PAYROLL MANAGEMENT SYSTEM

UML PROJECT REPORT

18CSC202J/ 18AIC203J - OBJECT ORIENTED DESIGN AND PROGRAMMING LABORATORY

(2018 Regulation)

II Year/ III Semester

Academic Year: 2022 -2023

By

SHRAVAN CHOWDARY (RA2111026010237)

RATHNA SEKHAR MAKKENA(RA2111026010264)

Under the guidance of

Dr. M . Uma Associate Professor Department of Computational Intelligence

FACULTY OF ENGINEERING AND TECHNOLOGY SCHOOL OF COMPUTING SRM INSTITUTE OF SCIENCE AND TECHNOLOGY Kattankulathur, Kancheepuram NOVEMBER 2022

BONAFIDE

This is to certify that **18CSC202J - OBJECT ORIENTED DESIGN AND PROGRAMMING LABORATORY project report** titled "**PAYROLL MANAGEMENT SYSTEM"** is the bonafide work of

SHRAVAN CHOWDARY (RA2111026010237) RATHNA SEKHAR MAKKENA(RA2111026010264)

who undertook the task of completing the project within the allotted time.

Signature of the Guide	**Signature of the II Year Academic Advisor**
Dr. M. Uma	----------
Assistant Professor	**Professor and Head**
Department of CINTEL,	Department of CINTEL
SRM Institute of Science and Technology	SRM Institute of Science and Technology

About the course:-

18CSC202J/ 8AIC203J - Object Oriented Design and Programming are 4 credit courses with **L T P C as 3-02-4** (Tutorial modified as Practical from 2018 Curriculum onwards)

Objectives:

The student should be made to:

- Learn the basics of OOP concepts in C++

- Learn the basics of OOP analysis and design skills.
- Be exposed to the UML design diagrams.
- Be familiar with the various testing techniques

Course Learning Rationale (CLR): The purpose of learning this course is to:

1. Utilize class and build domain model for real-time programs
2. Utilize method overloading and operator overloading for real-time application development programs
3. Utilize inline, friend and virtual functions and create application development programs
4. Utilize exceptional handling and collections for real-time object-oriented programming applications
5. Construct UML component diagram and deployment diagram for design of applications
6. Create programs using object-oriented approach and design methodologies for real-time application development

Course Learning Outcomes (CLO): At the end of this course, learners will be able to:

1. Identify the class and build domain model
2. Construct programs using method overloading and operator overloading
3. Create programs using inline, friend and virtual functions, construct programs using standard templates
4. Construct programs using exceptional handling and collections
5. Create UML component diagram and deployment diagram
6. Create programs using object oriented approach and design methodologies

Table 1: Rubrics for Laboratory Exercises

(Internal Mark Splitup:- As per Curriculum)

CLAP-1	5=(2(E-lab Completion) + 2(Simple Exercises)(from CodeZinger, and any other coding platform) + 1(HackerRank/Code chef/LeetCode Weekend Challenge)	Elab test
CLAP-2	7.5=(2.0(E-lab Completion)+ 2.0 (Simple Exercises)(from CodeZinger, and any other coding platform) + 3.5 (HackerRank/Code chef/LeetCode Weekend Challenge)	Elab test
CLAP-3	7.5=(2.0(E-lab Completion(80 Pgms)+ 2.0 (Simple Exercises)(from CodeZinger, and any other coding platform) + 3.5 (HackerRank/Code chef/LeetCode Weekend Challenge)	**2 Mark -** E-lab Completion **80 Program** Completion from 10 Session (Each session min 8 program) **2 Mark -** Code to UML conversion GCR Exercises **3.5 Mark - Hacker Rank** Coding challenge completion
CLAP-4	5= 3 (Model Practical) + 2(Oral Viva)	• **3 Mark** – Model Test • **2 Mark** – Oral Viva
Total	25	

COURSE ASSESSMENT PLAN FOR OODP LAB

S.No	List of Experiments	Course Learning Outcomes (CLO)	Blooms Level	PI	No of Programs in each session
1.	Implementation of I/O Operations in C++	CLO-1	Understand	2.8.1	10
2.	Implementation of Classes and Objects in C++	CLO-1	Apply	2.6.1	10
3.	To develop a problem statement. 1. From the problem statement, Identify Use Cases and develop the Use Case model. 2. From the problem statement, Identify the conceptual classes and develop a domain model with a UML Class diagram.	CLO-1	Analysis	4.6.1	Mini Project Given
4.	Implementation of Constructor Overloading and Method Overloading in C++	CLO-2	Apply	2.6.1	10
5.	Implementation of Operator Overloading in C++	CLO-2	Apply	2.6.1	10
6.	Using the identified scenarios, find the interaction between objects and represent them using UML Sequence diagrams and Collaboration diagrams	CLO-2	Analysis	4.6.1	Mini Project Given
7.	Implementation of Inheritance concepts in C++	CLO-3	Apply	2.6.1	10
8.	Implementation of Virtual function & interface concepts in C++	CLO-3	Apply	2.6.1	10
9.	Using the identified scenarios in your project, draw relevant state charts and activity diagrams.	CLO-3	Analysis	4.6.1	Mini Project Given
10.	Implementation of Templates in C++	CLO-3	Apply	2.6.1	10
11.	Implementation of Exception of Handling in C++	CLO-4	Apply	2.6.1	10
12.	Identify the User Interface, Domain objects, and Technical Services. Draw the partial layered, logical architecture diagram with UML package diagram notation such as Component Diagram, Deployment Diagram.	CLO-5	Analysis	4.6.1	Mini Project Given

13.	Implementation of STL Containers in C++	CLO-6	Apply	2.6.1	10
14.	Implementation of STL associate containers and algorithms in C++	CLO-6	Apply	2.6.1	10
15.	Implementation of Streams and File Handling in C++	CLO-6	Apply	2.6.1	10

LIST OF EXPERIMNENTS FOR UML DESIGN AND MODELLING:

To develop a mini-project by following the exercises listed below.

1. To develop a problem statement.

2. Identify Use Cases and develop the Use Case model.

3. Identify the conceptual classes and develop a domain model with UML Class diagram.

4. Using the identified scenarios, find the interaction between objects and represent them using UML Sequence diagrams.

5. Draw relevant state charts and activity diagrams.

6. Identify the User Interface, Domain objects, and Technical services. Draw the partial layered, logical architecture diagram with UML package diagram notation.

Suggested Software Tools for UML:

StarUML, Rational Suite, Argo UML (or) equivalent, Eclipse IDE and Junit

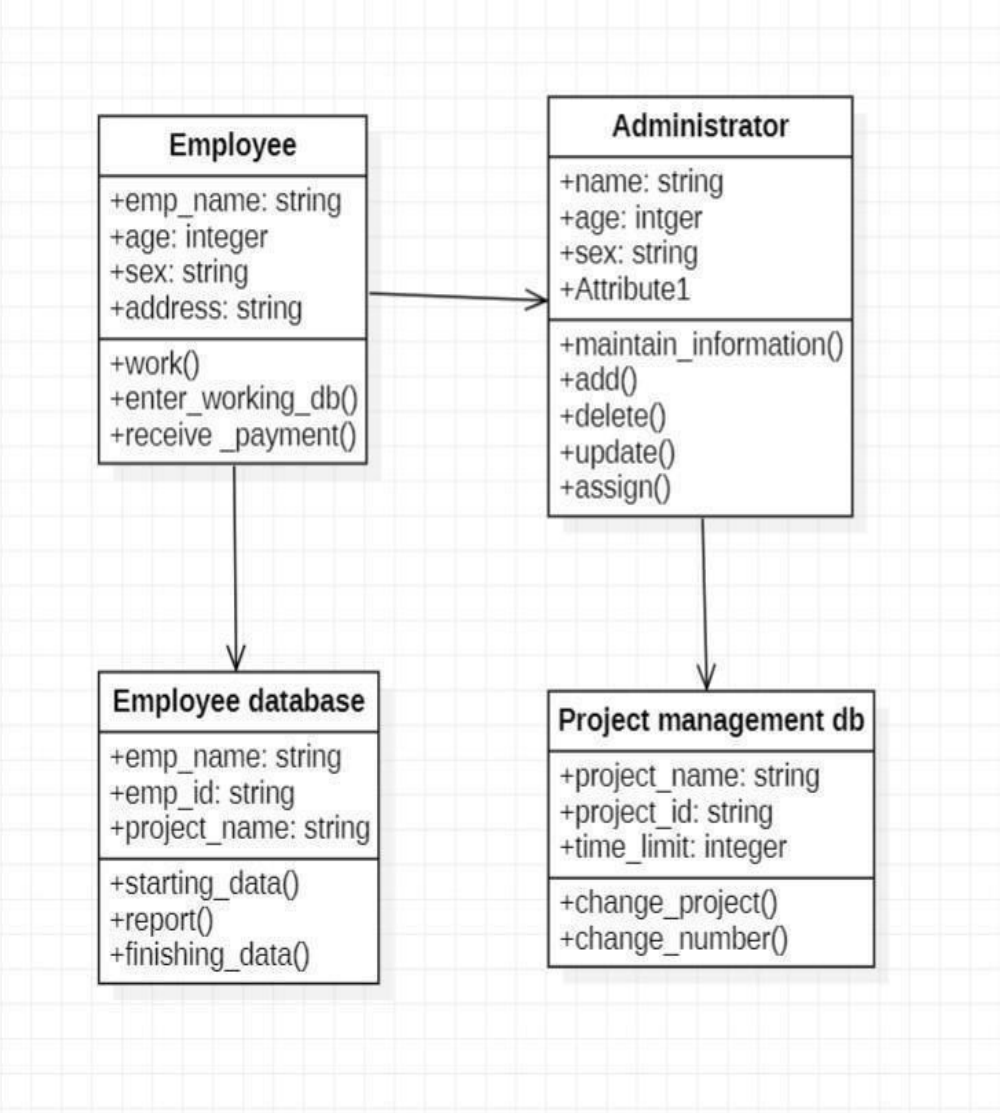
ABSTRACT

Payroll Management System UML Diagrams are diagrams formed to design the structure and behavior of the project. These UML Diagrams were used to visualize the possible users, processes, activities and sequence of events in Payroll Management System. Its purpose is to analyze, understand and know the needs of the project before developing it.

UML Diagrams are part of project documentation that represents the overall Payroll Management System function. Each of these diagrams shows different visualization of Payroll Management System structures and behavior towards data handling and interaction with the targeted users. The Structural UML Diagrams represents the rightful structure and features of the Payroll Management System. On the other hand, the Behavioral UML Diagrams illustrates the System's behavior towards the users and data handling.

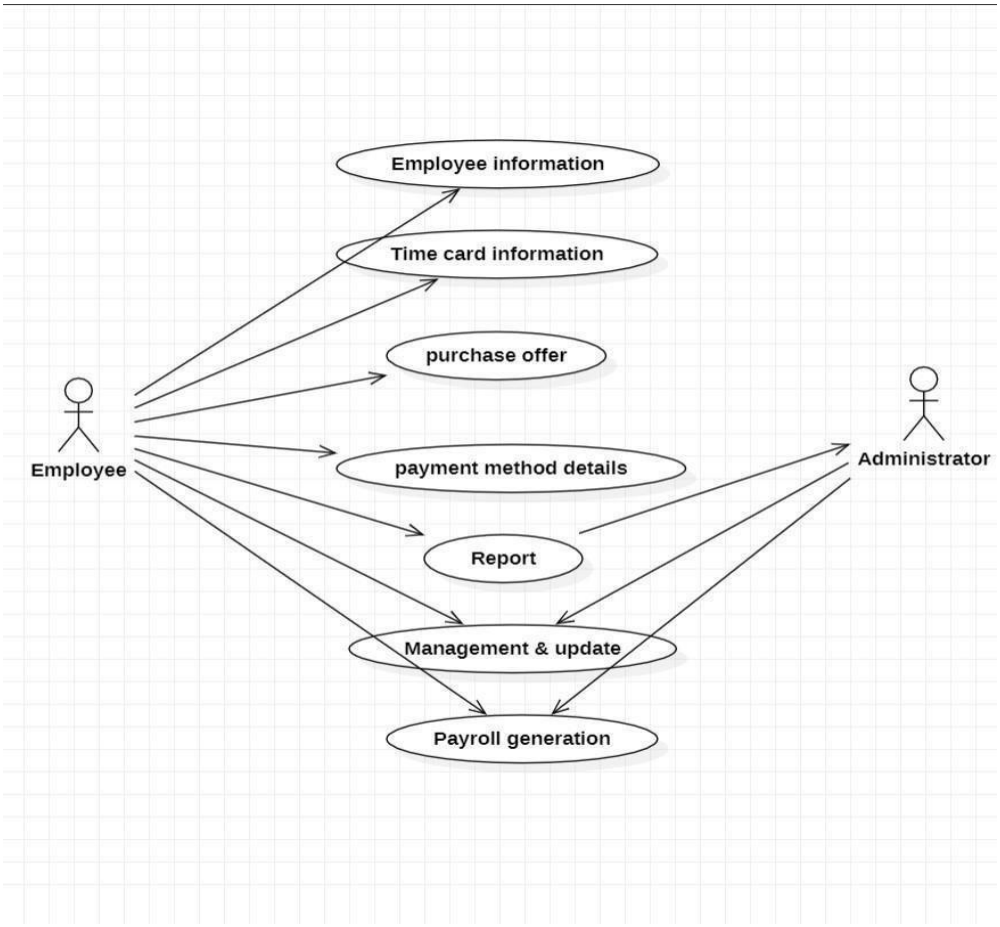
MODULE DESCRIPTION

Class diagram with explanation



Payroll Management System Class Diagram the Class diagram for Payroll Management System shows the structures of information or data that will be handled in the system. These data or information will be represented by classes. Each of the classes will have their attributes in accord to the methods they will use. So the UML Class diagram was illustrated by a box with 3 partitions and the upper part was the name of the class, the middles are the attributes and the bottom is for the methods. The arrows on them represents their relationships in each other.

Use case diagram with explanation



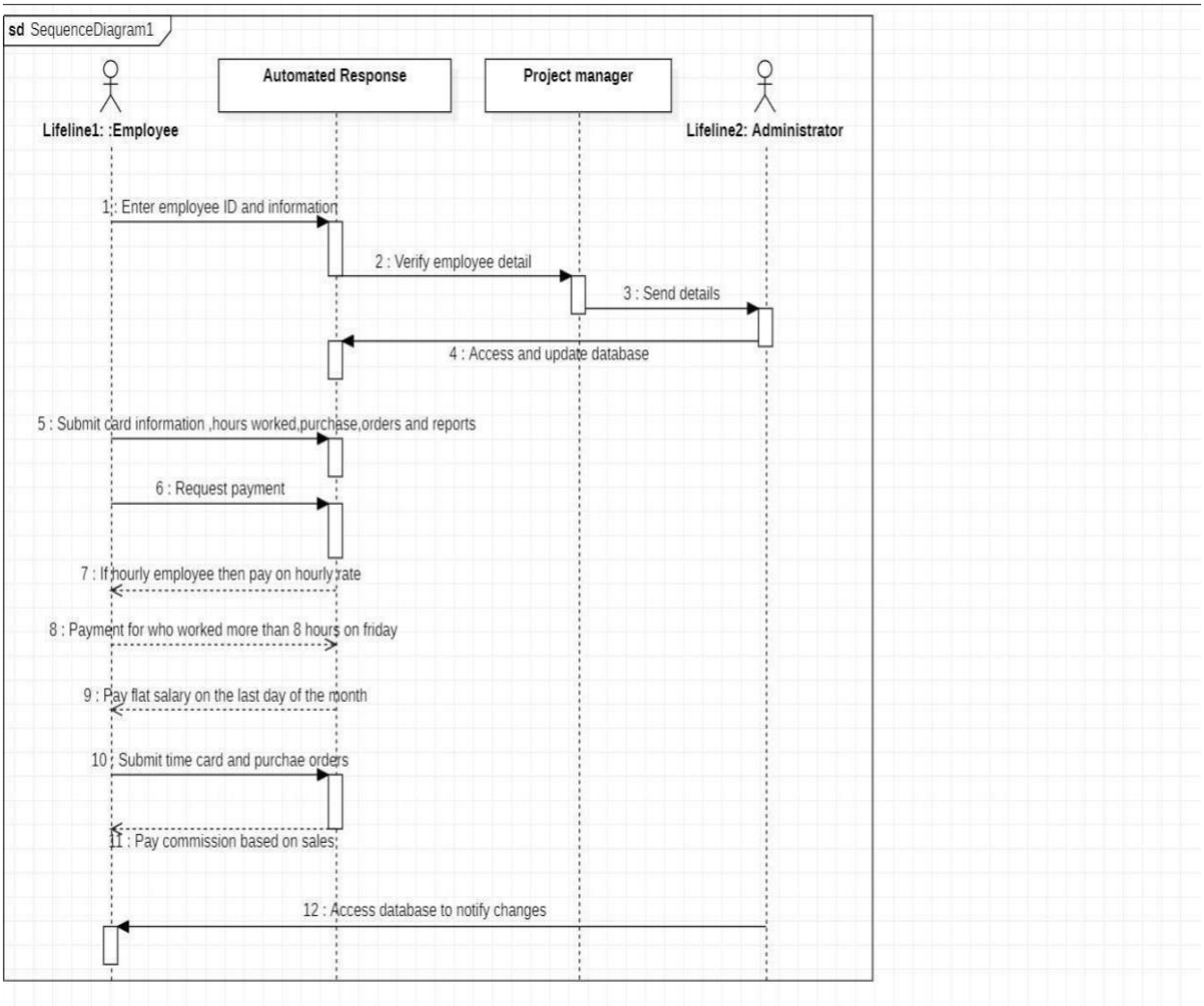
Unified Modeling Language (UML), a use case diagram can summarize the details of your system's users (also known as actors) and their interactions with the system. To build one, you'll use a set of specialized symbols and connectors.

- Scenarios in which your system or application interacts with people, organizations, or external systems
- Goals that your system or application helps those entities (known as actors) achieve
- The scope of your system

UML use case diagrams are ideal for:

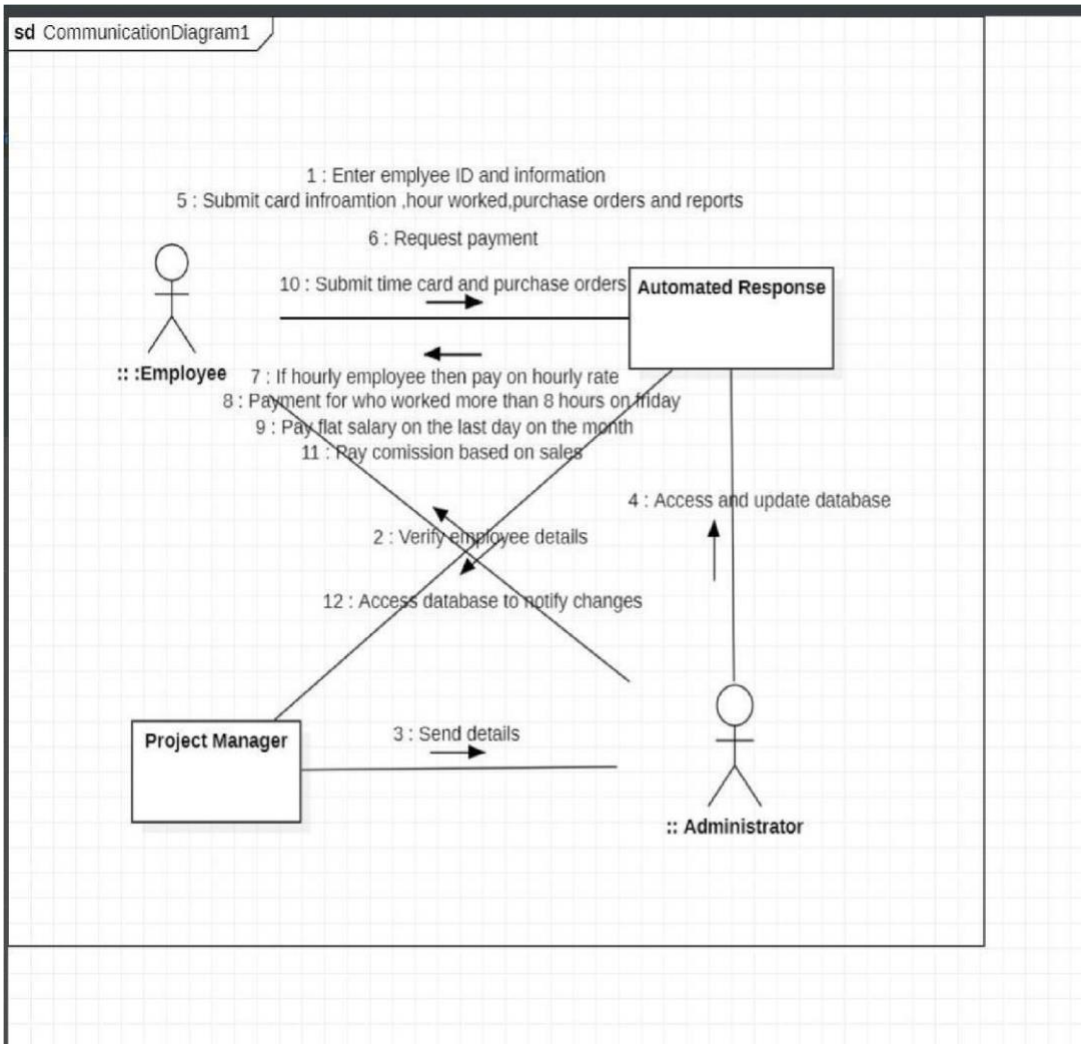
- Representing the goals of system-user interactions
- Defining and organizing functional requirements in a system
- Specifying the context and requirements of a system
- Modeling the basic flow of events in a use case

Sequence diagram with explanation



Payroll Management System Sequence Diagram the designed sequence diagram illustrates the series of events that occurs in Payroll Management System. In this illustration, the actors are represented by a stick man and the transactions or classes are represented by objects. It will give you clear explanation about the behavior of an Payroll Management System in terms of processing the flow of instructions. This designed sequence diagram is able to show programmers and readers about the sequence of messages between the actor and the object

Communication diagram with explanation



The collaboration diagram is used to show the relationship between the objects in a system. Both the sequence and the collaboration diagrams represent the same information but differently. Instead of showing the flow of messages, it depicts the architecture of the object residing in the system as it is based on object-oriented programming. An object consists of several features. Multiple objects present in the system are connected to each other. The collaboration diagram, which is also known as a communication diagram, is used to portray the object's architecture in the system.

Following are the components of a component diagram that are enlisted below:

1. **Objects:** The representation of an object is done by an object symbol with its name and class underlined, separated by a colon.

   In the collaboration diagram, objects are utilized in the following ways:

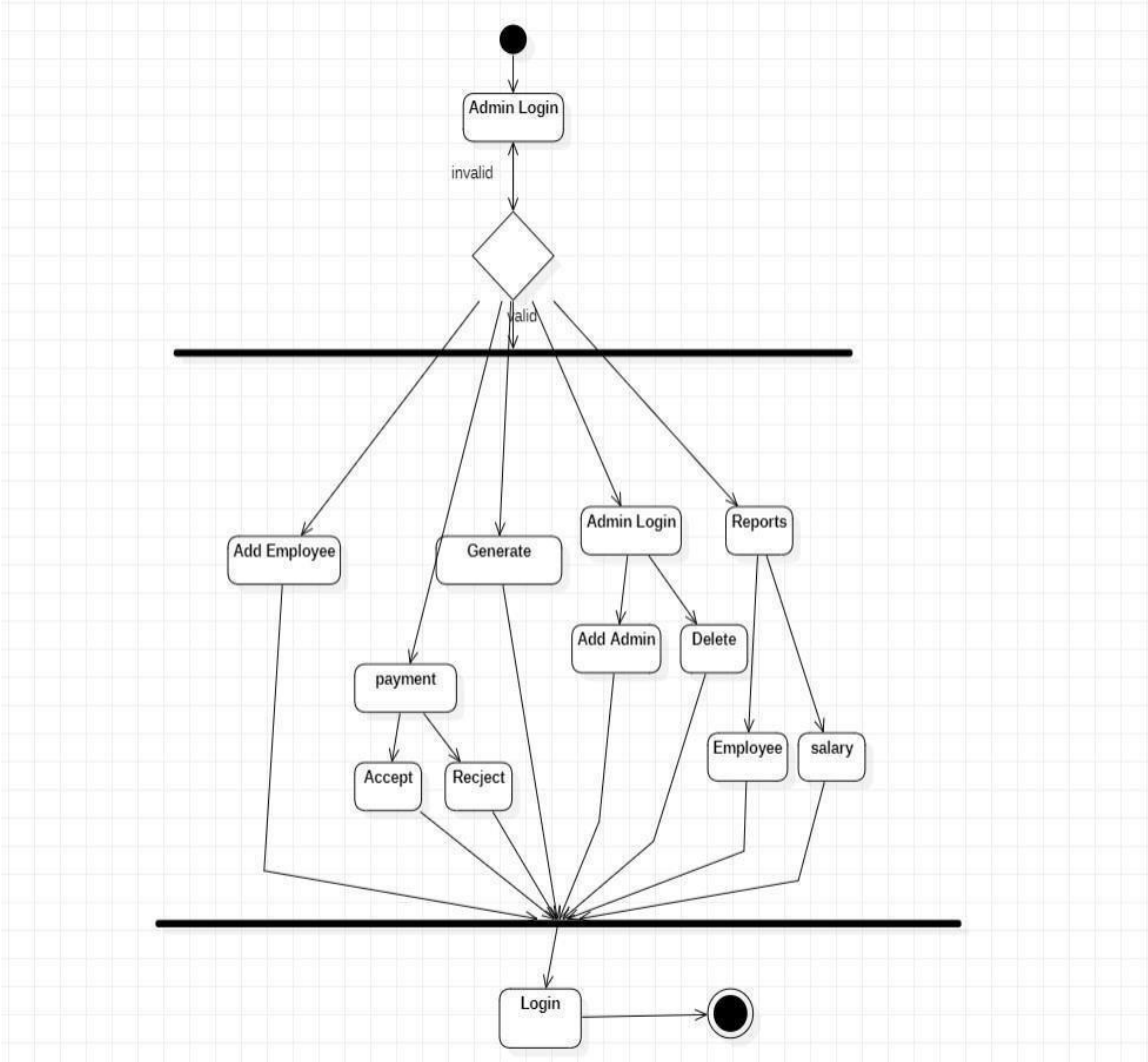
   - The object is represented by specifying their name and class. o It is not mandatory for every class to appear. o A class may constitute more than one object.
   - In the collaboration diagram, firstly, the object is created, and then its class is specified. o To differentiate one object from another object, it is necessary to name them.

2. **Actors:** In the collaboration diagram, the actor plays the main role as it invokes the interaction. Each actor has its respective role and name. In this, one actor initiates the use case.

3. **Links:** The link is an instance of association, which associates the objects and actors. It portrays a relationship between the objects through which the messages are sent. It is represented by a solid line. The link helps an object to connect with or navigate to another object, such that the message flows are attached to links.

4. **Messages:** It is a communication between objects which carries information and includes a sequence number, so that the activity may take place. It is represented by a labeled arrow, which is placed near a link. The messages are sent from the sender to the receiver, and the direction must be navigable in that particular direction. The receiver must understand the message.

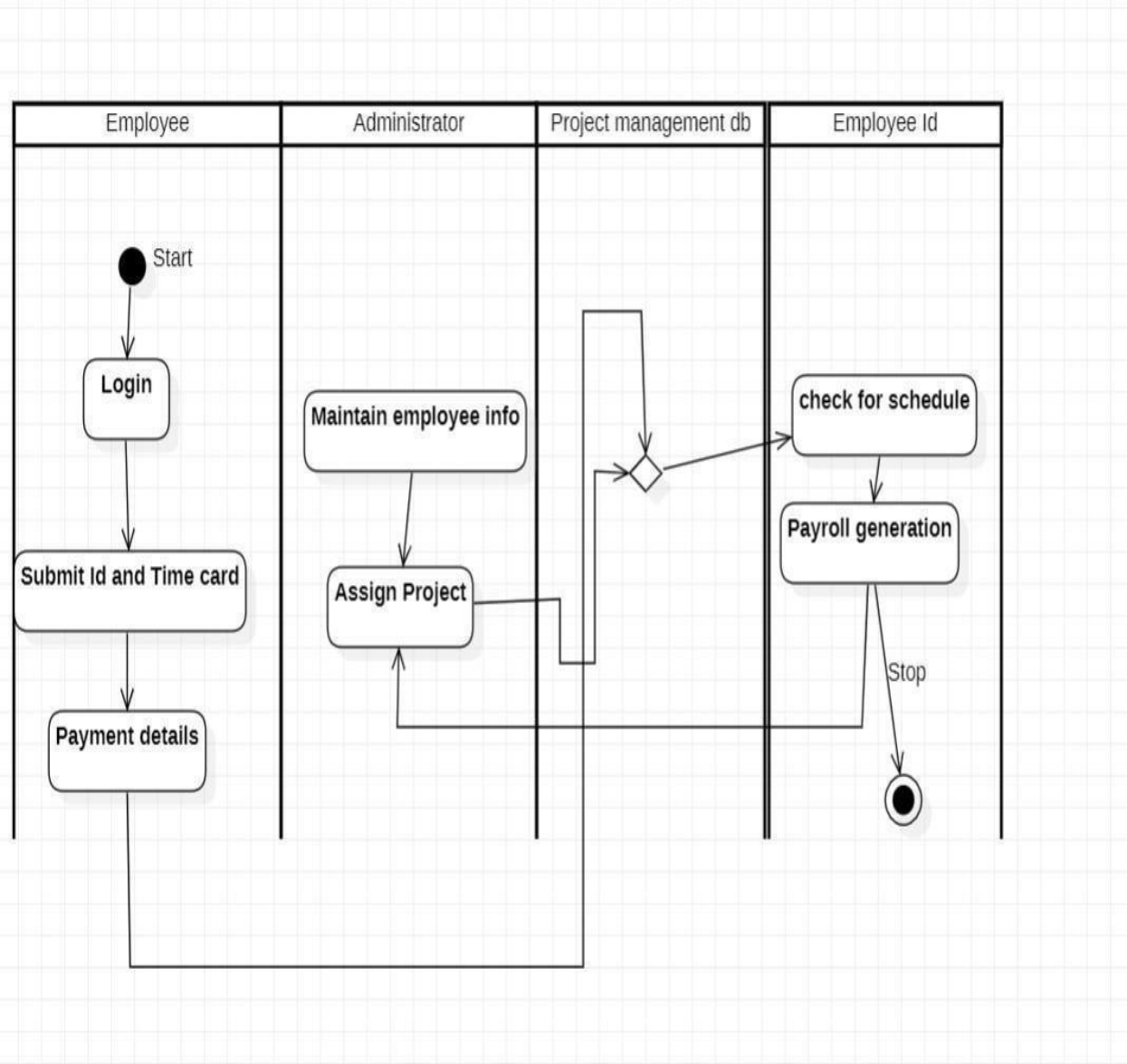
State chart diagram with explanation



A state chart diagram models the behaviour of a single object, specifying the sequence of events that an object goes through during its lifetime in response to events.

- A state is denoted by a round-cornered rectangle with the name of the state written inside it.
- The initial state is denoted by a filled black circle and may be labeled with a name. The final state is denoted by a circle with a dot inside and may also be labeled with a name.
- Transitions from one state to the next are denoted by lines with arrowheads. A transition may have a trigger, a guard and an effect, as below.
- State Actions can be done by defining an entry action for the state. The diagram below shows a state with an entry action and an exit action.
- A state can have a transition that returns to itself, as in the following diagram. This is most useful when an effect is associated with the transition.

Activity diagram with explanation



Activity diagrams present a number of benefits to users. Consider creating an activity diagram to:

- Demonstrate the logic of an algorithm.
- Describe the steps performed in a UML use case.
- Illustrate a business process or workflow between users and the system.
- Simplify and improve any process by clarifying complicated use cases.
- Model software architecture elements, such as method, function, and operation.

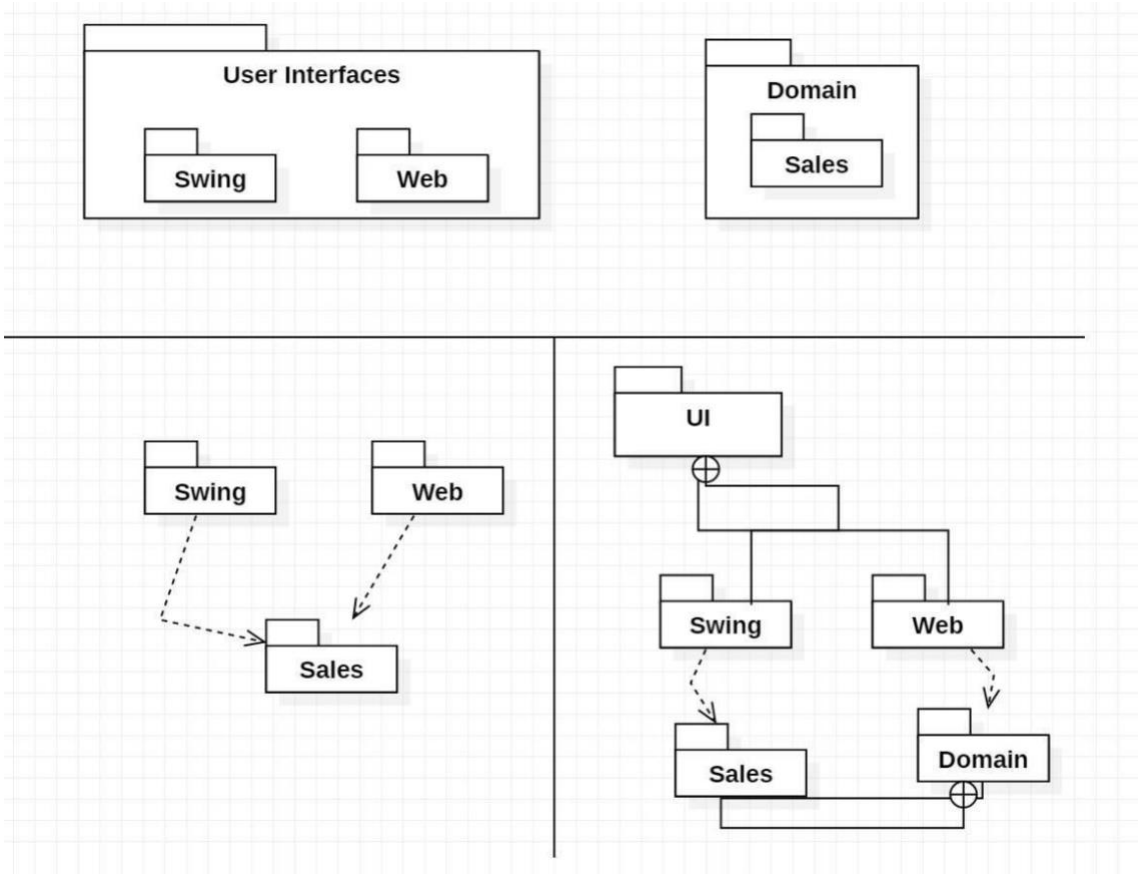
Basic components of an activity diagram

Before you begin making an activity diagram, you should first understand its makeup. Some of the most common components of an activity diagram include:

- Action: A step in the activity wherein the users or software perform a given task. In Lucidchart, actions are symbolized with round-edged rectangles.
- Decision node: A conditional branch in the flow that is represented by a diamond. It includes a single input and two or more outputs.
- Control flows: Another name for the connectors that show the flow between steps in the diagram.
- Start node: Symbolizes the beginning of the activity. The start node is represented by a black circle.

End node: Represents the final step in the activity. The end node is represented by an outlined black circle

Package diagram with explanation



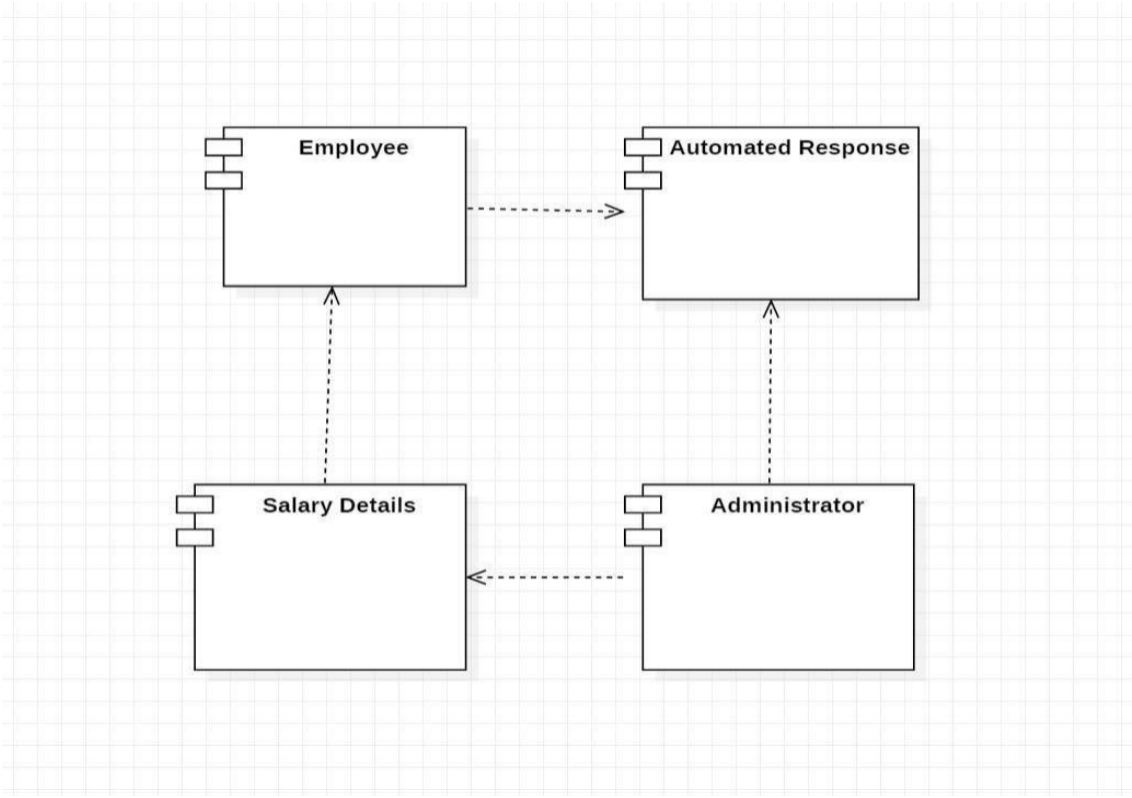
- Layers of an architecture represent the vertical slices, while partitions represent a horizontal division of relatively parallel subsystems of a layer.

- Model is a synonym for the domain layer of objects (it's an old OO term from the late 1970s).

- View is a synonym for UI objects, such as windows, Web pages, applets, and reports.

- Model (domain) objects should not have direct knowledge of view (UI) objects.

Component diagram with explanation



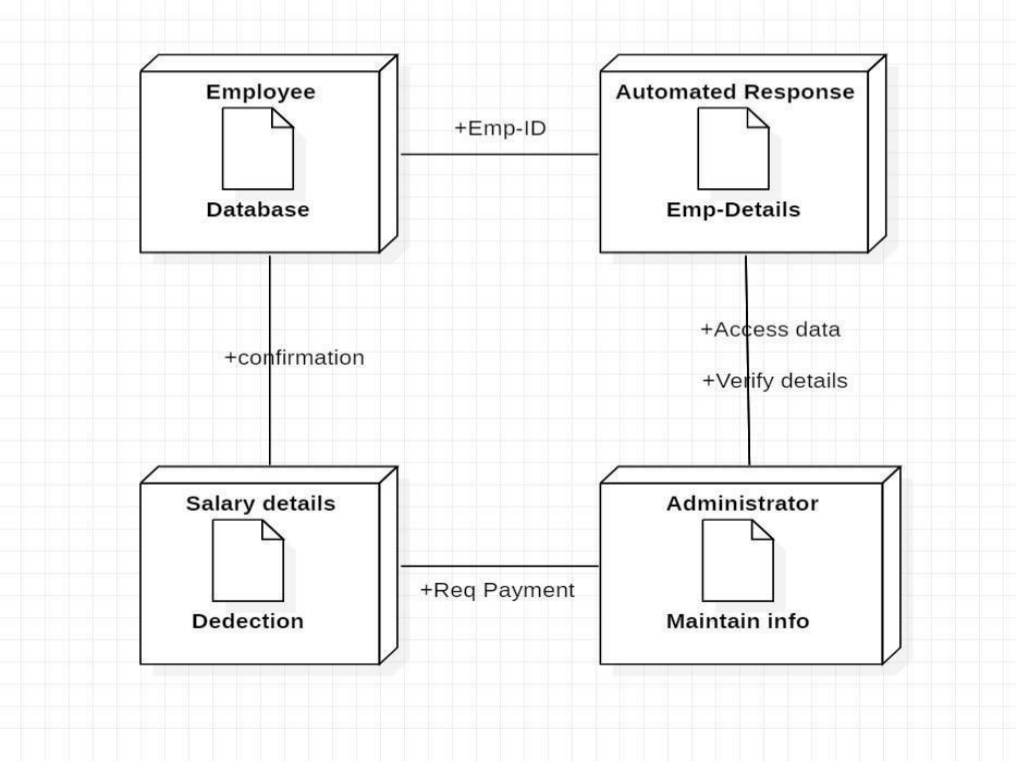
Component diagram for payroll management system is used to show how the system's parts work together to make the desired payroll operate correctly. This component diagram shows how the payroll components are organized and how they depend on each other. It gives a high-level look at the parts of a system.

- The potential components of a payroll management system can be part of software or hardware. They could be a database, a user interface, or something else that helps the payroll management system work.

- The UML component diagram shows how a payroll management system will be made up of a set of deployable components, such as dynamic-link library (DLL) files, executable files, or web services. Using well-defined interfaces, these parts communicate with each other and keep their internal details hidden from each other and the outside world.

- The component symbol is used for a person or thing that needs to do a stereotype function. It gives and takes behavior through interfaces, as well as through other parts. Components can be grouped into a node or another component and can be only one component.

Deployment diagram with explanation



Deployment diagram for payroll management system is used to represent the system's physical architecture. This UML deployment diagram shows the relationships between software and hardware components as well as the physical distribution of processes.

- Payroll Management System UML deployment diagram explains the sketch of the relationship between software and hardware. These hardware and software are labeled to clarify their part in the system's operation. They were represented by nodes and the connections were represented by labeled arrows.

- The nodes are the payroll management system, the Employee ,the Automated Response , the Salary Details , the Administrator**.** The system server node contains a developed database that will hold the details of the system online.

- For the connection, the system is connected to the server database using a private network which enables it to pass a connection to the devices and enable users to access the system and database. The employer and the employees then can communicate through the system.

Conclusion

As a whole, the UML Diagrams works together to achieve the most desired functions of the Payroll Management System Project. All of these were designed to guide programmers and beginners about the behavior and structure of Payroll Management System. By completing all the given Diagrams, the Payroll Management Project System development would be much easier and attainable. So those UML diagrams were given to teach you and guide you through your project development journey. You can use all of the given UML diagrams as your reference, or have them for your project development. The ideas presented in UML Diagrams were all based on Payroll Management System requirements.

References

https://www.coursehero.com/file/p6flr8p/11-Conclusion-Payroll-Management-System-softwareware-developed-for-a-company