Design Documentation

Created	@October 21, 2021 6:33 PM
Last Edited Time	@December 24, 2021 11:40 PM
Type	Technical Spec
Status	
Created By	
Last Edited By	
Stakeholders	

Course Information: BLG411E Software Engineering - Fall 2021

Report Name: Design Documentation

Project Title: Training Platform

Group Name: Group 2

Group Members:

- Ertuğrul Bektik - 150200709
- Mehmet Eymen Ünay - 040190218
- Saadet Sena Erdoğan - 150200731
- Umut Emre Bayramoğlu - 150200730

> 💡 **Notion Tip:** Create a new page and select `Project Kickoff` from the list of template options to automatically generate the format below. Learn more about database templates.

1. Introduction

1.1 Goal

This document helps the developer team to understand implementation basics. This way, the team can start to build their project knowing what should they expect from their system. Also, this document forces us to think about the whole architecture rather than only small pieces.

1.2 Contents

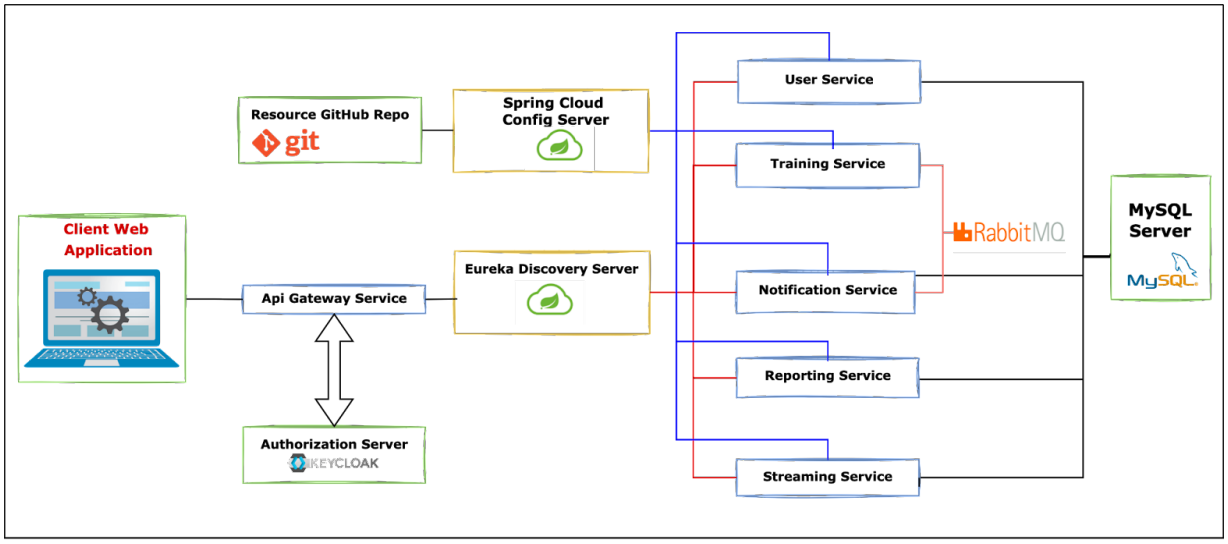
1.3 The Organization of The Document

Document is started with the purpose. After that, table of contents is provided. Later on, a high level architecture diagram for the whole system is included with its explanation for every part. Component diagram is added. Since this is a big system, adding every part to one component diagram was making reading hard. Therefore, for every component who has at least one subcomponent another component diagram is included. With these diagrams, system architecture section is concluded.

Moving on to the low level design part, a class diagram for each component is added. After that, for every use case a sequence diagram is included, too. This document is concluded with the data flow diagram.

2. System Architecture

2.1 System High-Level Architecture Diagram and Explanation



The high level system architecture of our project is shown above. The explanation of the services used in our architecture can be summarized as follows:

Api Gateway:

This service is used for API management of our microservices. When a request is received by this service, it handles and redirects it to the related microservice. For example, when the client sends a request to the 'api/training' endpoint, this request is redirected to the training service directly. In this way, the client app sends the requests to just one base URL instead of each microservices base URL. Besides that, authorization control of the received request is done in this service.

Authorization Server:

When the received request needs authorization, the Authorization Server checks the user access token that is inside of the request and performs the authorization of the user if the token is valid. Authorization server based on the **Keycloak** which is an open-source identity-access management tool developed by RedHat. OAuth2 authorization protocol is used on this service.

Eureka Discovery Server:

All microservices are connected to this server. Discovery server dynamically assigns the host and IP addresses of these servers. Also, it provides the functionality of intercommunication between microservices by a declarative web client called FeignClient.

Spring Cloud Config Server:

Config server keeps the external dependencies of microservices in one central place. In our project, the config server fetches these external dependencies from a GitHub repository named **microservices-config**. In addition to convenience of collecting the dependencies of the configuration server in a central place, the change to be made in one of the config files is to be updated without the need to rebuild the server.

User Service:

All system functionalities related to users are performed in this microservice.

Training Service:

All system functionalities related to trainings and lessons are performed in this microservice.

Notification Service:

This service is responsible for generating and sending the notifications.

Reporting Service:

This service is responsible for generating reports.

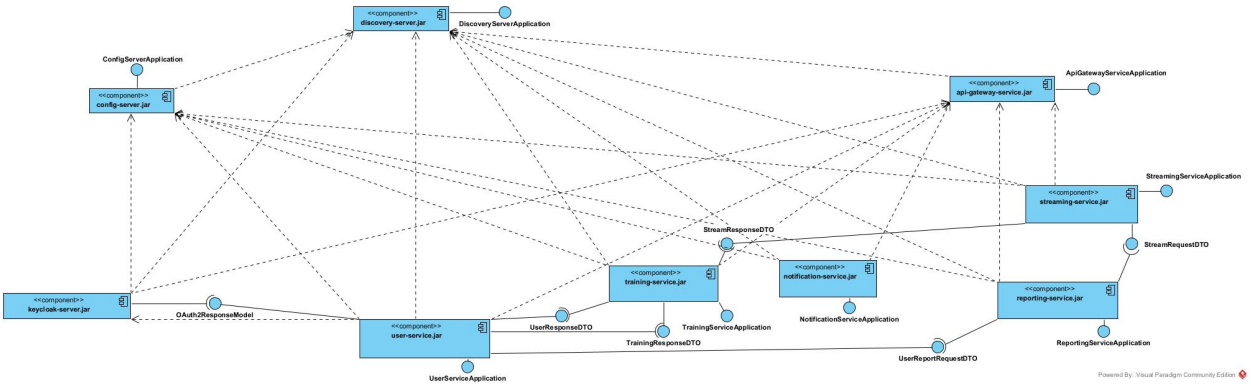
Streaming Service:

Streaming service delivers the offline training lesson video sources to the client. Also, it tracks the information about how long a user watched the video, did the user complete the video etc..

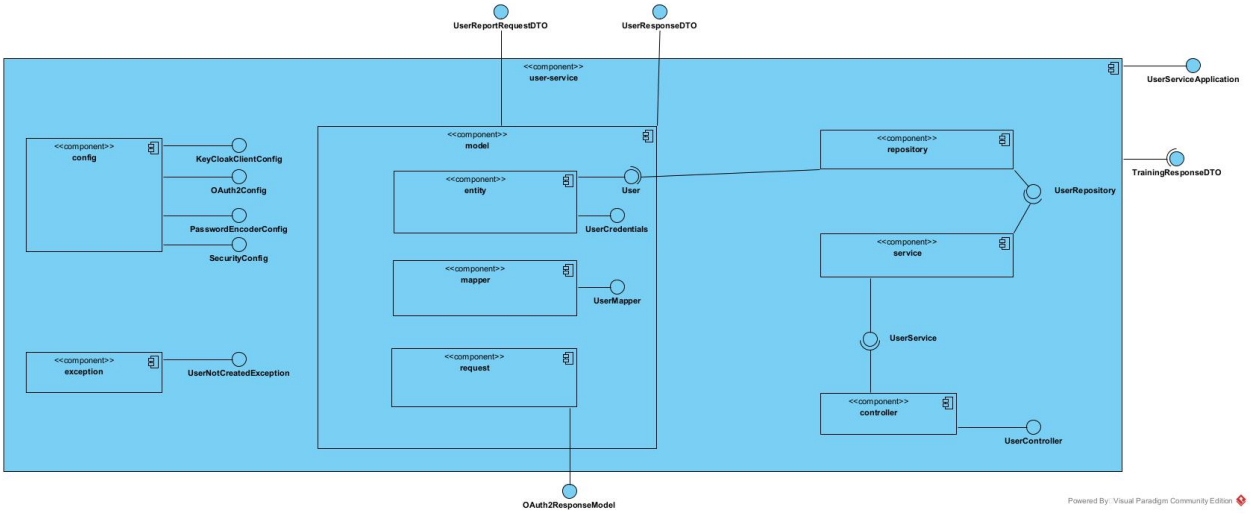
RabbitMQ Server:

RabbitMQ server enables the communication of training service and notification service. For example, when a user is assigned to a training, notification service is notified automatically and it sends the notification to the target user.

2.2 Component (Package) Diagram

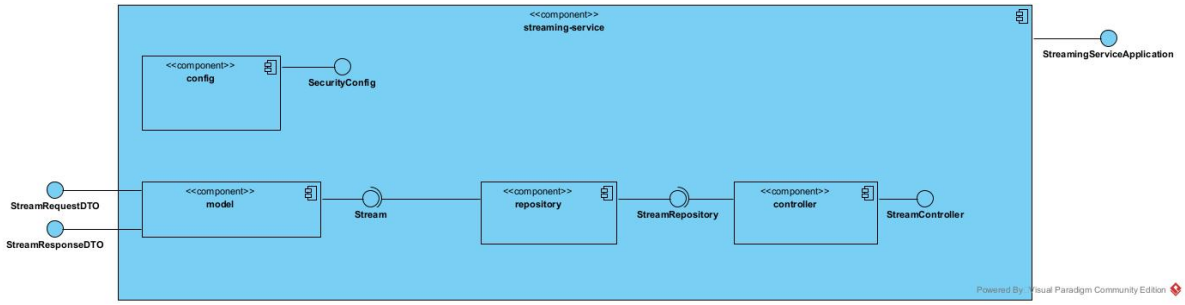


- From a general point of view, the project consists of these services. Each service has its own components, too. In order to keep things clear they are not shown here. For every one of them, component diagrams were drawn and included.
- Dashed arrows show dependencies. Microservices architecture is used in this project. End components do not have dependency on each other but there are some core components that every service depend on.
- To communicate between services some interfaces are used. They are included in the diagram.
- Not every component has subcomponents. Only the ones who have subcomponents are shown in the below.

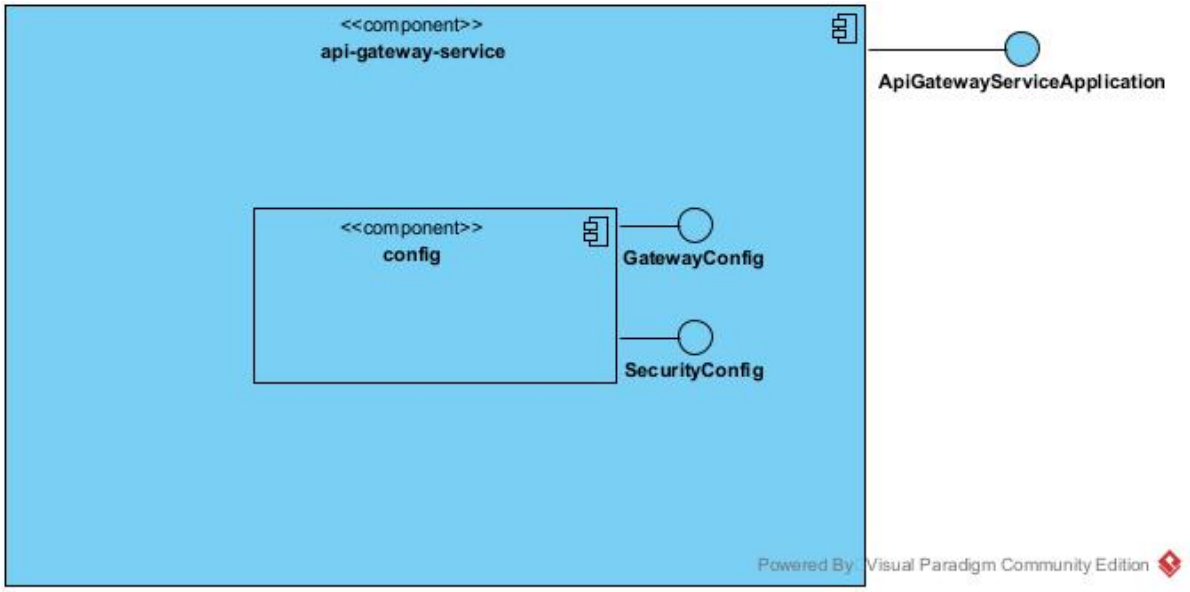


- This diagram is prepared for user service. Since this is a big diagram, including it in the general diagram would make reading harder, this is why it is included as a separate diagram.

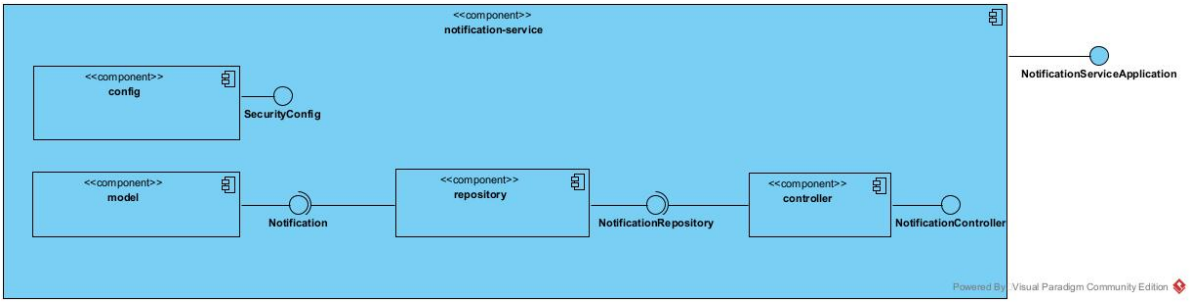
- User service has its own components inside. Some of them provide classes for other services. Rest is used internally.



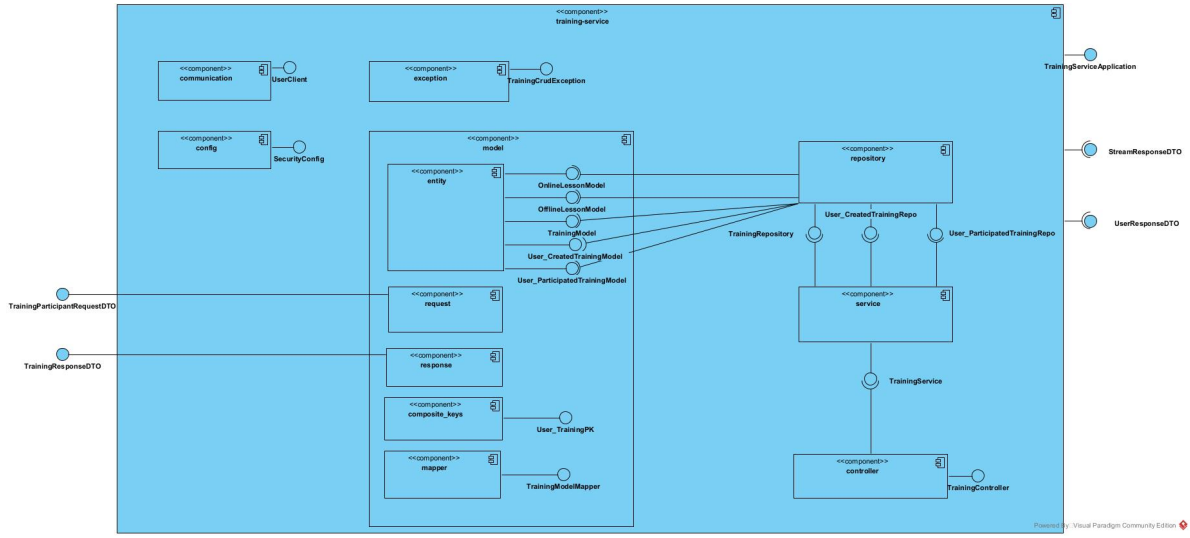
- This is the streaming service component diagram. Like other services, this one has its own components, too.



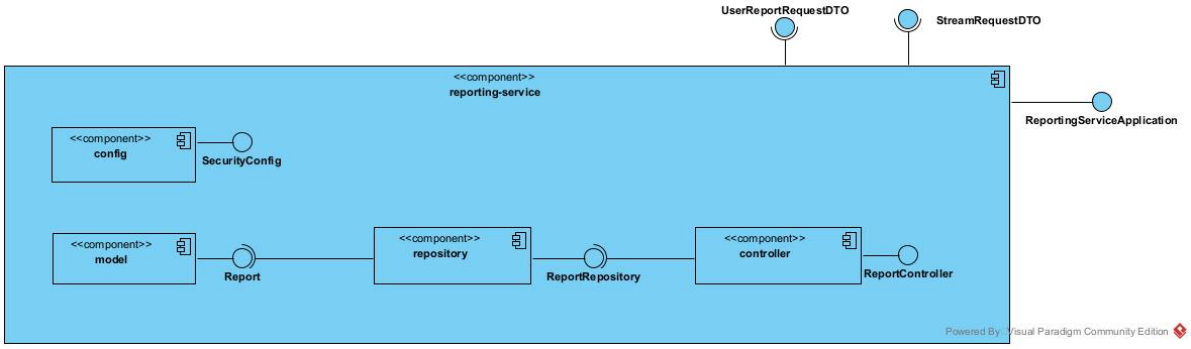
- API Gateway Service is one of the core services to enable microservices architecture. This service has one component inside, config component. The classes it provides maintenance for the system.



- In this project, notifications are sent to users from time to time. For this purpose, notification service is created.
- Like other services, notification service has its own components. They can be seen in the diagram.



- Training service is one of the big services.
- This service holds lessons too.
- It has its own components.
- Some classes are used internally, some of them are used to communicate with other microservices.

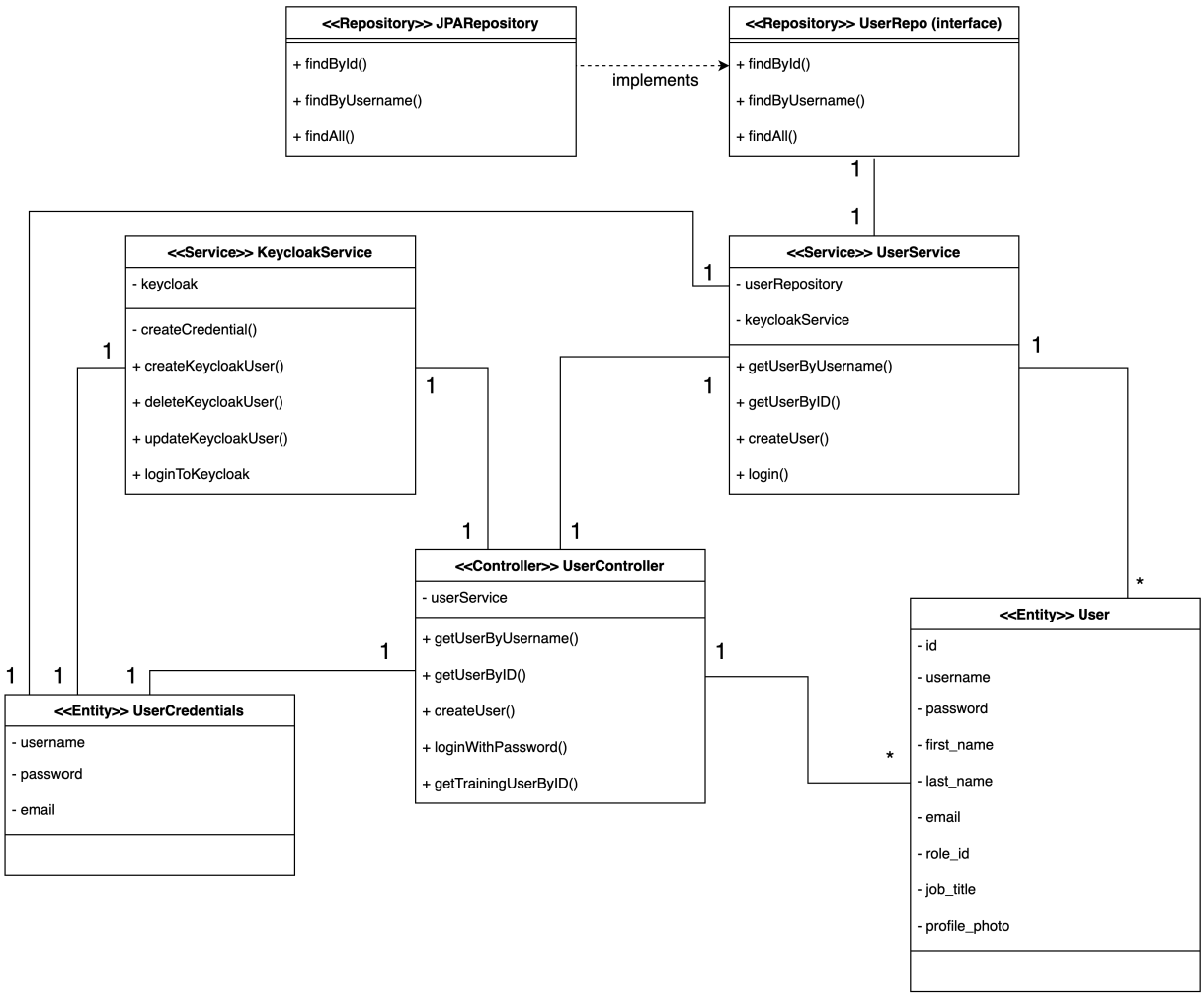


- In this project, a specific user can ask for reports about another user, this service is created for that purpose. (Hierarchy is managed in KeyCloak Service.)
- It has its own components inside.
- Some of the classes are used internally.
- This service is in communication with other microservices using some of its classes.

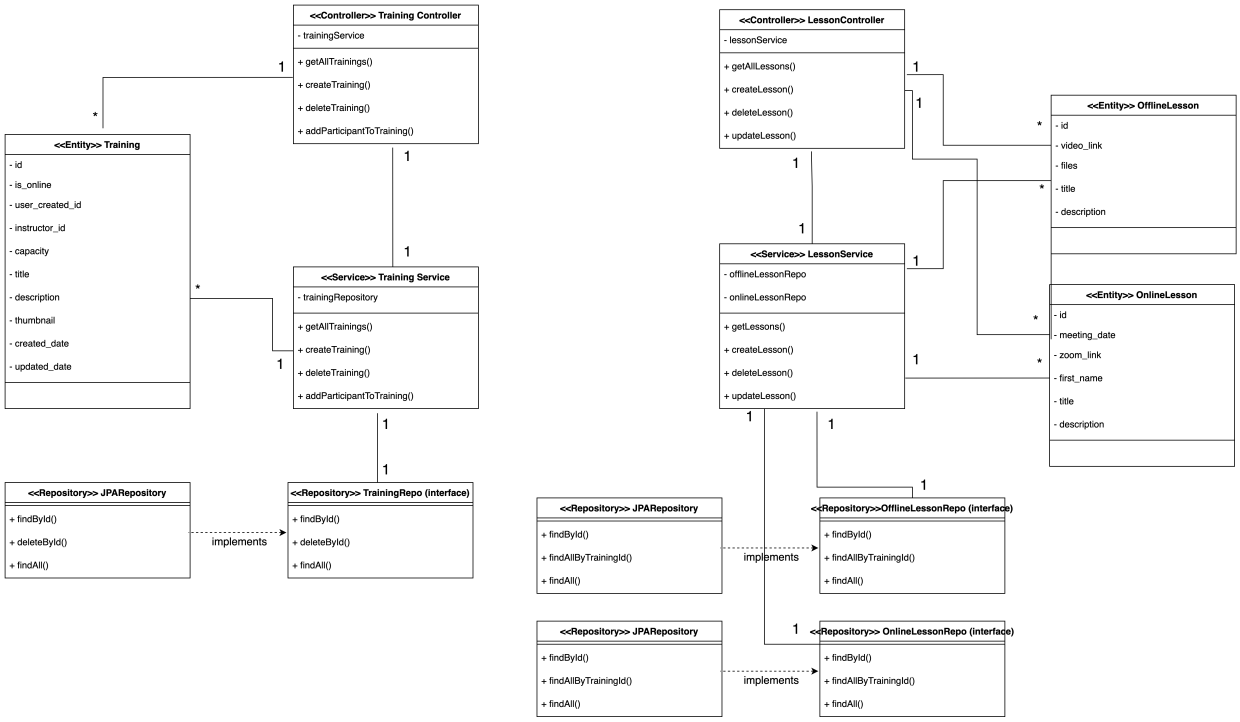
3. Low-Level Design

3.1 Class Diagrams

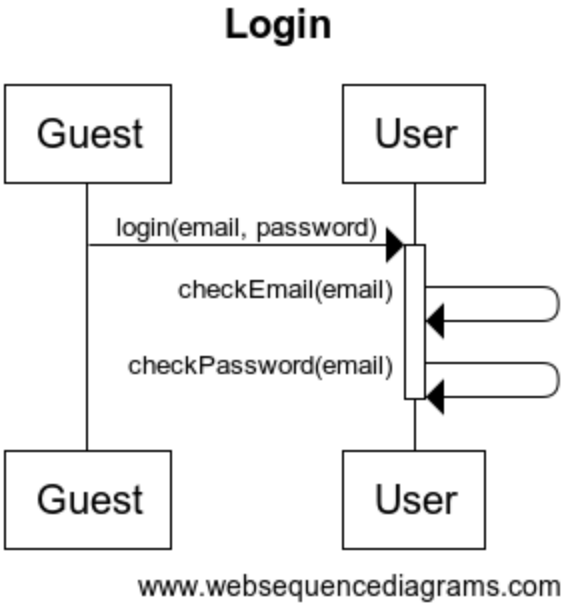
User Service Class Diagram



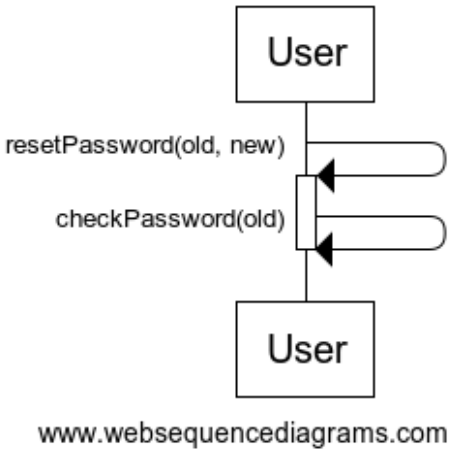
Training Service Class Diagram



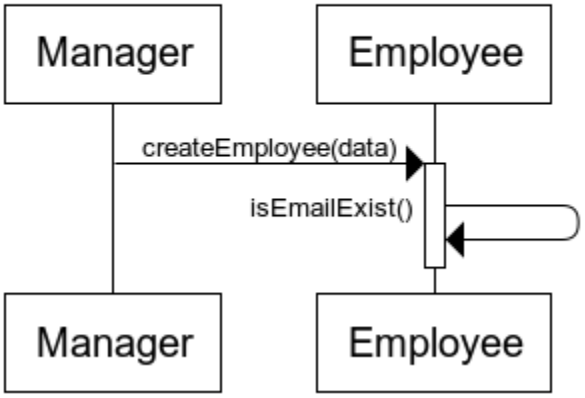
3.2 Sequence Diagrams



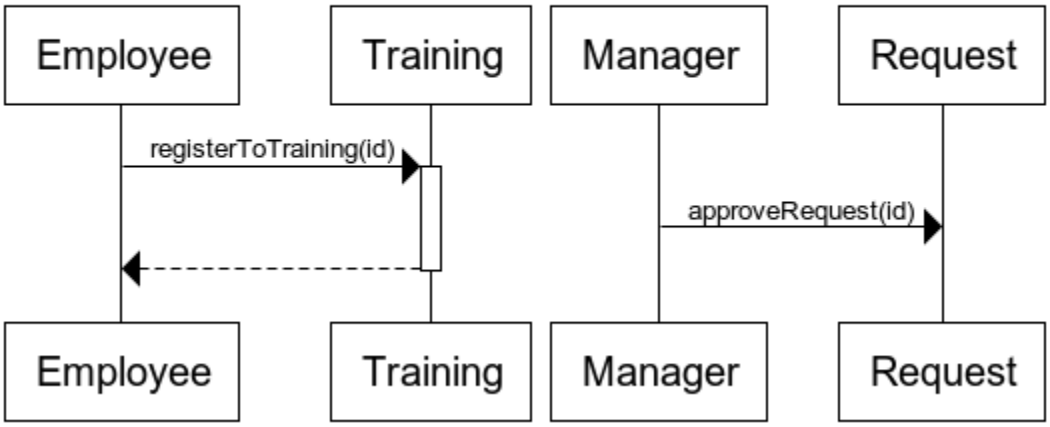
Reset Password



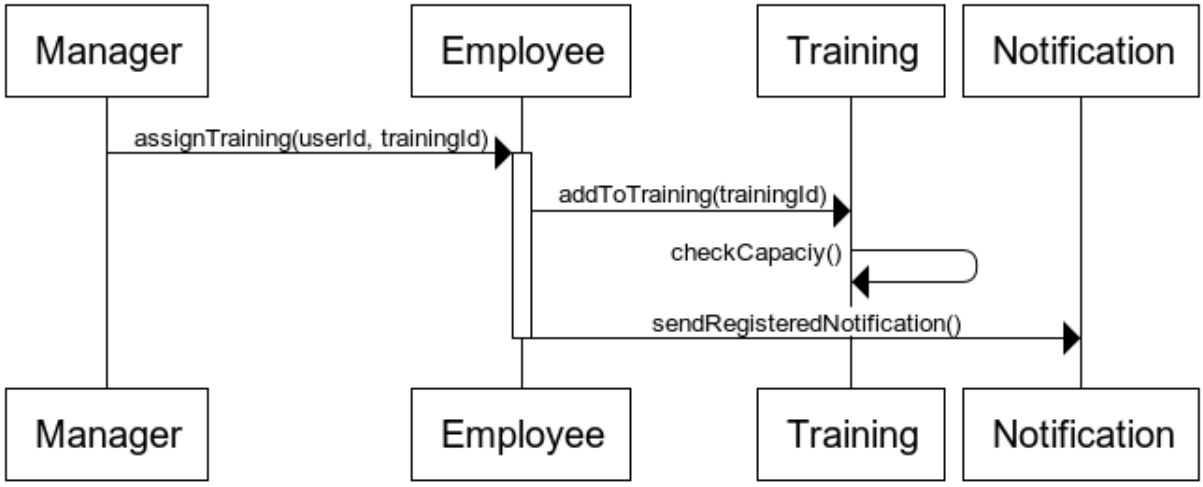
Create Employee

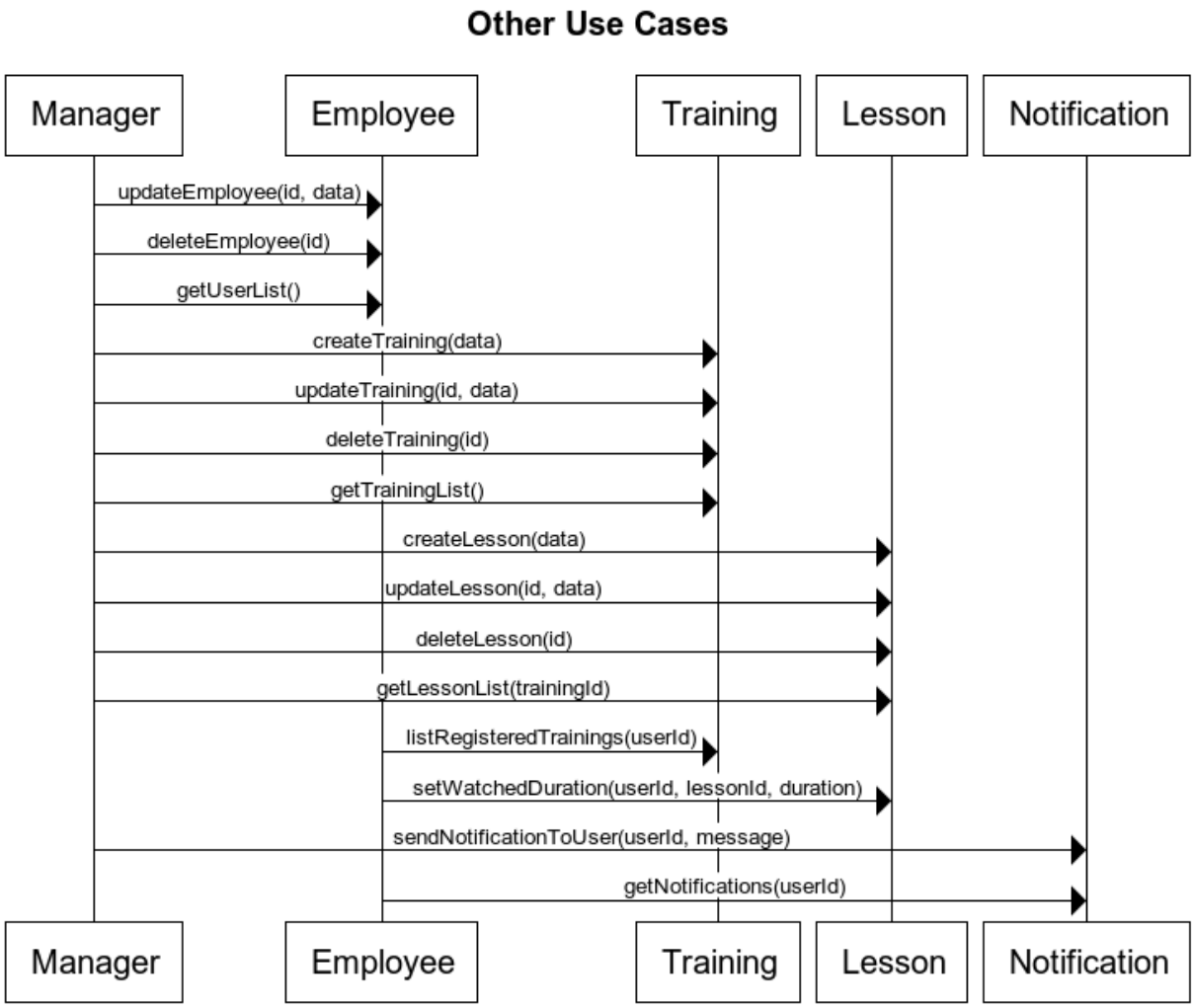


Register to Training



Assign Training

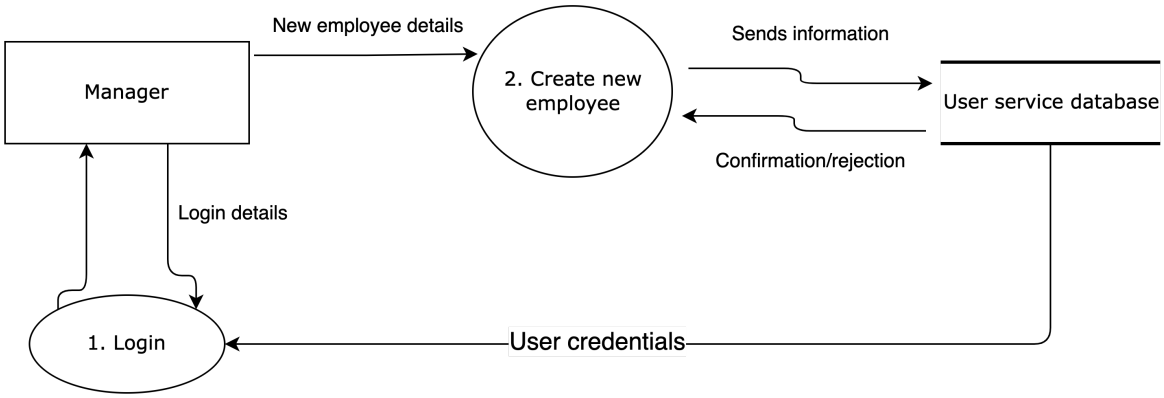




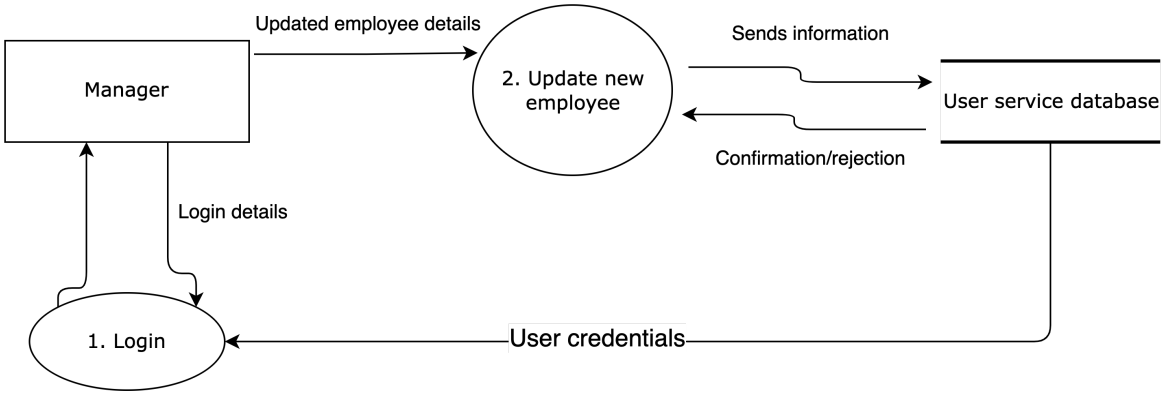
These use cases are independent from each other. All of them requires single operation; therefore, we merged the diagrams.

3.3 Data Flow Diagram

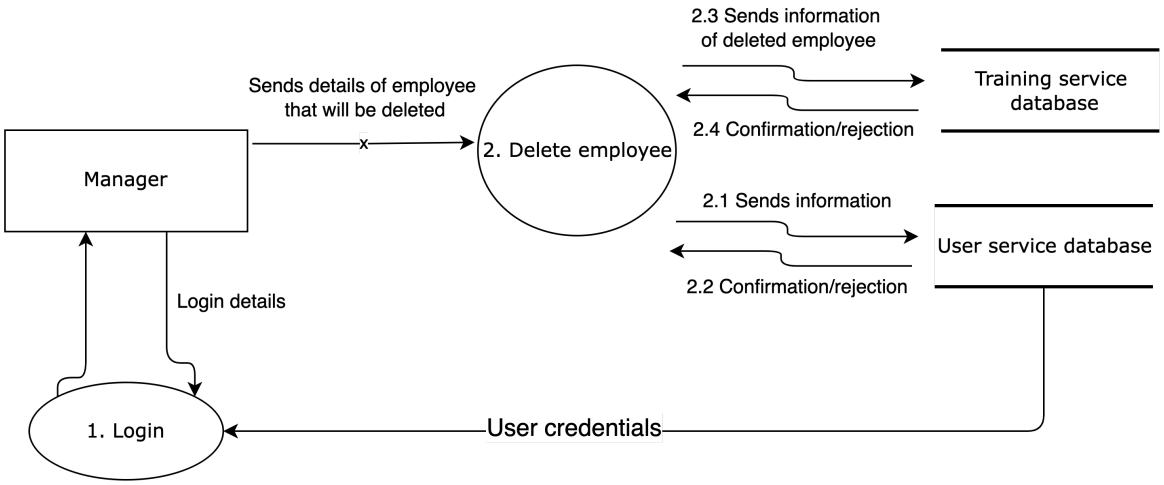
▼ Manager creates new employee



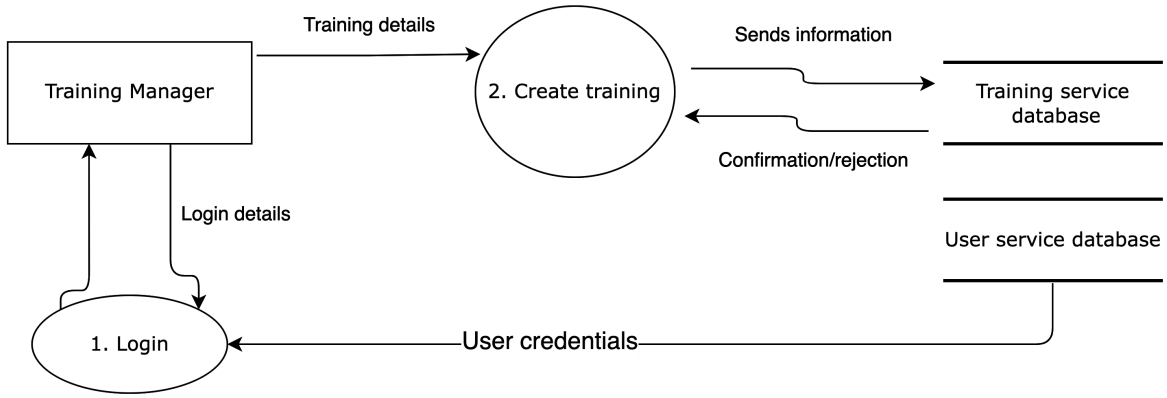
▼ Manager updates employee



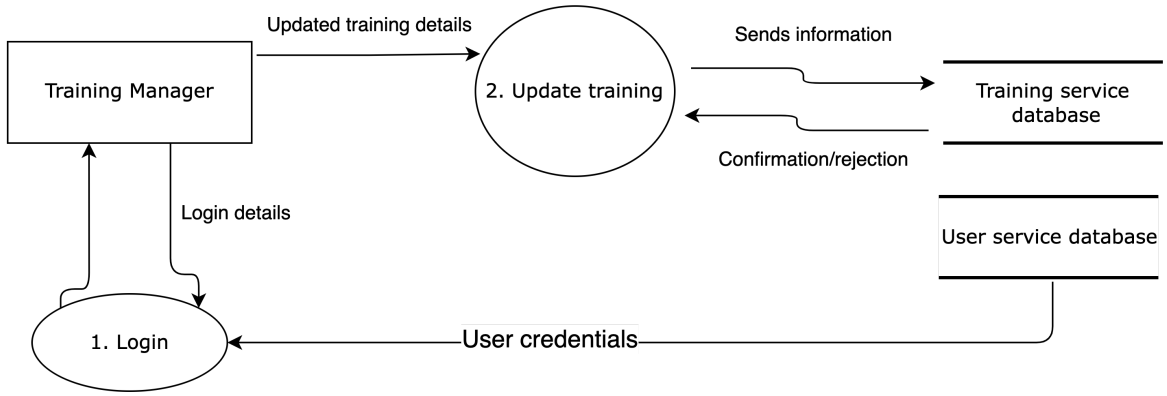
▼ Manager deletes employee



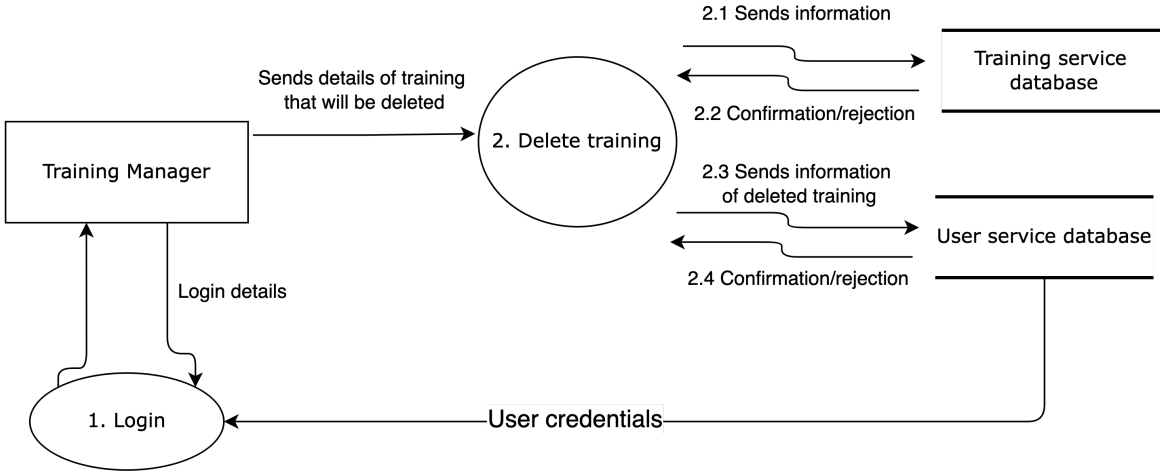
▼ Training Manager creates new training



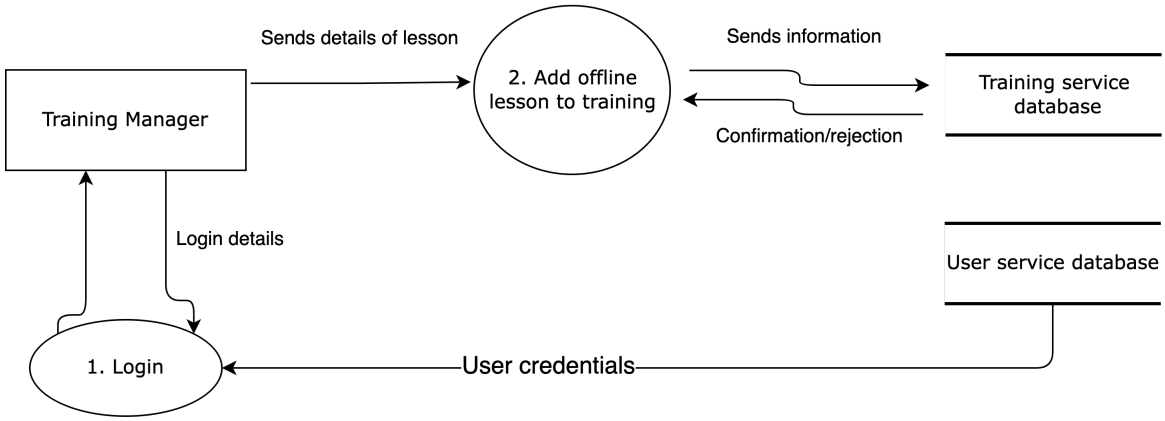
▼ Training Manager updates training



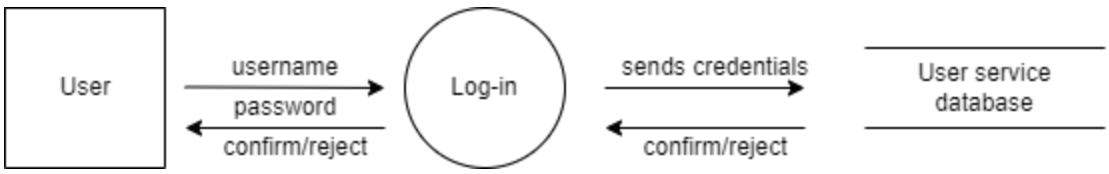
▼ Training Manager deletes training



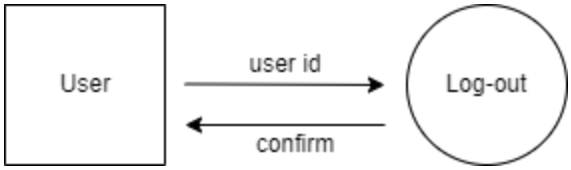
▼ Training Manager adds offline lesson to training



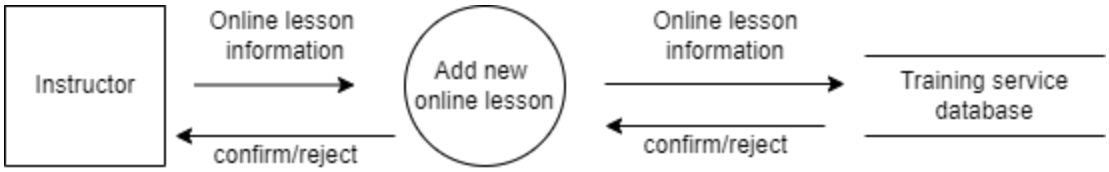
▼ User logs-in



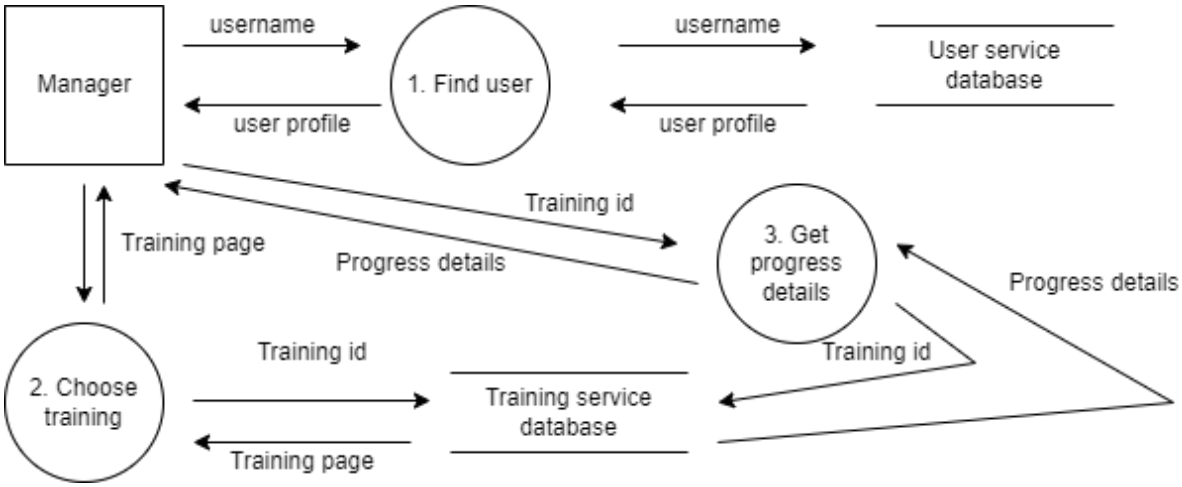
▼ User logs-out



▼ Instructor adds a new online lesson into a training

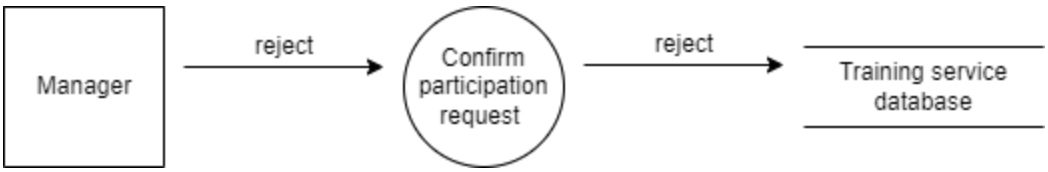


▼ Manager gets a specific user's progress report about a specific training



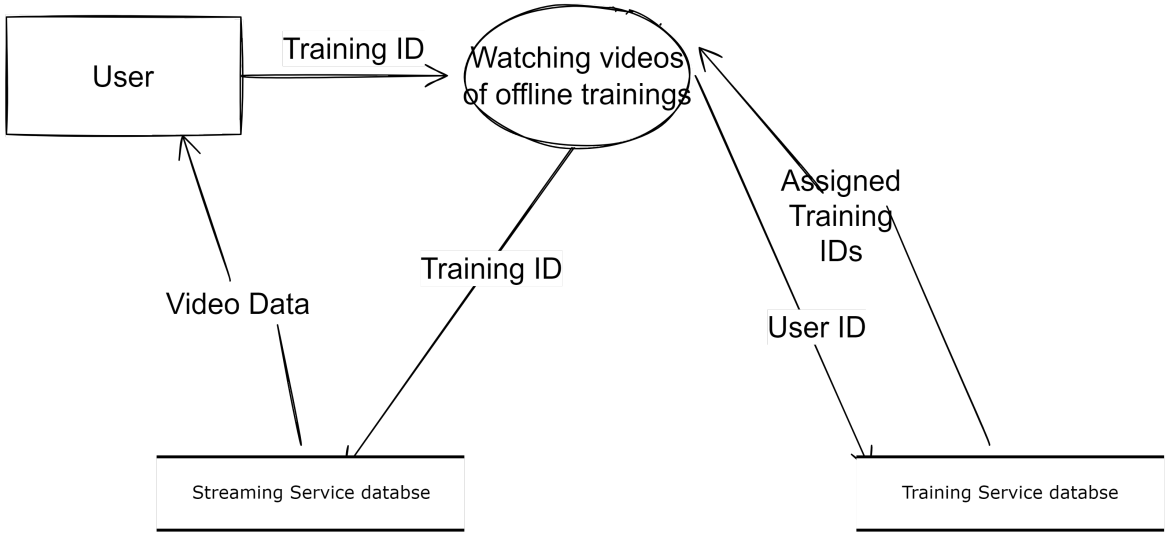
▼ Manager confirms a user's participation request to a training

▼ Manager rejects a user's participation request to a training

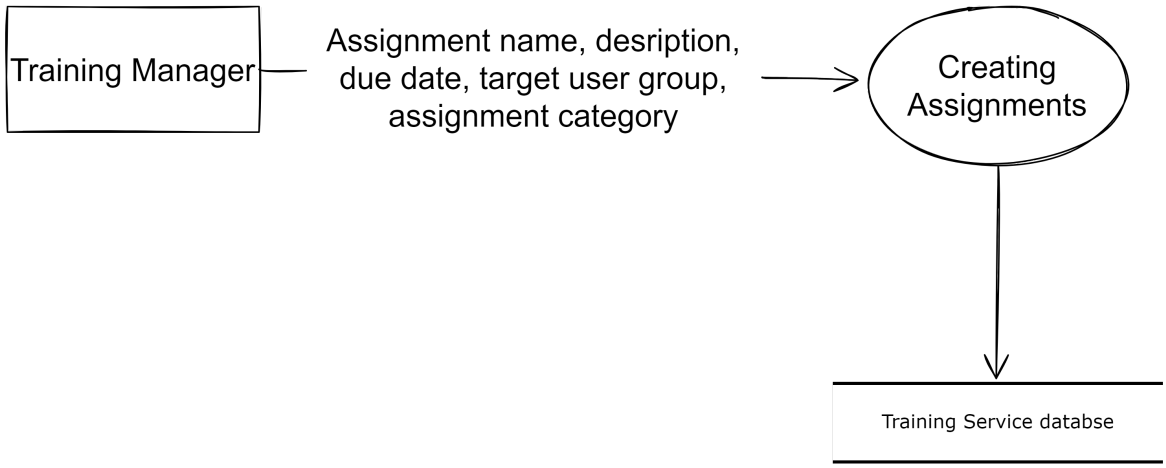


▼ Instructor gets the list of participants

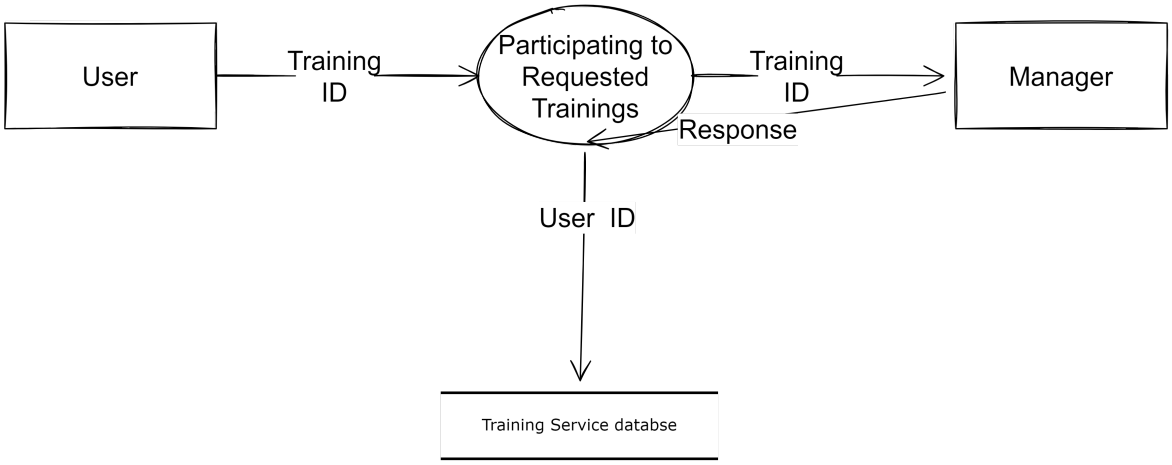
▼ User watches videos of offline trainings



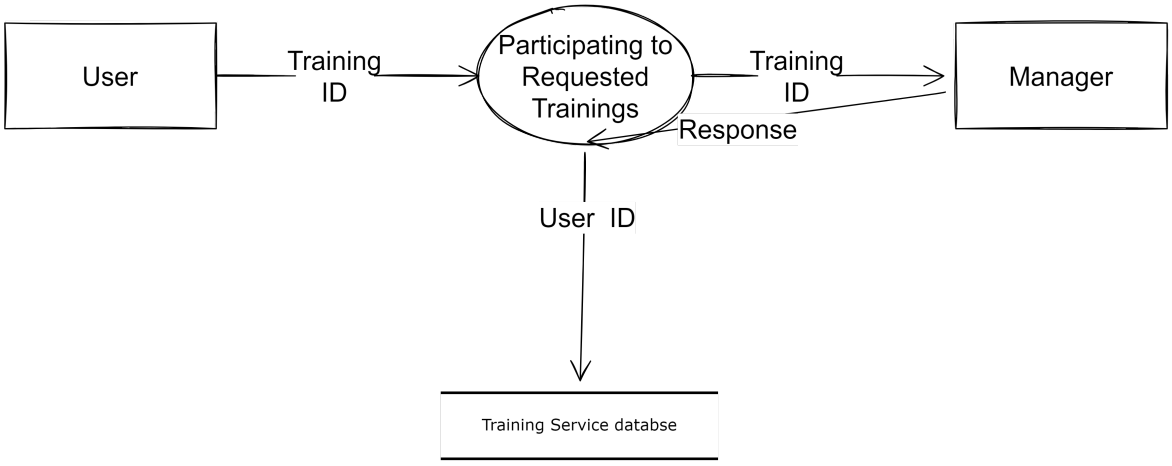
▼ Training Manager creating assignments



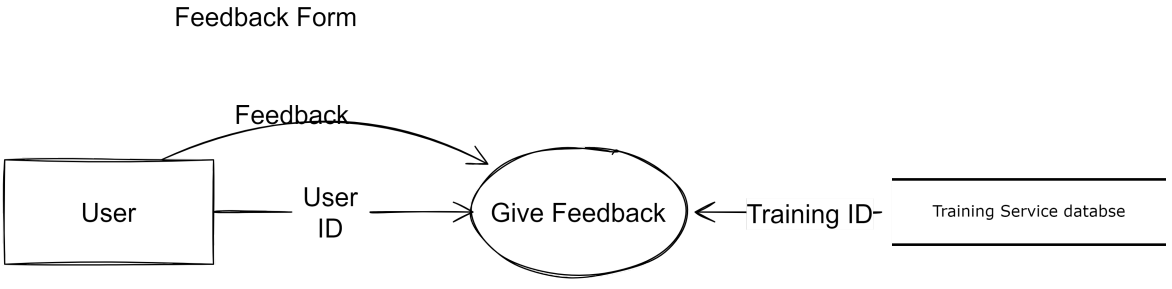
▼ User participating to requested trainings



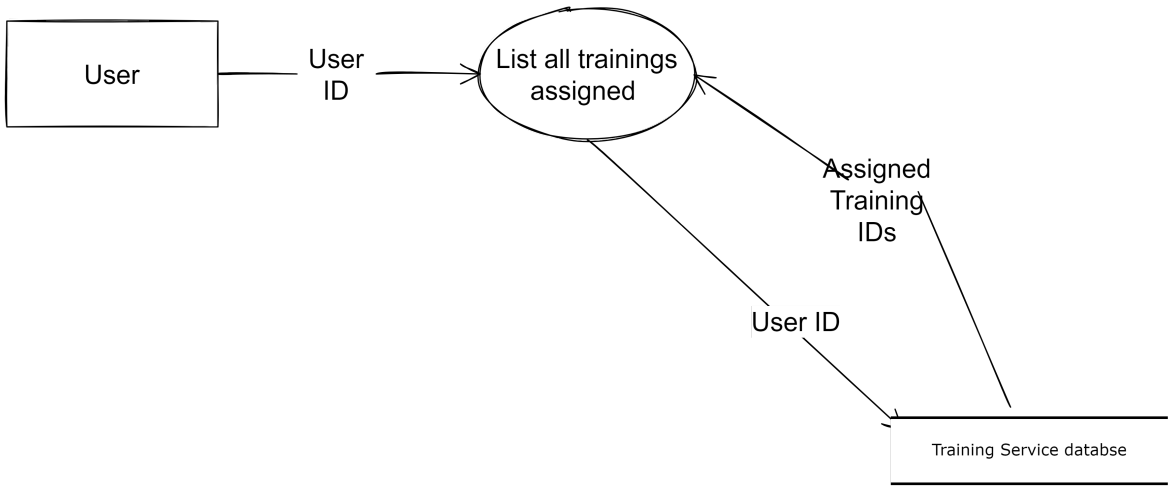
▼ User viewing notifications



▼ User giving feedback for a completed training



▼ User listing all trainings assigned to it



▼ User updating password