

# **Shri Sant Gajanan Maharaj College of Engineering, Shegaon**



**Topic: - QR based Attendance System**

**Subject - CSKILL LAB 4**

---

**Group No - 5**

---

**Presented By**

---

Mayur Nehare - 49

Nayan Sonkalyari – 51

---

## **INDEX**

<b>Sr NO</b>	<b>Title</b>	<b>Page No</b>
	<b>Introduction</b>	
1	Identifying the Requirements from problem statement.	1-3
2	Estimation of project metrics	4-5
3	Modeling UML use case diagrams and capturing use case scenarios.	6-8
4	ER Modeling from the problem statements	9-10
5	Identifying Domain Classes from the Problem Statements	11-13
6	Statechart and Activity Modeling	14-16
7	Modeling UML Class Diagrams and Sequence diagrams	17-18
8	Modeling Data Flow Diagrams	19-20

# **Introduction**

## **Abstract**

The proposed system is an application, which will be generating the QR Code by entering the student details with a login system provided to admin and mark students attendance. The student will need to scan the QR code in order to mark their attendance. The purpose of the QR based attendance system is to computerize the traditional way of recording attendance and provide an easiest and smart way to track attendance in institutions.

## **Problem statement**

- Development of a QR CODE BASED ATTENDANCE SYSTEM.
- Integrating System with QR code and SQLite to store attendance results.

## **Intended Audience**

The audience of this system will be:

1. Students
2. Registration office.

## **Feasibility**

- Economic feasibility: The developed system is time effective because attendance is marked automatically. It is also cost effective because of no use of paperwork.
- Technical feasibility: The system is economic and it does not use any other additional Hardware and software.
- Behavioral feasibility: The system is user friendly.

## Experiment No: - 1

**Aim: -** Identifying the Requirements from problem statement.

**Theory: -**

Requirements identification is the first step of any software development project. Until the requirements of a client have been clearly identified, and verified, no other task (design, coding, testing) could begin. Usually business analysts having domain knowledge on the subject matter discuss with clients and decide what features are to be implemented.

### Categorization of Requirements

Based on the target audience or subject matter, requirements can be classified into different types, as stated below:

- **User requirements:** They are written in natural language so that both customers can verify their requirements have been correctly identified
- **System requirements:** They are written involving technical terms and/or specifications, and are meant for the development or testing teams

Requirements can be classified into two groups based on what they describe:

- **Functional requirements (FRs):** These describe the functionality of a system -- how a system should react to a particular set of inputs and what should be the corresponding output.
- **Non-functional requirements (NFRs):** They are not directly related what functionalities are expected from the system. However, NFRs could typically define how the system should behave under certain situations. For example, a NFR could say that the system should work with 128MB RAM. Under such condition, a NFR could be more critical than a FR.

Table 01: Identifier and priority for software requirements		
#	Requirement	Priority
R1	Adding a New student	High
R2	Marking Attendance	High
R3	Login	High
R4	View Record	High

## **Functional Requirements**

### **1. Adding a New student**

Function: Sign up a new student to the system.

Priority: High (Required for first release)

Requirements: To add a new user to the system, all of them should have registered in the admission office. Then admin will make entries of their information in proposed system where their individual QR code will be generated.

### **2. Marking Attendance**

Function: Marking attendance

Priority: High (Required for college entry)

Requirements: When students enter college building, they must scan their QR generated in the above phase in input device. If the scan matches, students can enter the college. If the scan does not match, the student must check with the registration office to figure out the problem.

### **3. Login**

Function: Admin can Login in the system

Priority: High

Requirements: To log in the system Admin need to have their credentials with them. After entering the required field admin can access the system.

### **4. View Record**

Function: Monitoring data

Priority: High

Requirements: Admin Office has to regularly monitor the student's attendance report and if irregularity found then take action accordingly.

## **Non Functional Requirements**

### **1. Performance Requirement**

- The system should be accessible during college hours.
- The system should work without any fail and QR scan should take less time.

## **2. Security Requirements**

- The Record can only be accessed and managed by Admin.
- The Password should not be plain text instead it should contain a combination of uppercase and lowercase letters, symbols and numbers.

## **Conclusion: -**

In this experiment we learned how to identify functional and non-functional requirements from a given problem statement. Functional and non-functional requirements are the primary components of a Software Requirements Specification.

## Experiment No: - 2

**Aim: -** Estimation of project metrics

**Theory: -**

### Introduction

After gathering the entire requirements specific to software project usually we need to think about different solution strategy for the project. Expert business analysts are analyzing their benefits and as well as their shortcomings by means of cost, time and resources require to develop it.

### Project Estimation Techniques

A software project is not just about writing a few hundred lines of source code to achieve a particular objective. The scope of a software project is comparatively quite large, and such a project could take several years to complete. However, the phrase "quite large" could only give some (possibly vague) qualitative information. As in any other science and engineering discipline, one would be interested to measure how complex a project is. One of the major activities of the project planning phase, therefore, is to estimate various project parameters in order to take proper decisions. Some important project parameters that are estimated include:

- **Project size:** What would be the size of the code written say, in number of lines, files, modules?
- **Cost:** How much would it cost to develop a software? A software may be just pieces of code, but one has to pay to the managers, developers, and other project personnel.
- **Duration:** How long would it be before the software is delivered to the clients?
- **Effort:** How much effort from the team members would be required to create the software?

In this experiment we will focus on COCOMO model for estimating project metrics:

### COCOMO

COCOMO (Constructive Cost Model) was proposed by Boehm. According to him, there could be three categories of software projects: organic, semidetached, and embedded. The classification is done considering the characteristics of the software, the development team and environment. These product classes typically correspond to application, utility and system programs, respectively. Data processing programs could be considered as application programs. Compilers, linkers, are examples of utility programs. Operating systems, real-time system programs are examples of system programs. The concept of organic, semidetached, and embedded systems are described below.

- **Organic:** A development project is said to be of organic type, if
  - The project deals with developing a well understood application
  - The development team is small
  - The team members have prior experience in working with similar types of projects

- **Semidetached:** A development project can be categorized as semidetached type, if
  - The team consists of some experienced as well as inexperienced staff
  - Team members may have some experience on the type of system to be developed
- **Embedded:** Embedded type of development project are those, which
  - Aims to develop a software strongly related to machine hardware
  - Team size is usually large

Boehm suggested that estimation of project parameters should be done through three stages: Basic COCOMO, Intermediate COCOMO, and Complete COCOMO. In our project we are going with basic COCOMO model.

The basic COCOMO model helps to obtain a rough estimate of the project parameters. It estimates effort and time required for development in the following way:

$$\text{Effort} = a * (\text{KLOC})^b \text{ PM}$$

$$\text{Tdev} = 2.5 * (\text{Effort})^c \text{ Months}$$

Where,

- KLOC is the estimated size of the software expressed in Kilo Line of Code
- a, b, c are constants determined by the category of software project
- Effort denotes the total effort required for the software development, expressed in person months (PMs)
- Tdev denotes the estimated time required to develop the software (expressed in months)

The value of the constants a, b, c are given below:

Software project	a	b	c
Organic	2.4	1.05	0.38
Semi-detached	3.0	1.12	0.35
Embedded	3.6	1.20	0.32

As this project belongs to organic category basic COCOMO model, the projected effort required for this project becomes

$$\text{Effort} = 2.4 * (0.178)^{1.05} \text{ PM}$$

$$= 0.4 \text{ PM (approx)}$$

$$\text{Tdev} = 2.5 * (0.4)^{0.38} \text{ Months}$$

$$= 1.8 \text{ Months (approx)}$$

## Conclusion: -

In this experiment we learned how to estimate cost, effort and duration for a software project, and then select one solution approach which will be found suitable to fulfill the organizational goal.



## Experiment No: - 3

**Aim: -** Modeling UML use case diagrams and capturing use case scenarios.

**Theory: -**

From the given problem statement we can identify a list of actors and use cases as shown in tables 1 & 2 respectively. We assign an identifier to each use case, which we would be using to map from the software requirements identified earlier.

Table 2: List of actors

Actor	Description
Admin	Add student's record in the system and make track of student attendance.
Student	Mark their attendance.

Table 3: List of use cases

#	Use Case	Description
UC1	Login	Authenticates admin to let him avail the facilities
UC2	Add member	Allows admin to generate QR code based on students details
UC3	Mark Attendance	Allows Students to mark their attendance
UC4	View Record	Allows admin to regularly monitor the student's attendance report

Before presenting the details of individual use cases, let us do a mapping from requirements specification to use cases.

Table 4: Mapping functional requirements to use cases

FR #	FR Description	Use Case(s)
R1	Adding a New student	UC3
R2	Marking Attendance	UC1
R3	Login	UC2
R4	View Record	UC4

Now let us deal with the inner details of a few use cases and the actors with whom they are associated.

**Table 5: UC1 – Mark Attendance**

Use Case	UC1. Mark Attendance
Description	Allows Students to mark their attendance
Assumptions	<ol style="list-style-type: none"> <li>1. Student is already registered</li> <li>2. Student has QR provided by the admin</li> </ol>
Actors	<ul style="list-style-type: none"> <li>• Student</li> </ul>
Steps	<ol style="list-style-type: none"> <li>1. Student shows QR Code to the input device</li> <li>2. IF successful THEN Mark Attendance ELSE display error</li> </ol>

**Table 6: UC2 –Login**

Use Case	UC2. Login
Description	Authenticates admin to let him avail the facilities
Assumptions	<ol style="list-style-type: none"> <li>1. Admin is logged in</li> <li>2. The menu is available</li> </ol>
Actors	<ul style="list-style-type: none"> <li>• Admin</li> </ul>
Steps	<ol style="list-style-type: none"> <li>1. User types in user ID</li> <li>2. User types in password</li> <li>3. User press Enter</li> <li>4. IF successful THEN show home page ELSE display error</li> </ol>

**Table 7: UC3 – Add member**

Use Case	UC3. Add member
Description	Allows admin to generate QR code based on students details
Assumptions	Admin is logged in The menu is available
Actors	Admin
Steps	Admin adds details of the student At the same time QR is generated

**Table 8: UC4 – View Record**

Use Case	UC4. View Record
Description	Allows admin to regularly monitor the student's attendance report
Assumptions	1. Admin is logged in 2. The menu is available
Actors	<ul style="list-style-type: none"><li>Admin</li></ul>
Steps	1. Admin will view record to monitor students attendance

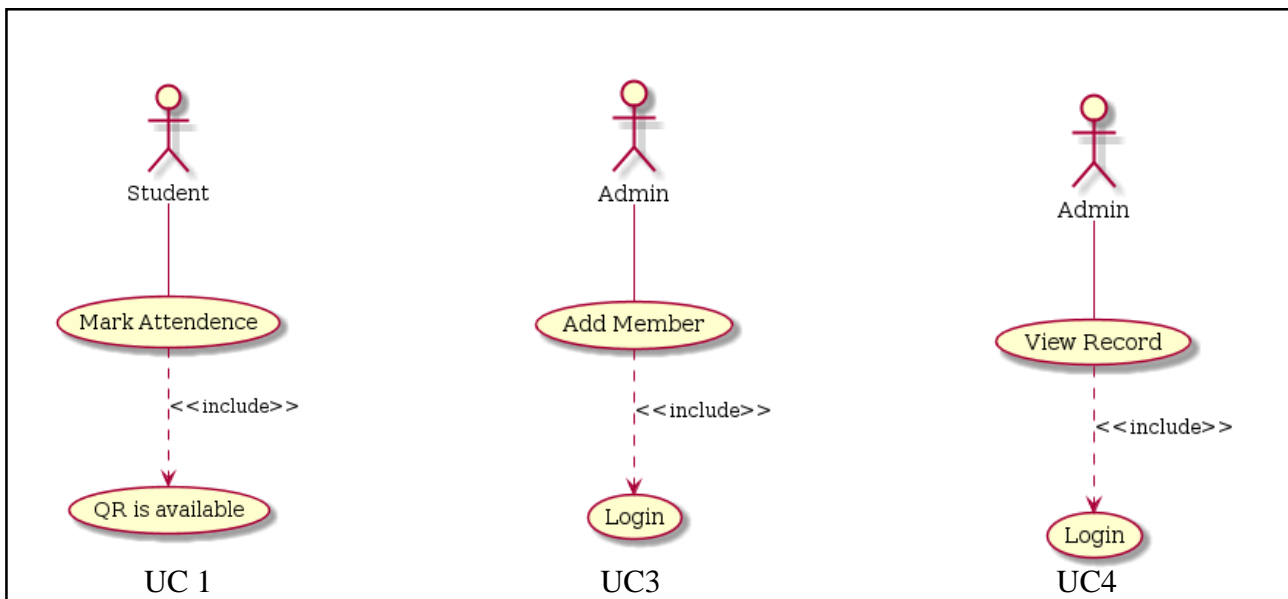


Figure 1: Use case diagrams showing different use cases

### Conclusion: -

In this experiment we learned how use cases and actors can be captured and how different use cases are related in a system.

## Experiment No: - 4

**Aim:** - ER modeling from the problem statements

**Theory:** -

### Introduction

Developing databases is a very important task to develop a system. Before going to form exact database tables and establishing relationships between them, we conceptually or logically can model our database using ER diagrams.

A robust database backend is essential for a high-quality information system. Database schema should be efficiently modeled, refined, and normalized. In this section we would develop a simple ER model for the QRAS.

### Entity Relationship Model

Entity-Relationship model is used to represent a logical design of a database to be created. In ER model, real world objects (or concepts) are abstracted as entities, and different possible associations among them are modeled as relationships.

For example, student and school -- they are two entities. Students study in school. So, these two entities are associated with a relationship "Studies in". As another example, consider a system where some job runs every night, which updates the database. Here, job and database could be two entities. They are associated with the relationship "Updates".

### Entity Set and Relationship Set

An entity set is a collection of all similar entities. For example, "Student" is an entity set that abstracts all students. Ram, John are specific entities belonging to this set. Similarly, a "Relationship" set is a set of similar relationships.

### Attributes of Entity

Attributes are the characteristics describing any entity belonging to an entity set. Any entity in a set can be described by zero or more attributes.

For example, any student has got a name, age, an address. At any given time a student can study only at one school. In the school he would have a roll number, and of course a grade in which he studies. These data are the attributes of the entity set Student.

### Keys

One or more attribute(s) of an entity set can be used to define the following keys:

- **Super key:** One or more attributes, which when taken together, helps to uniquely identify an entity in an entity set. For example, a school can have any number of students. However, if we know grade and roll number, then we can uniquely identify a student in that school.
- **Candidate key:** It is a minimal subset of a super key. In other words, a super key might contain extraneous attributes, which do not help in identifying an object uniquely. When such attributes are removed, the key formed so is called a candidate key.

- **Primary key:** A database might have more than one candidate key. Any candidate key chosen for a particular implementation of the database is called a primary key.
- **Prime attribute:** Any attribute taking part in a super key

The first step towards ER modeling is to identify the set of relevant entities from the given problem statement. The two primary, and obvious, entity sets in this context are "Student" and "Admin". The entity set "Student" represents all students who have registered themselves with the QRAS. While registering with the QRAS one has to furnish a lot of personal and professional information. This typically includes Name, Student ID (roll no for students), Contact No, and Department in this institute. All these pieces of information that user has to provide are sufficient to describe a particular member. These characteristics are the attributes of the entities belonging to the entity set "Student".

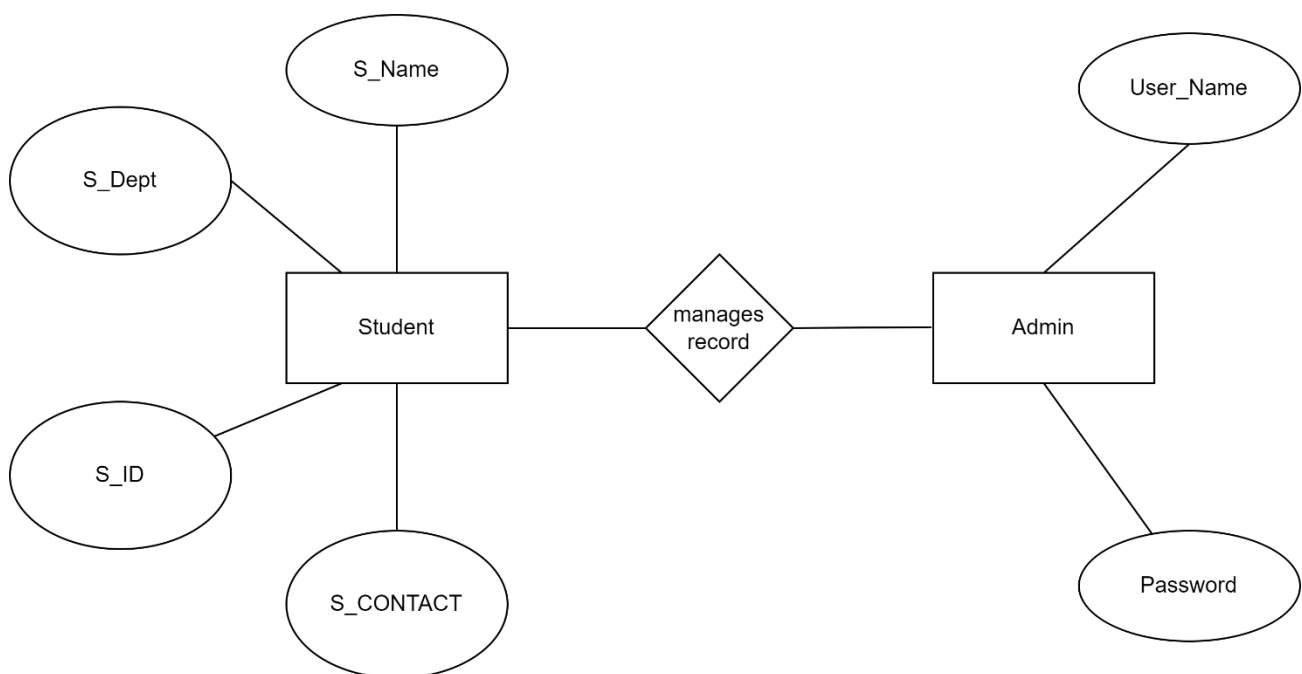


Figure 2: ER Diagram

### Conclusion: -

In this experiment we learned how to find the entities, its attributes and how the relationships between the entities can be established for a system.

## Experiment No: - 5

**Aim: -** Identifying Domain Classes from the Problem Statements

**Theory: -**

### Introduction

Same types of objects are typically implemented by class in object oriented programming. As the structural unit of the system can be represented through the classes, so, it is very important to identify the classes before start implementing all the logical flows of the system.

### Domain Class

In Object Oriented paradigm Domain Object Model has become subject of interest for its excellent problem comprehending capabilities towards the goal of designing a good software system. Domain Model, as a conceptual model gives proper understanding of problem description through its highly effective component – the Domain Classes. Domain classes are the abstraction of key entities, concepts or ideas presented in the problem statement. Domain classes are used for representing business activities during the analysis phase.

### Steps to Identify Domain Classes from Problem Statement

We now present the steps to identify domain classes from a given problem statement. This approach is mostly based on the “Grammatical approach using nouns”.

1. Make a list of potential objects by finding out the nouns and noun phrases from narrative problem statement
2. Apply subject matter expertise (or domain knowledge) to identify additional classes
3. Filter out the redundant or irrelevant classes
4. Classify all potential objects based on categories. We follow the category table as described by Ross

Categories	Explanation
People	Humans who carry out some function
Places	Areas set aside for people or things
Things	Physical objects
Organizations	Collection of people, resources, facilities and capabilities having a defined mission
Concepts	Principles or Ideas not tangible
Events	Things that happen (usually at a given date and time), or as a steps in an ordered sequence

5. Group the objects based on similar attributes. While grouping we should remember that
  1. Different nouns (or noun phrases) can actually refer to the same thing (examples: house, home, abode)
  2. Same nouns (or noun phrases) could refer to different things or concepts (example: I go to school every day / This school of thought agrees with the theory)
6. Give related names to each group to generate the final list of top level classes
7. Iterate over to refine the list of classes

From the given problem statement we can identify the following nouns and noun phrases:

- QR code based attendance system
- QR Code
- Institution
- Student
- Admin
- Login
- System
- Attendance
- Record
- Software
- Registration Office
- Software Engineering
- Information

People	Places	Things
Student Admin	Registration Office	QR Code System
Organizations	Concepts	Events
Institution	QR code based attendance system Software Engineering Login Record Software Information	Attendance

The nouns and noun phrases in the problem statement no of classes. However, all of them may not be relevant. If we filter these entries, we might find that the following set of classes directly relate to the business activities of QRAS:

- Student
- Admin
- Attendance
- Record

### **Conclusion: -**

In this experiment we learned how to identify the classes from a given problem statement.



## Experiment No: - 6

**Aim: -** Statechart and Activity Modeling

**Theory: -**

### Introduction

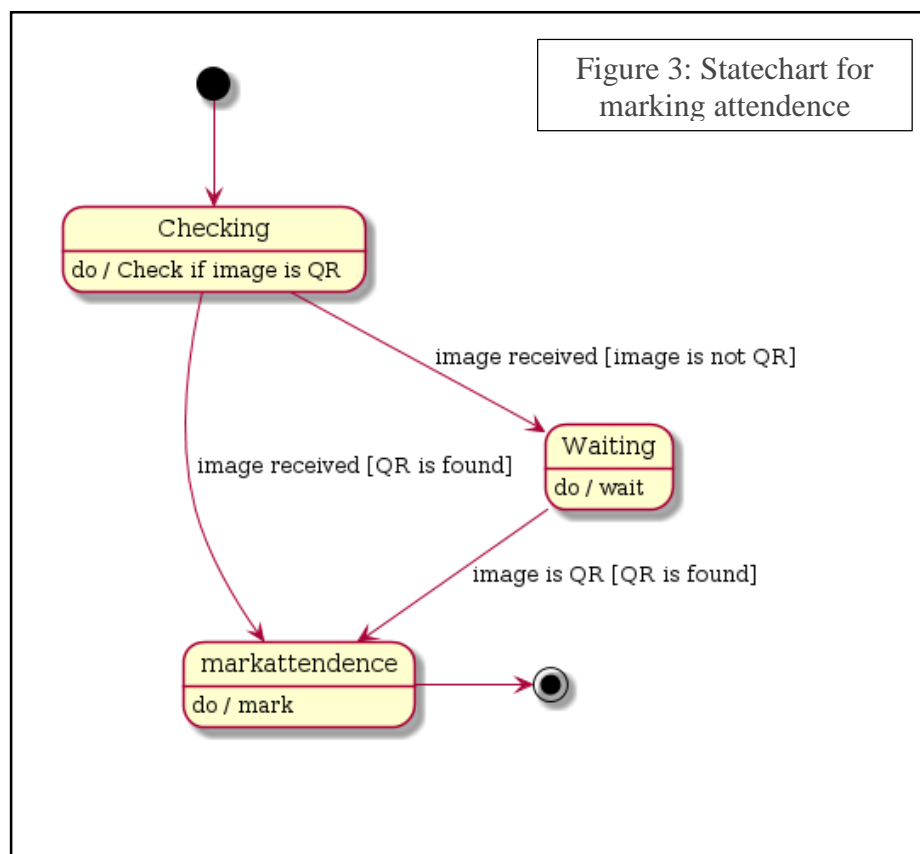
Capturing the dynamic view of a system is very important for a developer to develop the logic for a system. State chart diagrams and activity diagrams are two popular UML diagram to visualize the dynamic behavior of an information system.

### Statechart diagram

It is a pictorial representation of a system, with all its states, and different events that lead transition from one state to another.

### Activity diagrams

It fall under the category of behavioural diagrams in Unified Modeling Language. It is a high level diagram used to visually represent the flow of control in a system. It has similarities with traditional flow charts. However, it is more powerful than a simple flow chart since it can represent various other concepts like concurrent activities, their joining, and so on



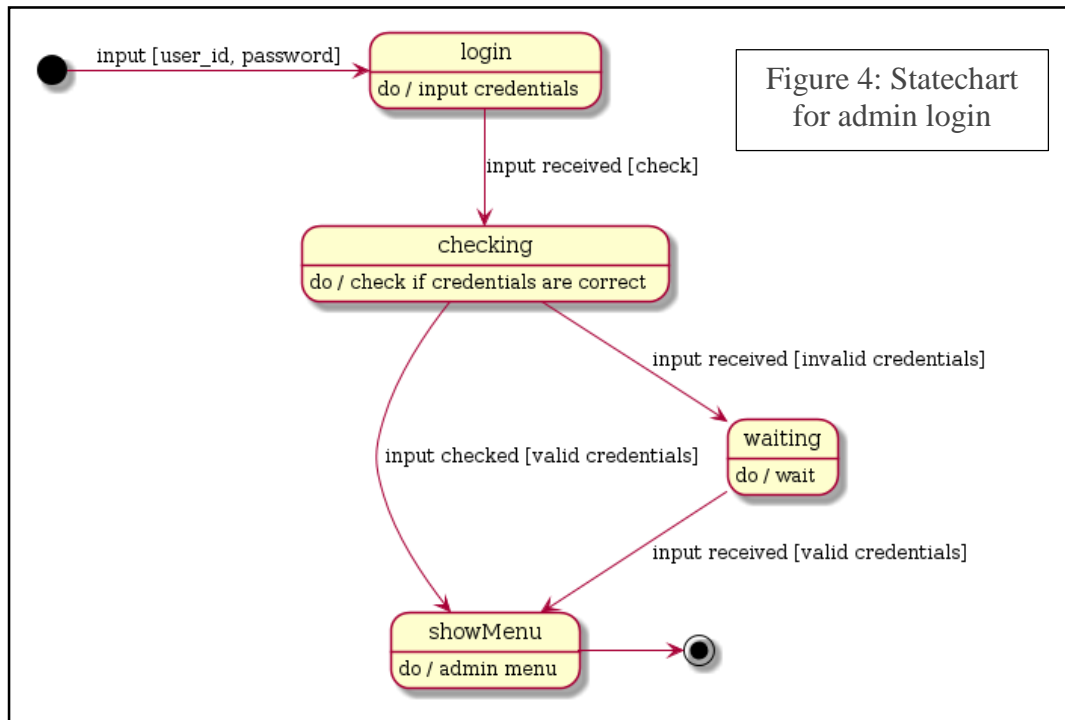


Figure 4: Statechart for admin login

A new user fills up the registration form for this system (either online or in paper), and submits to the registration office. Of course, an already registered student can't create another account for himself (or, herself). For students who don't have an account already and have submitted their registration forms, the admin verifies the information provided, possibly against the central database used by the institution. If all information have been provided correctly, admin goes on with creating a new account for the student. Otherwise, the user is asked to provide all and correct information in his (her) registration form. Once a new account has been created for the user, he (she) is being issued a QR, which is to be used while marking attendance

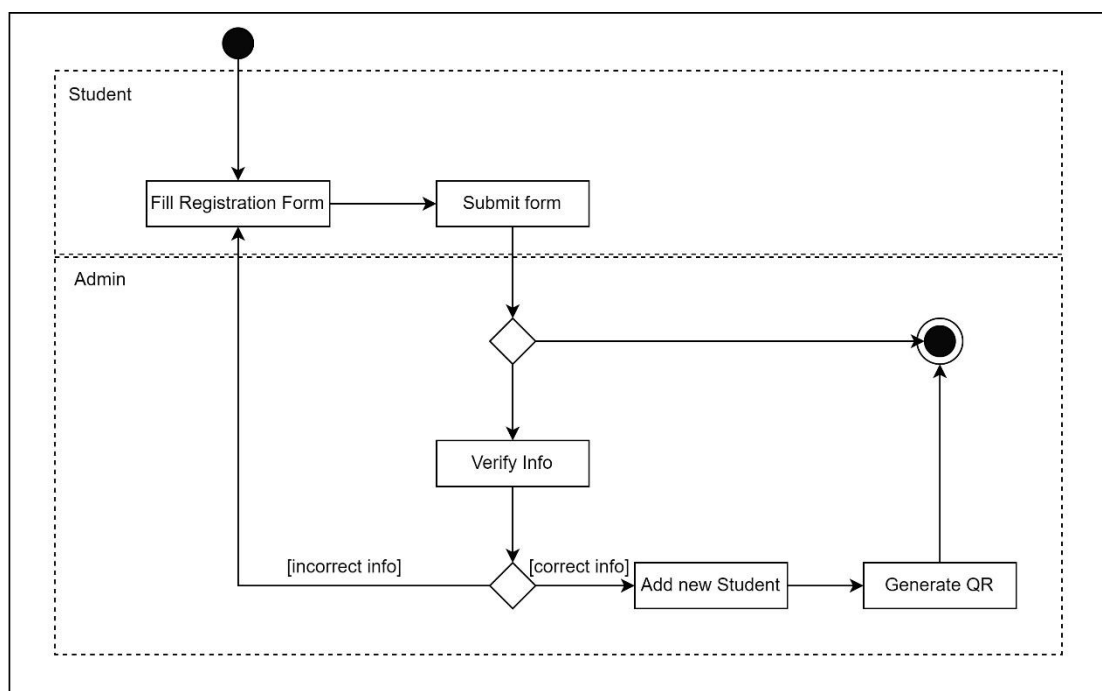
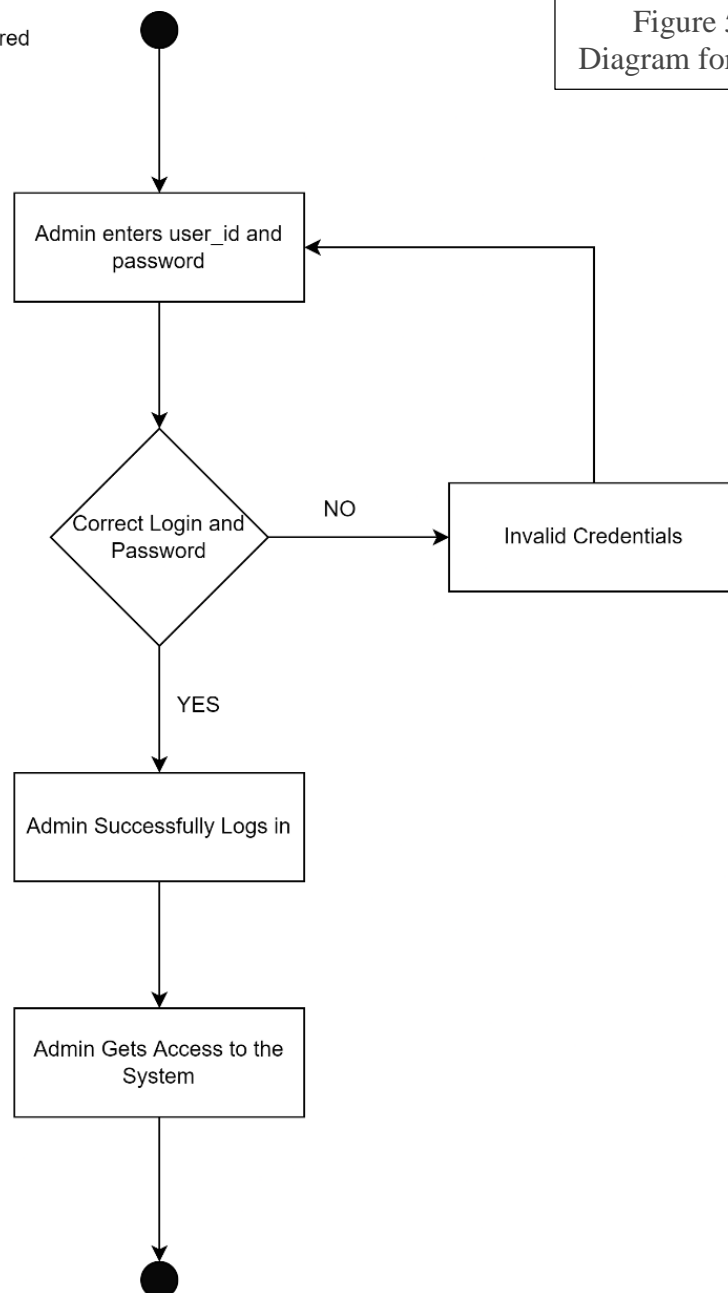


Figure 5: Activity Diagram for Adding new student

Admin already Registered

Figure 5: Activity Diagram for Admin Login



### Conclusion: -

In this experiment we learned about the different components of activity diagram and state chart diagram and how these can be used to represent the dynamic nature of an information system.

## Experiment No: - 7

**Aim: -** Modeling UML Class Diagrams and Sequence diagrams

**Theory: -**

### Introduction

Classes are the structural units in object oriented system design approach, so it is essential to know all the relationships that exist between the classes, in a system. All objects in a system are also interacting to each other by means of passing messages from one object to another. Sequence diagram shows these interactions with time ordering of the messages.

### Class diagram

It is a graphical representation for describing a system in context of its static construction.

### Sequence diagram

It represents the behavioral aspects of a system. Sequence diagram shows the interactions between the objects by means of passing messages from one object to another with respect to time in a system.

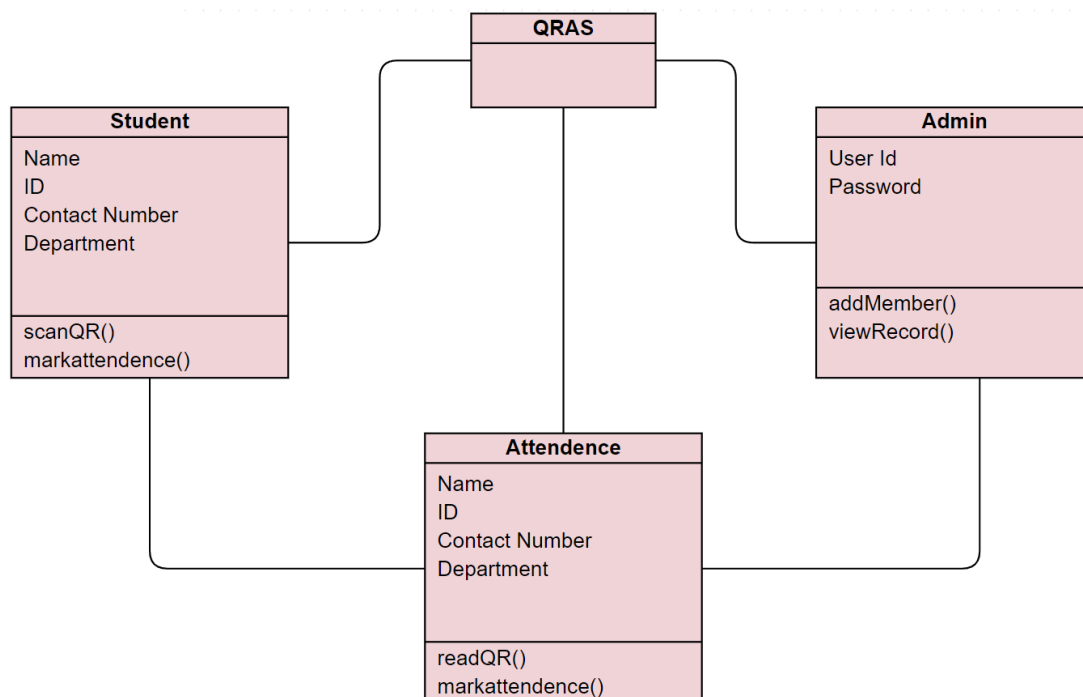


Figure 6: UML Class Diagram for QRAS

Take the "Marking Attendance" use case and represent the involved steps in a sequence diagram. We assume that the student requests for QR and admin generates QR which is then made available for student. A student scans the QR using this system. This is shown by the "QR\_Scan()". At this point the system checks whether that student is registered by verifying the QR. A new transaction is saved corresponding to the student's attendance. Finally, a success message is sent back to "Student" indicating that the attendance is successfully marked.

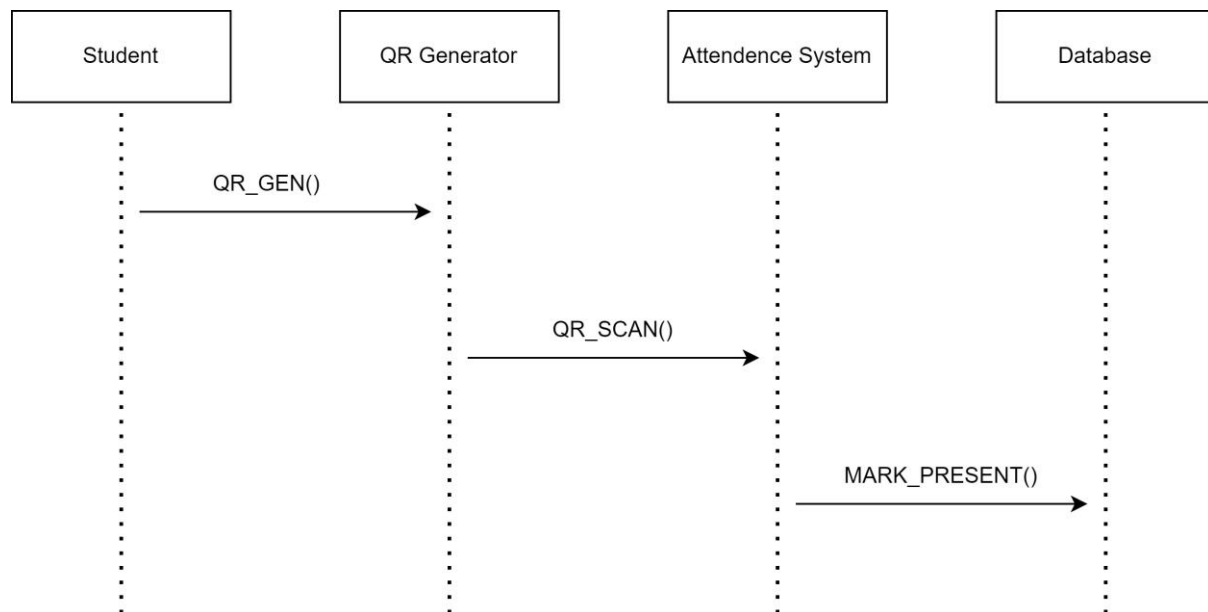


Figure 5: Sequence Diagram for marking attendance

### Conclusion: -

In this experiment we learned about the representation of class diagram and sequence diagram. We also learn about different relationships that exist among the classes, in a system.

## Experiment No: - 8

**Aim: -** Modeling Data Flow Diagrams

**Theory: -**

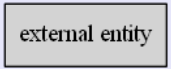


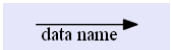
### Introduction

Information Systems (IS) help in managing and updating the vast business-related information. Before designing such an IS, it is helpful to identify the various stakeholders, and the information that they would be exchanging with the system. An IS, however, is a large software comprised of several modules, which, in turn, share the process the available data. These data are often stored in databases for further references. A Data Flow Diagram (DFD) is used to pictorially represent the functionalities of the ISs by focusing on the sources and destinations of the data flowing in the system.

### Data Flow Diagram

DFD provides the functional overview of a system. The graphical representation easily overcomes any gap between 'user and system analyst' and 'analyst and system designer' in understanding a system. Starting from an overview of the system it explores detailed design of a system through a hierarchy. DFD shows the external entities from which data flows into the process and also the other flows of data within a system. It also includes the transformations of data flow by the process and the data stores to read or write a data.

### Graphical notations for Data Flow Diagram

Term	Notation	Remarks
External entity		Name of the external entity is written inside the rectangle
Process		Name of the process is written inside the circle
Data store		A left-right open rectangle is denoted as data store; name of the data store is written inside the shape
Data flow		Data flow is represented by a directed arc with its data name

### Explanation of Symbols used in DFD

- **Process:** Processes are represented by circle. The name of the process is written into the circle. The name of the process is usually given in such a way that represents the functionality of the process.
- **External entity:** External entities are only appear in context diagram. External entities are represented by a rectangle and the name of the external entity is written into the shape. These send data to be processed and again receive the processed data.
- **Data store:** Data stores are represented by a left-right open rectangle. Name of the data store is written in between two horizontal lines of the open rectangle. Data stores are used as repositories from which data can be flown in or flown out to or from a process.
- **Data flow:** Data flows are shown as a directed edge between two components of a Data Flow Diagram. Data can flow from external entity to process, data store to process, in between two processes and vice-versa.

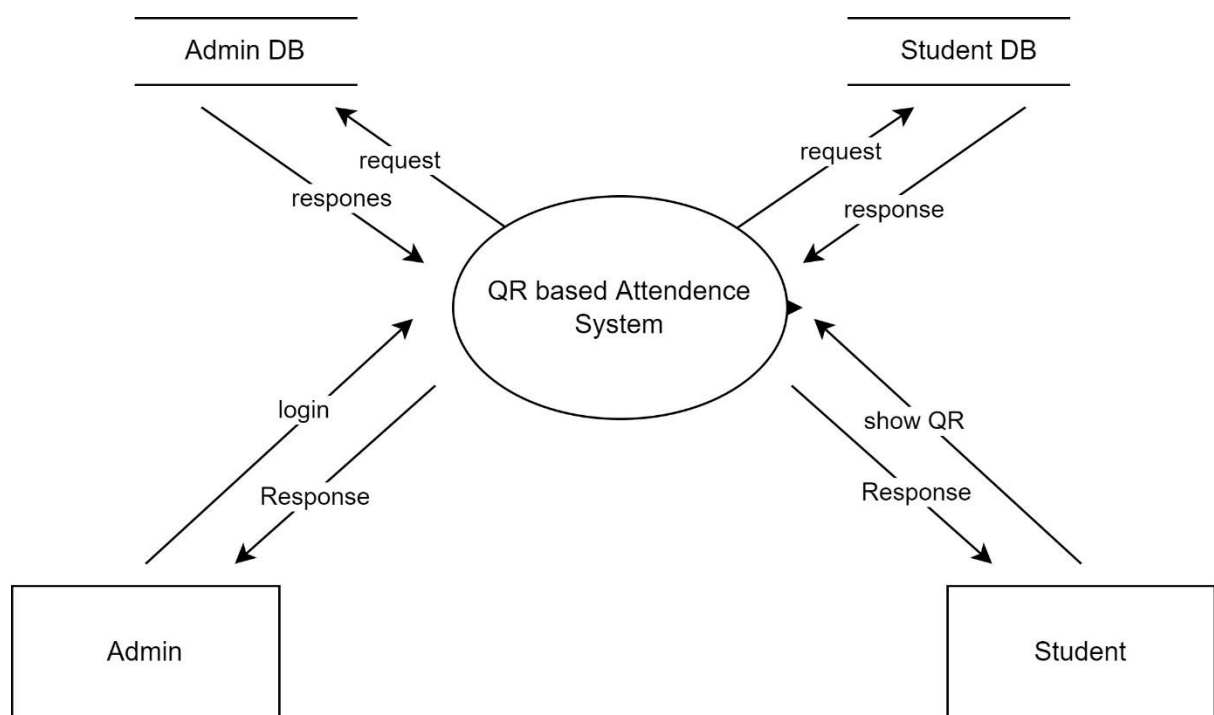


Figure 7: DataFlow Diagram

### Conclusion: -

In this experiment we learned about the representation of data flow diagram.