

**The University Team Project Platform**

**Software Requirement Specification**

2021.04.25.

**Introduction to Software Engineering 41**

**TEAM 14 (Hello Team Project)**

Team Leader Jaehyun Ju

Team Member Min Jang

Team Member Heesoo Jung

Team Member Byeongsu Woo

Team Member Seongwook Lim

**CONTENTS**

[**1.**](#_heading=h.gjdgxs) **Introduction** 7

[1.1.](#_heading=h.30j0zll) Purpose 7

[1.2.](#_heading=h.1fob9te) Scope 7

[1.3.](#_heading=h.3znysh7) Definitions, Acronyms, and Abbreviation 8

[1.4.](#_heading=h.1t3h5sf) References 10

[1.5.](#_heading=h.4d34og8) Overview 10

[**2.**](#_heading=h.17dp8vu) **Overall Description** 11

[2.1.](#_heading=h.3rdcrjn) Product Perspective 11

[2.1.1.](#_heading=h.26in1rg) System Interfaces 11

[2.1.2.](#_heading=h.lnxbz9) User Interfaces 12

[2.1.3.](#_heading=h.35nkun2) AWS Instance Interfaces 12

[2.1.4.](#_heading=h.1ksv4uv) Web Browser Interfaces 12

[2.1.5.](#_heading=h.44sinio) Communications Interfaces 12

[2.1.6.](#_heading=h.z337ya) Operations 12

[2.1.6.1.](#_heading=h.3j2qqm3) System administrator 12

[2.1.6.2.](#_heading=h.1y810tw) Students 13

[2.1.6.3.](#_heading=h.m6xlv3sut03c) Professors & TAs 13

[2.2.](#_heading=h.4i7ojhp) Product Functions 14

[2.2.1.](#_heading=h.2xcytpi) Log-in 14

[2.2.2.](#_heading=h.1l9de7toruk)Manage My Information 14

[2.2.3.](#_heading=h.vfbb0n11o5ng)Choose a course 14

[2.2.4.](#_heading=h.1ci93xb)Get student information list 14

[2.2.5.](#_heading=h.3whwml4)Get student specific information 14

[2.2.6.](#_heading=h.2bn6wsx)Deliver a team build request to professor 14

[2.2.7.](#_heading=h.qsh70q)Team project communication platform 14

[2.2.8.](#_heading=h.3as4poj)Rating team member 15

[2.2.9.](#_heading=h.1pxezwc) Managing the teams 15

[2.2.10.](#_heading=h.49x2ik5) Updating user data 15

[2.2.11.](#_heading=h.w0nckcc0hbb2) Construct / Destroy the course 15

[2.2.12.](#_heading=h.akscbu7qs6xo) Report management 15

[2.3.](#_heading=h.2p2csry) User Characteristics 15

[2.3.1.](#_heading=h.147n2zr) System Administrator 15

[2.3.2.](#_heading=h.3o7alnk) Professor & TA 16

[2.3.3.](#_heading=h.23ckvvd) Student 16

[2.4.](#_heading=h.ihv636) Constraints 16

[2.5.](#_heading=h.32hioqz) Assumptions and Dependencies 17

[**3.**](#_heading=h.1hmsyys) **Specific Requirements** 17

[3.1.](#_heading=h.41mghml) External Interface Requirements 17

[3.1.1.](#_heading=h.b0ueuxdbs6j3)User Interfaces 17

[3.1.2.](#_heading=h.37m2jsg)AWS architectures 25

[3.1.3.](#_heading=h.46r0co2) Communication Interfaces 27

[3.2.](#_heading=h.111kx3o) Functional Requirements 28

[3.2.1.](#_heading=h.3l18frh) Use Case 28

[3.2.2.](#_heading=h.4bvk7pj) Use Case Diagram 32

[3.2.3.](#_heading=h.1664s55) Data Dictionary 32

[3.2.4.](#_heading=h.pkwqa1) Data Flow Diagram 36

[3.3.](#_heading=h.1opuj5n) Performance Requirements 36

[3.3.1.](#_heading=h.2nusc19) Static numerical requirement 36

[3.3.2.](#_heading=h.1302m92) Dynamic numerical requirement 37

[3.4.](#_heading=h.3mzq4wv) Logical Database Requirements 37

[3.5.](#_heading=h.2250f4o) Design Constraints 38

[3.6.](#_heading=h.haapch) Standards compliance 38

[3.7.](#_heading=h.319y80a) Software System Characteristics 38

[3.7.1.](#_heading=h.1gf8i83) Product Requirements 38

[3.7.1.1.](#_heading=h.40ew0vw) Interface Requirements 38

[3.7.1.2.](#_heading=h.2fk6b3p) Performance Requirements 39

[3.7.1.3.](#_heading=h.upglbi) Usability Requirements 39

[3.7.1.4.](#_heading=h.3ep43zb) Security Requirements 39

[3.7.2.](#_heading=h.1tuee74) Organizational Requirements 39

[3.7.2.1.](#_heading=h.4du1wux) Environmental Requirements 39

[3.7.2.2.](#_heading=h.2szc72q) Operational Requirement 39

[3.7.3.](#_heading=h.184mhaj) External Requirements 40

[3.7.3.1.](#_heading=h.3s49zyc) Safety / Security Requirement 40

[3.7.3.2.](#_heading=h.279ka65) Regulatory Requirement 40

[3.8.](#_heading=h.meukdy) Organizing the Specific Requirements 40

[3.8.1.](#_heading=h.36ei31r) Context Model 41

[3.8.2.](#_heading=h.45jfvxd) Process Model 42

[3.8.3.](#_heading=h.zu0gcz) Interaction Model 42

[3.8.4.](#_heading=h.3jtnz0s) Behavior Model 42

[3.8.4.1.](#_heading=h.1yyy98l) Data Flow Diagram 42

[3.8.4.2.](#_heading=h.4iylrwe) Sequence Diagram 42

[3.9.](#_heading=h.1d96cc0) System Architecture 44

[3.10.](#_heading=h.2ce457m) System Evolution 44

[3.10.1.](#_heading=h.rjefff) Limitation and Assumption 44

[3.10.2.](#_heading=h.rjefff)Evolutions of team project method 44

[3.10.3.](#_heading=h.3bj1y38) Evolution of team matching 45

[**4.**](#_heading=h.1qoc8b1) **Supporting Information 4**5

[4.1.](#_heading=h.4anzqyu) Software Requirement Specification 45

[4.2.](#_heading=h.2pta16n) Document History 45

**LIST OF FIGURES**

[[Figure 1] Use case diagram 3](#_heading=h.2r0uhxc)2

[[Figure 2] Entity Relationship Diagram 35](#_heading=h.2afmg28)

[[Figure 3] Data flow diagram 36](#_heading=h.39kk8xu)

[[Figure 4] Context model 4](#_heading=h.1ljsd9k)1

[[Figure 5] Overall process model 4](#_heading=h.2koq656)2

[[Figure 6] Sequence diagram 4](#_heading=h.2y3w247)3

[[Figure 7] System architecture of the system 4](#_heading=h.3x8tuzt)3

**LIST OF TABLES**

[Table 1] Table of acronyms and abbreviations 8

[Table 2] Table of terms and definitions 8

[Table 3] Login Page 16

[Table 4] Student Profile Page 17

[Table 5] Course Selection Bar 18

[Table 6] Member Searching Page 19

[Table 7] Team Building Request Page 20

[Table 8] Team Communication Page 22

[Table 9] Team Management Page 23

[Table 10] AWS architecture diagram 24

[Table 11] Communication interface of client and host 26

[Table 12] Use case of account register 27

[Table 13] Use case of log-in/out 27

[Table 14] Use case of profile 28

[Table 15] Use case of making team 29

[Table 16] Use case of team project space 29

[Table 17] Use case of finding team member 30

[Table 18] Use case of team member rating 31

[Table 19] Student-from school 31

[Table 20] Profile 32

[Table 21] Time schedule 32

[Table 22] Professor 32

[Table 23] Course 33

[Table 24] Class 33

[Table 25] Comment & Rating 34

[Table 26] Document History 44

1. **Introduction**
   1. **Purpose**

This document is a Software Requirements Specification (SRS) for providing Hello Team Project (The university team project platform) services. This service is designed and implemented by Team 14 of the Introduction to Software Engineering at Sungkyunkwan University. The requirements for this are summarized, analyzed, and the system is designed and implemented based on the contents described.

In this document, Team 14 is the main reader, and Team 14 designs and implements the functions of the university team project platform according to this specification. Additionally, professors, TAs, and team members in the Introduction to Software Engineering class can be the main readers.

The purpose of this document is to outline and publish the Requirement Specification for a new internet platform for university online team project system. Unlike many others, Hello Team Project gets a personal rating for team members and uses that for making matching the appropriate team algorithm. This rating is also used for team recruiting for someone who is searching for an enthusiastic team member.

* 1. **Scope**

The Hello Team Project system is meant to ease the exhausting hours of searching for the enthusiastic member for the perfect team. And, It is a platform for professors or TAs that makes them manage team projects easily. Also it provides the team project space for convenient communication, by giving a channel for each team. The system itself has a rating system that is accumulated for the next team project, thus it helps students to be passionate when they do a team project. The system is based on a relational database. We will have a database server supporting a lot of ratings, user comments and self resume for evaluating the individual user clearly. Above all, we hope to provide a comfortable user experience along with the best team project to the students, professors and TAs.

* 1. **Definitions, Acronyms, and Abbreviation**

The following table explains the acronyms and abbreviations used in this document.

[Table 1] Table of acronyms and abbreviations

|  |  |
| --- | --- |
| **Acronyms& Abbreviations** | **Explanation** |
| AWS | Amazon Web Service |
| ECR | Elastic Container Registry |
| ECS | Elastic Container Service |
| S3 | Simple Storage Service |
| RDS | Relational Database Service |
| DBMS | Database Management System |
| CPU | Central Processing Unit |
| HTML | Hypertext Markup Language |
| HTTP | Hypertext Transfer Protocol |
| JSON | Javascript Object Notation |
| UI | User Interface |
| TA | Teaching Assistant |
| API | Application Programming Interface |
| CRUD | Create, Read, Update, Delete |

The following table defines certain technical terms used in this document.

[Table 2] Table of terms and definitions

|  |  |
| --- | --- |
| **Terms** | **Definitions** |
| User | Someone who uses a system. Professors, TAs and students. |
| Classmate | Student who attends the same lecture. |
| System administrator | Someone who manages the account information and student comment. |
| Mattermost | An open source collaboration platform |
| Back-End | Application part that is not directly accessed by the user, such as the server and database |
| Front-End | The user interface, also known as the presentation layer of an application |
| Algorithm | A set of rules or procedures followed by a computer in problem-solving operations |
| Client (user device) | A user device/user that connected to server |
| Server | A computer or computer program which manages access to a centralized resource or service in a network |
| Software | The programs and other operating information used by a computer |
| Network | For connect devices together so that they can share information. In this system, it usually means internet |

* 1. **References**
* IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications, In IEEEXplore Digital Library  
  http://ieeexplore.ieee.org/Xplore/guesthome.jsp
* Team 1. “Software Requirement Specification”. SKKU, Last Modified: May. 14, 2020.  
  https://github.com/skkuse/2020spring\_41class\_team1/blob/master/docs/SRS\_TEAM1.pdf
  1. **Overview**

The remainder of this Software Requirements Specifications Document includes three chapters and appendixes. The second chapter provides an overall description of the product perspective, including the several interfaces, the system functionality and system interaction with other systems. This chapter also introduces different types of stakeholders and their interaction with the system. Additionally, the chapter also mentions item details, the system constraints, assumptions, and dependencies of the product. The third chapter provides the requirements specification in detailed terms, including a description of the different system interfaces and the software system characteristics. A range of specification techniques are used to specify the requirements more precisely for a variety of users. It also shows a lot of use cases and a data dictionary. The fourth chapter deals mainly with the prioritization of the requirements. It includes a supporting documentation, a timeline of the documentation of this SRS for the application, Hello Team Project. All members contributed equally to the production of this project. We hope that you, the reader, enjoy viewing this document.

1. **Overall Description**
   1. **Product Perspective**

This product is designed for individuals who are suffering for free riding or lazy team members. This application will make those people comfortable by team recruiting place, team member rating and team members and communication place. The product can also help TAs or professors who want to make fair and convenient team member assignments. The rating system will help the lazy person to work harder because the system will save the rating for the individual permanently, and other users can use the rating for future team projects.

* + 1. **System Interfaces**

All services are hosted on AWS Cloud and utilize the resources provided by AWS. There are four types of servers, a web-frontend server that renders web page views and sends data to clients, a web application server, a mattermost server, and a service-proxy server that connects web and mattermost.

These servers are built in docker image. These images are uploaded to the repository of the Amazon ECR to define the task of the Amazon ECS.

Two or more subnet groups are load balanced by the Amazon Application Load Balancer, and each subnet group is auto scaled automatically by the Amazon Auto Scaling Group.

Each container inside the ECS can also be connected to the Amazon S3 responsible for uploading and downloading files and to the Amazon RDS responsible for DBMS.

* + 1. **User Interfaces**

An interface is provided through the web page. The page is different for users.

The students can access three main pages.

One is a simple information and searching, team building page for students. Another is the team project communication platform and the other page is specific comments for individual students.

The professors or TAs can access the team management page that provides a team member managing function. Also, they can use the team project platform to manage some notices or students’ inquiry.

* + 1. **AWS Instance Interfaces**

Each instance is hosted in the Seoul Region. In addition, the minimum spec for each instance is t4g.xlarge (4 vCPUs, 16 GiB memory).

* + 1. **Web browser Interfaces**

The system is intended for HTML living standard and suitable best at Chrome.

* + 1. **Communications Interfaces**

User device and server communicate with HTTP protocol in JSON format.

* + 1. **Operations**Login
    - Users can login using a university account.
      1. **System administrator**
  + Class management and comments management
    - Make classes or delete classes. Assigning a pro fessor and students in the class.
    - Read the comment and blind and delete inappropriate comments
  + Link User database
    - Getting and updating user accounts data using the university system.
      1. **Students**
  + My Profile
    - Students can write their information or resume for team matching.
    - Average rating is represented
    - The comments of the student are represented
  + Member Searching
    - Student can search or sort classmates for team members they want
    - Student can go to specific classmate’s mypage by clicking the classmate’s card
    - Students can make a team building request. The request will go to team member candidates first. If all members accept the request, the request will go to Professor, or TAs finally.
  + Team Communication Space (Mattermost)
    - Students can give an instant message to other classmates or the professor for team matching or asking.
    - Each class has one Mattermost server and each team is given its own private channel.
    - Can use some useful team project tools such as gantt chart, and ghost leg.
      1. **Professors & TAs**
  + Team management
    - Can see the list of team building requests and the list of teams already made.
    - Can choose the team matching algorithm for students who don’t have a team.
  1. **Product Functions**
     1. **Log-in**

User information is already in the database by administrator, so each individual including students, professors and TAs can just log-in with their University account.

* + 1. **Manage My information**

If the user is a student, the user is on My information page. On the page, the user can see their information such as introduction, comments. The user can modify their information and can ask the system administrator to blind or delete some inappropriate comments on their information. The user can make a nested comment to each comment for the user.

* + 1. **Choose a course**

After logging in, users can select a course among the courses they are taking.

* + 1. **Get student information list**

After Selecting the courses, get the information list of classmates that include student ID, name and rating. Sorted classmate list is shown. Users can click the name of a certain classmate.

* + 1. **Get student specific information**

If users click classmate’s name in the student information list page, the page shows the specific information of that classmate. The specific information contains overall rating, comments and resume.

* + 1. **Deliver a team build request to professor**

A student can make a team in the student information list page. When the request is made, the request goes to the team member candidates. If all candidates accept the request, the request is delivered to TAs or professors. If the request is accepted, a channel for his/her project team is made in the team project communication platform.

* + 1. **Team project communication platform**

Team project communication platform is the Mattermost hosted server. A user can communicate in the channel with his/her team. Even in the non project term, students can give an instant message to someone for a team building. Also the instant messaging can be used for the question to the professor or TAs. In the team private channel, team members can chat with each other and share some files.

A user can use functions that are useful for team projects such as gantt chart and ghost leg.

* + 1. **Rating team member**

After the project is done, a user can leave comments and ratings of team members in their student specific information page.

* + 1. **Managing the teams**

Professor can make some regulations making the team. Professor can check whether there are team building requests. If the professor approves the team building request, the team is added to the team list and the channel for the team is made in the team project communication platform.

Professor can check the teams and members in the courses.

Professor can assign teams for the students who failed to make a team.

* + 1. **Updating user data**

The system administrator shall get the user account data from the university server. And the data shall be synchronized with the university server.

* + 1. **Construct / Destroy the Course**

The system administrator may construct the course or destroy the course.

* + 1. **Report management**

The system administrator may delete or blind the comments that are reported.

* 1. **User Characteristics**
     1. **System Administrator**

System administrator is limited to someone who has sufficient knowledge of the system and who has a general understanding of the system. It is assumed that the system administrator has sufficient capabilities to deal with system problems. Also, it is assumed that the system administrator has knowledge about the database management system to synchronize our database with a university’s user information well.

* + 1. **Professor & TA**

The professors and TAs are the users that want to make meaningful and fair team projects. The professor and TA want to know how projects are done by many teams. Most of them have a warm heart to make a team for outsiders who failed to make a team.

* + 1. **Student**

Students are assumed to be people who have to make a team and do team projects for the course. It is assumed that students are not accustomed to rate the other students, and some of them may have an inclination to be a free rider.

* 1. **Constraints**

The system will be designed and implemented based on the contents mentioned in this document. Other details are designed and implemented by selecting the direction preferred by the developer, but the following items are observed.

* Use the technology that has already been widely proven.
* The service response speed does not exceed 3 seconds.
* Develop the student team project tool such as Gantt chart or ghost leg.
* Avoid using technology or software that requires a separate license or pays for royalty. (Exclude this provision if this is the only technology or software that the system must require.)
* Decide in the direction of seeking improvement of overall system performance.
* Decide in a more user-friendly and convenient direction
* Use open source software whenever possible
* Consider the system cost and maintenance cost
* Consider future scalability and availability of the system
* Optimize the source code to prevent waste of system resources
* Consider future maintenance and add sufficient comments when writing the source code
* Develop with HTML5, CSS3, and Javascript ES6
  1. **Assumptions and Dependencies**

All systems in this document are written on the assumption that they are designed and implemented based on web site and open source. Therefore, all contents are written based on the HTML5, CSS3, and Javascript ES6.

1. **Specific Requirements**
   1. **External Interface Requirements**
      1. **User Interfaces**

[Table 3] Login page

|  |  |
| --- | --- |
| **Name** | **Login page** |
| Purpose/Description | For user Log-in |
| Input source/ Output destination | Host server / Client  Client / Host server |
| Range/  Accuracy/  Margin of error | N/A |
| Time/ Velocity | Asynchronous user input/ Instant execution of a user instruction |
| Relationship with other input/outputs | After receiving all the inputs, the user device transmits the input data to the server for processing the input data and request desired output data |
| Format and configuration of screen | 1. A screen shows the ID box and PW box for log in.  2. Users are to click a desired button to interact with the system |
| Format and configuration of window | N/A |
| Data type | text data that contains ID and PW |
| Exit message | “Log-In FAILED!” (If log - in failed) |

[Table 4] Student Profile Page

|  |  |
| --- | --- |
| **Name** | **Student Profile Page** |
| Purpose/Description | A User can see the information of the candidate team members on this page.  Or the user can make his/her resume or manage comments for being an attractive student for other recruiters. |
| Input source/ Output destination | Host server / Client |
| Range/  Accuracy/  Margin of error | At first, there is no rating and there is no resume for the student. |
| Time/ Velocity | Asynchronous user input/ Communication time between the server and the user device |
| Relationship with other input/outputs | The comments and rate that is acquired by “Member Searching page” is shown on this page. |
| Format and configuration of screen | 1.The resume, name and average ratings are shown at the top of the window.  2.The ratings and comments of the specific user are shown. Some comments could be blinded. There is text besides each comment that notifies where is the course and when is the course this comment comes from. |
| Format and configuration of window | There is a button with the symbol of a siren for reporting or blinding the malicious comments. The report is shown to the administrator.. |
| Data type | Text, Image, table |
| Exit message | N/A |

[Table 5] Course Selection Bar

|  |  |
| --- | --- |
| **Name** | **Course Selection Page** |
| Purpose/Description | For selection the course that the user is currently involved in. |
| Input source/ Output destination | Host server / Client  Client / Host server |
| Range/  Accuracy/  Margin of error | N/A |
| Time/ Velocity | After all the input data were received to the user device/ Communication time between the server and the user device |
| Relationship with other input/outputs | The buttons link the corresponding course page. |
| Format and configuration of screen | There are course buttons. If a user clicks a button, the user can enter the corresponding course page. |
| Format and configuration of window | N/A |
| Data type | HTTP query |
| Exit message | N/A |

[Table 6]Member searching page

|  |  |
| --- | --- |
| **Name** | Member searching page |
| Purpose/Description | Present a list of team members that want to be a team with the user.  Present a list of students who were the previous team members. The user can rate them by clicking their name.  Present a list of students that include the student’s name and rating.  The user can learn more about the student by clicking his/her name. |
| Input source/ Output destination | Host server / Client, Client / Host server |
| Range/  Accuracy/  Margin of error | N/A |
| Time/ Velocity | After all the students data is sorted by rating. |
| Relationship with other input/outputs | The student can accept or deny the team request at the top list.  If the user clicks the name of students, the page is linked to the “Student Profile Page” of that student.  If the user clicks the name of students who was a previous team member, the student can make comments and rate the student.. |
| Format and configuration of screen | 1. The people who want to be the same team with the user are shown at the top.  2. The people who have to be rated (previous team members) are shown at the top of the page.  2. The list of students of the class are shown at the middle of the page. As default, the list is sorted by name. It can be sorted by rating.  4. The student name is the hyperlink for pages of the other “student profile page” |
| Format and configuration of window | A user can accept or deny the team request.  A user can make the team request by clicking “Make team building request”.  A user can choose the sorting algorithm. |
| Data type | Table, Text |
| Exit message | N/A |

[Table 7] Team Building Request Page

|  |  |
| --- | --- |
| **Name** | **Team Building Request Page** |
| Purpose/Description | A User can make a team building request on this page.  A User can find future team member in the below student list  When a user makes a request, the alarm goes to future team members. If all of them agree to team building, the request goes to the professor. |
| Input source/ Output destination | Host server / Client |
| Range/  Accuracy/  Margin of error | If there is no searching result, the search result table is empty.  It is not allowed to make duplicate team members. |
| Time/ Velocity | Asynchronous user input/ Communication time between the server and the user device |
| Relationship with other input/outputs | The team building request is used to alarm the future team member and finally used to make a request going to the professor. |
| Format and configuration of screen | 1. In the team members table, the user information is added by default.  2. A user can search students at the below search table.  3. A user can click a student in the search table to add that student to team members.  4. A user can click the “make a request” button to make a team building request. |
| Format and configuration of window | N/A |
| Data type | Text, table |
| Exit message | N/A |

[Table 8] Team communication page

|  |  |
| --- | --- |
| **Name** | **Team communication page** |
| Purpose/Description | This page would be implemented by using Mattermost.  One course gets one server.  One channel gets one channel.  Users can communicate with team members by instant messaging.  Users can communicate with each other by instant messaging. |
| Input source/ Output destination | Host server / Client  Client / Host server |
| Range/  Accuracy/  Margin of error | N/A |
| Unit | Table |
| Time/ Velocity | After all the input data were received to the user device/ Communication time between the server and the user device |
| Relationship with other input/outputs | N/A |
| Format and configuration of screen | 1. The professors can give a notice throughout the course at “notice” channel.  2. There are channel buttons on the left, so when a user clicks that, the user can start team-communication.  3. When the user clicks the button ’Direct messages’, the user can message to the others |
| Format and configuration of window | N/A |
| Data type | Image, Text, file |

[Table 9] Team Management Page

|  |  |
| --- | --- |
| **Name** | **Team Management Page** |
| Purpose/Description | The professor can check the progress of the team making.  The professor can check the people who don’t have a team yet.  The professor can approve the team matching. |
| Input source/ Output destination | Client/ Server, Server/ Client |
| Range/  Accuracy/  Margin of error | N/A |
| Unit | N/A |
| Time/ Velocity | The computing time for executing automatic team making algorithm |
| Relationship with other input/outputs | The page gets the information of the teams and their members.  (The information is acquired by “Team Building Request Page”.) |
| Format and configuration of screen | 1.At the top, there are requests for the team matching. The professor can approve or deny the requests.  2.Below, the team information of teams that are approved are shown by list.  3.Below, the students who don’t have a team yet are shown. The professor can choose to use automatic team matching for them. |
| Format and configuration of window | There are buttons for accepting or denying team requests.  There is a button for automatic team building. |
| Data type | Table, HTTP Query |
| Instruction type | N/A |
| Exit message | N/A |

* + 1. **AWS Architectures**

[Table 10] AWS Architecture diagram

|  |  |
| --- | --- |
| **Name** | **AWS Architecture diagram** |
| Architecture |  |
| Purpose/Description | Our service operates on at least two groups of AZs. Each request is properly load balanced to multiple AZs via Application Load Balancer. This method ensures that the service is running reliably even when the system is congested with heavy traffic.  Each AZ has a public subnet and a private subnet. Instances on private subnet are only accessible on public subnet's Bastion EC2. The bastion server increases the security of each container.  When high CPU load occurs on the instances of each AZ, Amazon Auto Scaling Group automatically increases the instance to lower the CPU load. If the CPU load is lowered, the Auto Scaling Group will delete the instance to lower the billing cost. |
| Input source/ Output destination | Internet gateway/ECS instance, ECS instance/Internet gateway |
| Range/  Accuracy/  Margin of error | N/A |
| Unit | HTTP request |
| Time/ Velocity | Instant reaction |
| Relationship with other input/outputs | Related to all inputs/outputs from server |
| Format and configuration of screen | N/A |
| Format and configuration of window | N/A |
| Data type | HTTP |
| Instruction type | HTTP status code |
| Exit message | N/A |

* + 1. **Communication Interfaces**

[Table 11] Communication interface of client and host

|  |  |
| --- | --- |
| **Name** | **Client and Host** |
| Purpose/Description | Each client requests the connection to the host, requesting list of results of student information  Host provides a student list and student information to the client |
| Input source/  Output destination | Client/Host server |
| Unit | packet |
| Time/ Velocity | At least 10Mbps |
| Relationship with other input/outputs | Related to all inputs/outputs from server |
| Format and configuration of screen | N/A |
| Format and configuration of window | N/A |
| Data type | Query |
| Instruction type | Query statement |
| Exit message | N/A |

* 1. **Functional Requirements**
     1. **Use Case**

[Table 12] Use case of account register

|  |  |
| --- | --- |
| **Use case name** | **Account Register** |
| Actor | Admin of system |
| Description | It is a process that an admin registers all new students when the new semester starts. |
| Normal Course | Admin gets data for registering new students.  The registration process will be provided using mysql database, etc.  The admin is required to fill in some additional information according to the database column. The required information includes  Student ID  Default password  Name  Age  Gender  department  After filling out the form, all of the new users are registered.  Once user registered, course will be updated automatically every semester. |
| Precondition | The user is not registered to the system yet  The admin enters correct information.  The same Student ID should not be overlapped with that of other users |
| Post Condition | The password should be encrypted by bcrypt and saved to the user management database for security |
| Assumptions | N/A |

[Table 13] Use case of log-in/out

|  |  |
| --- | --- |
| **Use case name** | **Log-in/out** |
| Actor | Registered user |
| Description | Log-in is a process when a registered user of the system tries to get into the system for using the service  Log-out is a process when a user who has logged-in tries to get out of the system |
| Normal Course | <Log-in>  A user who has already registered to the system as a member wants to use the service from the system  The user enters the student ID and password that were the user set for registration or default password.  If the information is correct, the system allows the user to get into the system and the user is now able to take advantage of all the service provided by the system  <Log-out>  If the user wants to get out of the system, the user clicks the button ‘Log-out’  If the user closed the application without logging-out, the system arbitrarily closes the session for that user |
| Precondition | <Log-in>  The user should already be registered to the system  <Log-out>  The user should be in a logged-in status |
| Post Condition | The user should be connected to network |
| Assumptions | N/A |

[Table 14] Use case of profile

|  |  |
| --- | --- |
| **Use case name** | **Profile** |
| Actor | Registered user |
| Description | It is the page where the user is able to check and modify the user information that the user entered.  Users can also see the search history. |
| Normal Course | A user clicks the button ‘Student Info’’ which direct him to the profile page  The user can check the information and modify the personal information such as nickname, age, gender, which was required when registration.  The user can hide other people’s comments.  The user can require removing comments to the admin.  After modifying the information, the user clicks the button ‘Finish’ to return to the previous page |
| Precondition | The user should be in a logged-in status and connected to network |
| Post Condition | After exit from profile page, the information should be delivered to the server and updated |
| Assumptions | N/A |

[Table 15] Use case of making team

|  |  |
| --- | --- |
| **Use case name** | making team |
| Actor | Professors |
| Description | Professors can make teams using many options. |
| Normal Course | After logging- in, the user clicks the button ‘Student list’  Professor can choose one option for making the team.  Students can make their team before the professor makes teams.  Team will be made by an algorithm that the professor choosed. |
| Precondition | The user should be in a logged-in status and connected to network |
| Post Condition | The students of course should be saved to database |
| Assumptions | N/A |

[Table 16] Use case of team project space

|  |  |
| --- | --- |
| **Use case name** | team project space |
| Actor | Registered user |
| Description | This is a page where a user do their team project |
| Normal Course | The user clicks a communication button.  The user choose one space that he wants go  The user can see team project history and team chatting.  The user can upload team projects and do chatting. |
| Precondition | The user should be in a logged-in status, and should be in a team project. |
| Post Condition | N/A |
| Assumptions | N/A |

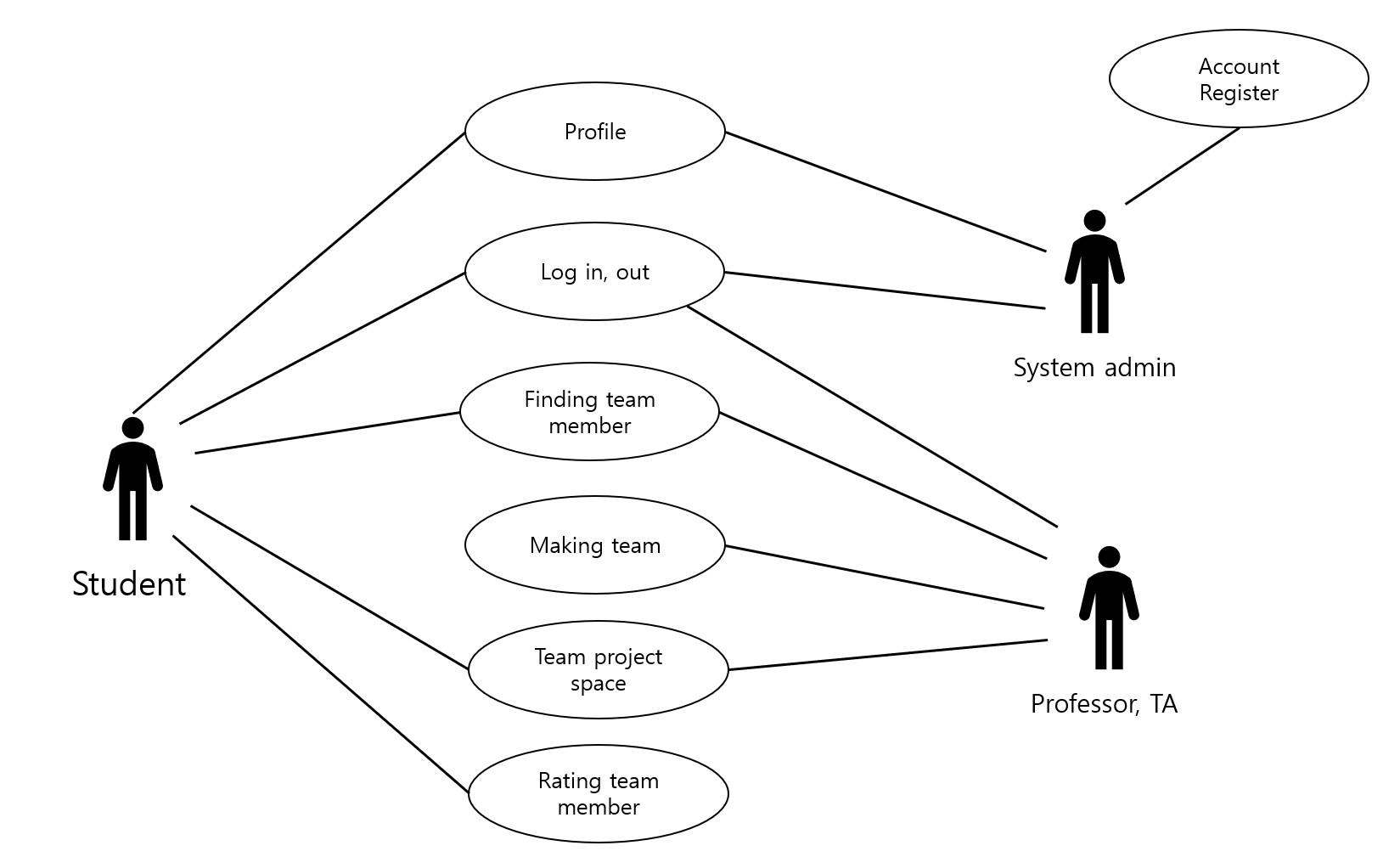
[Table 17] Use case of finding team member

|  |  |
| --- | --- |
| **Use case name** | finding team member |
| Actor | students |
| Description | This is a process than students makes team with team member he wants |
| Normal Course | There is a button ‘Team building request’.  students can see the list of people who have the courses after clicking the button.  he can make team with them who are not in team  The message will send to team member after he submit team.  after every team member admit the message, professor will get message.  after professor admit it, team is confirmed. |
| Precondition | The user should be in a logged-in status and connected to network |
| Post Condition | The new team should be saved to database |
| Assumptions | N/A |

[Table 18] Use case of team member rating

|  |  |
| --- | --- |
| **Use case name** | **Move to** **team member rating** |
| Actor | students |
| Description | This option will be given rating system of your team member. |
| Normal Course | students click team member’s profile after team project  students can give score for some options and comment.  students upload team member score. |
| Precondition | The user should be in a logged-in status and connected to network  The user should be same team. |
| Post Condition | N/A |
| Assumptions | The user did team project with them |

* + 1. **Use Case Diagram**



[Figure 1] Use case diagram

* + 1. **Data Dictionary**

[Table 19] Student-from school

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Key** | **Constraint** | **Description** |
| student number | PK,FK | Not Null | user’s student ID |
| name |  | Not Null | User name |
| email |  |  | User’s email |
| Account\_ID |  | Not Null | User’s university account ID |
| password |  | Not Null | User’s university account password |

[Table 20] Profile

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Key** | **Constraint** | **Description** |
| student number | PK | Not Null | User’s student number |
| Comment |  |  | Comment from others |
| Major |  | Not Null | User’s major |
| rating |  |  | User’s overall rating |
| project |  |  | experience and results of projects of the user |
| available language |  |  | User’s available language |

[Table 21] time schedule

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Key** | **Constraint** | **Description** |
| student number | PK | Not Null | User’s student number |
| monday |  |  | User’s available time on monday |
| tuesday |  |  | User’s available time on tuesday |
| wednesday |  |  | User’s available time on wednesday |
| thursday |  |  | User’s available time on thursday |
| friday |  |  | User’s available time on friday |
| saturday |  |  | User’s available time on saturday |
| sunday |  |  | User’s available time on sunday |

[Table 22] Professor

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Key** | **Constraint** | **Description** |
| name | FK | Not Null | Professor’s name |
| email |  |  | Professor’s email |
| Office room | PK | Not Null | Professor’s office room |

[Table 23]Course

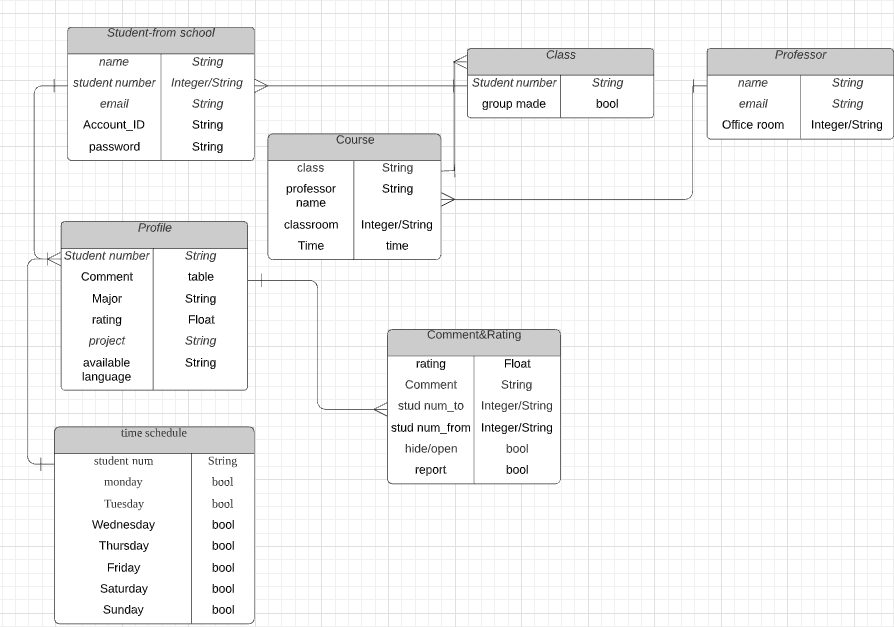
|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Key** | **Constraint** | **Description** |
| class | PK | Not Null | class name |
| professor name |  | Not Null | professor name of the class |
| classroom |  | Not Null | classroom where the class helds |
| time |  | Not Null | time of the class |

[Table 24] Class

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Key** | **Constraint** | **Description** |
| student number | PK | Not Null | student number of student who listens to this class |
| group made |  | boolean | whether the student made the group or not |

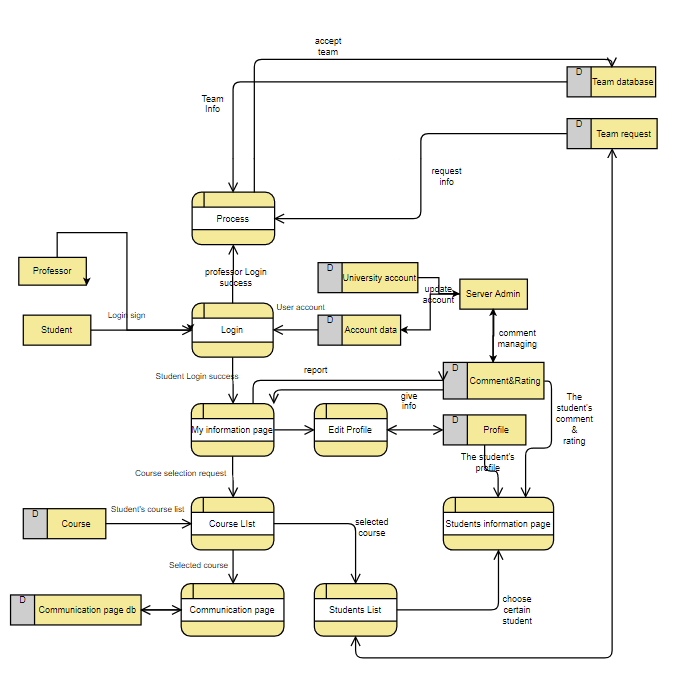
[Table 25] Comment&Rating

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Key** | **Constraint** | **Description** |
| rating |  | Not Null | rate the team members |
| Comment |  | Not Null | Add some comments about that |
| student number\_to | PK | Not Null | student number of receiver of the comment |
| student number\_from |  | Not Null | student number of sender of the comment |
| hide/open |  | boolean | whether this comment is hided or open to others |
| report |  | boolean | whether this comment is reported by the receiver |



[Figure 2] Entity Relationship Diagram

* + 1. **Data Flow Diagram**



[Figure 3] Data flow diagram

* 1. **Performance Requirements**

The following requirements are based on estimates and may be changed in the final application.

* + 1. **Static numerical requirement**
* Team manage page can be only accessed by professors and assigned TA’s.
* The system supports user account data(ID,password,major) from the university data. The system does not support multiple connections on the same IP address. However, after disconnecting, users can switch accounts and access.
* The system should run smoothly on Windows 10 and version of HTML5, CSS3, and Javascript ES6
  + 1. **Dynamic numerical requirement**
* The system runs smoothly for 200 simultaneous connected users. And the system handles at least 30 active user accounts and profiles per one class.
* The web should respond within 10 seconds.
* The login process should be completed within 3 seconds.
* student search and sorted results should be displayed within 5 seconds.
* When modifying or writing a profile, data must be stored on the database within 3 seconds, and database updates must be completed within 3 seconds.
* When we store data on the database, make sure that all the contents are well stored on the database at the right position.
* When some user makes some requests to make groups, all the selected members should get some letter which they approve or decline the request.
  1. **Logical Database Requirements**

The system uses a relational database management system from mariaDB. In the database, it should be able to store the user's basic information achieved from the university data. After that, if the user writes some additional information about their profile, the information written by the user should be stored in the database. Also, university course and professor information is also achieved from the university and there are comment and rating table for the user to write comments and give some rates about their team members at the end of the semester. For the sorting and searching student, some sql queries are used to show the results of the sorting and searching.

* 1. **Design Constraints**

The system should contain only components that can be distributed under the MIT license. The system should be able to access from Chrome, and the administrator must be able to access and manage our web site with some additional UI. The system must be designed to run on mariaDB database.

* 1. **Standards compliance**

All programs in the system are written in conventional programming techniques. The names of functions and variables in the program use camel notation. System management tool should be access through web browser in accordance with HTML5 standard.

* 1. **Software System Characteristics**

Software system characteristics are revealed through non-functional requirements. So, this section describes several non-functional requirements of the system. Non-functional requirements are classified Product Requirements, Organization Requirements, External Requirements as follows.

* + 1. **Product Requirements**

Product requirements specify or constrain the runtime behavior of the software. Our system should satisfy following requirements.

* + - 1. **Interface Requirements**

This is one of most important non-functional requirements of our system. The system should be easy to use by non-technical user and should be organized in such a way that user errors are minimized. The usage of technical terms should be minimized, explain it easily if needed. Each user shall be able to use all the system functions instinctively without learning manual. That means, user interface should be simple, instinctive, and easy to use.

* + - 1. **Performance Requirements**

There are several recommendation grouping algorithm for the leftover student. Therefore, we should provide some algorithm with similar processing time because if some algorithm takes long time and some algorithm are short, usability of the system will be degrades. We are planning to give the result of the final group within 10 seconds.

* + - 1. **Usability Requirements**

The system should provide its purposed service which is making a group of students and providing some communication space for them. Grouping system should be operated with no errors like one student on two teams.If the system uses some recommendation grouping algorithm, results should provide some reliability. At the communication service we provide, there are some tools to effectively do the team project. Therefore, the average number of errors made at the communication service we provide should not exceed 5 percent.

* + - 1. **Security Requirements**

The users should be properly announced that some of their communication will be checked by their professors and TA. At the login page, we have to make sure that an unauthorized user cannot gain access to the page and if the user who has successfully log-in to our platform does not have administrator authority, we have to make sure that this user cannot access to the team manage page. Also, we have to prevent some information about the user and block or blind some comments if there are some wrong information about the user.

* + 1. **Organizational Requirements**

These requirements are broad system requirements derived from policies and procedures in the customer’s and developer’s organizations

* + - 1. **Environmental Requirements**

Many of the web sites consist of versions of HTML5, CSS3, and Javascript ES6. Therefore, our platform is also based on this version.

* + - 1. **Operational Requirements**

Users of the system shall identify themselves using their student number and names. Users can see other classmates' information. Information of classmates is detailed when we click the student number. Details include some past projects, majors, available language and some times that they are available. The system works at any-time.

* + 1. **External Requirements**

It covers all requirements that are derived from factors external to the system and its development process.

* + - 1. **Safety / Security Requirements**

The platform should guarantee that any personal information that are confidential could not be accessed by an external systems or users. The system should be safe enough to prevent user’s data from being damaged by external systems or users.

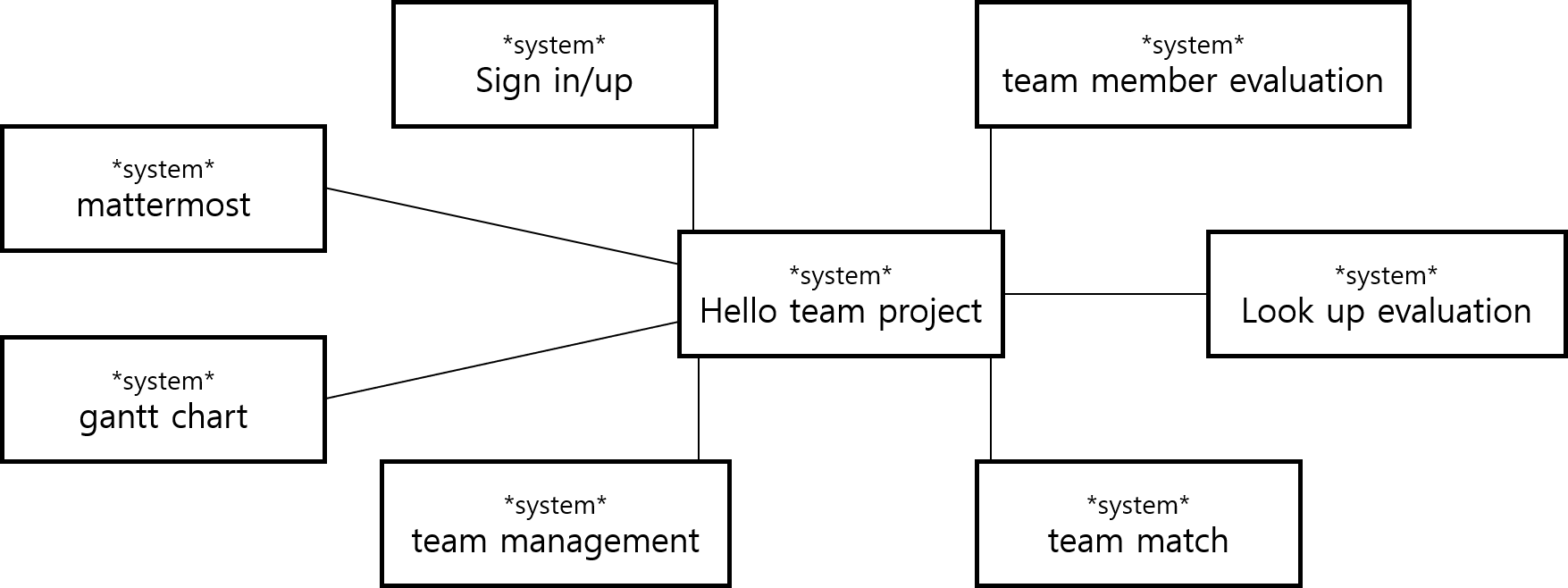
* + - * 1. **Regulatory Requirements**

The Hello System has a function that evaluates a team member, a person. Writing about others on limited but public websites. So, this function should be subject to legal interpretations of such articles as defamation, malicious comments, and slander. In addition, personal information about each person must be protected, which must also be subject to the legal interpretation of the country in which the system is serviced.

* 1. **Organizing the Specific Requirements**

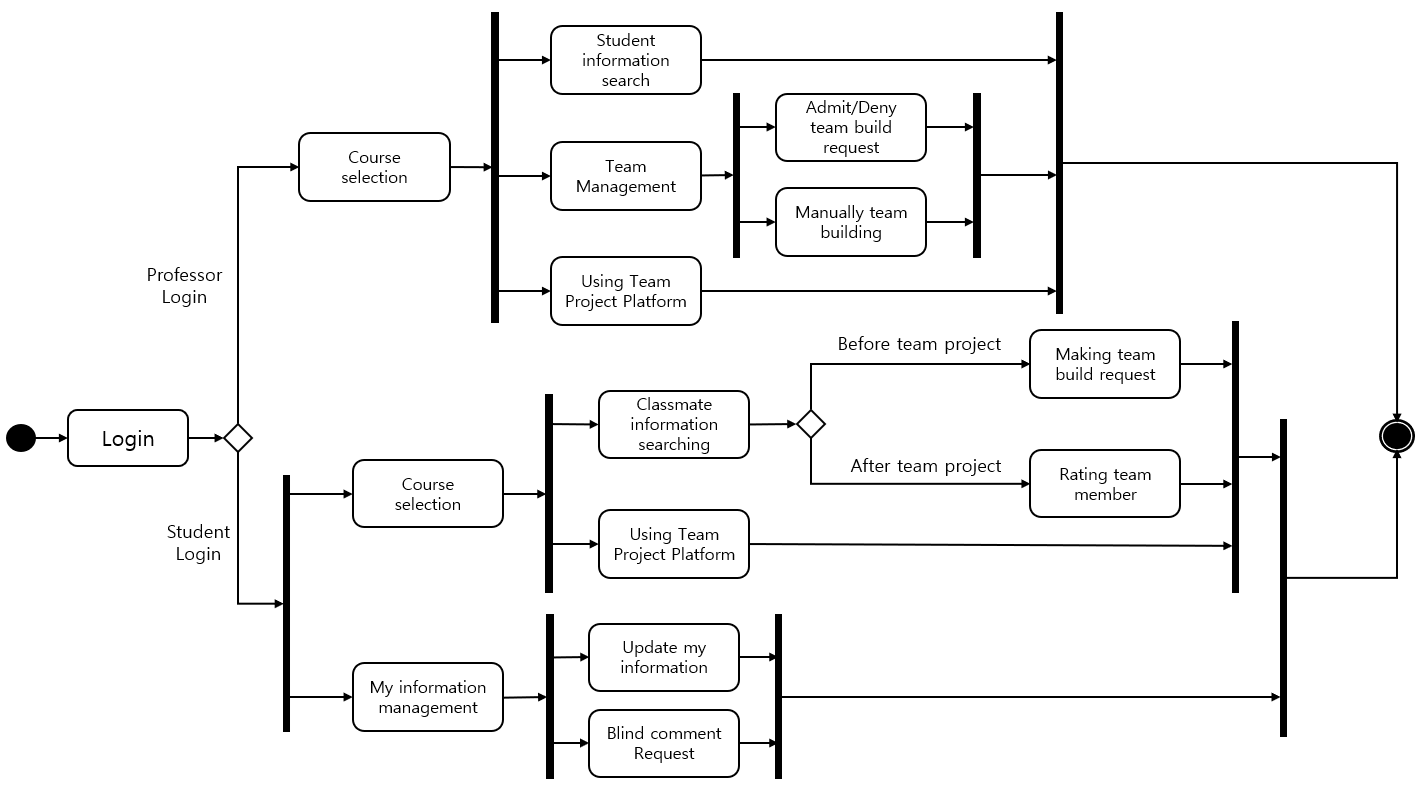
In this section, we describe the system model using graphical notation based on Unified Modeling Language (UML) and tabular form. System model describes the relationship among the system, sub-systems, components, and surrounding environments, showing more specific requirements.

* + 1. **Context Model**



[Figure 4] Context model

* + 1. **Process Model**



[Figure 5] Overall process model

* + 1. **Interaction Model**

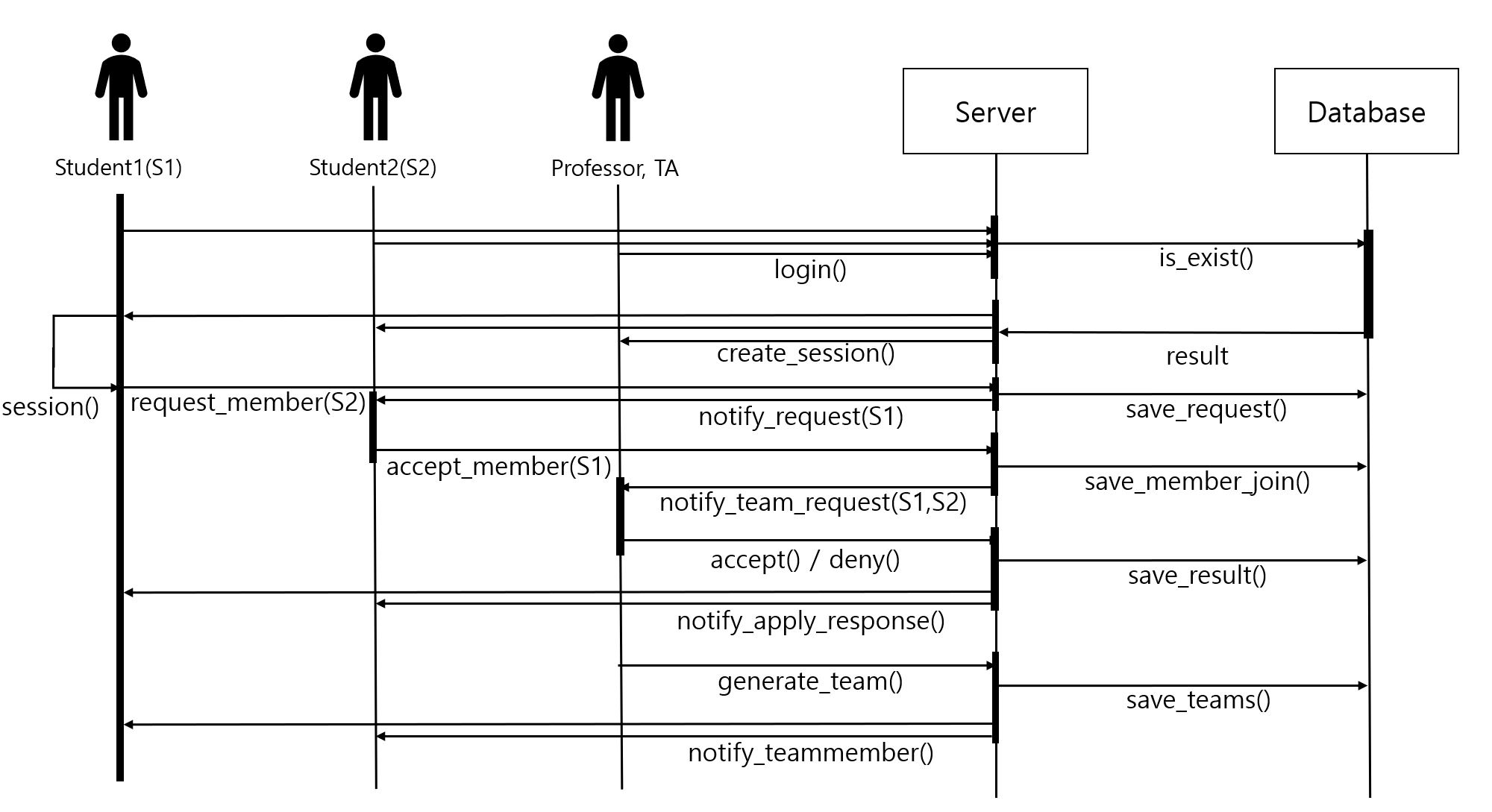
See 3.2.2. Use Case Diagram

* + 1. **Behavior Model**
       1. **Data Flow Diagram**

See 3.2.4. Data Flow Diagram

* + - 1. **Sequence Diagram**

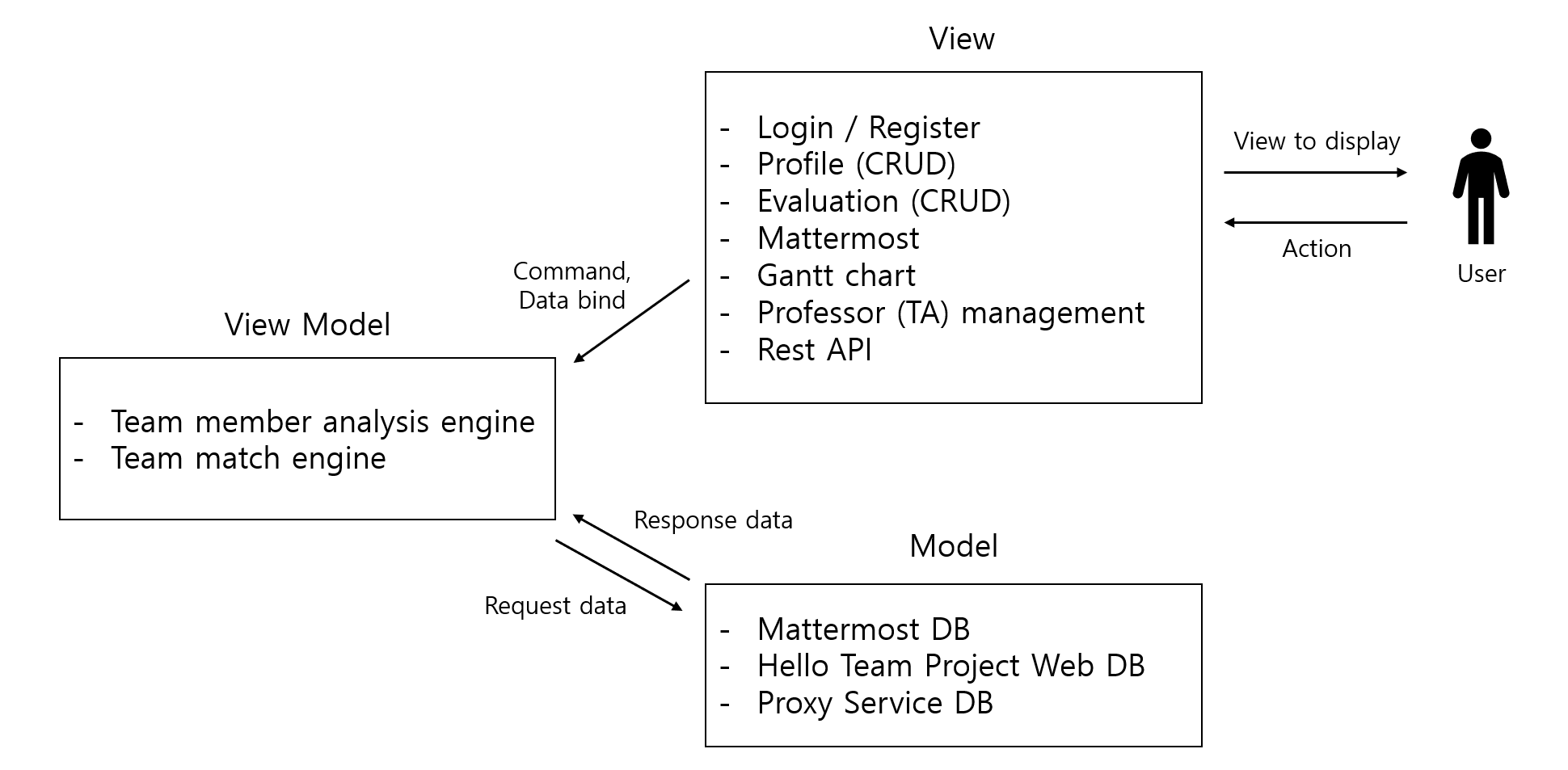
It describes mainly the sequence of processing team building. The diagram describes the situation when two students (S1, S2) make a team.



[Figure 6] Sequence diagram

* 1. **System Architecture**

This Section presents a high-level overview of the anticipated system architecture, showing the distribution of functions across system modules. We use MVVM patterns as graphical models of the architecture.



[Figure 7] System architecture of the system

* 1. **System Evolution**

In this section, we describe the most fundamental features of our system: teaming, setting up a meeting space between team members, and evolution of team member evaluation functions. This evolution is caused by changes in user requirements, changes in the preferred method of team projects over time, evolution of software, and changes in AWS policies or infrastructure. This section is important to system designers because it can help them predict future changes to the system and avoid conflicts with existing systems as much as possible.

* + 1. **Limitation and Assumption**

Our system is applicable to team projects at a discussable level through text chat among team members. If teams need voice-based discussions, not message-based discussions, or additional services such as real-time voting, external systems such as Google docs and KakaoTalk are needed. Our current system assumes that we proceed with the most simplified team project that enables all team members to discuss with only text-based chat and online task upload systems. In addition, current team member evaluation systems do not have systems that classify fraudulently written evaluations such as malicious evaluations, deviations, etc. at the system level. Therefore, it is assumed that not all users make malicious assessments.

* + 1. **Evolutions of Team Project Method**

The more team projects online, the more diverse the method of team projects will be created. The more diverse the team project methods become, the less competitive our system becomes. As team project methods become more diverse, our systems must also have more convenience features and plug-ins. For example, if a discussion is held rather than a discussion among team members, there is a limited amount of time for each panel to speak. After a certain period of time, there should be a function to automatically mute the speaker's microphone.

Therefore, we need to receive a survey of the types and methods of team projects that are mainly used by system users and implement additional features with many responses.

However, too many functions on a single page can make the organization of pages messy and make it difficult for users to learn and classify them. This leads to the problem of failing to utilize good features in the right place and eventually leads to a significant loss of use of the system. Therefore, a function that allows professors or TA to activate only the functions necessary for team projects may be required in advance.

* + 1. **Evolutions of Team Matching**

There are two ways to create a team in the current system. The first is the way the professor or TA automatically matches the team through a certain mechanism, and the second is to match the team among the desired students and apply to the professor. However, it is difficult for many students to match teams according to the number of teams and apply to professors. In fact, in many cases, two or three people apply for a team and the remaining number is randomly assigned, but the function of applying only for a part of the team is not currently in the function specification. It has a high priority in system evolution as most team applications occur.

1. **Supporting Information**
   1. **Software Requirement Specification**

This software requirements specification was written in accordance with the IEEE Recommendation (IEEE Recommended Practice for Software Requirements Specifications, IEEE-Std-830).

* 1. **Document History**

[Table 26] Document History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Writer** |
| 2021/04/17 | 0.1 | From 1.1 to 3.1.1 | Jae Hyun Ju Byeong Su Woo |
| 2021/04/18 | 0.2 | Addition of 2.4, 3.1.4,  Revision of 3.1.1 | Jae Hyun Ju  Byeong Su Woo |
| 2021/04/19 | 0.3 | Addition of 3.2.3 | Hee Soo Jung |
| 2021/04/19 | 0.4 | From 3.3 to 3.7 | Hee Soo Jung |
| 2021/04/20 | 0.5 | Revision of 2.1.1, 2.1.3 | Seongwook Lim |
| 2021/04/20 | 0.6 | From 3.7.3.2 to 3.8.4.1  Revision of 3.9 | Seongwook Lim |
| 2021/04/21 | 0.7 | Revision of 3.10 | Seongwook Lim |
| 2021/04/21 | 0.8 | Revision of 3.2.4 | Jaehyun Ju |
| 2021/04/21 | 0.9 | revision of 3.2.1 | Jang Min |
| 2021/04/24 | 1.0 | distribution |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |