**NoSQL Project evaluation document**

**Group: Nº4**

**Group name: IT2-B Group-4**

**Student names:**

* **Md Tasnim Kabir (658116)**
* **Jason Xie (659045)**
* **Jonathan Mauricio (677762)**
* **Andrei Schereper (678786)**
* **Manol Draganov (678785)**

Table of contents

[The login for the application 3](#_Toc119102829)

[Employee login 3](#_Toc119102830)

[ServiceDesk Employee login 3](#_Toc119102831)

[General evaluation/Application introduction: 3](#_Toc119102832)

[Rubric A 3](#_Toc119102833)

[Rubric B 3](#_Toc119102834)

[Rubric C 3](#_Toc119102835)

[Rubric D 3](#_Toc119102836)

[Rubric E 3](#_Toc119102837)

[Rubric F 3](#_Toc119102838)

[Rubric G 3](#_Toc119102839)

[Rubric H 4](#_Toc119102840)

[Rubric I 4](#_Toc119102841)

[Rubric J 4](#_Toc119102842)

[Student evaluations 5](#_Toc119102843)

[Md Tasnim Kabir 5](#_Toc119102844)

[Jason Xie 6](#_Toc119102845)

[Jonathan Mauricio 7](#_Toc119102846)

[Andrei Schereper 8](#_Toc119102847)

[Manol Draganov 9](#_Toc119102848)

[Appendix 10](#_Toc119102849)

[ERD: (Entity Relationship Diagram) 10](#_Toc119102850)

[EER: (Enhanced Entity Relationship): 10](#_Toc119102851)

[UML use case diagrams (Individual based on functionalities) 11](#_Toc119102852)

[Md Tasnim Kabir 11](#_Toc119102853)

[Jason Xie 12](#_Toc119102854)

[Jonathan Mauricio 13](#_Toc119102855)

[Andrei Schereper 14](#_Toc119102856)

[Manol Draganov 15](#_Toc119102857)

# The login for the application

## Employee login

Email: [JohnSmith@gmail.com](mailto:JasonSmith@gmail.com)

Password: 1234

## ServiceDesk Employee login

Email: [Jasonxie@gmail.com](mailto:Jasonxie@gmail.com)

Password: 1234

# General evaluation/Application introduction:

## Rubric A

The application uses a MongoDB database to store all our data.

## Rubric B

The application is fully written in C#.

## Rubric C

Common design patterns are used, specifically the MVC pattern and other useful patterns learned in the design interactions course.

## Rubric D

The application has a login portal that will authenticate the user as either an Employee (end-user) or as a ServiceDesk Employee with different permissions for both.

## Rubric E

Error 404

## Rubric F

The application follows the design provided by Garden Group as much as possible and wherever possible.

## Rubric G

The application design is based on a more modern theme with readable colors while trying to stay true to the design provided by Garden Group.

## Rubric H

ServiceDesk Employees can perform CRUD operations on all tickets while regular Employees can create their own tickets.

## Rubric I

The dashboard shows the current ticket status of all tickets when the user logged in is a ServiceDesk Employee and only shows the user’s tickets when the user is a regular Employee.

## Rubric J

The database consists of 3 collections (One for tickets, one for users, and one for archived tickets) With the two main collections of tickets and users containing 100 documents of each of the respective types.

# Student evaluations

## Md Tasnim Kabir

For this project, I have chosen incident management as a group work and for individual functionality I have implemented “searching through incident tickets”.

**Rubric K**

I worked on the incident management part where I had to display all the tickets in the data grid view and had to implement the update and delete functionalities of the tickets. Moreover, I also did the search functionality where you can search by subject, tickets and “and + or” search by priority and status of a ticket. To implement these functionalities, I have used multiple classes and they are Ticket\_Model, TicketDAL, Ticket\_Service and NoDesk UI and every time I have finished my work I push to my branch and merge with main.

**Rubric L**

Without my part nothing would have worked. Because, if I would not display all the tickets in the data grid view then it wouldn’t be possible to implement the delete, update, create, searching, filtering and dashboard functionalities.

**Rubric M**

When I was developing my functionalities, I had to switch my decisions a few times after discussions with my teammates. For example, first I displayed all the tickets in a table view. But due to some complexity of implementing other functionalities, I had to switch to a data grid view.

After that, I was confused about how to implement the search functionalities. I did some research on it and watched some videos on Moodle recommended by my teacher. Ultimately, I did that part by using the link and also for taking all the recent tickets I used a query.

**Rubric N**

Before making our database, we made a basic ERD diagram to get a general perception of our database and gradually we have changed our ERD during the development process. We also made an EER (class diagram ) to get a clear idea of our code. Then, I also make a UML use case diagram to capture the requirements of the system so that I get a clear idea of what I need to build.

**Rubric O**

For the extra functionality, I have chosen “searching through tickets”. It was really fun to make. You can search a ticket by its subject, ticket number and also you can do “and +or” search by priority and status. Moreover, you can also take the most recent tickets on top by clicking on the reset filter button.

## Jason Xie

For the project I made it so the users can login as Employee or Service Desk Employee. For the extra functionality I have filtering which filters the tickets by common keywords.

**Rubric K**

I was responsible for Logging in the user as an Employee or Service Desk Employee, Hashing the password and filtering the tickets. When logging in as a Employee, the Employee can only view the DashBoard and the TicketOverview with only search, filtering and add ticket functionality. And if you login as a Service Desk Employee you have all the functionalities.

**Rubric L**

Without the login part of the application users would not be able to login to the application. For hashing password, it is to ensure that the password is not stored in plain text in the database. And with filtering it is all about convenience, a company may have many tickets and filtering it will make it much easier to find the ticket you were looking for

**Rubric M**

At the start we did not make use of which type of user the employee is so all users could just view and modify all tickets but later it was changed so that on login the type of user is checked, and the correct functionalities are assigned. With hashing, at first it was not implemented but with discussion with fellow team members about the passwords being visible in plain text in the database we came to the conclusion that it would be necessary to implement it.

**Rubric N**

In the ERD we have a user entity that can be a normal employee or a service desk employee because the service desk employee has more functionalities than a normal employee. On the ERD it indicated that only the Service Desk Employees can make tickets but that is not true and has since been updated. I made use of Use Cases to get a better perception on how the login part would interact with the other parts of the application. With the class diagram we were able to implement methods that we know we need during the development stage.

**Rubric O**

For the extra functionality I chose filtering which filters the ticket by common keywords such as status, priority, deadline, and type.

## Jonathan Mauricio

My part of the project consisted of creating the User Management panel as well as implementing the close tickets and transferring tickets (individual) methods. Most of my work was done via pop-up forms and some basic data grids and buttons in the main application.

**Rubric K**

I was responsible for designing and implementing User Management, Closing Tickets, Transferring Tickets, and user generation. On top of this I also worked closely with my team members to help them with any problems they may have and have contributions in other’s parts. A lot of my code consisted of input validation and working directly with the database to insert documents and update documents.

**Rubric L**

I was responsible for a lot of critical functions of the application and without them the application wouldn’t be able to properly function. I was responsible for the original creation of the database and document templates. Without my part the ServiceDesk Employees would not be able to manage any users or add new employees so they could have the ability to create tickets. Without my part the ServiceDesk Employees also couldn’t close tickets, thus not being able to mark them as complete. I was also responsible for user generation, writing methods to quickly generate a number of users (in his case 100).

**Rubric M**

During the development process I had to make a lot of choices in how I would want to implement my specific parts. I had to make decisions on how I would validate user creation to ensure invalid users could not be created. And due to their not being an email system setup I originally had a random digit password generated for each user created randomly, which would then be shown on screen.

After some feedback from my group members and debating I decided to instead change this to a simple password input field for ease of use. When it came to generating 100 users for the database, I opted to use the random password generator to try and simulate how it would work in a real-world environment. When I worked on closing tickets, I had to decide on how I would handle closed tickets differently from opened tickets in the incident management page. This resulted in the checking of tickets and whether they were open and closed and disabling/enabling certain buttons in the UI to reflect this.

**Rubric N**

During the design process a basic ERD was also used for employees that I created alongside my group members. This listed all the properties of Employees to make for an easy time implementing it into MongoDB. I also made use of a UML diagram to see all the functionalities I had to develop. (Shown below)

**Rubric O**

Transferring tickets was interesting in how I wanted to handle a big list of users. After I generated 100 users, I noticed that a long, unorganized list was extremely difficult to look through and not very user friendly. So, I decided to sort that list alphabetically to make it easier to browse while removing the user from the list who the ticket was already assigned to since you shouldn’t be able to transfer a ticket to the same person.

## Andrei Schereper

For this project I have done the dashboard and gave an employee the ability to see his/her own tickets. For the additional functionality I have implemented the sorting by Priority, either ascending or descending through all the priority types (low, normal, high). I have also made a big impact on the project’s design.

**Rubric K**

The dashboard gives an employee a good overview of the tickets made by himself/herself. Two circular progress bars are used to show how many tickets are unfinished and how many tickets have passed the deadline. A normal employee sees a dashboard tailored to his/her tickets.  A service desk employee has an overview of all tickets. In the incident management I have also added the ability for a normal employee to see his/her tickets with all the details about it in a table view. There the employee can sort it by Priority which is my additional functionality. The normal employee or the service desk employee can choose to sort ascending from Low to High or descending from High to Low.

**Rubric L**

Without my addition to this project, the application would not have a graphical interface of the tickets within it. Also, without my part, a normal employee could not even view his/her own tickets. The design is easy on the eye and helps the user navigate easily within the app.

**Rubric M**

One of the first choices that our team had to take was about either to use panels or just forms. Taking in account the memory usage we have decided that for the 3 main pages of the app we should use panels and for the creation of users/tickets or login page we used separate forms as this way seemed the most natural out of all of them. Another major decision was about the login part. We have decided to use the email instead of the username or full name because of it was unique for each user. Small decisions were always discussed in our group chat or in person.

**Rubric N**

The ERD was used so we can have a clear view about how we will approach the development of this app. The User can either be an Employee or a Service Desk employee. The employee reports tickets to the service desk and the service desk creates the ticket. This is how we designed the ERD but later we enabled the employee to create tickets because we gained a better understanding of how the application should work. The EER contains a lot of information about our application

**Rubric O**

As previously stated, the additional functionality that I have added is the ability of the user to sort by priority. I have chosen to let the user sort the table either ascending or descending so they can have a clear and logical view of their tickets. The functionality is represented by a combo-box with 2 options and a button that sorts depending on what is chosen in the combo-box. The functionality is located within the incident management panel of the project.

## Manol Draganov

My part in the implementation of the project consists of creating the incident ticket generation functionality for all types of employees (regular and service desk) as well as implementing an additional functionality for archiving tickets based on date criteria. I have also contributed for finding suitable design patterns for the generic classes and implementing design that would reduce the code duplication.

**Rubric K**

The “Add ticket” functionality gives the ability of all users do create and store a ticket when an incident within The Group Garden occurs. Employees can enter an incident indicating its type, priority level, date of occurrence, and desired deadline of fixing. Employees has the right to choose a past date or current(today) for the incident, since the incident could have occured in the past, but it had not been registered. Every ticket must also have a subject, explaining in general the incident. Employees can input a description, explaining more in-depth the situation, but of course, this is optional. Regular employees can generate a ticket from their name, but service desk employees can create a ticket from the name of everybody, as wished in the requirements.

**Rubric L**

One of the base concepts of this application is for employees to create tickets for incidents they have experienced. Without my implementation on the generating ticket functionality, employees could not register incidents which makes the abilities of The Garden Group’s incident management simply useless. Indicents could not be registered, looked up and potentially solved.

**Rubric M**

During the development process, I faced a challenge with filtering tickets by date (on the archiving functionality) because our Ticket model was storing its date as a string in a custom format. Luckily, we switched to storing date as a DateTime type, but if we had not, I would’ve had to aggregate a collection using the $dateFromString expression, in order to extract the needed records. That would not have been the best technique and would have caused some problems in the future.

**Rubric N**

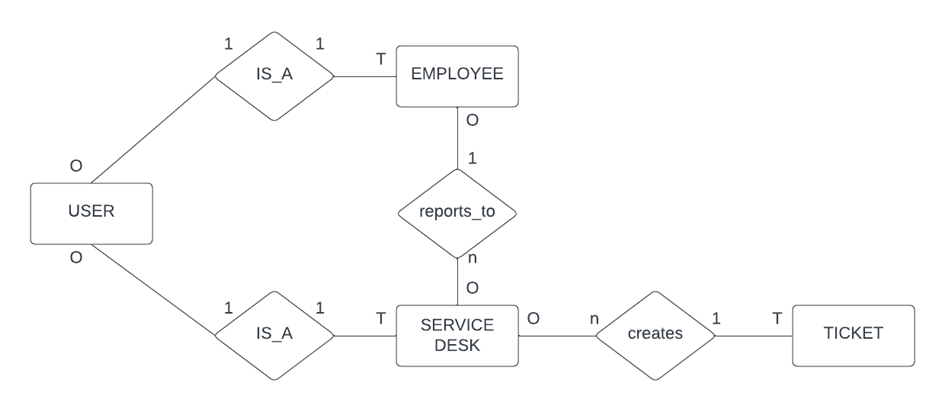
ERD has been used as an initial schema to understand how our database should be structured. EER has been generated and included to provide information on our class structures. Personal UMLs (for both functionalities implemented by me) are included in the appendix.

**Rubric O**

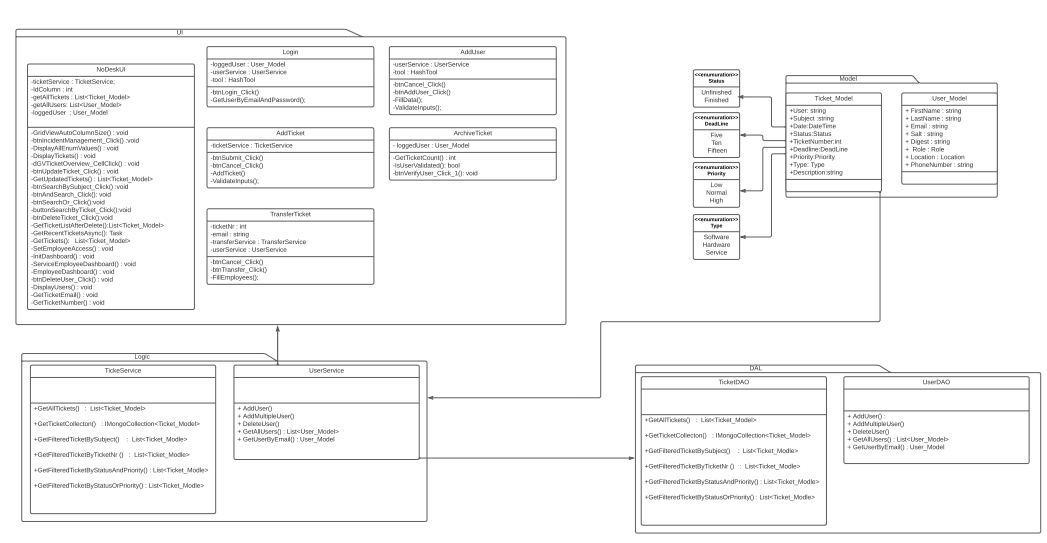
As an additional functionality, service desk employees(only) can archive tickets older then a chosen from them date. After a verification message has been confirmed (a Design To Forgive principle), the employee has to authenticate themselves by confirming their password. The user can see how many tickets will be archived after selecting the date. When archiving happens, tickets that meet the user’ criteria are found, removed from the active database, and put into an archive database, inaccessible for CRUD operations.

# Appendix

## ERD: (Entity Relationship Diagram)

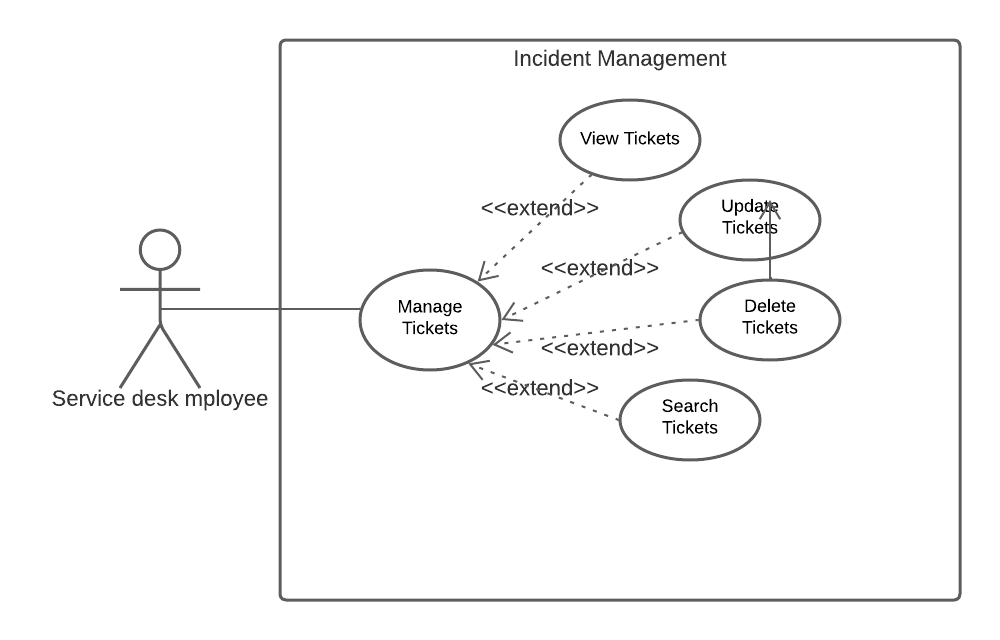
****

## EER: (Enhanced Entity Relationship):

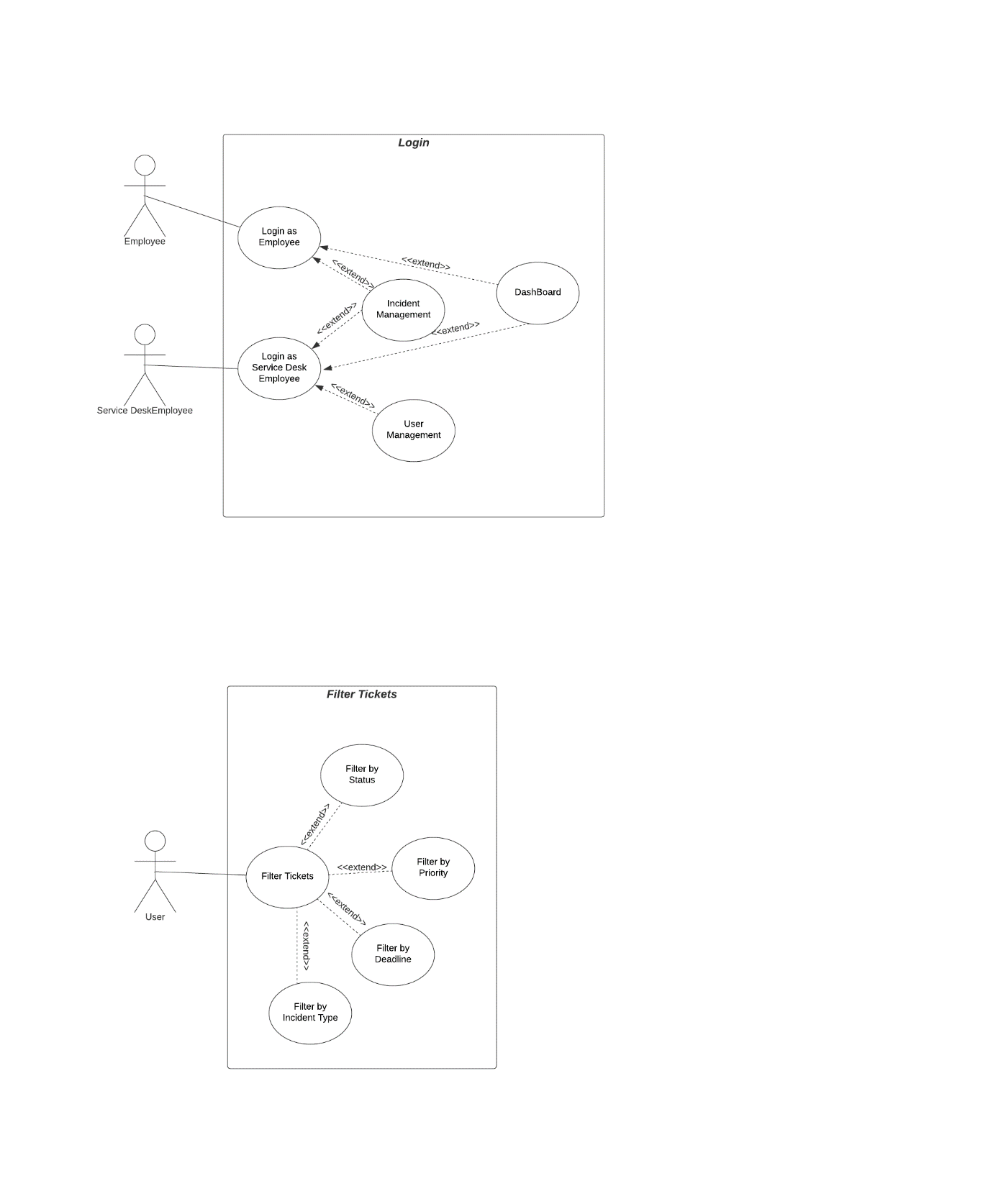


## UML use case diagrams (Individual based on functionalities)

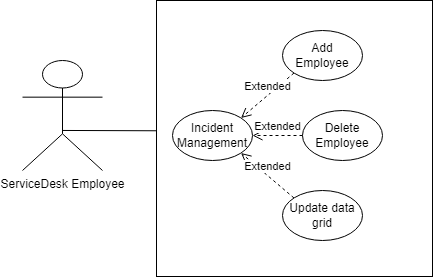
### Md Tasnim Kabir

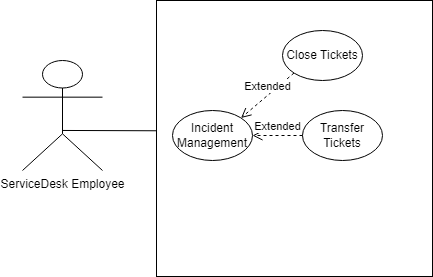


### Jason Xie

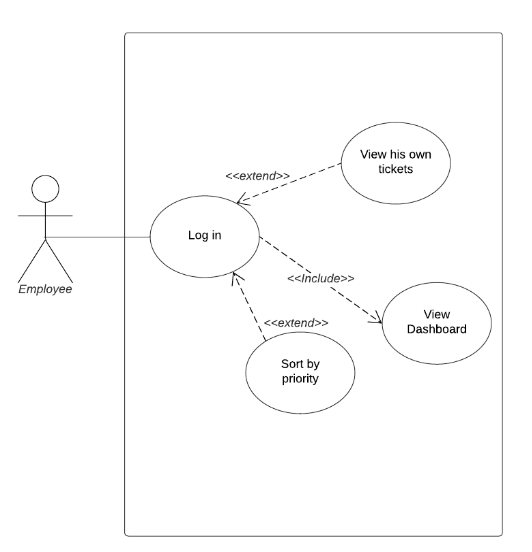


### Jonathan Mauricio





### Andrei Schereper



### Manol Draganov

