PAYROLL MANGER

A
Case Study Report
Submitted in partial fulfillment of the
Requirements for the Course of

SOFTWARE ENGINEERING LAB

IN

BE 2/4 (IT) IV-SEMESTER

By

T.Devi Srujana

1602-20-737-011

Department of Information Technology

Vasavi College of Engineering (Autonomous)

ACCREDITED BY NAAC WITH 'A++' GRADE

(Affiliated to Osmania University and Approved by AICTE)

Ibrahimbagh, Hyderabad-31

2022

Vasavi College of Engineering (Autonomous)

ACCREDITED BY NAAC WITH 'A++' GRADE

(Affiliated to Osmania University and Approved by AICTE)

Ibrahimbagh, Hyderabad-31

Department of Information Technology

DECLARATION BY THE CANDIDATE

I, **T.Devi srujana,** bearing hall ticket number, **1602-20-737-011,** hereby declare that the Case study report entitled **“Payroll manger”** under theguidance of **Ms. Sree Laxmi,** Assistant Professor, Department of Information Technology, VCE, Hyderabad is submitted in partial fulfillment of the requirement for the course of **Software Engineering - Lab** in BE 2/4 (IT) IV-Semester.

This is a record of bonafide work carried out by me and the Design embodied in this project report has not been submitted by any other

ABSTRACT

“Employee Database And Payroll Manager” is designed to make the existing manual system automatic with the help of computerised equipment and full-edged computer software, fulfilling their requirements, so that their valuable data and information can be stored for a longer period with easy access and manipulation of the same. The required software is easily available and easy to work with. This web application can maintain and view computerised records without getting redundant entries. The project describes how to manage user data for good performance and provide better services for the client.

Software Requirements Specification
Introduction

Purpose

The mainaim of developing a payroll manager is to find a efficient way to manage payment and automate functionalities involving leaves, payroll for employees and a detailes report about employee leaves and salary pakage. This is like a tool to manage inner operations of Company related to employee leaves and payroll.

Intended Audience and Reading Suggestions

The software requirement specification (SRS) document is written for a general audience, this document is meant for individuals directly concerned in the usage and implementation of Payroll Manager. This includes software developers, project consultants, and team managers. This document need not be read sequentially; users can leap to any section they find relevant.

Product Scope

This Application works in Multiple PC's installed on multiple Computers but sharing same database by which users of different department can use it sitting at different locations simultaneously. But in future we can make the Application where the database will be hosted in order to manage the all departments which will be located in different places and by keeping domain of Application as Online.

Overall Description

Product Perspective

This software is developed to cater the company employees leave management and it is totally self contained and works efficiently. It provides simple database and good and easy graphical user interface to both new as well as experienced user of the computer.

Product Functions

2.2.1 Master module

Designation : Contains the position or status of employee in departments.

Department : Contains the information about different departments in any company.

2.2.2 Employee module

Employee details :This module contains the whole detail of employees of any system.

2.2.3 Attendance module

Leave : This module is for keeping the records of leave taken by any employee.

Attendance : This module is for keeping the records of employee's presence.

2.2.4 Salary module

Allowance : This module is for calculating the allowance given to employee by the company.

Deduction : This module calculates the amount from number of days taken as leaves and deduce these amount from salary.

Pay Slip : This module is for generating the final pay slip.

User Classes and Characteristics

2.3.1 End Users

End user should have basic idea about computer operations and database.

2.3.2 Administrator

- Administrator must be having good knowledge of database management system and the computer because administrator have to manage user rights.
- If the network connection does not work properly than our system should not work as intended and also the product is installed properly at web server.
- Recovery of data after a system crash will be possible only if backups are taken at regular intervals so backup should be there.
- Manual interfaces cannot be fully avoided. Documented proofs like data entry of employees etc. will have to be verified by the concerned management staff before entering it into the computerized system.

Operating Environment

The operating environment in which the software will work is windows xp, windows 7, windows 10, Linux, unix, macOS.

Design and Implementation Constraints

RUP will be used for the design purposes and UML will be used for the diagrams in MS Visio. Python language will be used for the development and for the database side Microsoft SQL server will be used. So, we must be familiar with these constraints.

User Documentation

A user manual will be provided along with the software and an online help will be provided by sending a quary to the specified details.

Assumptions and Dependencies

2.7.1 Assumptions:

- The code should be free with compilation errors/syntax errors.
- The product must have an interface which is simple enough to understand.

2.7.2 Dependencies:

- All necessary hardware and software are available for implementing and use of the tool.
- The proposed system would be designed, developed and implemented based on the software requirements specifications document.
- End users should have basic knowledge of computer and we also assure that the users will be given software training documentation and reference material.
- The system is not required to save generated reports.

External Interface Requirements

User Interfaces

This software provides a good graphical interface to the user.The user interface will be friendly easy for usage. The software will be interactive and self-explanatory.

Hardware Interfaces

The hardware to ensure that the software executes all the user requirements efficiently are as follows :

- Mouse
- Keyboard
- Monitor
- Printer
- Hard-disk 256 GB or more
- Ram with memory 4 GB or more
- Android device

Software Interfaces

The software used for Payroll Manager are :

Operating System	:	Windows XP, Windows 7 and above
Front End	:	Microsoft Visual Basic Net 2010
Backend	:	Microaost SQL Server 2008

Communications Interfaces

The communication architecture must follow the client-server model. A uniform interface must separate the client roles from the server roles. Communication between the client and server should utilize a REST-compliant web service and must be served over HTTP Secure (HTTPS). The client-server communication must be stateless.

System Features

Admin Login

4.1.1 Description and Priority

Admin will access the system with admin ID and password. This has high priority as admin is the role played by the manager or owner of the company and is needed to know about the employee details efficiently to manage work and allocate salary to the employee. Admin has the access to add a new employee, modify the salary package and details of the employee.

4.1.2 Stimulus/Response Sequences

Stimulus	Respoence
Log In	A new page displaying text feilds to enter ID and password in shown
View Attendance	A new page is displayed showing the working days and leaves.

Check Details	A new page is displayed with the admins personal details.
Check Salary	A new page is displayed with the admins work punctuality, leaves report and salary package
Check Employee Details	A new page is displayed with text feilds to enter employee details Then the employees personal details, work punctuality and salary package can be viewed and modified if necessary

4.1.3 Functional Requirements

Login credentials

- Admin can Enter the login details i.e. username and password and can view his/her personal details.
- Admin can view and modify every employee detail in the company.

Add Employee

- Administrator can add employee into the system, by providing employee's details i.e. ID, name, image, phoneNumber, emailAddress, address, rank, salary.

Search Employee

- Administrator can search the employee from the system, by providing employee's ID/name/phoneNumber.

Update Employee

- Administrator can update the employee's information into the system, by providing employee's details i.e. name, image, phoneNumber, emailAddress, address, rank,salary on the basis of ID.

Generate Attendance Report

- Administrator can generate attendance report by providing specific detail.

Modify Salary

- Administrator/DEO can add salary of an employee by providing salary details, i.e. ID, date, month, deduction and bonus.

Generate Salary Report

- Administrator can generate Salary report by providing specific detail.

Employee Login

4.2.1 Description and Priority

Employee will access the system through employee ID and password. This has medium priority as the information is available to the employee in read-only format. Employee can check his personal details, punctuality report, his leaves status and salary package. Any changes or modifications to be made can only be done with the admin login and can be done by the employee if given the perrmission.

4.2.2 Stimulus/Response Sequences

Stimulus	Respoence
Log In	A new page displaying text feilds to enter ID and password in shown
View Attendance	A new page is displayed showing the working days and leaves.
Check Details	A new page is displayed with the employee personal details
Check Salary	A new page is displayed with the employee work punctuality, leaves report and salary package

4.2.3 Functional Requirements

Login credentials

- Employee will be provide its login credentials to enter into the system i.e. username and password.
- Employee can enter the login details and view his/her details.

View Attendance Report

- Employee can view his/her attendance report by providing login credentials.

View salary Report

- Employee can view his/her salary details and take a print of it .

Additional Requirements

Performance Requirements

- The overall system should be fast and error free.
- It should have built in error checking and correction facilities.
- The system should be able to handle large amount of data comfortably.

Safety Requirements

To ensure safety of the details, they are backuped into a hard disk on the system and as a precaution it is better to store a hard copy of the details.

Security Requirements

- The access to the software is given only to valid operators. We need a specific ID and password to get access to the software.
- Communication needs to be restricted when the application is validating the user or licence.

Software Quality Attributes

Reliability

- In order to ensure reliability, this system is being designed using software that is established to be stable and easy to use.

Availability

- This system is designed to run 24/7 and be readily available to the user.

Business Rules

Admin :-

- only admin can add modify and view the details of every employee.
- Admin can view his/her personal details.

Employee :-

- Employee can only view his/her details

Other Requirements

The software works more efficiently in the latest version of the software so it better to update the system.

Appendix : To Be Determined List

In further development of this software addition of GPS Tracking System can be done to track the employee location in case the employee is needed to travel as specifid by his/her manager.

The software works more efficiently in the latest version of the software so it better to update the system.

UML Introduction:

The UML is a language for specifying, constructing, visualizing, and documenting the software system and its components. The UML is a graphical language with sets of rules and semantics. The rules and semantics of a model are expressed in English in a form known as OCL (Object Constraint Language). OCL uses simple logic for specifying the properties of a system. The UML is not intended to be a visual programming language. However, it has a much closer mapping to object-oriented programming languages, so that the best of both can be obtained. The UML is much simpler than other methods preceding it. UML is appropriate for modeling systems, ranging from enterprise information system to distributed web-based application and even to real time embedded system**.**

UML includes nine diagrams:

Use Case Diagram: To illustrate the user intersection with the system

Class Diagram: To illustrate logical structure.

Object Diagram: To illustrate the physical structure of the software.

Deployment Diagram: To shows the mapping of the software- hardware.

Interaction Diagram: To show the collaboration of group of objects.

Sequence & Collaboration: To illustrate behavior.

State Chart & Activity Diagram: To illustrate flow of events

Use-case Diagram: A use case diagram shows interactions of actors with a system in terms of functions called use-case. A use case is description of a functionality that the system provides. The actors are external to the system but who interacts with thesystem; they may be external persons, external system and hardware.

Class Diagram: A class diagram shows the static structure of classes in the system.

The classes represent the "things" that are handled in the system. Class can be related toeach other in a number of ways: associated, dependent, specialized or packaged. Class represents attributes and operations.

Object Diagram: An object diagram is a variant of the class diagram and uses almost identical notation. The difference between the two is that an object diagram shows a number of object instances of classes, instead of actual classes. Objects are written in rectangular box with their names. object diagrams are written in rectangular box with their names. Object diagrams are not important as class diagrams but show the actual instances of a class and the relationships.

State Diagram: A state diagram is typically a component to the description of a class. It shows all possible states that the object of the class can have, and which events causethe state to change. An event can be another object that sends a message to it. A change is called a transition. State diagrams are not drawn for all classes, only for those that have a number of well-defined states.

State diagram can also be drawn for the system as a whole.

Sequence Diagram: A sequence diagram shows a dynamic collaboration between a number of objects and shows an interaction between objects. This diagram shows a number of objects with vertical lines that represent object lifeline. Messages passing between objects are shown with arrows. This represents the scenario of functions how they apply.

Collaboration Diagram: It is similar to a sequence diagram however the relationships are only shown. If time or sequence of events is important then sequence diagrams are more relevant.

Activity Diagram: An activity diagram shows a sequential flow of activities. This is typically used to describe the activities performed in an operation.

Deployment Diagram: It represents the processors and devices to develop the system ,it shows deployment view of the system.

Component Diagram: A component diagram shows the physical structure of the code in terms of code components. A component can be a source code component, a binary component or an executable component. A component contains information about logical class or classes it implement .

USECASE DIAGRAM

DEFINITION:

Use case Diagram is a collection of use cases, actors and relationships that exists between them. Use-case diagrams graphically depict system behavior (use cases).

These diagrams present a high-level view of how the system is used as viewed from an outsider's (actor's) perspective. A use-case diagram may depict all or some of the use cases of a system.

CONTENTS:

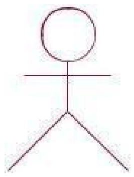
A use-case diagram can contain:

1. Actors ("things" outside the system):

Actors represent system users. They help delimit the system and give a clearer pictureof what the system should do. It is important to note that an actor interacts with but has no control over the use cases.

An actor is someone or something that:

- Interacts with or uses the system
- Provides input to and receives information from the system
- Is external to the system and has no control over the use cases



Actor

2·**Use cases** (system boundaries identifying what the system should do):

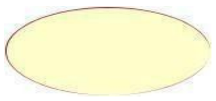
A more detailed description might characterize a use case as:

- a pattern of behavior the system exhibits
- a sequence of related transactions performed by an actor and the system
- delivering something of value to the actor.

Use cases provide a means to:

- capture system requirements
- communicate with the end users and domain experts
- test the system

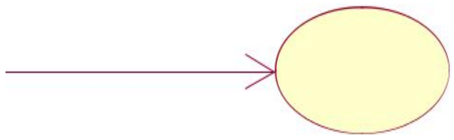
Use cases are best discovered by examining the actors and defining what the actor will be able to do with the system.



use case

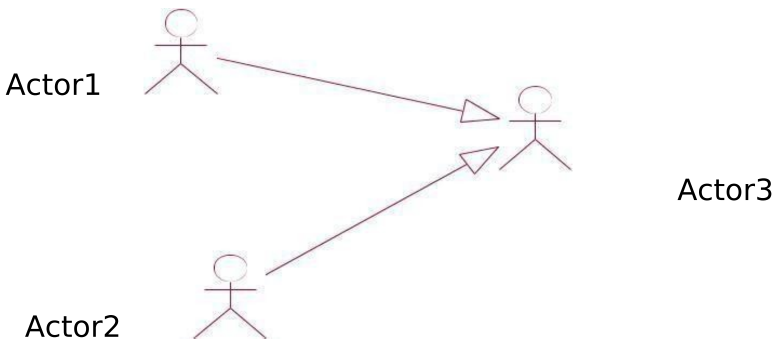
2. Interactions or relationships between actors and use cases in the system including the associations, dependencies, and generalizations.

Association Relationship: An association provides a pathway for communication. The communication can be between use cases, actors, classes or interfaces. Associations are the most general of all relationships and consequentially the most semantically weak. If two objects are usually considered independently, the relationship is an association.



Generalization Relationship:

A generalize relationship is a relationship between a more general class or use case and a more specific class or use case. A generalization is shown as a solid-line path from the more specific element to a more general element. The tip or a generalization is a large hollow triangle pointing to the more general element.



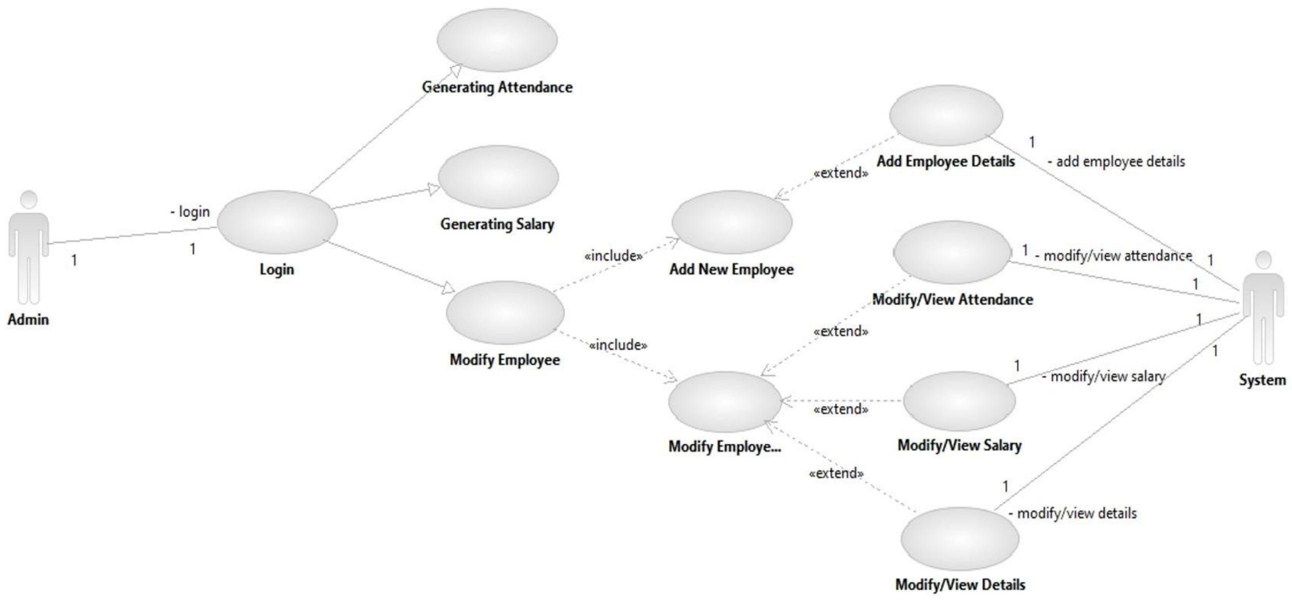
Dependency Relationship:

A dependency is a relationship between two model elements in which a change to one model element will affect the other model element. Use a dependency relationship to connect model elements with the same level of meaning. Typically, on class diagrams, a dependency relationship indicates that the operations of the client invoke operations of the supplier.

DIAGRAM:

AIM: Use Case Diagram for payroll manger

REQUIREMENTS:



CLASS DIAGRAM

DEFINITION:

A class diagram shows the existence of classes and their relationships in the logical design of a system.

A class diagram may represent all or part of the class structure of a system.

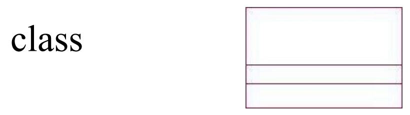
CONTENTS:

A class diagram contains

1. Classes
2. Relationships
3. Interfaces

1) Classes:

A class is a set of objects that share a common structure and common behavior (the same attributes, operations, relationships and semantics). Aclass is an abstraction of real-world items. When these items exist in the real world, they are instances of the class and are referred to as objects.

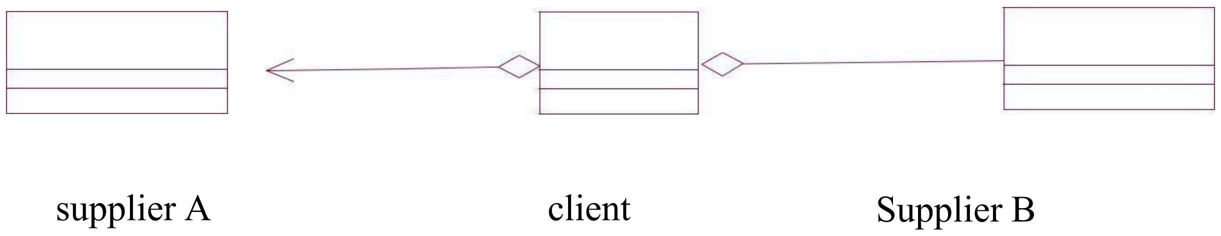


2) Relationships:

Aggregate Relationship

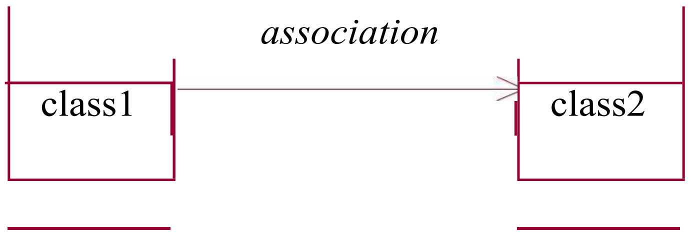
Use the aggregate relationship to show a whole and part relationship between two classes.

The class at the client end of the aggregate relationship is sometimes called the aggregate class. An instance of the aggregate class is an aggregate object. The class at the supplier end of the aggregate relationship is the part whose instances are contained or owned by theaggregate object.

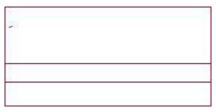


Association Relationship

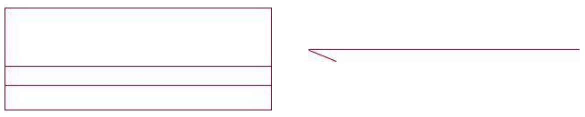
An association provides a pathway for communication. The communication can be between use cases, actors, classes or interfaces. Associations are the most general of all relationships and consequentially the most semantically weak. If two objects are usually considered independently, the relationship is an association



Dependency Relationship

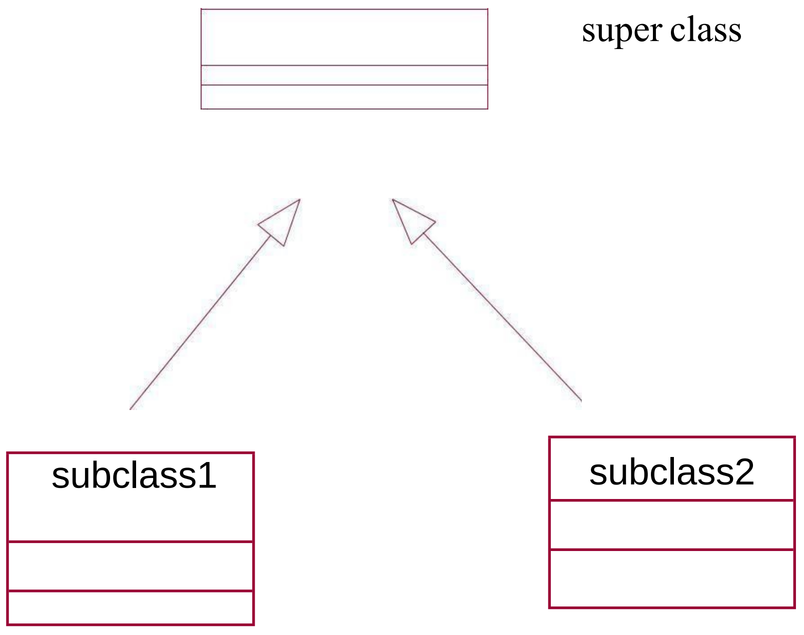


A dependency is a relationship between two model elements in which a change to onemodel element will affect the other model element. Use a dependency relationship to connect model elements with the same level of meaning



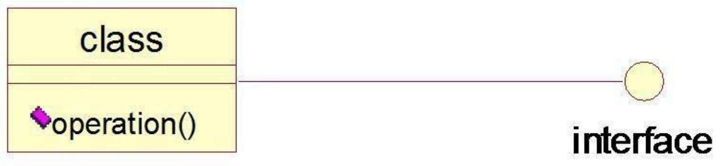
Generalize Relationship

A generalize relationship between classes shows that the subclass shares the structure or behavior defined in one or more super classes. Use a generalize relationship to show a "is-a" relationship between classes.



Realize Relationship

A realization is a relationship between classes, interfaces, components, and packages that connects a client element with a supplier element. A realization relationship between classes and interfaces and between components and interfaces shows that the class realizes the operations offered by the interface.



3) Interfaces

An interface specifies the externally visible operations of a class and/or component, andhas no implementation of its own. An interface typically specifies only a limited part of the behavior of a class or component.

Interfaces belong to the logical view but can occur in class, use case and componentdiagrams.

An interface in a component diagram is displayed as a small circle with a line to the component that realizes the interface:

Adornments

You can further define an interface using the Class Specification. Some class specification fields correspond to adornment and compartment information that youcan display in the class diagram.

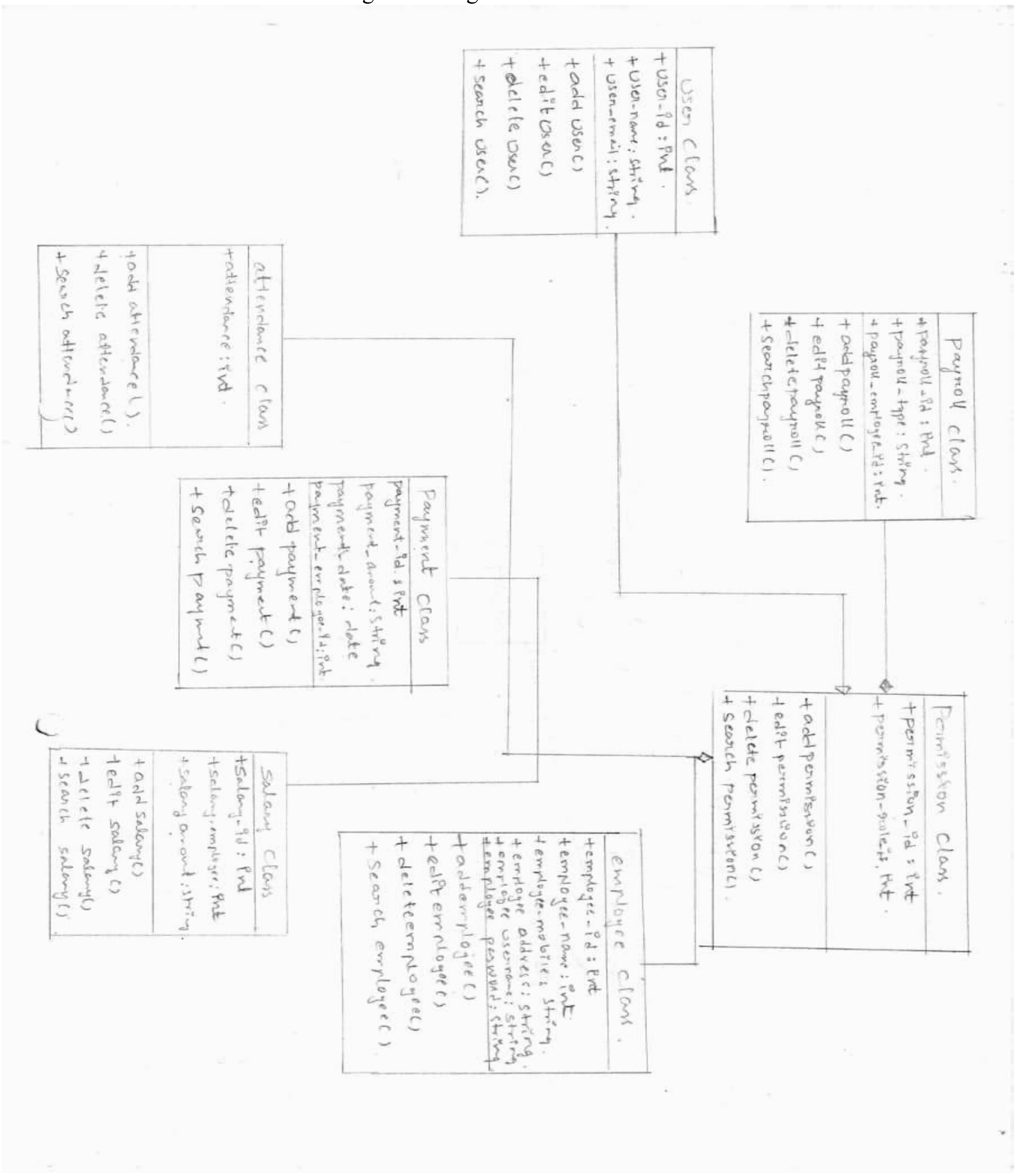
DIAGRAM:

AIM: To draw simple Class diagram for Payroll manger.

REQUIREMENTS:

HARDWARE: PIII Processor, 512 MB RAM, 80GB

SOFTWARE: IBM software using Class diagram tools.



SEQUENCE DIAGRAM

A sequence diagram is a graphical view of a scenario that shows object interaction in a time-based sequence¾what happens first, what happens next.

Sequence diagrams establish the roles of objects and help provide essential informationto determine class responsibilities and interfaces.

This type of diagram is best used during early analysis phases in design because theyare simple and easy to comprehend.

Sequence diagrams are normally associated with use cases.

Sequence diagrams are closely related to collaboration diagrams and both are alternate representations of an interaction.

There are two main differences between sequence and collaboration diagrams:Sequence

diagrams show time-based object interaction

While collaboration diagrams show how objects associate with each other.

CONTENTS:

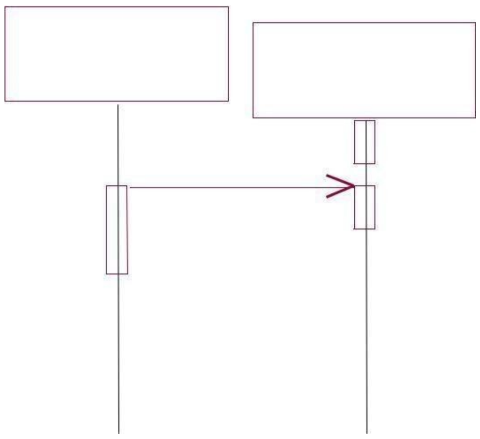
A sequence diagram has two dimensions: typically, vertical placement represents time and horizontal placement represents different objects.

The following tools located on the sequence diagram toolbox enable you to modelsequence diagrams:

Object: An object has state, behavior, and identity. The structure and behavior of similar objects are defined in their common class. Each object in a diagram indicates some instance of a class. An object that is not named is referred to as a class instance.

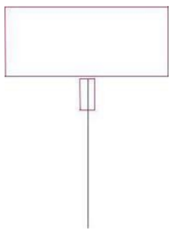
If you use the same name for several object icons appearing in the same collaboration oractivity diagram, they are assumed to represent the same object; otherwise, each object icon represents a distinct object. Object icons appearing in different diagrams denote different objects, even if their names are identical. Objects can be named three different ways: object name, object name and class, or just by the class name itself.

Message Icons: A message icon represents the communication between objects indicating that an action will follow. The message icon is a horizontal, solid arrowconnecting two lifelines together. The following message icons show three different ways a message icon can appear message icon only, message icon with sequence number, and message icon with sequence number and message label.



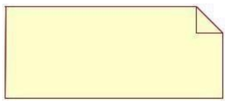
Each message icon represents a message passed between two objects, and indicates the direction of message is going. A message icon in a collaboration diagram can represent multiple messages. A message icon in a sequence diagram represents exactly one message.

Focus of Control: Focus of Control (FOC) is an advanced notational technique that enhances sequence diagrams. It shows the period of time during which an object is performing an action, either directly or through an underlying procedure. FOC is portrayed through narrow rectangles that adorn lifelines. The length of an FOC indicates theamount of time it takes for a message to be performed. When you move a message vertically, each dependent message will move vertically as well. Also, you can move a FOC vertically off of the source FOC to make it detached and independent. An illustration of a sequence diagram with FOC notation follows:



Message to Self: A Message to Self is a tool that sends a message from one object back tothe same object. It does not involve other objects because the message returns to the same object. The sender of a message is the same as the receiver.

Note: A note captures the assumptions and decisions applied during analysis and design. Notes may contain any information, including plain text, fragments of code, or references to other documents. Notes are also used as a means of linking diagrams. A note holds an unlimited amount of text and can be sized accordingly.



Notes behave like labels. They are available on all diagram toolboxes, but they are not considered part of the model. Notes may be deleted like any other item on a diagram.

Note Anchor: A note anchor connects a note to the element that it affects. To draw anote anchor, place a note on the diagram and connect the note to an element with the note anchor icon.

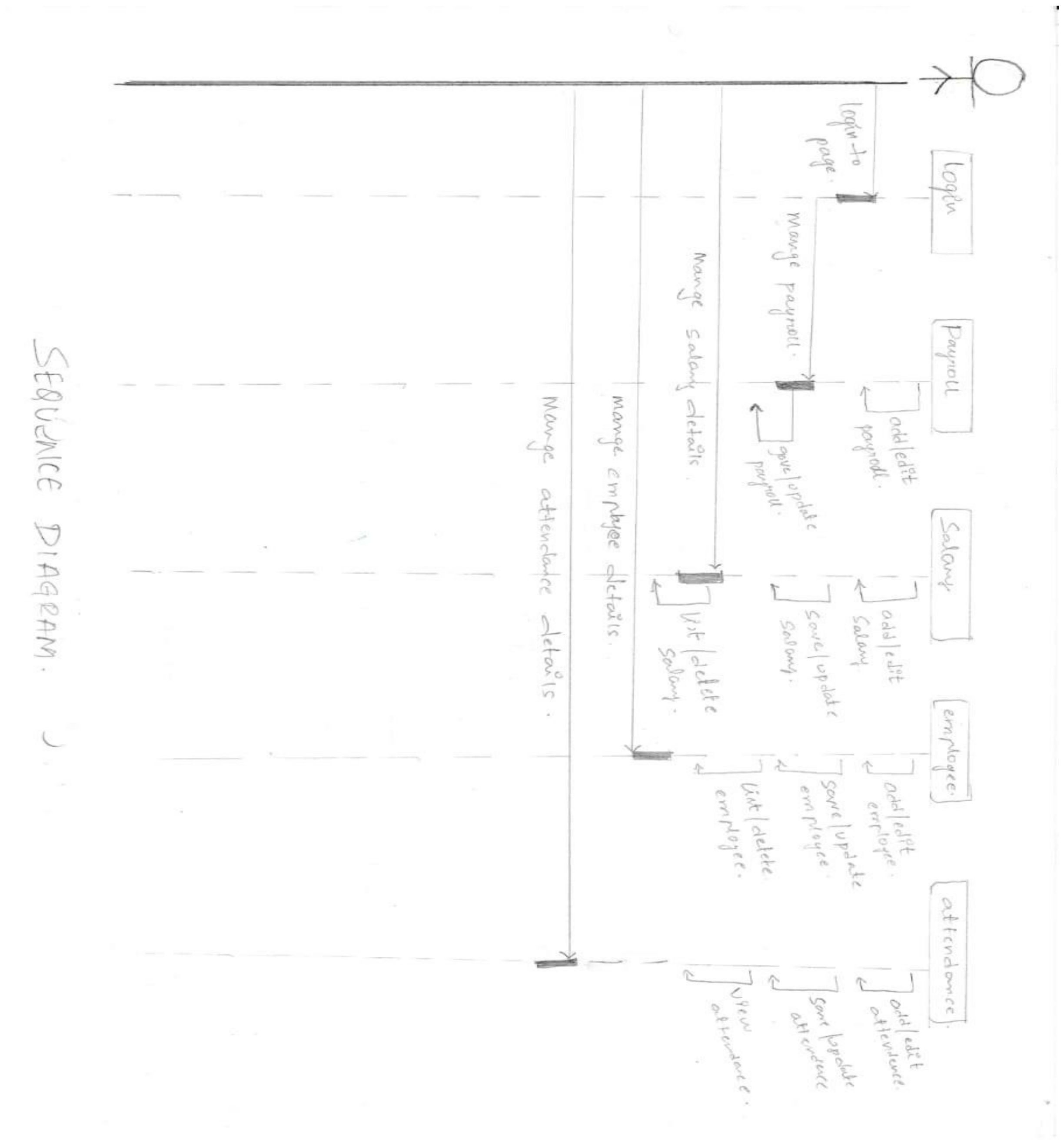
DIAGRAM:

AIM: To draw sequence diagram for Active Park Assist.

REQUIREMENTS:

HARDWARE : PIII Processor, 512 MB RAM, 80GB Harddisk.

SOFTWARE : In Rational Architecture software using sequence Diagram tool.



COLLABORATION DIAGRAM

Collaboration diagrams and sequence diagrams are alternate representations of an interaction. A collaboration diagram is an interaction diagram that shows the order of messages that implement an operation or a transaction. A sequence diagram shows object interaction in a time-based sequence.

Collaboration diagrams show objects, their links, and their messages. They can also contain simple class instances and class utility instances. Each collaboration diagram provides a view of interactions or structural relationships that occur between objects and object-like entities in the current model.

The Create Collaboration Diagram Command creates a collaboration diagram from information contained in the sequence diagram. The Create Sequence Diagram Command creates a sequence diagram from information contained in the interaction's collaboration diagram. The Go to Sequence Diagram and Go to CollaborationDiagram commands traverse between an interaction's two representations.

Collaboration diagrams contain icons representing objects. You can create one or more collaboration diagrams to depict interactions for each logical package in your model. Such collaboration diagrams are themselves contained by the logical package enclosing the objects they depict.

An Object Specification enables you to display and modify the properties andrelationships of an object. The information in a specification is presented textually. Some of this information can also be displayed inside the icons representing objects in collaboration diagrams.
You can change properties or relationships by editing the specification or modifying
the icon on the diagram. The associated diagrams or specifications are automatically updated.

During: Analysis

Use Collaboration Diagrams: To Indicate the semantics of the primary and secondary interactions.

Design Show the semantics of mechanisms in the logical design of the system.
Use collaboration diagrams as the primary vehicle to describe interactions that expressyour decisions about the behavior of the system.

Object:

An object has state, behavior, and identity. The structure and behavior of similar objects aredefined in their common class. Each object in a diagram indicates some instance of a class. An object that is not named is referred to as a class instance.

If you use the same name for several object icons appearing in the same collaboration oractivity diagram, they are assumed to represent the same object; otherwise, each object icon represents a distinct object. Object icons appearing in different diagrams denote different objects, even if their names are identical. Objects can be named three different ways: object name, object name and class, or just by the class name itself.

If you specify the name of the object's class in the Object Specification, the name must identify a class defined in the model.

Graphical Depiction

The object icon is similar to a class icon except that the name is underlined:
If you have multiple objects that are instances of the same class, you can modify the object icon by clicking Multiple Instances in the Object Specification. When you selectthis field, the icon is changed from one object to three staggered objects:

Concurrency

An object's concurrency is defined by the concurrency of its class.

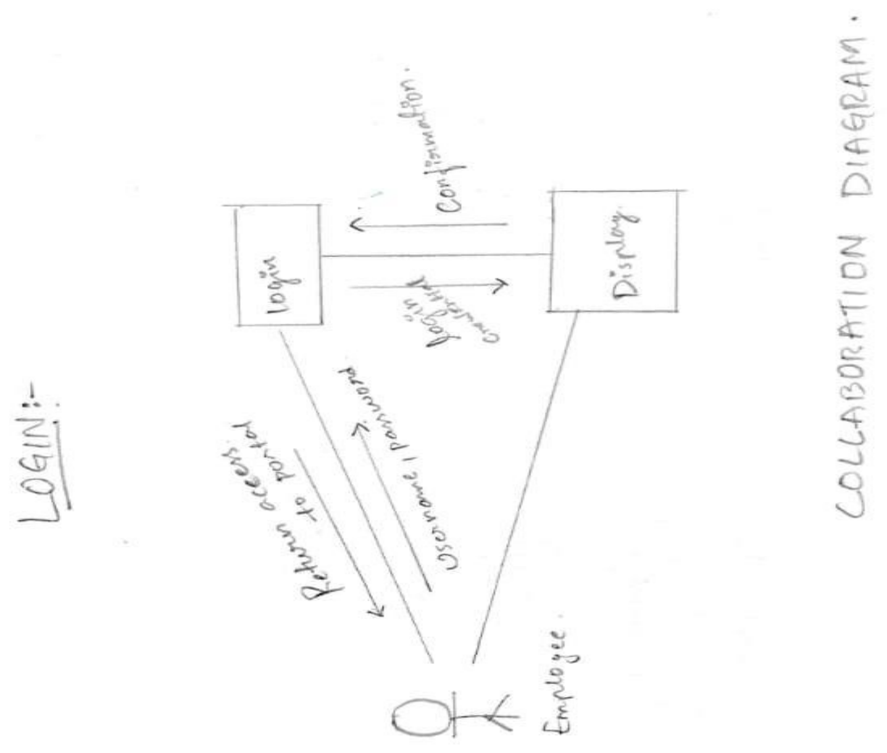
You can display concurrency by clicking Show Concurrency from the object's shortcutmenu. The adornment is displayed at the bottom of the object ic

DIAGRAM:

AIM: To draw Collaboration diagram for Active Park Assist.
REQUIREMENTS:

HARDWARE **:** PIII Processor, 512 MB RAM, 80GB

SOFTWARE **:** In Rational Architecture software using Collaboration Diagram tools



ACTIVITY DIAGRAM

DEFINITION:

Activity diagrams provide a way to model the workflow of a business process. You can also use activity diagrams to model code-specific information such as a class operation. Activity diagrams are very similar to a flowchart because you can model a workflow from activity to activity.

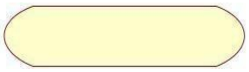
An activity diagram is basically a special case of a state machine in which most of the states are activities and most of the transitions are implicitly triggered by completion of the actions in the source activities. The main difference between activity diagrams and state charts is activity diagrams areactivity centric, while state charts are state centric.

An activity diagram is typically used for modeling the sequence of activities in a process, whereas a state chart is better suited to model the discrete stages of an object's lifetime.

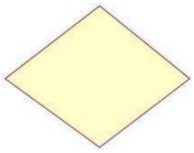
CONTENTS:

Activity Diagram Tools
You can use the following tools on the activity diagram toolbox to model activitydiagrams:

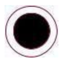
- **Activities:** An activity represents the performance of task or duty in a workflow. It may also represent the execution of a statement in a procedure. An activity is similar toa state but expresses the intent that there is no significant waiting (for events) in an activity. Transitions connect activities with other model elements and object flows connect activities with objects.



Decisions: A decision represents a specific location on an activity diagram or state chart diagram where the workflow may branch based upon guard conditions. There may be more than two outgoing transitions with different guard conditions, but for themost part, a decision will have only two outgoing transitions determined by a Boolean expression



End State: An end state represents a final or terminal state on an activity diagram or state chart diagram. Place an end state when you want to explicitly show the end of a workflow on an activity diagram or the end of a state chart diagram. Transitions can only occur into an end state; however, there can be any number of end states per context.



Object: Rational Rose allows objects on activity, collaboration, and sequence diagrams. Specific to activity diagrams, objects are model elements that represent something you can feel and touch. It might be helpful to think of objects as the nouns of the activity diagram and activities as the verbs of the activity diagram. Further, objects on activity diagrams allow you to diagram the input and output relationships between activities. In the following diagram, the Submit Defect and Fix Defects can be thought of as the verbs and the defect objects as the nouns in the activity diagram vocabulary. Objects are connected to activities through object flows.

Object Flow: An object flow on an activity diagram represents the relationship betweenan activity and the object that creates it (as an output) or uses it (as an input).

Rational Rose draws object flows as dashed arrows rather than solid arrows to distinguish them from ordinary transitions. Object flows look identical to dependencies that appear on other diagram types.

Start State: A start state (also called an "initial state") explicitly shows the beginning of a workflow on an activity diagram or the beginning of the execution of a state machine on a state chart diagram. You can have only one start state for each state machine because each workflow/execution of a state machine begins in the same place.

●

Normally, only one outgoing transition can be placed from the start state. However, multiple transitions may be placed on a start state if at least one of them is labeled with a condition. No incoming transitions are allowed.

States: A state represents a condition or situation during the life of an object during which it satisfies some condition or waits for some event. Each state represents the cumulative history of its behavior.

Swim lanes: Swim lanes are helpful when modeling a business workflow because they can represent organizational units or roles within a business model. Swim lanes arevery similar to an object because they provide a way to tell who is performing a certain role. Swim lanes only appear on activity diagrams. You should place activities within swim lanes to determine which unit is responsible for carrying out the specific activity. For more information on swim lanes, look at the swim lane sample.

When a swim lane is dragged onto an activity diagram, it becomes a swim lane view. Swim lanes appear as small icons in the browser while a swim lane views appearbetween the thin, vertical lines with a header that can be renamed and relocated.

Synchronizations: Synchronizations enable you to see a simultaneous workflow in an activity diagram or state chart diagram. Synchronizations visually define forks and joins representing parallel workflow.

Transitions: A state transition indicates that an object in the source state will perform certain specified actions and enter the destination state when a specified event occurs or when certain conditions are satisfied. A state transition is a relationship between two states, two activities, or between an activity and a state.

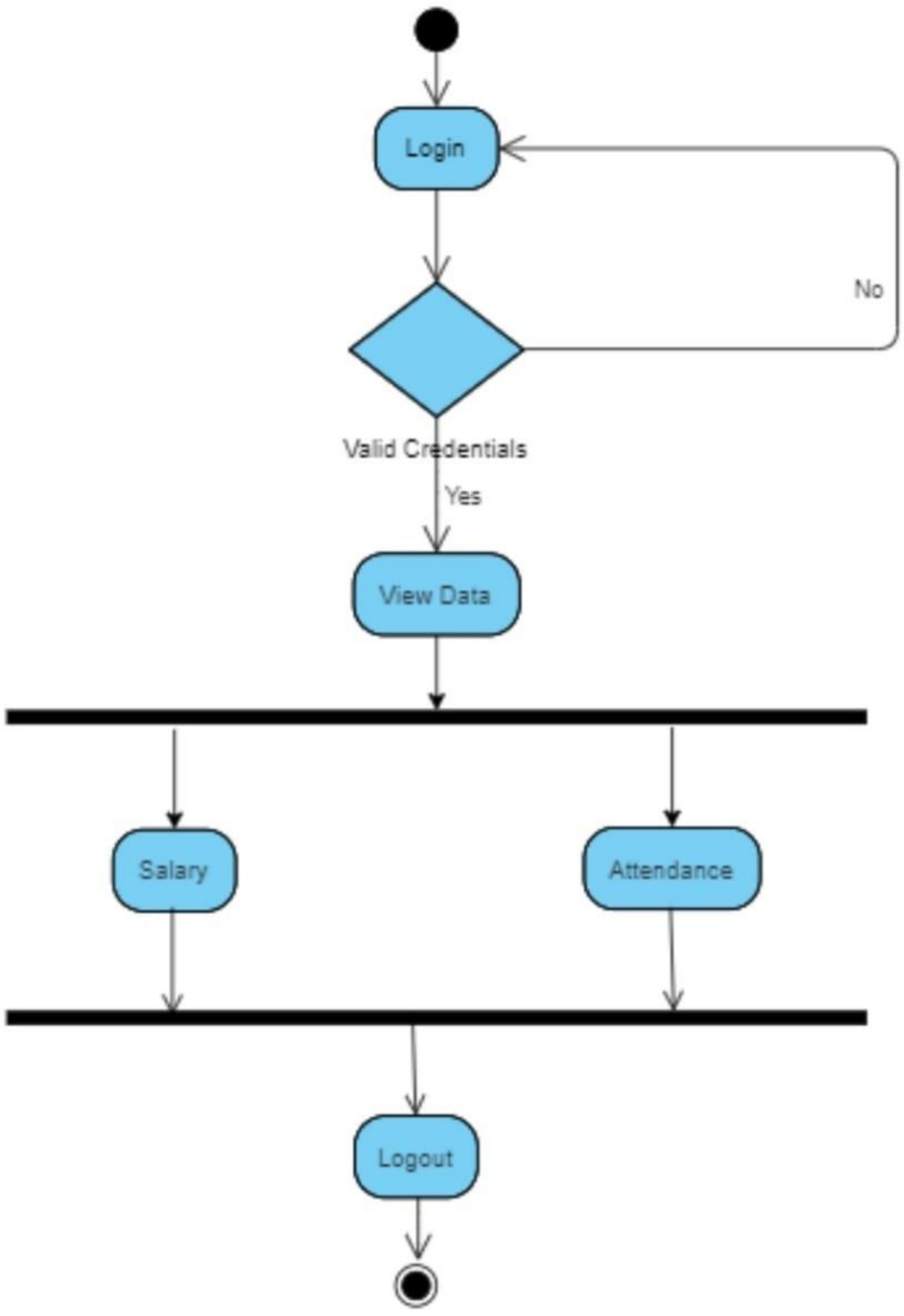
DIAGRAM:

AIM: To draw Activity diagram for Active Park Assist.

REQUIREMENTS:

HARDWARE **:** PIII Processor, 512 MB RAM, 80GB

SOFTWARE: In Rational Architecture software using Activity diagram tools.

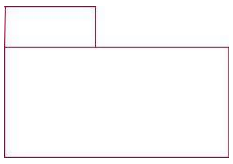


COMPONENT DIAGRAM

Component diagrams provide a physical view of the current model. A component diagram shows the organizations and dependencies among software components, including source code components, binary code components, and executable components. These diagrams also show the externally visible behavior of the components by displaying the interfaces of the components. Calling dependencies among components are shown as dependency relationships between components and interfaces on other components. Note that the interfaces actually belong to the logical view, but they can occur both in class diagrams and in component diagrams.

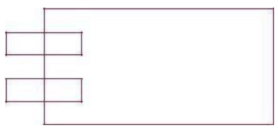
Component diagrams contain:

· **<u>Component packages</u>:** Component packages represent clusters of logically related components, or major pieces of your system. Component packages parallel the role played by logical packages for class diagrams. They allow you to partition the physical model of the system.

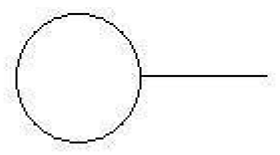


· **<u>Components</u>**: A component represents a software module (source code, binary code, executable, DLL, etc.) with a well-defined interface. The interface of a component is represented by one or several interface elements that the component provides. Components are used to show compiler and run-time dependencies, as well as interface and calling dependencies among software modules. They also show which components implement a specific class.

A system may be composed of several software modules of different kinds. Each software module is represented by a component in the model. To distinguish different kinds of components from each other, stereotypes are used.



<u>Interfaces:</u> An interface specifies the externally-visible operations of a class and/or component, and has no implementation of its own. An interface typically specifies only alimited part of the



behavior of a class or component.

· **Dependency relationships:** A dependency is a relationship between two model elements in which a change to one model element will affect the other model element. Use a dependency relationship to connect model elements with the same level of meaning. Typically, on class diagrams, a dependency relationship indicates that theoperations of the client invoke operations of the supplier.

You can create one or more component diagrams to depict the component packages and components at the top level of the component view, or to depict the contents of each component package. Such component diagrams belong to the component package that they depict.

A Component Package Specification enables you to display and modify the properties of a component package. Similarly, a Component Specification and a Class Specification enables you to display and modify the properties of a component and an interface, respectively. The information in these specifications is presented textually. Some of this information can also be displayed inside the icons representing component packages and components in component diagrams, and interfaces in class diagrams.

You can change properties of, or relationships among, component packages, components, and interfaces by editing the specification or modifying the icon on the diagram. The affected diagrams or specifications are automatically updated.

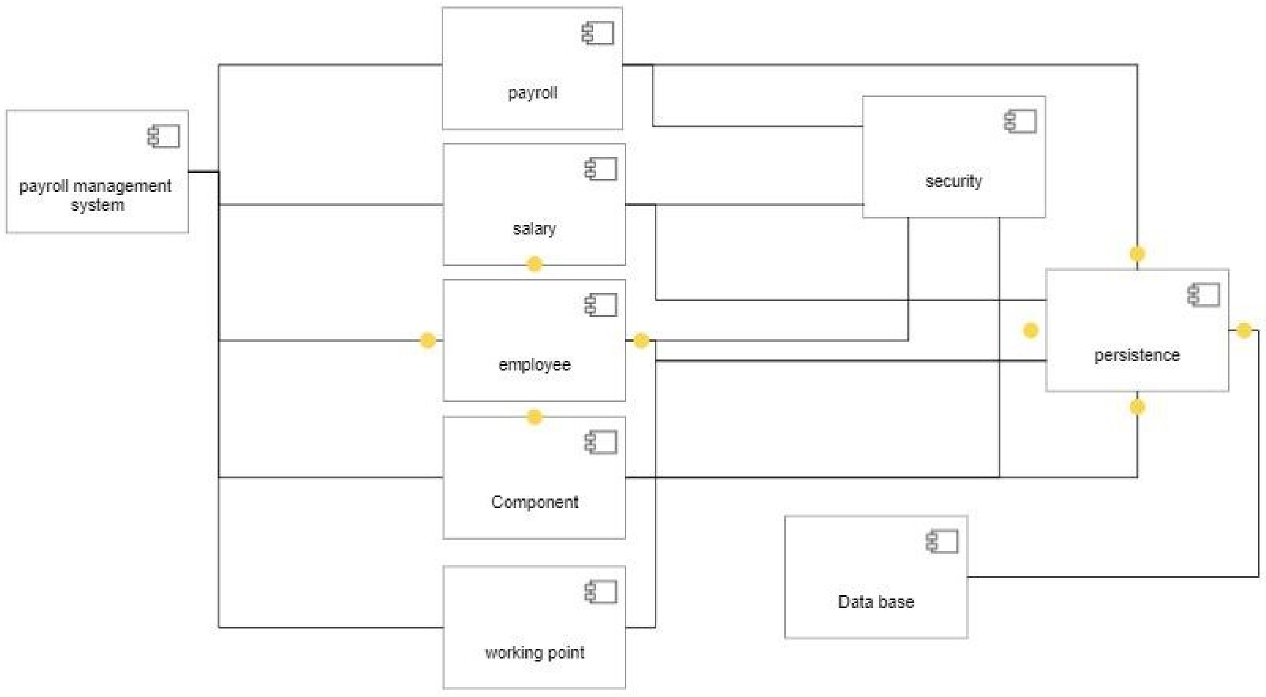
DIAGRAM:

AIM: To draw Component diagram for Active Park Assist.

REQUIREMENTS:

HARDWARE **:** PIII Processor, 512 MB RAM, 80GB

SOFTWARE **:** In Rational Architecture software using component diagramtools.



DEPLOYMENT DIAGRAM

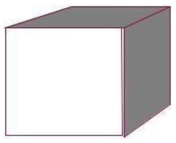
A deployment diagram shows processors, devices, and connections. Each model contains a single deployment diagram which shows the connections between its processors and devices, and the allocation of its processes to processors.

Processor Specifications, Device Specifications, and Connection Specifications enable you to display and modify the respective properties. The information in a specificationis presented textually; some of this information can also be displayed inside the icons.

You can change properties or relationships by editing the specification or modifyingthe icon on the diagram. The deployment diagram specifications are automatically updated.

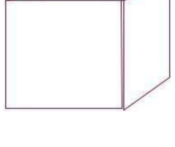
CONTENTS:

Processor:-A processor is a hardware component capable of executing programs.



Devices:

A device is a hardware component with no computing power. Each device musthave a name. Device names can be generic, such as "modem" or "terminal."



Connections:

A connection represents some type of hardware coupling between two entities. An entity is either a processor or a device. The hardware coupling can be direct, such asan RS232 cable, or indirect, such as satellite-to-ground communication. Connectionsare usually bi-directional.

DIAGRAM:

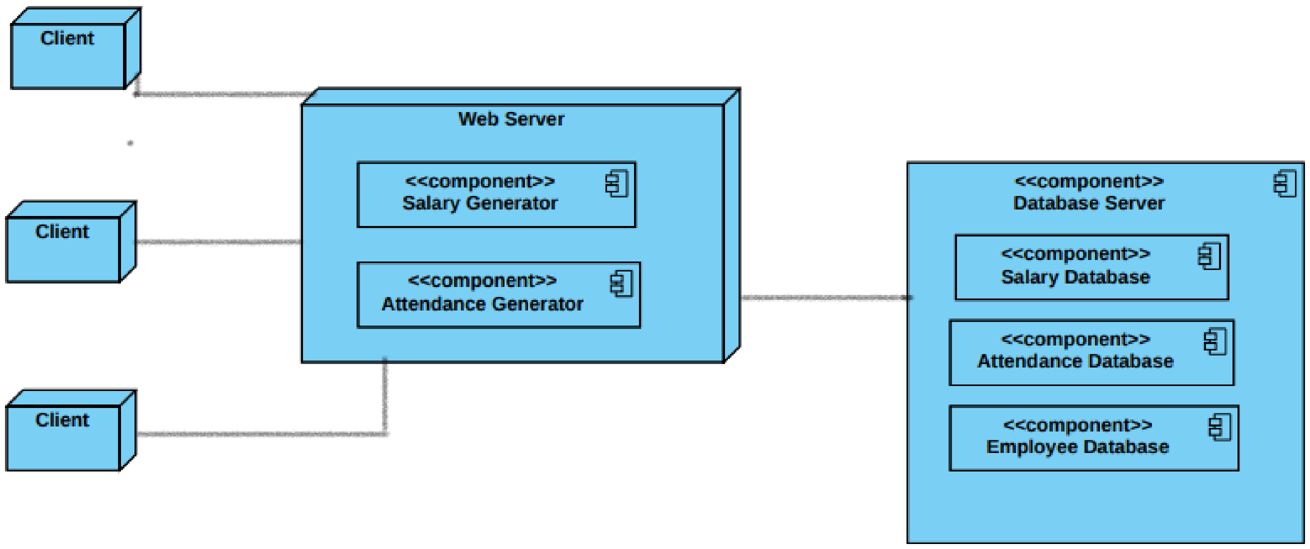
AIM:

To draw Deployment diagram for Active Park Assist.

REQUIREMENTS:

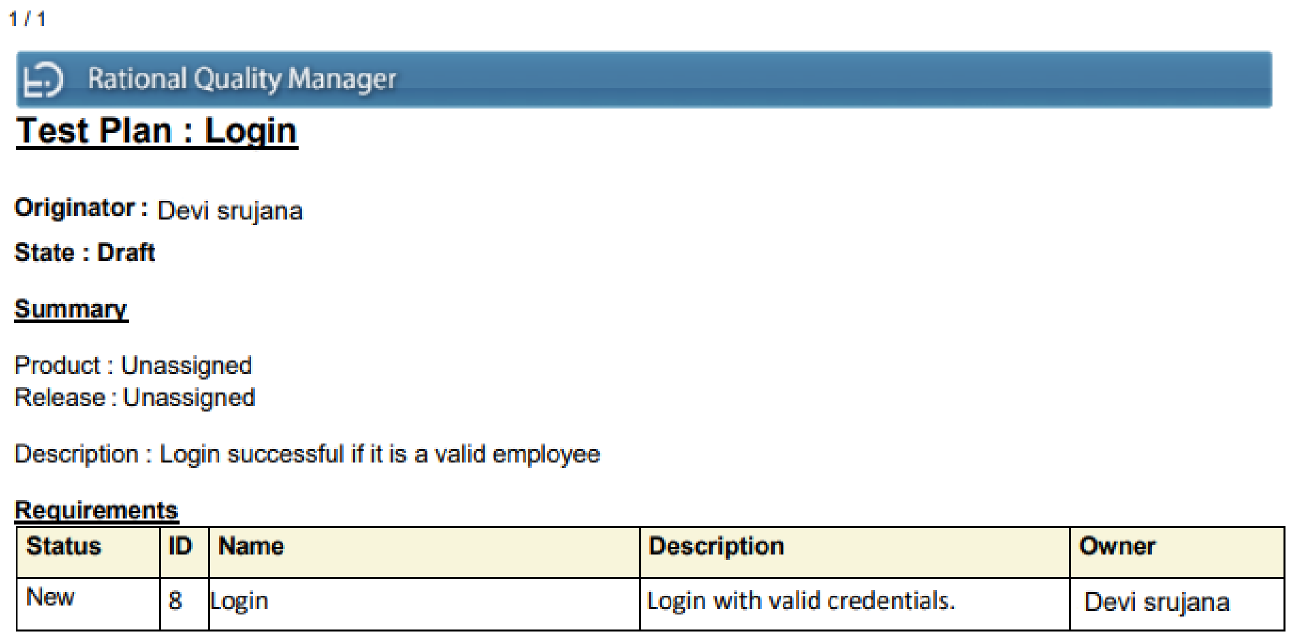
HARDWARE : PIII Processor, 512 MB RAM, 80GB

SOFTWARE: In Rational Architecture software using Deployment diagram tools.



TEST CASES

A good Test Case template maintains test artifact consistency for the test team and makes it easy for all stakeholders to understand the test cases. Writing test case in a standard format lessen the test effort and the error rate. Test cases format are more desirable in case if you are reviewing test case from experts.



1 / 1

Rational Quality Manager

Test Plan : Login

Originator : Devi srujana

State : Draft

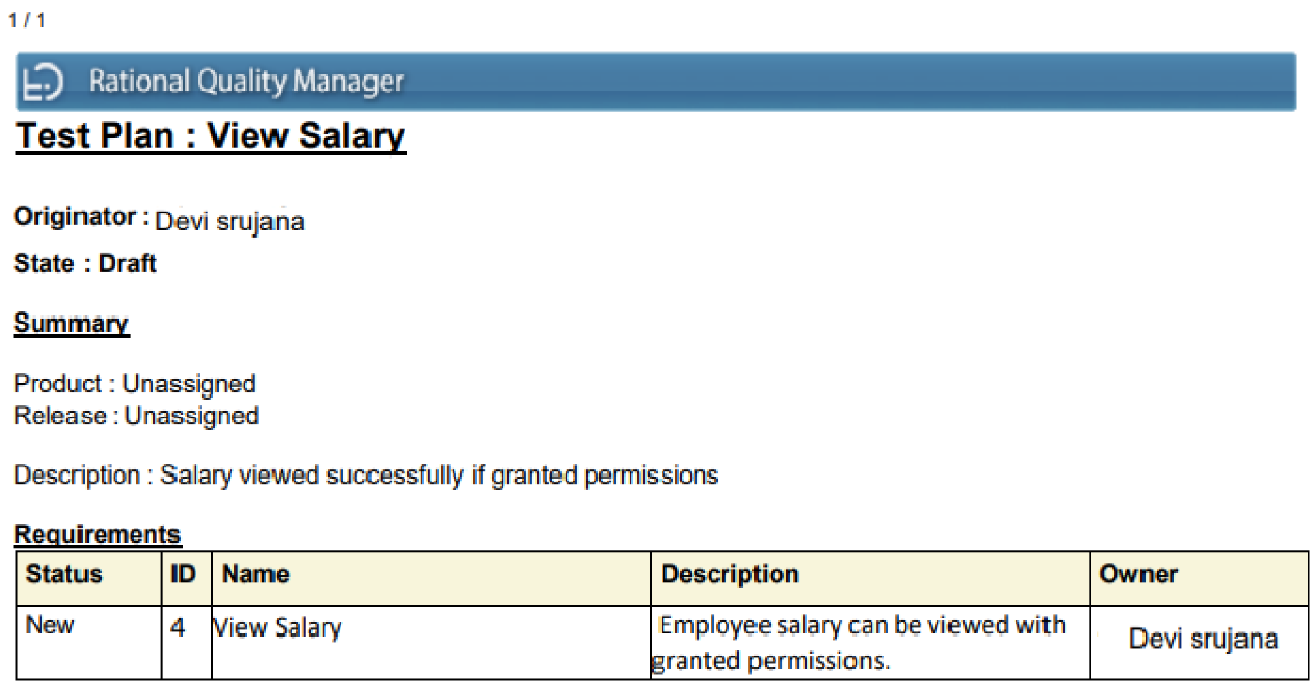
Summary

Product : Unassigned
Release : Unassigned

Description : Login successful if it is a valid employee

Requirements

Status	ID	Name	Description	Owner
New	8	Login	Login with valid credentials.	Devi srujana



1 / 1

Rational Quality Manager

Test Plan : View Salary

Originator : Devi srujana

State : Draft

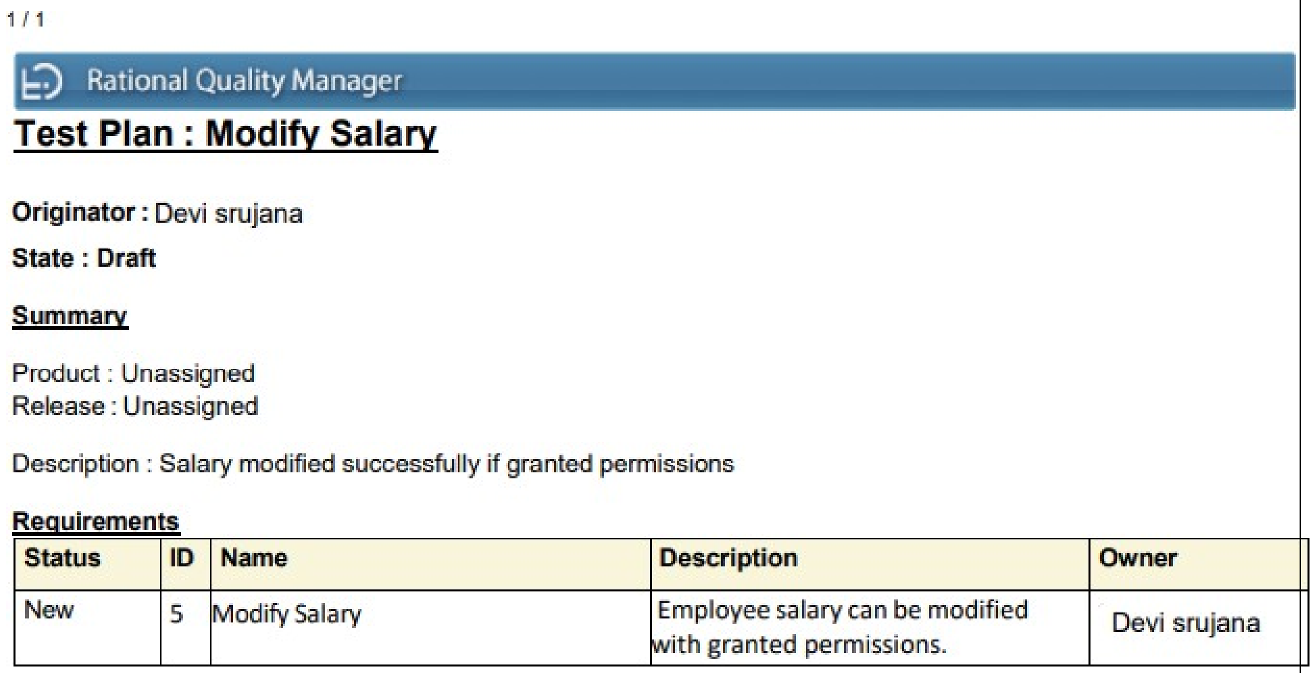
Summary

Product : Unassigned
Release : Unassigned

Description : Salary viewed successfully if granted permissions

Requirements

Status	ID	Name	Description	Owner
New	4	View Salary	Employee salary can be viewed with granted permissions.	Devi srujana



1 / 1

Rational Quality Manager

Test Plan : Modify Salary

Originator : Devi srujana

State : Draft

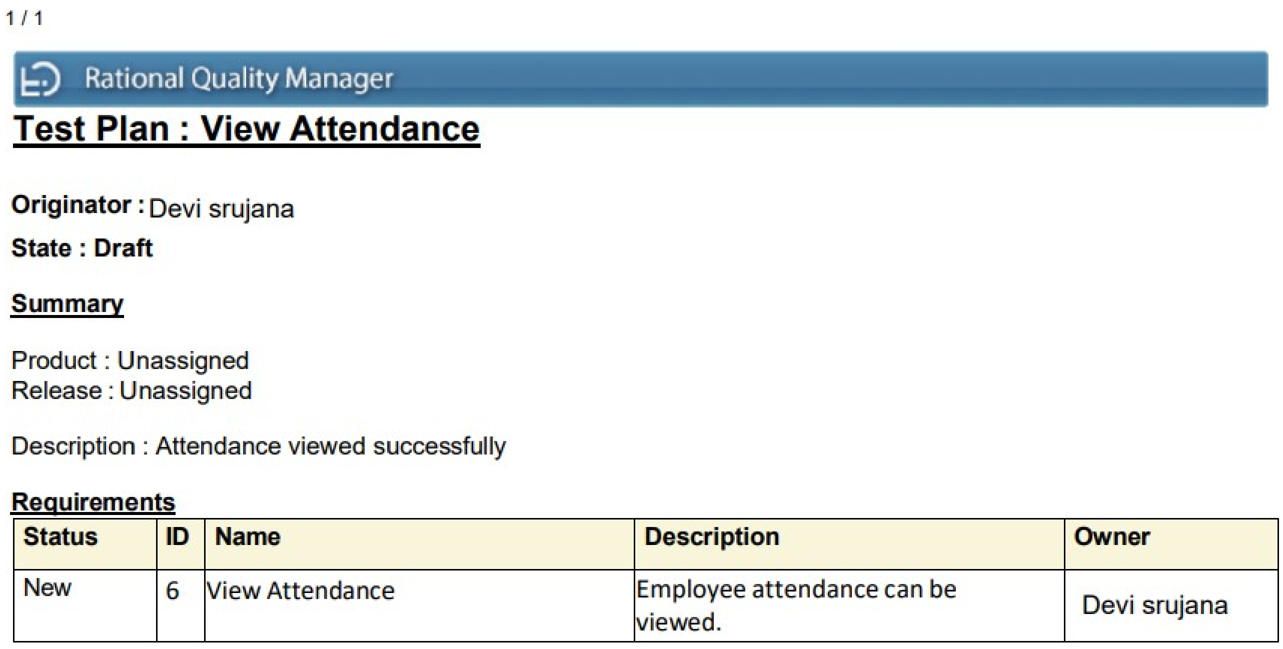
Summary

Product : Unassigned
Release : Unassigned

Description : Salary modified successfully if granted permissions

Requirements

Status	ID	Name	Description	Owner
New	5	Modify Salary	Employee salary can be modified with granted permissions.	Devi srujana



1 / 1

Rational Quality Manager

Test Plan : View Attendance

Originator : Devi srujana

State : Draft

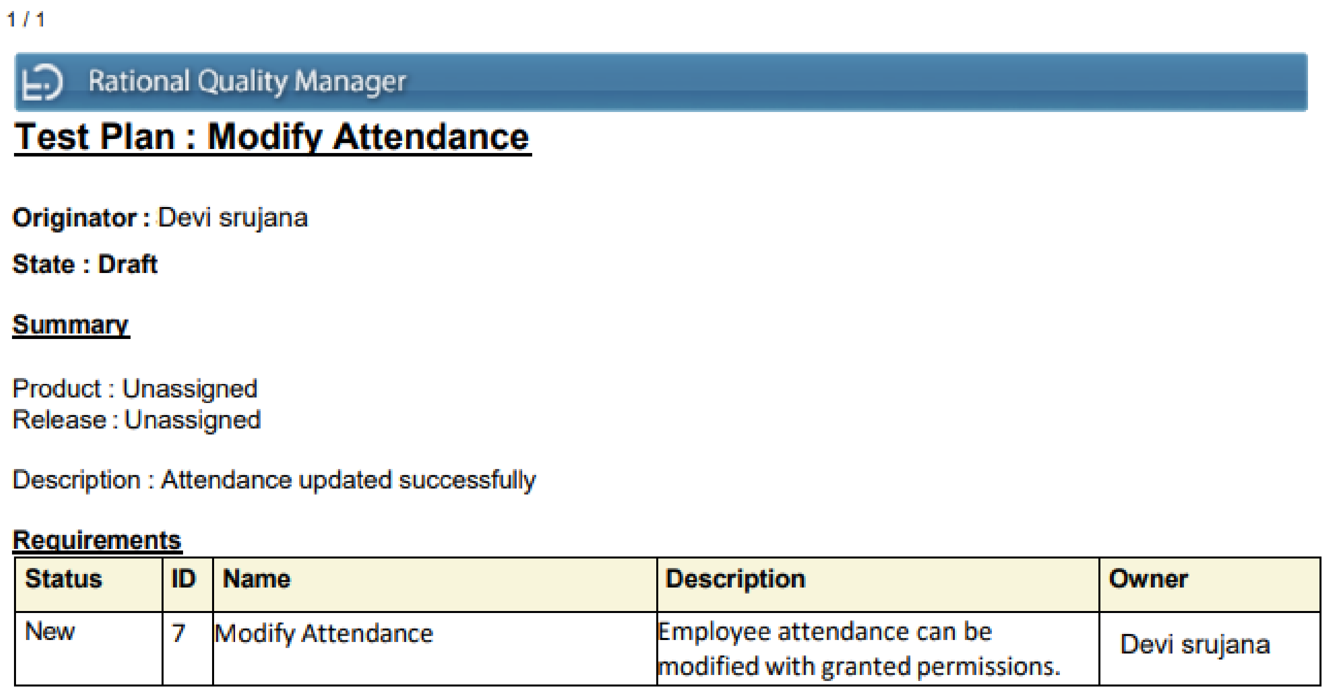
Summary

Product : Unassigned
Release : Unassigned

Description : Attendance viewed successfully

Requirements

Status	ID	Name	Description	Owner
New	6	View Attendance	Employee attendance can be viewed.	Devi srujana



1 / 1

Rational Quality Manager

Test Plan : Modify Attendance

Originator : Devi srujana

State : Draft

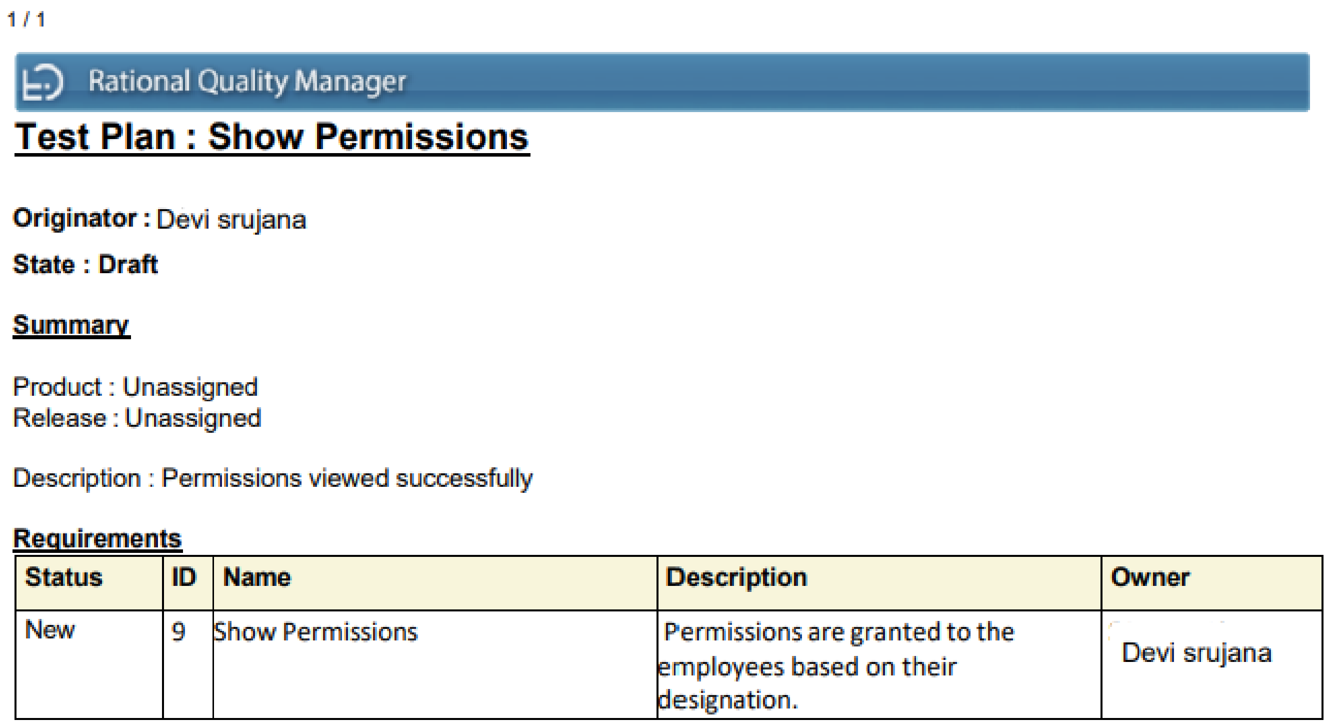
Summary

Product : Unassigned
Release : Unassigned

Description : Attendance updated successfully

Requirements

Status	ID	Name	Description	Owner
New	7	Modify Attendance	Employee attendance can be modified with granted permissions.	Devi srujana



1 / 1

Rational Quality Manager

Test Plan : Show Permissions

Originator : Devi srujana

State : Draft

Summary

Product : Unassigned
Release : Unassigned

Description : Permissions viewed successfully

Requirements

Status	ID	Name	Description	Owner
New	9	Show Permissions	Permissions are granted to the employees based on their designation.	Devi srujana

Rational Quality Manager

Test Plan : View Details

Originator : Devi srujana

State : Draft

Summary

Product : Unassigned
Release : Unassigned

Description : View employee details

Requirements

Status	ID	Name	Description	Owner
New	10	View Details	Employee details are viewed which can also be modified.	Devi srujana

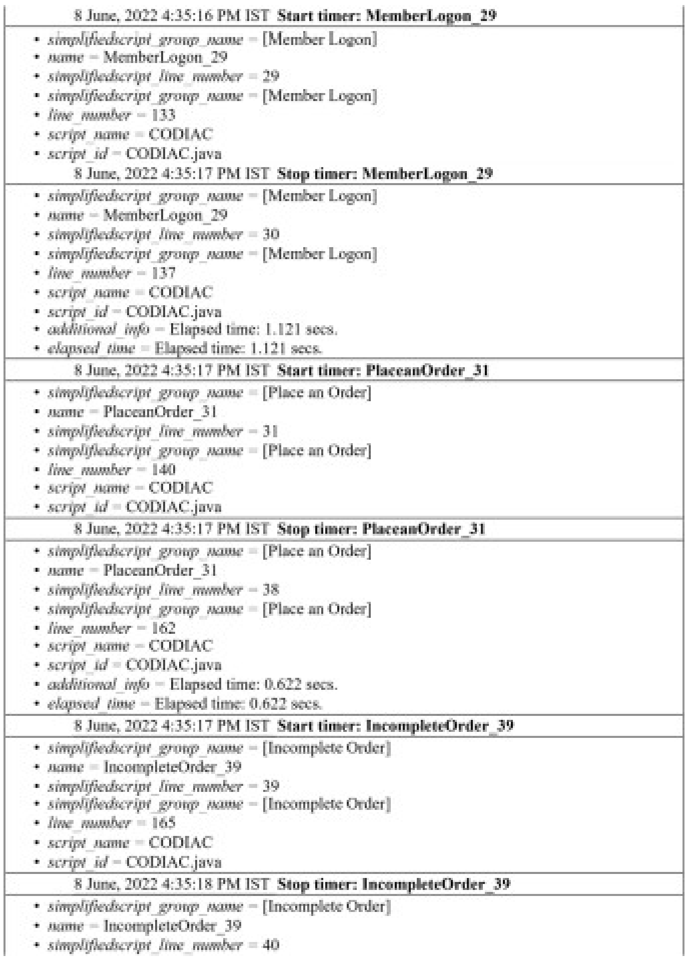
```
        8 June, 2022 4:35:12 PM IST Script start [CODIAC]
 • line_number = 1
 • script_iter_count = 0
 • script_name = CODIAC
 • script_id = CODIAC.java
        8 June, 2022 4:35:12 PM IST Start timer: ClassicsCD_1
 • simplifiedscript_group_name = [ClassicsCD]
 • name = ClassicsCD_1
 • simplifiedscript_line_number = 1
 • simplifiedscript_group_name = [ClassicsCD]
 • line_number = 44
 • script_name = CODIAC
 • script_id = CODIAC.java
        8 June, 2022 4:35:13 PM IST Stop timer: ClassicsCD_1
 • simplifiedscript_group_name = [ClassicsCD]
 • name = ClassicsCD_1
 • simplifiedscript_line_number = 2
 • simplifiedscript_group_name = [ClassicsCD]
 • line_number = 48
 • script_name = CODIAC
 • script_id = CODIAC.java
 • additional_info = Elapsed time: 0.198 secs.
 • elapsed_time = Elapsed time: 0.198 secs.
        8 June, 2022 4:35:13 PM IST Start timer: MemberLogon_3
 • simplifiedscript_group_name = [Member Logon]
 • name = MemberLogon_3
 • simplifiedscript_line_number = 3
 • simplifiedscript_group_name = [Member Logon]
 • line_number = 51
 • script_name = CODIAC
 • script_id = CODIAC.java
        8 June, 2022 4:35:13 PM IST Stop timer: MemberLogon_3
 • simplifiedscript_group_name = [Member Logon]
 • name = MemberLogon_3
 • simplifiedscript_line_number = 5
 • simplifiedscript_group_name = [Member Logon]
 • line_number = 58
 • script_name = CODIAC
 • script_id = CODIAC.java
 • additional_info = Elapsed time: 0.205 secs.
 • elapsed_time = Elapsed time: 0.205 secs.
        8 June, 2022 4:35:13 PM IST Start timer: PlaceanOrder_6
```

file:///C:/Users/Administrator/IBM/rationalsdp/USER/CODIAC_logs/CODIAC/rational   08-06-2022

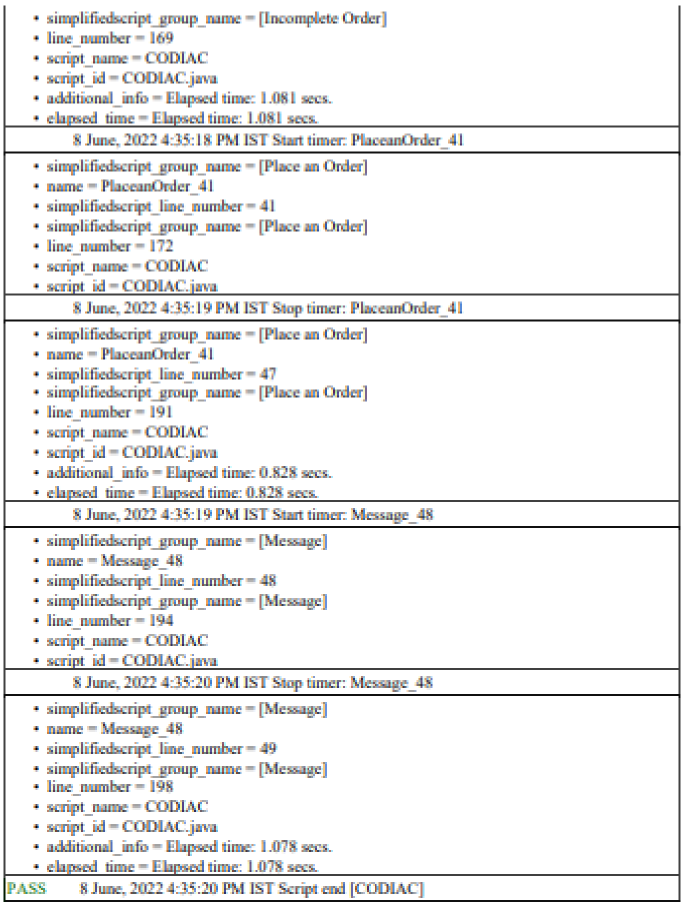
Playback Log for CODIAC   Page 2 of 6

```
 •      simplifiedscript_group_name = [Place an Order]
 •      name = PlaceanOrder_6
 •      simplifiedscript_line_number = 6
 •      simplifiedscript_group_name = [Place an Order] • line_number = 61
```

```
 • script_name = CODIAC
 • script_id = CODIAC.java
        8 June, 2022 4:35:15 PM IST  Stop timer: PlaceanOrder_6
 • simplifiedscript_group_name = [Place an Order]
 • name = PlaceanOrder_6
 • simplifiedscript_line_number = 24
 • simplifiedscript_group_name = [Place an Order]
 • line_number = 116
 • script_name = CODIAC
 • script_id = CODIAC.java
 • additional_info = Elapsed time: 2.669 secs.
 • elapsed_time = Elapsed time: 2.669 secs.
        8 June, 2022 4:35:15 PM IST  Start timer: Message_25
 • simplifiedscript_group_name = [Message]
 • name = Message_25
 • simplifiedscript_line_number = 25
 • simplifiedscript_group_name = [Message]
 • line_number = 119
 • script_name = CODIAC
 • script_id = CODIAC.java
        8 June, 2022 4:35:16 PM IST  Stop timer: Message_25
 • simplifiedscript_group_name = [Message]
 • name = Message_25
 • simplifiedscript_line_number = 26
 • simplifiedscript_group_name = [Message]
 • line_number = 123
 • script_name = CODIAC
 • script_id = CODIAC.java
 • additional_info = Elapsed time: 0.082 secs.
 • elapsed_time = Elapsed time: 0.082 secs.
        8 June, 2022 4:35:16 PM IST  Start timer: ClassicsCD_27
 • simplifiedscript_group_name = [ClassicsCD]
 • name = ClassicsCD_27
 • simplifiedscript_line_number = 27
 • simplifiedscript_group_name = [ClassicsCD]
 • line_number = 126
 • script_name = CODIAC
 • script_id = CODIAC.java
        8 June, 2022 4:35:16 PM IST  Stop timer: ClassicsCD_27
 • simplifiedscript_group_name = [ClassicsCD]
 • name = ClassicsCD_27
 • simplifiedscript_line_number = 28
 • simplifiedscript_group_name = [ClassicsCD]
 • line_number = 130
 • script_name = CODIAC
 • script_id = CODIAC.java
 • additional_info = Elapsed time: 0.095 secs.
 • elapsed_time = Elapsed time: 0.095 secs.
```



```
8 June, 2022 4:35:16 PM IST  Start timer: MemberLogon_29
• simplifiedscript_group_name = [Member Logon]
• name = MemberLogon_29
• simplifiedscript_line_number = 29
• simplifiedscript_group_name = [Member Logon]
• line_number = 133
• script_name = CODIAC
• script_id = CODIAC.java
8 June, 2022 4:35:17 PM IST  Stop timer: MemberLogon_29
• simplifiedscript_group_name = [Member Logon]
• name = MemberLogon_29
• simplifiedscript_line_number = 30
• simplifiedscript_group_name = [Member Logon]
• line_number = 137
• script_name = CODIAC
• script_id = CODIAC.java
• additional_info = Elapsed time: 1.121 secs.
• elapsed_time = Elapsed time: 1.121 secs.
8 June, 2022 4:35:17 PM IST  Start timer: PlaceanOrder_31
• simplifiedscript_group_name = [Place an Order]
• name = PlaceanOrder_31
• simplifiedscript_line_number = 31
• simplifiedscript_group_name = [Place an Order]
• line_number = 140
• script_name = CODIAC
• script_id = CODIAC.java
8 June, 2022 4:35:17 PM IST  Stop timer: PlaceanOrder_31
• simplifiedscript_group_name = [Place an Order]
• name = PlaceanOrder_31
• simplifiedscript_line_number = 38
• simplifiedscript_group_name = [Place an Order]
• line_number = 162
• script_name = CODIAC
• script_id = CODIAC.java
• additional_info = Elapsed time: 0.622 secs.
• elapsed_time = Elapsed time: 0.622 secs.
8 June, 2022 4:35:17 PM IST  Start timer: IncompleteOrder_39
• simplifiedscript_group_name = [Incomplete Order]
• name = IncompleteOrder_39
• simplifiedscript_line_number = 39
• simplifiedscript_group_name = [Incomplete Order]
• line_number = 165
• script_name = CODIAC
• script_id = CODIAC.java
8 June, 2022 4:35:18 PM IST  Stop timer: IncompleteOrder_39
• simplifiedscript_group_name = [Incomplete Order]
• name = IncompleteOrder_39
• simplifiedscript_line_number = 40
```



```
• simplifiedscript_group_name = [Incomplete Order]
• line_number = 169
• script_name = CODIAC
• script_id = CODIAC.java
• additional_info = Elapsed time: 1.081 secs.
• elapsed_time = Elapsed time: 1.081 secs.
8 June, 2022 4:35:18 PM IST Start timer: PlaceanOrder_41
• simplifiedscript_group_name = [Place an Order]
• name = PlaceanOrder_41
• simplifiedscript_line_number = 41
• simplifiedscript_group_name = [Place an Order]
• line_number = 172
• script_name = CODIAC
• script_id = CODIAC.java
8 June, 2022 4:35:19 PM IST Stop timer: PlaceanOrder_41
• simplifiedscript_group_name = [Place an Order]
• name = PlaceanOrder_41
• simplifiedscript_line_number = 47
• simplifiedscript_group_name = [Place an Order]
• line_number = 191
• script_name = CODIAC
• script_id = CODIAC.java
• additional_info = Elapsed time: 0.828 secs.
• elapsed_time = Elapsed time: 0.828 secs.
8 June, 2022 4:35:19 PM IST Start timer: Message_48
• simplifiedscript_group_name = [Message]
• name = Message_48
• simplifiedscript_line_number = 48
• simplifiedscript_group_name = [Message]
• line_number = 194
• script_name = CODIAC
• script_id = CODIAC.java
8 June, 2022 4:35:20 PM IST Stop timer: Message_48
• simplifiedscript_group_name = [Message]
• name = Message_48
• simplifiedscript_line_number = 49
• simplifiedscript_group_name = [Message]
• line_number = 198
• script_name = CODIAC
• script_id = CODIAC.java
• additional_info = Elapsed time: 1.078 secs.
• elapsed_time = Elapsed time: 1.078 secs.
PASS    8 June, 2022 4:35:20 PM IST Script end [CODIAC]
```

Application Development

Aim: To generate a java code from a UML class diagram for Road Trip Planner.

Software Tools used:

Rational Software Architect

Description:

Software development is a process by which standalone or individual software is created using a specific programming language. Software development may also be called application development and software design. Application development is basically the process of creating a computer program or set of programs that can assist the daily functionalities of the user or business. This is the "original" type of programming. These are 'standard' applications that perform their duties on traditional desktop operating systems, such as Windows, Mac, or Linux. It's often considered a programme, executed on demand by the user, that opens its interface in the confines of the OS that it's running in.

Java, VB.NET, C/C++,C#, Python.

Procedure:

1. Open RSA:File ☾ New ☾ Project
2. Select a wizard as UML Project
3. Enter Project Name
4. Under Create new UML Model ☾ Template ☾ Blank Model
5. Click Finish.
6. Design a class diagram/Select Existing Model and the Save the diagram.
7. Select Blank model
8. Go to menu bar :Modeling ☾ Transfer ☾ New Configuration.

9. Under forward Transformation ☾ IBM Rational Transformations

10. Select UML to

Javav5.0 ☾ next12.Under source

and target 13.Select your project

from source

14.Project name ☾ models ☾ Blank model

15.Click create new target container.

16. Enter Java code Project Name(Passport Java)-->Next

17.Java Settings-->Finish, then(observe Target as Project Name Java)

18.Click Next

19.UML to Java Options-->Next

20.Collections click Next

21.Mapping-->Next

22.Common ☾ Next

23.Select Transformation Options as create source to target relationship.

24.Click Finish.

25. From the Project Explorer ,select Passport UML to JavaCode.tc ☾ Right click ☾ Transform ☾ UML to Javav5.0

26. From Explorer:- Expand Passport Java Folder ☾ Expand default package ☾ See your classes,Java files

Double Click filename.Java file you can see Java Code.

```
/**
 *
 */

/**
 * @author STUDENT
 * @generated "UML to Java V5.0 (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
 */
public class Salary {
    /**
     * @generated "UML to Java V5.0 (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
     */
    private Integer id;

    /**
     * @return the id
     * @generated "UML to Java V5.0 (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
     */
    public Integer getId() {
        // begin-user-code
        return id;
    // end-user-code
    }

    /**
     * @param theId  the id to set
     * @generated "UML to Java V5.0 (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
     */
    public void setId(Integer theId) {
        // begin-user-code
        id = theId;
    // end-user-code
    }

    /**
     * @generated "UML to Java V5.0 (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
     */
    private Integer s_calculation;

    /**
     * @return the s_calculation
     * @generated "UML to Java V5.0 (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
     */
    public Integer getS_calculation() {
        // begin-user-code
        return s_calculation;
    // end-user-code
    }

    /**
     * @param theS_calculation  the s_calculation to set
     * @generated "UML to Java V5.0 (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
     */
    public void setS_calculation(Integer theS_calculation) {
        // begin-user-code
        s_calculation = theS_calculation;
    // end-user-code
    }

    /**
     * @generated "UML to Java V5.0 (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
```

```
	 */
	private String s_amount;

	/**
	 * @return the s_amount
	 * @generated "UML to Java V5.0 
	 (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
	 */
	public String getS_amount() {
		// begin-user-code
		return s_amount;
		// end-user-code
	}

	/**
	 * @param theS_amount  the s_amount to set
	 * @generated  "UML to Java V5.0 
	 (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
	 */
	public void setS_amount(String theS_amount) {
		// begin-user-code
		s_amount = theS_amount;
		// end-user-code
	}

	/**
	 * @generated "UML to Java V5.0 
	 (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
	 */
	private Employee_Details employee_details;

	/**
	 * @return the employee_details
	 * @generated "UML to Java V5.0 
	 (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
	 */
	public Employee_Details getEmployee_details() {
		// begin-user-code
		return employee_details;
		// end-user-code
	}

	/**
	 * @param theEmployee_details the employee_details to set
	 * @generated "UML to Java V5.0 
	 (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
	 */
	public void setEmployee_details(Employee_Details theEmployee_details) {
		// begin-user-code
		employee_details = theEmployee_details;
		// end-user-code
	}

	/**
	 * @generated "UML to Java V5.0 
	 (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
	 */
	public void calculate() {
		// begin-user-code
		// TODO Auto-generated method stub

		// end-user-code
	}

	/**
	 * @generated "UML to Java V5.0 
	 (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
	 */
	public void modify() {
```

```java
        // begin-user-code
        // TODO Auto-generated method stub

        // end-user-code
    }
}
```

```
/**
 *
 */

/**
 * @author STUDENT
 * @generated "UML to Java V5.0
 (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
 */
public class Higher_Designation extends Employee_Details {
    /**
     * @generated "UML to Java V5.0
     (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
     */
    private Integer id;

    /**
     * @return the id
     * @generated "UML to Java V5.0
     (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
     */
    public Integer getId() {
        // begin-user-code
        return id;
    // end-user-code }

    /**
     * @param theId  the id to set
     * @generated "UML to Java V5.0
     (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
     */
    public void setId(Integer theId) {
        // begin-user-code
        id = theId;
    // end-user-code }

    /**
     * @generated "UML to Java V5.0
     (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
     */
    private String name;

    /**
     * @return the name
     * @generated "UML to Java V5.0
     (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
     */
    public String getName() {
        // begin-user-code
        return name;
    // end-user-code }

    /**
     * @param theName  the name to set
     * @generated "UML to Java V5.0
     (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
     */
    public void setName(String theName) {
        // begin-user-code
        name = theName;
    // end-user-code }

    /**
     * @generated "UML to Java V5.0
     (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
     */
    private String username;

    /**
     * @return the username
     * @generated "UML to Java V5.0
     (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
     */
    public String getUsername() {
        // begin-user-code
        return username;
    // end-user-code }

    /**
     * @param theUsername  the username to set
     * @generated "UML to Java V5.0
     (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
     */
    public void setUsername(String theUsername) {
        // begin-user-code
        username = theUsername;
    // end-user-code }

    /**
     * @generated "UML to Java V5.0
     (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
     */
    private String password;

    /**
     * @return the password
     * @generated "UML to Java V5.0
     (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
     */
    public String getPassword() {
        // begin-user-code
        return password;
    // end-user-code }

    /**
     * @param thePassword  the password to set
     * @generated "UML to Java V5.0
     (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
     */
    public void setPassword(String thePassword) {
        // begin-user-code
        password = thePassword;
        // end-user-code
    }

    /**
     * @generated "UML to Java V5.0
     (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
     */
    private String permission;

    /**
     * @return the permission
     * @generated "UML to Java V5.0
     (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
     */
    public String getPermission() {
        // begin-user-code
        return permission;
        // end-user-code
    }
```

```java

    /**
     * @param thePermission  the permission to set
     * @generated  "UML to Java V5.0
     (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
     */
    public void setPermission(String thePermission) {
        // begin-user-code
        permission = thePermission;
        // end-user-code
    }

    /**
     * @generated "UML to Java V5.0
     (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
     */
    private Employee_Details employee_details;

    /**
     * @return the employee_details
     * @generated "UML to Java V5.0
     (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
     */
    public Employee_Details getEmployee_details() {
        // begin-user-code
        return employee_details;
        // end-user-code
    }

    /**
     * @param theEmployee_details the employee_details to set
     * @generated "UML to Java V5.0
     (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
     */
    public void setEmployee_details(Employee_Details theEmployee_details) {
        // begin-user-code
        employee_details = theEmployee_details;
        // end-user-code
    }

    /**
     * @generated "UML to Java V5.0
     (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
     */
    public void has_permission() {
        // begin-user-code
        // TODO Auto-generated method stub

        // end-user-code
    }

    /**
     * @generated "UML to Java V5.0
     (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
     */
    public void modify() {
        // begin-user-code
        // TODO Auto-generated method stub

        // end-user-code
    }
}
```

```
/**
 *
 */

/**
 * @author STUDENT
 * @generated "UML to Java V5.0
 (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
 */
public class Lower_Designation extends Employee_Details {
    /**
     * @generated "UML to Java V5.0
     (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
     */
    private Integer id;

    /**
     * @return the id
     * @generated "UML to Java V5.0
     (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
     */
    public Integer getId() {
        // begin-user-code
        return id;
    // end-user-code }

    /**
     * @param theId  the id to set
     * @generated "UML to Java V5.0
     (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
     */
    public void setId(Integer theId) {
        // begin-user-code
        id = theId;
    // end-user-code }

    /**
     * @generated "UML to Java V5.0
     (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
     */
    private String name;

    /**
     * @return the name
     * @generated "UML to Java V5.0
     (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
     */
    public String getName() {
        // begin-user-code
        return name;
    // end-user-code }

    /**
     * @param theName  the name to set
     * @generated "UML to Java V5.0
     (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
     */
    public void setName(String theName) {
        // begin-user-code
        name = theName;
    // end-user-code }

    /**
     * @generated "UML to Java V5.0
     (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
```

```java
     */
    private String username;

    /**
     * @return the username
     * @generated "UML to Java V5.0
     (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
     */
    public String getUsername() {
        // begin-user-code
        return username;
    // end-user-code
    }

    /**
     * @param theUsername  the username to set
     * @generated "UML to Java V5.0
     (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
     */
    public void setUsername(String theUsername) {
        // begin-user-code
        username = theUsername;
    // end-user-code
    }

    /**
     * @generated "UML to Java V5.0
     (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
     */
    private String password;

    /**
     * @return the password
     * @generated "UML to Java V5.0
     (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
     */
    public String getPassword() {
        // begin-user-code
        return password;
    // end-user-code
    }

    /**
     * @param thePassword  the password to set
     * @generated "UML to Java V5.0
     (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
     */
    public void setPassword(String thePassword) {
        // begin-user-code
        password = thePassword;
        // end-user-code
    }

    /**
     * @generated "UML to Java V5.0
     (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
     */
    private String permission;

    /**
     * @return the permission
     * @generated "UML to Java V5.0
     (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
     */
    public String getPermission() {
        // begin-user-code
        return permission;
        // end-user-code
    }

    /**
     * @param thePermission  the permission to set
     * @generated "UML to Java V5.0
     (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
     */
    public void setPermission(String thePermission) {
        // begin-user-code
        permission = thePermission;
        // end-user-code
    }

    /**
     * @generated "UML to Java V5.0
     (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
     */
    public void has_permission() {
        // begin-user-code
        // TODO Auto-generated method stub

        // end-user-code
    }

    /**
     * @generated "UML to Java V5.0
     (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
     */
    public void modify() {
        // begin-user-code
        // TODO Auto-generated method stub

        // end-user-code
    }

    /**
     * @generated "UML to Java V5.0
     (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
     */
    public void create() {
        // begin-user-code
        // TODO Auto-generated method stub

        // end-user-code
    }
}
```

```java
/**
 *
 */

/**
 * @author STUDENT
 * @generated "UML to Java V5.0
 (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
 */
public class Employee_Details {
    /**
     * @generated "UML to Java V5.0
     (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
     */
    public Integer id;

    /**
     * @return the id
     * @generated "UML to Java V5.0
     (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
     */
    public Integer getId() {
        // begin-user-code
        return id;
    // end-user-code
    }

    /**
     * @param theId  the id to set
     * @generated "UML to Java V5.0
     (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
     */
    public void setId(Integer theId) {
        // begin-user-code
        id = theId;
    // end-user-code
    }

    /**
     * @generated "UML to Java V5.0
     (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
     */
    private String first_name;

    /**
     * @return the first_name
     * @generated "UML to Java V5.0
     (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
     */
    public String getFirst_name() {
        // begin-user-code
        return first_name;
    // end-user-code
    }

    /**
     * @param theFirst_name  the first_name to set
     * @generated "UML to Java V5.0
     (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
     */
    public void setFirst_name(String theFirst_name) {
        // begin-user-code
        first_name = theFirst_name;
    // end-user-code
    }

    /**
     * @generated "UML to Java V5.0
     (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
     */
    private String last_name;

    /**
     * @return the last_name
     * @generated "UML to Java V5.0
     (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
     */
    public String getLast_name() {
        // begin-user-code
        return last_name;
    // end-user-code }

    /**
     * @param theLast_name  the last_name to set
     * @generated "UML to Java V5.0
     (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
     */
    public void setLast_name(String theLast_name) {
        // begin-user-code
        last_name = theLast_name;
    // end-user-code }

    /**
     * @generated "UML to Java V5.0
     (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
     */
    private Integer age;

    /**
     * @return the age
     * @generated "UML to Java V5.0
     (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
     */
    public Integer getAge() {
        // begin-user-code
        return age;
    // end-user-code }

    /**
     * @param theAge  the age to set
     * @generated "UML to Java V5.0
     (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
     */
    public void setAge(Integer theAge) {
        // begin-user-code
        age = theAge;
        // end-user-code
    }

    /**
     * @generated "UML to Java V5.0
     (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
     */
    private String status;

    /**
     * @return the status
     * @generated "UML to Java V5.0
     (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
     */
    public String getStatus() {
        // begin-user-code
        return status;
        // end-user-code
    }
    /**
     * @param theStatus  the status to set
     * @generated "UML to Java V5.0
     (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
     */
    public void setStatus(String theStatus) {
        // begin-user-code
        status = theStatus;
        // end-user-code
    }

    /**
     * @generated "UML to Java V5.0
     (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
     */
    private String username;

    /**
     * @return the username
     * @generated "UML to Java V5.0
     (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
     */
    public String getUsername() {
        // begin-user-code
        return username;
        // end-user-code
    }

    /**
     * @param theUsername  the username to set
     * @generated "UML to Java V5.0
     (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
     */
    public void setUsername(String theUsername) {
        // begin-user-code
        username = theUsername;
        // end-user-code
    }

    /**
     * @generated "UML to Java V5.0
     (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
     */
    private Salary salary;

    /**
     * @return the salary
     * @generated "UML to Java V5.0
     (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
     */
    public Salary getSalary() {
        // begin-user-code
        return salary;
        // end-user-code
    }

    /**
     * @param theSalary the salary to set
     * @generated "UML to Java V5.0
     (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
     */
    public void setSalary(Salary theSalary) {
        // begin-user-code
        salary = theSalary;
        // end-user-code
    }

    /**
```

```java
     * @generated "UML to Java V5.0
     (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
     */
    private Attendance attendance;

    /**
     * @return the attendance
     * @generated "UML to Java V5.0
     (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
     */
    public Attendance getAttendance() {
        // begin-user-code
        return attendance;
        // end-user-code
    }

    /**
     * @param theAttendance the attendance to set
     * @generated "UML to Java V5.0
     (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
     */
    public void setAttendance(Attendance theAttendance) {
        // begin-user-code
        attendance = theAttendance;
        // end-user-code
    }

    /**
     * @generated "UML to Java V5.0
     (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
     */
    public void create() {
        // begin-user-code
        // TODO Auto-generated method stub

        // end-user-code
    }

    /**
     * @generated "UML to Java V5.0
     (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
     */
    public void modify() {
        // begin-user-code
        // TODO Auto-generated method stub

        // end-user-code
    }
}
```

```
/**
 *
 */

/**
 * @author STUDENT
 * @generated "UML to Java V5.0
 (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
 */
public class Attendance {
    /**
     * @generated "UML to Java V5.0
     (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
     */
    private Integer id;

    /**
     * @return the id
     * @generated "UML to Java V5.0
     (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
     */
    public Integer getId() {
        // begin-user-code
        return id;
    // end-user-code }

    /**
     * @param theId  the id to set
     * @generated "UML to Java V5.0
     (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
     */
    public void setId(Integer theId) {
        // begin-user-code
        id = theId;
    // end-user-code }

    /**
     * @generated "UML to Java V5.0
     (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
     */
    private String type;

    /**
     * @return the type
     * @generated "UML to Java V5.0
     (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
     */
    public String getType() {
        // begin-user-code
        return type;
    // end-user-code }

    /**
     * @param theType the type to set
     * @generated "UML to Java V5.0
     (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
     */
    public void setType(String theType) {
        // begin-user-code
        type = theType;
    // end-user-code }

    /**
     * @generated "UML to Java V5.0
     (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
     */
    private String description;

    /**
     * @return the description
     * @generated "UML to Java V5.0
     (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
     */
    public String getDescription() {
        // begin-user-code
        return description;
        // end-user-code
    }

    /**
     * @param theDescription  the description to set
     * @generated  "UML to Java V5.0
     (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
     */
    public void setDescription(String theDescription) {
        // begin-user-code
        description = theDescription;
        // end-user-code
    }

    /**
     * @generated "UML to Java V5.0
     (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
     */
    private Employee_Details employee_details;

    /**
     * @return the employee_details
     * @generated "UML to Java V5.0
     (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
     */
    public Employee_Details getEmployee_details() {
        // begin-user-code
        return employee_details;
        // end-user-code
    }

    /**
     * @param theEmployee_details the employee_details to set
     * @generated "UML to Java V5.0
     (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
     */
    public void setEmployee_details(Employee_Details theEmployee_details) {
        // begin-user-code
        employee_details = theEmployee_details;
        // end-user-code
    }

    /**
     * @generated "UML to Java V5.0
     (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
     */
    public void generate() {
        // begin-user-code
        // TODO Auto-generated method stub

        // end-user-code
    }

    /**
     * @generated "UML to Java V5.0
     (com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
     */
    public void modify() {
        // begin-user-code
        // TODO Auto-generated method stub

        // end-user-code
    }
}
```

GITHUB:-

https://github.com/Devisrujanat/Payroll-manger.git

1.3 Development Process

A process model is chosen based on the nature of the project and application, the methods and tools to be used for functionalities that are required as per user. For the Railway Reservation system we choose **spiral model**.

1.4 Effort, Schedule and Team:

The team comprises of the following 2 persons: T.Devi srujana and C.samiksha

Total Effort:3.2 person-months (65 person-days)

Project duration:3.6 months

1.5 Assumptions made:

No major assumptions beyond what is stated in the SRS.

2. Detailed Effort and Schedule

The phase wise estimates were obtained earlier. To summarize the total effort is 65 person-days. Of this the distribution is design: 0.4 (12 days), detailed design: 0.6 (16 days), coding: 1.0 (25 days), and integration: 0.4 (12 days). As the project staff (students) are spending on the project about 1/4th to 1/3rd of their total time, the durations of the tasks must be suitably fixed. The overall schedule for the project is given below.The total estimated effort in person-days is: 65

#	Test	Estimated Effort (person – days)	Start Date (dd/mm/ yyyy	End date (dd/m m/ Yyyy)	Person	Actual Effort (manhr' s)
1	System design	12	Mar 17	Apr 5	A,B	
2	Detailed design	16	Apr 5	May 3	A,B	
3	Coding input module	9	May 3	June 5	A	
4	Coding Schedule module	9	May 3	June 5	B	
5	Coding output module	7	May 3	June 5	A	
6	Test planning	4	May 17	June 5	A,B	
7	Testing and integration	5	June 5	Jun 25	A,B	
8	Rework and Final	3	Jun 25	Jul 4	A,B	

3. Team Organization

We will have a small team of two persons A and B. Both were assigned equal time and effort in this project development. The assignment of tasks to them will be maintained in the detailed schedule, a high-level view of which is given above.

4. Hardware and Software resources required

The only hardware resource required is a workstation with notepad++ and command prompt with java version 8+.

5. Quality Plan

The quality control process for this project will consist of the following:

- SRS Review: The SRS will be reviewed by both.
- Design Review: Design document will be reviewed by both.
- Unit Testing: Each programmer is responsible for Unit Testing his module.
- System Testing: Will be done according to the system test plan, which will be reviewed.

6. Risk Management Plan

There are no risks with this project that might need any explicit mitigation.

7. Project Tracking

Three basic methods will be used for monitoring – project logs, weekly meetings, and reviews. As there is no timesheet system, both of us will record this activity in a project notebook and report the hours for each activity in the meetings.

Reviews will be held as per the quality plan.

https://youtu.be/cGkHjby1xKM Gantt charts are used to track the project progress

Conclusion:-

The process of payroll manger has been fully automated with this software. This web app can now avail to check the attendance and salary of the employee.

Reference:-

https://www.tutorialspoint.com/uml/uml_standard_diagrams

https://www.geeksforgeeks.org/software-engineering-system-configurationmanagement

https://www.javatpoint.com/functional-testing