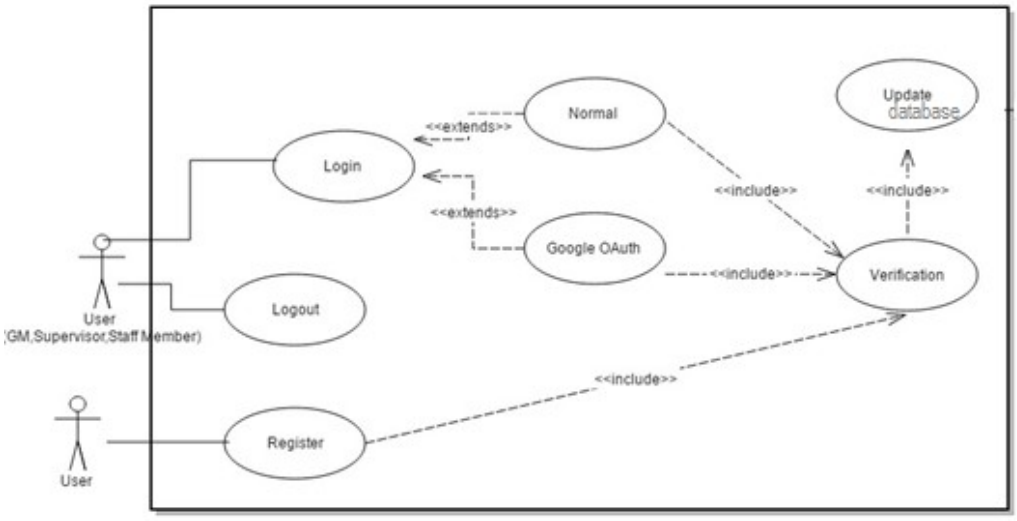
Project Report

Facility Management Services (FMS) System

Based on the requirements we have come up with the following design of the system through use case and class diagrams.

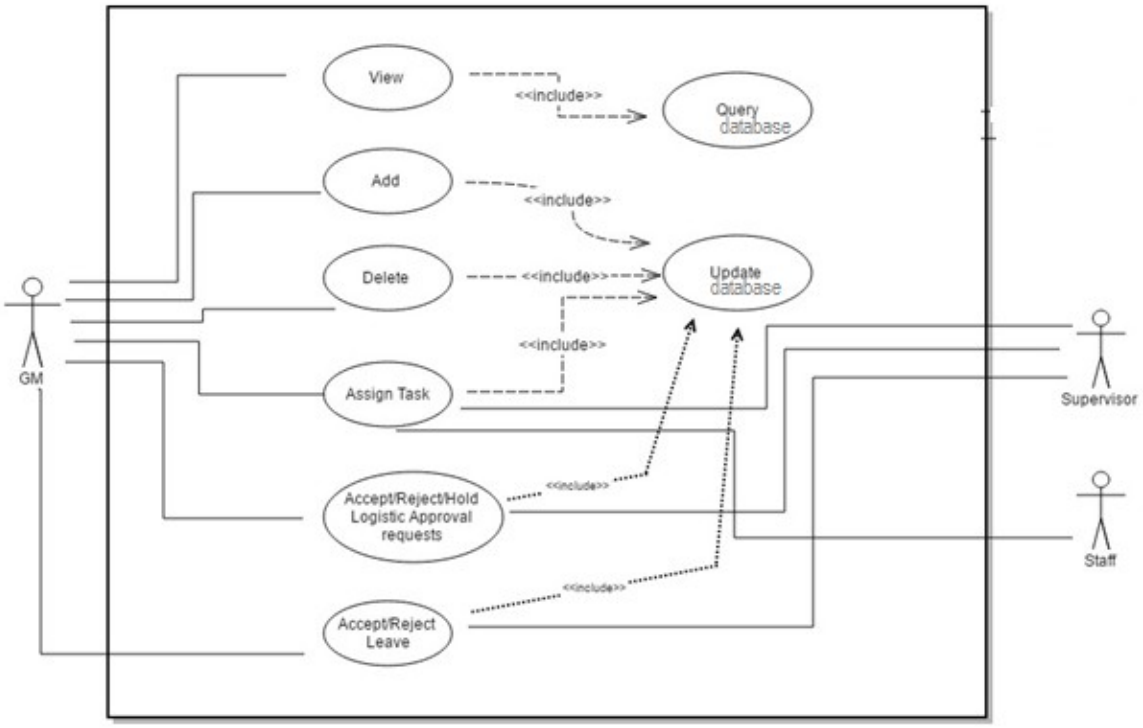
Use Case Diagram

1. **Login and register**



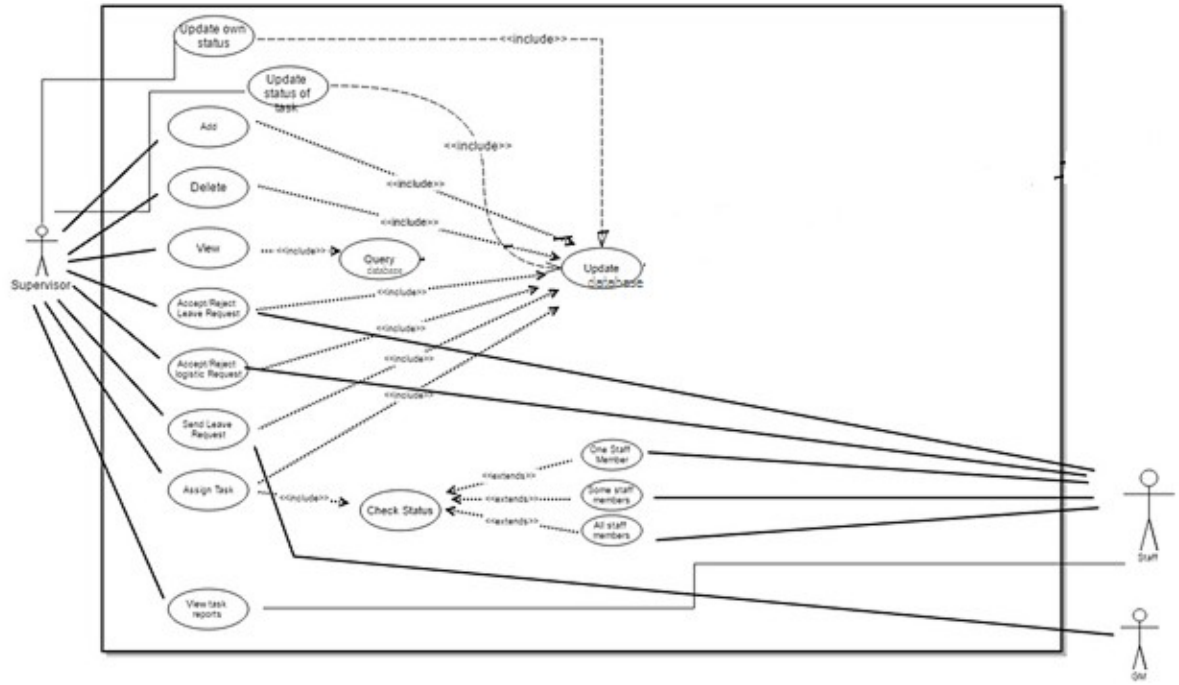
After verification during LOGIN the appropriate user is logged in .Their respective use case diagrams are give below. They describe the functionalities (behaviour) of each of them.
After verification (i.e., no user with the entered username exists) in REGISTER case the data is updated in the database for further retrieval by the GM/supervisor to add/view/delete the new user.

2. **GM**



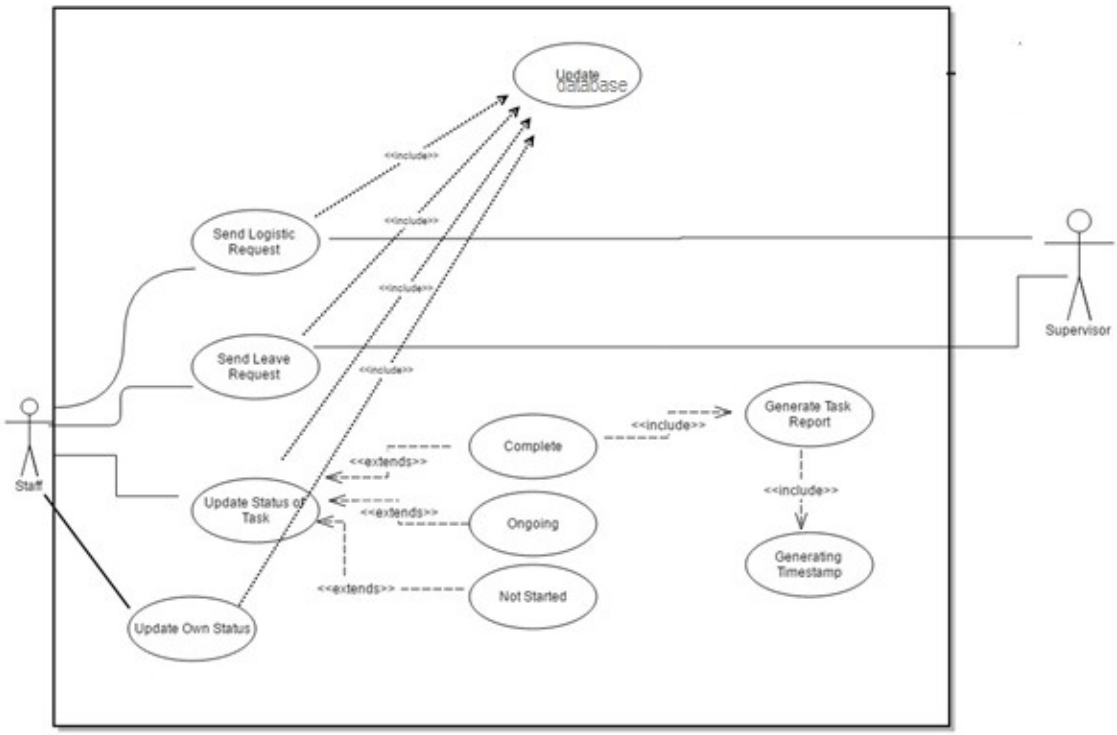
GM can add/view/delete the new user (staff and supervisor) using the database that holds the new entries. For viewing the database is queried while add and delete require updating the database. It can assign task to supervisor and staff and accept/reject/hold logistic approval requests by the supervisor of each department and accept/reject leave requests by the supervisors.

3. **Supervisor**



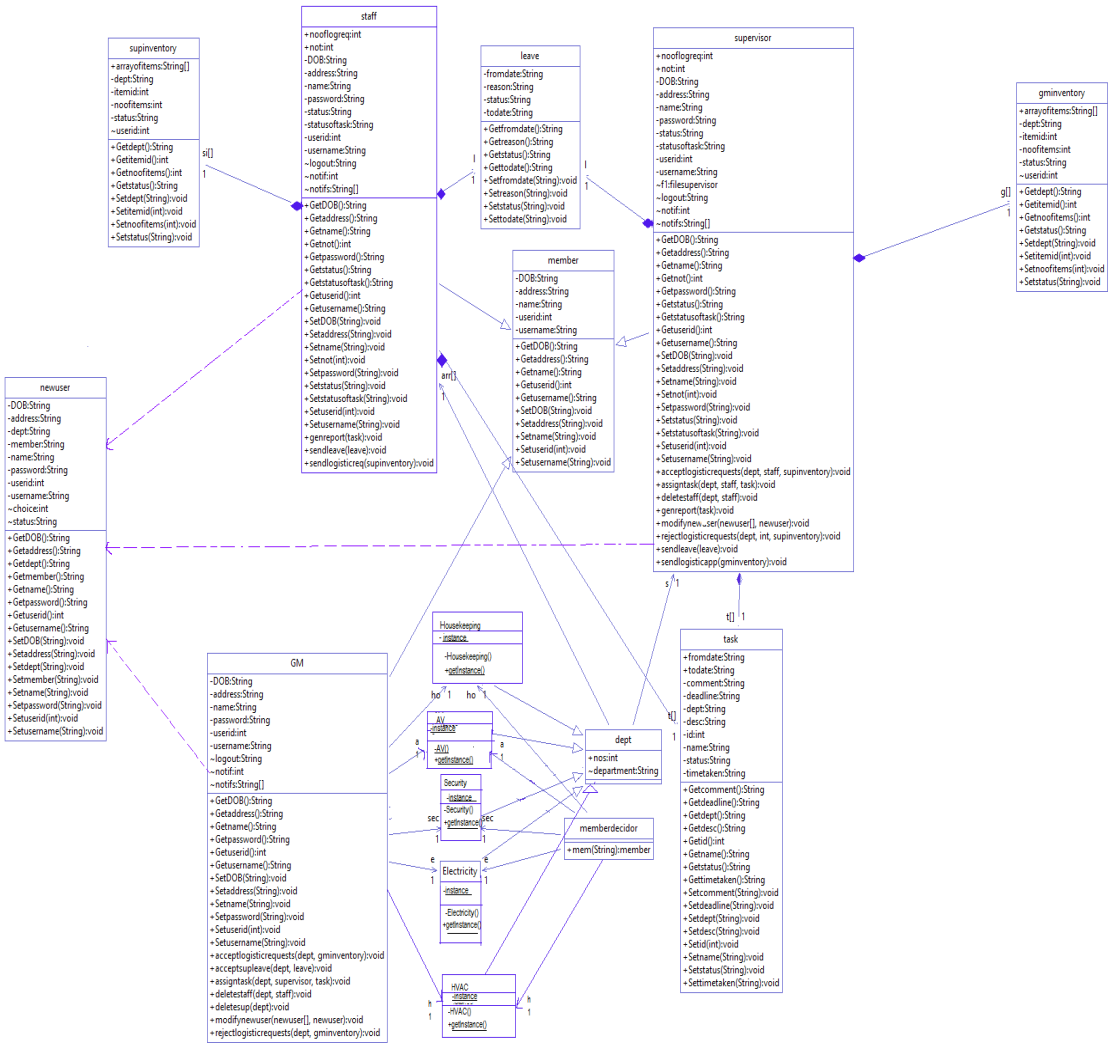
Supervisors can add/view/delete the new user (staff) using the database that holds the new entries. For viewing the database is queried while add and delete require updating the database. It can assign task to staff and accept/reject/hold logistic requirement requests by the staff member of their department and accept/reject leave requests by the staff members. It can view task reports of staffer, send leave request to GM and update status of task and own status. All functionalities except view task report and view new users (staff which requires query) requires updating database.

4. **Staff**



Staffer can send logistic request and leave request to supervisor of his/her department, update his/her task status (complete/ongoing/not started) and own status. If task is complete task report is generated with timestamp. All functionalities require updating the database.

Class diagram



Description

The system includes 5 classes for each department extending from a common class "dept". 3 classes 1 each for GM, supervisor and staff and a class for leave, task, inventory of GM and inventory of supervisor. Each department has a supervisor object and an array of staff object (there is an IS-A relation between each department and class dept and an association relation between class dept and supervisors and staff). All members (GM, supervisors, staff) have common fields name, username (unique), password, DOB, address, userid (unique and system generated).Each field has its set of getters and setters. There is composition relationship between supervisor/staff and their leave, tasks and set of logistic request because the tasks, leave and set of logistic request associated with them cannot exist if the supervisor/staff dies.GM has an association relation with the departments. Departments can exist independent of GM. There are other classes for file handling and GUI purposes. Separate classes for supervisor ,staff and GM accounts with notification bar at home page and dynamic running clock alongwith other necessary tabs.There is a member class which is the parent class of all the people associated with the FMS system (GM, Staff and Supervisors).

Design patterns used

1. Singleton pattern

This pattern involves a single class which is responsible to create an object while making sure that only single object gets created. This class provides a way to access its only object which can be accessed directly without need to instantiate the object of the class. As there are only 5 departments their object is created once and is used everywhere.

2. Factory pattern

In Factory pattern, we create object without exposing the creation logic to the client and refer to newly created object using a common class. In our code it has been used for the creation of staff, supervisor and admin objects as their creation logic is of no use to the client. They do not have to know about how other members and their functionalities are being implemented.

3. Facade pattern

This pattern involves a single class which provides simplified methods required by client and delegates calls to methods of existing system classes. Since reading of database is a compulsory task before any process gets executed it has the method create for all classes which calls their specified create methods.

Contribution

Backend , GUI, design patterns, database handling, initial and final report, doxygen comments, git commits–Sarthika Dhawan (2015170)

GUI, database creation,doxygen comments, initial report- Shubham Kumar (2015098)