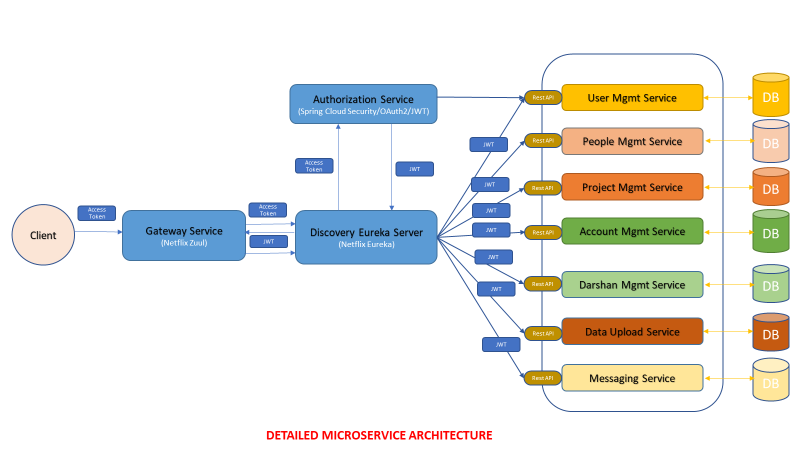
**BJJD**

# Project Architecture



## Gateway Service

## Authentication Service

## Eureka Server

Eureka Server or service will register every microservice and then the client microservice will look up the Eureka server to get a dependent microservice to get the job done without knowing the IP address of dependent microservice.

Eureka Server is owned by Netflix and in this, Spring Cloud offers a declarative way to register and invoke services by Java annotation.

It’s duty to give names to each microservice. Why?

* No need to hardcode the IP addresses of microservices.
* What if services use dynamic IP addresses; when autoscaling.

So, every service register itself with Eureka, and pings Eureka server to notify that it’s alive.

## Microservices

We will discuss all the following microservices in details:

* User Management Service
* People Management Service
* Darshan Management Service
* Project Management Service
* Account Management Service
* Data Upload Service
* Messaging Service

# Use Case: User Management Service

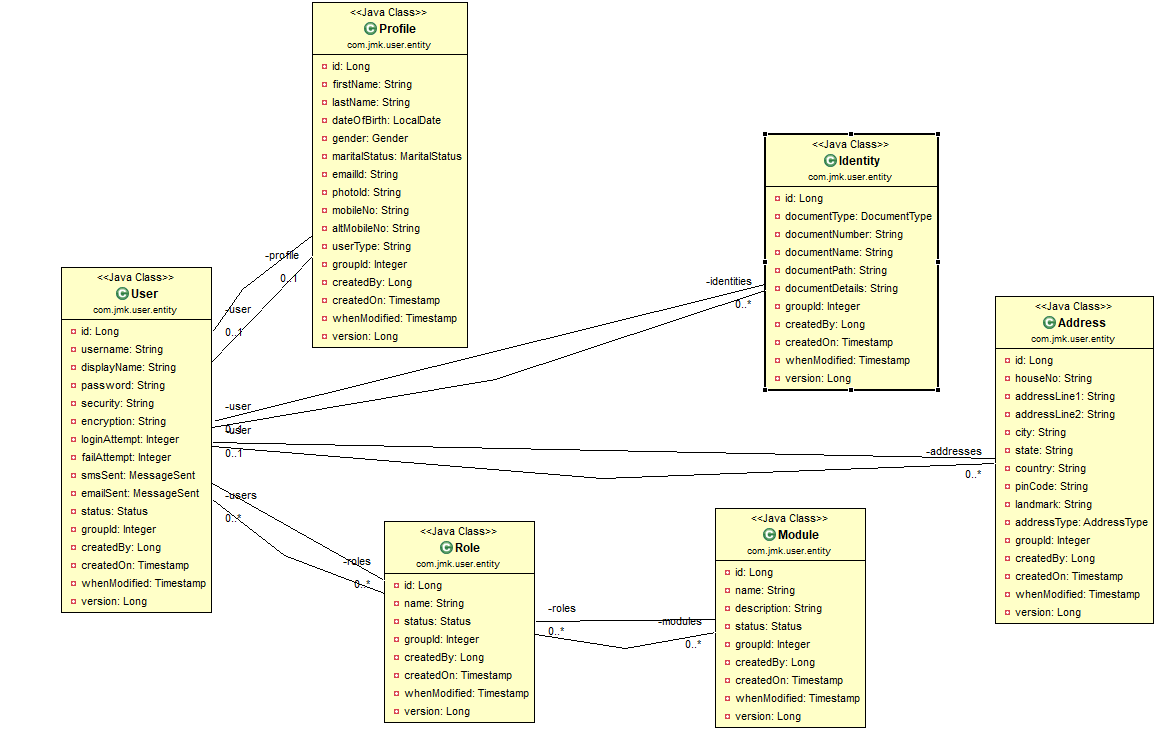
## Description

User Management Service is developed to manage the user. It will have the following functions.

* Managing User Profile
* Managing Roles
* Managing Modules

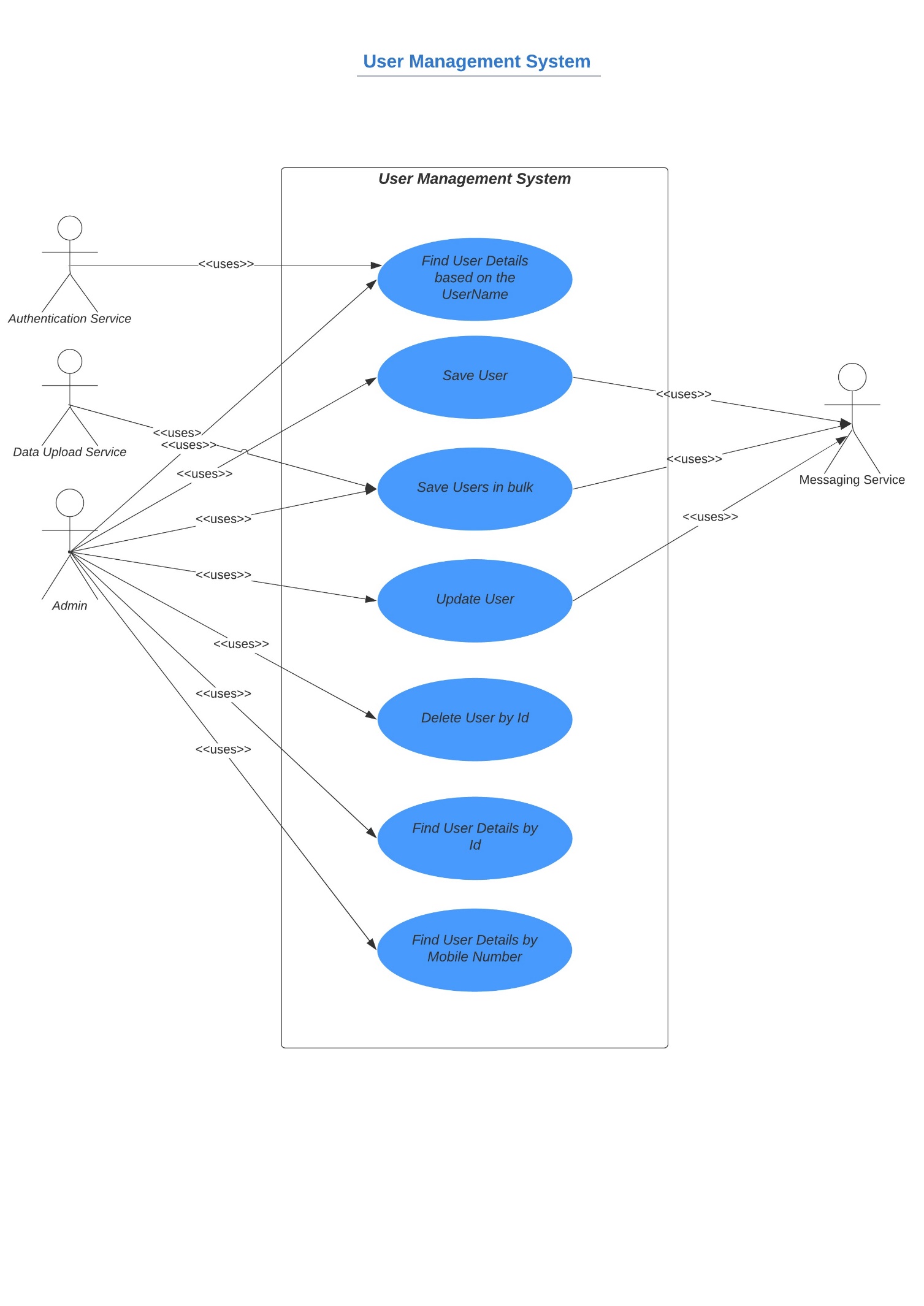
## Entity Relationship Diagram

## Class Diagram



## Use Case Diagram

A use case diagram is a representation of user’s interaction with the system that shows the relationship between the user and the different use cases in which user is involved.



## Use Case Details

# Use Case: People Management Service

## Description

## Entity Relationship Diagram

## Class Diagram

## Use Case Diagram

## Use Case Details

# Use Case: Darshan Management Service

## Description

## Entity Relationship Diagram

## Class Diagram

## Use Case Diagram

## Use Case Details

# Use Case: Account Management Service

## Description

An Account Management Service is developed as microservice with the following functionalities:

* Managing Donations
* Managing Expenses

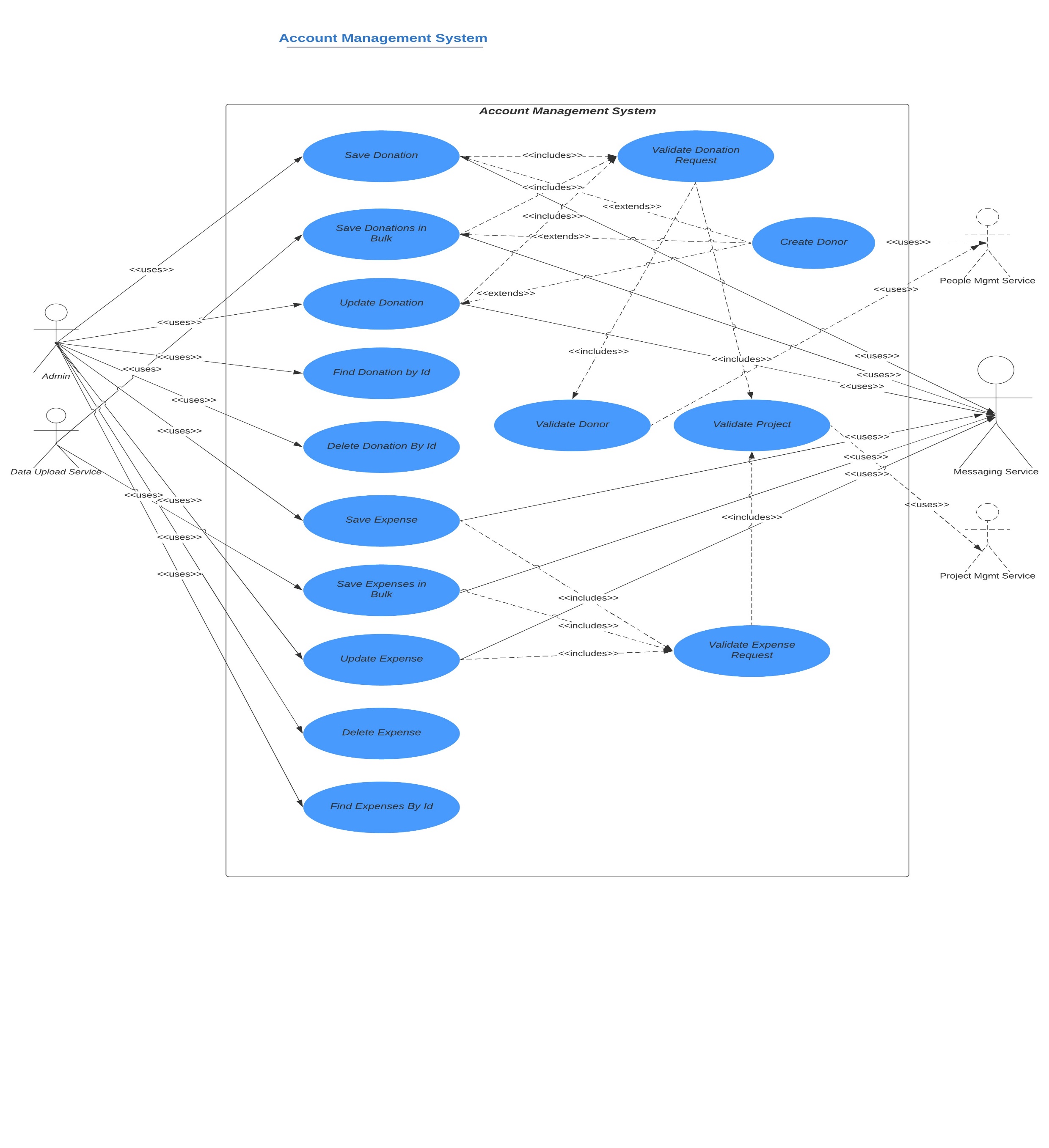
## Entity Relationship Diagram

## Class Diagram



## Use Case Diagram

A use case diagram is a representation of user’s interaction with the system that shows the relationship between the user and the different use cases in which user is involved.



## Microservices Communication

Synch REST Request

Synch REST Request

Account Management System

Synch REST Request

Asynch REST Request

Asynch Kafka/RabbitMQ Request

## Use Case Details

### Manage the donations

#### Saving the donation

* Donation Request must be valid.
* If donation Request is valid
* Validate the donor if either of the following conditions met:
  + donor id and donor type are provided in the request and the donor should exists in the system. To check the donor existence, it will interact with **microservice**: people-mgmt-service.
  + If donor is Devotee and its id does not exist in the request.
* Validate the project exists. To check the project existence, it will interact with **microservice**: project-mgmt-service.
* Else If Donation Request Not Valid
* Error message should be triggered for the user.
* Creating the donor if required. If donor is devotee and its id does not exist it means Devotee is new and we need to create the devotee by interacting **microservice**: people-mgmt-service.
* Save the donation and return the saved Donation result.

#### Saving the donations in bulk

* Donation Request must be valid.
* If donation Request is valid
* Validate the donor if either of the following conditions met:
  + donor id and donor type are provided in the request and the donor should exists in the system. To check the donor existence, it will interact with **microservice**: people-mgmt-service.
  + If donor is Devotee and its id does not exist in the request.
* Validate the project exists. To check the project existence, it will interact with **microservice**: project-mgmt-service.
* Else If Donation Request Not Valid
* Error message should be triggered for the user.
* Creating the donor if required. If donor is devotee and its id does not exist it means Devotee is new and we need to create the devotee by interacting **microservice**: people-mgmt-service.
* Save the donation and return the saved Donation result.

#### Update the donation

* Donation Request must be valid.
* If donation Request is valid
* Validate the donor existence:
  + donor id and donor type are provided in the request and the donor should exists in the system. To check the donor existence, it will interact with **microservice**: people-mgmt-service.
* Validate the project exists. To check the project existence, it will interact with **microservice**: project-mgmt-service.
* Else If Donation Request Not Valid
* Error message should be triggered for the user.
* Update the donation request and return the updated Donation result.

#### Find the donation results based on the Id

#### Delete the donation based on the Id

### Manage the expenses

#### Saving the expense

* Donation Request must be valid.
* If donation Request is valid
* Validate the donor if either of the following conditions met:
  + donor id and donor type are provided in the request and the donor should exists in the system. To check the donor existence, it will interact with **microservice**: people-mgmt-service.
  + If donor is Devotee and its id does not exist in the request.
* Validate the project exists. To check the project existence, it will interact with **microservice**: project-mgmt-service.
* Else If Donation Request Not Valid
* Error message should be triggered for the user.
* Creating the donor if required. If donor is devotee and its id does not exist it means Devotee is new and we need to create the devotee by interacting **microservice**: people-mgmt-service.
* Save the donation and return the saved Donation result.

#### Saving the expenses in bulk

* Donation Request must be valid.
* If donation Request is valid
* Validate the donor if either of the following conditions met:
  + donor id and donor type are provided in the request and the donor should exists in the system. To check the donor existence, it will interact with **microservice**: people-mgmt-service.
  + If donor is Devotee and its id does not exist in the request.
* Validate the project exists. To check the project existence, it will interact with **microservice**: project-mgmt-service.
* Else If Donation Request Not Valid
* Error message should be triggered for the user.
* Creating the donor if required. If donor is devotee and its id does not exist it means Devotee is new and we need to create the devotee by interacting **microservice**: people-mgmt-service.
* Save the donation and return the saved Donation result.

#### Update the expense

* Donation Request must be valid.
* If donation Request is valid
* Validate the donor existence:
  + donor id and donor type are provided in the request and the donor should exists in the system. To check the donor existence, it will interact with **microservice**: people-mgmt-service.
* Validate the project exists. To check the project existence, it will interact with **microservice**: project-mgmt-service.
* Else If Donation Request Not Valid
* Error message should be triggered for the user.
* Update the donation request and return the updated Donation result.

#### Find the expense results based on the Id

#### Delete the expense based on the Id

# Use Case: Data Upload Service

## Description

Data

## Entity Relationship Diagram

## Class Diagram

## Use Case Diagram

## Use Case Details

# Use Case: Messaging Service

## Description

## Entity Relationship Diagram

## Class Diagram

## Use Case Diagram

## Use Case Details