**Low Level Design Document**

**for**

**Employee Management System (EMS)**

Version 1.0

**Table of Contents**

Table of Contents……………………….……………………….……………………….……

1. Introduction……………………….……………………….……………………….……….

1.1 Purpose……………………….……………………….……………………….…..

1.2 Reading Suggestions……………………….……………………….……………..

2. Detailed System Design……………………….……………………….…………………...

2.1 EDLL Diagram……………………….……………………….……………………

2.2 EMS Diagram……………………….……………………….……………………

2.3 Class Description……………………….……………………….………………...

2.4 Relationship between Classes……………………….……………………….……

**1. Introduction**

**1.1 Purpose**

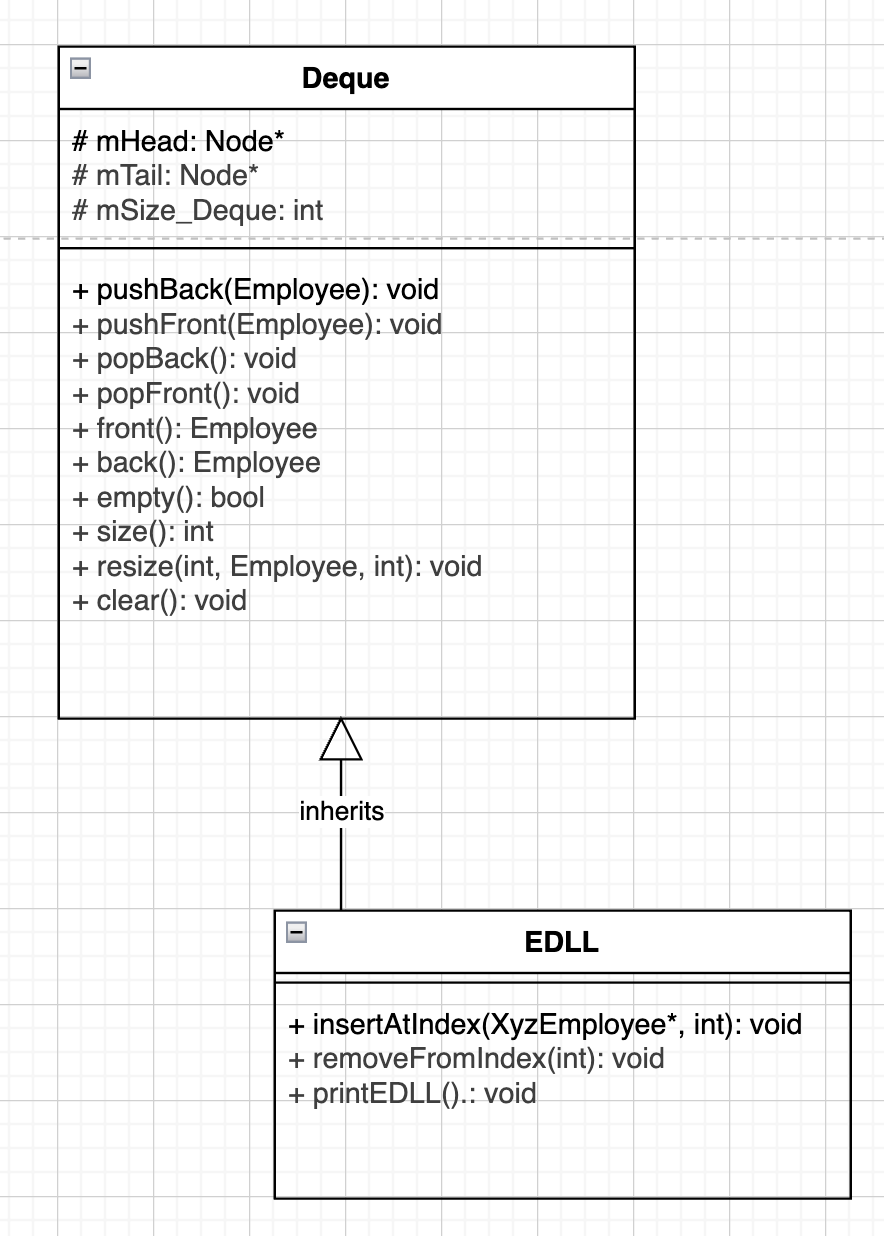
Employee Management System or EMS is created for management of employees of XYZ corporation. This EMS will provide a simplified menu-based interface for employee management operations including adding, removing or modifying employee details.

**1.2 Reading Suggestions**

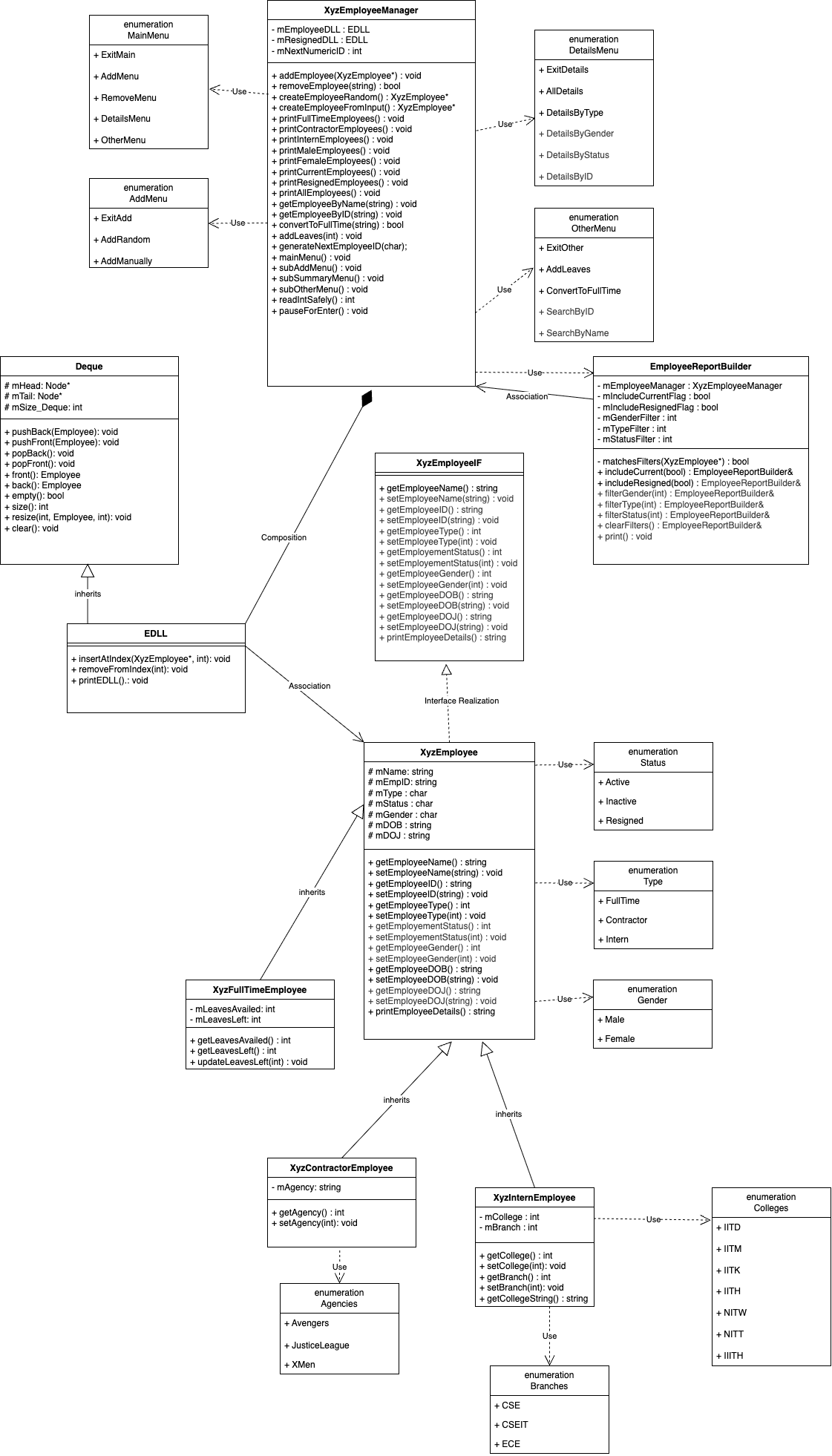
Document is primarily intended for Review members for C++ Assignment 2.

**2. Detailed System Design**

**2.1 EDLL Diagram**

****

**2.2 EMS Diagram**

****

**2.3 Class Description**

**2.3.1 XyzEmployeeManager**

The management class of Employee Management System (EMS) that is responsible for adding, removing or modifying employee details according to user input from the main menu. XyzEmployeeManager maintains Employee Deque list of Active/Inactive and Resigned employees and performs operations over them.

|  |  |
| --- | --- |
| **Component** | **Description** |
| mEmployeeDLL | Deque for Storing Active/Inactive Employees of all types – Full Time, Contractors and Interns |
| mResignedDLL | Deque for storing resigned employees of all types |
| mNextNumericID | Store value for incremental employee ID |
| addEmployee(XyzEmployee\*) | Method to add Employee to EDLLs |
| removeEmployee(string) | Method to add Employee to mResignedDLL |
| createEmployeeRandom() | Create a random employee |
| createEmployeeFromInput() | Create an employee based on user input |
| printFullTimeEmployees() | Prints all Full time employees’ details |
| printContractorEmployees() | Prints all Contractor employees’ details |
| printInternEmployees() | Prints all Intern employees’ details |
| printCurrentEmployees() | Prints all current (active/inactive) employees’ details |
| printResignedEmployees() | Prints all Resigned employees’ details |
| printMaleEmployees() | Prints all male employee details |
| printFemaleEmployees() | Prints all female employee details |
| printAllEmployees() | Prints all employees’ details |
| getEmployeeByName(string) | Prints employee’s details by name |
| getEmployeeByID(string) | Prints employee’s details by employee ID |
| convertToFullTime(string) | Converts Contractor or Interns to Full Time Employee |
| addLeaves(int) | Update leaves for all full time employees |
| generateNextEmployeeID(char) | Generate next Employee ID |
| mainMenu() | EMS Main Menu |
| subAddMenu() | EMS Add Sub Menu |
| subSummaryMenu() | EMS Summary sub menu |
| subOtherMenu() | EMS Other sub menu |
| readIntSafely() | Read function for integer inputs |
| pauseForEnter() | Enter key buffer |
| getCurrentSize() | Returns size of mEmployeeDLL |
| getResignedSize() | Returns size of mResignedDLL |
| getCurrentEmployeeAt(int) | Returns XyzEmployee\* at index in mEmployeeDLL |
| getResignedEmployeeAt(int) | Returns XyzEmployee\* at index in mResignedDLL |

**2.3.2 XyzEmployeeIF**

Employee Interface class which will be used to realize EmployeeBase class for creating and maintaining employee objects.

|  |  |
| --- | --- |
| **Component** | **Description** |
| getEmployeeName() | Interface functions |
| setEmployeeName(string) |
| getEmployeeID() |
| setEmployeeID(string) |
| getEmployeeType() |
| setEmployeeType(int) |
| getEmployementStatus() |
| getEmployementStatus(int) |
| getEmployeeGender() |
| setEmployeeGender(int) |
| getEmployeeDOB() |
| setEmployeeDOB(string) |
| getEmployeeDOJ() |
| setEmployeeDOJ(string) |
| printEmployeeDetails(string) |

**2.3.3 XyzEmployee**

Employee Base class which has all employee details’ instances.

|  |  |
| --- | --- |
| **Component** | **Description** |
| mName | Instance to store Employee’s Name |
| mEmpID | Instance to store Employee’s ID |
| mDOB | Instance to store Employee’s Date of Birth |
| mDOJ | Instance to store Employee’s Date of Joining |
| mType | Instance to store Employee’s Type – Full Time, Contractor, Intern |
| mStatus | Instance to store Employee’s Status – Active/Inactive/Resigned |
| mGender | Instance to store Employee’s Name |
| getEmployeeName() | Returns Employee’s Name |
| setEmployeeName(string) | Sets Employee’s Name |
| getEmployeeID() | Returns Employee’s ID |
| setEmployeeID(string) | Sets Employee’s ID |
| getEmployeeType() | Returns Employee’s Type |
| setEmployeeType(int) | Sets Employee’s Type |
| getEmployementStatus() | Returns Employee’s Status |
| setEmployementStatus(int) | Sets Employee’s Status |
| getEmployeeGender() | Returns Employee’s Gender |
| setEmployeeGender(int) | Sets Employee’s Gender |
| getEmployeeDOB() | Returns Employee’s Date of Birth |
| setEmployeeDOB(string) | Sets Employee’s Date of Birth |
| getEmployeeDOJ() | Returns Employee’s Date of Joining |
| setEmployeeDOJ(string) | Sets Employee’s Date of Joining |
| printEmployeeDetails(string) | Print Employee details |

**2.3.4 XyzFullTimeEmployee**

Full Time Employee class having XyzEmployee class instances and methods along with extended capabilities including leaves details and related methods.

|  |  |
| --- | --- |
| **Component** | **Description** |
| mLeavesAvailed | Stores leaves availed by an employee |
| mLeavesLeft | Stores leaves left with an employee |
| getLeavesAvailed() | Returns leaves availed |
| getLeavesLeft() | Returns leaves left |
| updateLeavesLeft() | Updates by adding leaves to ‘leaves left’ with an employee |

**2.3.5 XyzContractorEmployee**

Contractor Employee class having XyzEmployee class instances and methods along with extended capabilities including agency details and related methods.

|  |  |
| --- | --- |
| **Component** | **Description** |
| mAgency | Stores Agency of the Contractor |
| getAgency() | Returns Agency of the Contractor |
| setAgency(int) | Sets Agency for the Contractor |

**2.3.6 XyzInternEmployee**

Intern Employee class having XyzEmployee class instances and methods along with extended capabilities including college and branch details and related methods.

|  |  |
| --- | --- |
| **Component** | **Description** |
| mCollege | Stores college name of intern |
| mBranch | Stores branch of Intern |
| getCollege() | Returns college of Intern |
| setCollege(int) | Sets college of Intern |
| getBranch() | Returns branch of intern |
| setBranch(int) | Sets branch of intern |

**2.3.7 Deque**

Standard Deque template class implementation which will be used as a base class to implement EDLL class to store employee data nodes.

|  |  |
| --- | --- |
| **Component** | **Description** |
| mHead | Pointer to head element of Deque |
| mTail | Pointer to tail element of Deque |
| mSize\_Deque | Stores size of Deque |
| pushBack(S) | Pushes Employee object to back of deque |
| pushFront(S) | Pushes Employee object to front of deque |
| popBack() | Pops from back of deque |
| popFront() | Pops from front of deque |
| front() | Prints first element of deque |
| back() | Prints last element of deque |
| empty() | Returns true if deque is empty |
| clear() | Clears deque and sets its head and tail pointers to null |
| size() | Returns size of deque |
| resize(int, S, int) | Resizes deque based on direction and new size. |

**2.3.8 EDLL**

Extended Doubly Linked List Class is used to create a Deque structure with extended capabilities which will be used by XyzEmployeeManager to create Deque to store employee data nodes.

|  |  |
| --- | --- |
| **Component** | **Description** |
| insertAtIndex(XyzEmployee\*, int) | Inserts a new employee at index n |
| removeFromIndex(int) | Removes the employee from deque at index n |

**2.3.9 EmployeeReportBuilder**

Builder Design Class for using in Summary menu to filter data based on input

|  |  |
| --- | --- |
| **Component** | **Description** |
| mEmployeeManager | XyzEmployeeManager instance (reference) |
| mIncludeCurrentFlag | Flag to filter Current employees |
| mIncludeResignedFlag | Flag to filter Resigned employees |
| mGenderFilter | Flag to filter Male/Female employees |
| mTypeFilter | Flag to filter Full-Time/Contractor/Intern employees |
| mStatusFilter | Flag to filter Active/Inactive/Resigned employees |
| includeCurrent(bool) | Check current flag |
| includeResigned(bool) | Check resigned flag |
| filterGender(int) | Check gender flag |
| filterType(int) | Check type flag |
| filterStatus(int) | Check status flag |
| clearFilters() | Clear all filters to default |
| print() | Print based on filters |

**2.4 Relationship between Classes**

**2.4.1 Interface Realization**

* XyzEmployee class realizes XyzEmployeeIF interface to create a base class to create employee objects and hold its details.

**2.4.2 Inheritance**

* XyzEmployee base class is inherited into XyzFullTimeEmployee, XyzContractorEmployee, and XyzInternEmployee.
* EDLL class is inherited from Deque Template class for XyzEmployee\* type.

**2.4.3 Composition**

* There is a composition relationship from EDLL to XyzEmployeeManager as XyzEmployeeManager has 2 class instances of EDLL class and EDLL class can only exist inside XyzEmployeeManager class.

**2.4.4 Association**

* There is an association relation between EDLL and XyzEmployee class as an EDLL class is associated with one or more XyzEmployee class objects.
* There is association relation between EmployeeReportBuilder and XyzEmployeeManager as there is an instance of EmployeeReportBuilder in XyzEmployeeManager.

**2.4.5 Dependency**

* XyzEmployeeManager uses EmployeeReportBuilder for one function.
* XyzEmployeeManager class depends on MainMenu, AddMenu, DetailsMenu, and OtherMenu enums.
* XyzEmployee base class depends upon Status, Type and Gender enums.
* XyzContractorEmployee class depends upon or uses Agencies enum.
* XyzInternEmployee class depends upon or uses Colleges and Branches enums.