SOFTWARE ENGINEERING PROJECT

END OF DOCUMENT

FACULTY OF ENGINEERING
AIN SHAMS UNIVERSITY
International Credit Hours Engineering Programs (i.CHEP)

Advanced Software Engineering Project

Due Date:

31/12/2021

Submitted to:

Dr. Gamal Ebrahim

Eng. Sally Shaker

Submitted by:

Omar Ashraf Mabrouk

Hussein Ahmed Hassan Selim

Abdelrahman Mohamed Salah

Zakaria Sobhy Abd El-Salam

Mahmoud Mohamed Seddik

Mohamed Ali Matar

Course Information

ASU Course Code CSE 232	***ASU Course Name*** Advanced Software Engineering
UEL Course Code EG 7421	***UEL Course Name*** Software Development 2
Semester Fall 2021	**Due Date** 31-12-2021

Participants Information

Full Name	ASU ID	UEL ID
Omar Ashraf Mabrouk	19P8102	2140624
Hussein Ahmed Hassan Selim	19P9614	2140571
Abdelrahman Mohamed Salah	19P9131	2140523
Zakaria Sobhy Abd El-Salam Soliman Madkour	19P2676	2140654
Mahmoud Mohamed Seddik	19P3374	2140584
Mohamed Ali Matar	19P5238	2140601

Abstract

This document contains the software analysis of an employee management system designed and developed by this team members. The idea of this software was inspired from the great diversity and massive number of employment cases in single firm, so an urge to control, manage, and aid these employees has risen. The general description of the software and its requirements are discussed thoroughly in sections two and three.

The reader of this document needs to be familiar with basic software concepts and diagrams as the system is modeled using mainly UML diagrams such as use-case diagram, class diagram, sequence diagram, state diagram,… as well as NON-UML diagrams as CRC diagram, context diagram, DFD diagram. This document will be of great help for developers who wish to modify, or add functionalities to the system, or integrate it with other systems, as it will give them a profound intuition of the underlying components of the system, their connetions, and their functionalities. Finally, this document could be taken as a case study for software engineering students as it applies all the object-oreinted analysis and design methodologies.

Table of Contents

1.0 Introduction

1.1 Purpose

The company have sub-companies each having its divisions and under each division there is a vast personnel structure. Our Employee Management System (EMS) is a stand-alone desktop application with Database API that stores the data on an online server. The software contains the needed key requirements and features present in any HR system and is applicable to many companies and businesses. This software helps organize, supervise, and monitor the personnel framework in any professional corporation.

1.2 Scope

This program is considered a powerful aid in organizing and managing a company structure either in its personnel management level or in its professional organization. Aside from its basic HR functions it has a task management section to help schedule and organize tasks, it also includes internship management section as well as talent enhancement section to help improve the employees' abilities.

1.3 Overview

The first few sections of this document discuss the required specifications of the software included until the use-case modeling. All these sections could be considered as the SRS document. In the next part the internal design of the system is discussed thoroughly through a sequence of diagrams in the following order. First class modelling which includes the detailed classes structures, their relationships as well as the CRC diagram. Then the swimlane and component diagrams followed by the sequence and state diagrams. Eventually we can take a look over the big picture of this software through both the context and DFD diagrams, and the system architecture. Finally, we shed light on the object-oriented methodologies used and then preview the testing performed on the system. The last section includes some decision-making studies such as the cost estimation of the software developed along with the time plan.

2.0 General Description

This program is considered a powerful aid in organizing and managing a company structure either in its personnel management level or in its professional organization. Aside from its basic HR functions it has a task management section to help schedule and organize tasks, it also includes internship management section as well as talent enhancement section to help improve the employees' abilities.

2.1 Product Perspective

This software will have the basic skeleton to perform the most common tasks of the HR department. Having said that, the software can be integrated with various systems such as a banking system to automatically pay the employees their base salary plus any extra compensations such as bonuses and overtime. It can also send personalized notifications/emails to the employees with latest news about the company and their team. This software isn't limited to any company as it can evolve to fit into any needed foundation.

2.2 General Capabilities

2.2.1 Login system to authenticate the access of potential users to the system.

2.2.2 Employee information management

- Admin level panel.
- Manager level panel.
- Recruitment and hiring employees (add/delete).
- Employee records and database.
    - Experience and Qualifications
    - Job history
    - Job description
    - Division
    - Salary history
    - Insurance plans

- Banking and taxes
- Type of employment (international/ domestic, full-time/ part-time)

2.2.3 Talent management

- Employee performance rating.
- Reviews and feedback.

2.2.4 Financial management

- Payroll.
- Bonus.
- Compensations.

2.2.5 Benefits management (Optional)

- Health insurance.
- Life insurance (if needed)
- Paid time off.
- Retirement plan and pension

2.2.6 Internship and training (Optional)

- Apply for internship.
- Issue certificate.
- Apply for further training

2.2.7 Task management (Optional)

- Reminder of meetings/deadlines
- Scheduling
- Project organization

2.3 General Constraints

This software has been designed using java programming language. It must be easily used by non-experts having no previous training to use the software; having a user-friendly interface to help users while using the software. Compatibility with any system and scalability of the software are

two highly emphasized aspects, additionally, the maintenance of the software should be easily carried out without any problems after the implementation has been done. The software security should be at the highest levels to create this highly reliable, responsive, and secure software.

2.4 User Characteristics

There are 2 main user classes:

-Managers

-Employees

Ordered from most frequent accessor to the lowest; for managers, this characterizes the ability to view, manage and control employee salaries, bonuses, promotion/demotion, and tasks; for HR managers, this characterizes the ability to view and analyze tasks (performance metrics) for employees, view financial records and sometimes manage employee data with the acceptance of a manager; for employees this characterizes the ability to view tasks, salaries, and bonuses.

2.5 Operating Environment

The software will be compatible with windows 7/8/10 and will be able to perform without any major errors. It requires at optimum 500 MB of RAM and Intel Core I3 3rd generation or its equivalent for an overall excellent performance, the environment need to have access to the internet for the software to make API calls to our cloud server which will modify the database according to the action taken by the user. It is advised to keep the computers in room with a temperature not exceeding 27 Celsius degrees and the humidity should not exceed 60% for optimum performance.

2.6 Assumptions and Dependencies

Since the system uses a custom designed API which is hosted on a third-party server (Heroku), the software will be operational if there's at least 10 API calls per month for the API to be

functional plus the software uses a NoSQL database type (MongoDB) which is hosted on a third-party service (MongoDB Atlas) so it can be accessed.

3.0 Specific Requirements

3.1 Functional Requirements

- Login System:

  The user of the system must enter his own username and password which was previously determined by employee, admin. So that the system could limit his privilege on using of the data this is to prevent unintended use of features, if a user has forgot his password, then he is able to change a user's password

- Talent Management:

  When the employee finishes his tasks, the program will record that he finished it in a certain time and then it will calculate how well he is performing by measuring the number of tasks that he has done in a certain day and average of this in a week (how long have he been doing tasks at the same rate?)

  The employee can assess his work, his managers also can review his performance and if this employee does an amount of work per day exceeding a certain criterion, then this employee will be rewarded either prioritizing him in the upcoming salary raise and in the yearly bonuses and by showing the HR and his managers his outstanding performance or if there is a slow-down in his performance then HR and his managers should be aware and take actions.

- Financial management:

  The managers should be able to review--financial records--and know when the monthly payment will be and how the money will be spent and where(while calculating the profit the company has gained), and if there's a specific salary decrease or an increase for a specific employee, the program should calculate the bonus of each employee upon his accomplishments and his performance (measured in functional requirement 2.) that he have done through the year, It also should take into account the request for salary increase done by any employee and this should be reviewed by his manager by accepting

or declining it, and should know from a manager if a certain employee has to have a compensation due to certain situation.

Employee information management

The core of this program of course is to manage and organize the information of the employee involved in a company's framework as follows:

a- The HR employee can add a new employee into the database of the company or removing an existing one already after getting the approval of the company manager. The process of adding a new employee involves creating a new employee object recording their basic data (Name, Address, Email, Birth date, Mobile number, type of employment - international/ domestic, full-time/ part-time, job description, salary, pension plan package, medical insurance-if exists, division, and contract)

b- The employee could access and review his information, for example an employee can get his contract information, or his salary, or update his certification and training. These field cannot be modified by the employee but requires the HR employee to be changed. On the other hand, the employee can change some data such as his address, his phone number).

c- The HR employee can access and modify the employee data if necessary, after getting the required approval from the manager. For instance, the HR employee can modify an employee's salary after confirmation from the parties considered in taking such decision.

	HR Employee	Ordinary Employee
Name		
Address		Yes
Contact info. (Phone Number/ Email)		Yes
Birth date		
type of employment	Yes	
job description	Yes	

salary	Yes	
pension plan package	Yes	
contract information	Yes	
division	Yes	
Additional training		Yes

d- The employee can file a request for salary raise after providing the sufficient requirements for such a raise.

e- The program organizes the daily tasks of the employee and send a notification when a deadline is near or when there is a meeting scheduled.

f- The employee can check his balance of paid vacations with the database.

g- The employee can apply a request for a vacation whether paid or unpaid one, the request is sent to the HR and the employee's direct manager within an hour from filing it and await approval within maximum 5 days from the concerned parties.

h- The employee could apply for promotion after providing the required documents to the system.

Benefits Management

Health insurance:

The information of the health insurance company including the name, the telephone number and official mail is stored. The medical facilities at which the health insurance is applicable is also stored.

Life insurance:

The information of the life insurance company including the name, the telephone number and official mail is stored.

Paid time off:

The paid time off allowed in general is 20 days per year not including paid medical leaves allowed. A counter is used to count days taken off by the employee. As long as the employee

has 20 or less days taken off no deduction from the salary occurs. When the employee exceeds the allowed 20 days the financial department is alerted and deductions from the salary occur.

Retirement plan and pension:

The agreed amount of funds for the employee's future benefit is stored. The pool of funds is invested on the employee's behalf, and the earnings on the investments generate income to the worker upon retirement.

Internship and Training

- Apply for internship:

An external person can apply for an internship in the company. He enters name, age, email, telephone number, current job, and the applied position. The system checks if a vacant spot for the requested training is available. He then is registered in the system after being accepted.

- Issue certificate:

The intern can request to issue a certificate in which two given options are given either be given a date for pickup or request it to be shipped.

- Apply for further training:

A registered intern can reapply for additional training.

Task Management

- Reminder of meetings/deadlines:

An email is sent to members of a certain project teams reminding them of important meetings or deadlines. The time this reminder is sent is determined by the admin it can be any number of days chosen before the specified date of the meeting or deadline.

- Scheduling:

A manager sets a schedule for meetings, projects, milestones, or deadlines. Automated email of the time of scheduled meeting to the employee. It also checks if an employee would have a clash between two meetings occur at the same time.

- Project Organization:

Responsible for storing a project information. Including project name, manager name, and employee's name, role of each employee, time frame and the deadline of the project.

3.2 Non-Functional Requirements

1- Does not require high processing abilities as it can be run on any processor whatever how primitive and basic it is single core processor is enough.

2- Does not need much memory (does not exceed 500 MB.)

3- Need to be connected to the internet to access the on the cloud database.

4- Maximum response time is required to be 100 ms.

5- The application needs to be developed in Java.

6- The program is required to be user-friendly as it does not need more than half an hour to be able to use it with ease.

7- Need to be developed within nine weeks.

8- Scalable as it requires some small modifications, and it will fit to any organizational professional framework.

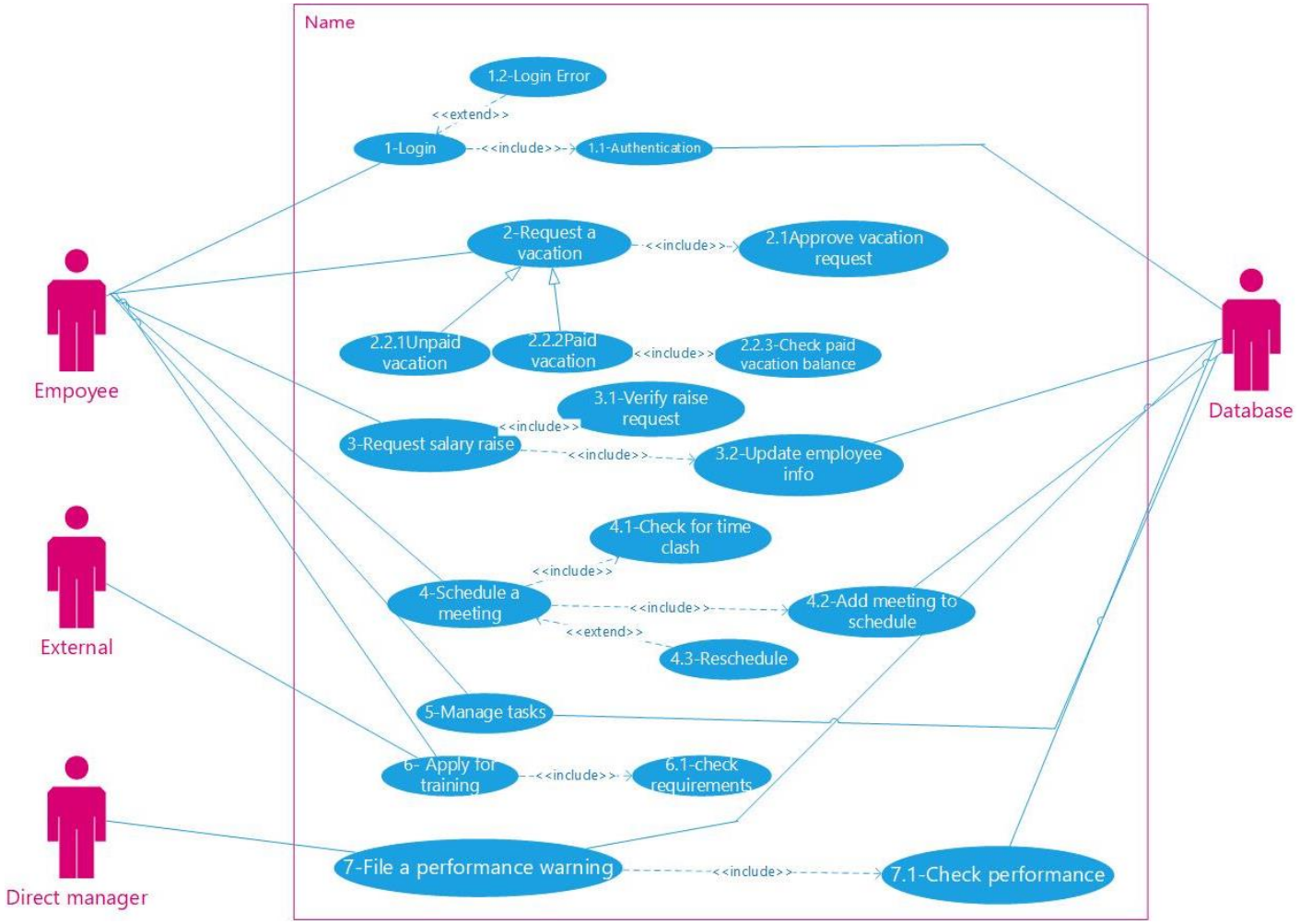
9- Compatible with any system if the JVM (Java Virtual Machine) is installed on that device.

10- Reliable as it need to have mean time of failure 1 every three month.

11- Maximum database access time within one day is 18 hours.

4.0 USE-CASE MODELING

4.1 Employee Use-Case Diagram



Use-Case Descriptive Narative

Use-Case Name	1-Login
Related Requirements	
Goal in Context	Log the user into his account
Preconditions	Program is installed and the user got his username and password
Successful end condition	Logged into the users account
Failed end condition	Can't login either due to wrong username, or password, or username/password mismatch
Primary Actor	Employee
Secondary Actor	Database
Trigger	Pressing login button
Included use-case	Authentication
Main Flow	1) The user enters the username and password. Include: Authentication 2.1) the username is checked if it exists in the database 2.2) if the username is found it is checked with the password registered 2.3) then the input data by the user are checked to be matching with that in the database 3) The user is then taken to the main screen after logging in
Extension	4) The user can't log in due to mismatching username and password

Use-Case Name	1.2- Login Error
Related Requirements	1-login
Goal in Context	Define the error in the logging in process to the user
Preconditions	Login is tried
Successful end condition	The user logs into the system
Failed end condition	The user can't log in
Primary Actor	Employee
Secondary Actor	----------
Trigger	Failed to log in
Main Flow	1) The username and password aren't matching. 2) The user is asked to re-enter the username and password. 3) If the process fails, more than twice the user is asked to go back to the program's admin

Use-Case Name	2-Request a Vacation
Related Requirements	Paid/Unpaid Vacation
Goal in Context	Apply for a vacation so that it could be verified
Preconditions	Be logged into your account
Successful end condition	Vacation request is sent successfully to whom it may concern
Failed end condition	Failed to send vacation request
Primary Actor	Employee
Secondary Actor	Database, HR
Included use-case	Approve vacation request
Main Flow	1) The user inputs the start/end date of the vacation. 2) The start and end dates are checked to be upcoming, and the end date is after the start date. 3) The request is then sent to the database to notify the HR employee responsible for approving the vacation request. 4) The respond of the HR employee is sent to the database and a notification is sent to the requestor.

Use-Case Name	2.1- Approve Vacation Request
Related Requirements	2-Request a Vacation
Goal in Context	Give a respond to a vacation request
Preconditions	A vacation request is filed by an employee
Successful end condition	A respond is delivered to the requestor
Failed end condition	Not respond is sent to the request
Primary Actor	HR Employee
Secondary Actor	Database, Employee
Trigger	A vacation is requested, and a notification is sent to the concerned HR employee
Main Flow	1) The notification of an Employee that requested a vacation is sent from the database to the concerned HR employee. 2) The employee responds to the vacation request either by approval or denial. 3) The respond is then registered in the database. 4) A notification with the respond is then sent to the vacation requestor

Use-Case Name	2.2.2- Paid vacation
Related Requirements	2- Request a vacation
Goal in Context	Request a vacation that is fully paid
Preconditions	A vacation is requested
Successful end condition	A paid vacation request is sent
Failed end condition	Failed to send the request
Primary Actor	Employee
Secondary Actor	Database, HR employee
Base use-case (inherited from)	2- Request a vacation
Included use-case	2.1) approve vacation request
Main Flow	1) The user requests a vacation as in the base use case by entering the start and end date. 2) The user is asked to enter the type of vacation requested which is paid. 3) The start and end date are sent to the database and the duration is calculated. 4) The requested duration is then compared with the paid vacation balance if there is enough balance a request is sent to the HR employee. Include: approve vacation request 5) After the vacation is reviewed and a respond is registered in the database the respond is sent to the user

Use-Case Name	3- Request Salary Raise
Related Requirements	Update user info
Goal in Context	Send a request to the concerned person about a salary raise
Preconditions	Be logged into the system
Successful end condition	Salary raise request is sent successful
Failed end condition	Failed to send salary raise request
Primary Actor	Employee
Secondary Actor	Database
Included use-case	3.1-Verify raise request 3.2-update employee info
Main Flow	1) The user enters the requested salary to be raised. 2) The requested salary is saved to the database. 3) The salary request is sent from the database to the concerned HR employee. Include: Verify raise request 4) The HR employee either approve or deny the request. 5) The respond of the HR employee is then saved to the database.

	Include: update employee info 6) If the request is approved, then the salary of the employee is updated to the new value in the database

Use-Case Name	4- schedule a meeting
Related Requirements	----------
Goal in Context	Add a meeting to the Schedule
Preconditions	Be logged into your account and have the authorization to schedule a meeting
Successful end condition	Meeting scheduled successfully
Failed end condition	Can't schedule a meeting
Primary Actor	Employee
Secondary Actor	Database
Included use-case	4.1- check for time clash 4.2- add meeting to schedule
Main Flow	1) The employee schedule a meeting by entering the date of the meeting. Include: check for time clash 2) Save the meeting time to the database. Include: add meeting to schedule 3) Send a notification to all concerned employees about the meeting

Use-Case Name	4.1- Check for time clash
Related Requirements	4- Scheduling a meeting
Goal in Context	Check if there is a clash between a new meeting which is to be registered and an already scheduled meeting
Preconditions	A date for a meeting is entered by a user
Successful end condition	There is no clash you can schedule the meeting
Failed end condition	There is a clash with a list of meetings maybe you would consider rescheduling the meeting
Primary Actor	Employee
Secondary Actor	Database
Trigger	Schedule a meeting
Main Flow	1) Check if there is an already scheduled meeting having the same time as that which is to be rescheduled. 2) If there is a clash suggest other time to schedule the new meeting, if there is not sent a message that there is no clash.

	3) Ask the user to choose a new date for the meeting and repeat the steps 1 and 2 again till you reach a valid date for the meeting

Use-Case Name	6- Apply for training
Related Requirements	----------
Goal in Context	The employee wants to file a request for training
Preconditions	Meet the training requirements
Successful end condition	Request sent
Failed end condition	Failed to send request
Primary Actor	Employee
Secondary Actor	Database
Included use-case	6.1- Check Requirements
Main Flow	1) The user inputs the required data for applying for a certain training. 2) The data input by the user are checked with the requirements. Include: Check Requirements 3) If the employee meets the required qualities, then the system should accept his request and save it to the database. 4) If the user does not meet the requirements send a notification with the fields that doesn't meet the requirements 5) In both cases send a notification of the status for the request.

Use-Case Name	6.1- Check Requirements
Related Requirements	6- Apply for training
Goal in Context	Check the requirements for applying into a certain training program to see if the applicant meets the requirements or not
Preconditions	An employee has applied for a training and provided the required entry data.
Successful end condition	Applicants' data checked against required criteria
Failed end condition	Failed to check applicant's data
Primary Actor	Employee
Secondary Actor	Database
Trigger	An employee has applied for training
Main Flow	1) Fetch the applicants' qualifications from the database. 2) Fetch the minimum qualifications required to enter the training program from the database. 3) Compare between the applicants' qualifications and minimum entry qualifications required.

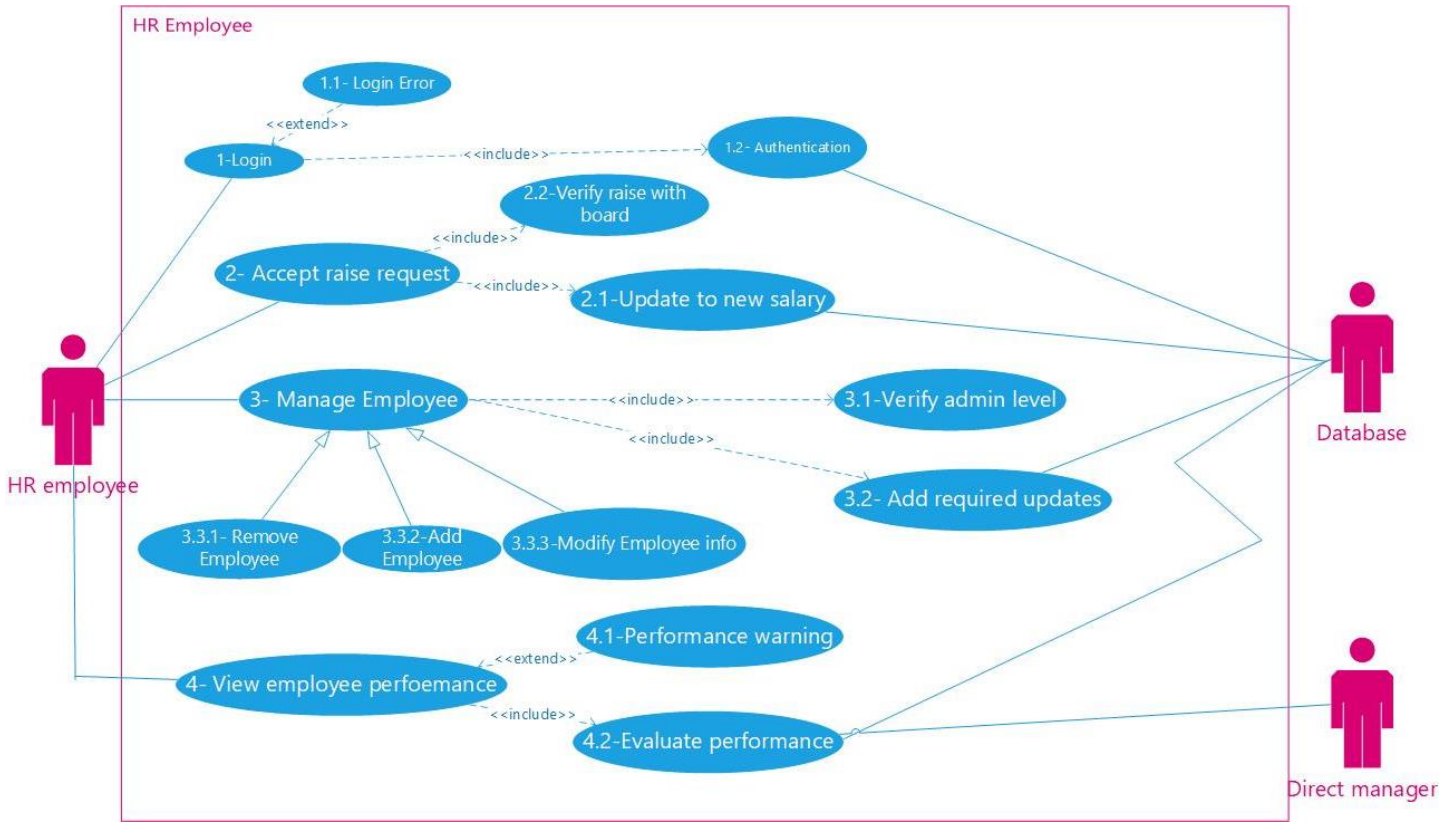
	4) If the applicants' qualifications are as or more than required accept his application 5) If the applicants' qualifications are less than required in some fields send a notification to the user with the fields need to be improved 6) Save the systems respond in the database. 7) Send the respond to the applicant

Use-Case Name	7- File a performance warning
Related Requirements	----------
Goal in Context	Send a warning to an employee due to his bad performance
Preconditions	Performance is calculated and a certain standard is provided for the system
Successful end condition	Warning is sent successfully to the employee
Failed end condition	Failed to send the warning
Primary Actor	Direct Manager
Secondary Actor	Database, Employee
Included use-case	7.1- Check Performance
Main Flow	1) Employee's performance is feed to the system. Include: Check Performance 2) If the performance is less than the standard the direct manager is notified 3) A warning should be sent to the employee. 4) A history of the warnings sent to the employee is saved in the database as a reference.

Use-Case Name	7.1- Check Performance
Related Requirements	7- File a performance warning
Goal in Context	Assess the performance of an employee
Preconditions	The employee's performance is provided to the system and a given standard is saved in the database
Successful end condition	Performance is checked successfully, and the result of the check is saved into the database
Failed end condition	Failed to conduct performance check
Primary Actor	Direct Manager
Secondary Actor	Database
Trigger	The system is asked to file a warning
Main Flow	1) The employee's performance is fetched from the database. 2) The standard performance is either entered by a manager or from a stored value in the database.

	3) The employee's performance is compared to standard performance. 4) The result of the check is then stored into the database

4.2 HR and Manager Use-Case Diagram



Use-Case Descriptive Narative

Use-Case Name	2- Accept Raise request
Related Requirements	
Goal in Context	The HR employee can accept a salary raise request made by an employee
Preconditions	A raise request is made by an employee
Successful end condition	The HR employee has made a clear decision in whether to accept or refuse the salary raise request and upon which the salary is updated
Failed end condition	
Primary Actor	HR employee
Secondary Actor	Database
Trigger	An employee has requested a salary raise
Included use-case	2.1-Update to new salary/2.2-verify raise with board
Main Flow	1. The system prompts the HR Employee that a certain employee has requested for a salary raise. 2. The HR will review the employee's request. 3. The HR will have meetings with the board to verify the salary raise.

	4. Once the change in salary is agreed upon ,the system will send the new salary to Database Include::Update to new salary

Use-Case Name	2.1- Update to new salary
Related Requirements	----------
Goal in Context	Change the value of the salary for an employee in Database
Preconditions	The salary raise is accepted by HR
Successful end condition	The value of salary is changed successfully
Failed end condition	There is an error in connection with the database
Primary Actor	Database
Secondary Actor	----------
Trigger	The employee salary is changed
Main Flow	1) Search the database for the employee by his username. Update the Employee salary to match the updated salary

Use-Case Name	3- Manage Employee
Related Requirements	2.2.2 Employee information management
Goal in Context	The HR employee can add/remove or modify employee info
Preconditions	The HR employee should have an admin level privilege
Successful end condition	Data is updated/created or removed successfully, and those modifications is accepted by the database.
Failed end condition	Data is not updated/created or removed properly, or the updating employee has no admin level privilege.
Primary Actor	HR employee
Secondary Actor	Database
Trigger	HR has opened the employee editing screen
Included use-case	Verify Admin level, add required updates
Main Flow	1) The HR opens the employee editing screen. Include: Verify Admin level. Include: add required updates.

Use-Case Name	3.3.1- Remove Employee
Related Requirements	2.2.2 Employee information management
Goal in Context	The HR employee can remove an employee after he is efforts is no longer needed
Preconditions	The HR employee should have an admin level privilege

Successful end condition	Employee Data is removed successfully from the system
Failed end condition	----------*
Primary Actor	HR Employee
Secondary Actor	Database
Trigger	HR employee chooses to remove an employee from the system
Base use-case (inherited from)	3- Manage Employee
Main Flow	The HR selects the username of the employee he wishes to remove from the database.

Use-Case Name	3.3.2- Add Employee
Related Requirements	3- Manage Employee
Goal in Context	Adding a new employee to the system
Preconditions	The list of employees in the system is accessed
Successful end condition	The information of the employee is added to the list of employees and is stored in the database successfully
Failed end condition	Failure in adding the employee to the system
Primary Actor	HR-employee
Secondary Actor	Database
Trigger	The system is asked to add a new employee
Main Flow	1) the HR-employee opens the list of all employees. 2) he then chooses to add new employee. 3) the information of the employee is then entered. 4) the HR-employee then saves this information in the list and the database.

Use-Case Name	4- View Employee performance
Related Requirements	----------
Goal in Context	Assess the performance of an employee
Preconditions	The employee's performance is provided to the system and a given standard is saved in the database
Successful end condition	Performance is checked successfully, and the result of the check is saved into the database
Failed end condition	Failed to conduct performance check
Primary Actor	HR Employee
Secondary Actor	Database
Trigger	The system is asked to view employee performance
Main Flow	1) The employee's performance is fetched from the database. 2) The standard performance is either entered by a manager or from a stored value in the database.

	3) The employee's performance is compared to standard performance. 4) The result of the check is then stored into the database

Use-Case Name	4.1- Performance warning
Related Requirements	4- View Employee performance
Goal in Context	Issue a performance warning to an employee whose performance is lacking
Preconditions	1) The employee's performance is provided to the system and a given standard is saved in the database 2) The system finds the employee performance to be poor
Successful end condition	The warning is issued correctly, and the result of the issue is saved into the database and sent to the employee
Failed end condition	Failed to issue the warning
Primary Actor	HR employee
Secondary Actor	Database – employee
Trigger	The system is asked to file a warning
Main Flow	1) the employee performance is provided to the system 2) the HR employee checks the performance of the employee and compares it to his peers 3) if his performance is found to be lacking the warning is issued

Use-Case Name	4.2- Evaluate Performance
Related Requirements	4- View Employee performance
Goal in Context	Adding evaluation, the performance of the employee
Preconditions	The employee's information is stored in the system. The manager puts an evaluation of the employee
Successful end condition	Manager manages to update the employee performance and it is saved on the database
Failed end condition	Performance failed to be updated
Primary Actor	Direct Manager
Secondary Actor	Database
Trigger	The system is asked to evaluate a certain employee performance
Included use-case	4-view performance
Main Flow	1) The manager views the performance of a certain employee Include: view performance 2) the manager chooses to update the performance of the employee

5.0 Class Modelling

5.1 Noun Extraction

- **Stage 1**: Concise problem definition

  An employee management software is required to be developed which keeps records of the employees in an enterprise as well as their necessary information.

- **Stage 2**: Informal strategy

  This employee management system (EMS) helps organize the HR department of an enterprise. It keeps record of all the employees on the managerial and non-managerial level, it keeps their personal data (name, age, gender, phone number, address…), as well as their professional data as (experience and qualification, job history, job description, division, type of employment, salary, bonus, internships, trainings …), besides social data and benefits as (health insurance, life insurance, pension plan, banking and taxes, raise request, vacation request…).

  Beside these main functionalities the (EMS) provides secondary functionalities that can help the employees carrying on their work, as it allows them to set up teams, create tasks, monitor their performance, organize meetings and events.

- **Stage 3**: Formalize the strategy

  This employee management system (EMS) helps organize the HR department of an enterprise. It keeps record of all the employees on the managerial and non-managerial level, it keeps their personal data (name, age, gender, phone number, address…), as well as their professional data as (experience and qualification, job history, job description, division, type of employment, salary, bonus, internships, trainings …), besides social data and benefits as (health insurance, life insurance, pension plan, banking and taxes, raise request, vacation request …).

  Beside these main functionalities the (EMS) provides secondary functionalities that can help the employees carrying on their work, as it allows them to set up teams, create tasks, monitor their performance, organize meetings and events.

- **Finally**: Considering the candidate list

  employee management system → X too general

  HR department → X outside the problem boundary

  Enterprise → X outside the problem boundary

  Record → X too vague

  employees → √

  managerial → X could be as a flag in the employee class

  non-managerial → X could be as a flag in the employee class

  personal data → X too general

  name → included as an attribute inside the employee class

  age → included as an attribute inside the employee class

  gender → √

  phone number → included as an attribute inside the employee class

  address → included as an attribute inside the employee class

  professional data → too general

  experience → could be included inside the employee class

  qualification → could be included inside the employee class

  job history → could be included inside the employee class

  job description → could be included inside the employee class

  division → could be included inside the employee class

  type of employment → √

  salary → could be included inside the employee class

  bonus → could be included inside the employee class

  internships → could be included inside the employee class

  trainings → could be included inside the employee class

  social data → too general

  health insurance → outside scope (could be added in a system update if necessary)

  life insurance → outside scope (could be added in a system update if necessary)

  pension plan → outside scope (could be added in a system update if necessary)

  banking → too general

taxes→too general

raise request→√

vacation request→√

teams→√

tasks→√

performance→could be included inside the employee class

meetings→ redundant could be considered as an event

events→√

Possible list of classes

1. Employee
2. Raise request
3. Vacation request
4. Team
5. Task
6. Event
7. Gender
8. Type of employment

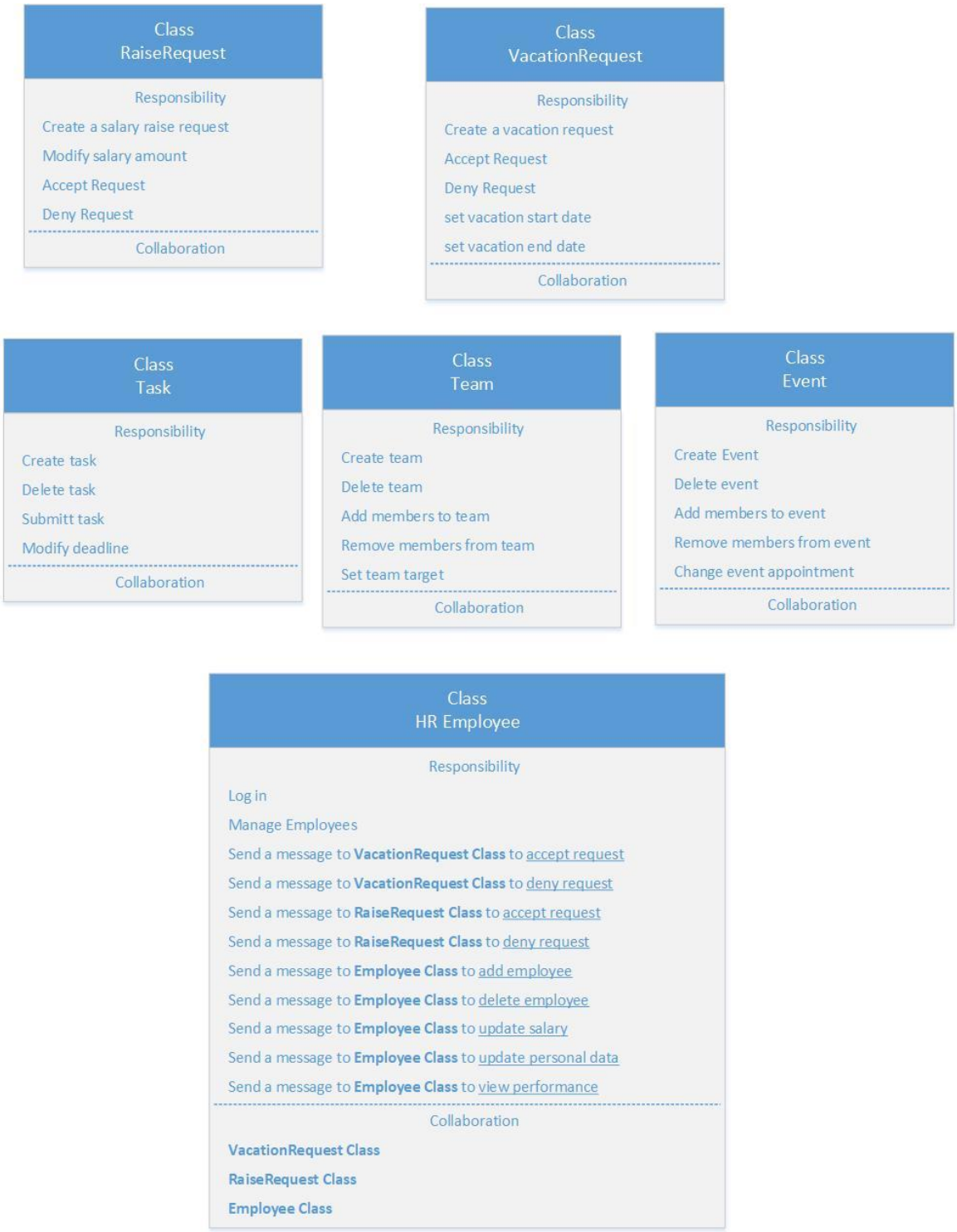
We can by recognition see that each of the raise request and vacation request may have many possibilites as this request may be 1) pending approval, 2) refused, 3) accepted, which will in turn make us add a new class called request status

9. Request status

Now that we have nearly finished the noun extraction we could proceed with the next step of the class design which is the CRC card and UML class diagram.

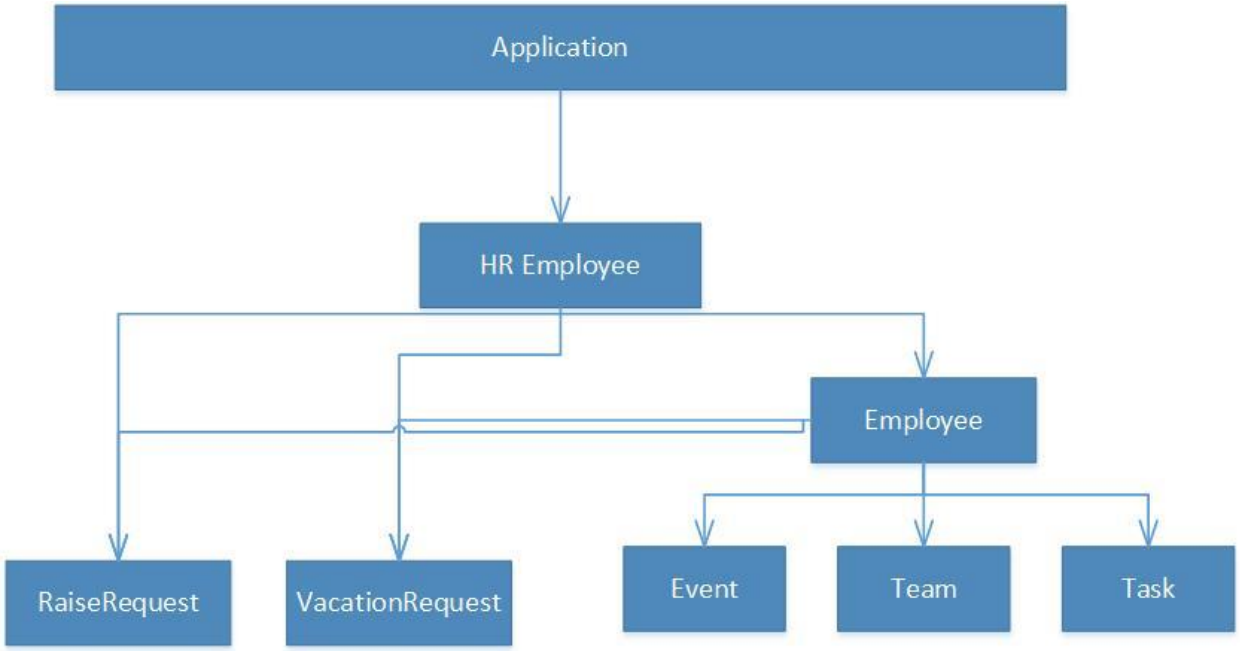
5.2 CRC Card

5.2.1 CRC Card Diagram



Class Employee
Responsibility
Add employee
Delete employee
Log in
Update personal data
Update salary
view performance
Apply for training
Send a message to **RaiseRequest Class** to create a salary raise request
Send a message to **RaiseRequest Class** to modify salary amount
Send a message to **VacationRequest Class** to create a vacation request
Send a message to **VacationRequest Class** to set vacation start date
Send a message to **VacationRequest Class** to set vacation end date
Send a message to **Team Class** to create a team
Send a message to **Team Class** to Add members to team
Send a message to **Team Class** to Remove members from team
Send a message to **Team Class** to Set team target
Send a message to **Task Class** to create task
Send a message to **Task Class** to Delete task
Send a message to **Task Class** to Submitt task
Send a message to **Task Class** to Modify deadline
Send a message to **Event Class** to create event
Send a message to **Event Class** to Delete event
Send a message to **Event Class** to Add members to event
Send a message to **Event Class** to Remove members from event
Send a message to **Event Class** to Change event appointment
Collaboration
RaiseRequest Class
VacationRequest Class
Team Class
Task Class
Event Class

5.2.2 Client-object relationship



So an instance of employee must be instantiated in the application program of the java application, which is logically correct since the employee is the primary actor of the system (i.e.: the employee instantiates the system use.)

5.4 UML Class Diagram

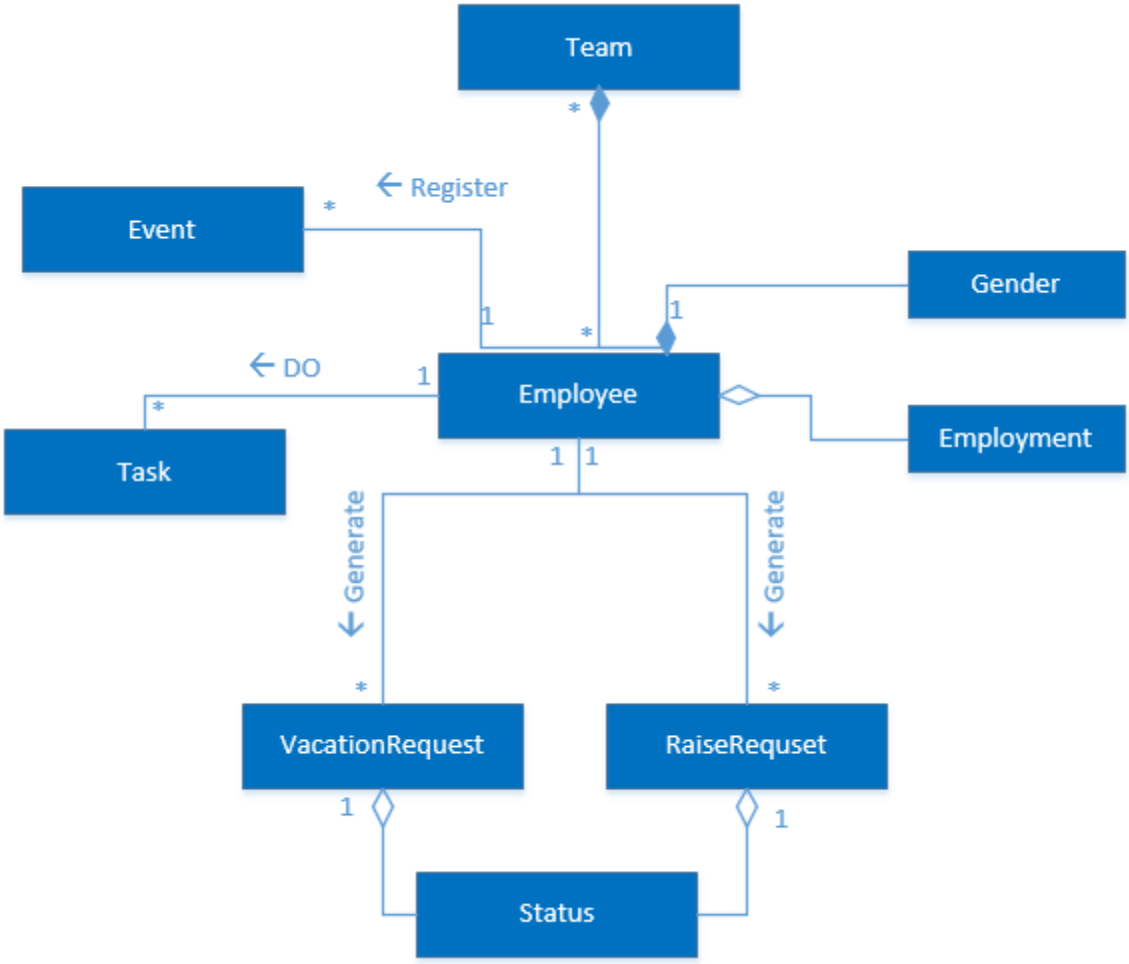
5.4.1 Detailed class diagram

Team
- employees : Employee[] -managerID : String - rating: float -teamID: String - tasks : Task[] - vacationRequests : VacationRequests[] - raiseRequests: RaiseRequest[]
+ Team(managerID : String, employees: Employee[]) +getEmployees(): Employee[] +getManagerID(): String +getRaiseRequests():raiseRequest[] +getRating():float +getTasks(): Task[] +getVacationRequests():vacationRequests[] +setEmployees(employees: Employee[]) +setManagerID(managerID: String) +setRaiseRequests(raiseRequest: RaiseRequest) +setRating(rating: float) +setTasks(tasks: Task[]) +setvacationRequests(vacationRequests: VacationRequest[]) +getTeams():ArrayList<Team> +selectTeams(teamID: String, team: Object):ArrayList<Team> +addTeam(team: Team):String +deleteTeam(team: Team):String +updateSpecificOfTeam(Team, String, Object):string

Employment	Gender	Status
fullTime	male	pending
partTime	female	accepted
	others	denied

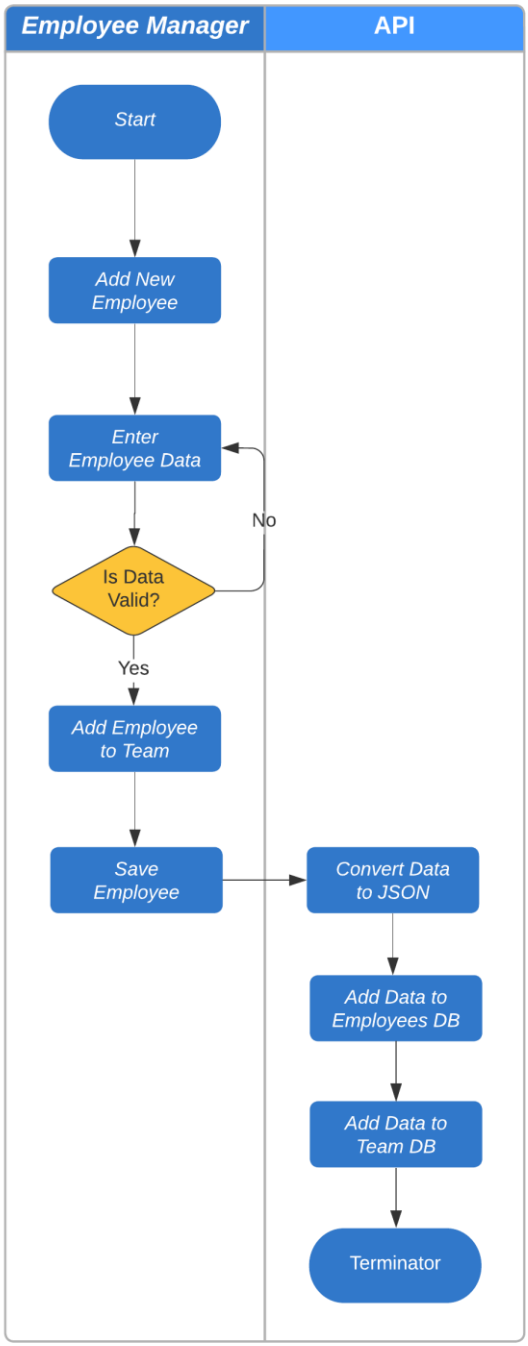
```
Employee
-name:String
-phone:String
-email:String
-id:String
-address:String
-nationality:String
-gender:Gender
-birthday:Date
-events:Event[]
-extraCompensations:double
-jobDescription:String
-pass:String
-possition:String
-rank:int
-rating:Float
-salary:int
-tasks:Task[]
-totalCompletedTasks:int
-username:String
-typeOfEmplloyment:Employment
-password:String
-vacations:VacationRequest[]
+Employee(name:String, gender:Gender,rank:int, pass:String, jobDescription:String,
typeOfEmplloyment:Employment, birthday:Date,nationality:String ,salary:int, username:String, possition:String)
+getName():String
+getBirthday():Date
+getEvents():Event[]
+getGender():Gender
+getExtraCompensations():double
+getPhone():String
+getJobDescription():String
+getEmail():String
+getSalary():int
+getTasks():Task[]
+getTotalCompletedTasks():int
+getPass():String
+getID():String
+getUsername():String
+getPossition():String
+getTypeOfEmplloyment():Employment
+getAddress():String
+getPassword():String
+getVacations():VacationRequest[]
+getRank():int
+getRating():Float
+setName(name:String)
+setBirthday(birthday:Date)
+setEvents(events:Event[])
+setExtraCompensations(double)
+setGender(Gender)
+setPhone(String)
+setEmail(String)
+setSalary(int)
+setJobDescription(String)
+setTotalCompletedTasks(int)
+setPass(String)
+setID(String)
+setUsername(String)
+setPassword(String)
+setTypeOfEmplloyment(Employment)
+setAddress(String)
+setVacations(VacationRequest[])
+setRank(int)
+setPossition(String)
+setRating(Float)
+setTasks(Task[])
+getEmployees():ArrayList<Employee>
+getEmployee(Employee)
+deleteEmployee(Employee)
+addEmployee(Employee)
+selectEmployees(String, Object):ArrayList<Employee>
+updateAllOfEmployee(Employee)
+updateSpecificOfEmployee(Employee, Object[], Object[])
+login(username:String, password:String):String
eventScheme(Event{}):JSONArray
taskScheme(Task{}):JSONArray
vacationReqScheme(VacationRequest[]):JSONArray
raiseReqScheme(RaiseRequest[]):JSONArray
```

5.4.2 Relationships and multiplicity

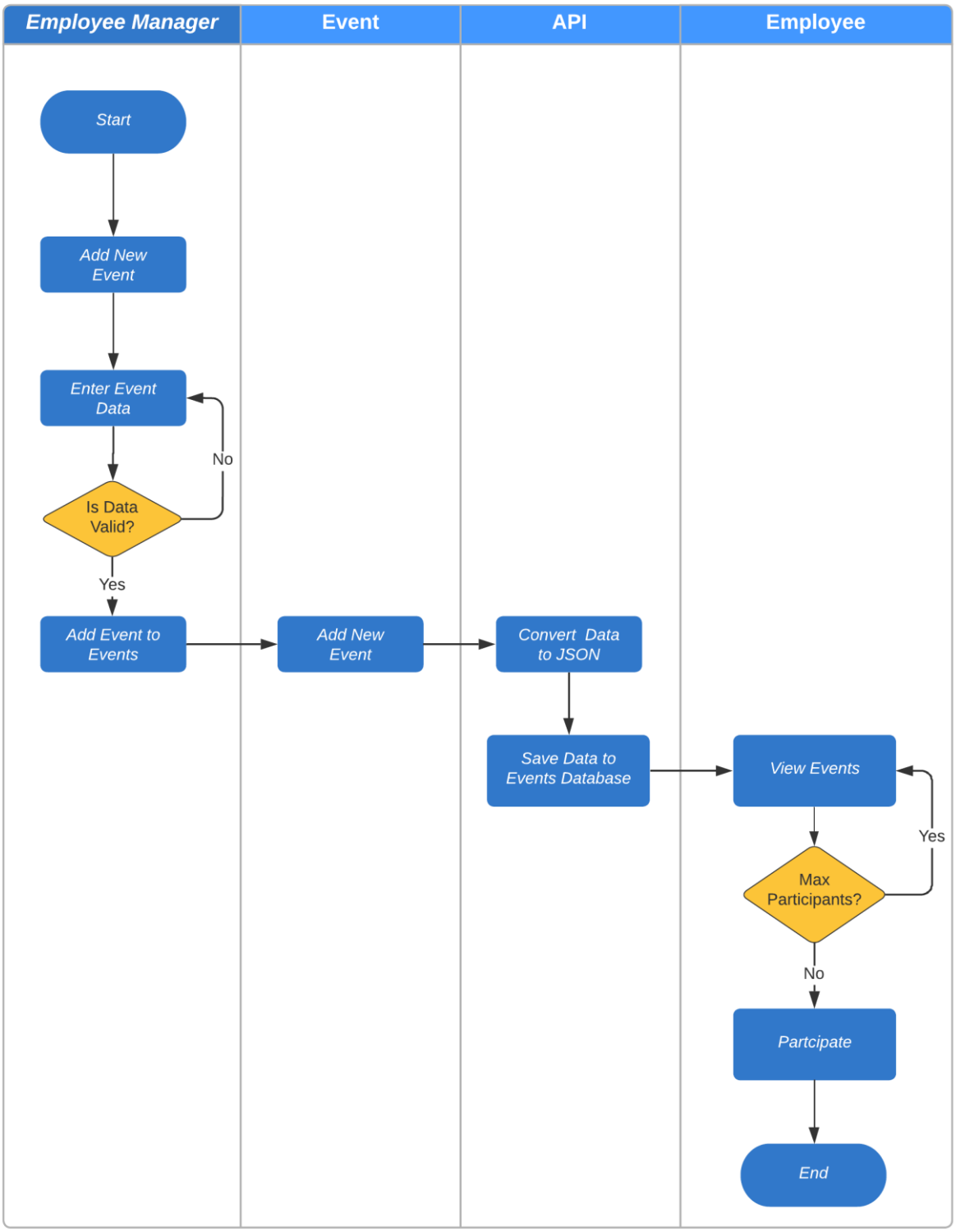


6.0 Swimlane Diagram

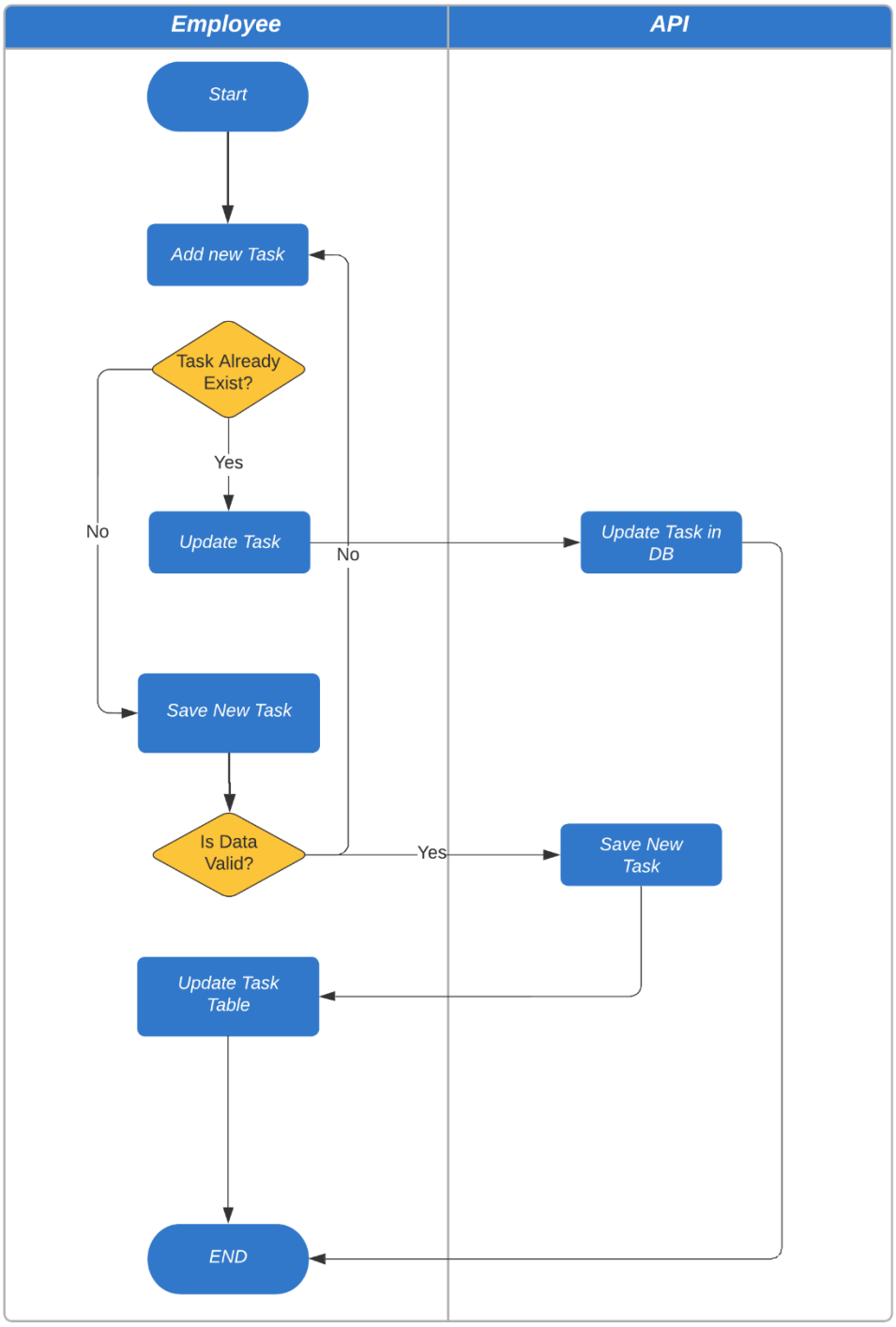
Add New Employee Swimlane:



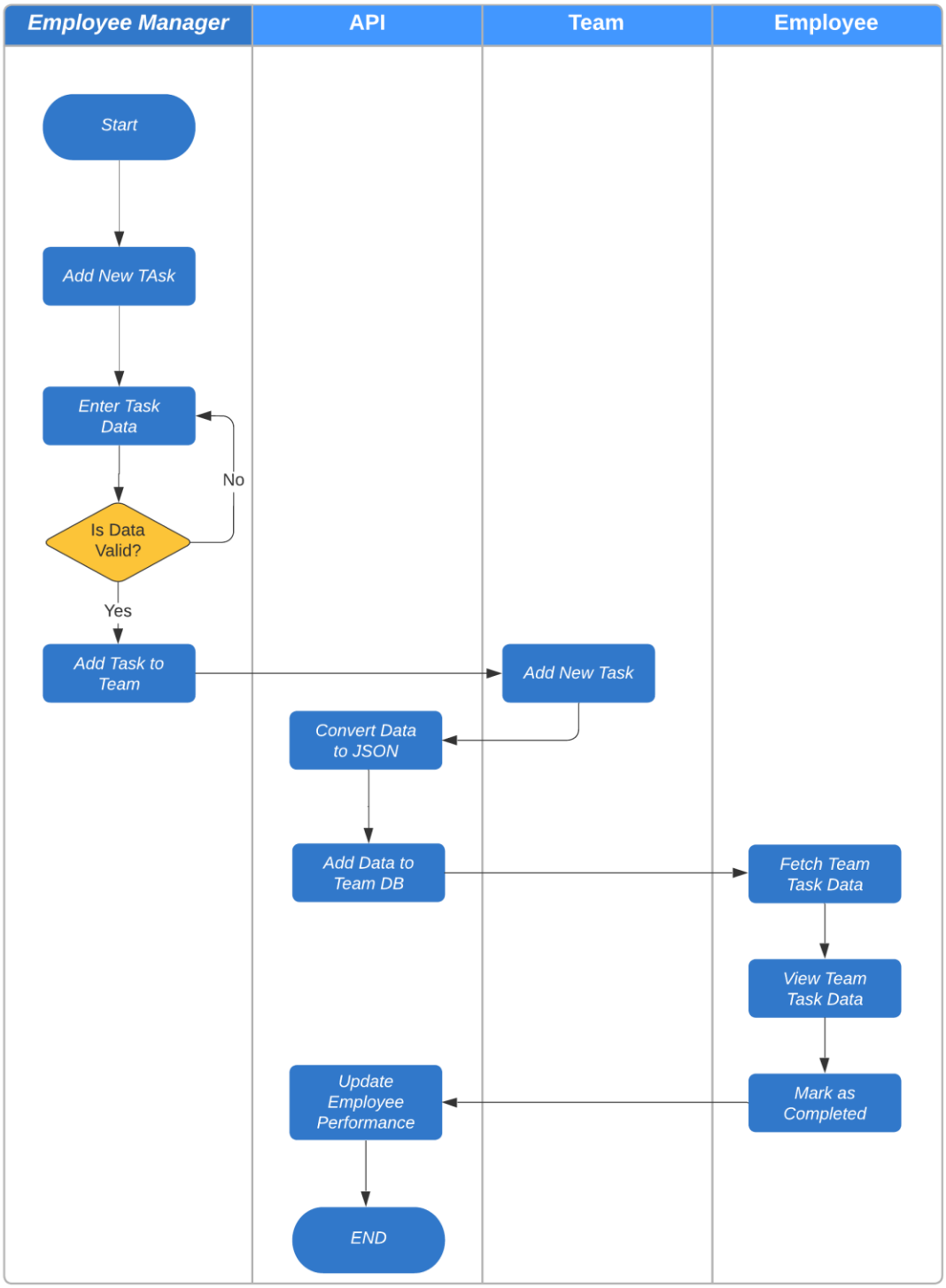
Event Handling Swimlane



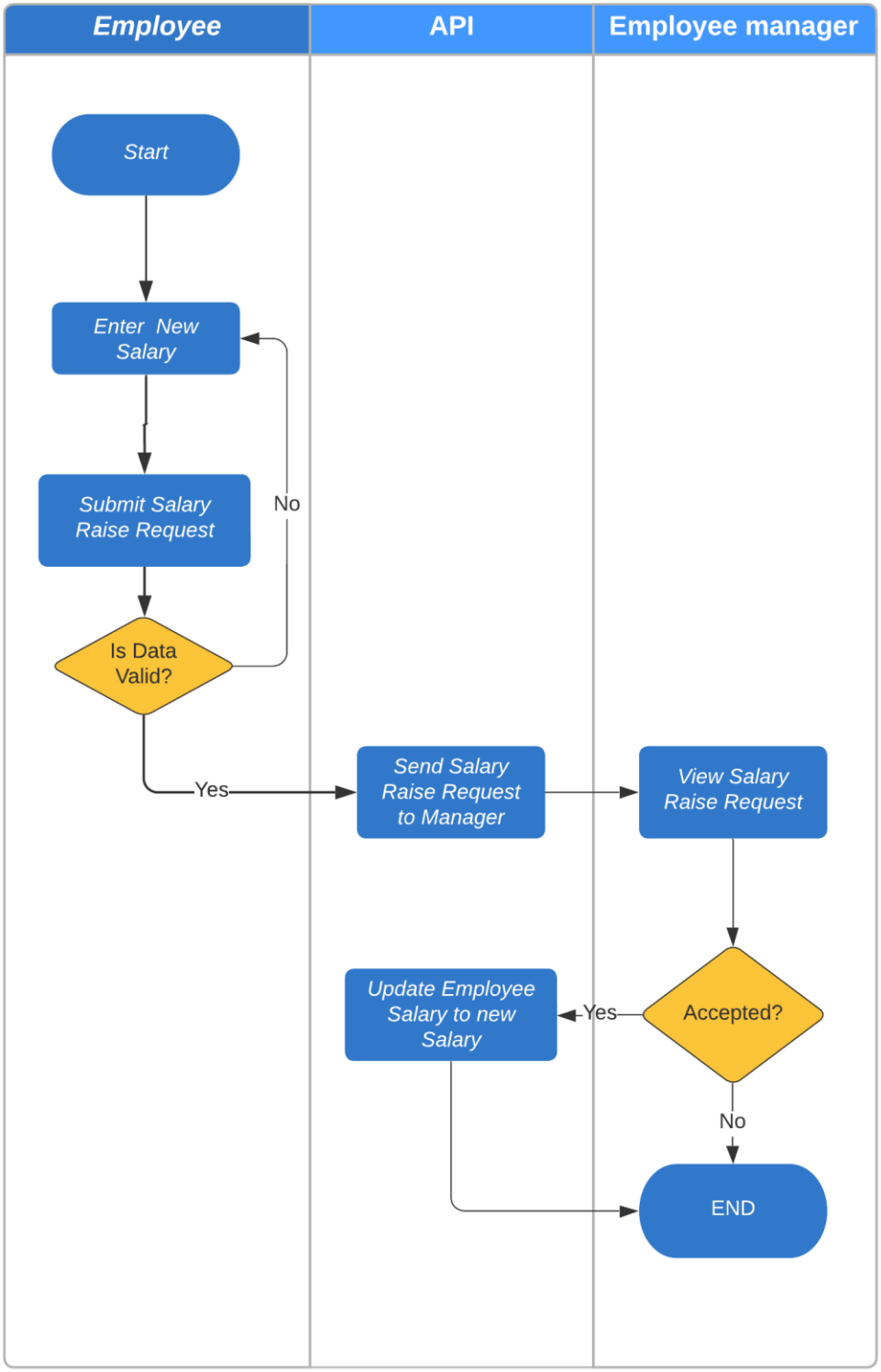
Personal Task Swimlane:



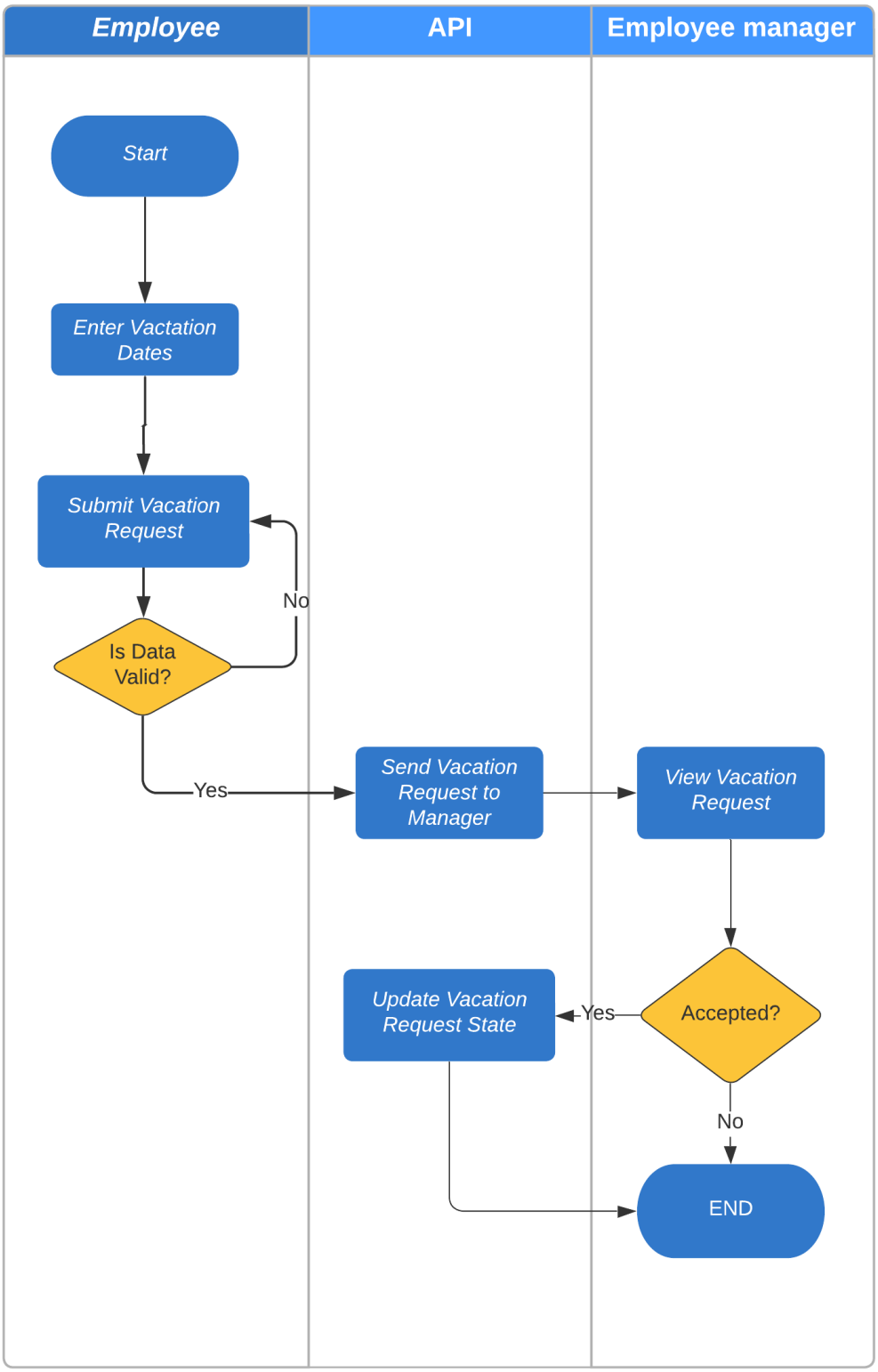
Team Task Swimlane:



Salary Raise Request Swimlane:



Vacation Request Swimlane:



Swimlane Descriptions:

Add New Employee Swimlane:

1. *Manager Select Add New Employee Option.*
2. *Manager Enter Employee Data.*
3. *Check If Data is Valid.*
4. *Add Employee Data to Manager's Team.*
5. *Save Employee Data Locally.*
6. *Convert New Employee Data to JSON.*
7. *Send JSON Data to Employees Database.*
8. *Add Employee ID to Team Database.*
9. *END.*

Event Handling Swimlane:

1. *Manager Select Add New Event Option.*
2. *Manager Enter Event Data.*
3. *Check If Data is Valid.*
4. *Add Event to Events Locally.*
5. *Convert Event Data to JSON.*
6. *Save Event Data to Events DB.*
7. *Employee View Available Events.*
8. *If Employee Wants to Register Check Max Participants*
9. *If Max Participants Not Reached, then Complete Registration.*
10. *END.*

Add Personal Task Swimlane:

1. *Employee Selects Add New Task Option.*
2. *Employee Enter New Task Data.*
3. *Check If Task Already Exist, If Not Jump to Step 6.*
4. *If It Is Update Already Existing Task Locally.*

5. *Save It Into DB, Jump to Step 10.*
6. *Save New Task Locally.*
7. *Check If Data is Valid.*
8. *Save New Task into DB.*
9. *Update Task Table.*
10. *END.*

Add Team Task:

1. *Employee Manager Selects Add New Task Option.*
2. *Employee Manager Enter New Task Data.*
3. *Check If Data is Valid.*
4. *Add Task to Team Locally.*
5. *Convert Updated Team Data to JSON.*
6. *Update Team DB with The New Data.*
7. *Employee Fetch New Task Data.*
8. *View New Task Data.*
9. *Work on New Task and Mark it as Completed.*
10. *Update Employee Performance.*
11. *END.*

Salary Raise Request Swimlane:

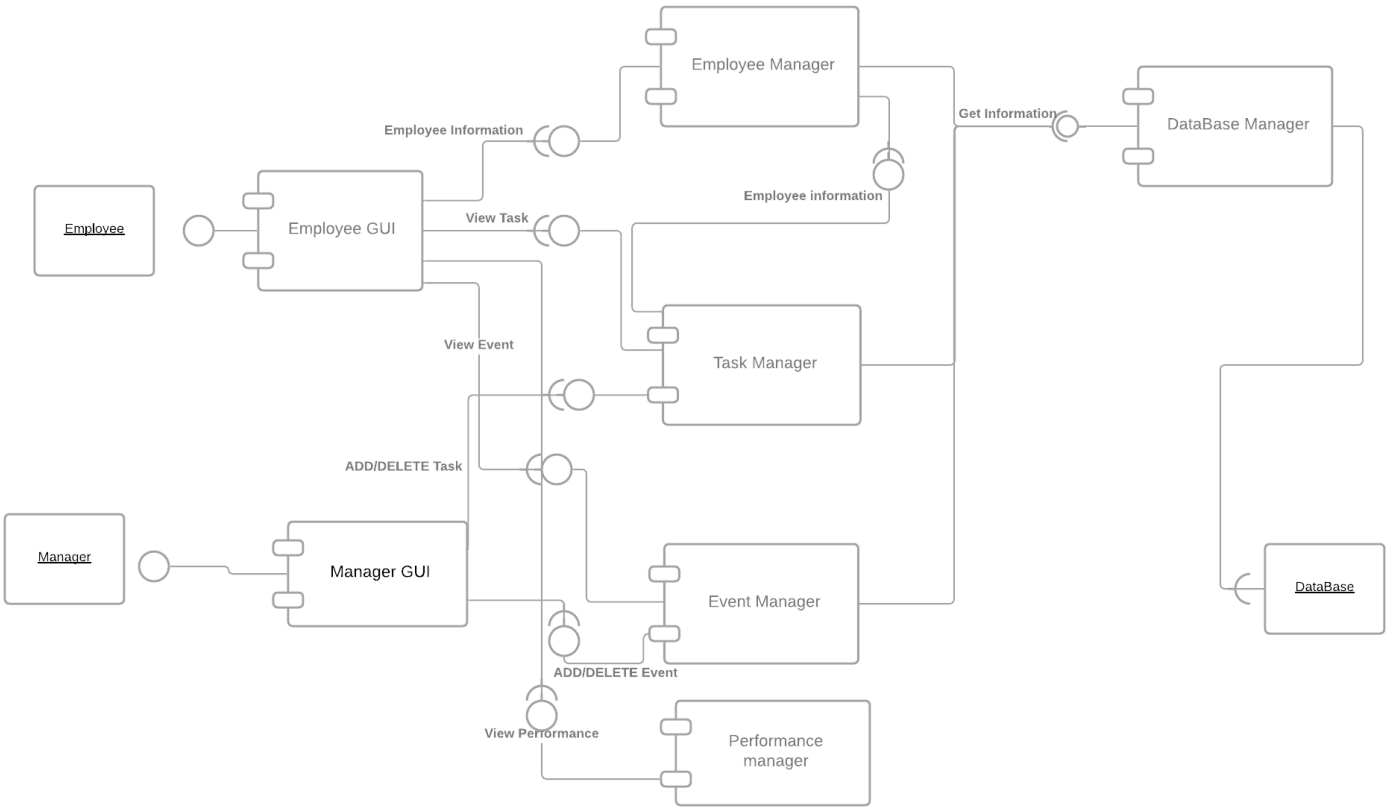
1. *Employee Enter New Salary.*
2. *Employee Submit Salary Raise Request.*
3. *Check If Data Is Valid.*
4. *Send Salary Request to Manager.*
5. *If Accepted, Update Employee Info with New Salary.*
6. *If Not, Request is Rejected, and Employee Data is Not Updated With The Wished Salary.*
7. *END.*

Vacation Request Swimlane:

1. *Employee Enter Vacation Dates.*

2. *Employee Submit Vacation Request.*
3. *Check If Data Is Valid.*
4. *Send Vacation Request to Manager.*
5. *If Accepted, Update Vacation Request State.*
6. *If Not, Request is Rejected.*
7. *END.*

7.0 Component Diagram



7.1 Components relation description:

Due to the nature of this software all the components are generalized for any type of company therefore, reusable and require minimal changes depending on the environment to be deployed in and according to the clients' requests. The users are abstract (employee, manager) with the tasks also being abstract.

The software mainly consists of seven major components.

1. Employee GUI: provides all the information and capabilities which are given to the employee. Requires tasks from the task manager, events from the event manager, performance from the performance manager and information from the employee manager.

2. Manager GUI: provides all the information and capabilities which are given to the manager. Requires tasks from the Task Manager, events from the Event Manager, performance from the Performance Manager and information from the Employee Manager.

3. Employee Manager: provides employee's personal information to the Employee GUI and the Manager GUI. Requires information from the database manager.

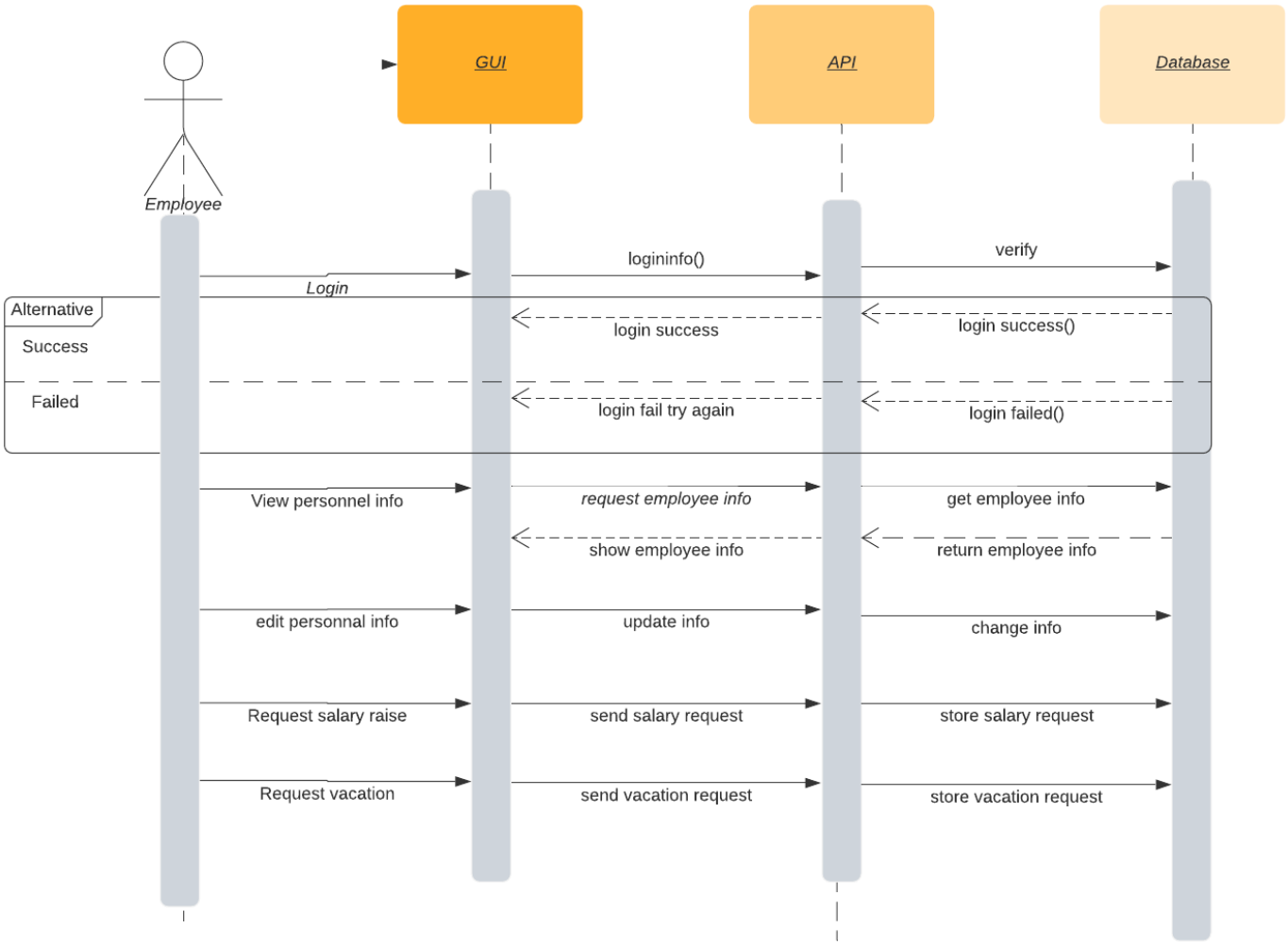
4. Task Manager: provides employee's tasks history to the employee GUI and the manager GUI. Requires tasks information from the Database Manager.

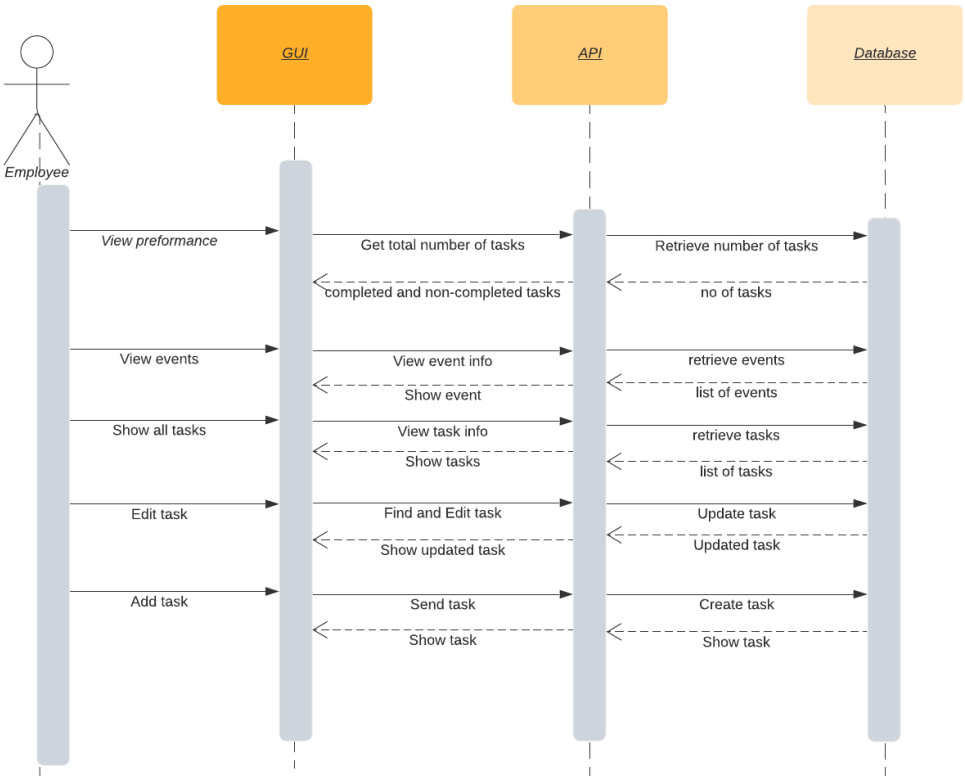
5. Event Manager: provides company's and employee's events to the employee GUI and the manager GUI. Requires events information from the Database Manager.

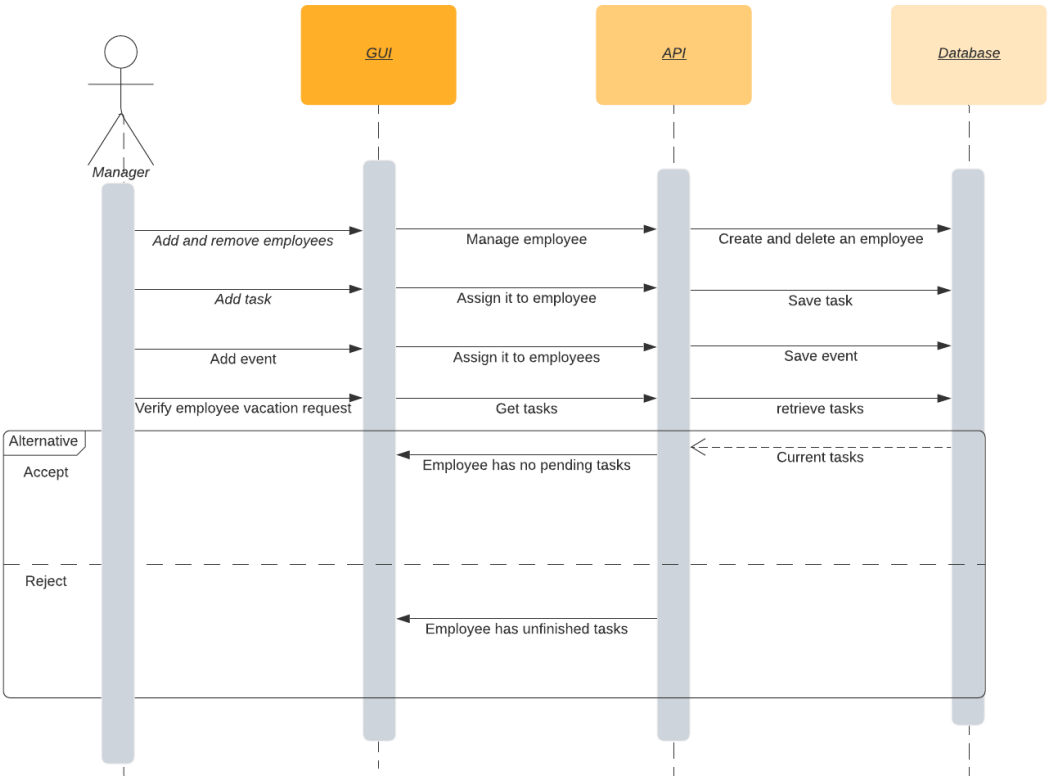
6. Performance Manager: provides employee's performance score to the employee GUI and the manager GUI.

7. Database Manager: provides all applicable information to the different components in the system (Employee Manager, Task Manager, and Event Manager) which is retrieved from the database of the company.

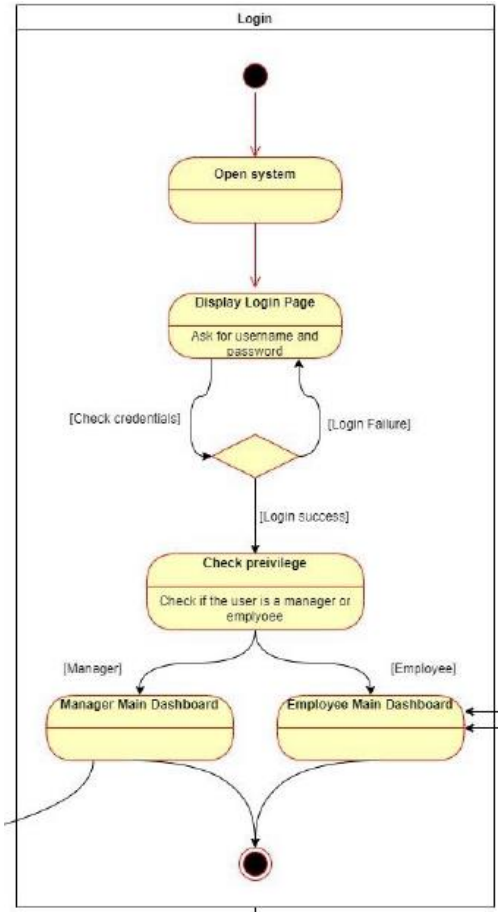
8.0 Sequence Diagram




GUI
API
Database
Employee
View preformance
Get total number of tasks
Retrieve number of tasks
completed and non-completed tasks
no of tasks
View events
View event info
retrieve events
Show event
list of events
Show all tasks
View task info
retrieve tasks
Show tasks
list of tasks
Edit task
Find and Edit task
Update task
Show updated task
Updated task
Add task
Send task
Create task
Show task
Show task


GUI
API
Database
Manager
Add and remove employees
Manage employee
Create and delete an employee
Add task
Assign it to employee
Save task
Add event
Assign it to employees
Save event
Verify employee vacation request
Get tasks
retrieve tasks
Alternative
Accept
Employee has no pending tasks
Current tasks
Reject
Employee has unfinished tasks

9.0 State Diagram

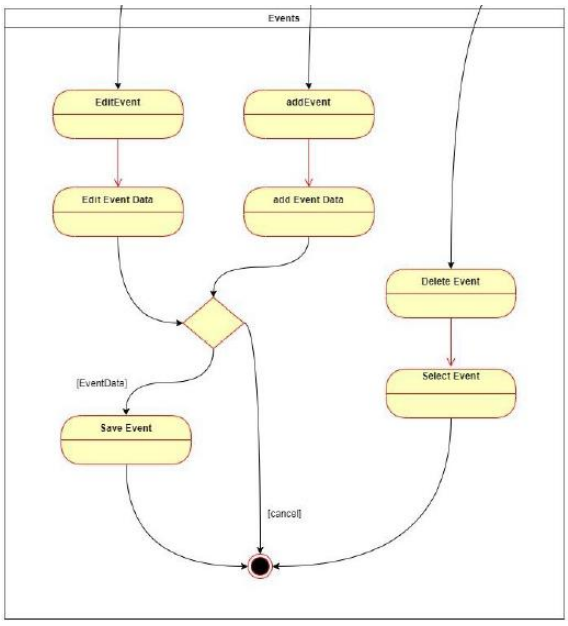


Login Frame:

By running the application the system opens {Open System} and displays login page

{Display Login Page}.

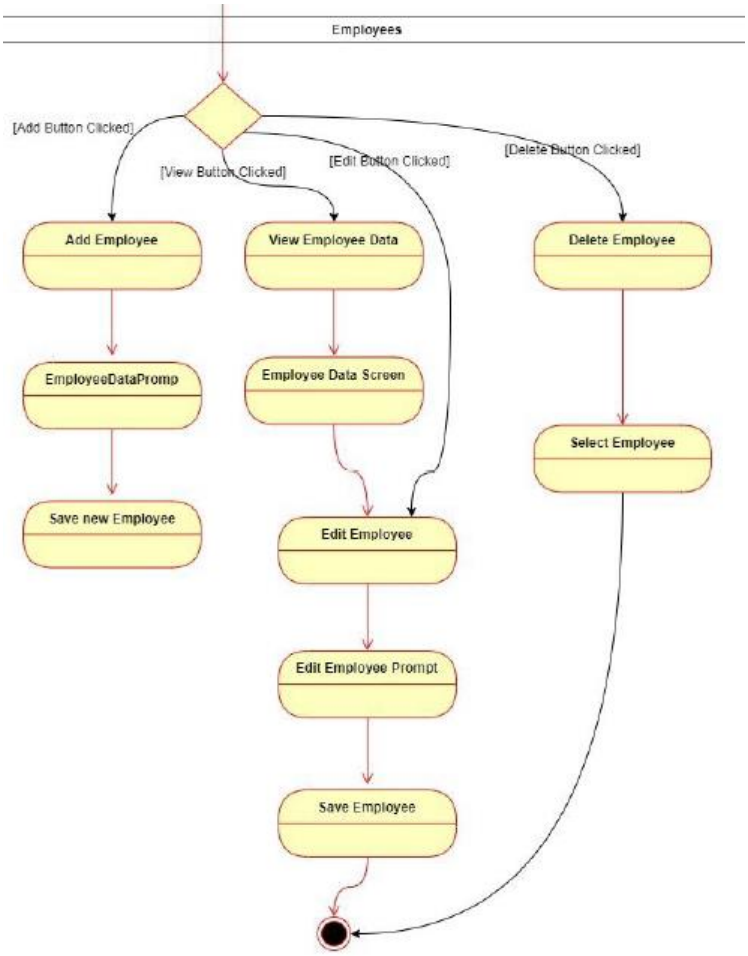
{Display Login Page} then has [login failure] which goes to the login page by {Display Login Page} or in case of success, the process of {Check Privilege}which is done to check if this was an employee or a manager; therefore, it is called to switch either to {Manager Main Dashboard} or {Employee Main Dashboard}.



Events Frame:

From the Main Screen: In case the Manager clicks on {Edit Event} or {add Event} both process execute {Edit Event Data} and {add Event Data} respectively; which pops a screen prompting for new data, however, a choice is then given to either cancel or add/edit data.

In case the Manager clicks on {delete Event} after selecting {Select Event}, the process is executed deleting this selected event.



Employees Frame:

From the Main Screen: In case of Manager clicking on {View All Employee} a screen is shown with:

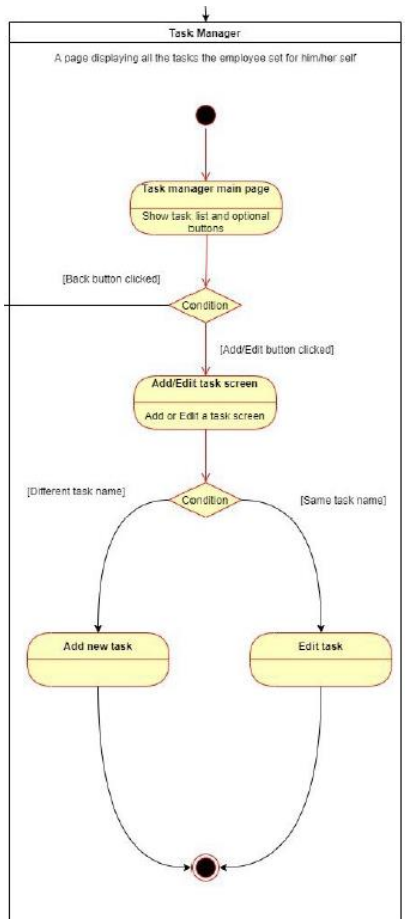
{Add Employee}: Upon clicking shows input screen for user by process {Employee Data prompt} and saves new employee to database by {Save New Employee}.

{View Employee Data}: Upon clicking navigates to Employee data screen by {Employee Data Screen} which can be used to edit employee to prompt user for input {Edit Employee Prompt} and then saves employee by {Save Employee}.

{Delete Event}: Upon clicking Delete event {Delete Event} process is executed by selecting the employee wanted for deletion {Select Employee}

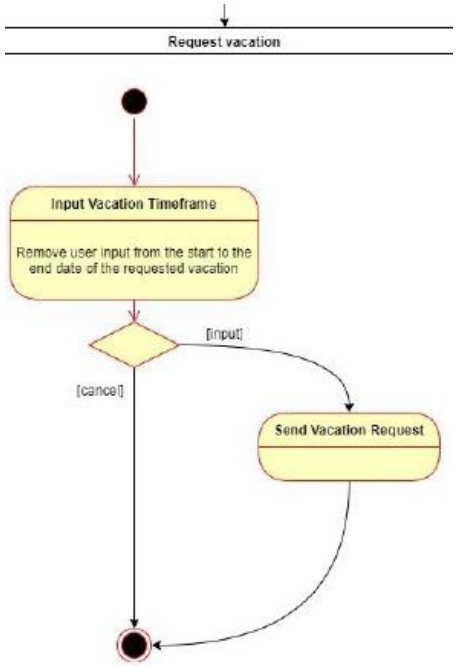
Task Manager Frame:

From the Main Screen: Navigate to Task Manager by clicking go to Task Manager; this show the tasks and functions by the process {Task Manager Main Page}. Add, edit or delete will be used by adding new in prompt screen using {Add New Task} and editing existing data by a prompt screen using {Edit Task} this then ends the current state. An option exists to return back to the main dashboard {Employee Main Dashboard}; upon clicking back to main.



Request Vacation Frame:

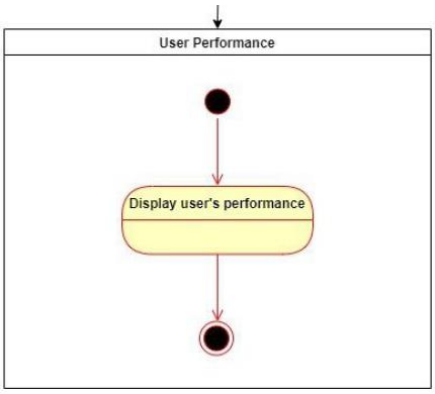
From the Main Screen: Navigate to request vacation screen by clicking on Request vacation button which will then prompt by {Input vacation Timeframe}, and has 2 choices either to cancel and return back to main or {Send Vacation Request} for state to end.



User Performance Frame:

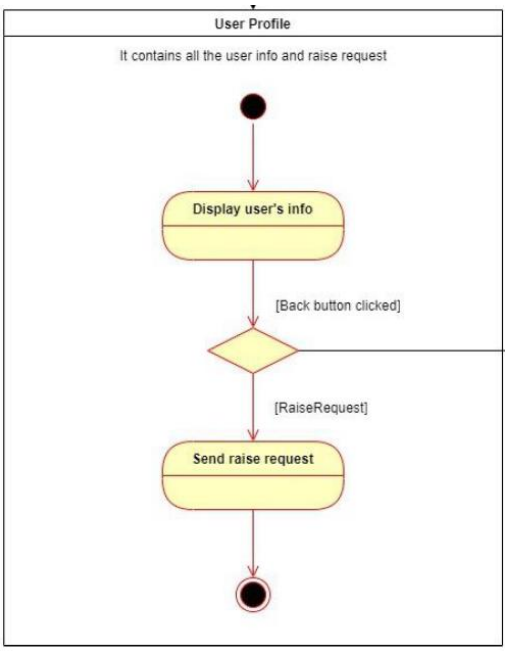
From the Main Screen: Navigate to performance of employee by clicking on the my performance button; which then displays user performance by {Display User Performance} and ends the state.

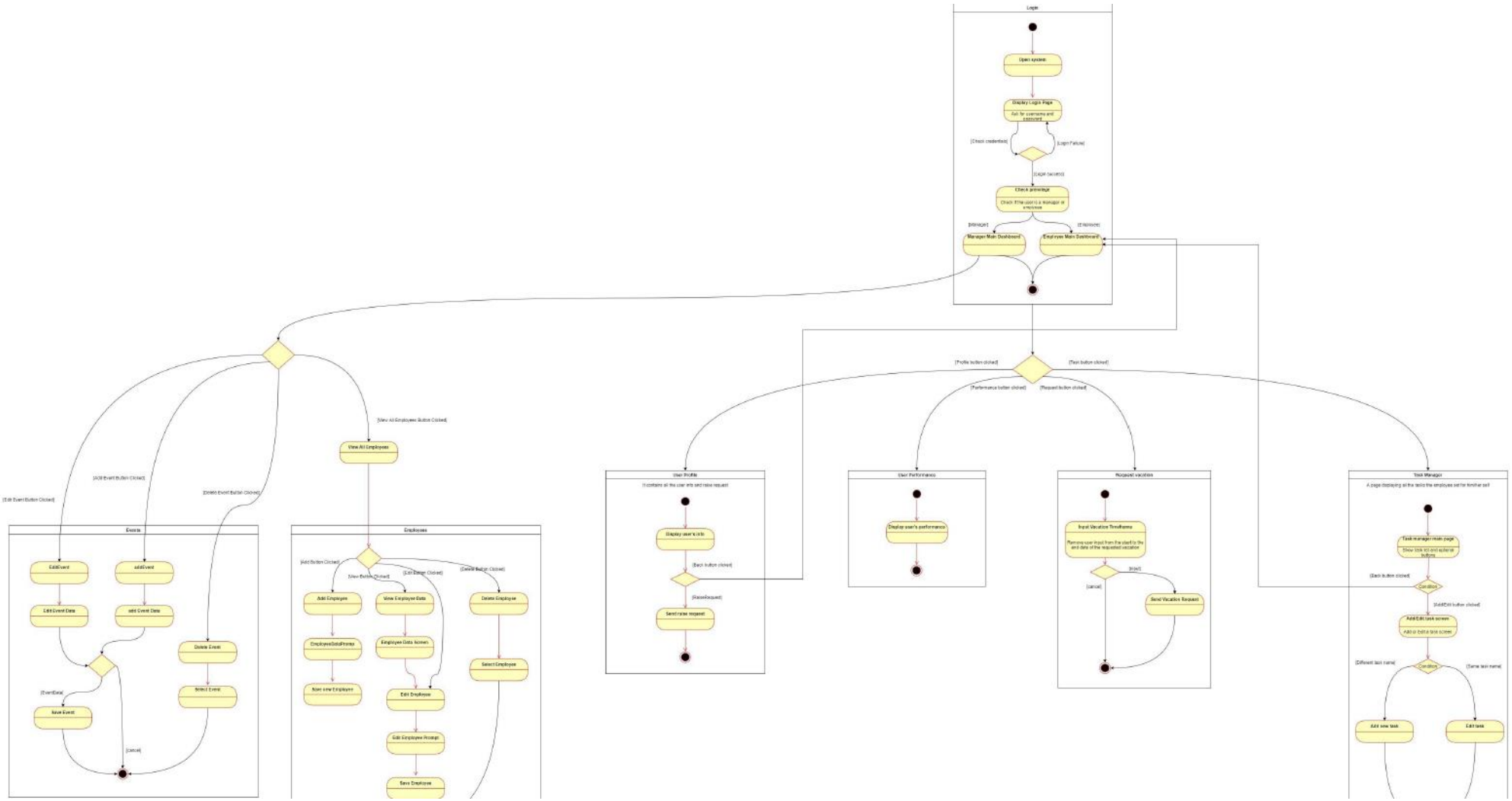
It can also navigate back to main screen by button back to main.



User Profile Frame:

From the Main Screen: Navigate to User Profile of the employee by clicking on my profile which displays information using {Display Info}, then gives you 2 choices either to go back to main by clicking on it or send salary raise request by {Send Raise Request} and ends the state.




Login
Open system
Display Login Page
Ask for username and password
[Check credentials]
[Login Failure]
[Login success]
Check privilege
Check if the user is a manager or employee
[Manager]
[Employee]
Manager Main Dashboard
Employee Main Dashboard
[Profile button clicked]
[Task button clicked]
[Performance button clicked]
[Request button clicked]
[View All Employees Button Clicked]
View All Employees
[Edit Event Button Clicked]
[Add Event Button Clicked]
[Delete Event Button Clicked]
Events
EditEvent
addEvent
Edit Event Data
add Event Data
Delete Event
Select Event
[EventData]
Save Event
[cancel]
Employees
[Add Button Clicked]
[View Button Clicked]
[Edit Button Clicked]
[Delete Button Clicked]
Add Employee
View Employee Data
Delete Employee
EmployeeDataPrompt
Employee Data Screen
Select Employee
Save new Employee
Edit Employee
Edit Employee Prompt
Save Employee
User Profile
It contains all the user info and raise request
Display user's info
[Back button clicked]
[RaiseRequest]
Send raise request
User Performance
Display user's performance
Requested vacation
Input Vacation Timeframe
Remove user input from the start to the end date of the requested vacation
[save]
[cancel]
Send Vacation Request
Task Manager
A page displaying all the tasks the employee set for him/her self
Task manager main page
Show task list and editor buttons
[Back button clicked]
Condition
[Add/Edit button clicked]
Add/Edit task screen
Add or Edit a task screen
[Different task name]
[Same task name]
Add new task
Edit task

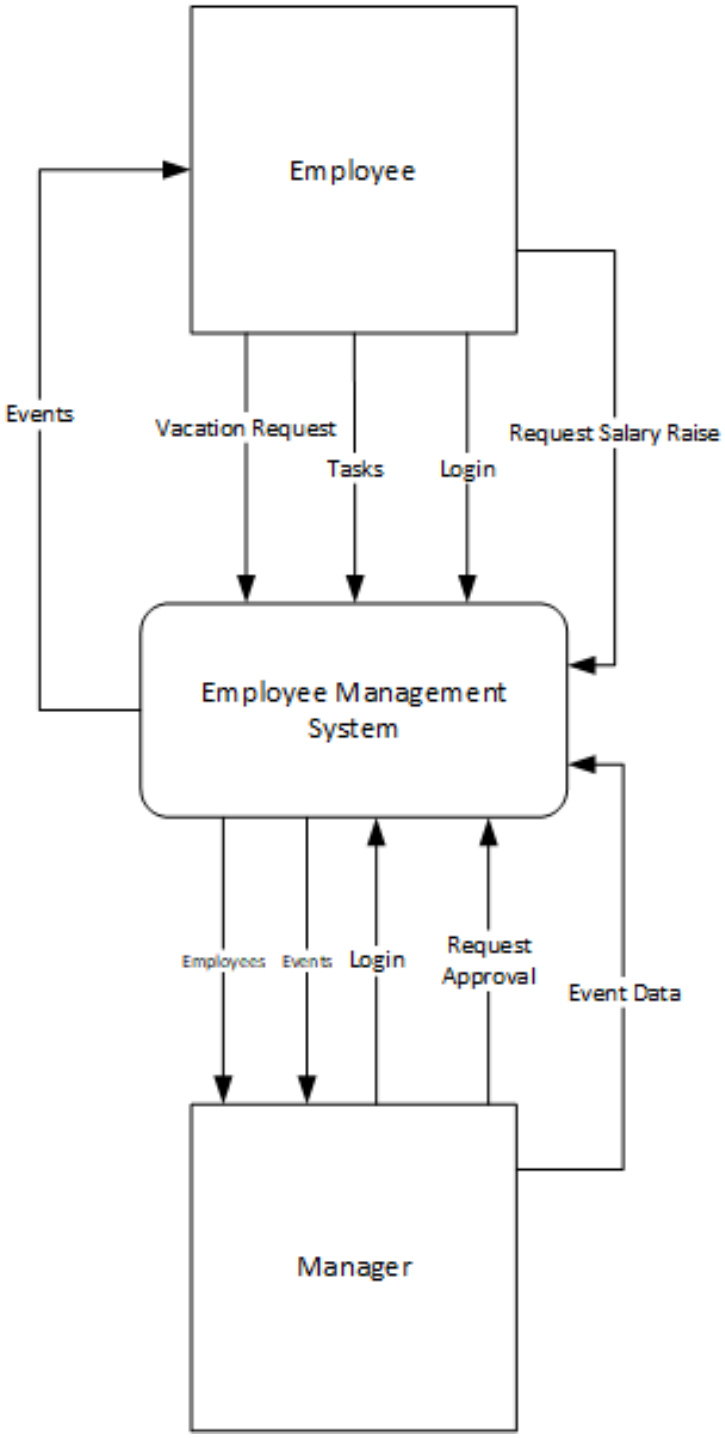
10.0 Context and DFD diagrams

10.1 Context Diagram

Employee Management System is communicating between 2 external entities Employee and Manager.

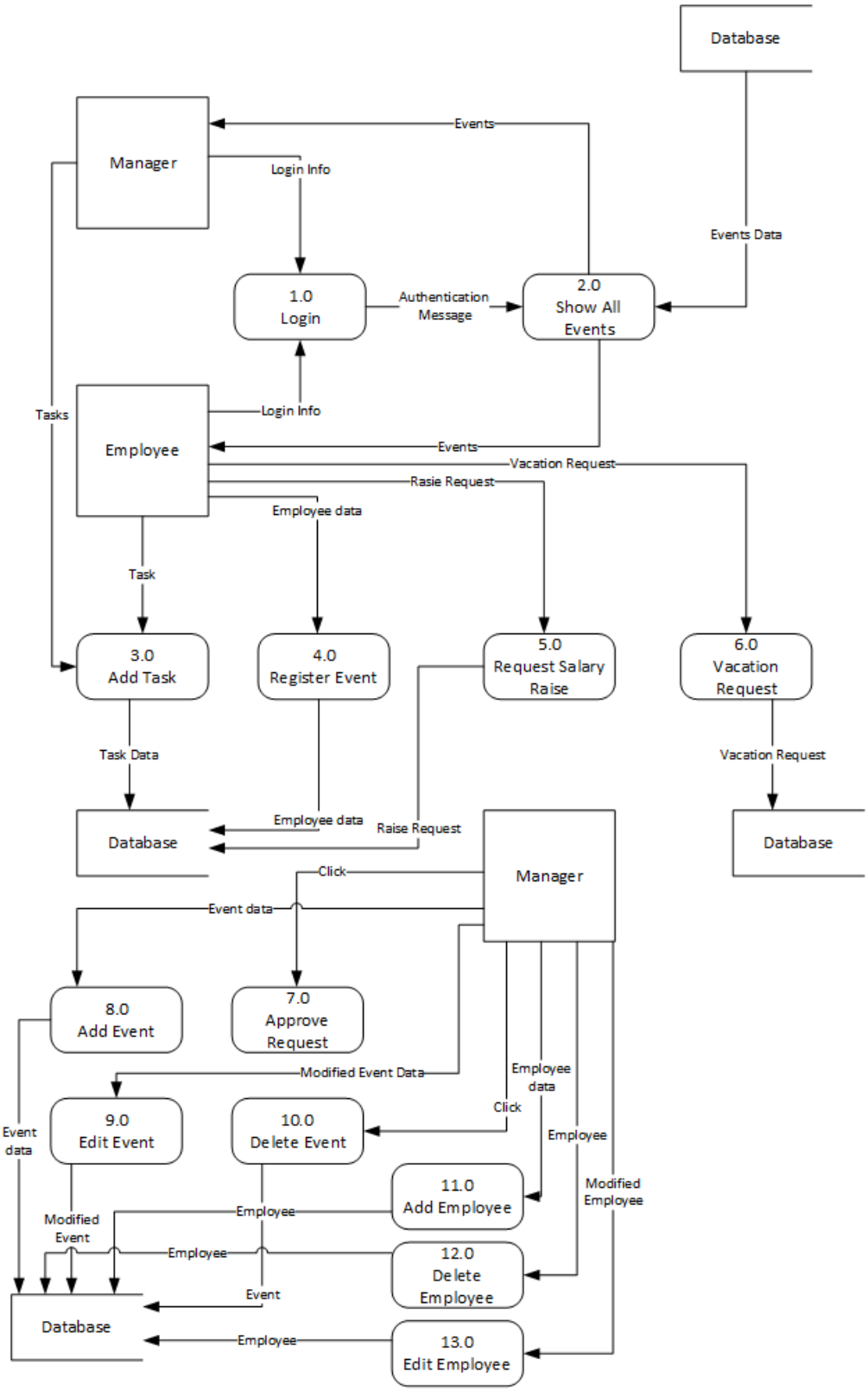
Employee: sends login information to enter the main screen, also he has the ability to send tasks information, and vacation /salary raise requests and view events details.

Manager: sends login information to enter the main screen, also he has the ability to send and view tasks /events information, vacation requests, and approve employee's requests.



10.2 DFD

10.2.1 Level-0 DFD



Contains an additional internal database communicating with the other external entities and the EMS.

1.0 **Login:** Both Employee and Manager send login info which will be authenticated and sent as a message as an output

2.0 **Show All Events**: This is the Main screen for both the employee and Manager, in which all events will be shown by receiving the authenticated message from 1.0, then getting events data from the database to be sent/shown to the Employee and the Manager

3.0 **Add Task:** This process takes tasks info from the Employee or the Manager to be added by sending information to the database

4.0 **Register Event:** Employee will register the event shown on screen to be saved and sent to the database

5.0 **Request Salary Raise:** Takes salary request information from the Employee and sends it to the database to be saved

6.0 **Vacation Request**: Takes vacation request information from the Employee and sends it to the database to be saved

7.0 **Approve Request:** Upon clicking the Employee's request is approved and modified in the database

8.0 **Add Event**: The Manager sends the event information to be added, then this data is sent to the database to be saved

9.0 **Edit Event:** The Manager sends the modified event information, then this data is sent to the database to be saved

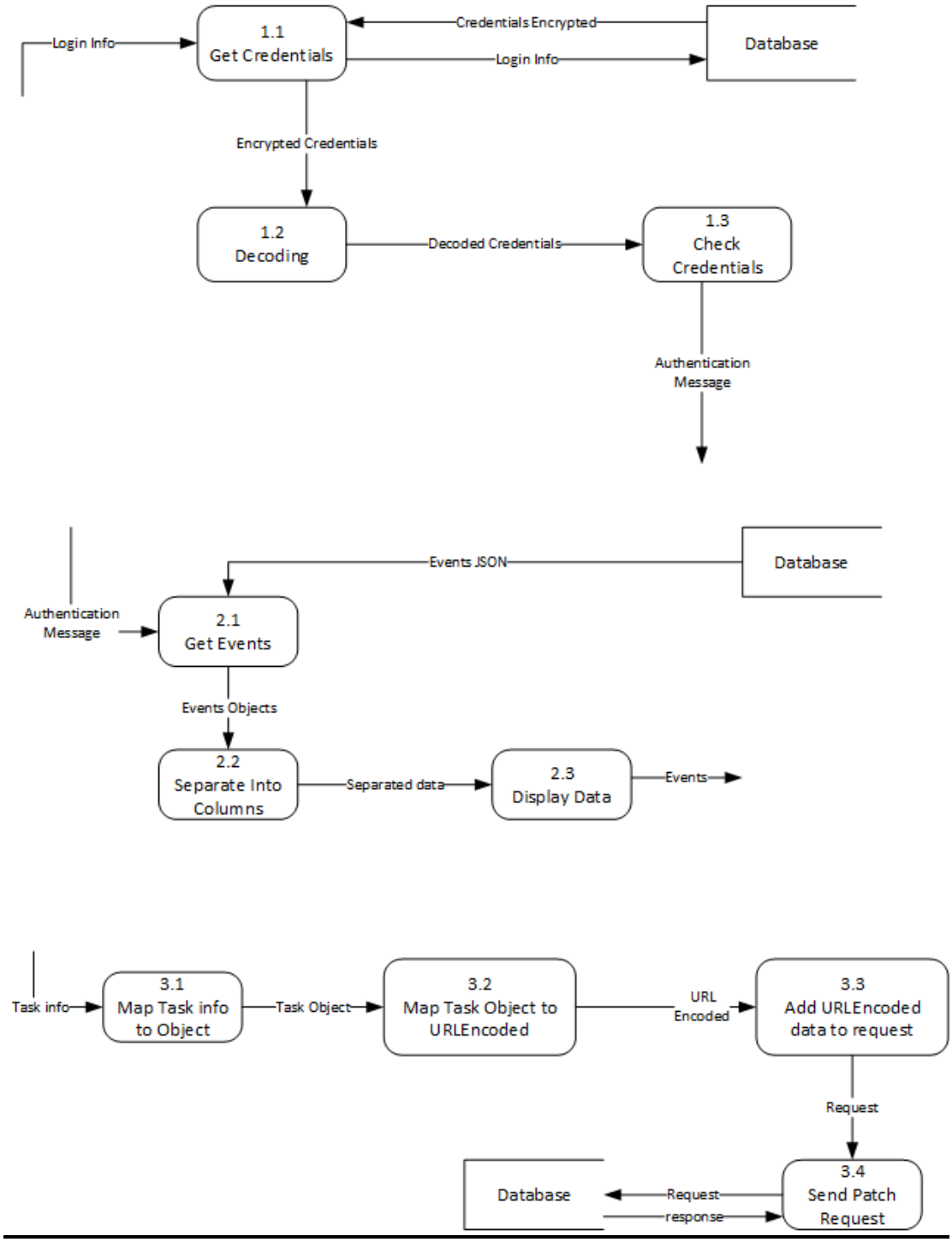
10.0 **Delete Event**: Upon clicking on an event this event object is deleted from the database

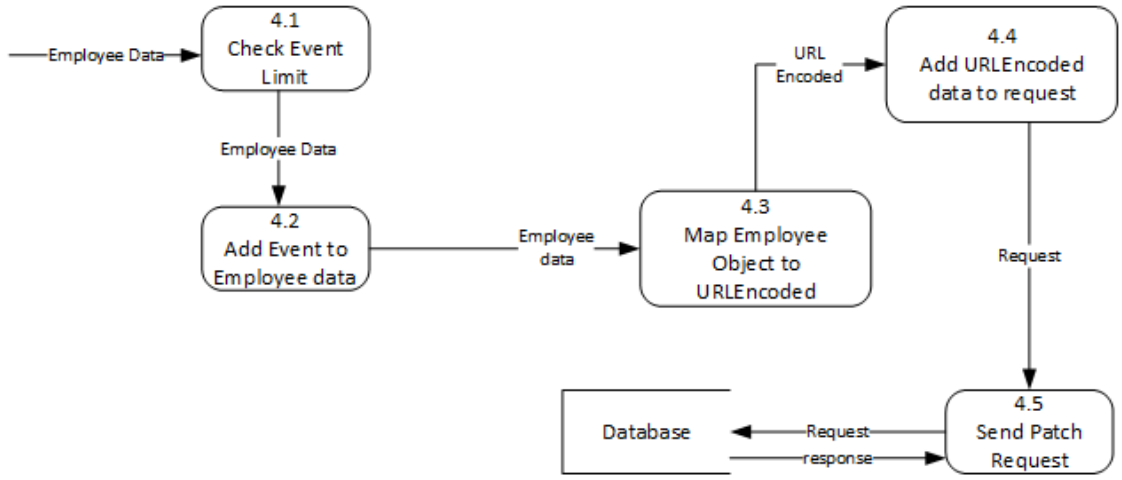
11.0 **Add Employee:** Takes new Employee Data and sends it as an object to the database to saved

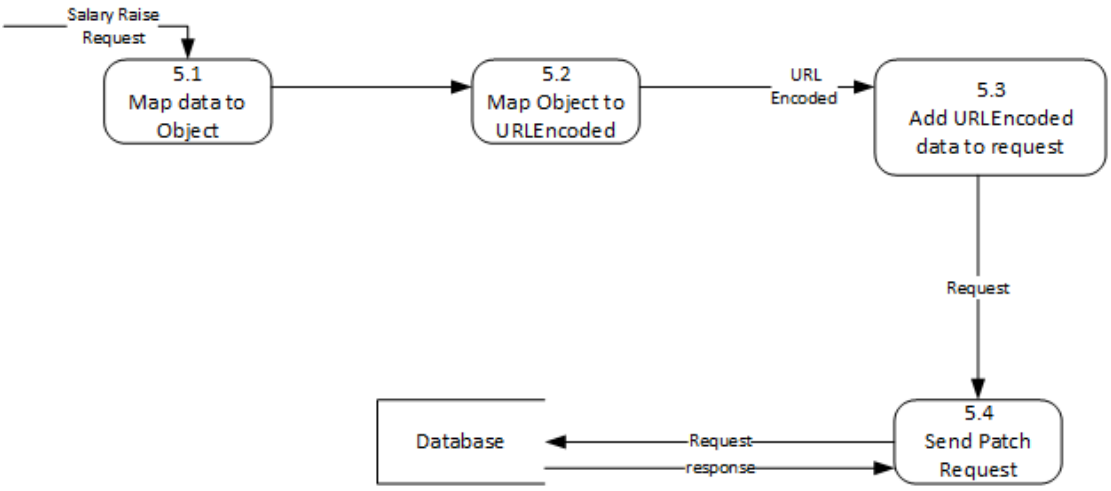
12.0 **Delete Employee:** Upon clicking this Employee will be deleted from the database

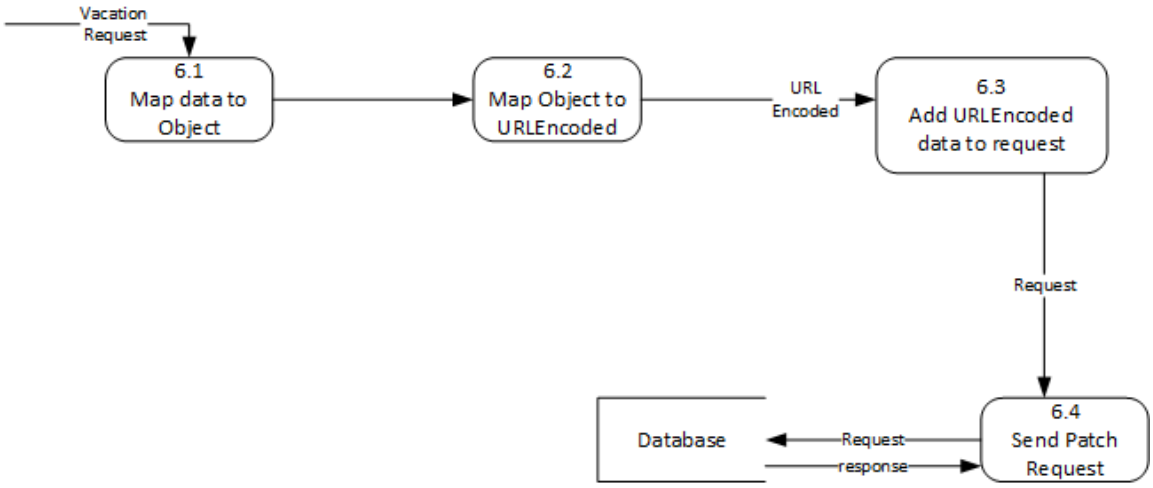
13.0 **Edit Employee:** The Manager sends the modified employee information, then this data is sent to the database to be saved

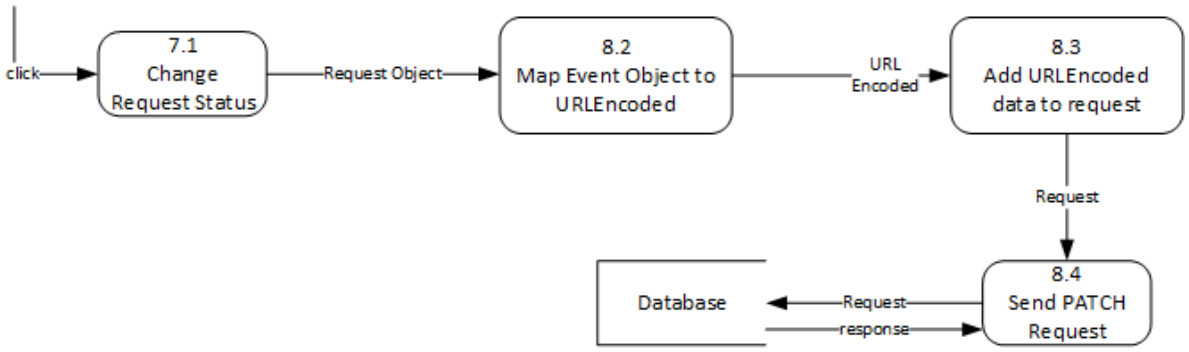
10.2.2 Level-1 DFD

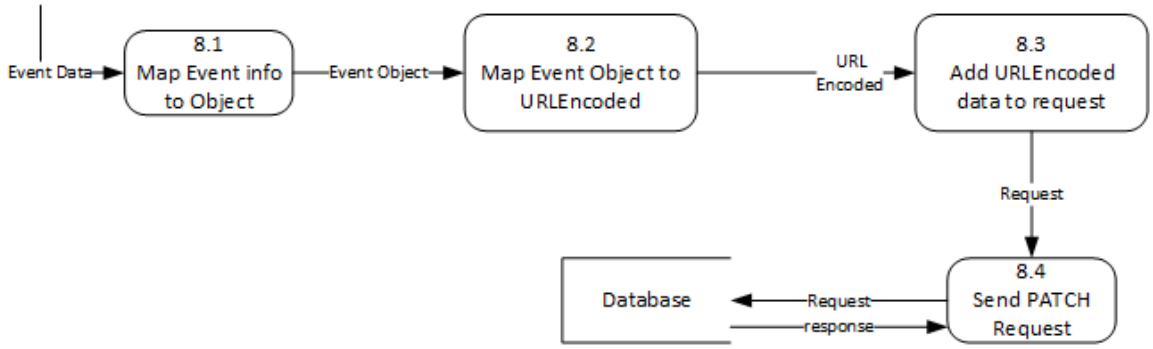


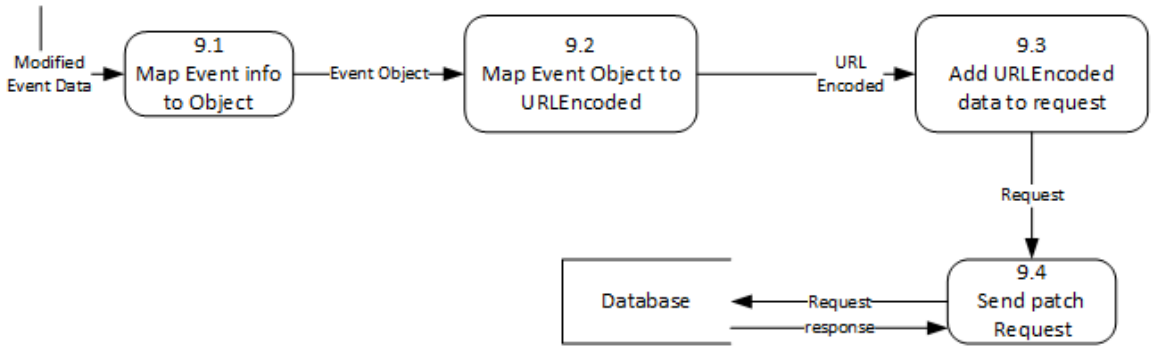

Employee Data
4.1
Check Event
Limit
Employee Data
4.2
Add Event to
Employee data
Employee
data
4.3
Map Employee
Object to
URLEncoded
URL
Encoded
4.4
Add URLEncoded
data to request
Request
Database
Request
response
4.5
Send Patch
Request

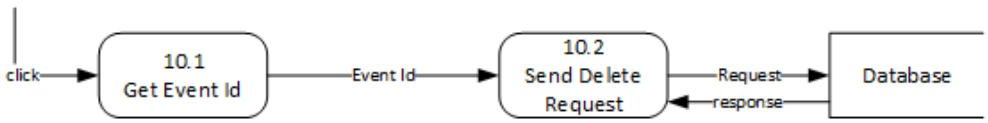

Salary Raise
Request
5.1
Map data to
Object
5.2
Map Object to
URLEncoded
URL
Encoded
5.3
Add URLEncoded
data to request
Request
Database
Request
response
5.4
Send Patch
Request

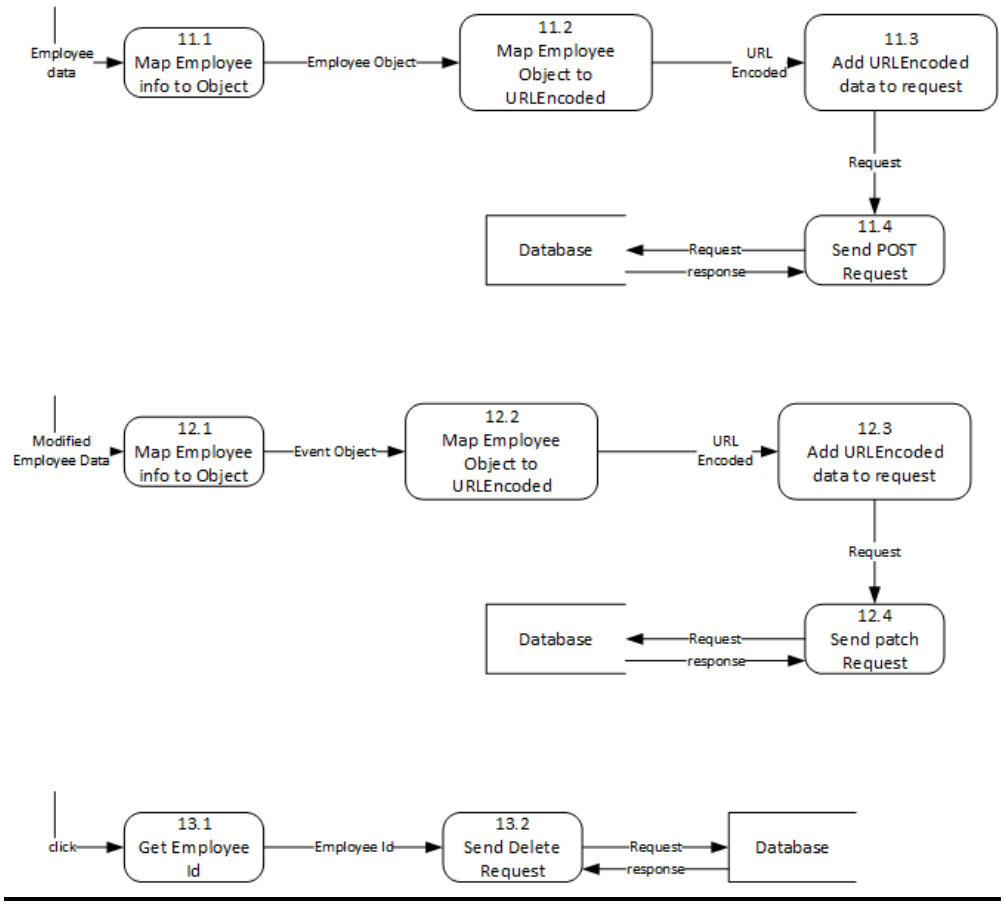

Vacation
Request
6.1
Map data to
Object
6.2
Map Object to
URLEncoded
URL
Encoded
6.3
Add URLEncoded
data to request
Request
Database
Request
response
6.4
Send Patch
Request


click
7.1
Change
Request Status
Request Object
8.2
Map Event Object to
URLEncoded
URL
Encoded
8.3
Add URLEncoded
data to request
Request
Database
Request
response
8.4
Send PATCH
Request


Event Data
8.1
Map Event info
to Object
Event Object
8.2
Map Event Object to
URLEncoded
URL
Encoded
8.3
Add URLEncoded
data to request
Request
Database
Request
response
8.4
Send PATCH
Request


Modified
Event Data
9.1
Map Event info
to Object
Event Object
9.2
Map Event Object to
URLEncoded
URL
Encoded
9.3
Add URLEncoded
data to request
Request
Database
Request
response
9.4
Send patch
Request


click
10.1
Get Event Id
Event Id
10.2
Send Delete
Request
Request
response
Database



1.0 **Login:**

1.1 **Get Credentials:** Takes login information from the Manager or the Employee then sends it to the database to be encrypted and returned; encrypted data is then sent to as an output

1.2 **Decoding:** Takes encrypted data and decodes it to be sent as an output

1.3 **Check Credentials:** This then takes the decoded credentials and checks them send an authentication message as output to be used to show the main screen

2.0 **Show All Events:**

2.1 **Get Events:** Extracts Events as a JSON object from the database, and takes the authentication message as a flag to show the main screen and its events

2.2 **Separate into columns:** This takes the Events object and separates it into logical columns to be sent as an output for display

2.3 **Display Data:** takes separated data and shows them on screen as Events data

3.0 **Add Task:**

3.1 **Map Task info to Object:** Takes new Tasks info inputted from the Employee or the Manager and converts it to an object to be passed

3.2 **Map Task Object to URL-Encoded:** takes the Task object and converts it to URL-Encoded to be passed

3.3 **Add URL-Encoded Data to Reques**t: The URL-Encoded is then passed to be added and a request is generated to be sent as an output

3.4 **Send Patch Request:** Receives the request from the URL-Encoded and sends the request to the database to receive a response of confirmation

4.0 **Register Event:**

4.1 **Check Event Limit:** Takes Employee Data who took the action of clicking and sends it as an output after checking the limit of Employees for this event

4.2 **Add Event to Employee Data:** Takes this employee data and adds the event to its data to be sent forward

4.3 **Map Employee Object to URL-Encoded:** Takes the final employee data to convert it to URL-Encoded to be passed

4.4 **Add URL-Encoded Data to Request:** Takes the URL-Encoded and adds it to a request (generated) to be sent

4.5 **Send Patch Request:** This is responsible to deliver the request to the database and receives a response of confirmation

5.0 **Request Salary Raise:**

5.1 **Map Data to Object:** Takes the salary raise request inputted details and send them forward after converting them to an object

5.2 **Map Object to URL-Encoded:** Takes the object and converts it to URL-Encoded to be passed forward

5.3 **Add URL-Encoded Data to Request:** Takes the URL-Encoded and adds it to a request (generated) to be sent

5.4 **Send Patch Request:** This is responsible to deliver the request to the database and receives a response of confirmation

6.0 **Vacation Request:**

6.1 **Map Data to Object:** Takes the vacation request inputted details and send them forward after converting them to an object

6.2 **Map Object to URL-Encoded:** Takes the object and converts it to URL-Encoded to be passed forward

6.3 **Add URL-Encoded Data to Request:** Takes the URL-Encoded and adds it to a request (generated) to be sent

6.4 **Send Patch Request:** This is responsible to deliver the request to the database and receives a response of confirmation

7.0 **Approve Request:**

7.1 **Change Request Status:** takes the action of clicking and sends a request object to be passed

7.2 **Map Object to URL-Encoded:** Takes the object and converts it to URL-Encoded to be passed forward

7.3 **Add URL-Encoded Data to Request:** Takes the URL-Encoded and adds it to a request (generated) to be sent

7.4 **Send Patch Request:** This is responsible to deliver the request to the database and receives a response of confirmation

8.0 **Add Event:**

8.1 **Map Event Info to Object:** Takes the Event inputted details and send them forward after converting them to an object

8.2 **Map Object to URL-Encoded:** Takes the object and converts it to URL-Encoded to be passed forward

8.3 **Add URL-Encoded Data to Request:** Takes the URL-Encoded and adds it to a request (generated) to be sent

8.4 **Send Patch Request:** This is responsible to deliver the request to the database and receives a response of confirmation

9.0 **Edit Event:**

9.1 **Map Event Info to Object:** Takes the Modified Event inputted details and send them forward after converting them to an object

9.2 **Map Object to URL-Encoded:** Takes the object and converts it to URL-Encoded to be passed forward

9.3 **Add URL-Encoded Data to Request:** Takes the URL-Encoded and adds it to a request (generated) to be sent

9.4 **Send Patch Request:** This is responsible to deliver the request to the database and receives a response of confirmation

10.0 **Delete Event**

10.1 **Get Event ID:** Upon clicking get ID method is called to extract the events ID to be passed

10.2 **Send Delete Request:** Takes the ID of the Event and sends a request containing the ID to the database and receives a confirmation of deletion

11.0 **Add Employee:**

11.1 **Map Event Info to Object:** Takes the new Employee inputted details and send them forward after converting them to an object

11.2 **Map Object to URL-Encoded:** Takes the object and converts it to URL-Encoded to be passed forward

11.3 **Add URL-Encoded Data to Request:** Takes the URL-Encoded and adds it to a request (generated) to be sent

11.4 **Send Patch Request:** This is responsible to deliver the request to the database and receives a response of confirmation

12.0 **Edit Employee:**

12.1 **Map Event Info to Object:** Takes the modified Employee inputted details and send them forward after converting them to an object

12.2 **Map Object to URL-Encoded:** Takes the object and converts it to URL-Encoded to be passed forward

12.3 **Add URL-Encoded Data to Request:** Takes the URL-Encoded and adds it to a request (generated) to be sent

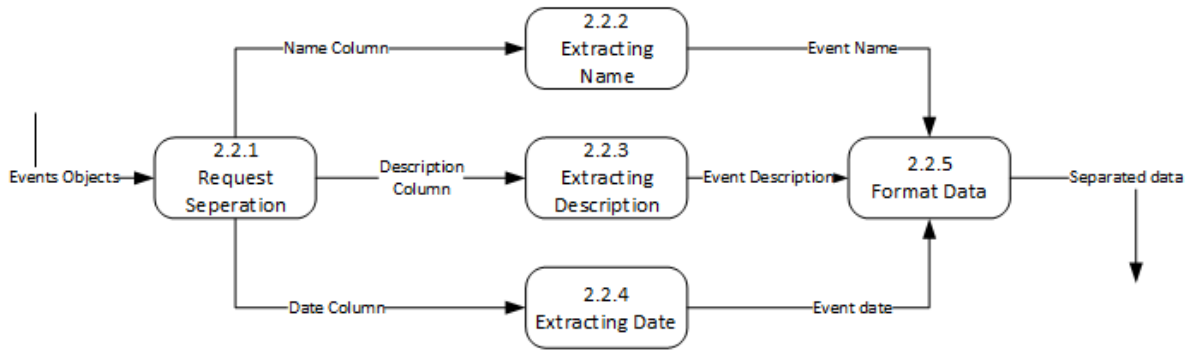
12.4 **Send Patch Request:** This is responsible to deliver the request to the database and receives a response of confirmation

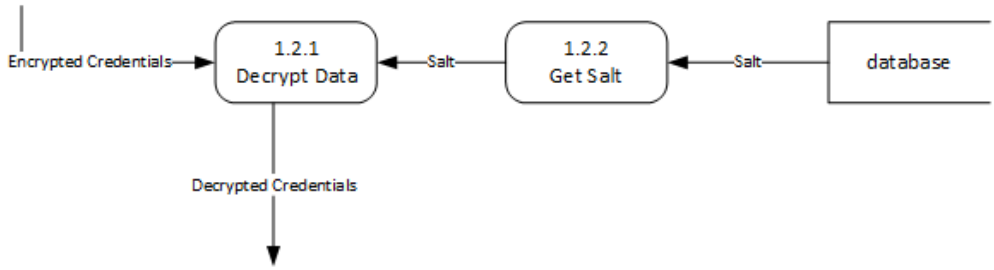
13.0 **Delete Employee**

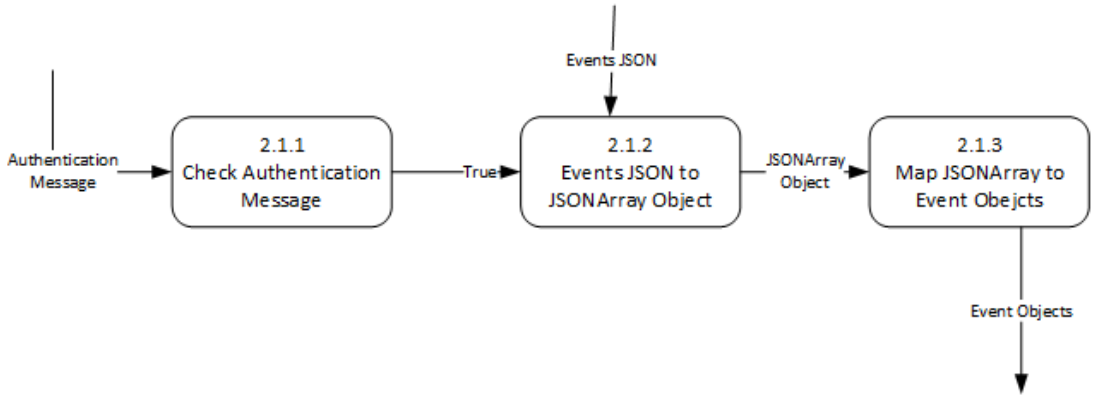
13.1 **Get Employee ID:** Upon clicking get ID method is called to extract the events ID to be passed

Send Delete Request: Takes the ID of the Event and sends a request containing the ID to the database and receives a confirmation of deletion.

9.2.3 Level-2 DFD







1.0 Login:

1.2.1 **Decrypt Data:** Takes the Encrypted Credentials and the Salt to decrypt the credentials received and output it as decrypted Data

1.2.2 **Get Salt:** Gets salt from the database to be passed to Decrypt Data (1.2.1)

2.0 **Show All Events:**

2.2.1 **Request Separation:** Takes the Events Object and extracts each column separately to be passed

2.2.2 **Extracting Name:** Gets the required Name of the Event

2.2.3 **Extracting Description:** Gets the required Description of the Event

2.2.4 **Extracting Date:** Gets the required Name of the Event

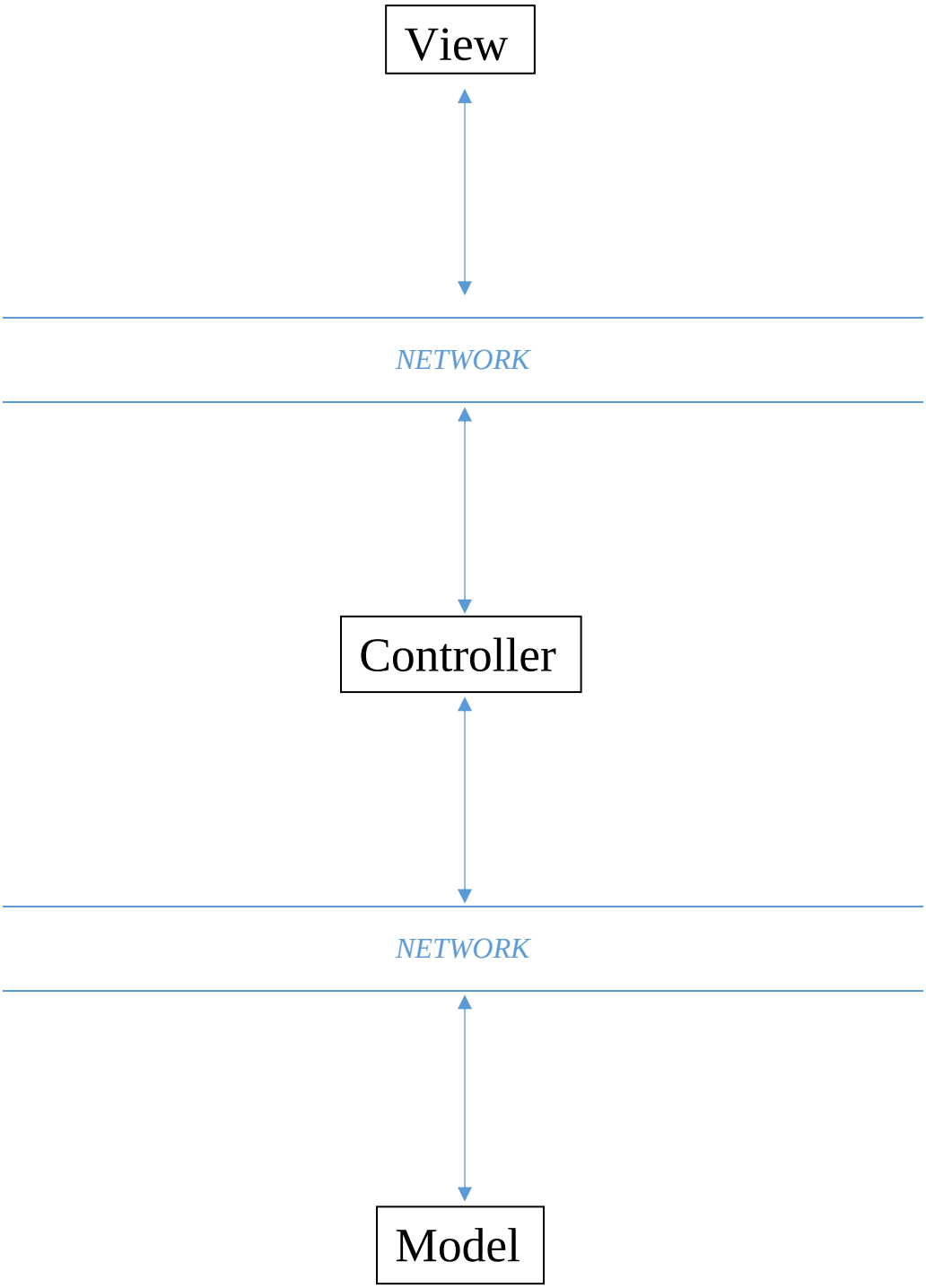
2.2.5 **Format Data:** takes all required data (Name, Description and Date) of the required Event to be formatted for final separation and send as separated data of event

2.1.1 **Check Authentication Message:** Takes the Authentication Message and sends checks if it is valid returning a Boolean answer

2.1.2 **Events JSON to JSON Array Object:** takes the Boolean answer from 2.1.1 and takes the JSON Events object to convert it to an array object of JSON type to be passed

2.1.3 **Map JSON Array to Event Objects:** Takes the JSON Array Object and converts it to Event object to be passed forward and used to be shown

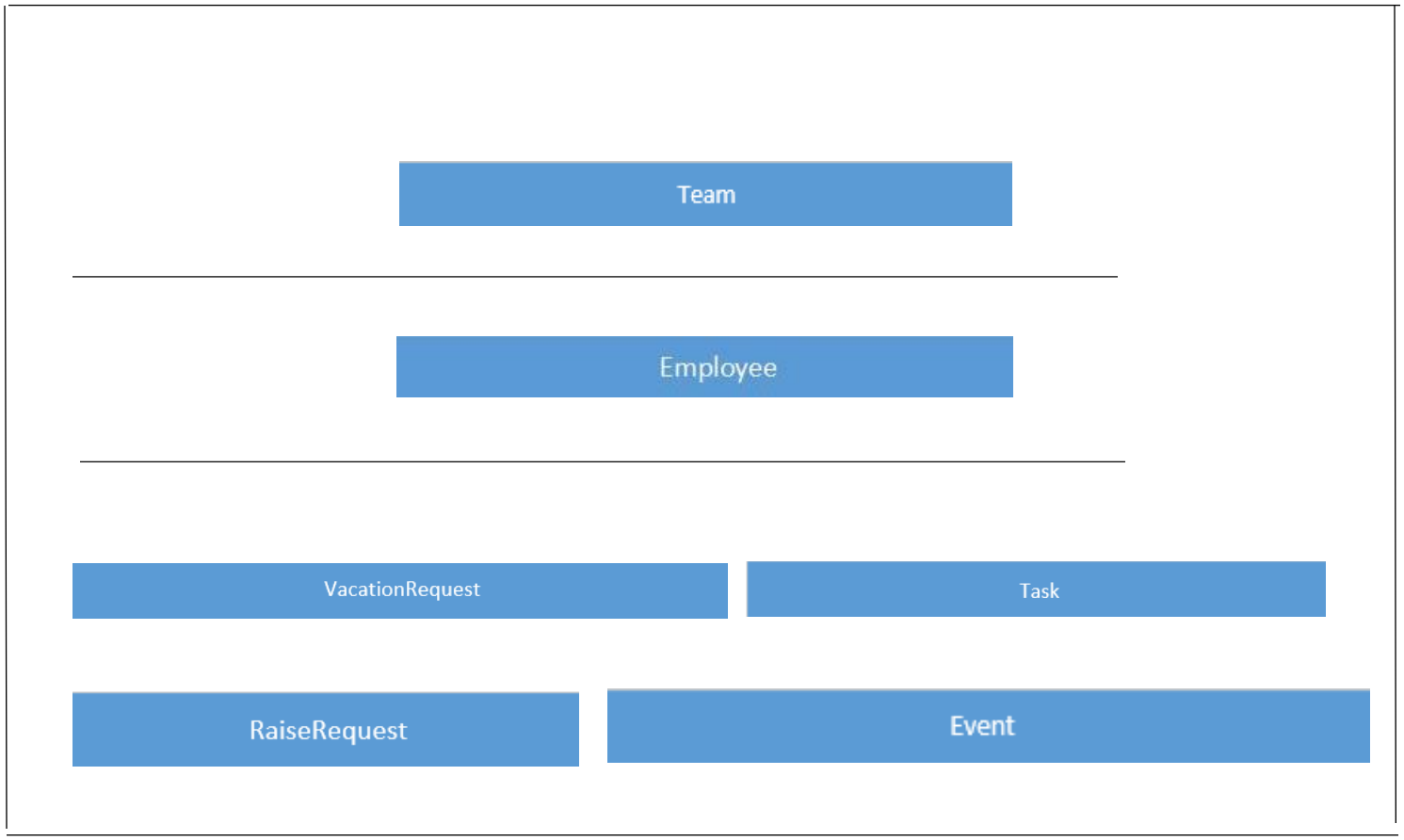
11.0 System Architecture



View:

It is the part of the program the user can see and interact with. It includes the graphical user interface, which includes the windows presented to users when the software starts.

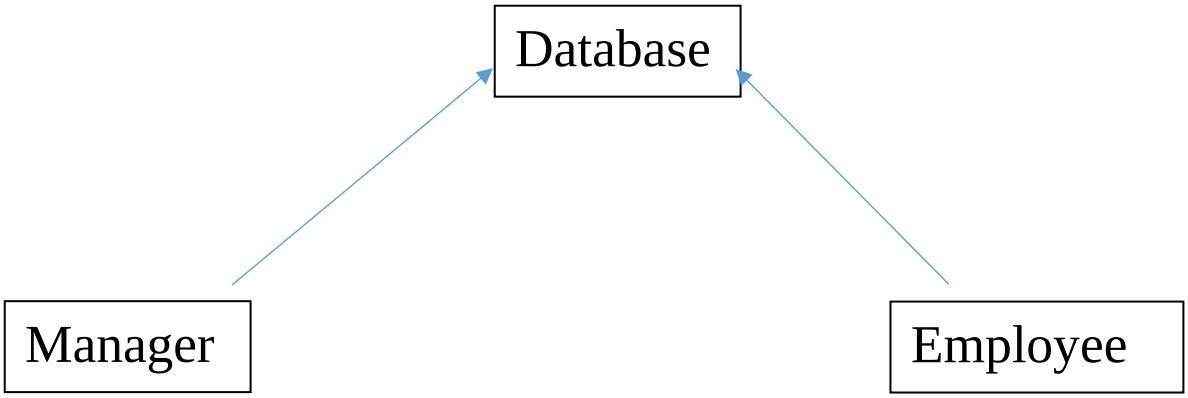
Controller: Layered Architecture



These are all the classes used to create objects, this object oriented architecture is used to represent the controller which is the part responsible of manipulating the data in a program. The controller is represented in terms of classes since these classes are responsible of using and manipulating the program's data, and controls the interaction between the model and the view if the program.

Model:

The model is represented in a data centered architecture since the clients all share the same database. The model is the data used by the program, all the program's data is stored in the database and can be accessed by the company's staff.



Merged Architecture:

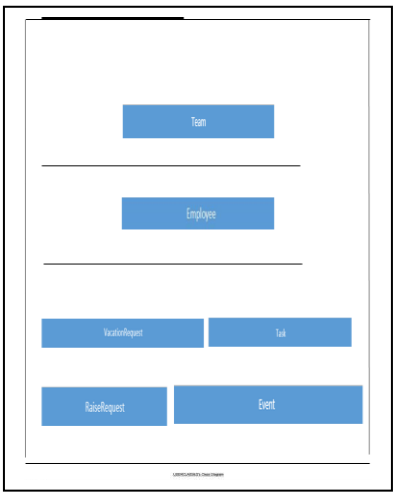
GUI

(Software Frontend)

View

NETWORK

Controller



NETWORK

Model

The merged architecture divides the program structure into three parts represented the MVC structure, while the connection between the parts of the program is showed in a client server structure to show the interconnection of the program parts.

12. OOAD Methodologies and Comparisons

The two chosen methodologies are Jacobson's methodology and Rumbaugh's methodology.

12.1. Jacobson Methodology

This methodology is also known as Object Oriented Software Engineering.

The methodology was adopted as it all scenarios for understanding the system requirements through use cases and use-case diagram.

Use Cases are used to describe the whole system by showing the interaction between all users and the system.

(Refer to section 4 for the use-case diagram and the narrative description for all use cases).

12.2. Rumbaugh Methodology

This methodology approach towards the analysis, design, and implementation is that of OMT (object modeling technique).

This methodology was adopted as it separates the analysis phase into object model, dynamic model and functional model through class diagram, state diagram and data flow diagram respectively.

In the system design phase, the system architectural style is determined. (Refer to section 11 for architectural model).

<u>Analysis phase:</u>

Object model: is presented by the class diagram. (Refer to section 5 for class diagram).

Dynamic model: is presented by the state diagram. (Refer to section 9 for state diagram).

Functional model: describes how data is flowing, where data is stored and how it is processed by the different processes. (Refer to section 10 for DFD diagram).

13.0 Testing

13.1 Class Level Testing

Class testing in object oriented is equivalent to unit testing in conventional software. And, also driven by its operations and the state behavior of the class. Its behavior in terms of its objects can be tested by invoking its operations, so specific test sequences has been used to invoke and test the methods of each class and the effects left afterwards.

Test Sequence:

Employee Class:

1. addEmployee
2. Use setters (e.g. setName, setBirthday…)
3. Use getters (e.g. getName, getBirthday…)
4. selectEmployees
5. updateSpecificOfEmployee
6. updateAllOfEmployee
7. getEmployee
8. getEmployees
9. (Repeat steps from 2 to 8)
10. deleteEmployee.

Results: New Employee added to database using setters all information has been installed to the database, and then getters used to extract info back from the database for user display. Then, adding, updating and deleting of Employee object has been tested and verified by logging in after each test to see results in the app and by checking Database.

Team Class:

1. addTeam
2. Use setters (e.g. setEmployee, setManagerID…)
3. Use getters (e.g. getEmployee, getManagerID…)
4. selectTeams
5. updateSpecificOfTeam
6. getTeams
7. (Repeat from 2 to 6)
8. deleteTeam

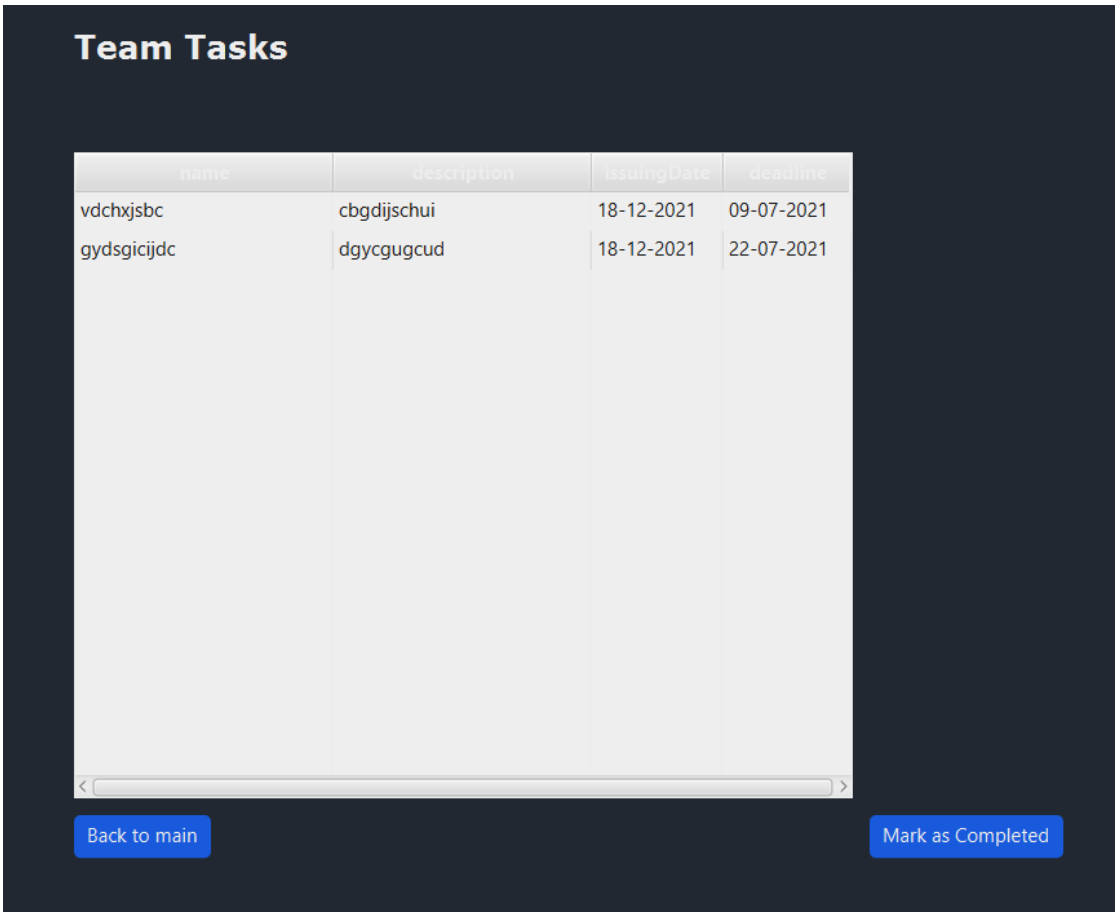
Results: New Team added to database using setters all information has been installed to the database, and then getters used to extract info back from the database for user display. Then, adding, updating and deleting of Team object has been tested and verified by logging in after each test to see results in the app and by checking Database.

Task Class:

1. addTask
2. Use setters (e.g. setName, setDeadline…)
3. Use getters (e.g. getName, getDeadline…)
4. updateTask
5. markAsCompleted
6. (Repeat from 2 to 5)
7. deleteTask

Results: New Task added to Employee using setters and then getters used to extract info back from the database for user display. Then, adding, updating and deleting of Task object has been tested and verified by logging in after each test to see results in the app and by checking Database.

Actual Test:

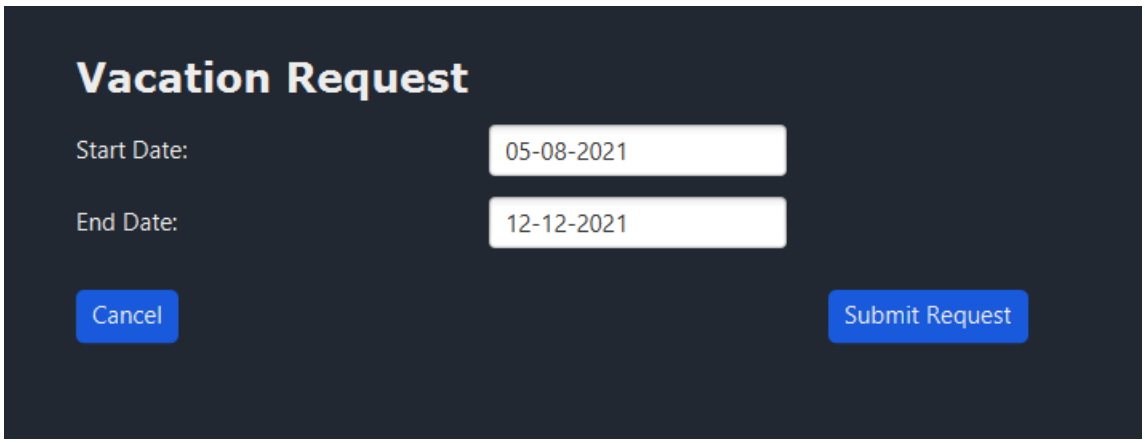


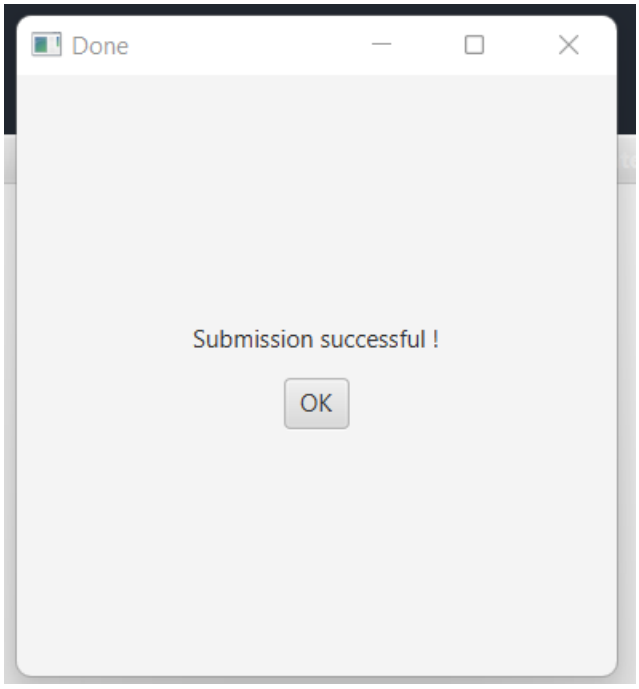
Vacation Request Class:

1. Use setters (e.g. setStartDate, setEndDate…)
2. Use getters (e.g. getStartDate, getEndDate…)
3. submitVacation
4. submitTeamVacation

Results: Submit Vacation Request for Team and individual Employees after using setters and then getters to extract info back from the database for user display. Verified from Database and after logging in from Team manager's ID.

Actual Test:





Raise Request Class:

1. Use setters (e.g. setEmployeeID, setStatus…)
2. Use getters (e.g. getEmployeeID, getStatus…)
3. raiseRequest.

Results: Submit Raise Request from an employee object after using setters and then getters to extract info back from the database for manager user display. Verified from Database and after logging in from Team manager's ID.

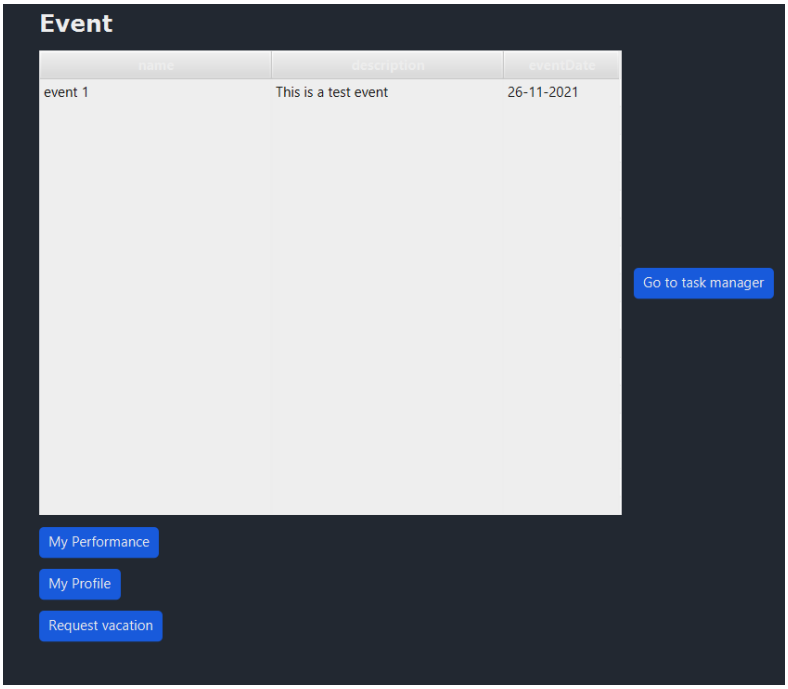
Actual Test:

Event Class:

1. addEvent,
2. Use setters (e.g. setName, setID…)
3. Use getters (e.g. getName, getID…)
4. getEvents
5. updateEvent
6. updateTeam
7. (Repeat from 2 to 6)
8. deleteEvent

Results: New Event added to database using setters all information has been installed to the database, and then getters used to extract info back from the database for user display. Then, adding, updating and deleting of E object has been tested and verified by logging in after each test to see results in the app and by checking Database.

Actual Test:

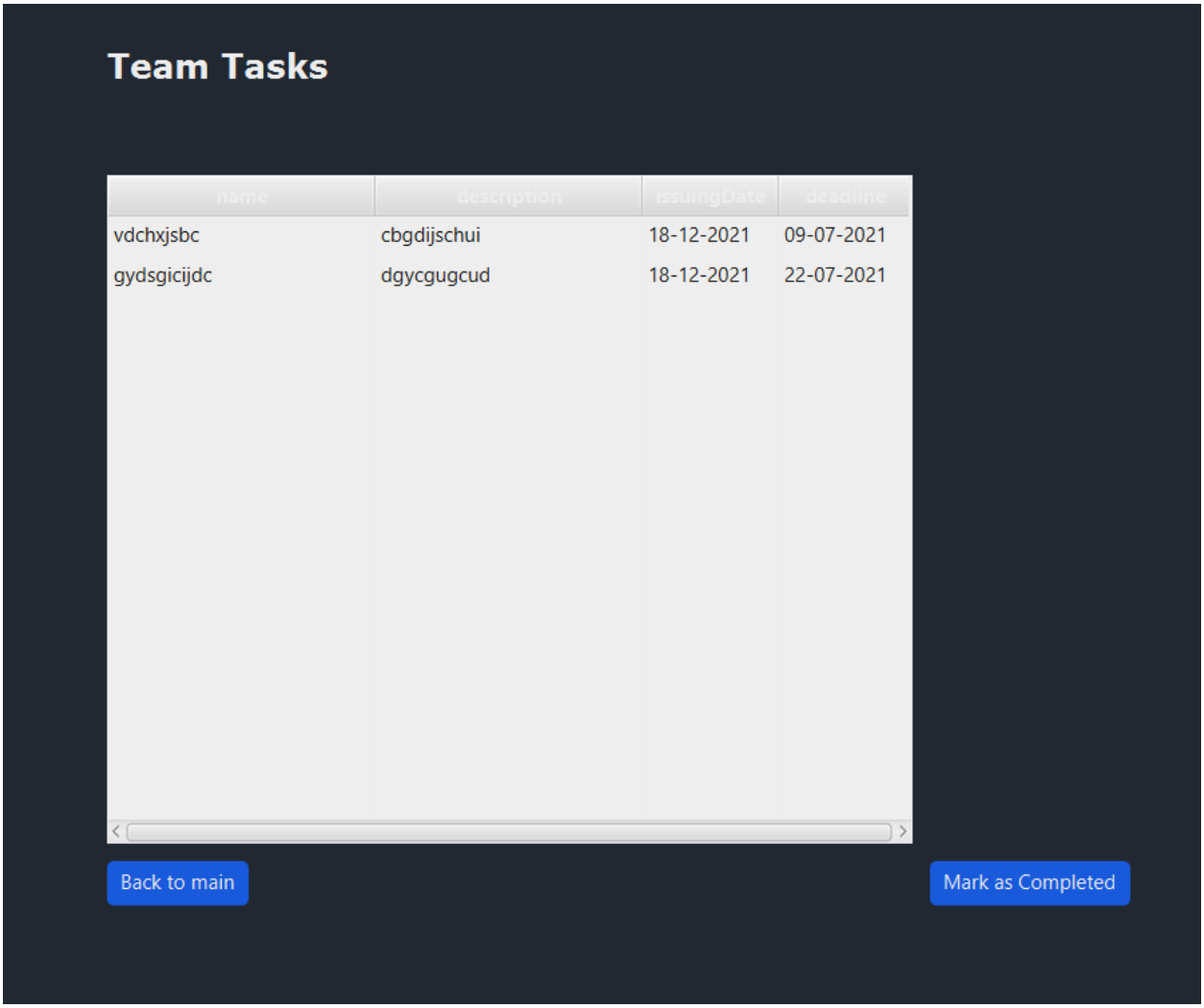


13.2 Integration Testing

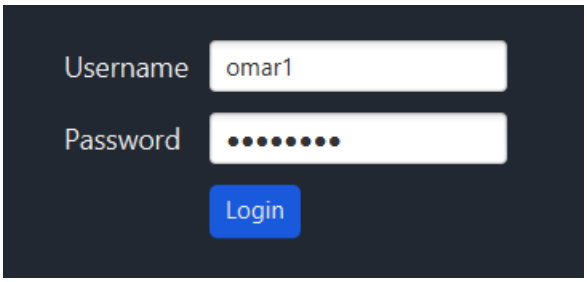
Integration testing used to test interaction between modules and to verify that the different modules are well connected and work well together.

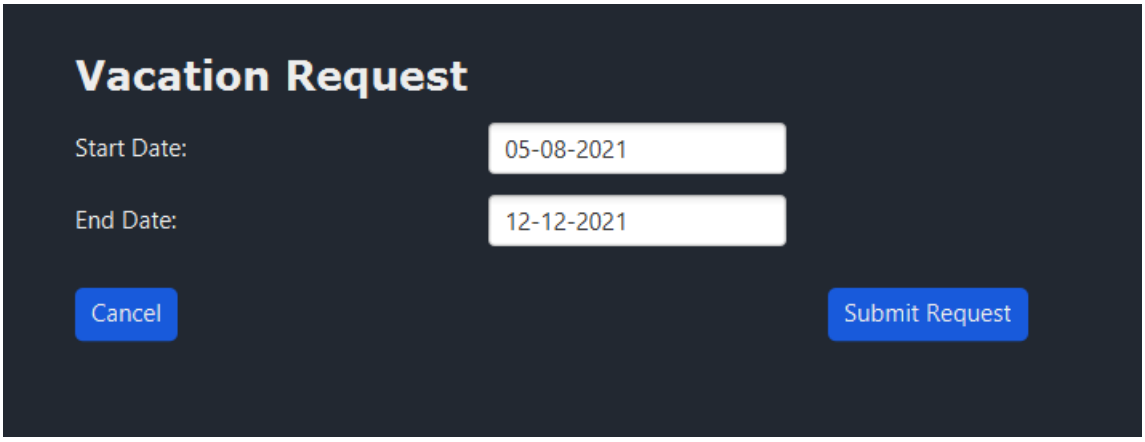
We used to different functionality to ensure the interconnection between modules are working as expected and showing no errors or bugs; before moving on to system testing to test system as a whole.

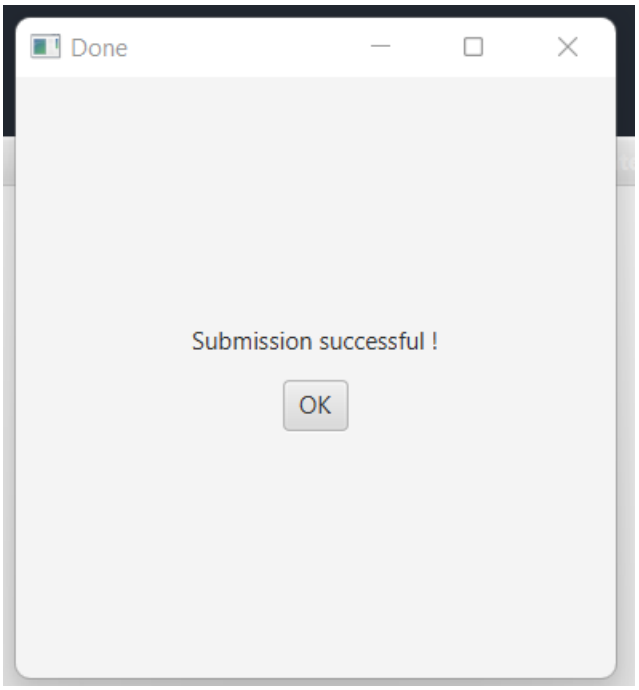
1) **Actual Test -> Check whether Task Class and Team Class are working simultaneously together when joined, by assigning a number of tasks to the current employee's team.**



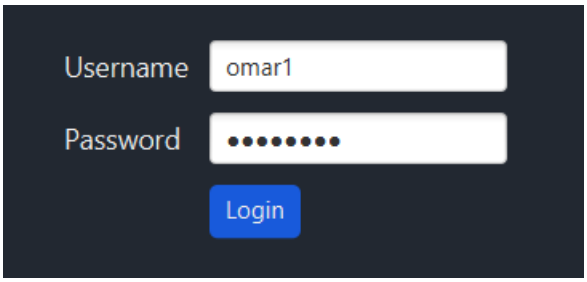
2) **Actual Test -> Check whether Employee Class and RequestVacation Class are working simultaneously together when joined, by assigning a request to the current employee.**

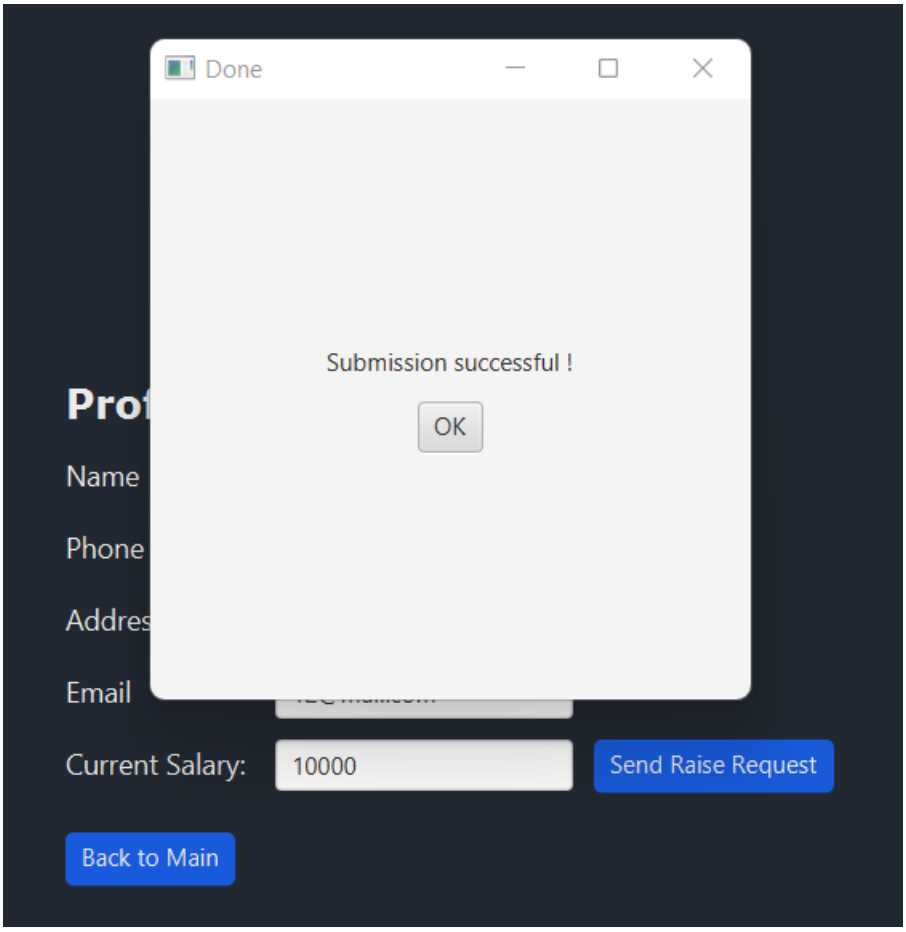




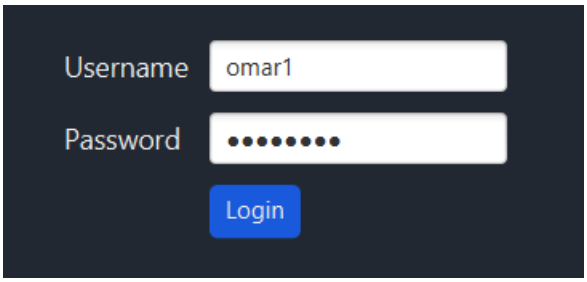


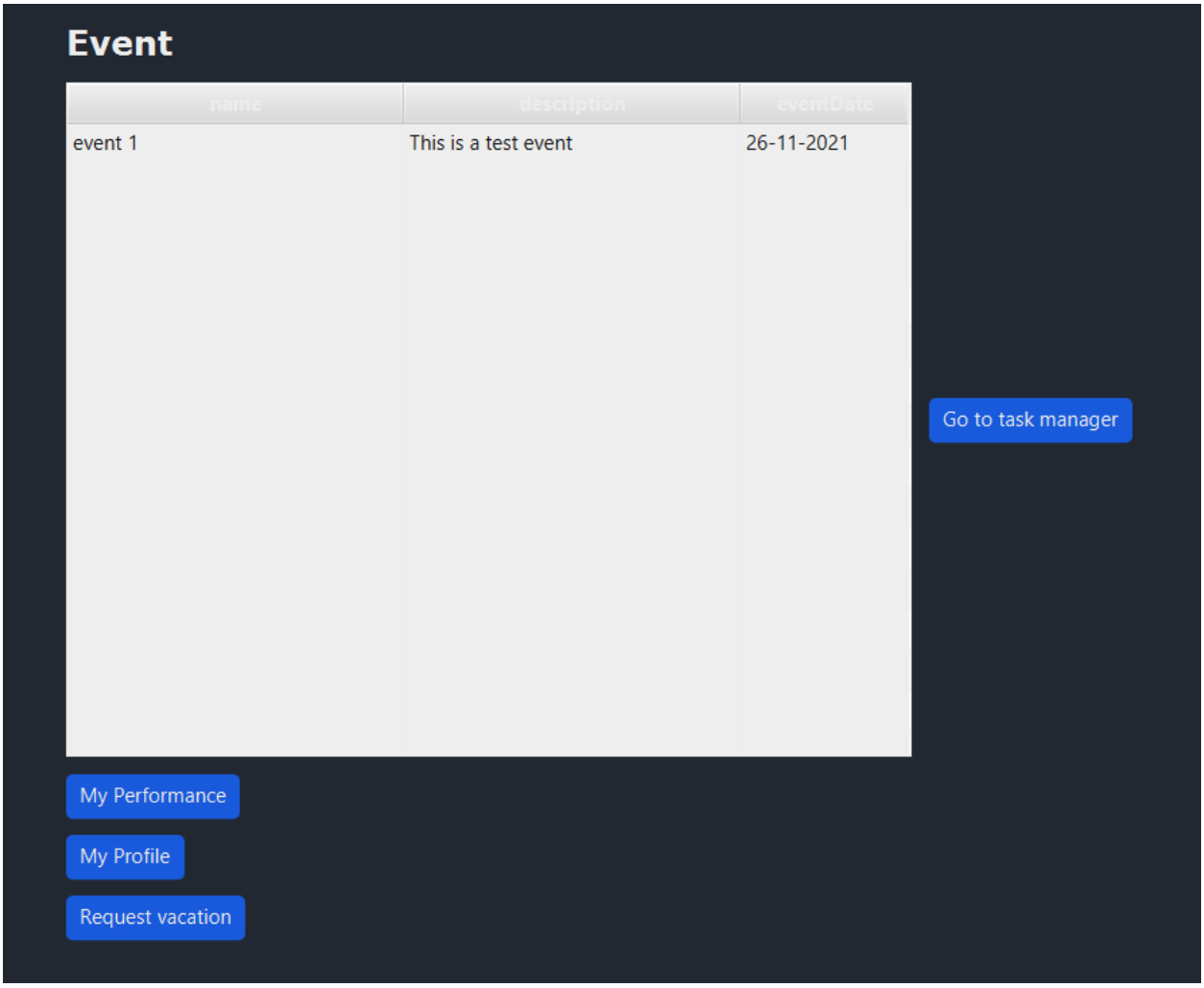
3) **Actual Test -> Check whether Employee Class and RequestVacation Class are working simultaneously together when joined, by assigning a request to the current employee.**





4) **Actual Test -> Check whether Employee Class and Event Class are working simultaneously together when joined, by checking current employee's events at the main screen.**





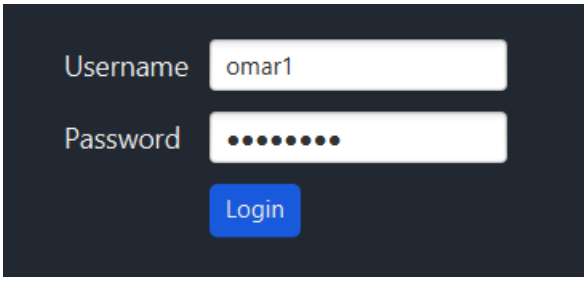
13.3 System Testing

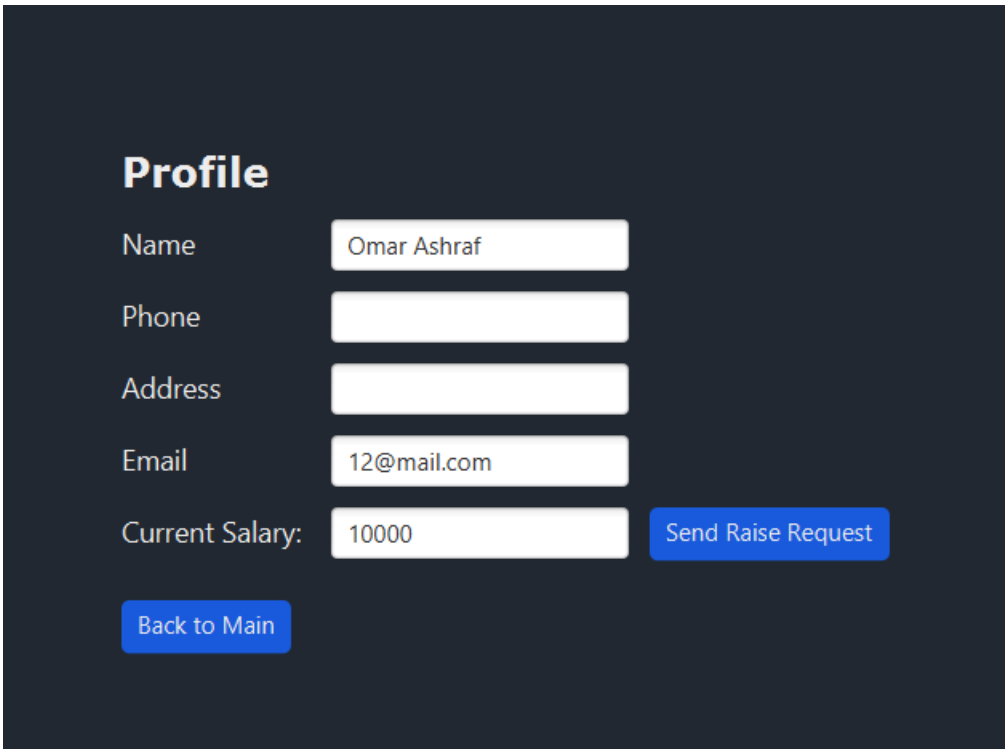
To test the entire system we used black box testing, so our testing did not rely on the way the code was designed and coded, but was evaluated against the specification; to test what the program is supposed to do.

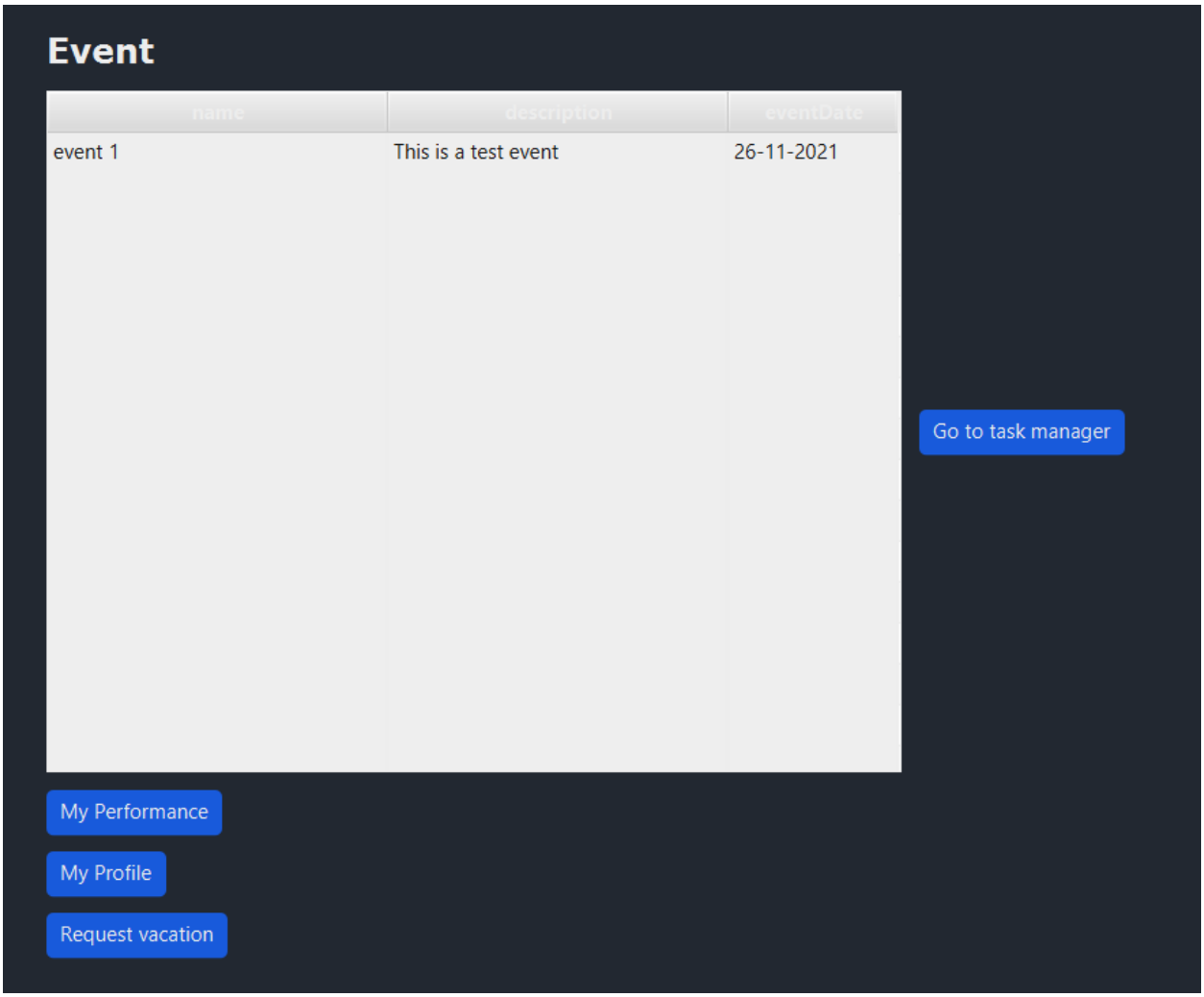
We used general functionality of the system with known outputs and after entering given inputs we compared these expected outputs with the actual outputs to ensure the system is working as expected.

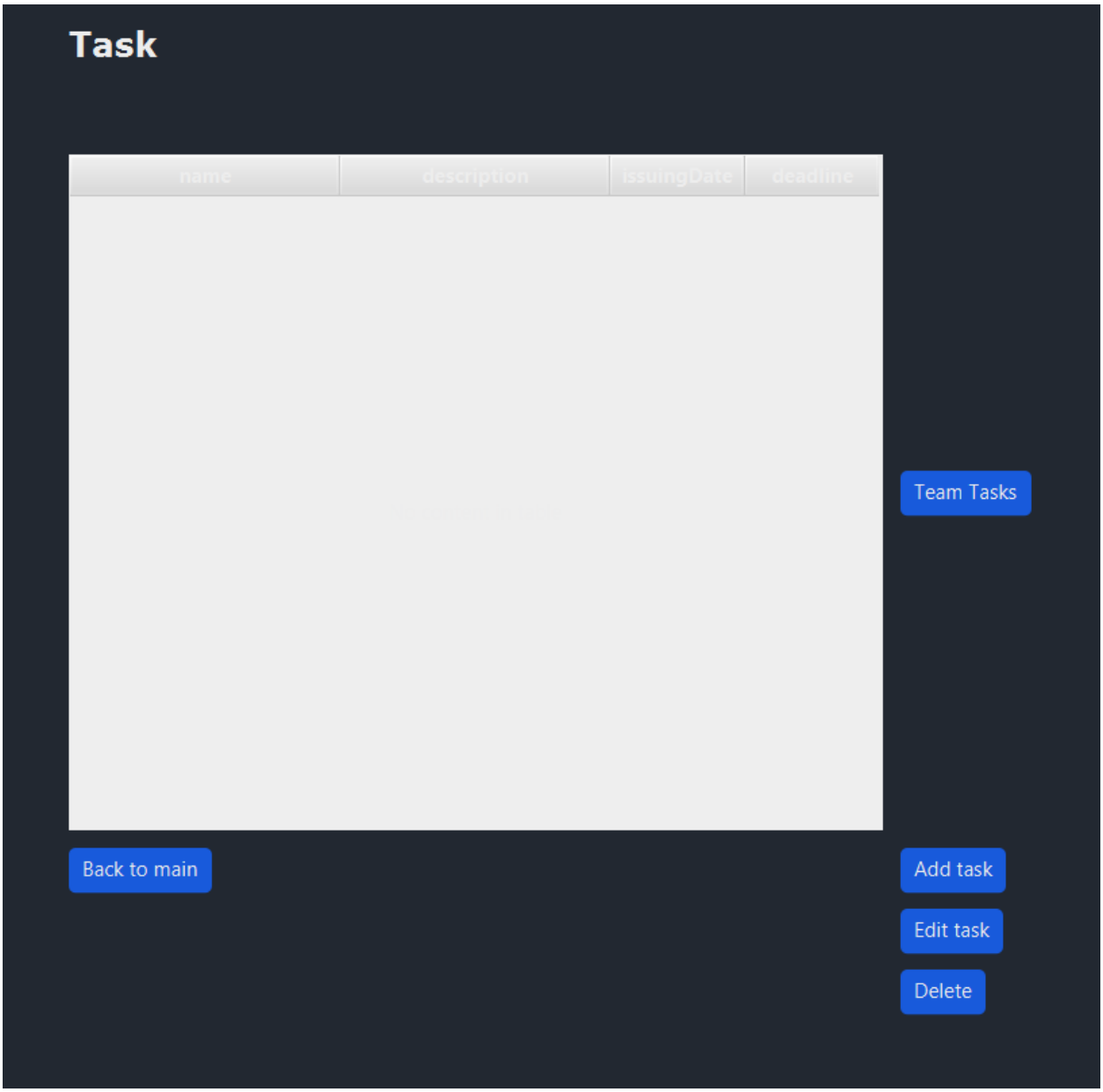
Actual Test -> Create a new Employee with new:

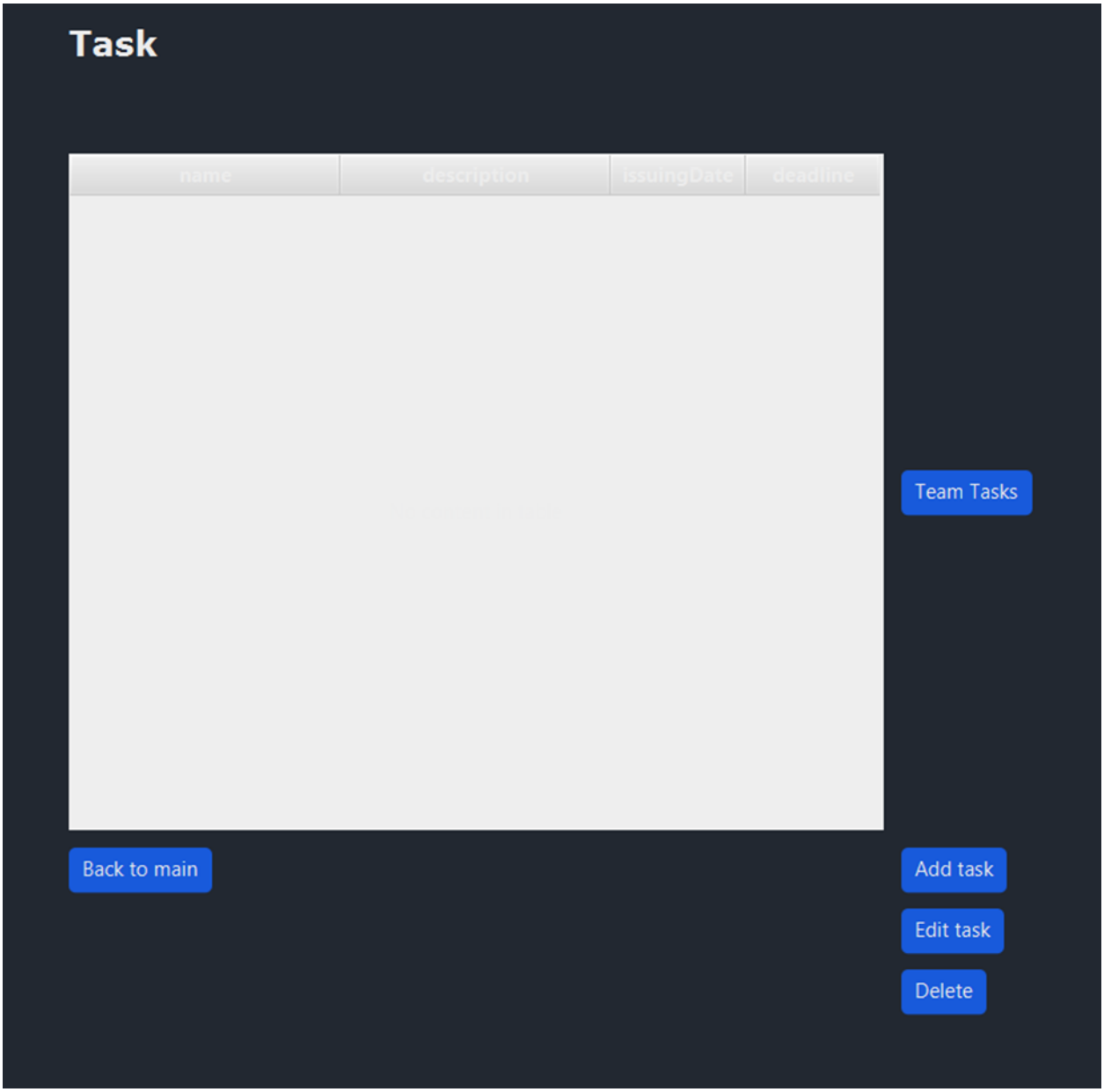
1) **Profile**
2) **Events**
3) **Tasks**
4) **Team Tasks**
5) **Calculated Performance (based on finished tasks)**
6) **Pending Requests**

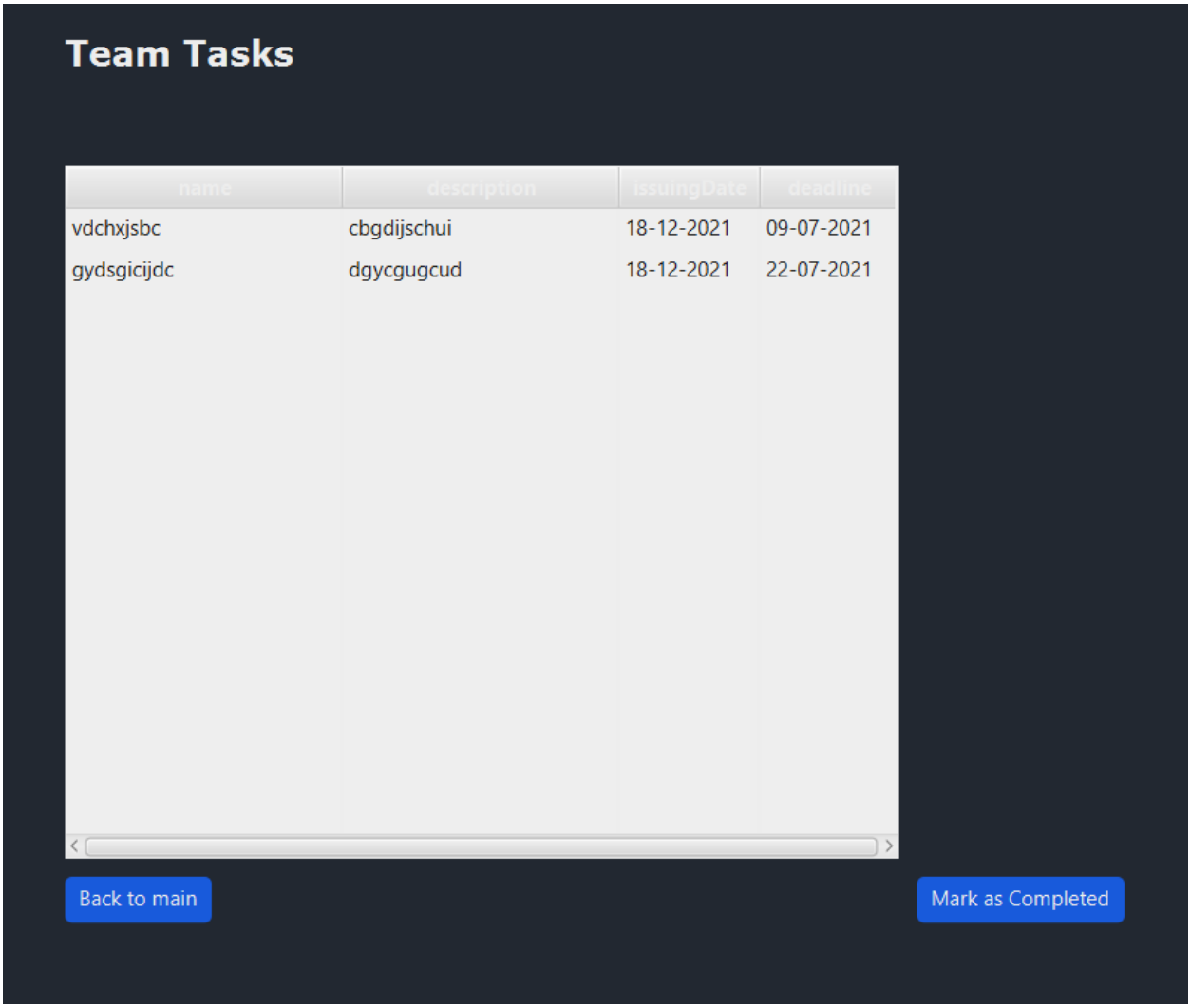


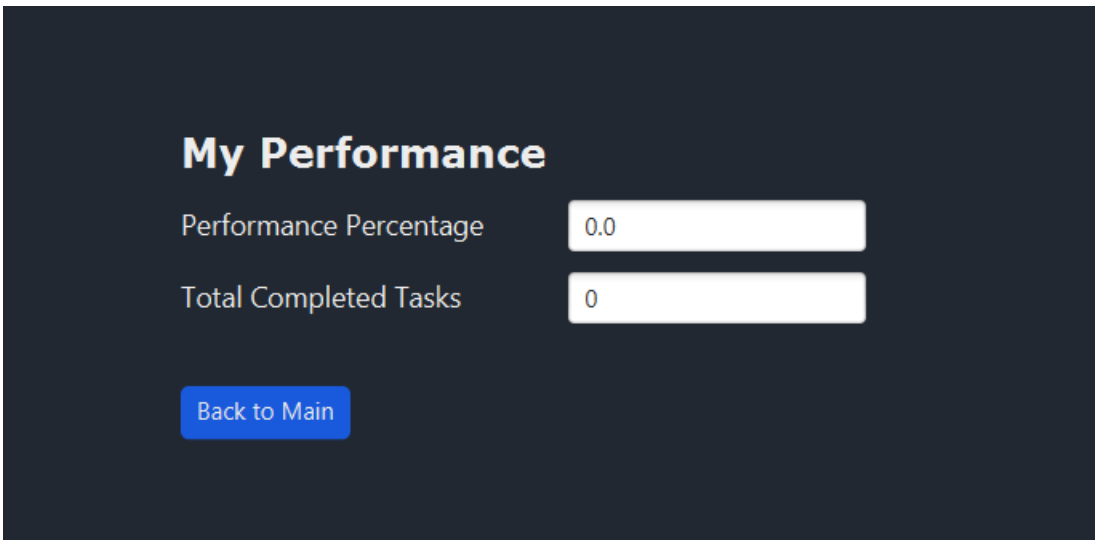

Profile
Name
Omar Ashraf
Phone
Address
Email
12@mail.com
Current Salary:
10000
Send Raise Request
Back to Main


Event
name
description
eventDate
event 1
This is a test event
26-11-2021
Go to task manager
My Performance
My Profile
Request vacation


Task
name
description
issuingDate
deadline
No content in table
Team Tasks
Back to main
Add task
Edit task
Delete


Task
name
description
issuingDate
deadline
No content in table
Team Tasks
Back to main
Add task
Edit task
Delete


Team Tasks
name
description
issuingDate
deadline
vdchxjsbc
cbgdijschui
18-12-2021
09-07-2021
gydsgicijdc
dgycgugcud
18-12-2021
22-07-2021
Back to main
Mark as Completed


My Performance
Performance Percentage
0.0
Total Completed Tasks
0
Back to Main

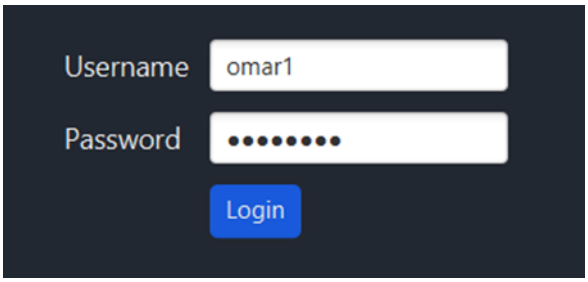
13.4 Other possible tests

Beside these tests performed above there are other testing modes that could be possibly applied on such software.

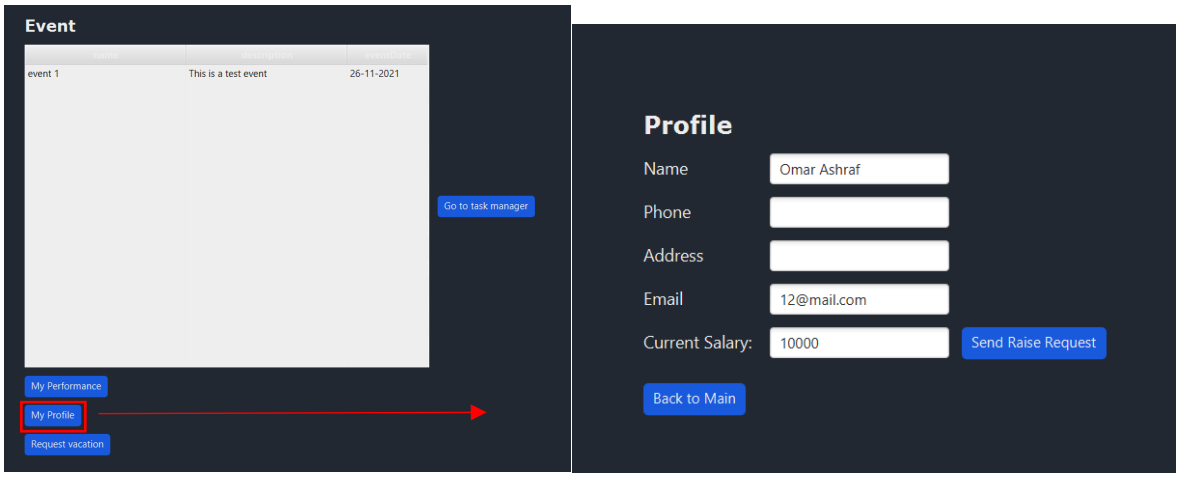
1) Beta testing: in this type of testing, we deploy the software to the client's environment and leave the user to perform his own testing then get his feedback before the final software deployment.
2) Sanity testing: after the beta testing we get feedback from the user. If there is a bug that needs to be fixed then we need to apply the sanity testing after fixing the bug to ensure that this problem has been addressed appropriately.
3) Spike testing: in this test we try to stress the system with a large number of users to see its capabilities. This case will be obvious when all the employees log into all at the same time which occurs at the start of the working day by default. So, this test is important because the system actually faces spikes of users every working day.
4) Stress testing: this testing mode is used to find the limits of the system through demanding resources in an abnormal manner. Imagine for example that a user creates 1000 tasks and waits for the systems response, in real life situations it is far beyond possible that a user could have 1000 tasks at the same time.
5) Smoke testing: of course, the system is viable to face smoke testing since it is the first output of the SDLC so if we use an evolutionary model where we could add additional functionalities to the system, we will be required to apply the smoke testing to ensure that the basic functionalities are working correctly.
6) Regression testing: as stated in the point above this software is subjected to updates so any code change will require regression testing to ensure that the added code hasn't affected any existing functionality.

14.0 User Guide

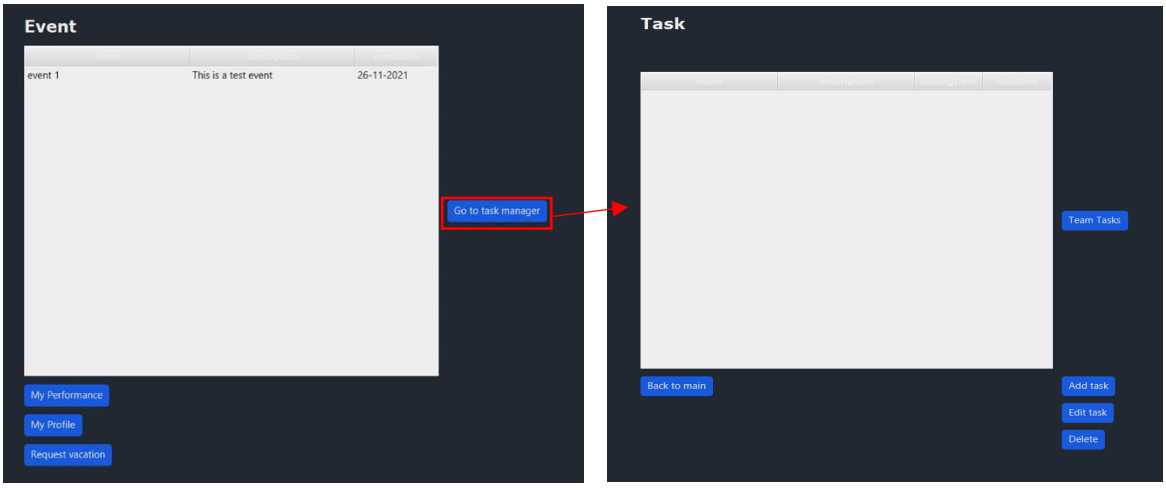
1) Login: the user enters the username and password, provided that he is added to the system by the admin, then it's the login button.



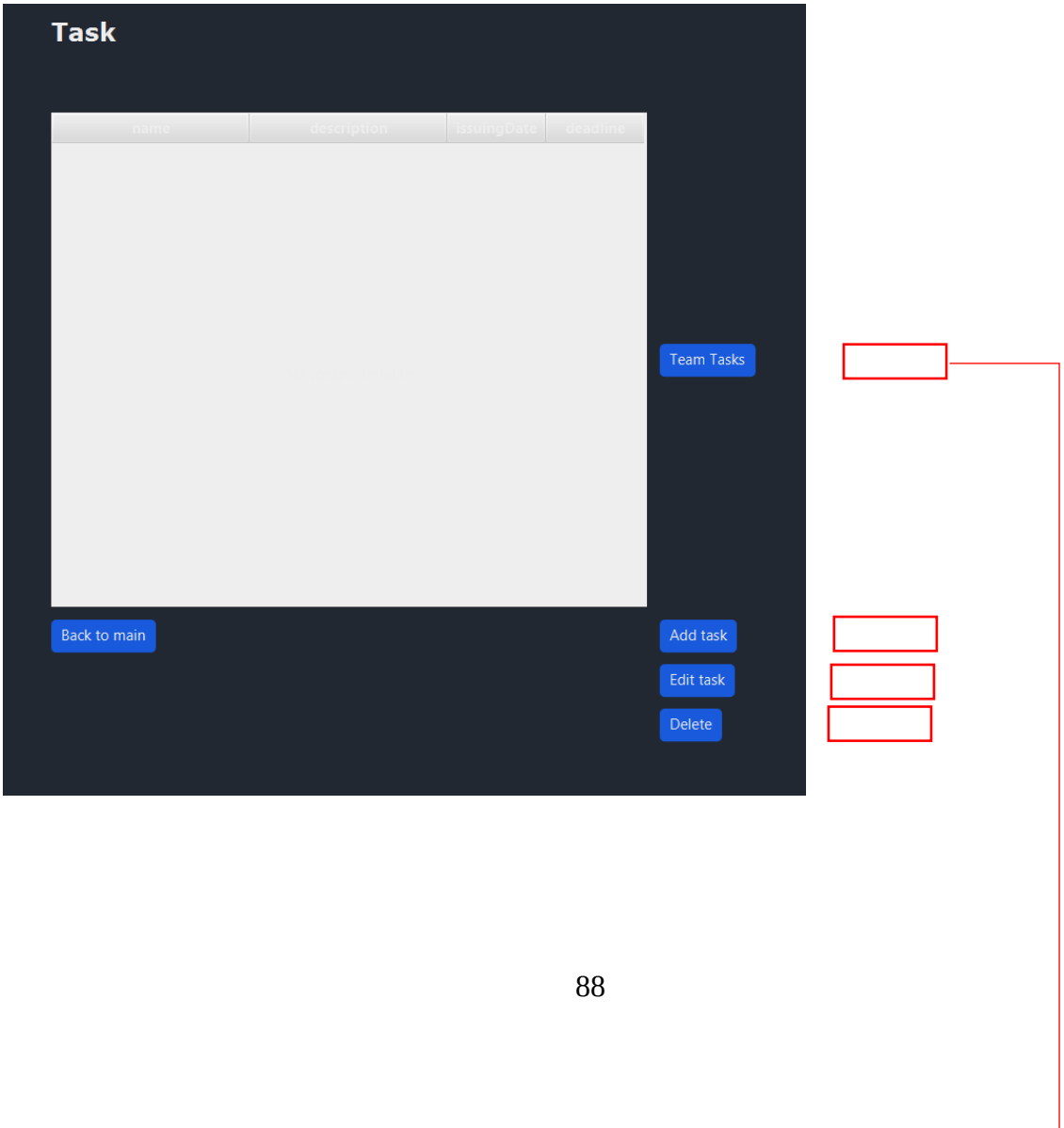
2) Change profile: in this part the user can change his profile data or request a raise in the salary this of course requires the user to be loge in to the system then when the main screen shows up you press profile to take you to the profile page.

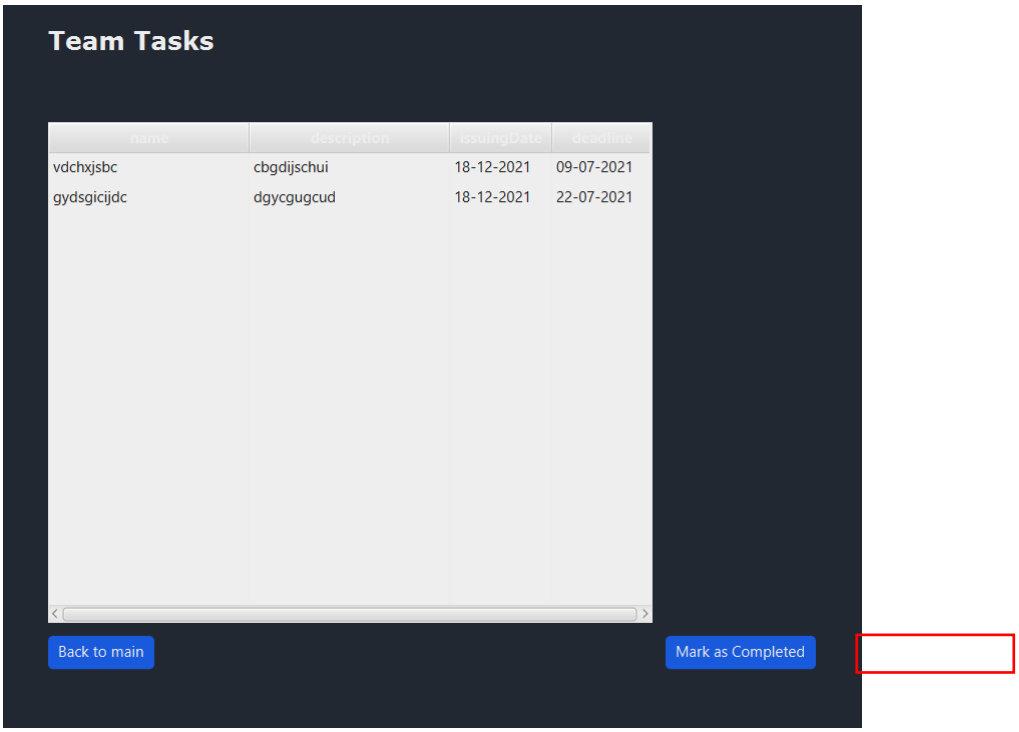


3) Task management

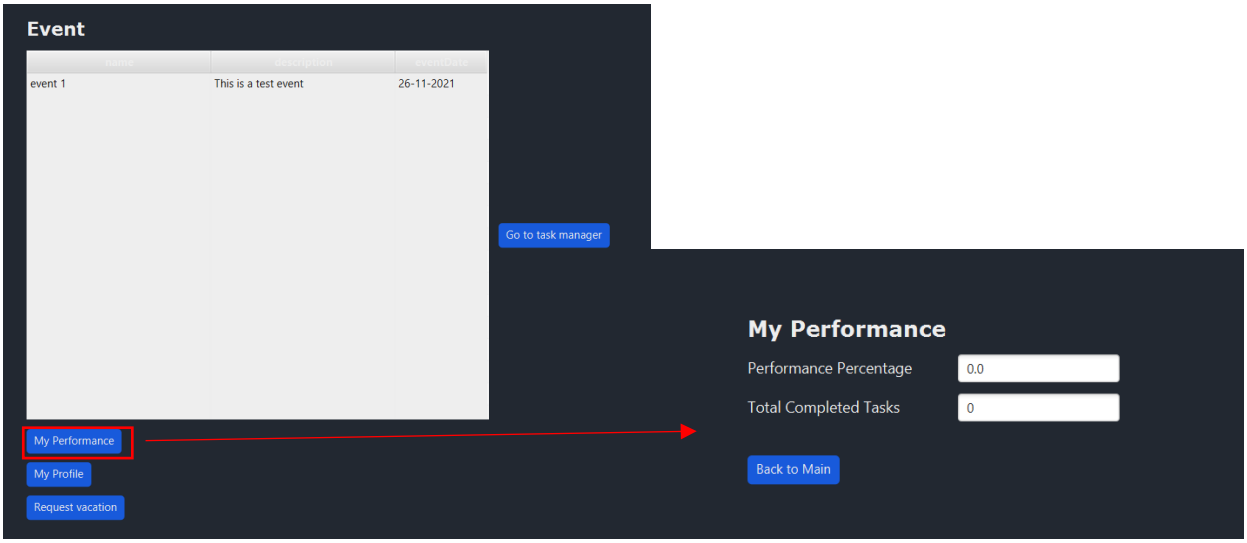


From the task screen the user could add task, edit task, delete task, mark task as finished, and finally if he is in a team, to view team tasks.

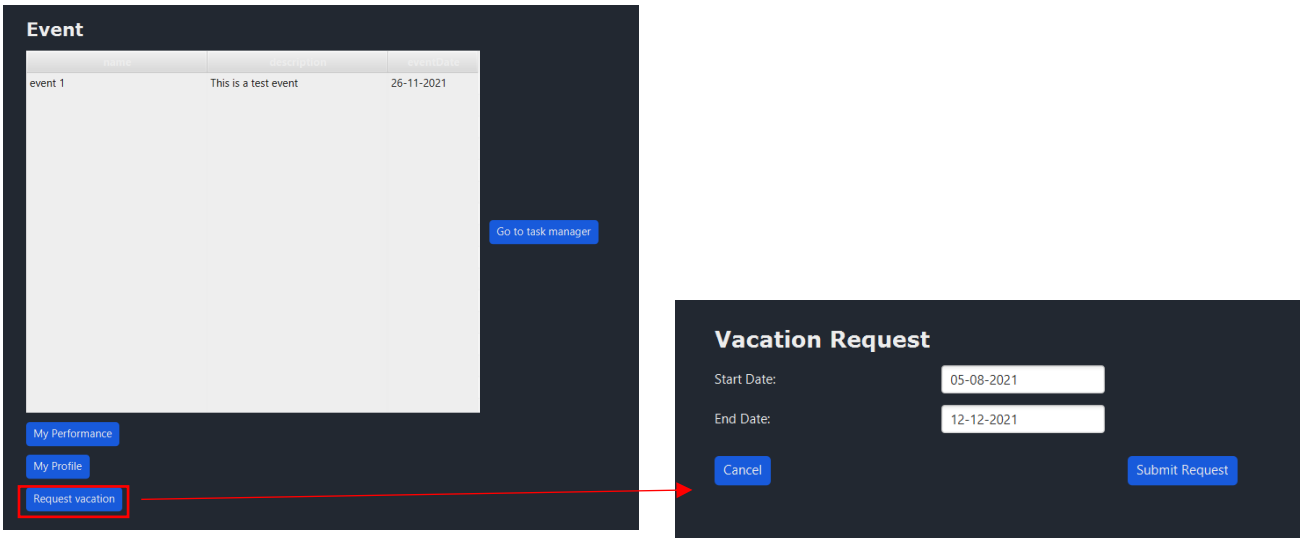




4) View performance: from the main dashboard the user will press on my performance button to open a performance window where he could view his completed tasks as well as his performance percentage.



5) Request vacation: from the main dashboard the user will press request vacation button, a new window will appear where he should enter the required vacation start and end dates, then he will await till he receives a notification respond to this request.



15.0 Cost Estimation

Function points

1. Backup and recovery -> 5

2. Data communication -> 3

3. Distributed processing functions -> 1

4. Is performance critical? -> 1

5. Existing operating environment -> 0

6. On-line data entry -> 5

7. Input transaction built over multiple screens -> 0

8. Master files updated on-line -> 4

9. Complexity of inputs, outputs, files, inquires -> 3

10. Complexity of processing -> 0

11. Code design for re-use -> 4

12. Are conversion/installation included in design? -> 3

13. Multiple installation -> 0

14. Application designed to facilitate change by the user -> 3

Function points total: 32

Measurement parameter	**count**	**Weighting factor**	
Number of user inputs	4	4	16
Number of user outputs	5	5	25
Number of user inquires	8	4	32
Number of files	8	8	64
Number of ext.interfaces	0		
Count-total	25		137
Complexity multiplier			1
Function points			32

$$\mathrm{FP} = \mathrm{UFC} * \left[0.65 + 0.01 * \sum_{i=1}^{i=14} F_i \right] = 137 * [0.65 + 0.01 * 32] = 133$$

Productivity of one person is 10 function points per month.

Team consists of 6 persons, time taken by team of 6 is about 2 months.

Average salary for a person per month for a person is 8000 LE.

Salary for the team of 6 for a duration of 2 months is 96000 LE.

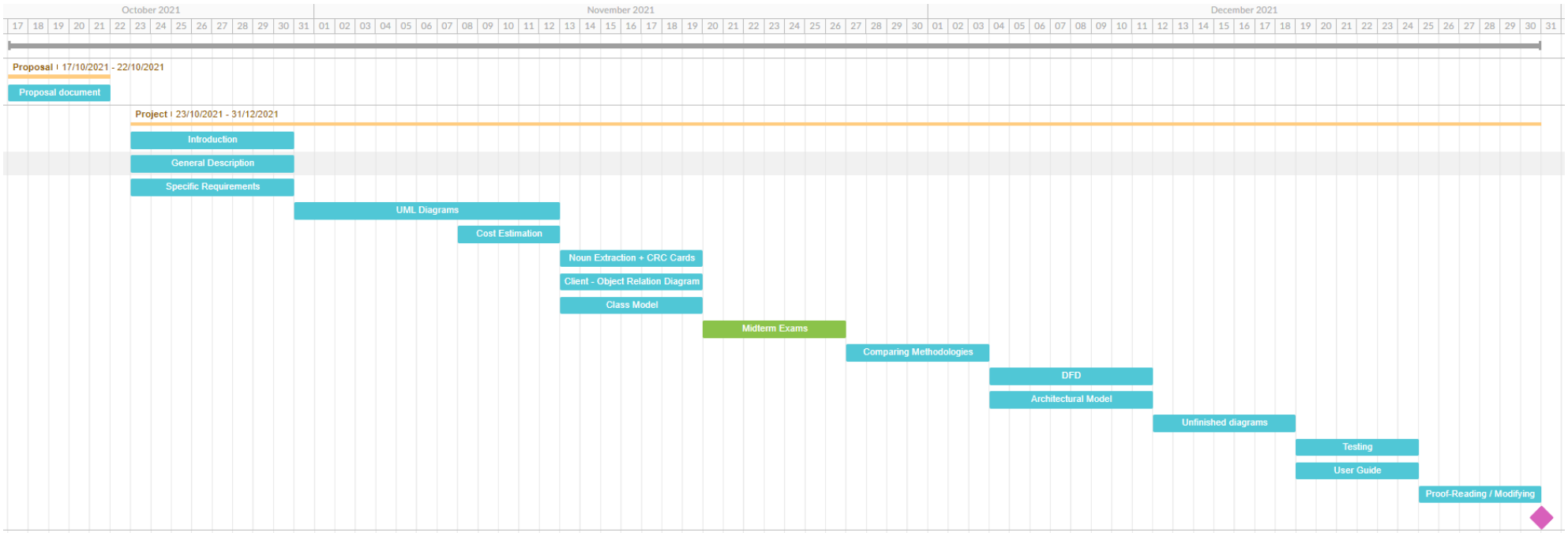
Free IDES are used no cost for licenses for programs.

For a profit of 20 %

$\text{Software price} = \text{cost} + \text{profit} = 96000 + 0.2 \times 96000 = 115200 \text{ LE.}$

$\text{Effort} = \text{A} \times Size^{B} \times 1 = \text{A} \times (133 \times 66)^{B} \times 1$

16.0 Time Plan



SOFTWARE ENGINEERING PROJECT

END OF DOCUMENT