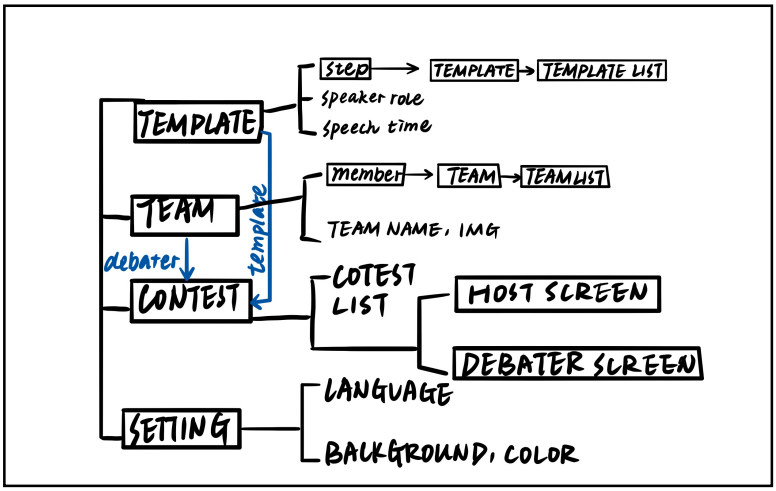
Criterion B-Design

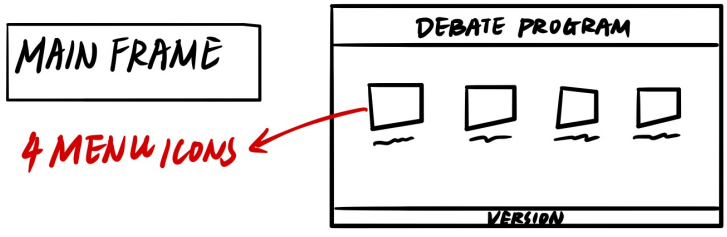
Overview

Based on the first interview with my advisor, I broke down my program into 4 main sections. Decomposition helps me to simplify and clarify the problem. I started with building a structure chart since I already had the abstract interface design and function design in my mind. The diagram below shows the initial structure design of my debate program.

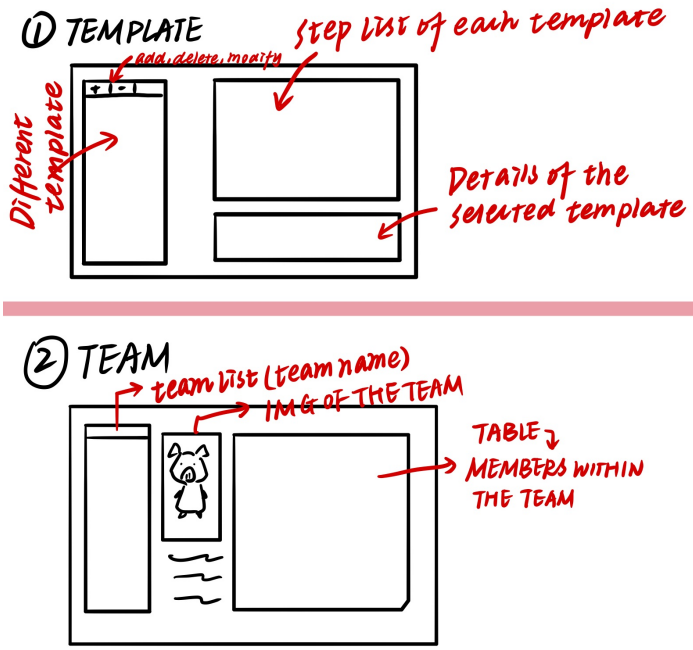


Overview_ structure chart

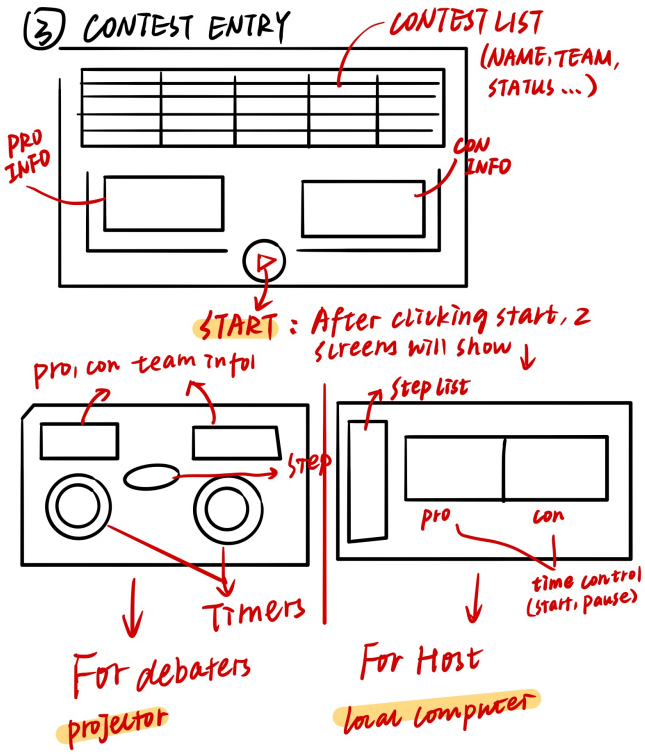
The image above shows a rough design of the program's structure. After that, each part was designed separately. I sketched the prototype of each section (template, team, contest, setting). Prototype designing helps to give clients and my advisor a general concept of what does the program look like and to gain feedbacks from the client.



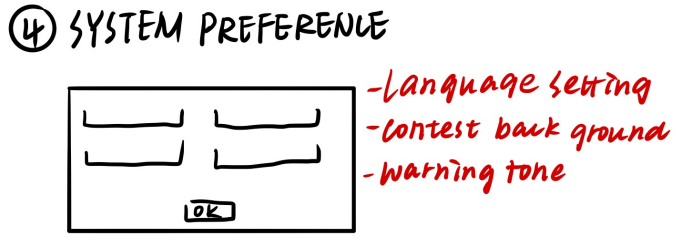
1. The main frame design contains 4 menu icons. The four sections are designed based on the structure chart above.



2. The design of template frame and team frame is based on the data structure and the relationship between the objects within the program, which will be discussed further in later part of this documentation.

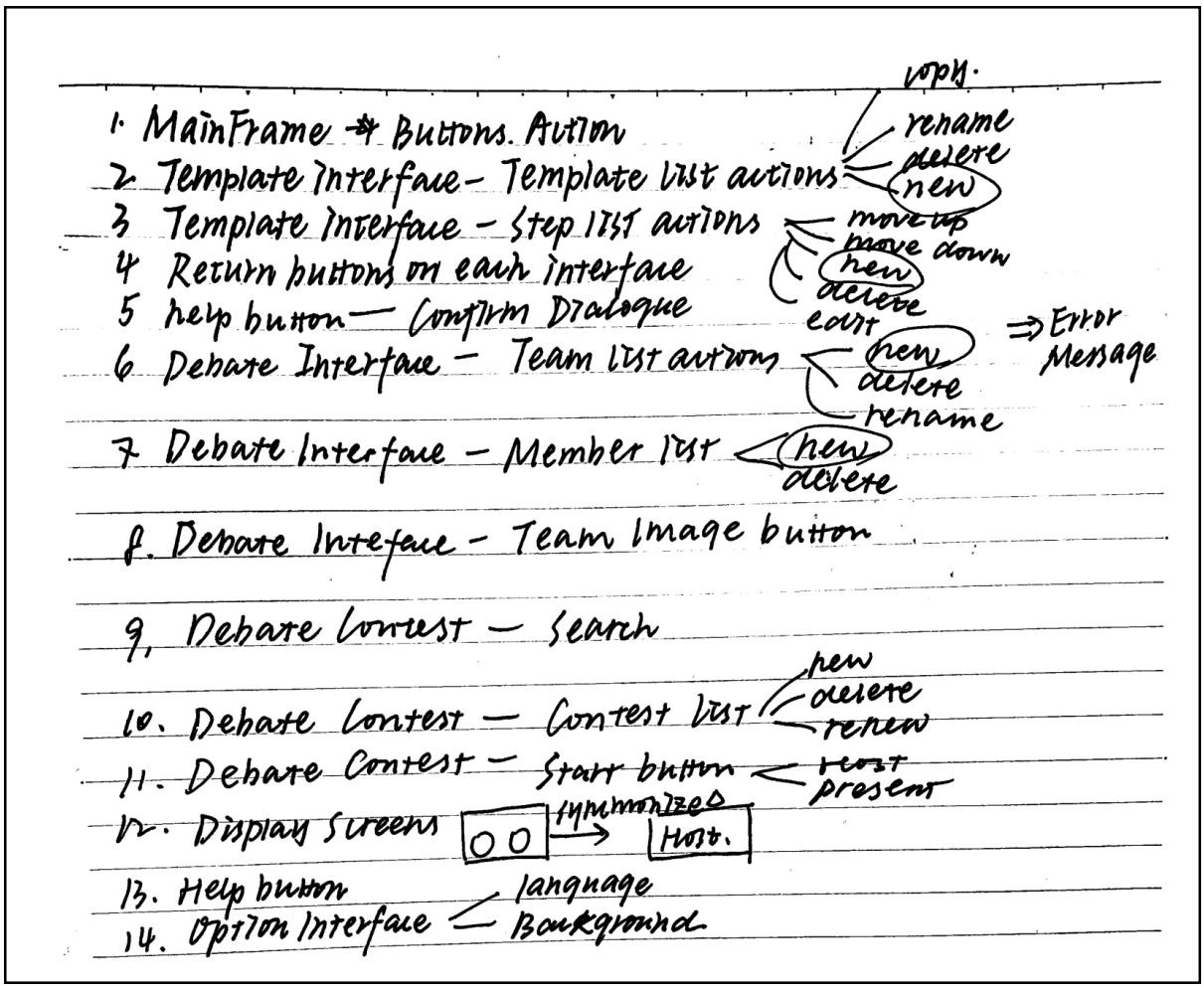


3. GUI design 3 shows the interface for contest selection and creation; the 2 screens at the bottom shows the present interface and the host interface during the debate. The design of the double screen satisfies **success criteria 3.**



4. System preference design meet **success criteria 7** which allows customization of the program's interface and makes the program more **user friendly**.

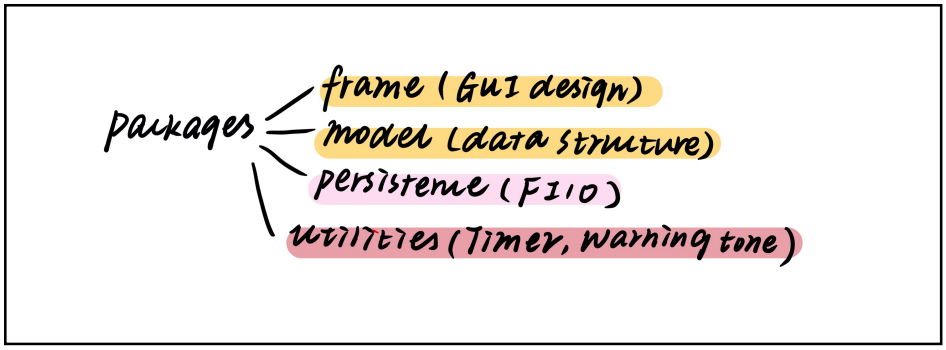
I communicated the prototype design with the debate club members who can be regarded as the end user of my program. (Since the client is myself, I needed to find someone else to discuss my program with.) The picture below is a rough draft of the function the program should include after getting the advises from my club mates. It helped me to construct the UML diagram and the test plan.



Overview_ rough draft_ functionality

UML diagram

According to the structure design above, the program was divided into 3 packages: Frame, Model, Persistence. **Frame** is the package for interface and presentation designing; **model** is for managing the data within the program; **persistence**, just as its name implies, is for locally data storage and necessary file input and output. Later, one more package called “utilities” is added. Since my program need timer and audio plugins, the “Utilities” package is for collecting those functions. It fulfills **success criteria 2 and 3**, which is for adding warning tones and the timer in the program.



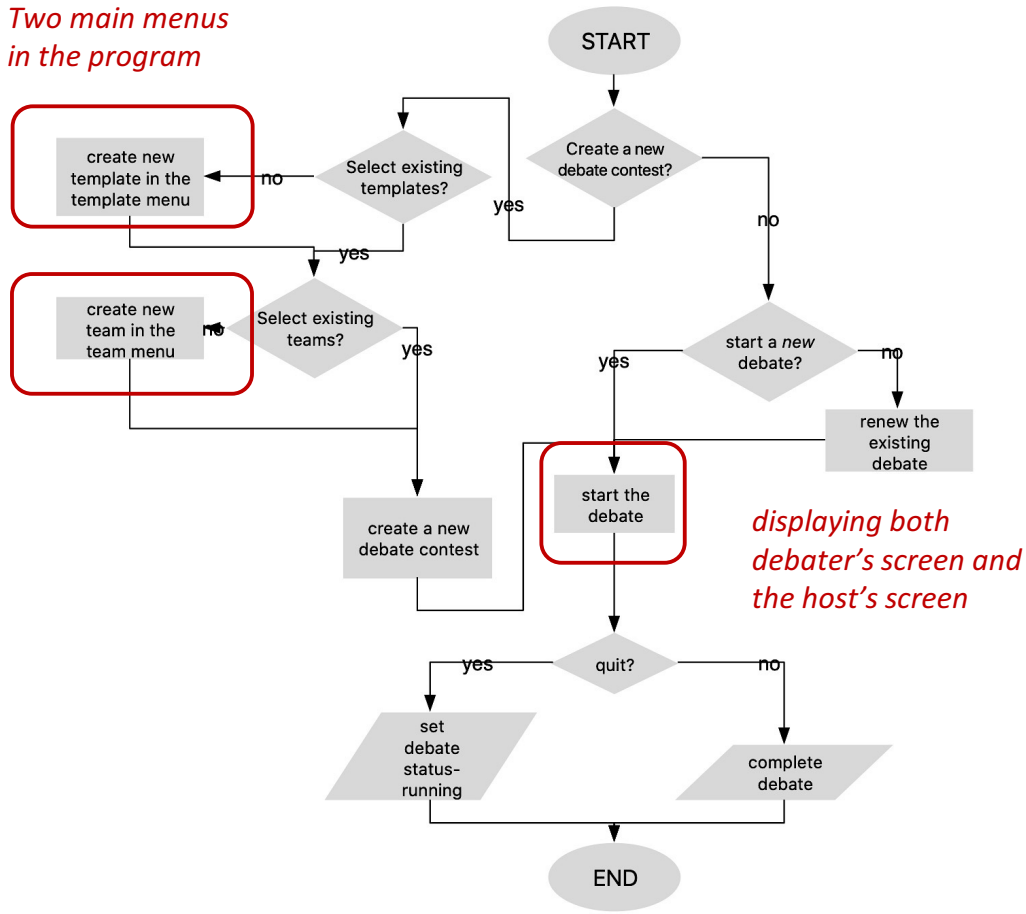
UML_packages

The general UML diagram was not finished during the design stage. The basic connection between these four packages are clear, though. Classes within the **Model** package will

Flowchart & Pseudocode

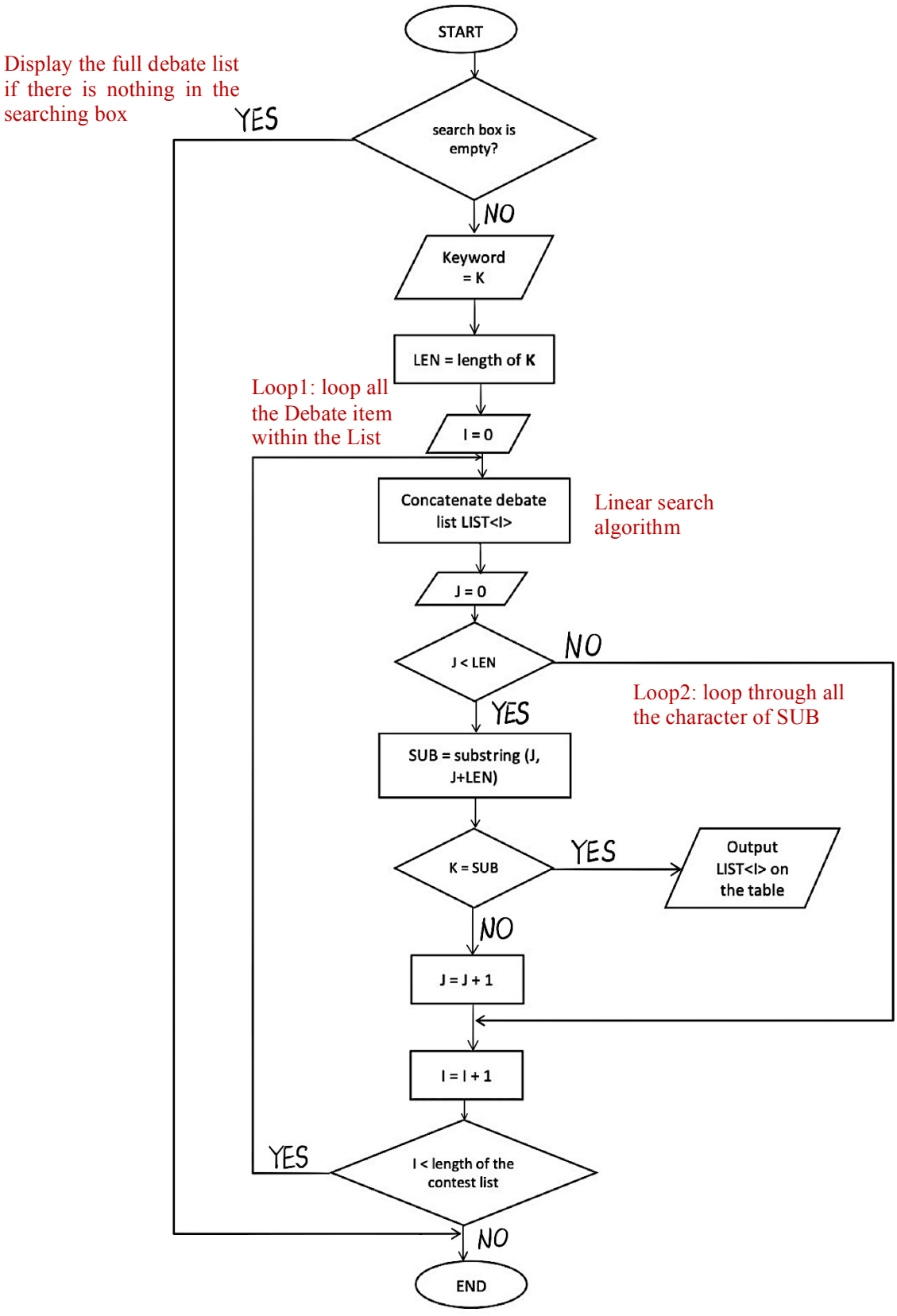
1. Over all program

Since the four main menus within the program have parallel relationships, the flowchart shows the main process of creating and starting a debate contest. The blue rectangles below show the process that involve class outside of debate contest.



FlowChart_ Overall Program

2. Searching algorithm in the Contest Frame



FlowChart_Searching Algorithm

Here is the detailed pseudocode of the searching function of the Debate List table:

```
if (NOT debateList= null) {
   loop (Entry <String, Debate> entry :
   debateList.getDebateMap().entrySet())
        key = entry.getKey()
        debate = entry.getValue()
        if (NOT txtSearch.getText()= null)
            search = txtSearch.getText().toLowerCase();
            length = search.length();
            concatenation = debate.getDebateName() + "#" +
            debate.getTemplateName() + "#" +
            debate.getProsTeamName()+ "#" +
            debate.getConsTeamName() + "#" +
            debate.getStatusStr()
            concatenation = concatenation. toLowerCase();
            count = 0
            size = concatenation. length() - length - 1
            loop while (count < size)
                count = count +1
                j = i + length
                fraction = concatenation.substring(i, j)
                if (fraction. equals(search))
                    Vector<String> row = new Vector<String>()
                        row.add(debate.getDebateName())
                        row.add(debate.getTemplateName()
                        row.add(debate.getProsTeamName()
                        row.add(debate.getConsTeamName()
                        row.add(debate.getStatusStr()
                        rowData.add(row)
                        break
                end if
            end loop
        end if
   end loop
```

Access to all the items within the debate collection

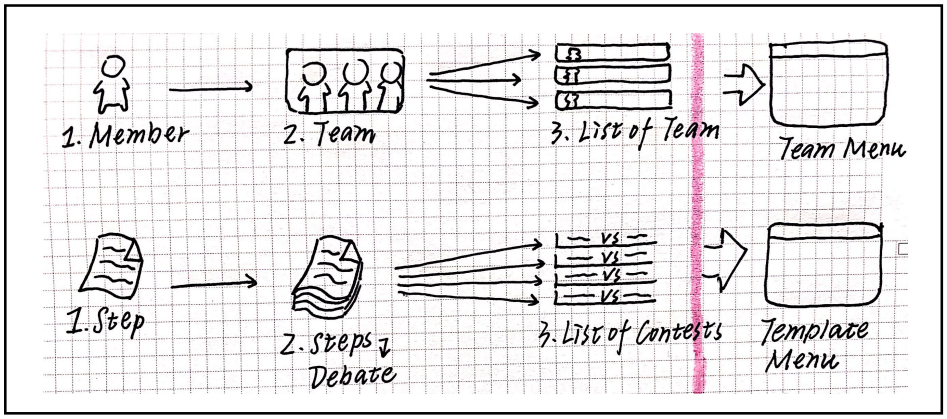
Judging whether the user enter the keyword in the search box

Convert all the text to lower case

Concatenate the information in a debate item

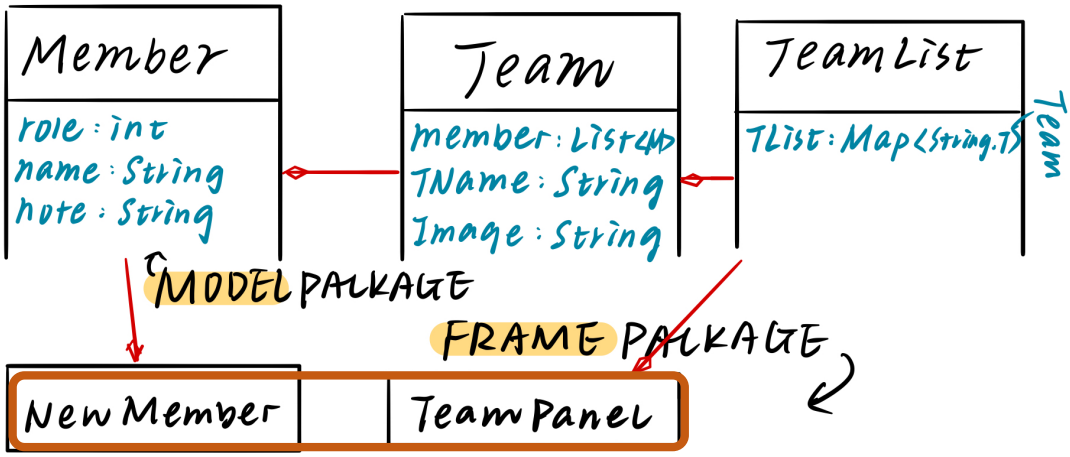
Add the matched debate item to the table

Data Structure

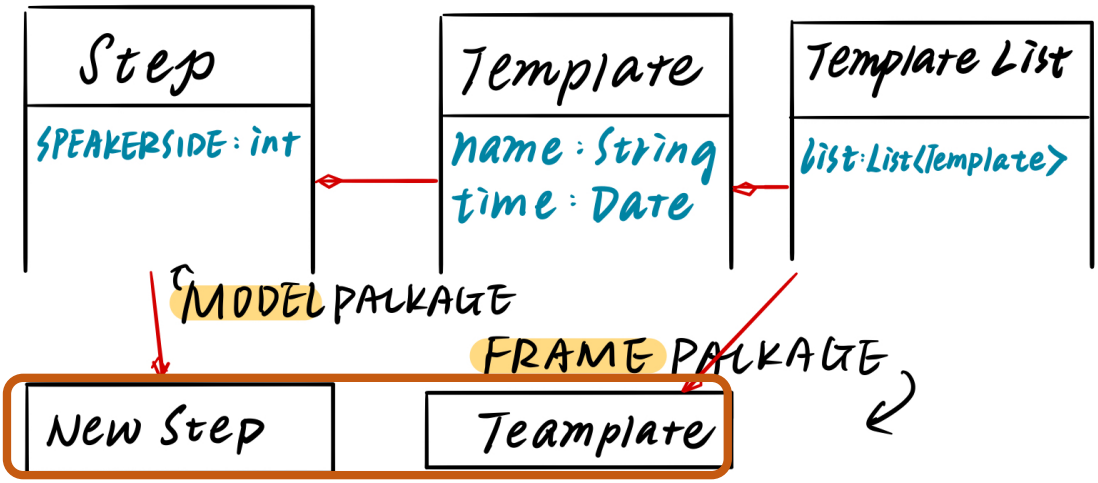


Data Structure_1

The graph above shows initial idea of the data hierarchy within the program. Generally, Member and Step is the basic unit (object) to compose Team List and Contest List respectively. Thus, the first step is to create Member object and Step object, and give them proper attribute. For example, for a Member object, it needs: 1. The role of the object 2. Name. For a Team object, it may need: 1. Team photo 2. Team name 3. Members objects. The logic of creating a Debate object is similar to that of creating a Team object.

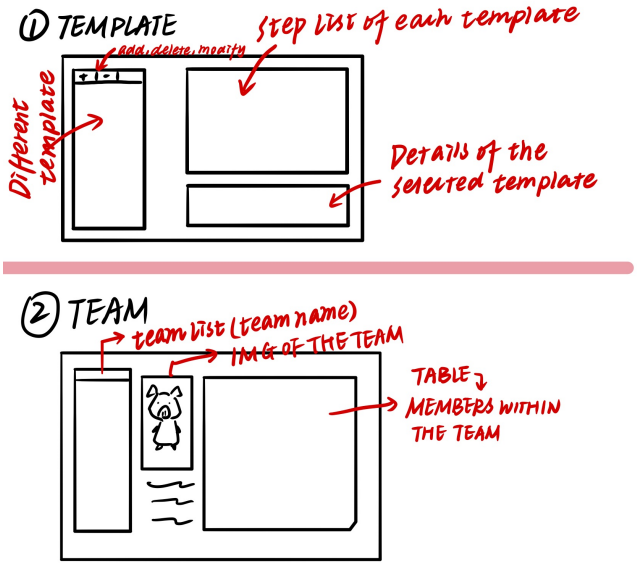


Data Structure_ Member-Team-Team List



Data Structure_ Step-Template-Template List

The UML diagrams above only show public attributes and public methods. The classes shown in the orange rectangles are from the frame package, which show the corresponding interface that utilize the objects mentioned previously. More detailed information will be discussed in Criterion C.



Test Plan

Task No.	Content	Success criteria	Description	Success criteria
1	*Main Frame* Four menus' buttons	Clicking the menu button, the frame should be replaced by the menu page.	NA	4
2	*Template Frame* Template List Actions	Four Responsive functions for the template list table (copy, rename, delete, new)	**copy button**: copy the selected template; add the exactly same template to the template list **rename button**: show Dialogue, Users could type in the new template name to rename the selected template. The type in data could be any string. Click "OK" to save, "Cancel" to withdraw the action. **Delete button**: Delete the template from the list. **New button**: Show error message if the text areas are not filled or filled with wrong data type.	4a
3	*Template Frame* Step List Actions	Five Responsive functions for the step list table. (move up, move down, delete, new, edit)	Users can do four things with the template list table: **move up/down**: Adjust the arrangement of the step. **New button:** Create a new template in a show-up	1, 4a

			panel. Error message appears if the data is incomplete or in a wrong form. **Delete button**: Delete the step from the list. **Edit button**: similar as the New button. Users could edit the existing information directly by clicking it.	
4	Return button on each interface	Return button will lead the user to the prior layer.	NA	1
5	Help button	Show the help & contact information. A Confirm Dialogue.	NA	
6	*Debate Frame* Team list actions	Three Responsive functions for the team list table. (rename, delete, new)	**New**: creating a debate only requires the name of the debate. The name of the debate can be any string. If the user does not enter the name of the debate, an error message will show up.	4b
7	*Debate Frame* Member list actions	Two Responsive functions for the member list table. (delete, new)	**New**: error handler.	4b
8	*Debate Frame*	Team Image button	Team Image button should lead the user to access the	4b

	Add team Image		local files. Only file in image format can be selected (jpeg, jpg…)	
9	*Contest Frame* Search function	Search the contest(s)	User should be able to search the contest(s) by entering the key word in the text area. Only the contests which contain the key word entered by user will show in the list below. A "return" button can lead the user back to full contest list review.	5
10	*Contest Frame* Contest List actions	Three Responsive functions for the contest list table. (new, delete, renew)	By clicking the renew button, user should be able to set the status of completed/running debate to a new debate.	4c, 6
11	*Contest Frame* Start the contest, double screens	Allows two screen display	Start button should lead the user to the display screens (one screen for the debaters and the audiences, the other screen for the host) different screens on the projector and the personal computer respectively.	3
12	*Present screen* and *Host screen* synchronization timer	The time on both screen should be the same no matter what action is taken	NA	3
13	*Host Screen* Timer control	Two Responsive functions for the	NA	3b

		timer (start/resume, pause)		
14	Warning tone function	tone at 30s, 5s count down, the end	If a step is less than 30 seconds, there will not be 30s warning tone. The warning tone is dependent on the timer on both screens.	2
15	*Option Frame* Language option	Should be able to choose Chinese/English	The in-program labels, buttons should be able to switch between Chinese and English.	4d
16	*Option Frame* Debate background option	User should be able to choose the background of the *Present Screen* from the local files.	NA	4d, 7
17	Persistence	Restart the program, the information from last time will not lose.	The data in the program should be stored locally after any changes.	8