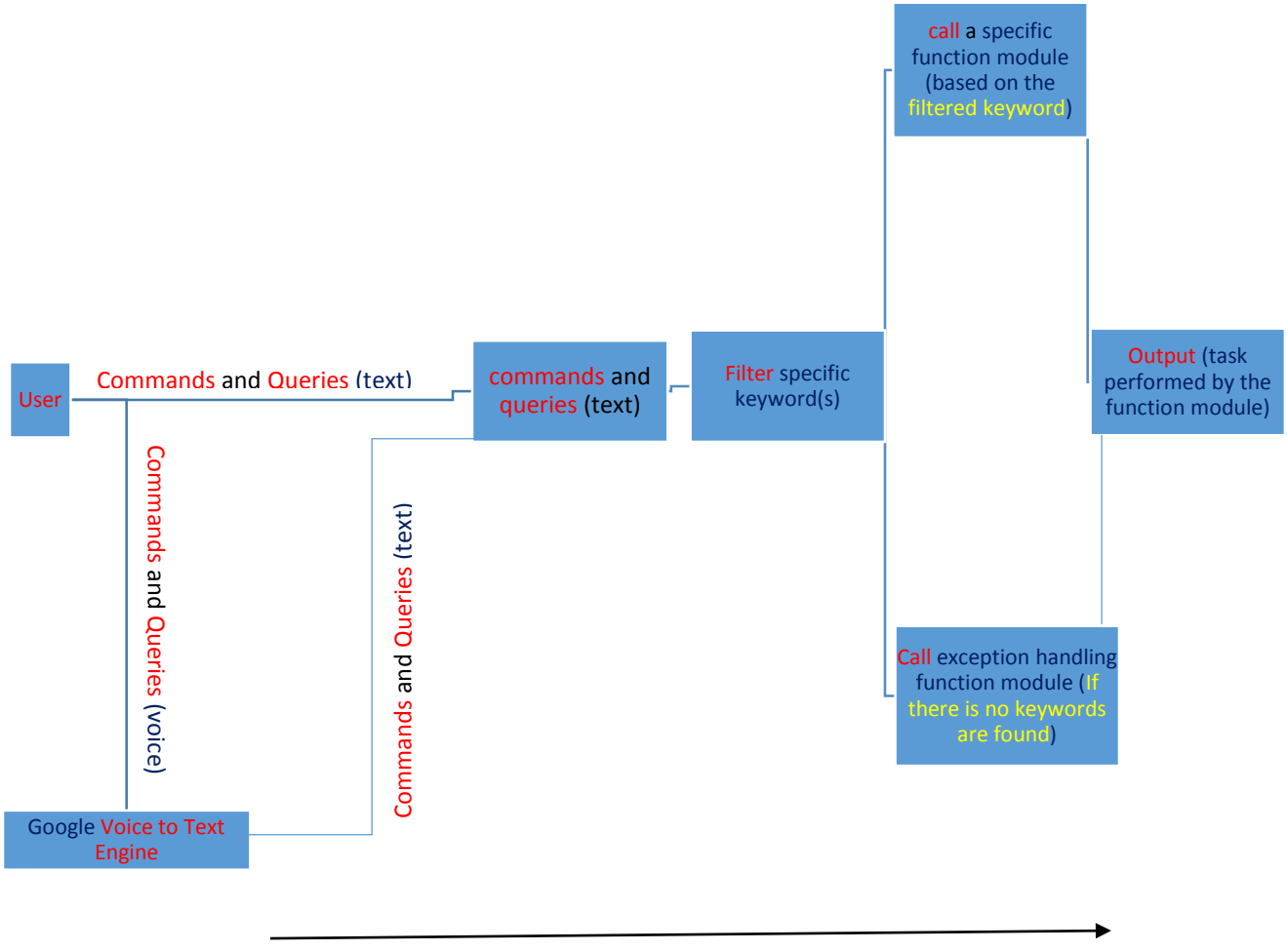
Function Modules are a group of algorithms and programming flow-logics, which are responsible for solving/answering and executing a particular types of queries and commands. Modules are also independent in nature, for which we can easily modify/update any particular modules without disturbing other modules. SURM is highly based on modules and keywords structure system.

Just like the function module, there are also some structural modules available. They do not solve queries just like function modules, instead, they provide structure to the entire SURM and make the program flow in the right direction.

By default there are almost 20 to 25 built-in function modules and some structural modules are available, but as an open-source project you can add more to it and extend its capabilities.



On the other hand, whereas keywords are used to call a specific function module for a specific type of work. With the help of various structural modules, the entire SURM framework is designed in a way that whenever the user gives some command/queries to it, it will automatically filter some specific keyword(s) from that command/queries and then use those keywords to call a particular function module for solving/answering.

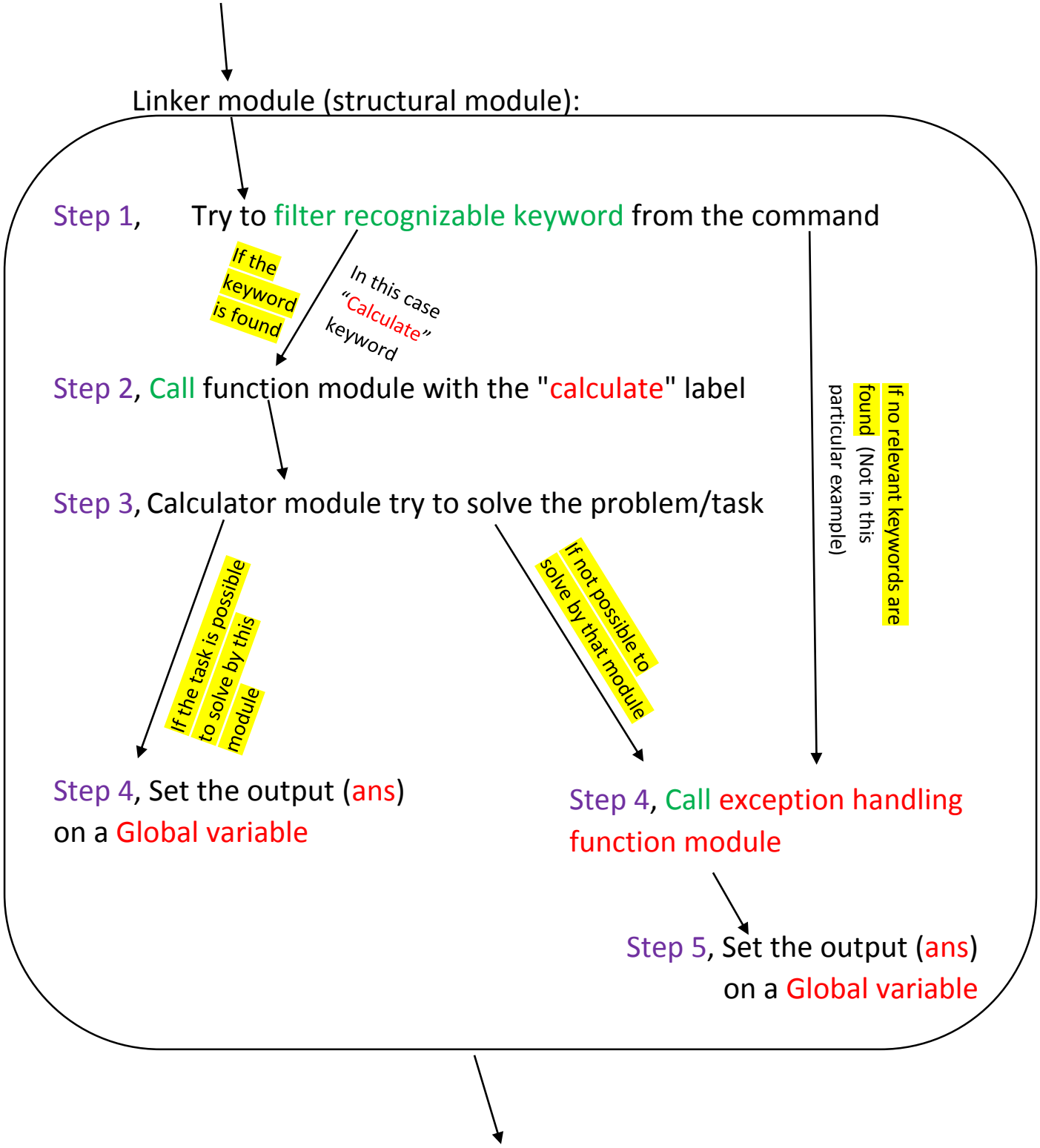


2. Modules and Keywords Structure

2.2 Sample Structure (Using of Keywords & Modules, examples):

For example, (Example 1)

The user: "calculate 580 +980"



Provide/Give the output of information(s) from the Global variable (in this case 1560)

For example, (Example 2)

The user: “Please turn on the BlueTooth please…”

Linker module (structural module):

Step 1, Try to filter recognizable keyword from the command

If the keyword is found

In this case “BlueTooth” keyword

Step 2, Call function module with the "Bluetooth" label

Step 3, Bluetooth module with additional keyword “on”

if the task is possible to solve by this module

If not possible to solve by that module

If no relevant keywords are found (Not in this particular example)

Step 4, Do the action (turning Bluetooth on) and Set the output (action completed) on a Global variable

Step 4, Call exception handling function module

Step 5, Set the output on a Global variable

Provide/Give the output of information(s) from the Global variable (in this case action completed)

For example, (Example 3)

The user: “What is the population of Kolkata?”

Linker module (structural module):

Step 1, Try to filter recognizable keyword from the command

If the keyword is found

In this case no recognizable keyword

Step 2, Call function module (by default, there is no such module to solve that)

Step 3, That module try to solve the problem (in case you have made such module)

if the task is possible to solve by this module

If not possible to solve by that module

If no relevant keywords are found (as this particular example)

Step 4, Do the action (not in this particular case) and Set the output (not in this case) on a Global variable

Step 4, Call exception handling function module (in this case using online Wolfram|Alfa API)

Step 5, Set the output on a Global variable

Provide/Give the output of information(s) from the Global variable (in this case …4.497 million…)

2. Modules and Keywords Structure

2.3 Module Calling By Keyword Matching (Sample Program Syntax):

In the final version of SURM, the task of both step 1 and step 2 is done within a single structural module. That is called Phase_action_keywords (named by me, N Paul☺). Phase action keywords tries to match at least one or more keyword(s) from the command and then accordingly calls a function module. It is mainly a single direction if-else conditional flow path. Just like this programming syntax:

```
if (condition) {action to be done if condition is true;}

else if (condition) {action to be done if this condition is true;}

else {action to be done if no above conditions are true;}
```

For example,

```
if (command has ‘wifi’ or ‘wi-fi’ keyword) {

Call Wi-Fi function module;

}

else if (command has ‘calculate’ or ‘calculation’ or ‘sum of’ keyword) {

Call calculator function module;

}

else if (command has ‘wikipedia’ or ‘information’ keyword) {

Call Wikipedia function module;

}

else if (command has ‘pronounce’ keyword) {

Call pronounce function module;

}
```

2. Modules and Keywords Structure

2.4 Limitation and Limitation Prevention steps:

Flaws and limitations of this structure:

This simple structure can execute the program of a specific module based on the condition (in this case keywords matching) we will provide. But there are two major problems with it.

- The program will always be performed from top to bottom. So every single time of executing, a comparatively lower portion of the flow will get low priority than a higher portion of the flow. It can be caused significant performance difference between module executing at the higher portion v/s module executing at a lower portion.
- And the second problem is the keyword(s) overlapping issue. For example in the above if-else statement when a user command is "SURM turn the wifi on please" then, 1st if condition will be true and Wi-Fi module will be performed. But when the command is "Please pronounce 2.4Ghz wifi is also a wireless method of data transferring" then, it will satisfy both the 1st and 3rd if condition (because it has both 'pronounce' and 'Wi-Fi' keyword) but as this is a single flow path structure, only 1st if-then action will be executed (i.e. calling Wi-Fi module) [instead of executing the 4th if-then action (i.e. calling pronounce module)].

Error Prevention steps:

Unfortunately in the single straight line if-else flow structural, we just can't avoid performance difference issues. Although it may be not much noticeable for 20 to 25 modules (I mean for a short amount of modules). Maybe in the future, I or someone else will implement any other bug-free structure in the SURM.

On the other hand, by implementing 'do not match condition(s)' beside the match condition(s), we can easily prevent to happen keywords overlapping issue.

For example, in the if-else statement if we put both ‘match condition’ (‘wifi’ or ‘wi-fi’ keyword) and also ‘don’t match condition’ (‘pronounce’ keyword) and join them with 'and logic gate', then we can more perfectly specify our condition satisfaction level.

For example,

if (command has ‘wifi’ or ‘wi-fi’ keyword && command does not has ‘pronounce’ keyword) {

Call Wi-Fi function module;

}

else if (command has ‘calculate’ or ‘calculation’ or ‘sum of’ keyword && command does not has ‘pronounce’ keyword) {

Call calculator function module;

}

else if (command has ‘wikipedia’ or ‘information’ keyword && command does not has ‘pronounce’ keyword) {

Call Wikipedia function module;

}

else if (command has ‘pronounce’ keyword) {

Call pronounce function module;

}

:. When the command has ‘wifi’ or ‘wi-fi’ word(s) and also does not has ‘pronounce’ word in its sentence, only then the 1st condition will satisfy. Otherwise, the system will go to the next conditions and verify if, condition-wise they are satisfied or not. [In this case "Please pronounce 2.4Ghz wifi is also a wireless method of data transferring" command can only satisfies 4th condition of the if-else statement (i.e. calling pronounce function module)] :. Our 2nd major problem just have been solved☺.

3. Variables

3.1 Introduction to variables:

Variables: Variables are used for storing data/values. And if needed, those data can be easily retrieved via variables. They actually act like data holders. You can find more information about variables on the internet, but here is some basic information about variables that are related to SURM.

Although variables can be divided in many ways, we generally divide them into two categories depending on the scope of access. First one is called a ***local variable***, which can only be accessed 'within the specified function'(where it is declared) or locally. And the second one is called ***Global Variables***, which can be accessed 'from any function'(wherever the declaration is made) or globally. In SURM, we used both local and global type variables. Where local variables have been used to properly perform internal tasks and global variables have been used to establish a link between tasks. For example, %counter, %counter11, %counter12 are local variables of the counter module and they are used to perform the counter module correctly. On the other hand, %My_quere, %AI_outcome are the global variable used to collect the user's command as input and to publish the result of the counter module as the output of the user's command, respectively.

3. Variables

3.2 Declaring Rules:

Rules for declaring variables in tasker:

"Variables which have an **all-lower-case** name (e.g. %fruit_bar) are ***local***, meaning that their value is specific to the task or scene in which they are used.

Variables which have **one or more capital letters** in their name (e.g. %Car, %WIFI) are ***global***, meaning that wherever they are accessed from the same value is returned." – Tasker's Variables help page

3. Variables

3.2 Some important global variables:

Although multiple global variables have been used in SURM, only 4 of these global variables are used to link a new module. Among other variables, some variables are used to properly complete the theming features or C-UI mode while on the other hand some variables are used to store data temporarily. Click Here To See Those 4 Global Variables.

4. Keywords Matching Mechanism

(used in SURM)

Matching mechanism: To know about the availability of a particular keyword in the command, we use the matching mechanism. In programming, sometimes this mechanism is also referred to with the term 'pattern matching'. In Tasker we have 2 types of matching mechanism *(i) Simple matching (~)* and (ii) Regex Matching (~R). Although in most cases we have only used simple matching. For more info it is recommended to visit Tasker's Pattern Matching help page. To avoid any letter case (low/high/camel) issues during the matching process, the system converts all user commands to lower-case commands and then starts the matching process.

In simple matching, we can use the wildcard character (e.g like '*'), which can act as a placeholder of any number of any unspecified character. For example: 'dog' will match (~) against 'dog'

But 'dog' will not match (!~) against 'the dog' or 'dog is' or 'the dog is' or 'dogfish' or 'hotdog' or 'endogamy'

To match them we need a symbol that can represents one or more characters.

'* dog' will match (~) against 'the dog' or anything that ends with ' dog'

'dog *' will match (~) against 'dog is' or anything that starts with 'dog '

'* dog *' will match (~) against 'the dog is' or anything that has ' dog ' inside it

'dog*' will match (~) against 'dogfish' or anything that starts with 'dog' [may or may not have space (' ') after 'dog']

'*dog' will match (~) against 'hotdog' or anything that ends with 'dog' [may or may not have space (' ') before 'dog']

'*dog*' will match (~) against 'endogamy' or anything that has 'dog' inside it [may or may not have space (' ') before and after 'dog']

To use all match parameters in just one match condition, we can use slash (‘/’) as the divider.

For example: ‘dog/* dog/dog */* dog */dog*/*dog/*dog*’ will match against ‘dog’ or ‘the dog’ or ‘dog is’ or ‘the dog is’ or ‘dogfish’ or ‘hotdog’ or ‘endogamy’ (all of the above).

But there is one small problem, if we use 'dog*/*dog/*dog*' [that is ‘dog’ and ‘*’ with no space (‘ ‘)], it will overlap any word(s) or sentence that has 'dog' inside it. E.g. ‘dog*’ will match both ‘dog is’ and ‘dogfish’ words; ‘*dog’ will match both ‘the dog’ and ‘hot dog’ words; ‘*dog*’ will match both ‘the dog is’ and also ‘endogamy’ words.

:. Ignoring space matching can create huge difference in meaning, so we will try to use space (‘ ‘) in matching parameters as much as possible.

5. The Sub-question Mechanism *(Sub Keywords):*

Counter-Question from SURM (Sub Keyword): In general, when we (user) ask SURM to do something, SURM completes the task accordingly. But sometimes the task is not clear to the SURM, as a result, SURM asks the user for some more information about that and then completes that task. In simple words sometimes SURM needs more information to do the job. More information = more command = more keyword = More specifications on what to do. For example,

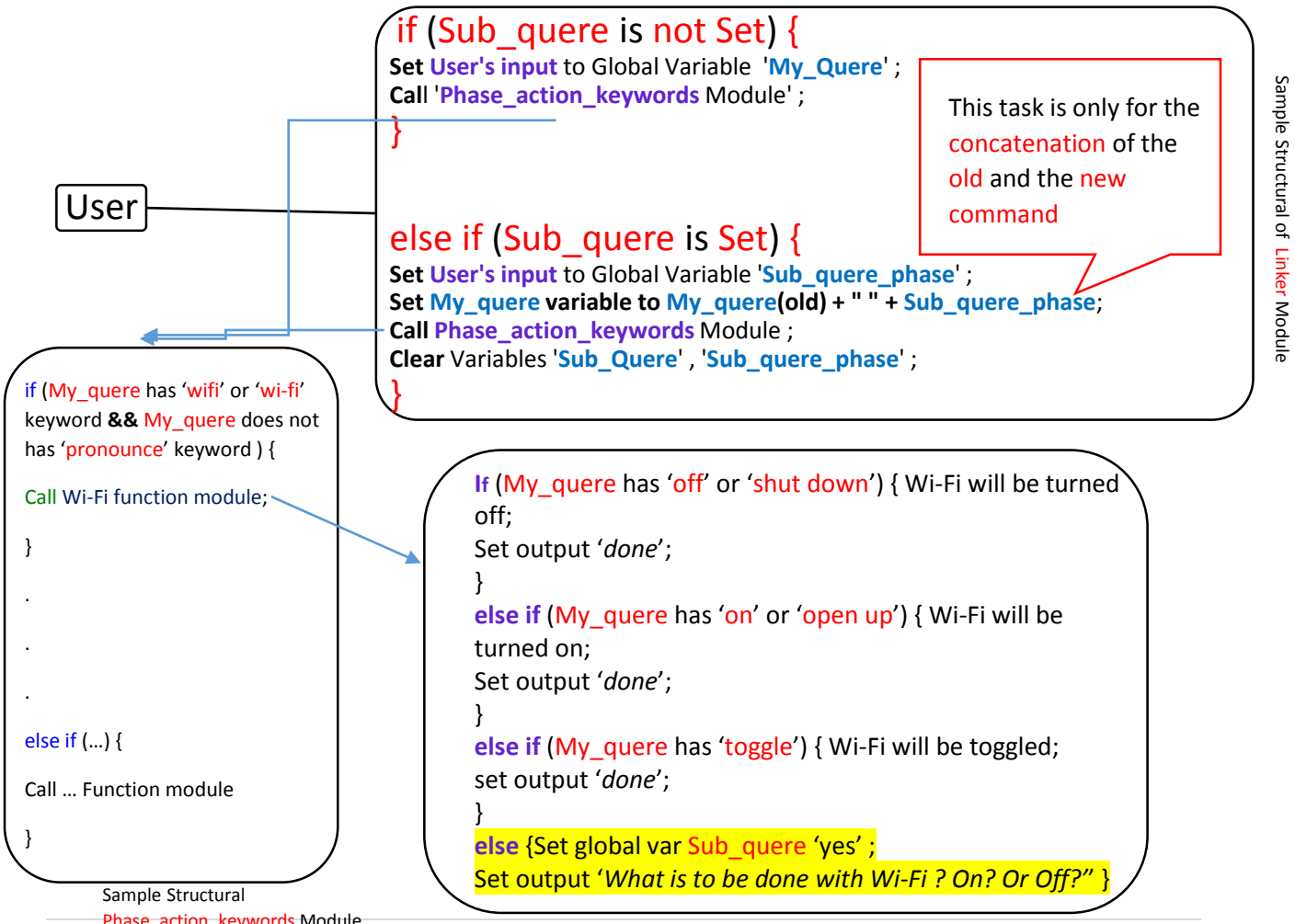
User command: “turn on the Wi-Fi please”. SURM will turn on the Wi-Fi.

When user command: “off the Wi-Fi”. SURM will turn off Wi-Fi.

But, when User command: “do something with the Wi-Fi”. In this case, SURM knows that there is something involved with the Wi-Fi task, but still has no idea what to do with the Wi-Fi (should it be turned off or turned on). So SURM will simply call the Wi-Fi module and then that module will ask the user a counter-question what to do with WiFi. The

user will provide further instructions on the task (in this case on or off or toggle) and SURM will use those to properly execute the operation.

For this purpose, we have used two global variables first one is named "Sub_quere" and the 2nd one is "Sub_quere_phase" (yes, I know there is a spelling mistake😂. It should be 'query' and not 'quere'. Please take that just as variable naming. ^_^). In general user command is stored on a global variable called My_quere and Phase_action_keyword module is also based on My_quere variable. But when the Sub_quere variable is set, then user command is temporarily stored on another global variable named Sub_quere_phase. This Sub_quere_phase variable is used for rewriting old My_quere variable. When the rewrite is done, SURM calls the Phase_action_keywords module again to finish the task. But since now My_quere has the right keywords (old+new), SURM will no longer have to worry about the task specification.

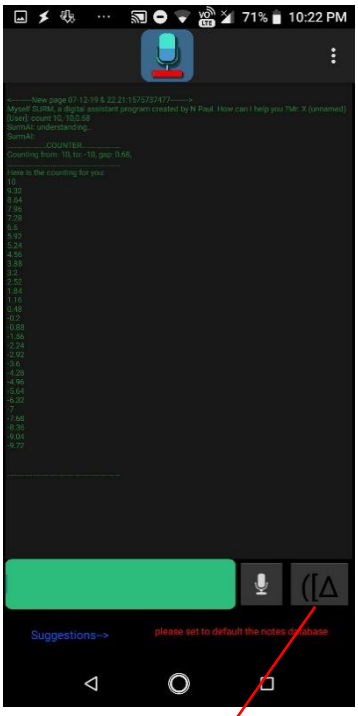


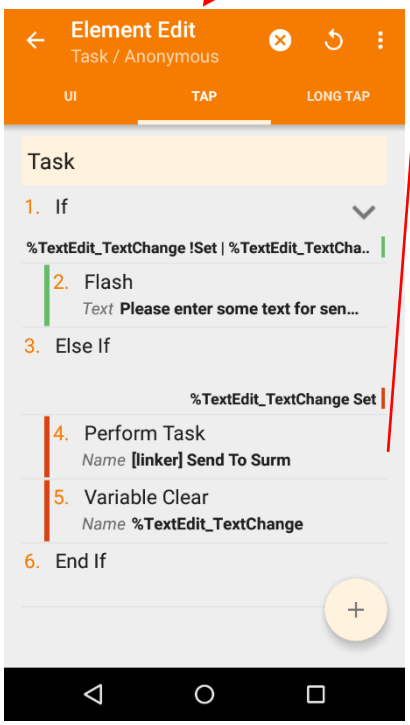
Sample Structural Phase_action_keywords Module

Sample Structural Wi-Fi function module

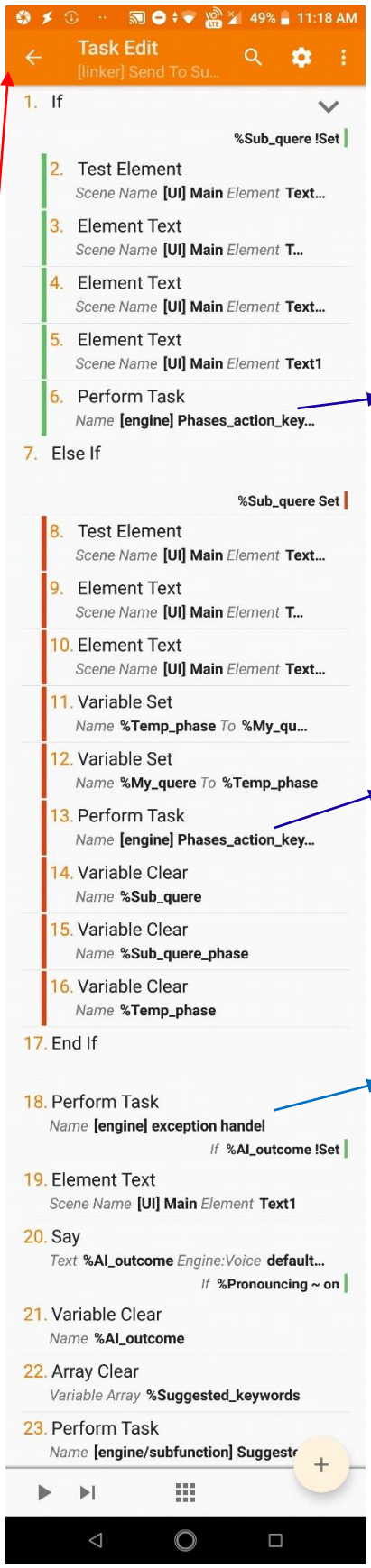
6. Actual Structure

6.1 [linker] modules (Send to SURM & Voice Only):





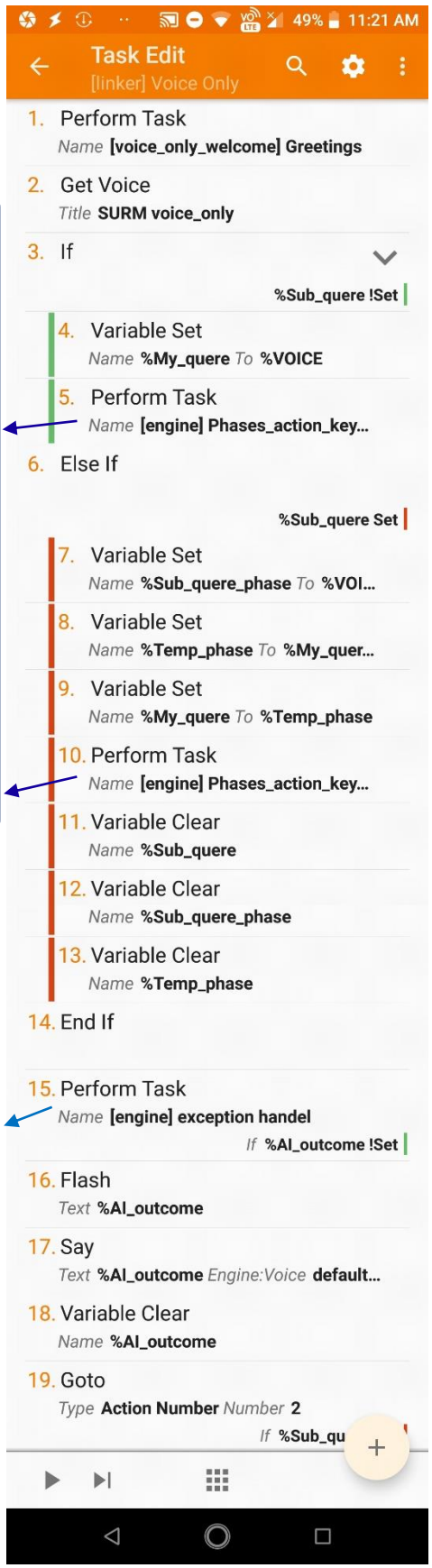
Actual Structural of [linker] Send To SURM module (C-UI mode)



[engine] Phase_action_keywords Module

[engine] exception handel Module

Actual Structural of [linker] Voice Only module (Voice only mode)



The '[linker] Send To SURM' module is called when the Command Execution button is clicked (in C-UI mode)

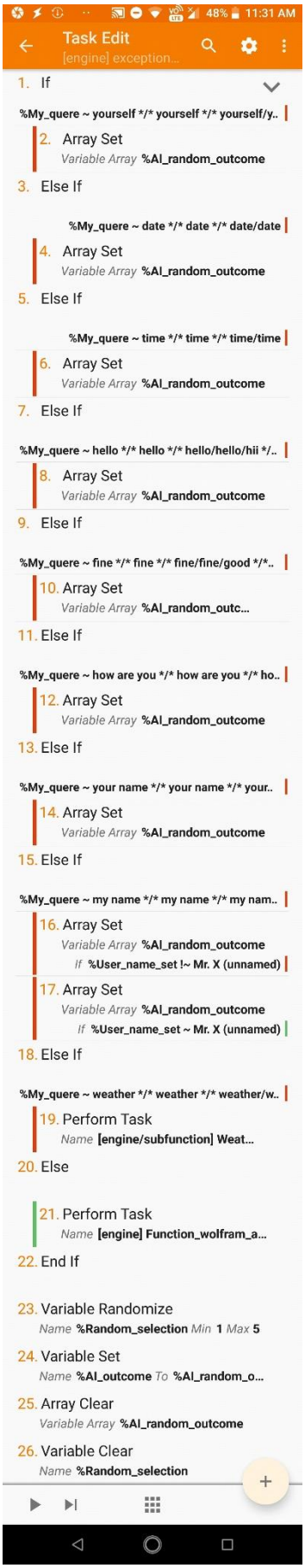
The '[linker] Voice Only' module is called when the the user clicked voice only mode button (voice only mode)

6. Actual Structure

6.2 [engine] Phase_action_keyword & [engine] exception handel Module

Actual structure of *[engine] Phase_action_keywords* Module

***Actual structure [engine] exception handel* Module**



Note: To view them properly, you are recommended to use zoom in feature of your pdf viewer.

7. Supported Keywords

7.1 For normal case:

Here is a list of all available built-in valid keyword (with main keywords and their sub-keywords and some beta keywords) to call a specific function module. To view the list properly, you are recommended to use zoom in feature of your pdf viewer.

Commands(Keywords)		Sub Keywords	Call Function Module ( )	Example Commands	Note
Must Have	Must Not Have				
wifi , wi-fi	wikipedia , information , search , find , look for , research , pronounce	off , shut down , on , open up , toggle	***Function Wi-Fi***	please turn on wi-fi , can you shut down the wifi , what is the current status of wifi?	
bluetooth , blue-tooth	wikipedia , information , search , find , look for , research , pronounce	off , shut down , on , open up , toggle	***Function Bluetooth***	on bluetooth please , bluetooth , toggle the blue-tooth	
calculate , calculation , sum of , total of , solve , math , calculator	wikipedia , information , search , find , look for , research , pronounce		***Function Calculate***	do a sum of 28 + 95 , what is the calculation of sin(30)+tan(45) , please do a math problem (4800/56) +((86-2)/2)	'cancel mathmode' is special keyword for canceling math mode
dial	wikipedia , information , search , find , look for , research , pronounce		***Function Dial***	copy to my dial screen 121 , dial 14425 number	
call	wikipedia , information , search , find , look for , research , pronounce		***Function Call***	call this 7895 phone number , call *121#	
take , capture && photo , picture , selfie	wikipedia , information , search , find , look for , research , pronounce	rear , main , back , front , selfie , sub	***Function Take Photo***	Please take a picture ,please take a picture from rear camera , capture a selfie	
open , launch , go to , browse , run	wikipedia , information , search , find , look for , research , pronounce		***Function Open Apps***	open facebook lite , launch youtube , go to www.google.com , run wikipedia.com	
remember , set , save , take , note	default , wikipedia , information , search , find , look for , research , pronounce	user name , my name , I am , (music && dir , folder , path , directory) , note	***Function Set***	SURM, remember my name as nirmal paul , save the user name to your database as n paul , take a note 20 boys are in my class , take music directory as sdcard0/music/new/ ,	'as', 'to' are 2 special keyword for both user name and music directory set. 'I am' is a special keyword only for user name set. '[dev.it]', '[dev.this], '[dev.all] are some dev/beta keyword for note set and used as reference keyword for current/all output by SURM.
play , sing , listen && music , audio , mp3 , song	wikipedia , information , search , find , look for , research , pronounce		***Function Play Music***	play a random song , can you sing any song ,	
stop , silent , cancel , pause && music , audio , mp3 , song	wikipedia , information , search , find , look for , research , pronounce		***Function Stop Music***	Please stop music playback , stop this music	
count , or anything that have 'count' keyword	wikipedia , information , search , find , look for , research , pronounce	','(coma is a important sub keyword of count function)	***Function Count***	Count from 10,20,2 , count 20,10,1 , can you start counting from -5,5,0.5	'cancel countmode' is special keyword for canceling count mode
wikipedia , information	search , find , look for , research , pronounce , current information	about	***Function Info from Wikipedia***	Give me some information about google , Wikipedia and the topic is about bee	'cancel wikimode' is special keyword for canceling wiki mode
pick , select , choose , decide , decision	wikipedia , information , search , find , look for , research , pronounce	number , from , to, option , selection , choice	***Function Decision Maker***	Can you pick any number from -752 to +956 , please pick anything from 1000 to	

				10 , select any one of them option 'head','tail' , decide what is best for me selection JavaScript,Java,Ruby,C# ,	
pronounce	wikipedia , information , search , find , look for , research		***Function Pronounce***	Please pronounce [dev.it] , pronounce I am N Paul and it's my hobby to learn about programming , surm can you pronounce Detroit ,	'[dev.it]', '[dev.this], '[dev.all] are some dev/beta keyword and used as reference keyword for current/all output by SURM.
notes && show , display , reveal , list , number of , no. of	wikipedia , information , search , find , look for , research , pronounce	(Particular number of notes e.g. 1 or 2 or 5), any , all	***Function Show Notes***	what is the total number of saved notes in your DB , Show me all previously saved notes , reveal notes number 1 , display any notes and surprise me.	
reset , forget , delete , set to default , remove , undo set , unset , clear	wikipedia , information , search , find , look for , research , pronounce	user name , my name , i am , (music && dir , folder , path , directory) , (note && Any number of notes e.g. 1 , all) , screen , display , log , logs , text	***Function Forget***	Please forget my name , please remove notes number 1 , set to default your notes database by deleting all available ,	
share , copy , send , capture		all , full , [dev.all] , current , last , this , it , that , [dev.it]	***Function Share Screen***	Copy the current information , copy it to clipboard , send all information to my clipboard ,	2 dev/beta keyword ([dev.all],[dev.it]) works here as like the sub keyword.
search , find , look for , research		(for , about) , (online , web , google , net , internet , website) ,	***Function Search***	Search in google about Wikipedia , look for my home , search for youtube	
joke , jokes , fact , facts , bored			***Function Fun and Interactive***	Tell me a joke , give me a random fact , Bored	

7.2 *For exceptional case(**exception handel** Module):*

Commands(Keywords)		Sub Keywords	For task	Example Commands	Notes
Must Have	**Must Not Have**				
yourself , about , you do , help , surm ,			For the basic introduction and help	Tell me about yourself , what can you do for me , please help me	
date ,			For showing/telling the date	What is today's date , date	
time ,			For showing/telling the time	What is current time , time	
hello , hii , hi , hey ,			For Hi-Hello	Hello , Hey you ,	
fine , good ,			For responding "I am fine"	I am fine , You are so good ,	
how are you ,			For asking "Who are you"	How are you	
your name , you ,			For asking "What is your name"	What is your name , who are you	
my name , i ,			For asking "What is my name"	Who am I , What is my name	
weather , ~~forecast~~ ,		of , at , in && current location , any_location_name user provide	for weather-related queries	what is the weather condition outside ,	OpenWeather API key is required for getting weather updates & information. Get API key (there is also a free plan available). Note: Forecast feature is not yet perfectly ready.
other than those keywords everything works as 'Wolfram|Alpha' module's keyword(s)			for else & everything	What is the chemical symbol of water , Who is the prime minister of India , father's brother's son , what is 35% of 857	Wolfram|Alpha API key is required for solving query. Get API key (there is also a free plan available).

8. Linking a new function module

Making/Linking a new function module to SURM: There are basically three things to keep in mind to create and link new function modules. 1. The structure of the function module (based on which the function module will perform a specific task), 2. Commands and Keywords (which will be used to call function module), 3. Various Global Variables (which will help the function module to interact with other modules or users).

(1) The structure of the function module: How and what a function module does depends on its structure. The structure of the function module also determines how efficiently or precisely that module will work. Differences in structure are one of the reasons why there are functional differences between different modules. For example: The structure of the calculator module is set up only for calculations so it can only complete calculations and on the other hand the Wikipedia module is designed to download data from Wikipedia only so it can only do that.

(2) Commands and Keywords: Function modules become completely useless unless they are called. To call a specific function module with its specific keyword(s) you just need to use else-if condition in the "Phase action keywords" module. As mentioned above, we can use the keywords to call a specific function module for a specific user command. The specific word or phrase pattern of the user's question or command can be used as the keyword(s). However, keep in mind that you may encounter two problems here (i) 'performance difference between module executing at the higher portion v/s module executing at a lower portion' and (ii) 'keyword(s) overlapping issue'. As mentioned in detail here.

(3) Various Global Variables: *Some of the important global variables that are used to attach new function modules:*

%My_quere	It is used to store all user commands or inputs. So that later SURM can use them.
%Sub_quere	It is set by the function module. When a function module needs more specification on a particular task, then that function module set this variable to counter-question
%Sub_quere_phase	If the %Sub_quere is set, this variable is used to temporarily store the user's commands and subsequently, this variable is used to set my-query (old+new command)
%AI_outcome	After processing, the function module or exception handler module generates answers or responses to user queries. Then the output is stored in this variable and released to the user. :. This variable is used to carry the output(answer) from SURM

Note: Although multiple global variables have been used in SURM, only four of these variables are used to link a new module. Among other variables, some variables are used to properly complete the theming features or C-UI mode while on the other hand some variables are used to store data temporarily.

9. Thanks!

Many thanks for trying out this project. If this project helps you (even if a little bit), I will be very happy and I will consider my efforts worthwhile. If you have any questions, please feel free to ask me, I will try to answer it as soon as possible.

I know SURM maximum features aren't as productive in our day-to-day's task. This is my little effort. Here I have tried to explain how the classic 'AI' Virtual Assistance program works.
And I've tried to show that even with mobile platforms (especially using Tasker), complex applications can be programmed.
For more future updates you can follow our Facebook page, YouTube channel and my GitHub profile.

Our YouTube Page: https://www.youtube.com/channel/UCY6JY8bTlR7hZEvhy6Pldxg/
Our FaceBook Page: https://www.facebook.com/a.New.Way.Technical/
My GitHub Page: https://github.com/nirmalpaul383

Thank you very much…