Robosoft Technologies
EMOTION ENGINEERING & DESIGN

# Training Assistant

## High-Level Design Document

v0.00

**Confidential**

Robosoft Technologies Pvt. Ltd.
Registered & Corporate Office
217, N.H.66, Santhekatte
New Udupi – 576 105
Karnataka, India
Tel: +91 820 2593930
Fax: +91 820 2593900

# Document Revision History

| Date | Version | Authors | Reviewed By | Approved By | Brief Description |
|---|---|---|---|---|---|
| <dd-Mmm-yy> | 0.01 | <Author> | <Reviewer> (One person/ Group Review - PQAL) | - | Initial draft |
| <dd-Mmm-yy> | 0.02 | <Author> | Verified by | - | List the changes incorporated as per review |
| <dd-Mmm-yy> | 0.03 | <Author> | <Customer> | <Customer> | List the changes incorporated as per Customer feedback and sent to Customer for approval |
| <dd-Mmm-yy> | 1.00 | | | | Approved by Customer and baselined |

# Contents

# Figures

# 1. High-Level Design

## 1.1 Introduction

### 1.1.1 Purpose

### 1.1.2 Scope

### 1.1.3 Audience

### 1.1.4 References

### 1.1.5 Terms and Abbreviations

## 1.2 Software Product Overview

## 1.3 Design Considerations

### 1.3.1 Assumptions and dependencies

### 1.3.2 Development methods

### 1.3.3 General constraints

## 1.4 Architectural Strategies

## 1.5 Software Product Architecture

## 1.6 Component Architecture

## 1.7 Bibliography

# 2. Reference

# 1. High-Level Design

## 1.1 Introduction

To develop solution for training assistant**.** Training assistant is a coordination of training functions and events, typically works in an office environment. Training assistant helps in training the employees or cross training of employees. Also keeps track of training progress and status.

We need to create system whereby we can create training schedule for employees and track their progress.

### 1.1.1 Purpose

The purpose of this High-Level Design document is to layout the design plan to represent a suitable model for coding. This document can be used as reference for how modules interact at a high level. It is also aimed at providing them with a complete reference for the development process and methodologies that will be used for the *Training Assistant* project

### 1.1.2 Scope

This document describes the structure of product, application architecture, development methodologies and application flow. This also describes design and development constraints of the product

### 1.1.3 Audience

This document is intended for the Customer, Executive Management, Project Manager and Development Team

### 1.1.4 References

1. Various online document related to training.

### 1.1.5 Terms and Abbreviations

| | |
|---|---|
| Front End | Vue Js |
| IDE | Visual Studio Code |
| MVC | Model-View-Controller |
| UI/UX | User Interface/User Experience |
| Back End | Node JS |

## 1.2 Software Product Overview

| Feature | Description |
|---|---|
| Login | This enables user to login to the application |
| Forgot password | Enables user to reset their password to access the Services. |
| Create Training | Admin can add create training programs and send Invitation to the employees. |
| Home page | This displays metrics of Training programs. Also list of training programs. |
| View Training Details | It will show particular training program details and list of employees' who accepted the training invitation |
| View Reportees | Manager can view his list of training repartees |
| Employee training details | This will show the training details of particular employee |
| Action View | Here employee can accept or reject the training invitation |

## 1.3 Design Considerations

### 1.3.1 Assumptions and dependencies

The following assumptions are made from the requirement.

1. Admin will create trainings and send Invitation to the employees
2. Manager can see all his reportees and there training status and can send Invitation to his reportees.
3. Admin can see list of training list and employee list
4. Admin and Manager can see employee training details.
5. Employee can accept or reject the training Invitation

### 1.3.2 Development methods

The application would be developed in Agile (via sprint) process.

### 1.3.3 General constraints

A major change in the requirements specification during the course of project development could have an impact on both technical design and the project schedule.

## 1.4 Architectural Strategies

1.Compilers/IDE/Languages: Visual studio, Vue js, JavaScript., Node Js, Bootstrap.
2. Source code management tool: Git
3. Database MySQL or  Postgress
5. Browser supported: IE11+  or above,chrome,Mozilla.
6. Communication: HTTPS.

## 1.5 Software Product Architecture

Modern software development needs to address complex business requirements. It also needs to take into account factors such as future extensibility and maintainability. A good design of a software system is vital to accomplish these goals. Design patterns would be used to organize the source code into different layers/component for maintainability and extensibility. The general advantages of using design patterns are,

1. Layer separation: Clear separation of responsibilities between components. This separation allows for an easier understanding and maintenance of the code base.
2. Modularity: Modularity allows to switch to a different implementation of view component in order to completely change application's UI, while all other components remain intact.
3. Easier testing: Since there are well defined boundaries between components, it becomes much easier to test each component in isolation (by e.g. mocking other components)
4. Unit testing : The purpose is to validate that each unit of the software performs as designed

Training Assistant will be developed using MVC design pattern

major aspects

1. Model:  The Model represents a collection of classes that explains the business model and the data model. It also defines the business rules for data means as how the data can be altered and manipulated

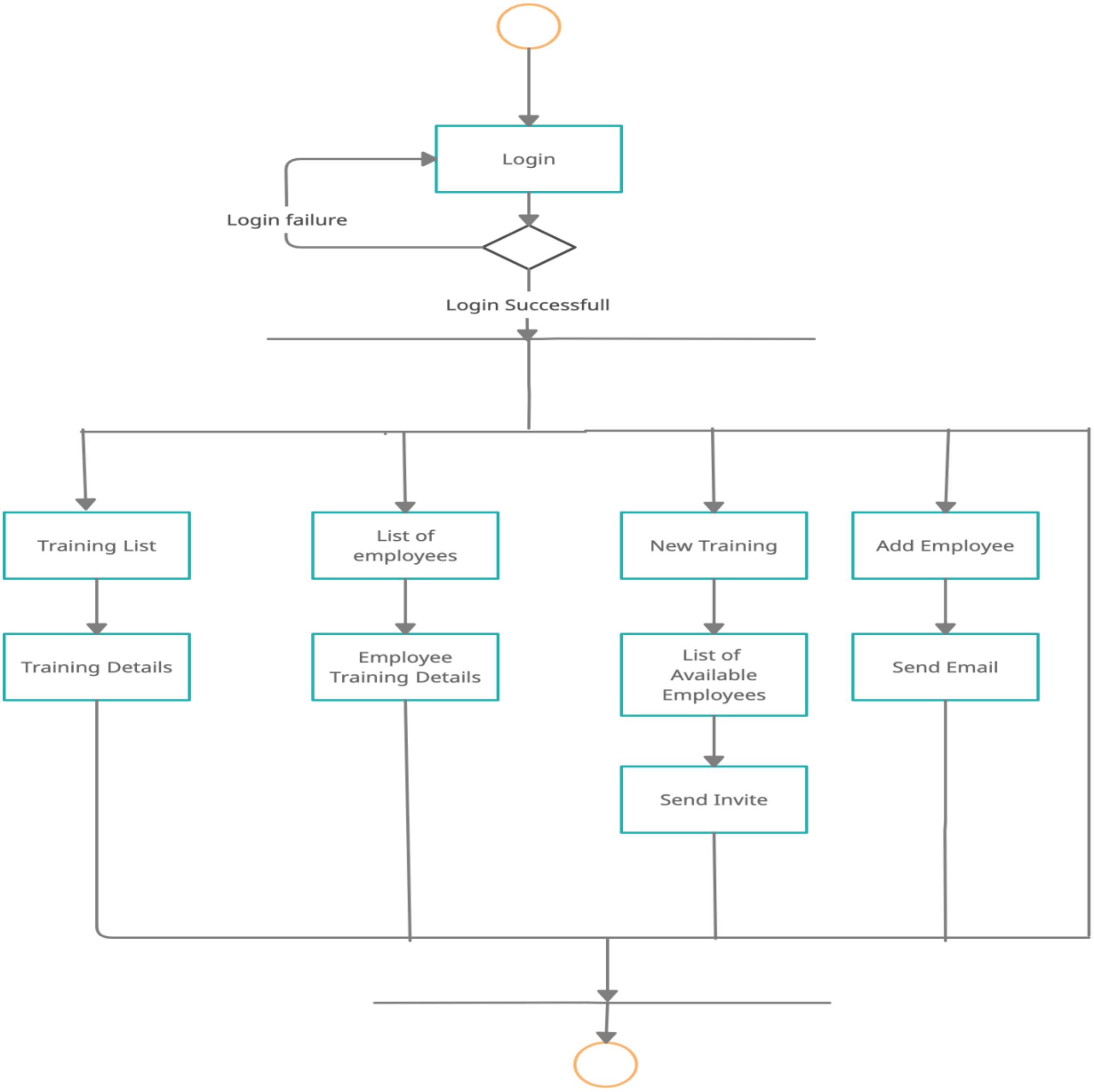
2. Controller: The controller plays major role. This layer does the all business logic and prepares the data to be displayed in UI.

3. View:  This represents UI.

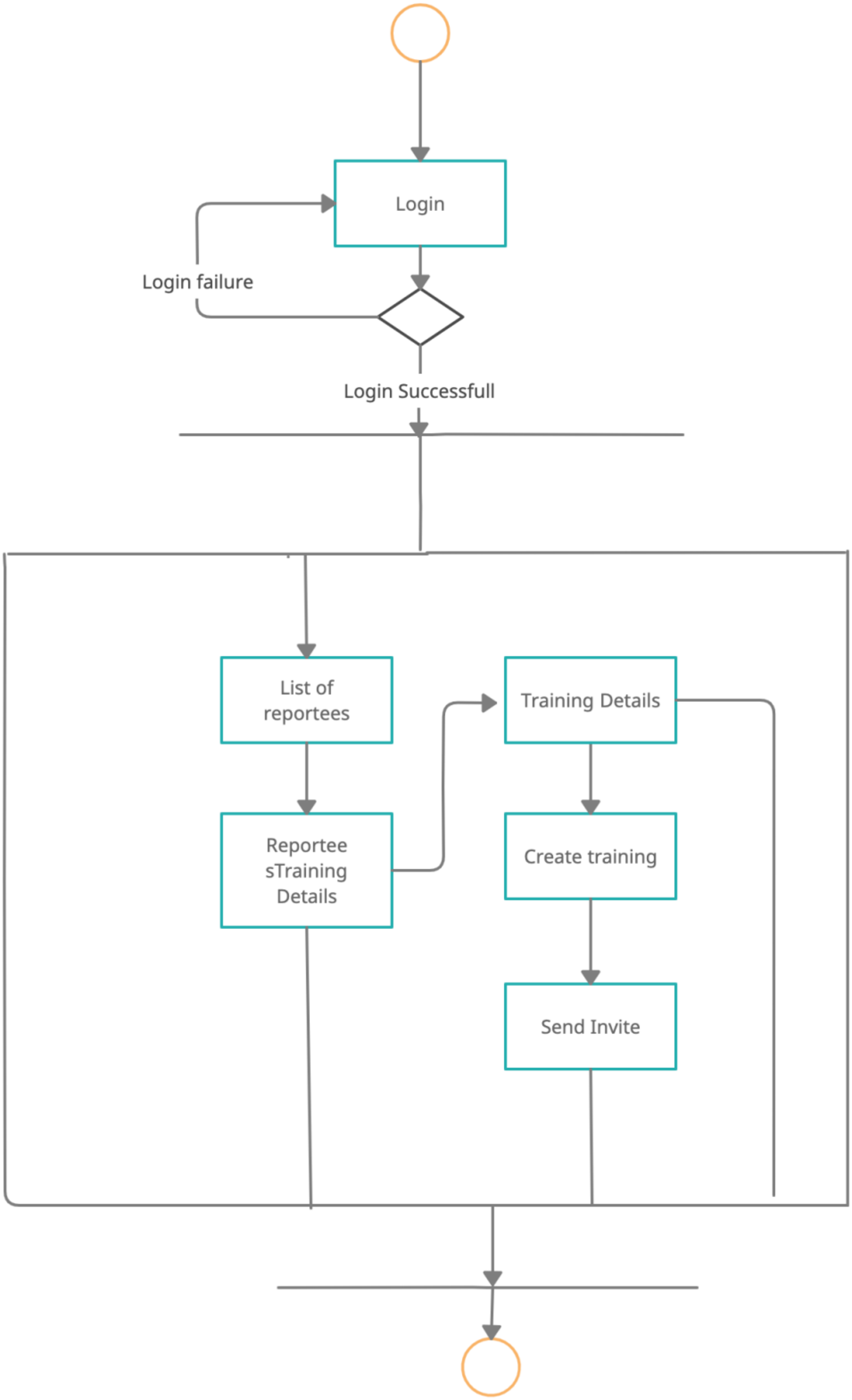
## 1.6 Component Architecture

### 1.6.1 State diagram

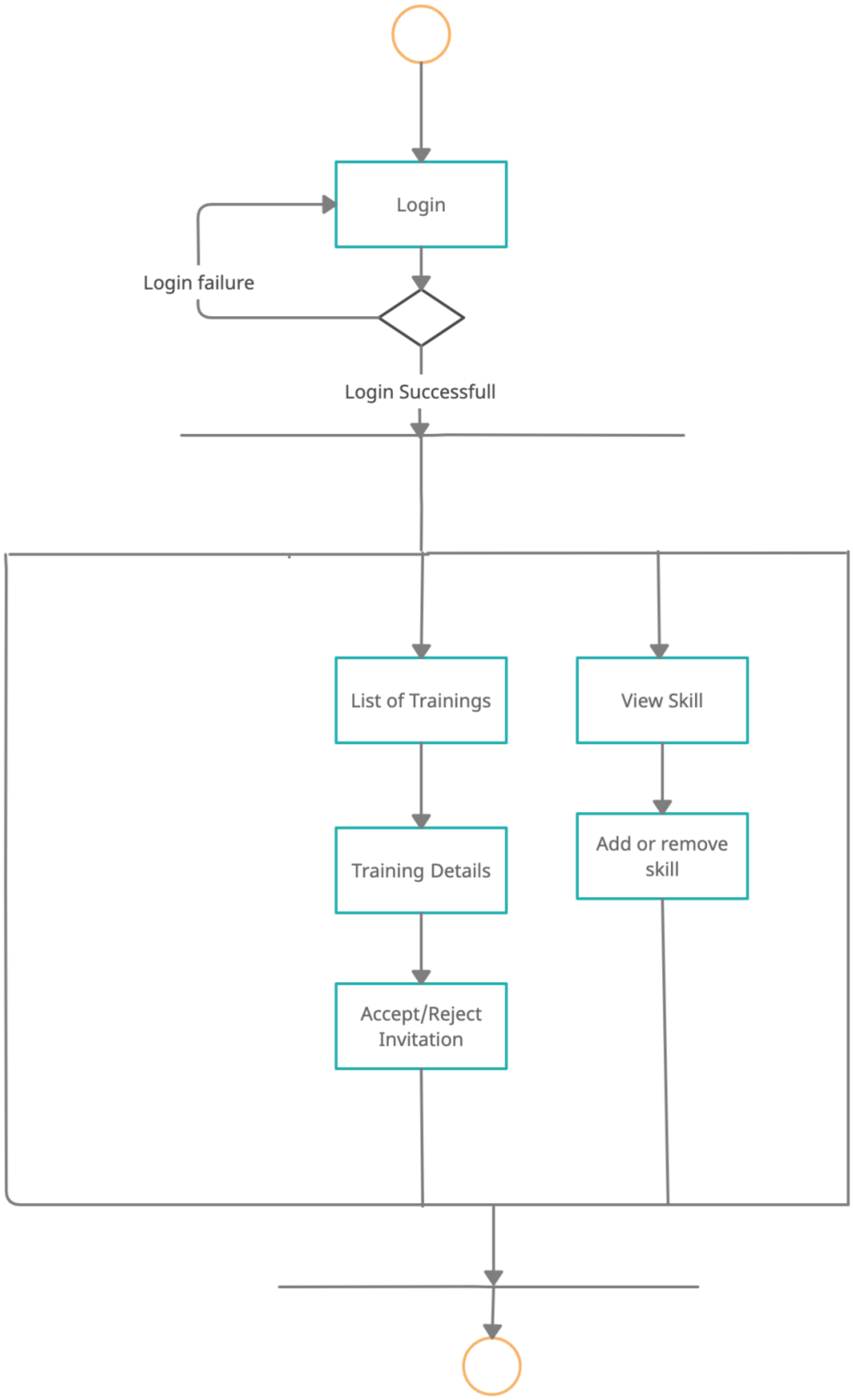
**a. Admin Flow:**



**b. Manager Flow:**

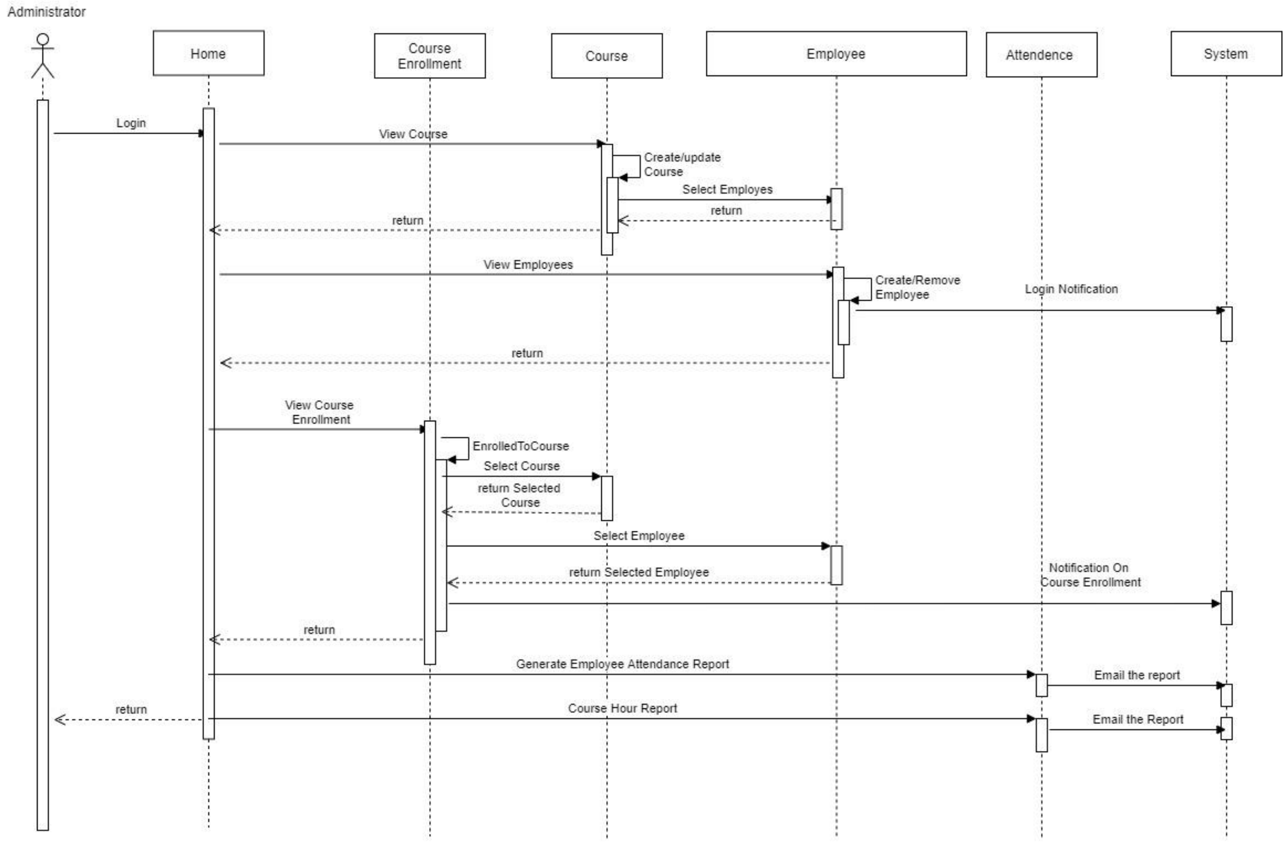


### C: Employee Flow

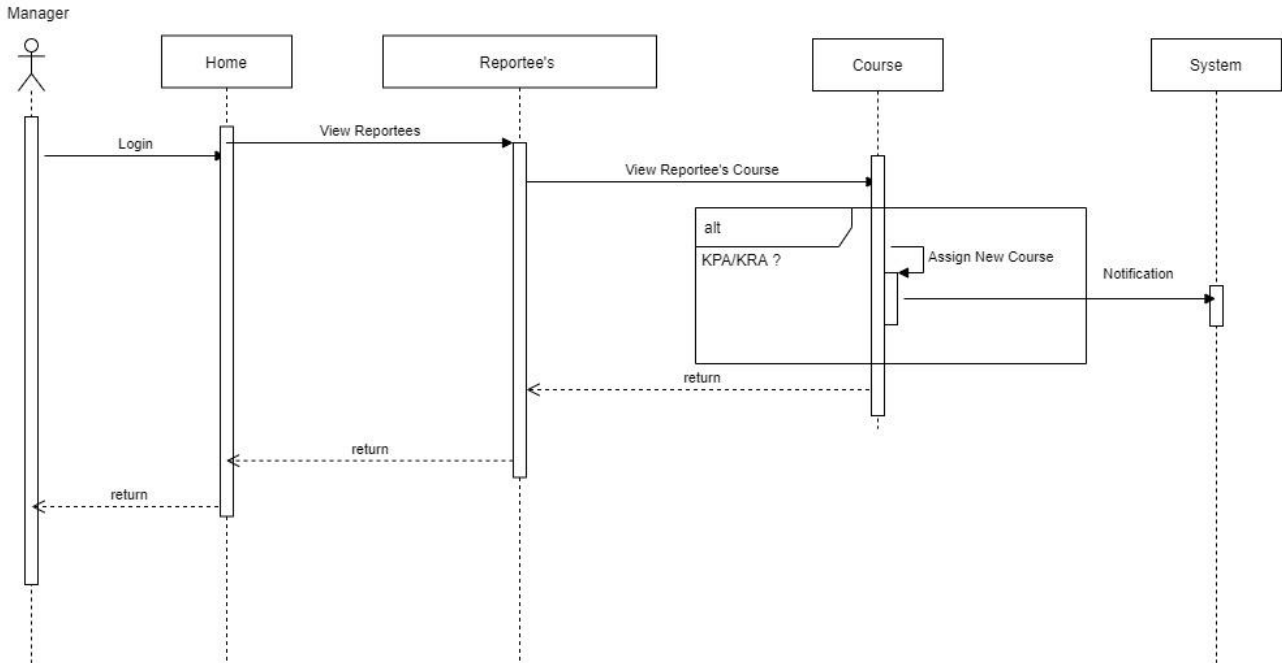


### 1.6.2 : Sequence Diagram

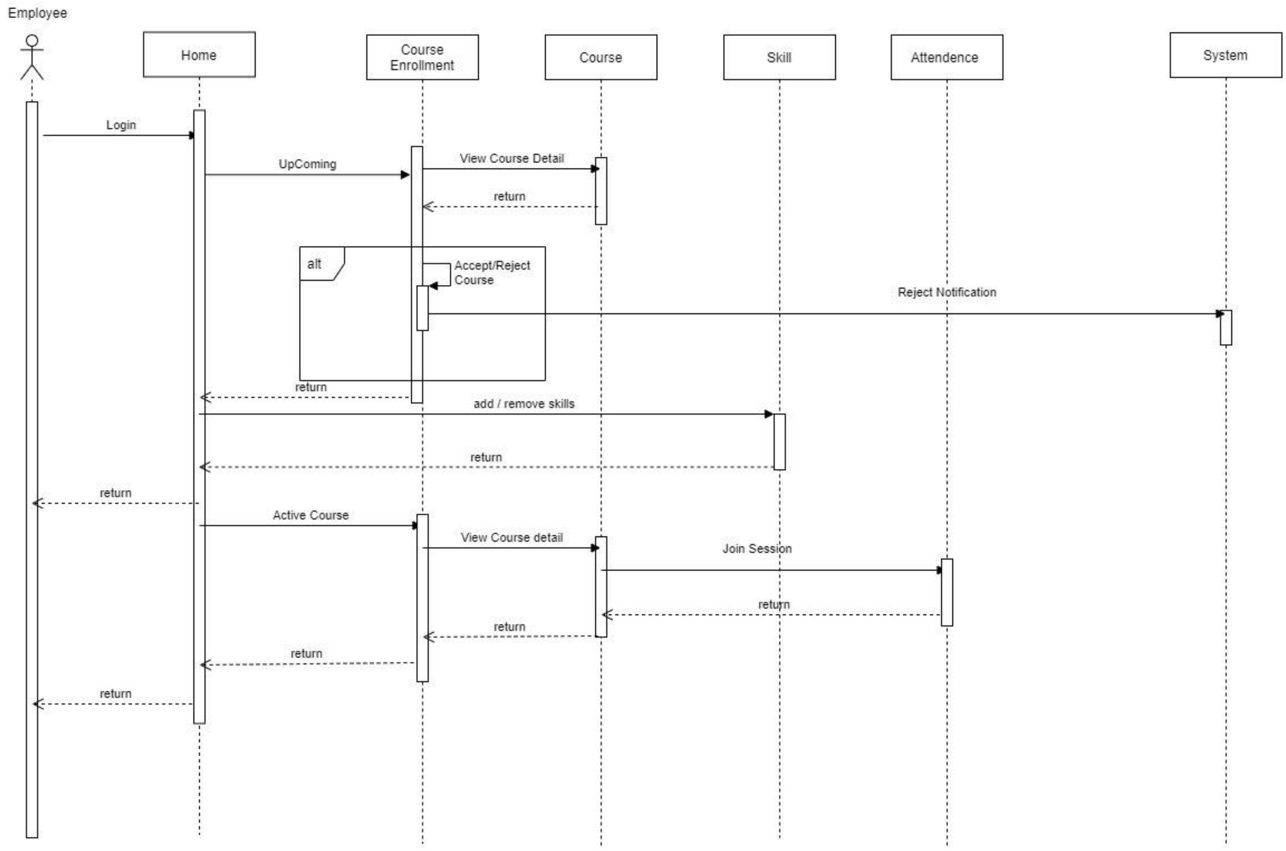
**Admin:**



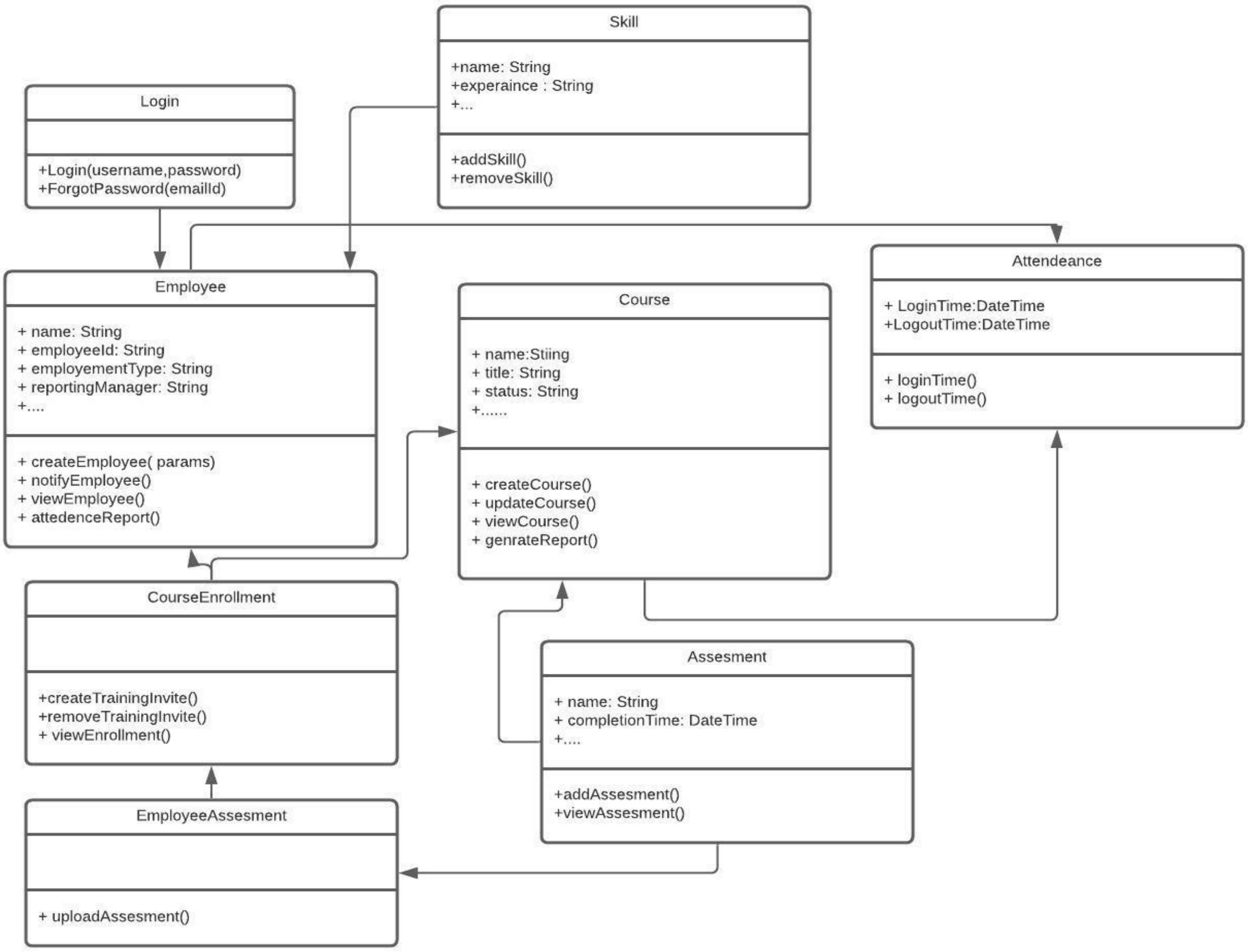
**Manager:**


Manager
Home
Reportee's
Course
System
Login
View Reportees
View Reportee's Course
alt
KPA/KRA ?
Assign New Course
Notification
return
return
return

### Employee:


Employee
Home
Course Enrollment
Course
Skill
Attendence
System
Login
UpComing
View Course Detail
return
alt
Accept/Reject Course
Reject Notification
return
add / remove skills
return
return
Active Course
View Course detail
Join Session
return
return
return
return

### 1.6.3 class diagram


Skill
+name: String
+experaince : String
+...
+addSkill()
+removeSkill()
Login
+Login(username,password)
+ForgotPassword(emailId)
Employee
+ name: String
+ employeeId: String
+ employementType: String
+ reportingManager: String
+....
+ createEmployee( params)
+ notifyEmployee()
+ viewEmployee()
+ attedenceReport()
Course
+ name:Stiing
+ title: String
+ status: String
+......
+ createCourse()
+ updateCourse()
+ viewCourse()
+ genrateReport()
Attendeance
+ LoginTime:DateTime
+LogoutTime:DateTime
+ loginTime()
+ logoutTime()
CourseEnrollment
+createTrainingInvite()
+removeTrainingInvite()
+ viewEnrollment()
Assesment
+ name: String
+ completionTime: DateTime
+....
+addAssesment()
+viewAssesment()
EmployeeAssesment
+ uploadAssesment()