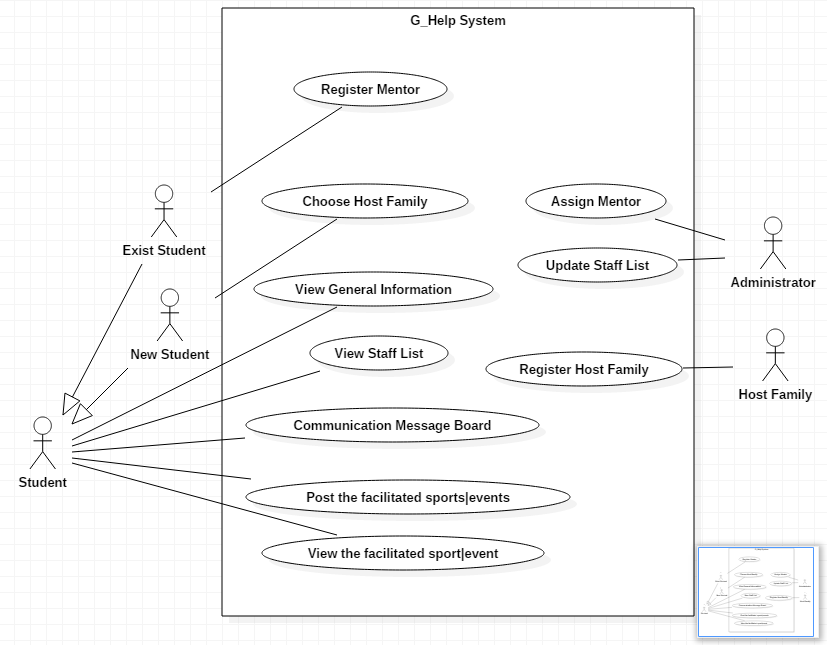
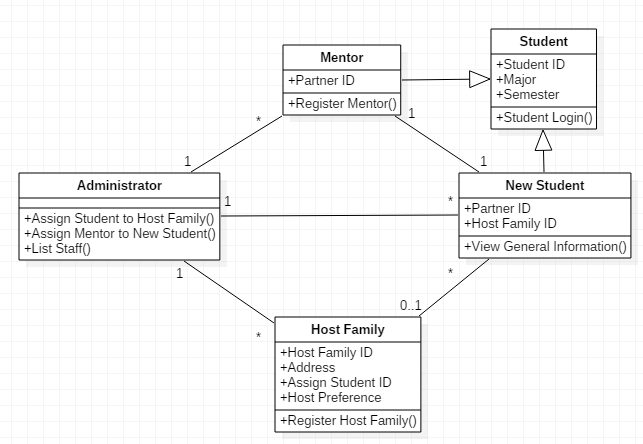
**System Description:**

Through this system, all students will be able to view their information such as personal information, academic grade and the events they like. They can choose their favorite events and put them into calendar; they can browse the information of staff of their departments and other research or any academic information to help them get familiar with school life better. Based on the criteria given by system, students reached the requirements could choose to register as mentors, and the administrator could assign those mentors to students, those mentors will also guide these new students through the whole semester. Also, the interested host family could register in system, and our system will list some host families to students by some rules, then students choose the host family, meanwhile the host family can choose to accept the students or not.

**Use Case diagram**



**Domain Modeling**



**Detail description for Use Cases**

1. Student Login

|  |  |
| --- | --- |
| Name | Student Login |
| Description | A student uses their account and password to login. The account and default password comes from external system |
| Actor | Student |
| Entry Condition | The student is a MUN student |
| Exit Condition | The student are authenticated |
| Flow of the event | 1. The student fill out the ID & Pass and then click the button for Login  2. Verify the information if the student submit the correct ID & Pass  if yes, 3-1  Otherwise, 3-2  3-1. The system move to login page  3-2. Notice to the student that the information is incorrect |
| Exception | An error page in case of the system failure |
| Special Requirement | None |

|  |  |
| --- | --- |
| Entity Object | - Student |
| Boundary Object | - Login Interface (Text boxes and buttons) |
| Control Object | - Student Login / Student Authentication |

2. View General Information

|  |  |
| --- | --- |
| Name | View General Information |
| Description | After new students login, the students can view the general information. The information are housing, classes, instructors and research possibilities |
| Actor | Student |
| Entry Condition | Login, User access this page |
| Exit Condition | Show the general information page |
| Flow of the event |  |
| Exception | The student is notified in case of the system failure or permission denied |
| Special Requirement | None |

|  |  |
| --- | --- |
| Entity Object | - Information of housing, classes, instructors and  research possibilities |
| Boundary Object | - The general information interface page for new students |
| Control Object | - Display the general information |

3. Register Mentor **(Essential Use Case)**

|  |  |
| --- | --- |
| Name | Register Mentor |
| Description | A student apply to mentor so the student can be a mentor |
| Actor | Existing Student |
| Entry Condition | Login, User access this page |
| Exit Condition | A student is registered in the list of mentor |
| Flow of the event | 1. The student click the button for register  2. Moves to the page that display guideline for mentor and \*criteria  to be a mentor  3. Click the register button  4. Verify the information if the student meet the criteria  if yes, 6-1  Otherwise, 6-2  5-1. The system add the student to the list of mentor and notice to the student that the student is successfully added to the list  5-2. Notice to the student that the student doesn’t meet the criteria |
| Exception | None |
| Special Requirement | None |

|  |  |
| --- | --- |
| Entity Object | - Student |
| Boundary Object | - View page of guide line for mentor  - Register button |
| Control Object | - Register Mentor |

\* Criteria: Students must have finished at least one year of their program

4. Assign Mentor **(Essential Use Case)**

|  |  |
| --- | --- |
| Name | Assign Mentor |
| Description | Administrator assign mentor to new student. Mentors must be registered in advance |
| Actor | Administrator |
| Entry Condition |  |
| Exit Condition | Mentors are assigned to New Students |
| Flow of the event | 1. In certain period of time (early of each semester), the system  starts to match mentor and new student  2. Order the mentors by criteria, for example GPA, and select  mentors with only the number of new students.  3. The system match one student to one mentor for every new  students  4. The system notify them they are matched through e-Mail or can be seen from their login page |
| Exception | New students would be notified who are not assigned to a mentor. New students would not able to be assigned if the number of mentors are not enough than new students |
| Special Requirement | None |

|  |  |
| --- | --- |
| Entity Object | - Mentor, New Student |
| Boundary Object | - The general information interface page for students |
| Control Object | - Select mentors by criteria (Mentor filter)  - Mentor and Student Match |

5. Register Host Family **(Essential Use Case)**

|  |  |
| --- | --- |
| Name | Register Host Family |
| Description | A family apply for host family so the family can host students |
| Actor | Host Family |
| Entry Condition | Host Family application form submitted |
| Exit Condition | A host family is registered in the list of host family |
| Flow of the event | 1. The family user click the button for application  2. The page moves to registration form page  3. The user fills out the form with \*the user information  4. Click the submit button  5. Verify the information,  if it is correctly filled, 6-1  Otherwise, 6-2  6-1. The system add the user to the list of host families and notice to the user that they are successfully added to the list  6-2. Notice to the user that their information is not proper |
| Exception | The user is alarmed if the form is not completely filled out |
| Special Requirement | None |

|  |  |
| --- | --- |
| Entity Object | - Host Family |
| Boundary Object | - Registration Form  - Submit Button |
| Control Object | - Host Family Register |

\* User provide information: 'Date', 'Applying Semester', 'Name', 'Address', 'e-Mail', 'Phone #',

'Occupation', '\*Preference', 'Message'

\* Preference: Type of student host family is looking for. Host family list the 3 most important

Characteristics such as country, language, sport/hobbies, religion, gender etc.

6. Choose Host Family **(Essential Use Case)**

|  |  |
| --- | --- |
| Name | Choose Host Family |
| Description | A new student choose a host family from the list of host families |
| Actor | New Student |
| Entry Condition | Login, User access choose host family page |
| Exit Condition | A new student is assigned to the selected host family |
| Flow of the event | 1. The student click the button for dialog of the list of host families  2. The system find the families from the list based on predefined criteria so student only can see the satisfying families  3. The student click the select button for one family  4. The system assigned the student to the selected family as a host  5. The system send an e-Mail to the family to notify the family is  select for host family |
| Exception | None |
| Special Requirement | None |

|  |  |
| --- | --- |
| Entity Object | - Host Family, Student |
| Boundary Object | - Host Family list page  - Select Button |
| Control Object | - Host Family Assign |

\* Student information: ‘Name’, 'Major', 'Nationality', 'Language', 'Address', 'Phone #', ‘e-Mail’

7. Post the facilitated Sport/Event

|  |  |
| --- | --- |
| Name | Post the facilitated Sport/Event |
| Description | Students can post an event or view events |
| Actor | Student |
| Entry Condition | Login, Student access to the facilitated Sport/Event page |
| Exit Condition | Post an event, View the list of events |
| Flow of the event | 1. Students can click the “event ” button on the homepage after  they login to open the event page   2. They can click the “post event” button to create an event   3. They will need to fill in a form for this event about the  location ,time, content ,aiming group, and so on.   4. At the bottom part of this page, student can choose to add this  event into a public calendar for future reminding |
| Exception | None |
| Special Requirement | None |

|  |  |
| --- | --- |
| Entity Object | - Event Information |
| Boundary Object | - View Button for the list of events  - Events List View page  - Post button to post an event |
| Control Object | - Post an event  - Show the list of events |

8. View the facilitated Sport/Event

|  |  |
| --- | --- |
| Name | View the facilitated Sport/Event |
| Description | Students can see an event or view events |
| Actor | Student |
| Entry Condition | Login, Student access to the facilitated Sport/Event page |
| Exit Condition | Show the list of events |
| Flow of the event |  |
| Exception | None |
| Special Requirement | None |

|  |  |
| --- | --- |
| Entity Object | - Event Information |
| Boundary Object | - Events List View page |
| Control Object | - Show the list of events |

9. View Staff List

|  |  |
| --- | --- |
| Name | View Staff List |
| Description | List \*staff information so students can see from their login page |
| Actor | Student |
| Entry Condition | Login, Student access Staff List page |
| Exit Condition | Show the information of staff |
| Flow of the event |  |
| Exception | None |
| Special Requirement | None |

|  |  |
| --- | --- |
| Entity Object | - \*Staff |
| Boundary Object | - Staff List page |
| Control Object | - Staff List |

\* Staff information: name, department, position, e-Mail, room and telephone

11. Update Staff List

|  |  |
| --- | --- |
| Name | Update Staff List |
| Description | Staff information is updated |
| Actor | Administrator |
| Entry Condition |  |
| Exit Condition | Update the information of staff |
| Flow of the event |  |
| Exception | None |
| Special Requirement | None |

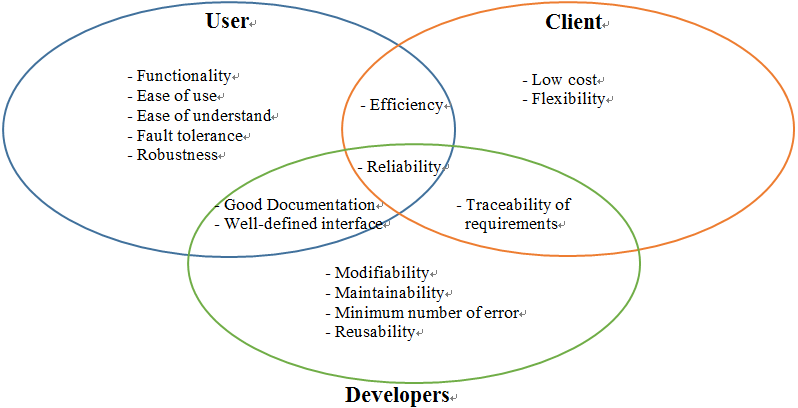
|  |  |
| --- | --- |
| Entity Object | - \*Staff |
| Boundary Object | - Staff List page |
| Control Object | - Staff List Update |

12. Communication Message Board

|  |  |
| --- | --- |
| Name | Communication Messages Board |
| Description | Students can communicate through leaving messages on Board |
| Actor | Student |
| Entry Condition | Login, User access message board page |
| Exit Condition | Message Board page |
| Flow of the event | 1. Students click “message board” button on the homepage after  they login to open the message board page.   2. The messages left on this page will be organized by the order of time and the titles.   3. Students can click the titles of message to review the detail. If they are interested in the message they can directly reply.   4. Students can also choose to leave a message by click the button “leave a message” on the top of the page .   5. They will go in to the page for creating a message, here body part for the message and provide their information on information part(optional) |
| Exception | None |
| Special Requirement | None |

|  |  |
| --- | --- |
| Entity Object | - Message Board |
| Boundary Object | - Message Board page |
| Control Object | - Message Board view |

**Design goals**



Addressing the goals

**- Functionality, Modifiability, Maintainability, Flexibility:**

We can achieve them through a good design of the system structure including

the layers and the classes, and the design patterns.

**- User-Friendliness, Ease of Use, Ease of Understand:**

We can achieve them through a good design on the webpage or give some hint onsome functions on the webpage (optional)

**- Robustness, Reliability, Efficiency, Low-cost:**

We can achieve them through optimizing on code and multi-test on the runtime on the code, we also need to set up criteria on the response time to avoid interrupting the user’s flow of thought

**- Fault Tolerance, Minimum Number of Error:**

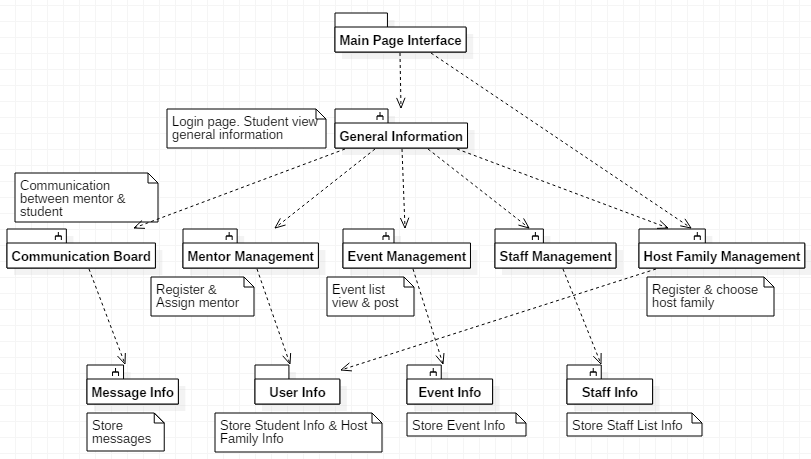
We can optimize the phase of system testing.

**- Good Documentation, Traceability of Requirements:**

We can achieve them through the frequent update and backup on the rationale

decisions and documenting all the original requirements.

**Decompose the system**



**Logical Architecture**

MVC Architectural Style

