Criterion B

Electronic Designs.......2

System Storyboard 3

UML Class Interaction Diagram 4

System Flow Charts 5

Basic Startup and Login Processes : 5

Algorithmic Flow Charts 6

Search An Employee 6

Delete Employee 6

.......6

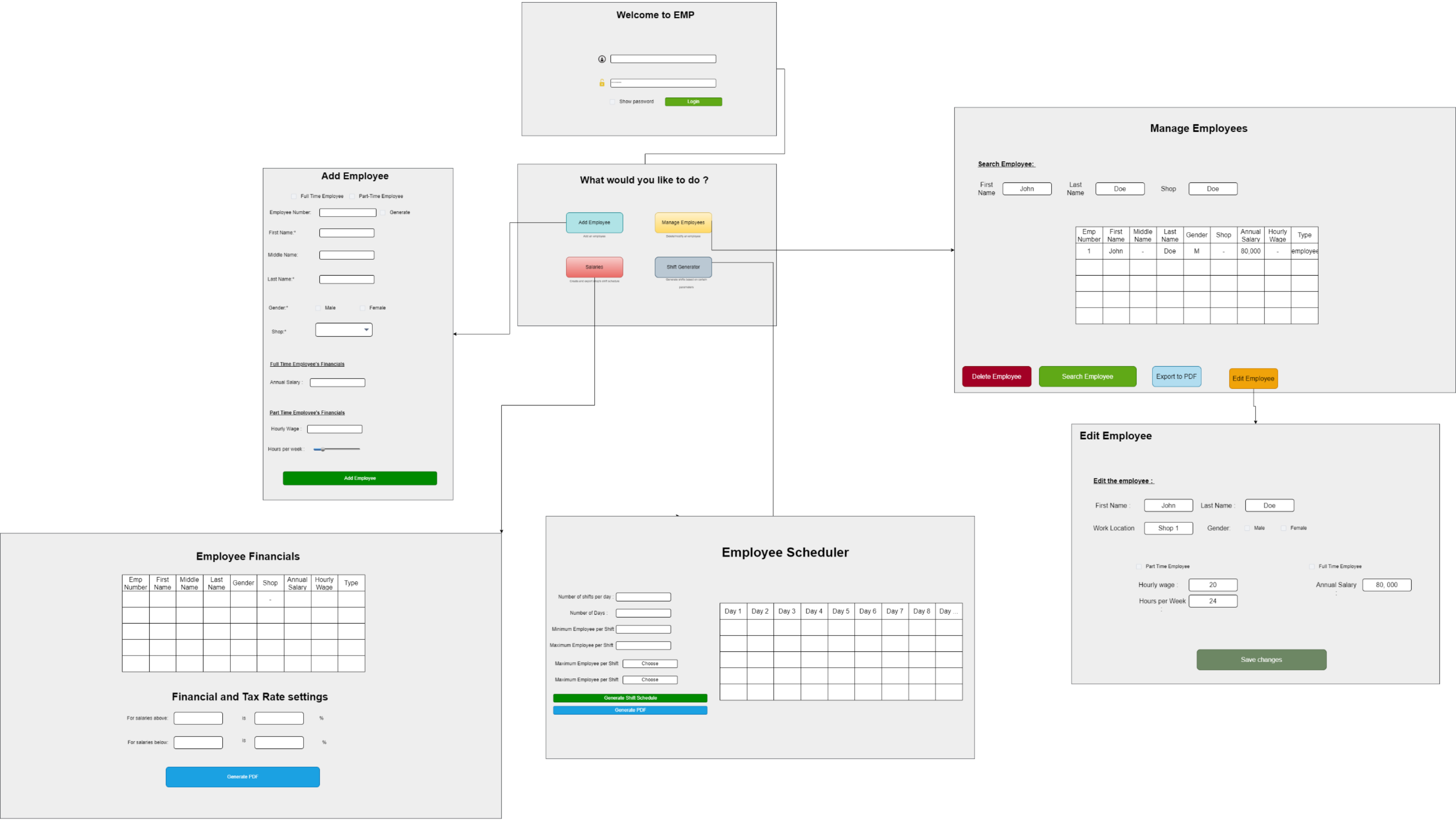
Get an Employee At a Specific Index 7

Generate Shift 8

Test Plan : 10

Bibliography 11

Electronic Designs



What would you like to do ?
Add Employee
Add an employee
Manage Employees
Delete/modify an employee
Salaries
Create and export shop's shift schedule
Shift Generator
Generate shifts based on certain parameters

Manage Employees

Search Employee:

First Name: John   Last Name: Doe   Shop: Doe

Emp Number	First Name	Middle Name	Last Name	Gender	Shop	Annual Salary	Hourly Wage	Type
1	John	-	Doe	M	-	80,000	-	employee

Edit Employee

Edit the employee :

First Name : John   Last Name : Doe

Work Location: Shop 1   Gender: □ Male □ Female

□ Part Time Employee

Hourly wage : 20

Hours per Week : 24

□ Full Time Employee

Annual Salary : 80, 000

Employee Financials

Emp Number	First Name	Middle Name	Last Name	Gender	Shop	Annual Salary	Hourly Wage	Type
					-			

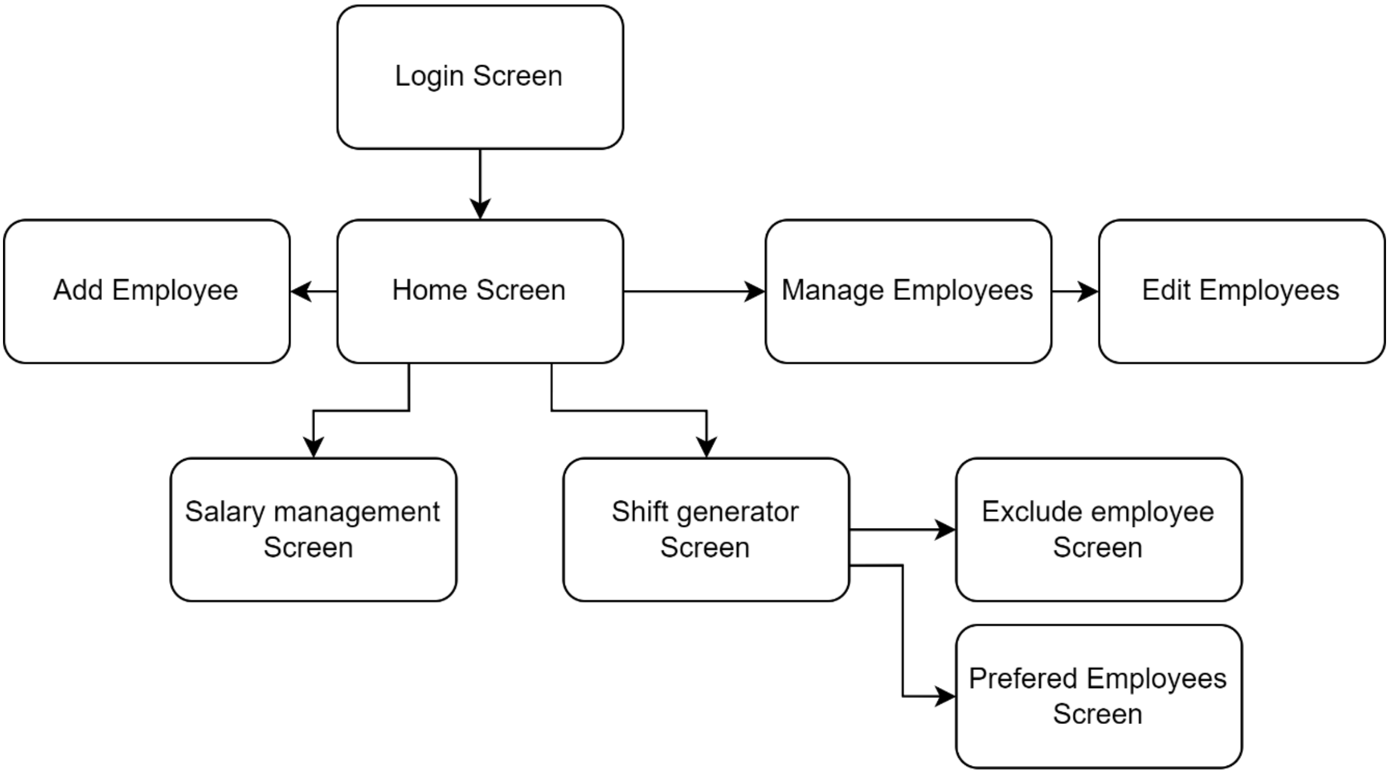
Financial and Tax Rate settings

Employee Scheduler

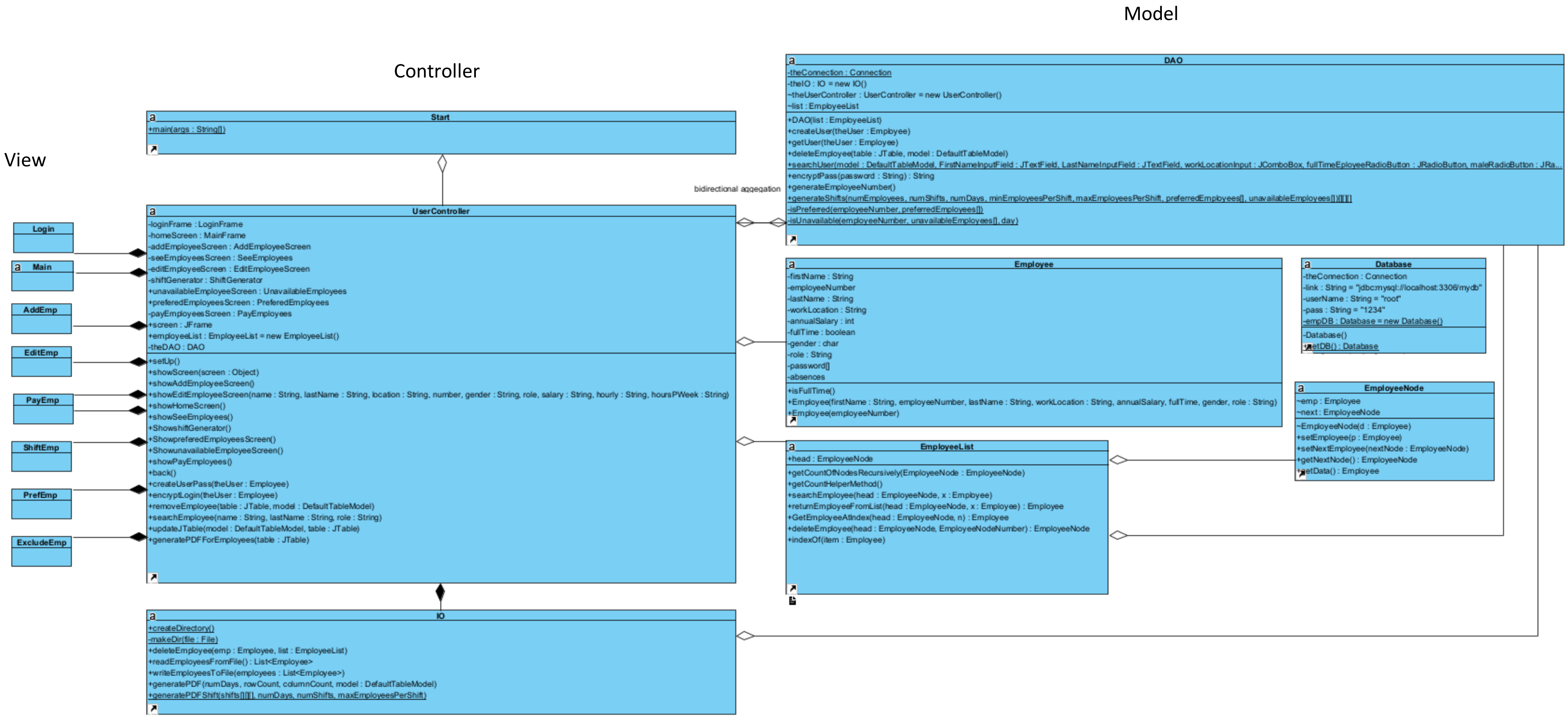
Number of shifts per day :
Number of Days :
Minimum Employee per Shift
Maximum Employee per Shift
Maximum Employee per Shift Choose
Maximum Employee per Shift Choose
Generate Shift Schedule
Generate PDF

Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7	Day 8	Day ...

System Storyboard



UML Class Interaction Diagram[1]

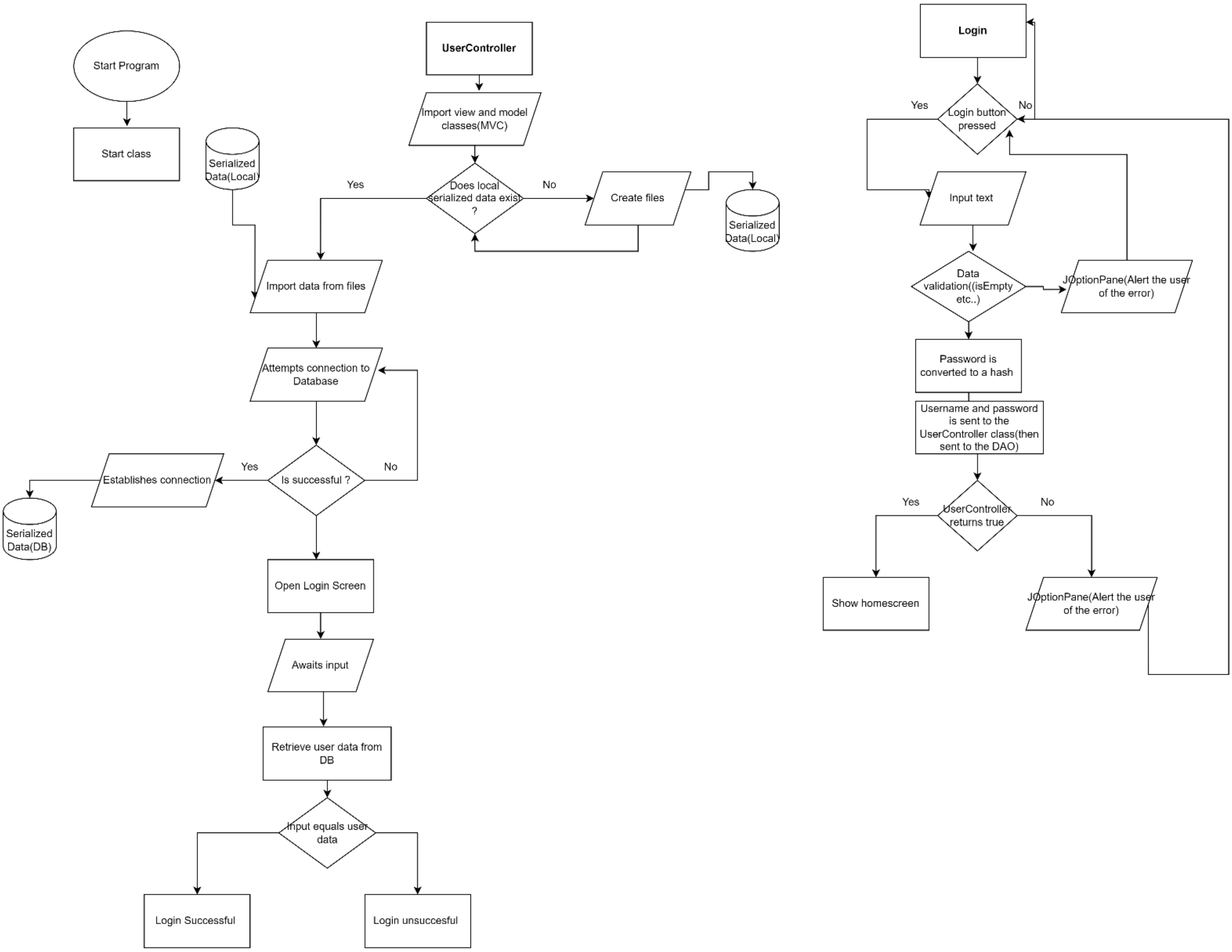

Model
Controller
View
DAO
-theConnection : Connection
-theIO : IO = new IO()
~theUserController : UserController = new UserController()
~list : EmployeeList
+DAO(list : EmployeeList)
+createUser(theUser : Employee)
+getUser(theUser : Employee)
+deleteEmployee(table : JTable, model : DefaultTableModel)
+searchUser(model : DefaultTableModel, FirstNameInputField : JTextField, LastNameInputField : JTextField, workLocationInput : JComboBox, fullTimeEployeeRadioButton : JRadioButton, maleRadioButton : JRa...
+encryptPass(password : String) : String
+generateEmployeeNumber()
+generateShifts(numEmployees, numShifts, numDays, minEmployeesPerShift, maxEmployeesPerShift, preferredEmployees[], unavailableEmployees[])[][][]
-isPreferred(employeeNumber, preferredEmployees[])
-isUnavailable(employeeNumber, unavailableEmployees[], day)
Start
+main(args : String[])
bidirectional aggegation
UserController
-loginFrame : LoginFrame
-homeScreen : MainFrame
-addEmployeeScreen : AddEmployeeScreen
-seeEmployeesScreen : SeeEmployees
-editEmployeeScreen : EditEmployeeScreen
-shiftGenerator : ShiftGenerator
+unavailableEmployeeScreen : UnavailableEmployees
+preferedEmployeesScreen : PreferedEmployees
-payEmployeesScreen : PayEmployees
+screen : JFrame
+employeeList : EmployeeList = new EmployeeList()
-theDAO : DAO
+setUp()
+showScreen(screen : Object)
+showAddEmployeeScreen()
+showEditEmployeeScreen(name : String, lastName : String, location : String, number, gender : String, role, salary : String, hourly : String, hoursPWeek : String)
+showHomeScreen()
+showSeeEmployees()
+ShowshiftGenerator()
+ShowpreferedEmployeesScreen()
+ShowunavailableEmployeeScreen()
+showPayEmployees()
+back()
+createUserPass(theUser : Employee)
+encryptLogin(theUser : Employee)
+removeEmployee(table : JTable, model : DefaultTableModel)
+searchEmployee(name : String, lastName : String, role : String)
+updateJTable(model : DefaultTableModel, table : JTable)
+generatePDFForEmployees(table : JTable)
Login
Main
AddEmp
EditEmp
PayEmp
ShiftEmp
PrefEmp
ExcludeEmp
Employee
-firstName : String
-employeeNumber
-lastName : String
-workLocation : String
-annualSalary : int
-fullTime : boolean
-gender : char
-role : String
-password[]
-absences
+isFullTime()
+Employee(firstName : String, employeeNumber, lastName : String, workLocation : String, annualSalary, fullTime, gender, role : String)
+Employee(employeeNumber)
Database
-theConnection : Connection
-link : String = "jdbc:mysql://localhost:3306/mydb"
-userName : String = "root"
-pass : String = "1234"
-empDB : Database = new Database()
-Database()
+getDB() : Database
EmployeeNode
~emp : Employee
~next : EmployeeNode
~EmployeeNode(d : Employee)
+setEmployee(p : Employee)
+setNextEmployee(nextNode : EmployeeNode)
+getNextNode() : EmployeeNode
+getData() : Employee
EmployeeList
+head : EmployeeNode
+getCountOfNodesRecursively(EmployeeNode : EmployeeNode)
+getCountHelperMethod()
+searchEmployee(head : EmployeeNode, x : Employee)
+returnEmployeeFromList(head : EmployeeNode, x : Employee) : Employee
+GetEmployeeAtIndex(head : EmployeeNode, n) : Employee
+deleteEmployee(head : EmployeeNode, EmployeeNodeNumber) : EmployeeNode
+indexOf(item : Employee)
IO
+createDirectory()
-makeDir(file : File)
+deleteEmployee(emp : Employee, list : EmployeeList)
+readEmployeesFromFile() : List<Employee>
+writeEmployeesToFile(employees : List<Employee>)
+generatePDF(numDays, rowCount, columnCount, model : DefaultTableModel)
+generatePDFShift(shifts[][][], numDays, numShifts, maxEmployeesPerShift)

Note : The View Package displayed in the UML contains only placeholder classes. These classes are intended to represent the user interface components, but their implementation details have not yet been determined(JButton, JTable, JTextField etc…).

[1] (Visual-paradigm.com, 2019)

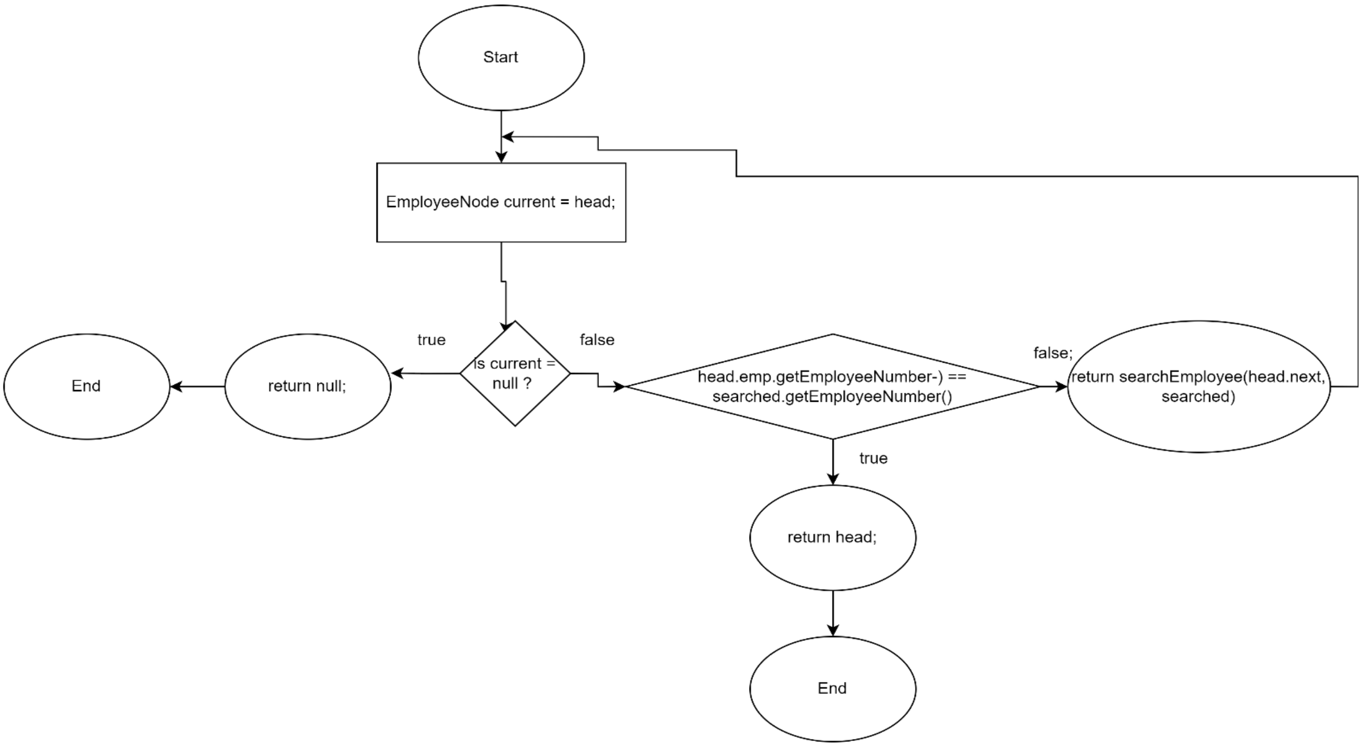
System Flow Charts

Basic Startup and Login Processes :


Start Program
Start class
Serialized Data(Local)
UserController
Import view and model classes(MVC)
Yes
Does local serialized data exist ?
No
Create files
Serialized Data(Local)
Import data from files
Attempts connection to Database
Yes
Is successful ?
No
Establishes connection
Serialized Data(DB)
Open Login Screen
Awaits input
Retrieve user data from DB
Input equals user data
Login Successful
Login unsuccesful
Login
Yes
Login button pressed
No
Input text
Data validation((isEmpty etc..)
JOptionPane(Alert the user of the error)
Password is converted to a hash
Username and password is sent to the UserController class(then sent to the DAO)
Yes
UserController returns true
No
Show homescreen
JOptionPane(Alert the user of the error)

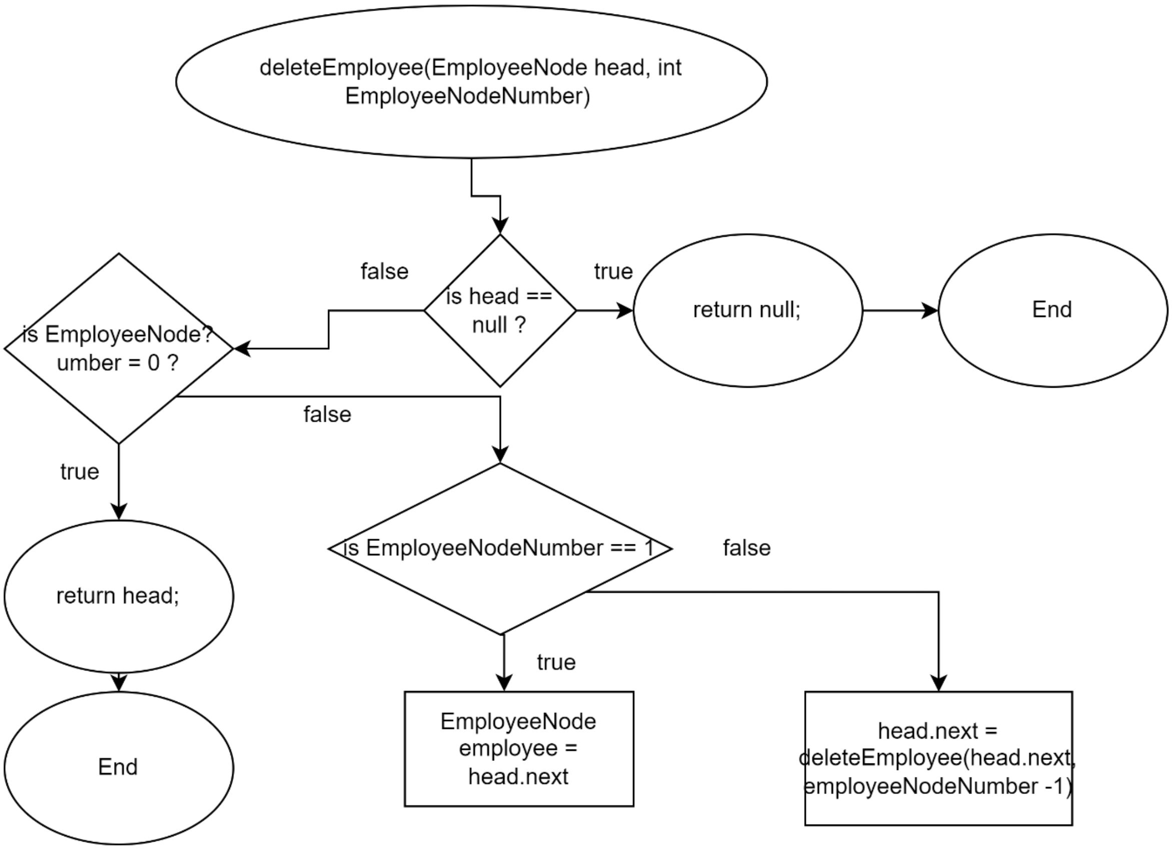
Algorithmic Flow Charts

Search An Employee



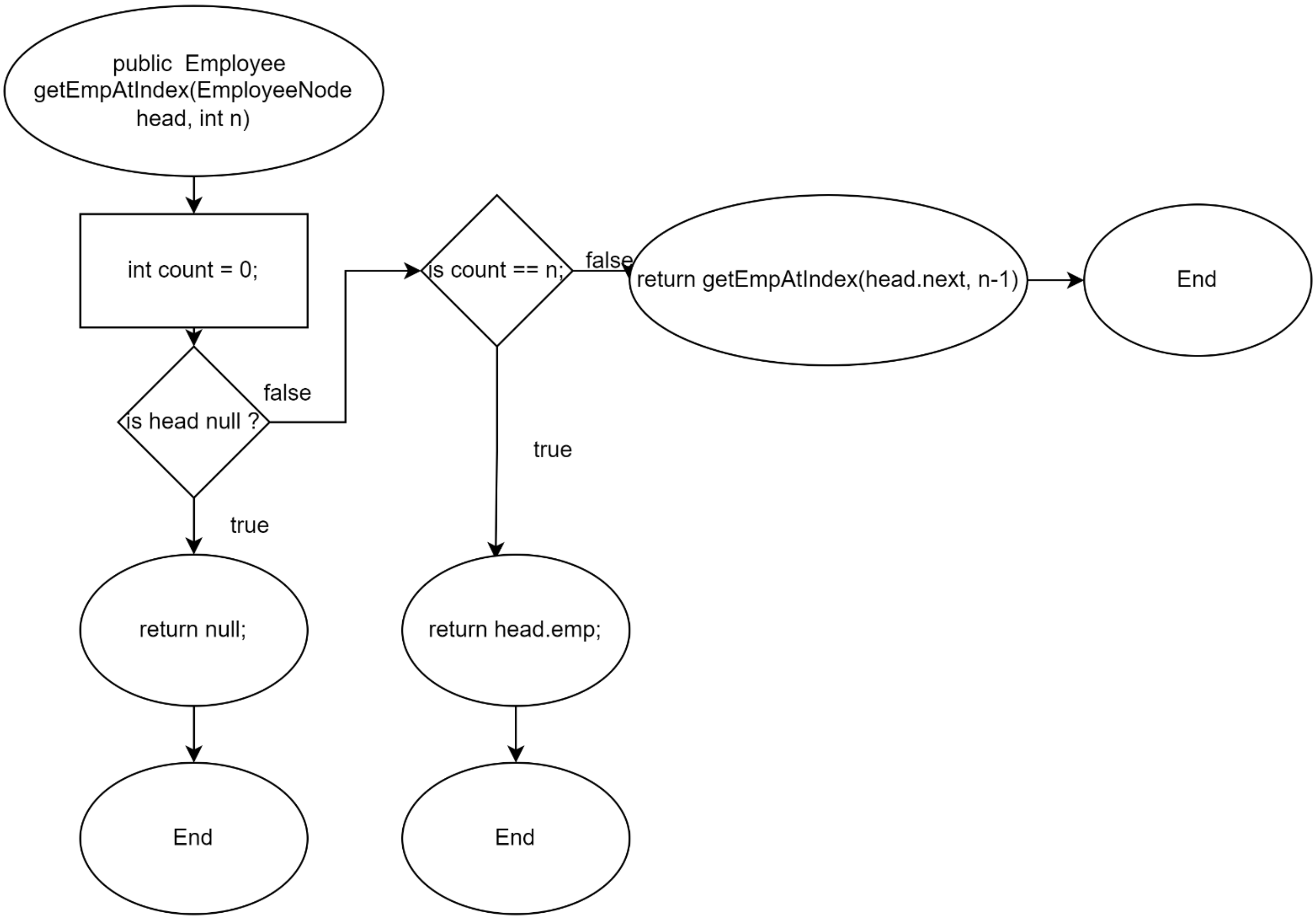
Start
EmployeeNode current = head;
true
false
is current == null ?
End
return null;
head.emp.getEmployeeNumber-) == searched.getEmployeeNumber()
false;
return searchEmployee(head.next, searched)
true
return head;
End

Delete Employee

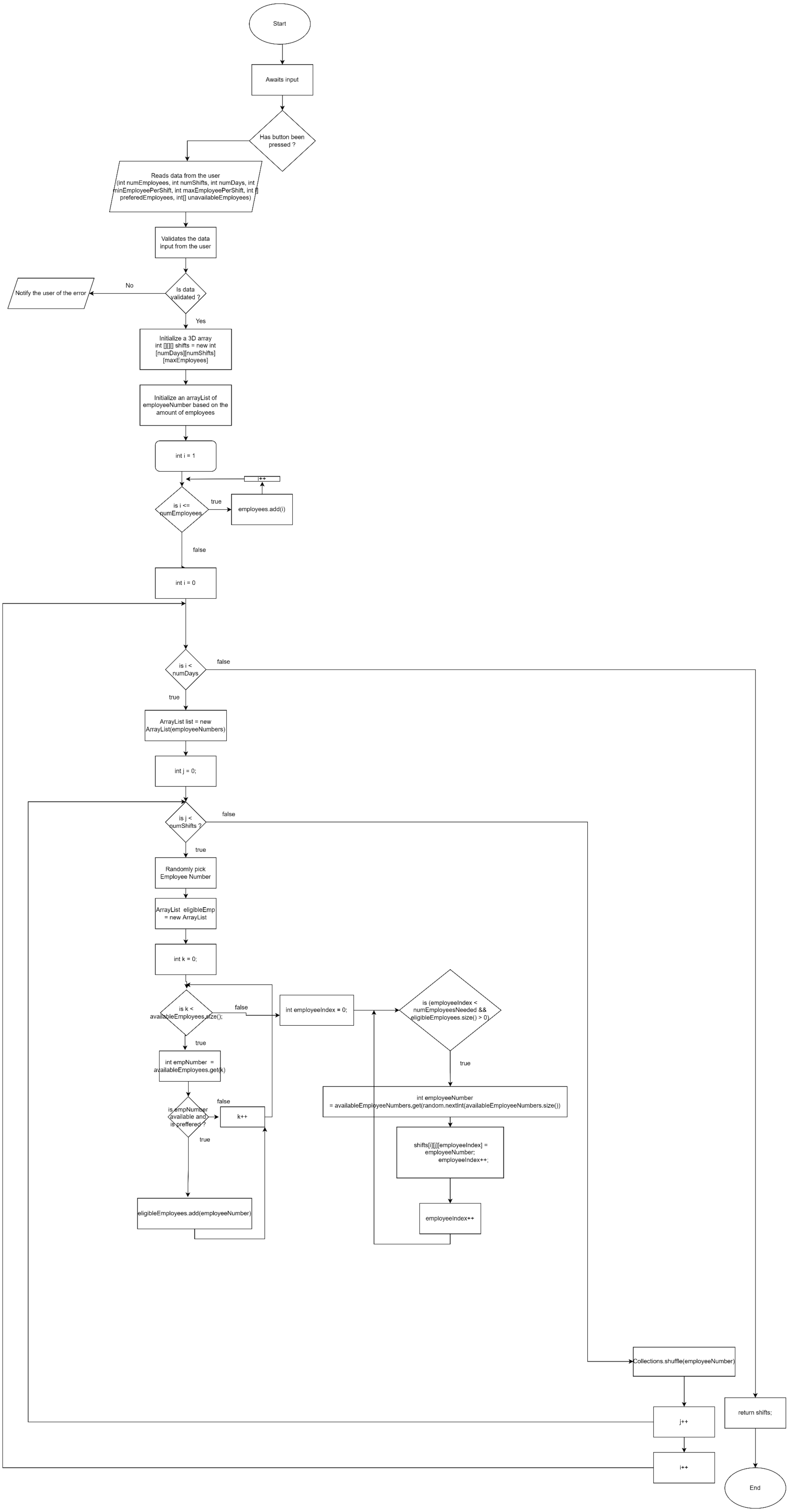


deleteEmployee(EmployeeNode head, int EmployeeNodeNumber)
false
is head == null ?
true
return null;
End
is EmployeeNode?umber = 0 ?
false
true
return head;
End
is EmployeeNodeNumber == 1
false
true
EmployeeNode employee = head.next
head.next = deleteEmployee(head.next, employeeNodeNumber -1)

Get an Employee At a Specific Index



Generate Shift


Start
Awaits input
Has button been pressed ?
Reads data from the user
(int numEmployees, int numShifts, int numDays, int minEmployeePerShift, int maxEmployeePerShift, int [] preferedEmployees, int[] unavailableEmployees)
Validates the data input from the user
Is data validated ?
No
Notify the user of the error
Yes
Initialize a 3D array int [][][] shifts = new int [numDays][numShifts][maxEmployees]
Initialize an arrayList of employeeNumber based on the amount of employees
int i = 1
i++
is i <= numEmployees
true
employees.add(i)
false
int i = 0
is i < numDays
false
true
ArrayList list = new ArrayList(employeeNumbers)
int j = 0;
is j < numShifts ?
false
true
Randomly pick Employee Number
ArrayList eligibleEmp = new ArrayList
int k = 0;
is k < availableEmployees.size();
false
int employeeIndex = 0;
is (employeeIndex < numEmployeesNeeded && eligibleEmployees.size() > 0)
true
int empNumber = availableEmployees.get(k)
is empNumber available and is preffered ?
false
k++
true
int employeeNumber = availableEmployeeNumbers.get(random.nextInt(availableEmployeeNumbers.size())
shifts[i][j][employeeIndex] = employeeNumber; employeeIndex++;
employeeIndex++
eligibleEmployees.add(employeeNumber)
Collections.shuffle(employeeNumber)
j++
return shifts;
i++
End

Justification of data structures :

A linked list will be used as the main data structure to store employees. This data structure is dynamic, meaning it can grow in size as the user adds more employees. This makes it well-suited for the task and allows for scalability as the business grows. To implement the linked list, a custom linked list class that has been specialized to store employees will be used. The custom linked list class will provide additional functionality that is not available in the default LinkedList interface, and will allow for future developments to be made easier as custom methods can be implemented, rather than sticking to a pre-made library.

For the generation of shift schedules, I decided to use a 3D array. The algorithm uses a 3D array to store the shifts, where the first dimension represents the day, the second dimension represents the shift, and the third dimension represents the employees assigned to the shift. The size of the third dimension is determined by the maximum number of employees that can be assigned to a shift. I chose this specific data structure over the creation of a class called "Shift", as Java's memory management system works better with arrays than with other data structures, such as lists, as arrays are allocated a contiguous block of memory, which is beneficial for cache performance. In contrast, lists are implemented as a series of linked objects, which can lead to poor cache performance and slower execution times.[2] Moreover, a 3D array allows for efficient indexing and manipulation of the assigned employees, as the algorithm can access and modify the assigned employees for a particular shift directly using array indexing. Furthermore, the 3D array can easily accommodate varying numbers of employees for each shift, as the size of the third dimension can be adjusted based on the maximum number of employees allowed per shift.

[2] (Pizlo and Vitek, 2008)

Test Plan :

No	Action to test	Method of testing	Success criteria	Tested	Result
1	Login	Enter correct details and login screen is displayed	1		
2	Add employees	After employee's information is entered, the employee is added to the DB and Employee List	2		
3	Delete employee	Employee and his information are deleted and un-serialized when deleted.	2		
4	Edit Employee	The employee's information can be modified after its original addition	4		
5	Shift generation	The shift schedule is generated according to user's information, and displayed appropriately	5		
6	PDF for employee's data	Generate PDF based on all of the employee's information in the correct folder	6		
7	PDF for shifts	Generate PDF based on the shift generated, in the correct folder	7		
8	Salaries and tax bracket adjustments for pay slip pdf generation	Pay slip is generated according to the user's input tax brackets, deductions are correct, and the pdf is generated in the right folder accordingly	8		
9	Serialization of data	Data is generated accordingly in a .ser format file. Data can be read without corruption of the stream header, and data can be removed when requested to be removed, without impacting the order of the data. DB serialization also works without any issues.	9		
10	Search for Employee	Program can return an employee based on data given by user. The program gives a list of the best fit employees for data given, such as name, last name.	3		

Word Count : 318

Bibliography

Visual-paradigm.com. (2019). *Ideal Modeling & Diagramming Tool for Agile Team Collaboration*. [online] Available at: https://www.visual-paradigm.com/.

Pizlo, F. and Vitek, J. (2008). Memory Management for Real-Time Java: State of the Art. *2008 11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC)*. doi:https://doi.org/10.1109/isorc.2008.40.