# Managing students' projects using cloud computing

System analysis and design in high level

Purpose: To understand and determine the scope of the system and its components and architecture.

Supervisors: Dr. Mouhib Alnoukari - Eng. Anas Abdulaziz

Students: Raghad al hossny – Kassem al Kelani

## project goals and services:

make a system which will automate and organized all aspects of students' projects completing process in Syrian private university (Junior Project, Senior Project 1, Senior Project 2), which is:

- Organizing the projects suggestions process that happened in the beginning of each semester.
- Full Automat the project registration process.
- Creating environment of valuable communication between supervisors and their own teams (increase the ability to track students' work in an organized manner).
- Managing and facilitation the projects evaluation process.

## The main actors with their roles and responsibilities:

1. student:
    - View the available projects list.
    - Request and register a project “in the begging of the semester”.
    - Exchange media with supervisor and other team members (communication).
    - View and track the projects evaluation plan, and execute its rules.

2. Supervisor:
    - Add a project suggestion.
    - Tracking, exchanging media, communication with his teams.

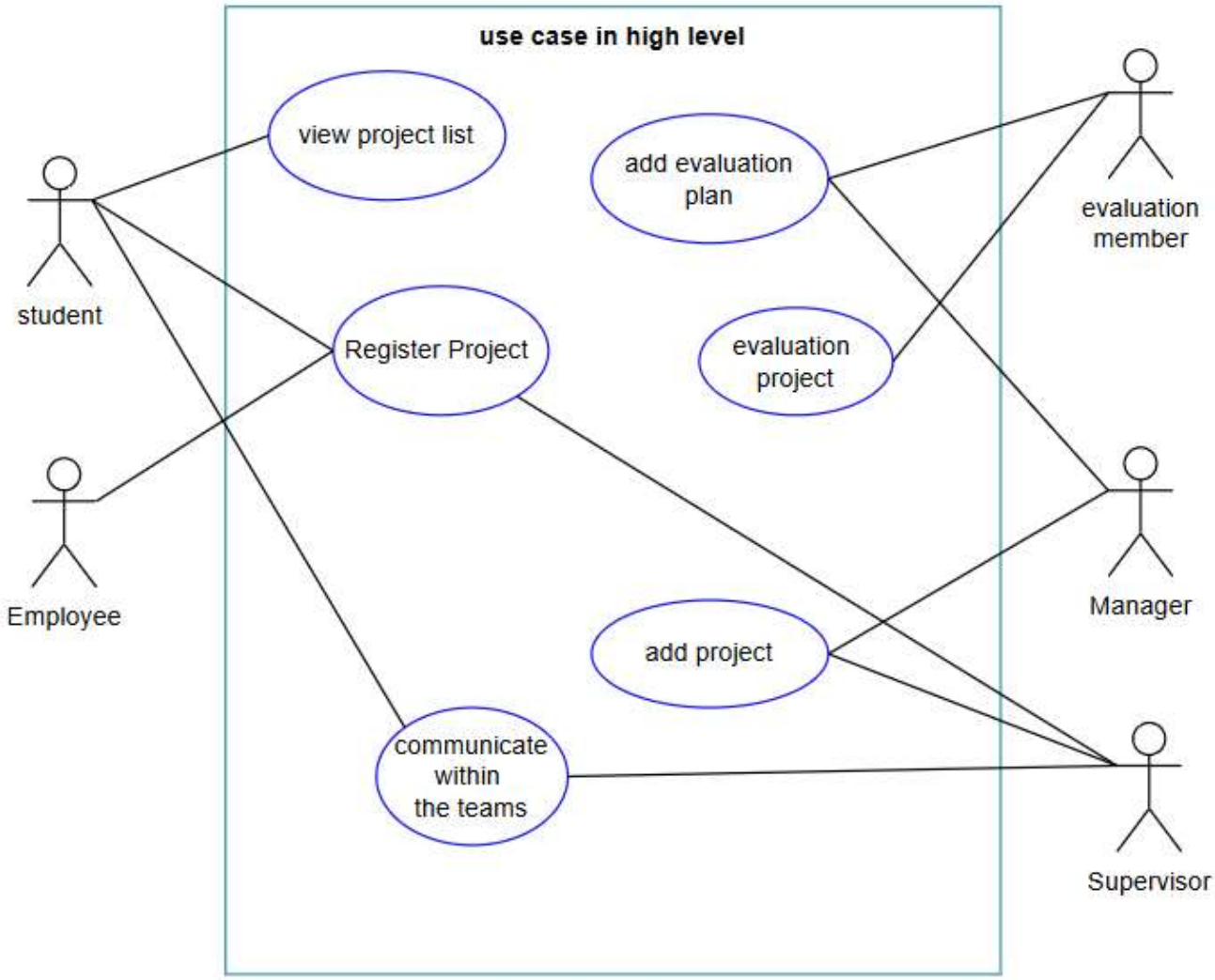
3. Manager:
    - Approve or rejects projects suggestions.
    - Approve and edit evaluation plan.
    - Track the progress of any team.
    - View simple reports.

4. evaluation team member:
    - make evaluation plan for the projects.
    - Evaluating teams.

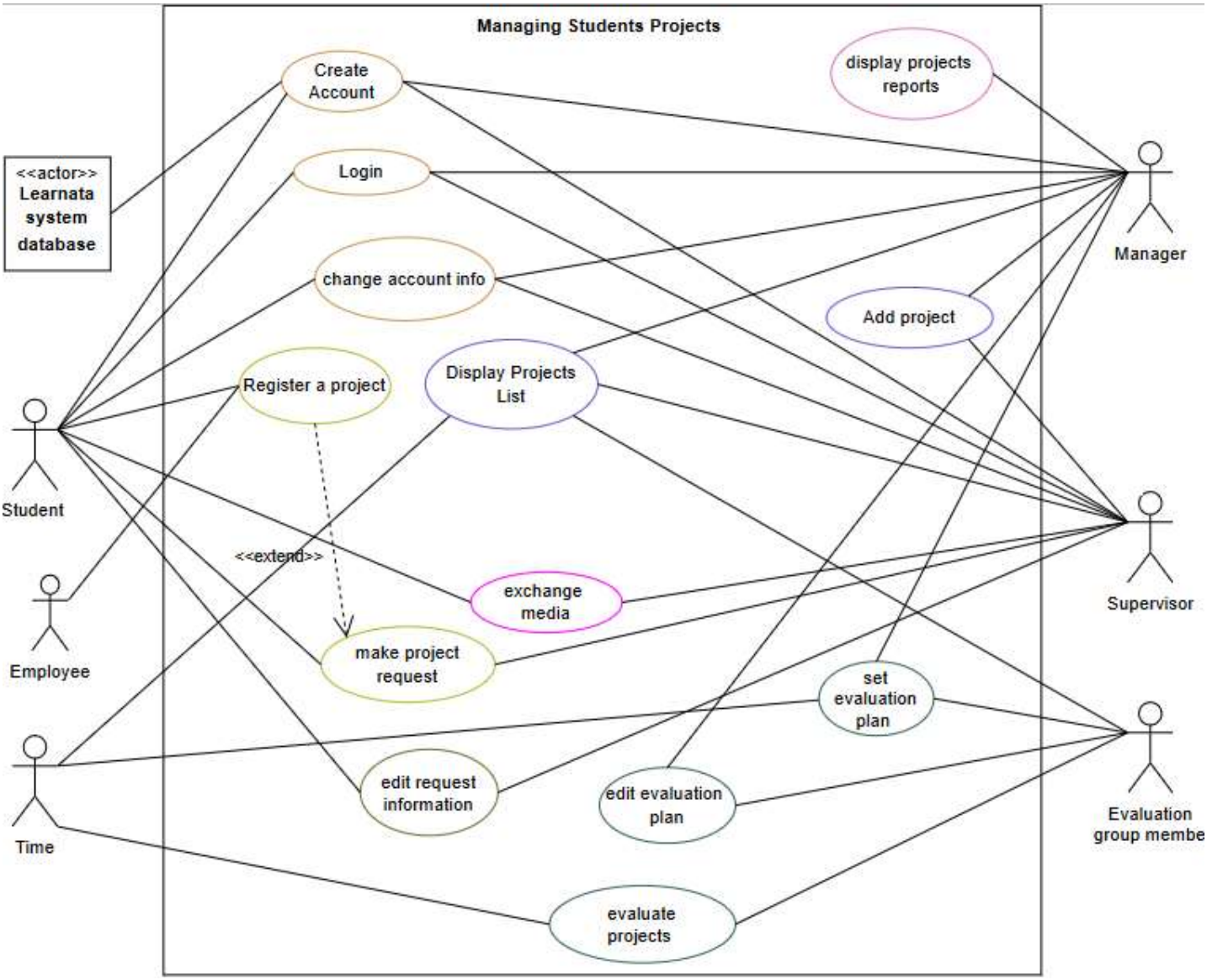
5. responsible employee:
    - receive registration requests from students.

6. The database of the university system “Learnata” (secondary actor):
    - to compare the ids (when user create his only account by his university id).
    - Check if the student information match with the project registration conditions (like: total number of his completed hours is more than or equal to 100).

## Use case diagram in high level:



## Detailed use case diagram:



## The specification of top three important use cases:

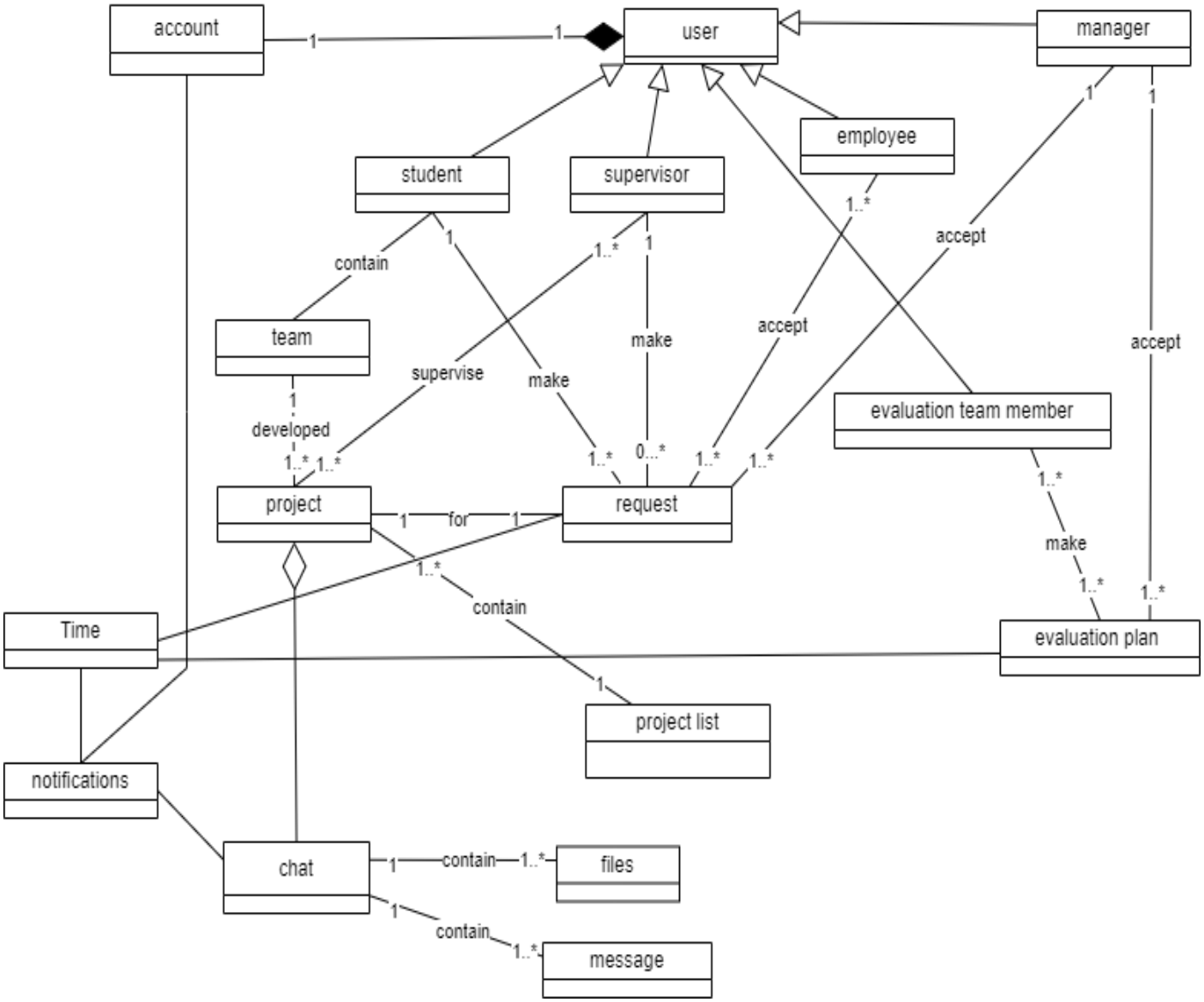
| | |
|---|---|
| Use case name | add project. |
| Participating Actors | initiated by supervisor<br>communicated with manager. |
| Flow of events | 1. First supervisor will ask to “make project suggestion”.<br>2. The system will ask the supervisor to enter project title.<br>3. Supervisor will enter the title.<br>4. System will compare titles in the database and return to the supervisor acceptance if there is no such project in the history, and ask him to complete the other informations.<br>5. Supervisor will enter the other informations required to make a new project (goal, description, plan and key steps).<br>6. The system will accept to make the project after complete the project information, and send the suggestion to the manager.<br>7. the manager makes an approve for the suggestion project.<br>8. System will send the approval message to supervisor, and add the project to the project list to be display to every on using the system. |
| Entry condition | the service is available (specific time determined by manager) |
| Exit conditions | project will be shown in the project list. |

| | |
|---|---|
| Use case name | make project request. |
| Participating Actors | initiated by student<br>communicated with supervisor, employee. |
| Flow of events | 1. After choosing an available project from project list, the student will take the option make request.<br>2. System will ask the student to add a team member.<br>3. Student will enter names and ids for his partners.<br>4. System will check the database for the ids and send emails for partners asked them to make a make request acceptance on their accounts.<br>5. Other students(partners) will enter their accounts and approve to make that request.<br>6. After this system will send the request to the supervisor of the project.<br>7. If the supervisor accepted it<br>8. system will send the approvement to the student and, registration request with project information and student to the responsible employee.<br>9. If the employee registers the project (he will check some conditions related to university rules outside our system responsibilities) he will choice accepted and done.<br>10. The system will send the acceptance message to the students, and make a communication environment for this team and their supervisor, and the service will end. |

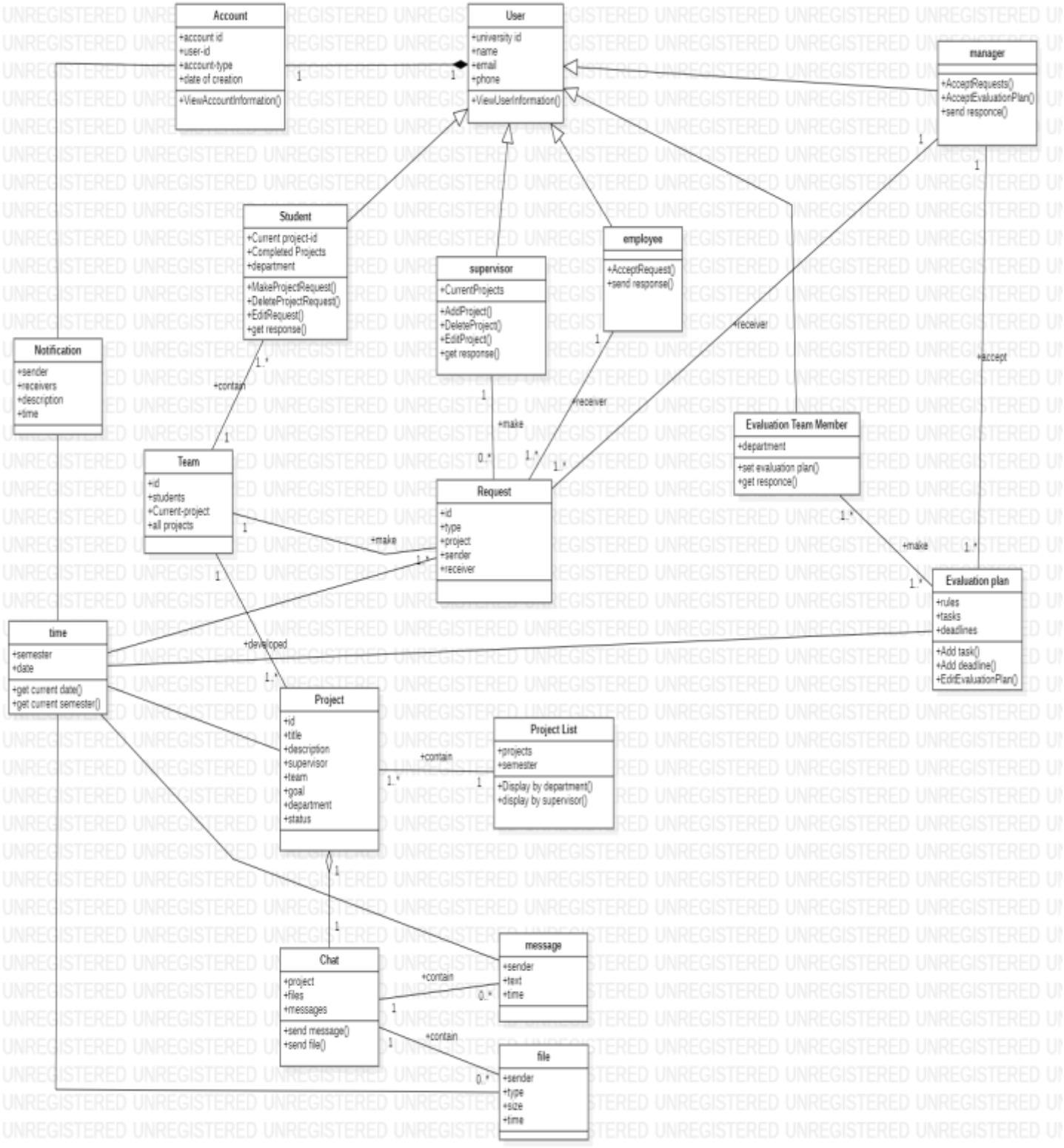
| | |
|---|---|
| Entry condition | the registration service is available (specific time), and all supervisors already added their projects (project list is available). |
| Exit conditions | student register their projects and have a communication page connects them with their team members. |

| | |
|---|---|
| Use case name | create student account. |
| Participating actors | initiated by student. |
| Flow of events | 1. Students will enter the system(website) first and ask to make account.<br>2. System will ask for their university id and some informations name, number, email (all information must be match with his information in the university system database).<br>3. Student will enter the information.<br>4. The system will communicate with the university system (learnata) database to check the id and other informations.<br>5. If the student is existed, the system will then check some conditions related to projects (number of completed hours more than or equal to 100, and some courses has to be complete), if all conditions are met the system will ask the user to enter a password for the account.<br>6. Student will enter a password.<br>7. System will check if that password met a determined security limit if that true system will make an account for the student and take him to it. |
| Entry condition | the account creator is student the student exist in the university and met some projects Conditions |
| Exit conditions | student has account and can start to use the system services. |

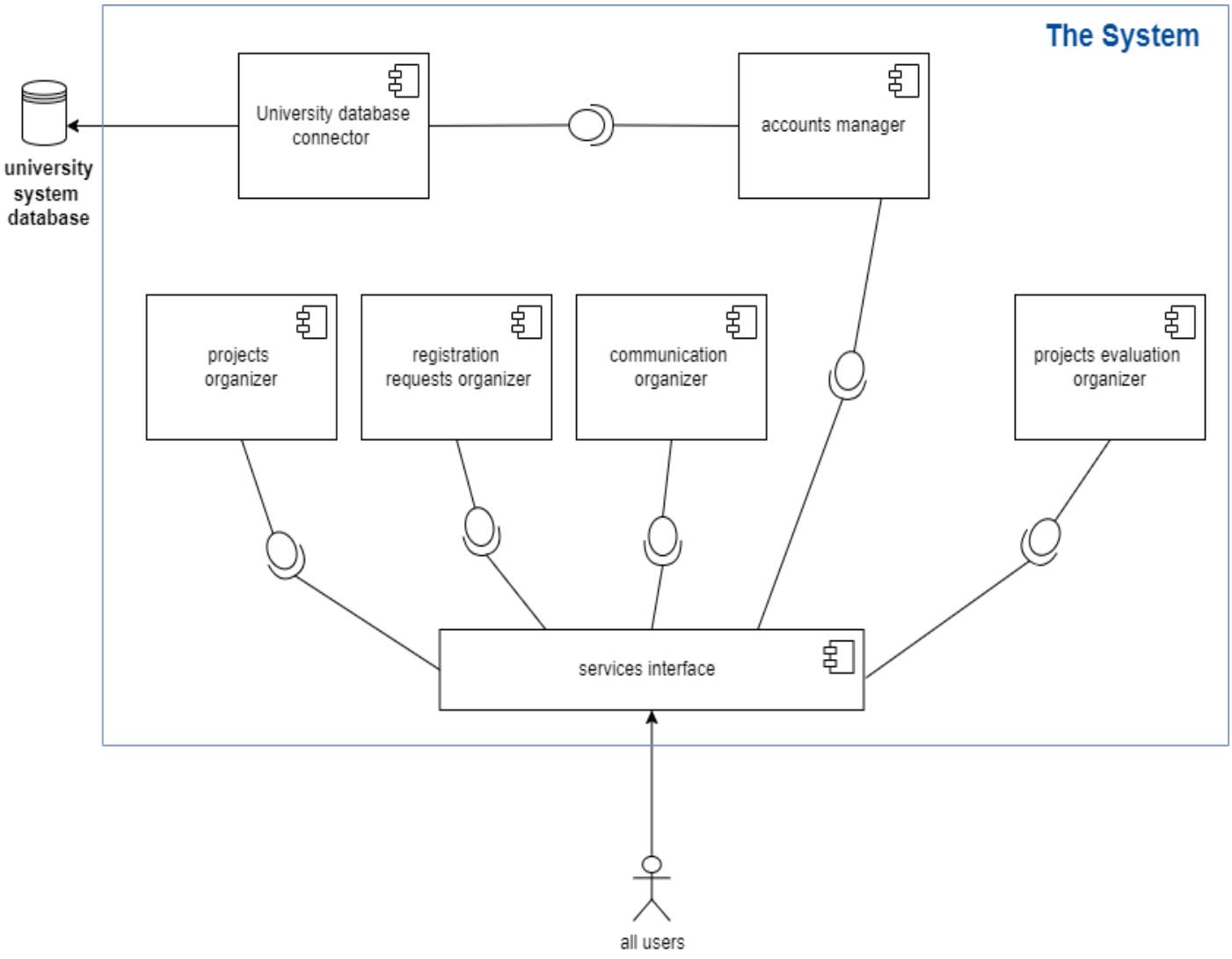
# Class diagram:



## Detailed class diagram:



## Components Diagram:



## Components responsibilities:

1. University database connector:
    - This component will communicate with the university database system when a user wants to make an account to check its informations and whether he is a member in the university (student or supervisor) by his id.
    - Also, it will check whether a student met with the conditions of the project registration, that determent by the university.
    - And it will give the result to the "Account manager component" so it can make a new account for this member.
2. Account manager:
    - Made everything related to accounts management like make new account after taking the approve from "University database connector component".
    - Delete account.
    - Edit account information.
    - It will give an interface for the "services interface component" only the output of its operations.
3. Projects organizer:
    - Organize the process of adding new projects by supervisors and take approve from manager.
    - Organize the process of displaying the projects list by multi options.
    - It will give an interface for the "services interface component" only the output of its operations.

4. Registration requests organizer:
    - Organize the process of making request for projects by students (team).
    - Organize the registration process for a project.
    - It will give an interface for the “services interface component” only the output of its operations.
5. Communication organizer:
    - Making a unique environment to communication for each team.
    - Organize messages and files exchanging process.
    - It will give an interface for the “services interface component” only the output of its operations.
6. Projects evaluation organizer:
    - Organize the process of making an evaluation plan (tasks and deadlines), each one related to each department, and make it visible for all teams to be a roadmap.
    - Organize the approval process from the manager for every edit operation.
    - It will give an interface for the “services interface component” only the output of its operations.
7. Services interface:
    - This component will have an interface from all the last 5 components to have their services and presented to the user so that user don’t have to know or communicate with all components otherwise only communicate with the “services interface component” and that will give a level of abstraction and encapsulation to all servicer.

## Architectural Style:

Note: the system will be developed to be a web application.

I will go first with Client – Server Architecture:

The client-server software architecture pattern is a model that divides a software system into two primary components: the client and the server. The client is the user-facing side of the system that handles the presentation and interaction with the user. The server is responsible for handling the business logic, data management, and other processing tasks. The client and the server communicate with each other through a network protocol.

Why choosing client –server architecture (advantages):

- Common to use with web applications: where the web browser acts as the client, and the web server handles all the backend logic and data management
- High performance: only two layers.

Disadvantages of client – sever architecture:

- Not for complex systems: with complex systems its better to distribute the logic into multi layers or components.
- Low level of security.

Because of this I will use other architecture inside the server side to solve the problems that come with client - server architecture. (Hybrid architecture)

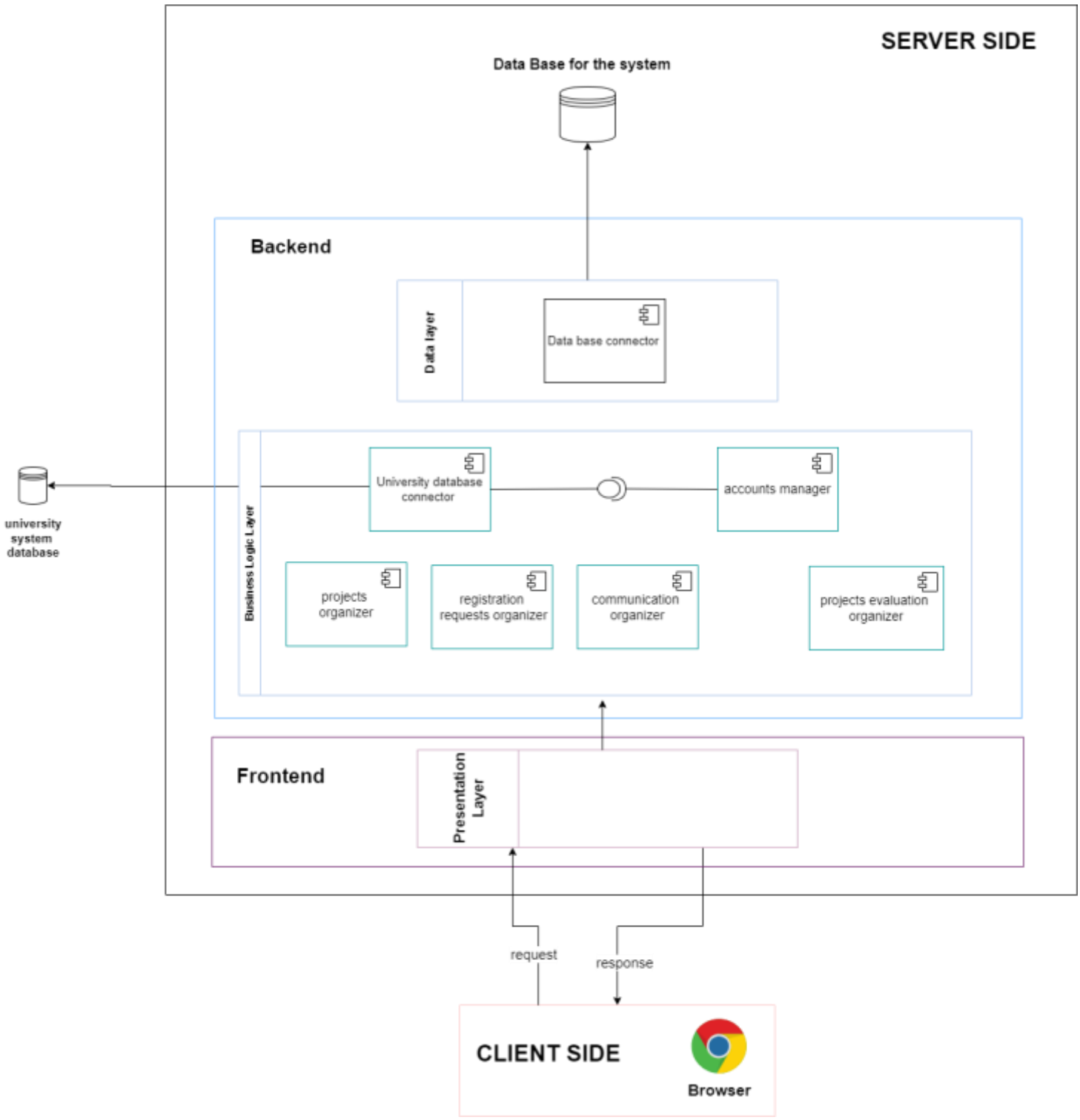
I will go with Layered Architecture for the server side (In a layered software architecture pattern, the system is divided into horizontal layers, each with its own specific responsibilities. The higher-level layers depend on the services provided by the lower-level layers, which enables them to be developed and tested independently).

because it will give us high level of security, the communication between each layer and the another is determent by the interfaces between them.

Also, it will separate the responsibilities so reduce the complexity. Note that both client – server and layered architecture gives us reusability.

So now we will have performance and security together.

For the architecture I delete the component “services interface”, because after using the layered architecture it will by default gives the encapsulation for services in the business logic layer that will connect with the frontend (the presentation layer).


SERVER SIDE
Data Base for the system
Backend
Data layer
Data base connector
Business Logic Layer
University database connector
accounts manager
projects organizer
registration requests organizer
communication organizer
projects evaluation organizer
university system database
Frontend
Presentation Layer
request
response
CLIENT SIDE
Browser

Sprint management:

Now, for the first sprint we will focuses on: “projects organizer component” means supervisor and some of the manager services

The requirements that we will developed:

| Req-ID | Title | Description | Actor |
|---|---|---|---|
| **1** | Create management account | Manager have only one unique related to his university id | Manager |
| **2** | Create supervisor account | Supervisor have only one account related to his university id | supervisor |
| **3** | Log in | Users can log in by their ids and password they choose | Supervisor, manager |
| **4** | Change password | Users can change password | Supervisor, manager |
| **5** | Change profile photo | Users can change account photo | Supervisor, manager |
| **6** | Request for add project | Supervisor can suggest a new project | supervisor |
| **7** | Edit request | Supervisor can edit the suggestion information as long as the manager doesn’t response | supervisor |
| **8** | Delete request | Supervisor can delete a suggestion | Supervisor |
| **9** | Accept or reject a project | Manager will get all projects and accept it or not | manager |

| | | | |
|---|---|---|---|
| **10** | Send response | Manager will make the response and the system will send it to the supervisor | manager |
| **11** | Get request response by email | Supervisors will get the response as a notification on their emails | supervisor |
| **12** | User friendly | The system interfaces should be easy to learn and use | |