**FPT Software Academy**

NguyenTC8Mock Project

**Time Keeping Software - Detailed Design**



**Version – 1.0**

**27th July 2024**

NNN BishopsXXXX

London

EC2N 4AY

Tel: +44 (0)20 7890 1234

Fax: +44 (0)20 7890 1235

# **Table of Contents**

[Table of Contents 2](#_Toc172964741)

[1 Introduction 4](#_Toc172964742)

[1.1 Scope 4](#_Toc172964743)

[1.2 Definitions, Acronyms and Abbreviations 4](#_Toc172964744)

[1.3 Overview 4](#_Toc172964745)

[1.4 References 6](#_Toc172964746)

[2 Software Requirements 7](#_Toc172964747)

[3 Design 8](#_Toc172964748)

[3.1 System Overview 8](#_Toc172964749)

[3.2 Sequence diagram 10](#_Toc172964750)

[3.2.1 Admin Mode 11](#_Toc172964751)

[3.2.2 Normal Mode 16](#_Toc172964752)

[3.3 Structure diagram 18](#_Toc172964753)

[3.3.1 Load data from file to build linked list 18](#_Toc172964754)

[3.3.2 Save to file to store data 19](#_Toc172964755)

[4 Implementation 20](#_Toc172964756)

[4.1 VendorExt-Project Time Keeping Software 20](#_Toc172964757)

[4.2 Source code structure 20](#_Toc172964758)

[4.2.1 Source code file name of components: 20](#_Toc172964759)

[4.2.2 Directory structure 20](#_Toc172964760)

[4.3 Common 21](#_Toc172964761)

[4.3.1 Constants 21](#_Toc172964762)

[4.3.2 Structure list 21](#_Toc172964763)

[4.3.3 Function list 22](#_Toc172964764)

[4.4 Features Detail 36](#_Toc172964765)

[4.4.1 Admin Mode 36](#_Toc172964766)

[4.4.2 Normal mode 46](#_Toc172964767)

RECORD OF CHANGES

|  |  |  |  |
| --- | --- | --- | --- |
| **Document Version History** | | | |
| **Version** | **Date** | **Author** | **Details** |
| 0.1 | 15th July, 2024 | Diep Pham | New Creation |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**APPROVERS**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Distribution List** | | | | |
| **Name** | **Role** | **Signature** | **Date** | **Version** |
| NguyenTC8 | Trainer |  |  | 1.0 |

**CONFIDENTIALITY**

This document is distributed on a restricted basis, is commercial in confidence to the recipient, and may not be used for any purpose other than that associated with an FPT Software Academy project. The contents of this document may not be disclosed to any third parties without the expressed advance written authorisation of FPT Software Academy.

# Introduction

In the modern era, as businesses continue to expand rapidly, the number of employees also rises significantly. Managing employee payroll becomes increasingly time-consuming if companies persist in using manual methods.

To streamline working time management and employee salary calculation, the “Time Keeping System” emerges as an invaluable tool for your company. This system is meticulously designed to automate and enhance the recording of employee check-in and check-out times, simplifying both timekeeping and salary calculations for businesses.

In essence, the “Time Keeping System” is a cutting-edge solution that effectively tracks and records daily attendance of employees. It ensures precise recording of check-in times when employees arrive and check-out times when they depart. By maintaining accurate records, this application aids the company's accounting department in calculating salaries based on actual hours worked, thereby ensuring fairness and efficiency in payroll management.

## Scope

As mentioned above, timekeeping software will support your company in managing all employee information and daily working time, as well as the reasons for employee salary deductions.

Definitions, Acronyms and Abbreviations

Table : Abbreviations in this document

|  |  |
| --- | --- |
| **Acronym** | **Reference** |
|  |  |
|  |  |
|  |  |
|  |  |

Overview

This document includes following sections:

1. **Introduction:**

This section introduces some overview information about Project Time Keeping Software

1. **Software Requirement:**

This section contains excerpted parts from the requirement (SRS document) which relate to Project Time Keeping Software to focus the functionality.

1. **Design:**

This section includes diagrams which describe functionality, from outline to detail, in Project Time Keeping Software. The diagrams are block diagram, sequence diagram, and others.

1. **Implementation:**

This section describes followings:

* List of parameters which are corresponding to requirement document.
* Source code’s directory structure of Project Time Keeping Software.
* Description about template files’ source code.
* List of constants, enum types, structures, other user-defined data types which are used.
* List of functions with detailed descriptions for using in source code.
* Some notice when coding Project Time Keeping Software.

References

|  |  |
| --- | --- |
| **Documentation Name** | **Description** |
| HCM24\_FRF\_EMB\_04 - CF Module - Mock Project.pdf | System Requirement Specification document which contains requirement of all features. |
| https://en.wikipedia.org/wiki/SREC\_(file\_format) | Technical specification about the S-Record file format. |
|  |  |

# Software Requirements

**Table 2: Software requirement table for Project Time Keeping Sofware**

|  |  |  |
| --- | --- | --- |
| **Module YYYY** | | |
| **No.** | **Requirement** | **Note** |
| **1** | **Admin mode** |  |
| **1.1** | Administrator must use secure default password to log in (12345678) |  |
| **1.2** | Administrator can change new password |  |
| **1.3** | The administrator can add new employees (including ID, full name, and basic salary). |  |
| **1.4** | The system will automatically generate unique employee ID to avoid duplication. |  |
| **1.5** | Administrator can view the list of all employee information |  |
| **1.6** | The administrator can view the working time records of any specified employee. |  |
| **1.7** | The system must be able to calculate salaries for all employees based on their actual working hours, including penalties for late check-in and early check-out. |  |
| **2** | **Normal mode** |  |
| **2.1** | Employees can check in by entering their Employee ID and confirming their identity. |  |
| **2.2** | Employees can check out by entering their Employee ID and confirming their identity |  |
| **2.3** | The system must automatically record the current time as the check-in or check-out time upon confirmation. |  |
| **2.4** | The system must save the recorded check-in and check-out times securely in a data file for future reference and salary calculations. |  |
| **2.5** | The system must provide a user-friendly interface |  |

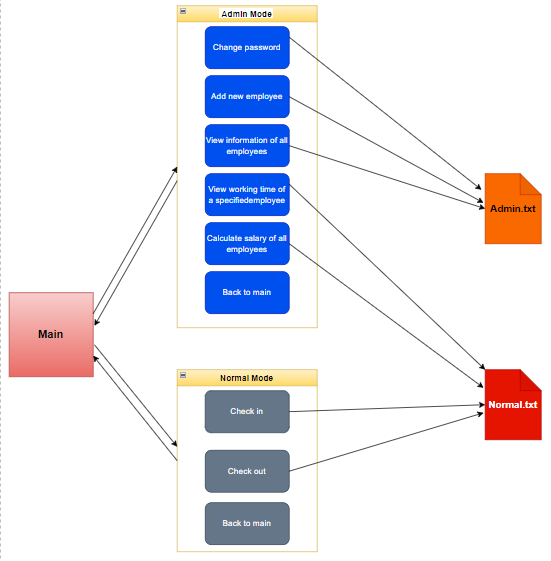
# Design

This section includes diagrams which describe behavior of components in Project Time Keeping Software.

The diagrams shown following are written in UML language using StarUML tool.

System Overview

The following diagram shows system architecture of PROJECT XXXX



**Figure 1: PROJECT TIME KEEPING SOFTWARE block diagram**

Currently, Project Time Keeping Software has 2 components Admin Mode and Normal mode

Behavior outline:

* Admin Mode: have 6 feature. When you access to this mode, you have to input password correct (default is: 123456789)
  + Change password. After change the password, the new password will be encrypted
  + Add new emloyee with 2 requires: name and basic salary of employee.
  + View information of all employees. Display ID, name and basic salary of each employee.
  + View working time of a specified employee by ID.
  + Calculate salary for all employees. System will requir input as the start time and end time to caculate salary (for the year)
  + Back to main
* Nornal mode: have 3 feature. This feature can not use for Saturday and Sunday cause that is weekend.
  + Check in
  + Check out
  + Back to main

Sequence diagram

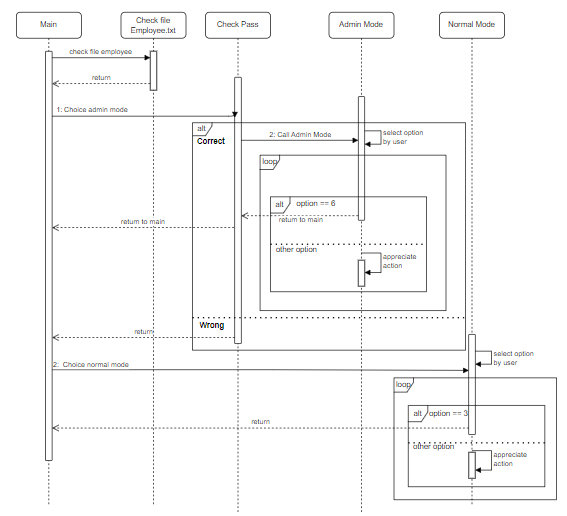


Figure : Overall sequence diagram

When turn on system, the screen will be displayed 3 option: admin mode, normal mode and quit program. As the same time, program check file Employee.txt, if not it will create a new file.

### Admin Mode

#### Change Pass Word

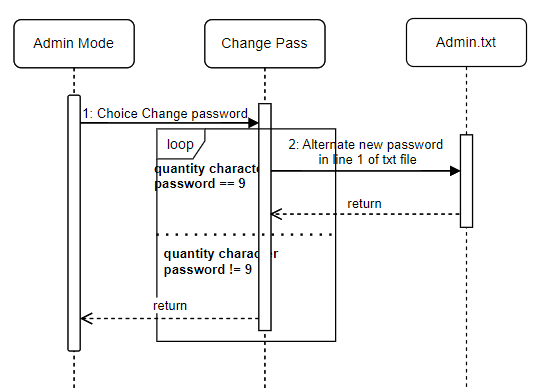


Figure 3: Change Password sequence diagram

When user select 1: Change password then Function Change Pass will be executed. This function require user input new password and check wheter the new password quantity of new password has enough 9 characacters. Password will be alternate after check.

#### Add new employee

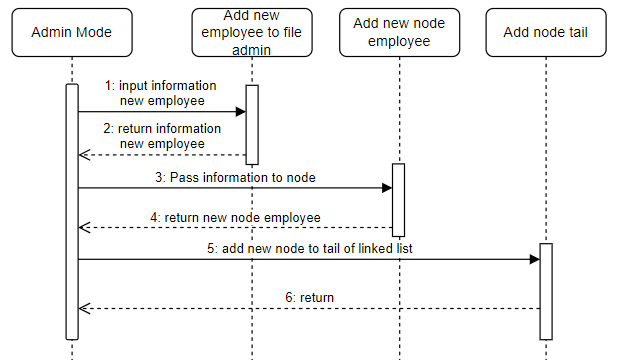


Figure 4: Add new employee sequence diagram

When user select 2 to choice feature add new employee. Program will sent a require input information of new employee include name and basic salary. After that program will create information of employee to Admin.txt file and program use that information to create new node employee then add node tail in linked list.

#### View information of all employee

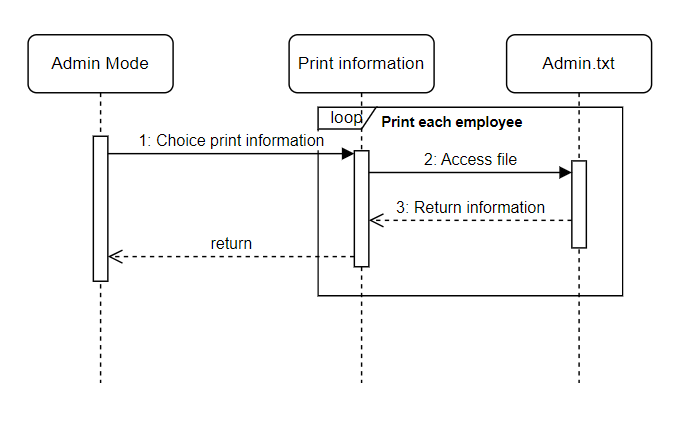


Figure 5: Print information of all employee

When user select 3, choice feature print information of all employees (ID, name, basic salary). The program will retrieve information from the Admin file and send back to print data.

#### View working time of a specified employee

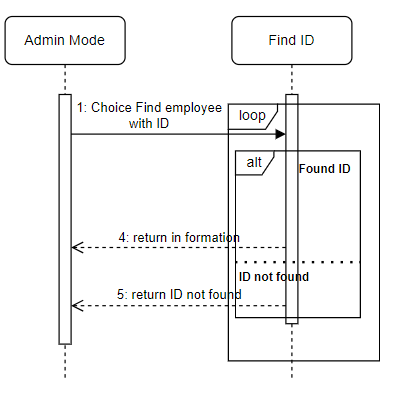


Figure 6: Print information with ID

When user select 4. Program will require input ID of employee than user want to view the working time, after user enter, the program will run a search loop in linked list. If there is an ID, it will print out the employee’s working information. Otherwise it will return “ID not found”.

#### Calculate salary of all employees

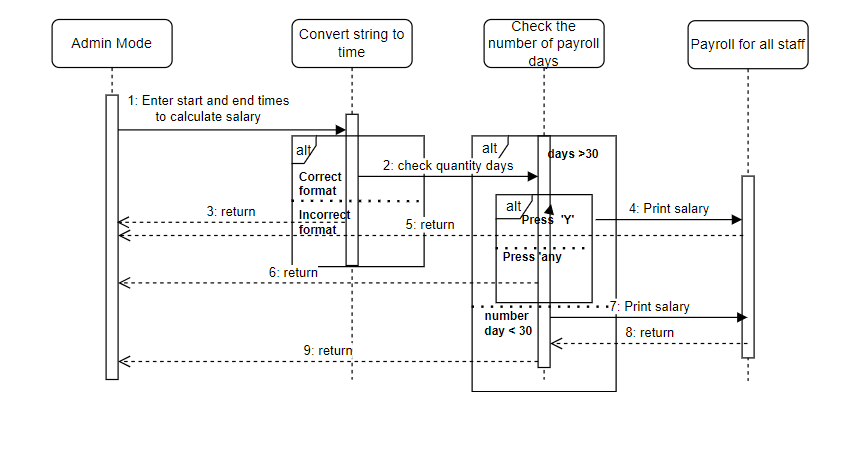


Figure 7: Calculate salary for all employees

When user select 5. Program will require user input the start and end date to calculate salary with format (DD/MM). If user input incorrect format, program will return. Otherwise if user input correct format program will convert string from buffer to time type and check to see how many days are between start and end. If quantity days greater than 30 days, program will ask user wants to continue. If user press ‘y’ or ‘Y’ program will calculate and print total salary of each employee. If user press any key, program will return. In case it less than 30 days than program calculate and print total salary of each employee.

### Normal Mode

In Normal mode has 3 options: Check in, check out and back to main

#### Check in

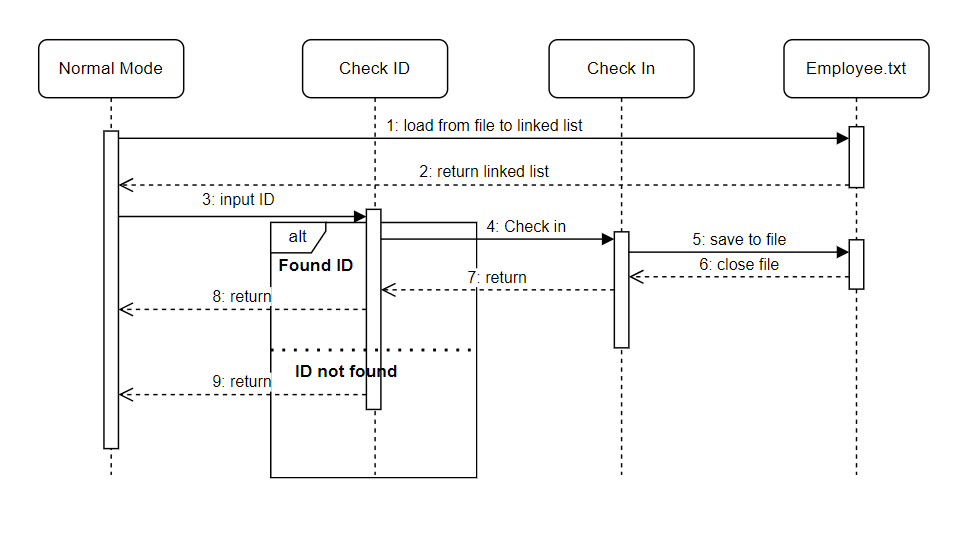


Figure 8: Check In sequence diagram

In check in mode, program require user input ID. After input ID program will verify that your information correct. If your ID has not been created, program will inform user that it can not find ID. If your ID correct user need press ‘y’ to check in and program return Normal Mode. If program check that user working time have checked out that day, it will not let user check in again and inform user need to contact Admin.

#### Check Out

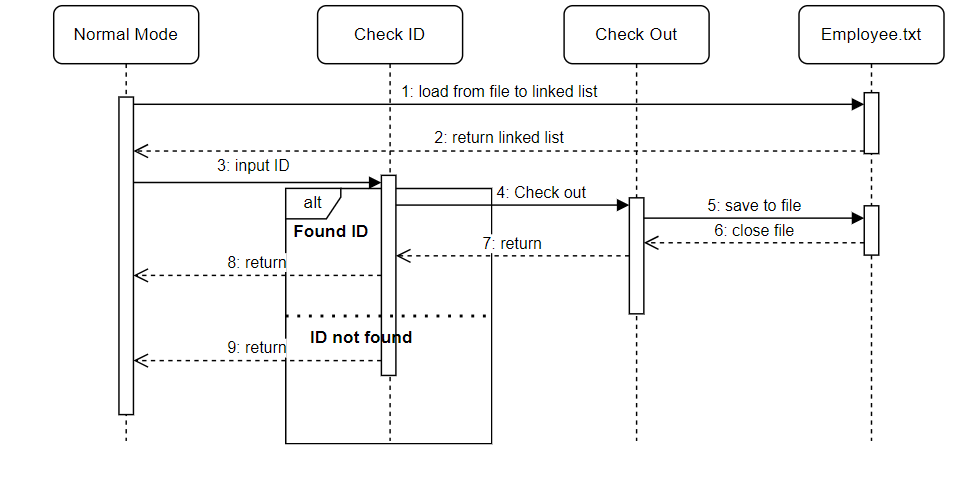


Figure 9: Check Out sequence diagram

In check out mode, program require user input ID. After input ID program will verify that your information correct. If your ID has not been created, program will inform user that it can not find ID. If your ID correct user need press ‘y’ to check out and program return Normal Mode. If the program checks that user have not check in that day, program inform that user forgot check in.

Structure diagram

### Load data from file to build linked list

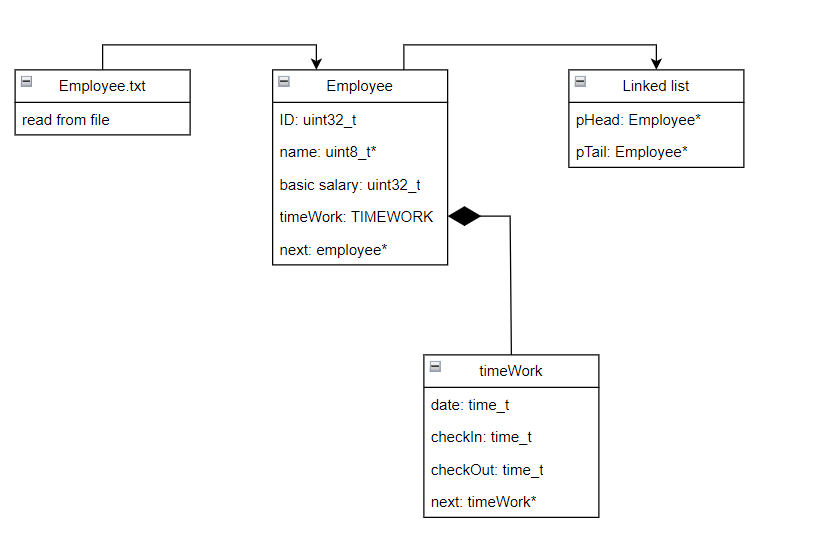


Figure 10: Load data from file Structure diagram

Information of employees include: ID, Name, basic salary, time working will loaded from file to each node in linked list to processing according to different requirements

### Save to file to store data

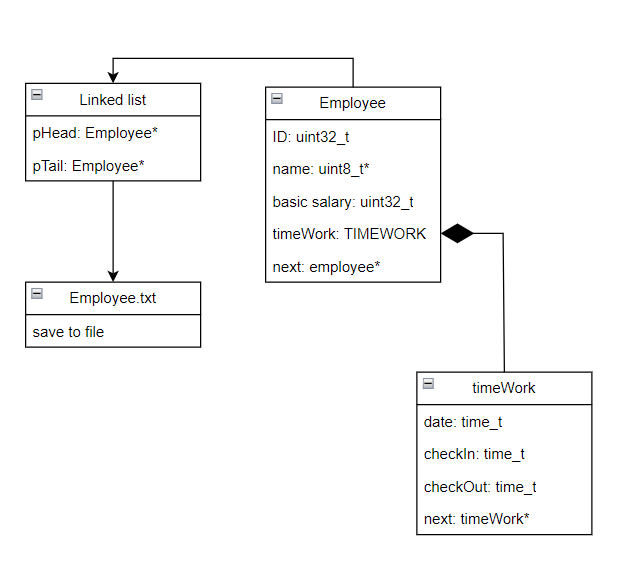


Figure : Save data to file structure diagram

All data will be saved to file. When necessary, these data can be used to process according to different requirements.

# Implementation

Git Repository : git clone https://dieppham97-admin@bitbucket.org/fa-emb-cf3/phamvandiep\_emb2404.git

VendorExt-Project Time Keeping Software

We enclose here the VendorExt-Project Time Keeping Software for what we collect from the SoR. The Data Model template and the dimclient data model will get this file as input its own data model.

Refer to Data Model file here:

…

Source code structure

### Source code file name of components:

**Common:**

main.h

main.c

**Feature 1:**

Admin-Mode.h

Admin-Mode.c

**Feature 2:**

Normal-Mode.h

Normal-Mode.c

### Directory structure

$TKS-MOCK/

├── Mock-Project

│   ├── inc

│      ├── main.h

│      ├── Admin-Mode.h

│      └── Normal-Mode.h

│   └── src

│   ├── main.c

│      ├── Admin-Mode.c

│      └── Normal-Mode.c

Common

### Constants

|  |  |  |
| --- | --- | --- |
| **Constant Name** | **Value** | **Description** |
| REQUEST\_WORKING\_HOURS\_IN\_MONTH | 184 | Maximum working hours in month |
| MAX\_NAME | 24 | Maximum length of name |

### Structure list

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Data Type** | **Description** |
| TIMEWORK: Structure to store the information working of staff | | |
| date | time\_t | Date |
| checkIn | time\_t | Check in time |
| checkOut | time\_t | Check out time |
| next | struct timeWork\* | Pointer to next struct |
| Related constants: No | | |
| *EMPLOYEE*: Structure to store information of staff. | | |
| ID | uint32\_t | ID of staff |
| sName | uint8\_t | Name of staff |
| basicSalary | uint32\_t | Basic salary of staff |
| timeWork | struct timeWork\* | Store information working of that staff |
| next | struct employ\* | Pointer to next struct |
| Related constants: No | | |
| *EMPLOYEE\_LIST:* | | |
| pHead | EMPLOYEE\* | Pointer to first node |
| pTail | EMPLOYEE\* | Pointer to last node |
| Related constants: No | | |

### Function list

|  |  |  |
| --- | --- | --- |
| Void checkPass() | | |
|  |  |  |
| Description:   * Check password before go to admin mode * Return data type: No | | |
| **Flowchart:** | | |

|  |  |  |
| --- | --- | --- |
| Void checkFileEmployee() | | |
|  |  |  |
| Description:   * Check if the employee.txt file exists * If not, create a new file * Return data type: No | | |
| **Flowchart:** | | |

|  |  |  |
| --- | --- | --- |
| Void createListEmploy | | |
| employeeList | EMPLOYEE\_LIST \* | Manage linked list |
| Description:   * Create and manage linked list with 2 pointers: pHead and pTail * Return data type: no | | |
| **Flowchart:** | | |

|  |  |  |
| --- | --- | --- |
| uint32\_t checkVaidNumber | | |
| sChoice | uint8\_t\* | String in entered from keyboard |
| Description:   * Check to see if the each character in string from keyboard is a number. * Return data type: uint32\_t | | |
| **Flowchart:** | | |

|  |  |  |
| --- | --- | --- |
| TIMEWORK\* addTimeWork | | |
| date | time\_t | Store the date |
| checkIn | time\_t | Store check-in time for each staff |
| checkOut | time\_t | Store check-out time for each staff |
| Description:   * Create a new node to sotre each employee’s working time * Return data type: pointer time work | | |
| **Flowchart:** | | |

|  |  |  |
| --- | --- | --- |
| EMPLOYEE\* addNewNodeEmploy | | |
| ID | uint32\_t | Store the ID |
| sName | Uint8\_t | Store the name |
| basicSalary | uint32\_t | Store basic salary |
| Description:   * Create a new node to sotre information each employee * Return data type: pointer employee | | |
| **Flowchart:** | | |

|  |  |  |
| --- | --- | --- |
| void addNodeHead | | |
| list | EMPLOYEE\_LIST\* | Use linked list to manage |
| newNodeEmployee | EMPLOYEE\* | Add the new node employee to the linked list |
| Description:   * After create new node employee then add to the top of the linked list * Return data type: no | | |
| **Flowchart:** | | |

|  |  |  |
| --- | --- | --- |
| void addNodeTail | | |
| list | EMPLOYEE\_LIST\* | Use linked list to manage |
| newNodeEmployee | EMPLOYEE\* | Add the new node employee to the linked list |
| Description:   * After create new node employee then add to the end of the linked list * Return data type: no | | |
| **Flowchart:** | | |

|  |  |  |
| --- | --- | --- |
| void saveEmloyeeToFile | | |
| list | EMPLOYEE\_LIST\* | Use linked list to manage linked list |
| Description:   * Save all information of employees managed by linked list to Employee.txt file * Return data type: no | | |
| **Flowchart:** | | |

|  |  |  |
| --- | --- | --- |
| time\_t parseDate | | |
| dateStr | uint8\_t\* | String with date format |
| tm | struct tm \* | Structure of time |
| Description:   * Convert string to date with format: DD-MM-YYYY * Return data type: time\_h | | |
| **Flowchart:** | | |

|  |  |  |
| --- | --- | --- |
| time\_t parseTime | | |
| timeStr | uint8\_t\* | String with time format |
| tm | struct tm \* | Structure of time |
| Description:   * Convert string to time with format: HH:MM * Return data type: time\_h | | |
| **Flowchart:** | | |

|  |  |  |
| --- | --- | --- |
| EMPLOYEE\_LIST\* loadEmployeeFromFile | | |
| list | EMPLOYEE\_LIST\* | Pass linked list |
| Description:   * Load file Employee.txt and convert all data to linked list. Data of employee include ID, name and basic salary stored in node employee and time working stored in pointer of struct employee. * Return data type: linked list | | |
| **Flowchart:** | | |

|  |  |  |
| --- | --- | --- |
| void checkFileEmployee | | |
|  |  |  |
| Description:   * Check if the employee file exists. If not, create a new file * Return data type: No | | |
| **Flowchart:** | | |

|  |  |  |
| --- | --- | --- |
| void freeMemory | | |
| list | EMPLOYEE\_LIST\* | Pass linked list |
| Description:   * Delete all nodes to free memory in heap segment. * Return data type: No | | |
| **Flowchart:** | | |

## Features Detail

Program has 2 main features:

* Admin mode is used by people with specific tasks.
* Normal mode is used by employee/staff to check in and check out every day.

### Admin Mode

#### Notes

Feature 1’ main functionality:

* Functionality 1: Change password.
* Functionality 2: Add information of new employee/staff.
* Functionality 3: View information of all employees (Employee ID, full name, basic salary).
* Functionality 4: View working time of a specified employee.
* Functionality 5: Calculate salary for all employee.
* Functionality 6: Back to main.

#### Function list

|  |  |  |
| --- | --- | --- |
| void adminMode | | |
|  |  |  |
| **Description:**   * Function is used select feature the user wants. * Returned data type: No | | |
| **Flowchart:** | | |

|  |  |  |
| --- | --- | --- |
| void changePass | | |
|  |  |  |
| **Description:**   * Function is used change password. New password has not contain space, user have to re-input. * Returned data type: No | | |
| **Flowchart:** | | |

|  |  |  |
| --- | --- | --- |
| uint8\_t\* encrypPassword | | |
| password | uint8\_t\* | String of password |
| **Description:**   * Function is used encryp password * Returned data type: string password after encryp | | |
| **Flowchart:** | | |

|  |  |  |
| --- | --- | --- |
| uint8\_t addNewEmployInFileAdmin | | |
| employ | EMPLOYEE\* | Infoformation of new employee |
| **Description:**   * Outline:   Initialize information of new employee in Admin.txt file   * Returned data type: uint8\_t.   The function returns whether the information initiation is successful or not. | | |
| **Flowchart:** | | |

|  |  |  |
| --- | --- | --- |
| void printAdmin | | |
|  |  |  |
| **Description:**   * Outline:   Information list of all employee.   * Returned data type: No | | |
| **Flowchart:** | | |

|  |  |  |
| --- | --- | --- |
| uint8\_t findID | | |
| list | EMPLOYEE\_LIST\* | Linked list of employee |
| inID | uint32\_t | ID want to check |
| **Description:**   * Outline:   Information and working time of employee want to check by ID   * Returned data type: uint8\_t.   The function returns whether the information check is successful or not. | | |
| **Flowchart:** | | |

|  |  |  |
| --- | --- | --- |
| uint8\_t checkValidDate | | |
| date | uint8\_t\* | String date want to check |
| **Description:**   * Outline:   The result of the string entered from the keyboard is valid for format DD/MM.   * Returned data type: uint8\_t.   The function returns whether the string is valid or not. | | |
| **Flowchart:** | | |

|  |  |  |
| --- | --- | --- |
| uint8\_t checkStartEndDate | | |
| inStart | uint8\_t\* | String start date is entered from the keyboard |
| inEnd | uint8\_t\* | String end date is entered from the keyboard |
| **Description:**   * Outline:   Input the start and end date after that check if it is valid.   * Returned data type: uint8\_t.   The function returns whether the string is valid or not. And return string of start and end date. | | |
| **Flowchart:** | | |

|  |  |  |
| --- | --- | --- |
| void calculateSalaries | | |
| list | EMPLOYEE\_LIST\* | Load information from linked list |
| requestedWorkingHours | double | Minimum number of working hours in 1 month |
| startDate | time\_t | The start date calculate salary |
| endDate | time\_t | The end date calculate salary |
| **Description:**   * Outline:   This function have feature calculate the salary of all employee and display the amounts deducted if violated: go to work late and leave early, a half day-off.  Saturday, Sunday and overtime work will not be paid.   * Returned data type: no. | | |
| **Flowchart:** | | |

### Normal mode

#### Notes

Normal mode have 3 feature:

* Functionality 1: Check in
* Functionality 2: Check out
* Functionality 3: Back to main.

#### Function list

|  |  |  |
| --- | --- | --- |
| void normalMode | | |
|  |  |  |
| **Description:**   * Outline:   Mode for employees to check in and check out   * Returned data type: no | | |
| **Flowchart:** | | |

|  |  |  |
| --- | --- | --- |
| void normalMode | | |
|  |  |  |
| **Description:**   * Outline:   Mode for employees to check in and check out   * Returned data type: no | | |
| **Flowchart:** | | |

|  |  |  |
| --- | --- | --- |
| void checkInByID | | |
| list | EMPLOYEE\_LIST\* | Linked list |
| inID | uint32\_t | ID of employee |
| **Description:**   * Outline:   Mode for employees to check in   * Returned data type: no | | |
| **Flowchart:** | | |

|  |  |  |
| --- | --- | --- |
| void checkInByID | | |
| list | EMPLOYEE\_LIST\* | Linked list |
| inID | uint32\_t | ID of employee |
| **Description:**   * Outline:   Mode for employees to checkout   * Returned data type: no | | |
| **Flowchart:** | | |