**CHAPTER THREE**

**SYSTEM ANALYSIS AND DESIGN**

**3.0 Introduction**

This chapter will establish and discuss the requirement necessary for the Faculty Facility Management System, the analysis and the design of the system to be developed, to ensure that all the requirements of the system is met (technical, economical and operational) and to outline the design process, document the process before been implemented.

**3.1 Software Development Life Cycle**

Software development life cycle is the process of planning, creating, testing and deploying an information system. These processes ensure good software is built with a well-defined structured sequence of stages in software development.

There are usually six stages in this cycle:

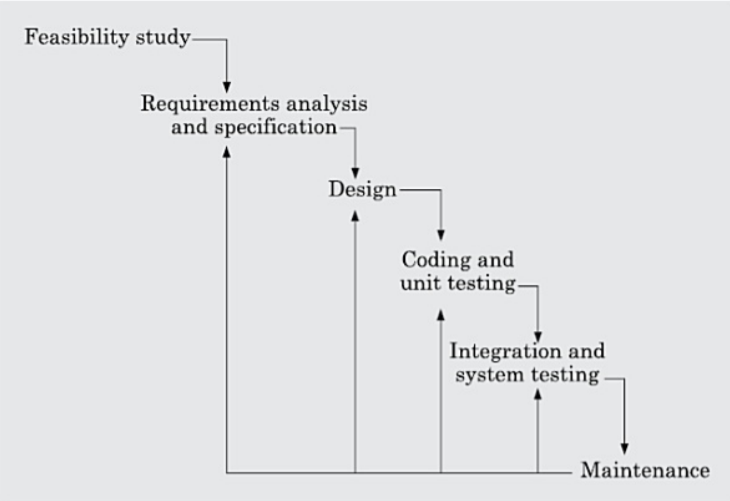
1. Requirement analysis
2. Design
3. Development and Testing
4. Implementation
5. Documentation
6. Evaluation

## 3.1.1 Iterative Model

## Iterative process starts with a simple implementation of a subset of the software requirements and iteratively enhances the evolving versions until the full system is implemented. At each iteration, design modifications are made and new functional capabilities are added. The basic idea behind this method is to develop a system through repeated cycles (iterative) and in smaller portions at a time (incremental).

## Iterative and Incremental development is a combination of both iterative design or iterative method and incremental build model for development. "During software development, more than one iteration of the software development cycle may be in progress at the same time." This process may be described as an "evolutionary acquisition" or "incremental build" approach."

## In this incremental model, the whole requirement is divided into various builds. During each iteration, the development module goes through the requirements, design, implementation and testing phases. Each subsequent release of the module adds function to the previous release. The process continues till the complete system is ready as per the requirement.

This system shall be developed using the iterative model. Because the requirements of the project are likely to change. So, when new things are developed and with constant suggestions the requirement and the design of the whole system is likely to change with time.

**Fig 3.0:***Iterative Model*

**3.2 Requirement Elicitation/Gathering**

The process of generating list of requirements (functional, system, technical etc.) from various stakeholders (users of the system, administrators, IT staff, etc.) that will be used as the basis for the formal requirements definition and understanding the user’s needs and constraints for the system, and this can be achieved through one or more of the following techniques:

1. **Brainstorming**
2. **Document analysis**
3. **Interviews**
4. **Observation**
5. **Prototyping**
6. **Survey/questionnaire**

**3.2.1 Functional Requirement**

Functional requirement describes what the system should do, its capabilities, usability, features and operations:

* A user shall be able to view the list of the staff offices and facilities.
* A user shall be able to report a problem.
* Staff shall be able to request for a facility
* A user shall be able to filter/search for a specific record
* Administrators shall be able to add record.
* Administrators shall be able to modify record.
* Administrators shall be able to delete record.
* The system shall be able to keep all records.

**3.3.2 Non-Functional Requirement**

Requirements that are not directly concerned with the services delivered by the system to its users, they are related to the system properties, below are lists of the non – functional requirement of the system:

* **User Friendly**:

The system is user friendly; the interfaces are well designed and simple language was used so that anybody can use it without any problem. Furthermore, the system is responsive.

* **Reliability:**

The system is reliable, client can visit the site anytime and explore the directory of staff offices and facilities in the faculty buildings. Also, the system administrators can visit it anytime and add, modify, or delete a record from the system.

* **Security:**

The system is secured; only system administrators are allowed to create, copy, modify or delete a record, and they have to be authenticated before they can be able to use the system.

* **Availability:**

The system if successfully implemented and deployed will be available 24/7 and to all users (both mobile and desktop users).

**Table 3.0**: Requirement Description of the System

|  |  |  |
| --- | --- | --- |
| **Req. No.** | **Requirement Description** | **Priority** |
| 1. | Some features of the system   * Admins should be able to login with their credentials. * Admin can add new record * Users can use the system to explore the directory * Users can use the system to report issues | High |
| 2. | Capabilities   * Only authenticated admin will have access to the system * The system has search capability * It can be used to report problems on facilities * Staff can use it to request for a facility | High |
| 3. | Ability to be viewed on various web browser and all different type of devices | High |
| 4. | Language Supported   * English | High |

**3.3 Description of the Existing System**

The current system being used in the faculty is done manually, where individuals have to fill form otherwise known as Maintenance/Repair Request Form (Job card) and submit to the maintenance service department of the University for processing, the form clearly stated that all maintenance/repair request must be in writing by completing the form. This does not or give little room for the student and staff to report the failure of a facility and whether a facility needs to be changed. The facility manager needs to walk through the faculty before having the knowledge of faults or redundancies. This method is practically time consuming, same request may be sent twice or more before being attended to.

**3.4 System Analysis**

This encompasses those tasks that go into determining the needs or conditions to meet the proposed system, taking account of possibly conflicting requirements of various stakeholders, analyzing, documenting, validating and managing the system requirements. This is critical to the success or failure of the system.

**3.4.1 Feasibility Analysis**

In this project consideration of the following feasibilities are considered to ensure that the project has no major obstruction. The following are taken into consideration:

* **Technical Feasibility**:

In this phase, the entire technological requirements to develop the system are radially available. (The operating system, development environment, Xampp Server, M.s Word), the system is flexible and can be expanded further and can give guarantee of accuracy.

* **Economic Feasibility:**

In this phase, a feasibility to determine whether the project goal can be within the resource limit allocated to it was carried out, the cost of the development tools was considered.

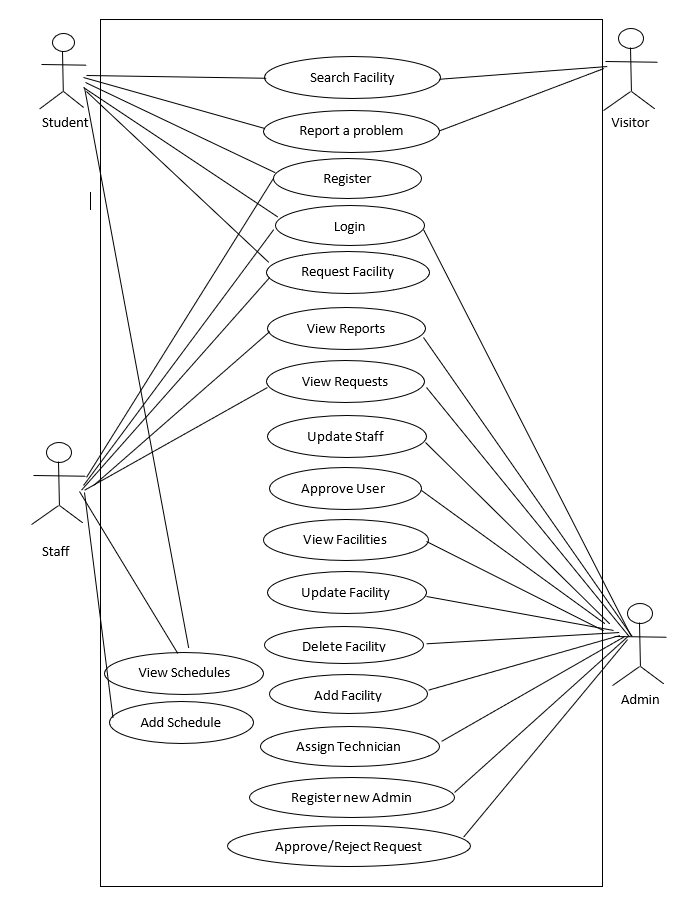
* **Operation Feasibility:**

The operational feasibility such as man-power, time and skills were carried out to make sure that all these factors are feasible and the system certifies user’s objectives.

**3.4.2 Requirement Analysis**

In this phase the entire requirement gathered are analyzed in details and put in diagrammatically so as to get clear idea of what they entail and they are related to each other. It has two basic approaches, the functional oriented approach and the object-oriented approach. For this particular project, object-oriented approach is chosen and will use a Use Case Diagram to represent the system.

**3.4.2.1 Use Case Diagram**



**Fig 3.1: Use Case Diagram**

**Description of the Use Case**

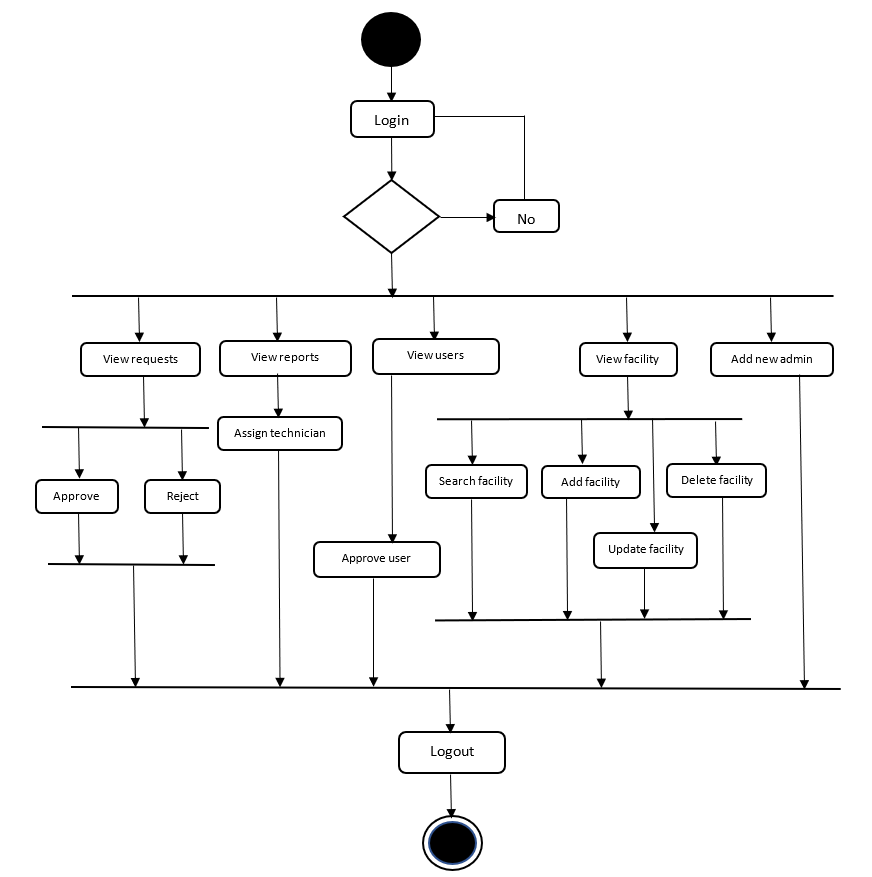
* **Login:** By providing the Login details, the admin is given access to the system.
  + **Actors:** Admin
  + **Precondition:** The system administrators must enter login details to be granted access to the system
  + **Post Condition:** If the details are correct and the use case is successful, an admin shall be logged into the system.
* **Report Faulty Facility:** User can report any faulty facility/equipment within the faculty.
  + **Actors:** User
  + **Precondition:** State the problem with the facility/equipment
  + **Post Condition:** Request send
* **View Reports:** User can view faulty facilities reported to the faculty
* **Actors:** User
* **Precondition:** Report at least one faulty facility
* **Post Condition:** View reports
* **Assign Technician:** Admin can assign a technician to temper with faulty facilities
* **Actors:** Admin
* **Precondition:** There must be at least one reported faults
* **Post Condition:** Assign technician
* **Add New Facility:** New facility can be added to the list of faculty facilities
  + **Actors:** Admin
  + **Precondition:** Login to the system
  + **Post Condition:** If successful new facility will be added.
* **Update Facility:** Added facility can be modified
  + **Actors:** Admin
  + **Precondition:** Login to the system
  + **Post Condition:** If successful the changes made to the facility will be saved.
* **Delete Facility:**  Facility can be removed from the list of the faculty facilities
  + **Actors:** Admin
  + **Precondition:** Login to the system
  + **Post Condition:** If successful, the selected facility will be deleted from the list.
* **Search:** The user can search for a staff office or other facilities
  + **Actors:** User
  + **Precondition:** user must enter a text specifying what he is looking for in the search box.
  + **Post Condition:** the specified facility or staff office is displayed to the user if it exists.
* **Update Staff:** Existing staff records can be easily updated or modified
* **Actors:** Admin
* **Precondition:** Staff must be registered into the system
* **Post Condition:** Records updated
* **Register:** New **s**tudent must register into the system in order to have access to its functionalities
* **Actors:** Student
* **Precondition:** A student must provide the required information
* **Post Condition:** Registered as student
* **Request Facility:** A staff can be able to request for a facility in the faculty
* **Actors:** Staff
* **Precondition:** Staff must be registered into the system
* **Post Condition:** Facility requested
* **Register New Admin:** New admin can be added to control and supervise the system
* **Actors:** Admin
* **Precondition:** Admin must provide the information of new admin
* **Post Condition:** New admin added

**3.5 System Design**

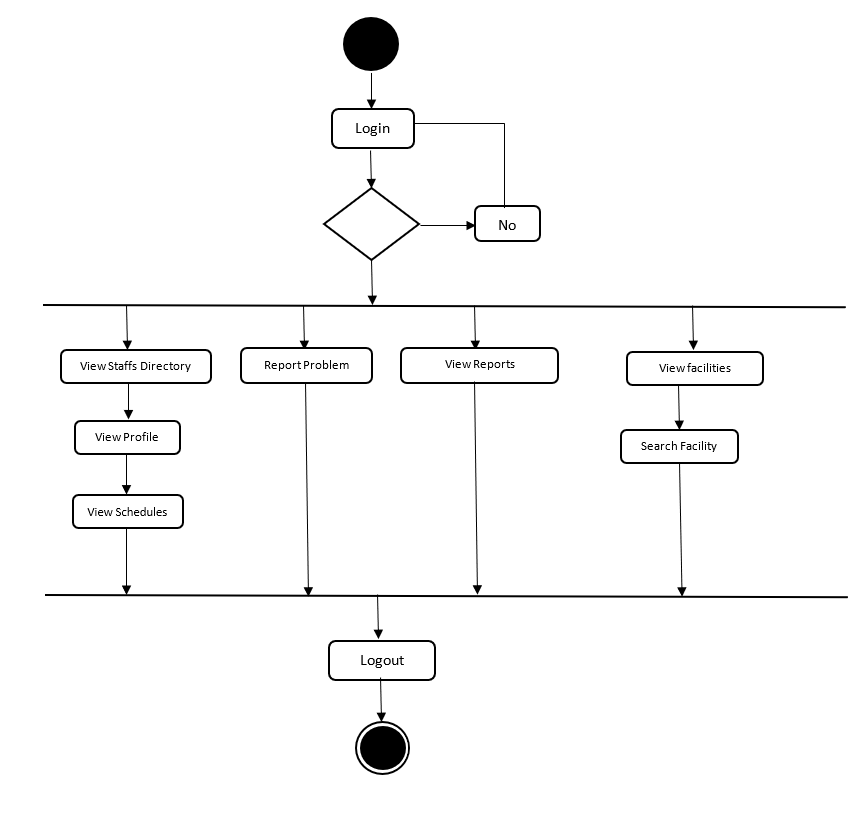
System design is the Process of transforming all the requirements gathered through system analysis into some suitable form that will describe the full structure of the system. Object oriented method was chosen and with that I will identify the classes, attributes and operations.

***3.5.1 Activity Diagram***

This is used to describe dynamic aspects of the system. It is basically a flow chart to represent the flow form one activity to another. For this project I designed two activity diagrams, for the admin and for the client.

****

**Fig 3.2: Activity Diagram for Admin**

****

**Fig 3.3: Activity Diagram for Student**

Request Facility

View Requests

Login

No

View Staff Schedules StaffhedulesDirectory

View facility

Logout

View Reports

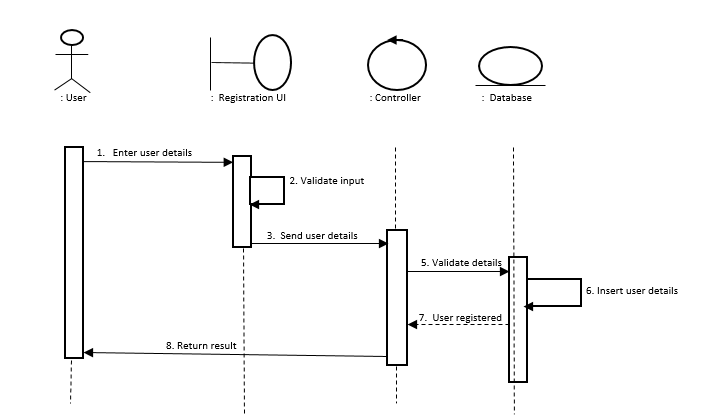
Report Problems

Add Staff Schedule

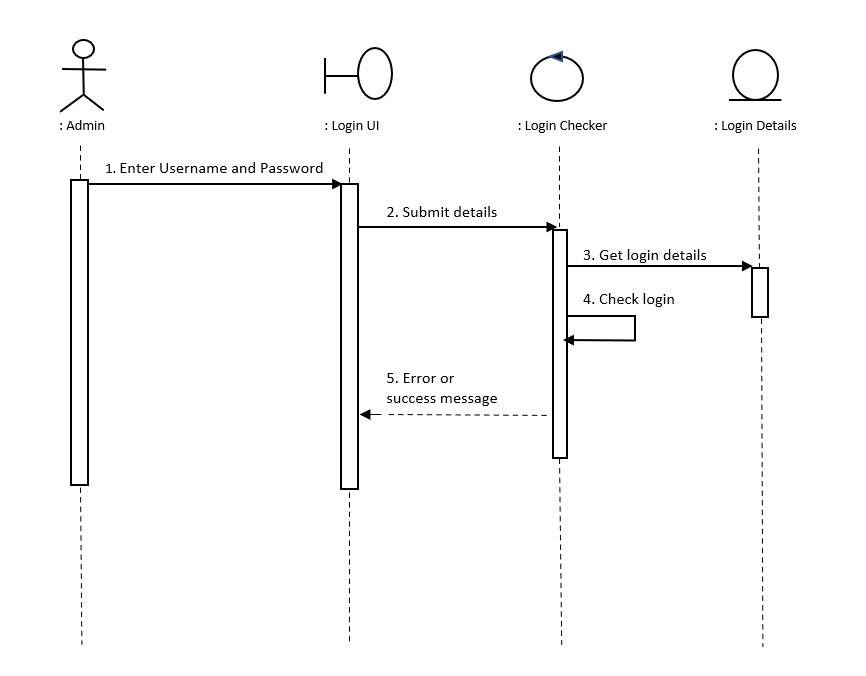
**Fig 3.4:** Activity Diagram for Staff

**3.5.2 Sequence Diagram**

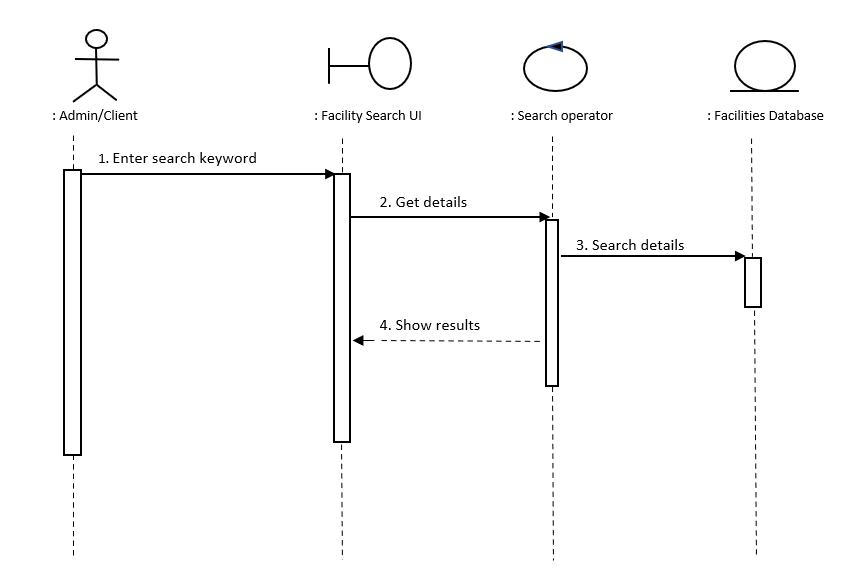
This is an interaction diagram that shows how objects operate with one another and in what order. It shows objects interactions arranged in time sequence.



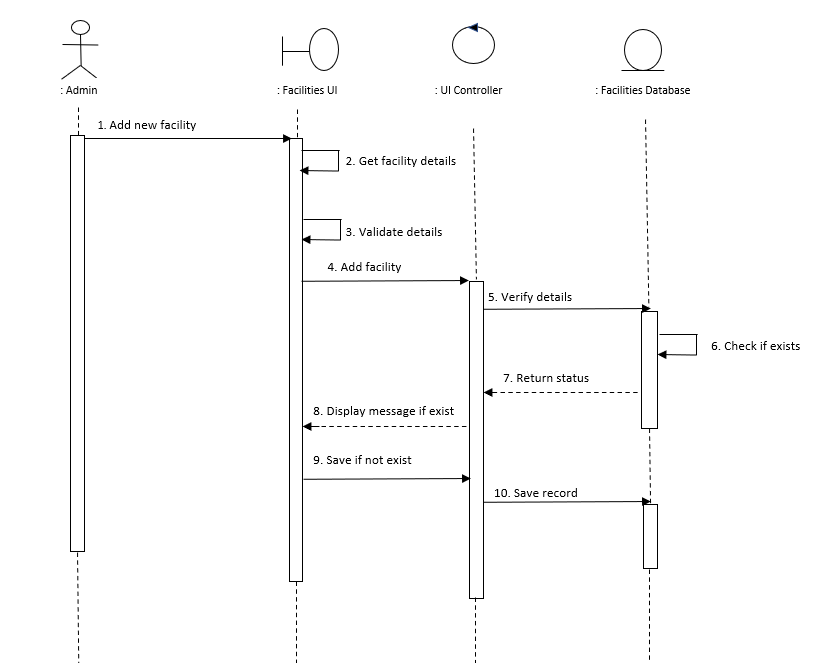
**Fig 3.5:** Sign Up Sequence Diagram

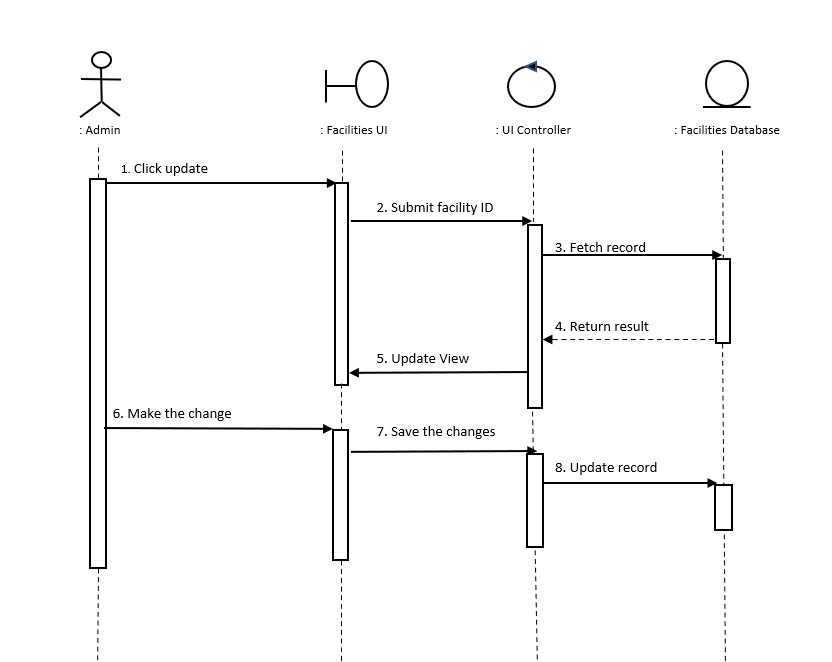


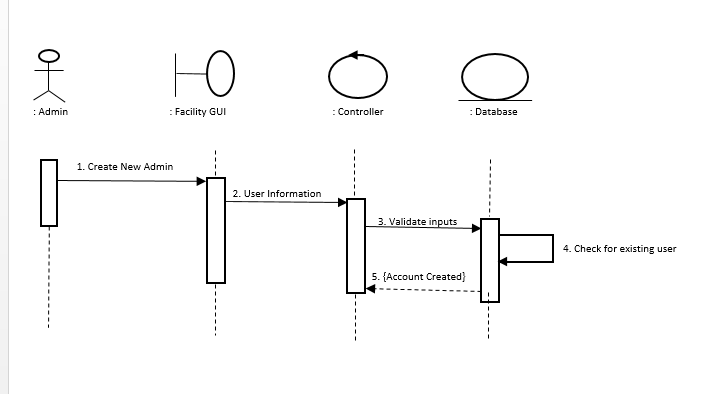
**Fig 3.6:** Login Sequence Diagram



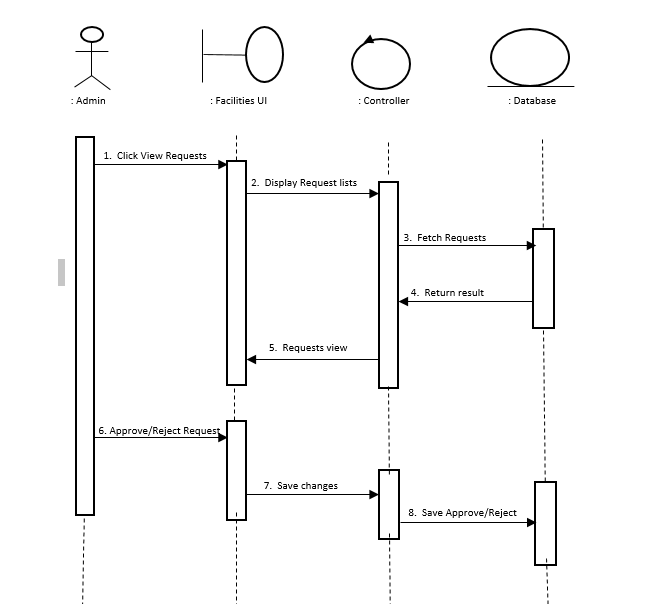
**Fig 3.7: Search Sequence Diagram**

**Fig 3.8: Add Facility Sequence Diagram**

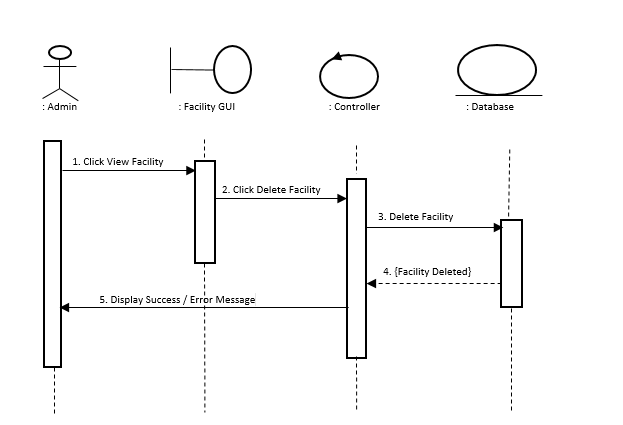
**Fig 3.9: Update Facility Sequence Diagram**



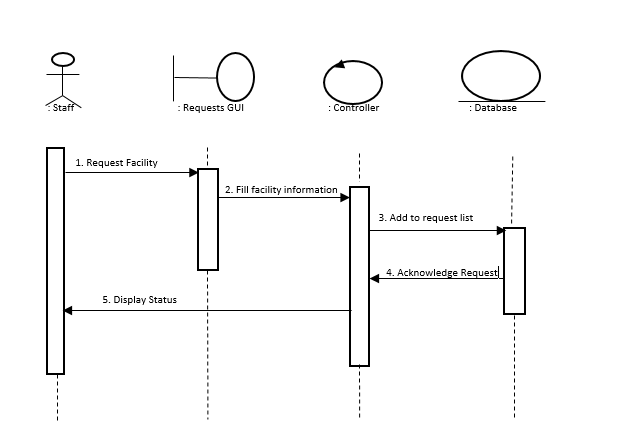
**Fig 3.10: Add New Admin Sequence Diagram**



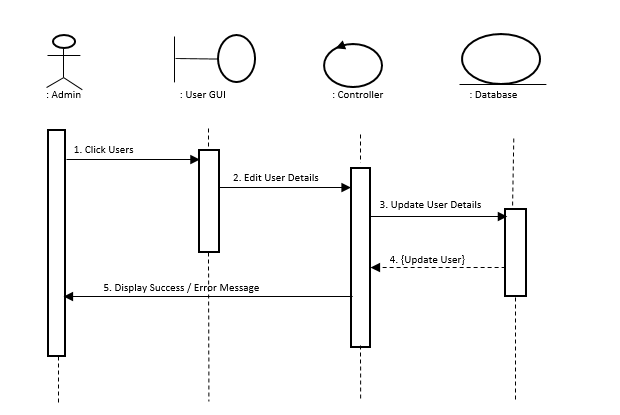
**Fig 3.11: Approve/Reject Request Sequence Diagram**



**Fig 3.12: Delete Facility Sequence Diagram**



**Fig 3.13: Request Facility Sequence Diagram**



**Fig 3.14: Update User Sequence Diagram**

**3.5.3 Class Diagram**

Class diagram is a static diagram that describes the structure of a system by showing the systems classes, their attributes, operations and the relationship among objects.

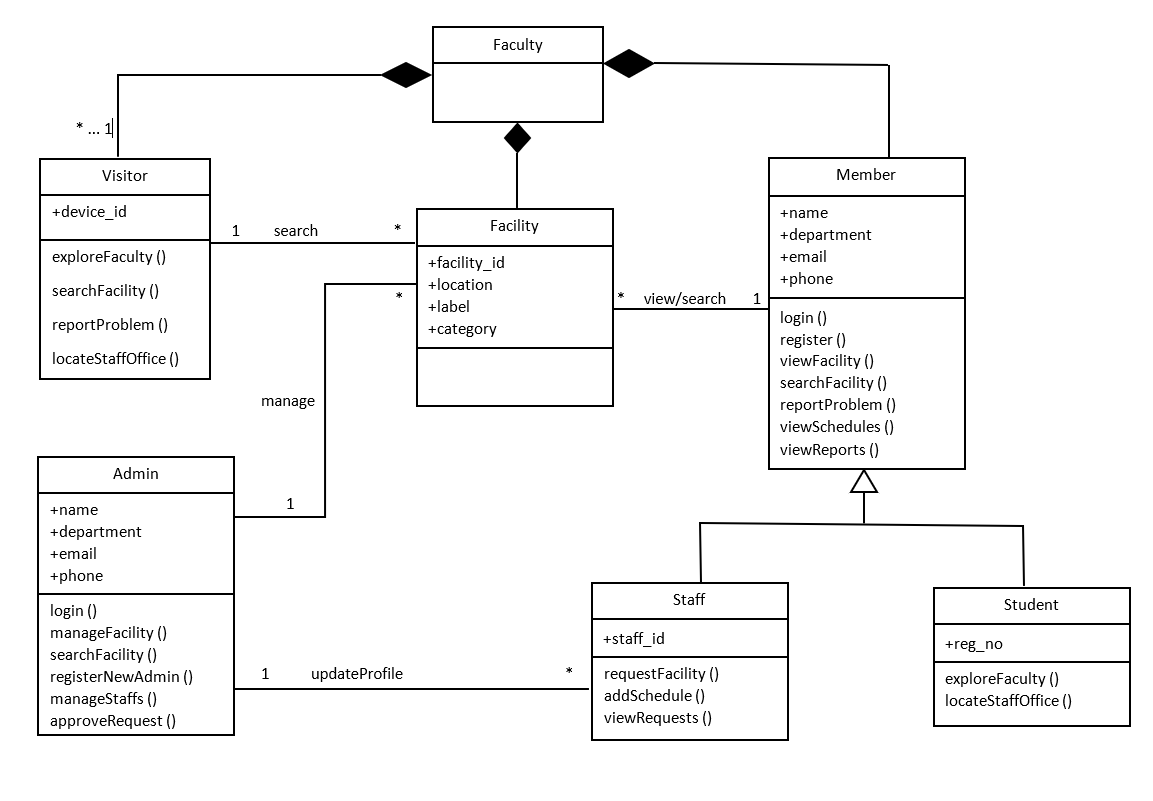


Fig 4.1: Class diagram

**3.6 Database Design**

Database design is the process of producing a detailed data model of database. This data contains

all the logical and physical design choices and physical storage parameters needed to generate a

design in a data definition language, which can then be used to create a database

The core of a Web database application is the database, which is the long-term memory that stores

information for the application. A database is an electronic file cabinet that stores information in

an organized manner so that you can find it when you need it. After all, storing information is

pointless if you can’t find it.

**3.6.1 Database Tables**

A database table is another name for database relation, though the difference is that a table is

usually, a multi-set of rows where a relation is a set and does not allow duplicates. Besides the

actual data rows, tables generally have associated with them some metadata, such as constraints

on the table or on the values within particular columns. Below are the tables of each component

of the system.

**Table 3.1:** Facilities table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Name | Type | Null | Extra | Default | Length |
| id (primary key) | int | No | AUTO\_INCREMENT | None | 11 |
| date\_added | date | No |  | None |  |
| id\_no | varchar | No |  | None | 20 |
| category | varchar | No |  | None | 30 |
| sub\_category | varchar | No |  | None | 30 |
| Type | varchar | No |  | None | 30 |
| location | varchar | No |  | None | 30 |
| Floor | varchar |  |  | None | 20 |
| Label | varchar | No |  | None | 30 |
| Status | varchar | No |  | None | 20 |

**Table 3.2:** admins table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Name | Type | Null | Extra | Default | Length |
| Id (primary key) | Int | No | AUTO\_INCREMENT | None | 11 |
| first\_name | varchar | No |  | None | 30 |
| last\_name | varchar | No |  | None | 30 |
| department | varchar | No |  | None | 30 |
| email | varchar | No |  | None | 30 |
| phone | varchar | No |  | None | 15 |

**Table 3.3:** staffs table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Name | Type | Null | Extra | Default | Length |
| id (primary key) | int | No | AUTO\_INCREMENT | None | 11 |
| staff\_id | varchar | No |  | None | 25 |
| full\_name | varchar | No |  | None | 30 |
| department | varchar | No |  | None | 50 |
| Email | varchar | No |  | None | 30 |
| Phone | varchar | No |  | None | 15 |
| Rank | varchar | No |  | None | 30 |
| Position | varchar | No |  | None | 30 |
| office\_no | varchar | No |  | None | 25 |

**Table 3.4:** students table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Name | Type | Null | Extra | Default | Length |
| id (primary key) | int | No | AUTO\_INCREMENT | None | 11 |
| reg\_no | varchar | No |  | None | 25 |
| department | varchar | No |  | None | 30 |
| full\_name | varchar | No |  | None | 50 |
| Email | varchar | No |  | None | 30 |
| Phone | varchar | No |  | None | 15 |

**Table 3.5:** requests table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Name | Type | Null | Extra | Default | Length |
| id (primary key) | int | No | AUTO\_INCREMENT | None | 11 |
| date\_requested | varchar | No |  | None | 25 |
| facility\_requested | varchar | No |  | None | 30 |
| requested\_by | varchar | No |  | None | 50 |
| \_id | varchar | No |  |  |  |
| Phone | varchar | No |  | None | 30 |
| Email | varchar | No |  | None | 15 |
| remark | varchar |  |  |  |  |
| Status | varchar |  |  |  |  |

**Table 3.6:** reports table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Name | Type | Null | Extra | Default | Length |
| id (primary key) | int | No | AUTO\_INCREMENT | None | 11 |
| date\_reported | varchar | No |  | None | 25 |
| facility\_reported | varchar | No |  | None | 30 |
| Location | varchar | No |  | None | 50 |
| Label | varchar | No |  | None | 20 |
| full\_name | varchar | No |  | None | 30 |
| problem\_desc | varchar | No |  | None | 15 |
| Img | varchar |  |  | None |  |
| Status | varchar |  |  | None |  |

**Table 3.7:** schedules table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Name | Type | Null | Extra | Default | Length |
| id (primary key) | int | No | AUTO\_INCREMENT | None | 11 |
| staff\_id | varchar | No |  | None | 30 |
| day | varchar | No |  | None | 15 |
| start-time | varchar | No |  | None | 15 |
| end\_time | varchar | No |  | None | 15 |
| event | varchar | No |  | None | 100 |
| event\_type | varchar | No |  | None | 15 |