SOFTWARE DESIGN DOCUMENT

(SDD)

TUITION MANAGEMENT SYSTEM



2016

MIND’S GOLDEN SDN BHD

To be submitted to the Software Planning & Requirement Workshop

Bachelor of Computer Science (Software Engineering)

**March 2016**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Approval List Table** | | | | | | |
| Index | 01 | 02 | 03 | 04 | 05 | 06 |
| Writed by:  Name:  Nik Hasnie Binti Nik Soh | Date: | Date: | Date: | Date: | Date: | Date: |
| Verified by:  Quality Manager  Name:  Nurul Nadiah Binti Mohd Fazli | Date: | Date: | Date: | Date: | Date: | Date: |
| Check by:  Configuration Manager  Name:  Zulaikha Binti Zuremi | Date | Date: | Date: | Date: | Date: | Date: |
| Approved by:  Work Package Manager  Name:  Nursyahirman Bin A.Rahman | Date | Date: | Date: | Date: | Date: | Date: |
| Authenticated by:  Project Manager  Name:  Muhammad Aizuddin Bin Nor Hazmi | Date | Date: | Date: | Date: | Date: | Date: |

|  |  |
| --- | --- |
| **Revision History** | |
| **Revision** | **Description** |
| 01 |  |
| 02 |  |
| 03 |  |
| 04 |  |
| 05 |  |
| 06 |  |

Table of Contents

[1. SCOPE 6](#_Toc447094299)

[1.1. Identification 6](#_Toc447094300)

[1.2. Overview of the System 6](#_Toc447094301)

[1.3. Overview of the Document 7](#_Toc447094302)

[2. REFERENCED DOCUMENTS 9](#_Toc447094303)

[2.1. Overview of the Documents 9](#_Toc447094304)

[2.2. Referenced Documents 9](#_Toc447094305)

[3. PRELIMINARY DESIGN 11](#_Toc447094306)

[3.1. System Overview 11](#_Toc447094307)

[3.1.1 System Architecture 11](#_Toc447094308)

[3.2. CSCI Design Description 22](#_Toc447094309)

[3.2.1 Registration 22](#_Toc447094310)

[3.2.2 Teaching Material 22](#_Toc447094311)

[3.2.3 Financial Management 22](#_Toc447094312)

[3.2.4 Attendance 23](#_Toc447094313)

[3.2.5 Student Performance 23](#_Toc447094314)

[4. DETAILED DESIGN 24](#_Toc447094315)

[4.1 Registration 24](#_Toc447094316)

[4.1.1 SystemUserInterface 24](#_Toc447094317)

[4.1.2. StudentManager 30](#_Toc447094318)

[4.1.3. TeacherManager 36](#_Toc447094319)

[4.1.4 StaffManager 41](#_Toc447094320)

[4.1.5. Student 45](#_Toc447094321)

[4.1.6. Teacher 50](#_Toc447094322)

[4.1.7. Staff 54](#_Toc447094323)

[4.1.8. SubjectManager 58](#_Toc447094324)

[4.1.9. Subject 63](#_Toc447094325)

[4.1.10. StudentSubjectRegistrationInterface 69](#_Toc447094326)

[4.1.11. ManageSubjectInterface 72](#_Toc447094327)

[4.1.12. ViewScheduleInterface 76](#_Toc447094328)

[4.2 Teaching Material Interface 79](#_Toc447094329)

[4.2.1 NoteManagementInterface Class Design 79](#_Toc447094330)

[4.2.2 ExerciseManagementInterface Class Design 83](#_Toc447094331)

[4.2.3 OnlineQuizManagementInterface Class Design 87](#_Toc447094332)

[4.2.4 E\_BooksInterface Class Design 90](#_Toc447094333)

[4.3 Financial Management 93](#_Toc447094334)

[4.3.1 FinancialManagementView 93](#_Toc447094335)

[4.3.2 TuitionPaymentView Class Design 96](#_Toc447094336)

[4.3.3 TuitionFeeManager Class Design 99](#_Toc447094337)

[4.3.4 TuitionFee Class Design 102](#_Toc447094338)

[4.3.5 SalaryPaymentView Class Design 105](#_Toc447094339)

[4.3.6 SalaryManager Class Design 108](#_Toc447094340)

[4.3.7 Salary Class Design 110](#_Toc447094341)

[4.3.8 ReportView Class Design 113](#_Toc447094342)

[4.3.9 Class PaySlipManager Class Design 116](#_Toc447094343)

[4.3.10 PaySlip Class Design 118](#_Toc447094344)

[4.3.11 ReceiptManager Class Design 120](#_Toc447094345)

[4.3.12 Class Receipt Design 122](#_Toc447094346)

[4.4 Attendance 124](#_Toc447094347)

[4.4.1 Attendance Interface Class Design 124](#_Toc447094348)

[4.4.2 Manage Warning Letter Interface Class Design 127](#_Toc447094349)

[4.4.3 Warning Letter Manager Class Design 129](#_Toc447094350)

[4.4.4 Warning Letter Class Design 132](#_Toc447094351)

[4.4.5 Attendance Manager Class Design 134](#_Toc447094352)

[4.4.6 Attendance Class Design 138](#_Toc447094353)

[4.5 Student Performance 142](#_Toc447094354)

[4.5.1. Student Performance Interface Design 142](#_Toc447094355)

[4.5.2. Evaluation Interface Class Design 144](#_Toc447094356)

[4.1.3. Evaluation Form Interface Class Design 148](#_Toc447094357)

[4.1.4. Evaluation Manager Class Design 150](#_Toc447094358)

[4.5.5. View Evaluation Interface Class Design 154](#_Toc447094359)

[4.1.6. Evaluation Class Design 156](#_Toc447094360)

[4.1.7. Result Interface Class Design 158](#_Toc447094361)

[4.1.8. Exercise Mark Interface Class Design 161](#_Toc447094362)

[4.1.9. Add Result Interface Class Design 163](#_Toc447094363)

[4.1.10. Edit Result Interface Class Design 165](#_Toc447094364)

[4.1.11. Mark Manager Class Design 167](#_Toc447094365)

[4.1.12. View Result Interface Class Design 170](#_Toc447094366)

[4.1.13. Mark Class Design 172](#_Toc447094367)

[5. NOTES 174](#_Toc447094368)

# SCOPE

## Identification

System name : Tuition Management System

Abbreviation : TMS

System ID No. : TMS 1212

## Overview of the System

Tuition Management System is a web-based system that will be used in tuition centre. This system is basically specializing on managing a tuition centre. There are five main modules, which are registration module, teaching module, student performance module, finance module and attendance module.

In registration module, new student and teacher will be registered into the system. After registration, user will be given id and password to login into the system. After any user log in the system, student and teacher can register subjects. Student registers subject to take and teacher register subject to teach the student.

In teaching module, teacher will add, modify or delete any teaching material for the student. Student can retrieve the teaching material that had been managed by teacher. Student can view, print or download the teaching material from this module. This module also provide online quiz for students.

In student performance module, it contains two options which are evaluation and result. Student can make evaluation for teacher and the same goes to teacher. All users in the system can view the evaluation. Teacher will key in final result in the system, student and staff are able to view the result.

For the finance module, the system will calculate the student fees based on the subject taken and what type of class or section they take.

Lastly, in attendance module, it will record attendance of student, teachers and staff. The entire user will get warning letter when absent more than 3 times.

Purpose of the System:

The purpose of TMS is to help user to manage tuition center in many aspects. TMS offer new functionality and features to existing system. By using this system, we can replace manual work for human, that means we can do all the work using online system. This system also can help human to work more effecience in their work as well.

The main purpose of the system is to ease the management of the tuition management system in term of manage all the data systematically and efficiently. TMS brings advantages to the 3 main users of the system:

1. Help staffs to manage registration systematically and increase efficiency.
2. Help students to access their note anywhere and take quiz online.
3. Help to manage anything that is related to financing.
4. Enable the student to view results and evaluation of their performance.
5. To ease to record attendance of student, teacher and staff.

## Overview of the Document

This paragraph summarizes the purpose and contents of Software Design Document (SDD). It specifies the requirement aspects related to the Tuition Management System.

In general, this SDD is divided into 5 sections as the following:

|  |  |
| --- | --- |
| Chapter 1 | Describes the scope identification, system overview and the document overview. |
| Chapter 2 | Referenced documents |
| Chapter 3 | Describes the preliminary design for the CSCI. The preliminary design will identify CSC, the description of each CSC design, the characteristics of each CSC and the traceability of requirements set forth in SRS and IRS. |
| Chapter 4 | Describes the detail design. |
| Chapter 5 | Notes and abbreviations. |

# REFERENCED DOCUMENTS

This section lists the document number and document name referenced in this document. Any discrepancies of this document in describing the software development process should be covered by the documents listed in this chapter. The following documents were referred as the basis for this SDD preparation.

Copies of specification, standards, drawings and publication requested by suppliers in contact with the specified supplying functions may be obtained by contacting the agency or directly through the contracting office.

## Overview of the Documents

Below are the list of standard that been used.

1. 1016-1998 - IEEE-SA Standards Board (23 September 1998). IEEE Recommended Practice for Software Design Descriptions. Retrieved from: <http://goo.gl/u6FzsA>
2. ANDIOS, (17 January 2013), Software Design Document Version 1.1 Mobcoll Project. Retrieved from: <http://161.139.18.27/14152/pluginfile.php/315714/mod_resource/content/1/SDD%20Mobcoll.pdf>
3. CS 480 / CS 481 ~ Project. Software Design Description (SDD) Guidelines. Retrieved from: <http://goo.gl/XyRYT7>
4. Michiel van der Wulp, (2009), ArgoUML User Manual: A tutorial and reference description. Retrieved from: <https://goo.gl/QRPlhv>

## 2.2. Referenced Documents

This section consists of the contractual documents and non-contractual documents.

1. Mind’s Golden Sdn Bhd, TMS-2015-01, (2015), Software Requirement Specification (SRS) Document for Tuition Management System.
2. Haryani Bt Ibrahim, (2007). Tuition Management System. Retrived from: <http://goo.gl/Tu6TFo>

# PRELIMINARY DESIGN

## 3.1. System Overview

Figure shows the overview of Tuition Management System.

|  |
| --- |
|  |

Figure 3.1 : System overview Tuition Management System

### 3.1.1 System Architecture

This paragraph identifies the internal organizational structure of the system. The relationship among system subsystem will be described.

#### 3.1.1.1. Static Organization

Figure shows the packages available in Tuition Management System.

|  |
| --- |
|  |

Figure 3.2 : Static Organization of Tuition Management System

This section describes the detail for each subsystem/package .

1. Registration

This package is responsible to register the user of the system that includes student, teacher and staff. It will also enable the student to register for the subject that they desired.

1. SystemUserInterface class

* This class will give user three choices whether to add user, edit user or deleting user. This class will need add system user interface class and edit system user interface.
* This class allow user to delete user and all its information.

1. AddSystemUserInterface

* This class give user three option, whether to add student, add teacher or add staff and need user to enter the student, teacher or staff information.

1. EditSystemUserInterface

* This class also provide user with three choices, either edit student, edit teacher or edit staff and need user to enter the student, teacher or staff information.

1. StudentManager

* This class send student and parent details to the database.
* This class send edited student and parent details to the database.
* This class also need user to enter student id in order to delete the student from the system.
* This class will verify the student id whether it exist or not in the database so that all of their data can be deleted.

1. TeacherManager

* This class send new teacher information to the database.
* This class send edited teacher information to the database.
* This class need teacher id in order to delete teacher from the database.
* Then, it will verify either teacher with that id is exist in the system in order to delete all their information.

1. StaffManager

* It also send new staff information to the database.
* It also send edited staff information to the database.
* In order to delete staff and all its information from the database, this class need user to enter staff id.

1. Student

* This class save data get from student manager.
* It will also verify and check the data of the student by using student id.
* It will delete the student details when receive the request from the user.

1. Teacher

* This class save the data get from teacher manager.
* It also verify and check data of the teacher by using teacher id.
* It will delete teacher information when receive the request from the user.

1. Staff

* This class save the data get from staff manager.
* It will verify the staff information by using staff id.
* It will delete staff information when receive the request from the user.

1. StudentSubjectRegistrationInterface

* This class allow user to enter student id in order to start registering the student’s subject.

1. SubjectManager

* This class provide subject for user to choose subject in order to registering student’s subject.
* This class send entered or updated the subject details in order to manage subject to the database.
* This class allow user to retrieve the subject details in order to view the subject details.
* This class also will provide the schedule for the student and teacher.

1. Subject

* This class will verify the subject, either exist or not.
* This class also record the student who register for particular subject.
* This class save the data get from subject manager.
* It can retrieve subject details.
* It can retrieve the schedule for student and teacher by checking the subject student register and teacher assign to.

1. ManageSubjectInterface

* This class provide three options for the user either add subject, edit subject or view subject.
* This class need staff to enter subject information

1. ViewScheduleInterface

* It allow user to view the schedule generated in subject manager and subject.

1. Teaching Material

This package is responsible to manage student and teachers note, exercise, online quizzes, and their e\_books. It consist of 13 classes which is TeachingMaterialInterface, NoteManagementInterface, NoteManager, Note, ExerciseManagementInterface, ExerciseManager, Exercise, OnlineQuizManagemenetInterface, OnlinequizManager, OnlineQuiz, E\_BooksManagementInterface, E\_BooksManager, E\_Books.

1. TeachingMaterialInterface

* Acts the main page for the teaching material function. It allows the users to navigate to exercise, note, online quiz, e\_book.

1. NoteManagementInterface

* Enables user to add note, update note, delete note, download note, print note, or save note.

1. NoteManagementManager

* Do all the operations involving note such as add note, update note, delete note, download note, print note, and save note.

1. Note

* Store notes of all students and able to retrieve notes if requested by the user.

1. ExerciseManagementInterface

* Enable the user to add exercise, update exercise, delete exercise, save exercise, download exercise, print exercise, or save the exercise.

1. ExerciseManager

* Do all the operations involving exercise such as add exercise, update exercise, delete exercise, save exercise, download exercise, print exercise, and save the exercise.

1. Exercise

* Store exercise of all the students and teachers that have been uploaded or deleted. It can also able to retrieve the exercise based on ID.

1. OnlineQuizManagementInterface

* Enable the user to upload quiz, open quiz, delete quiz, attend quiz and submit quiz.

1. OnlineQuizManager

* Do all the operations involving online quiz such as upload quiz, open quiz, delete quiz, attend quiz and submit quiz.

1. OnlineQuiz

* Store online quiz of all the users that have been uploaded or deleted. It also can be able to retrieve the online quiz based on ID.

1. E\_BooksManagementInterface

* Enable the user to upload the e\_books, add e\_books, delete e\_books, download e\_books, and preview e\_books.

1. E\_BooksManager

* Do all the operations involving e\_books such as upload e\_books, add e\_books, delete e\_books, download e\_books, and preview e\_books

1. E\_Books

* Store E\_Books of all the users that upload or deleted. It can also be able to retrieve the e\_books based in ID.

1. FinancialManagement

This package is responsible to manage tuition fee, salary and report. It consists of 11 classes that includes FinancialManagementView, TuitionPaymentView, TuitionFeeManager, SalaryPaymentView, SalaryManager, Salary, ReportView, PaySlipManager, PaySlip, ReceiptManager and Receipt.

* 1. FinancialManagementView
* Act as the main page for financial function. It allows the user to navigate to tuition payment, salary payment or report page.
  1. TuitionPaymentView
* Enable the user to either insert payment of tuition fee, view tuition fee or calculate tuition fee.
  1. TuitionFeeManager
* Do all the operations involving tuition fee calculations, inserting tuition fee payment and viewing tuition fee.
  1. TuitionFee
* Store tuition fee of all the students and tuition fee payment made. It also enable to retrieve tuition fee amount.
  1. SalaryPaymentView
* Enable the user to either insert payment of salary or view salary.
  1. SalaryManager
* Do all the operations involving salary calculations, inserting salary payment and viewing salary payment.
  1. Salary
* Store salary of all the teachers and salary payment made. It also enable to retrieve salary amount.
  1. ReportView
* Enable the user to either generate receipt, get receipt, generate pay slip or get pay slip.
  1. PaySlipManager
* Generate pay slip for teacher or view their pay slip.
  1. PaySlip
* Store pay slip that has been generated and enable the user to retrieve the pay slip.
  1. ReceiptManager
* Generate receipt for student or view their receipt for every payment made.
  1. Receipt
* Store receipt that has been generated and enable the user to retrieve the receipt.

1. Attendance

This package is responsible to manage attendance and manage warning letter. It consists of 6 classes that include AttendanceInterfaces, ManageWarningInterface, AttendanceManager, Attendance, WarningLetterManager, and WarningLetter.

1. AttendanceInterface.

* This class can view attendance, can add, delete and update attendance.

1. ManageWarningLetterInterface

* This class can check absent students that are more 3 times absent.

1. AttendanceManager

* This class can get attendance, set attendance, update attendance, delete attendance and enter date and time.
* This class also sends information about all attendance for staff, teacher and student to database.

1. WarningLetterManager

* This class sends information staffId, studentId, teachId, subjCode, date, stdAddr and information from student manager and warning letter to database.
* This class will allow generating warning letter and get student info.

1. WarningLetter

* Acts as database for warning letter information.
* This class allows setting warning letter by use stdId and subjCode.

1. Attendance

* Attendance act as database that are store the data about student information, teacher information and staff information like their id.
* It also will get attendance, check attendance, set attendance and delete attendance by use their id.

1. Student Performance Module

Brief explanation by stating the function and classes in the package.

1. StudentPerformanceInterface

* This class will provide interface for two option. The first option is evaluation interface and the second option is result interface.

1. EvaluationInterface

* This interface provide user with 3 function, which is add student evaluation, add teacher evaluation and also edit evaluation for both teacher and student.

1. EvaluationFormInterface

* Evaluation form interface use to represent the interface for user to give evaluation to student and teacher.

1. EvaluationManager

* After user give comment to student or teacher, evaluation manager will receive the evaluation and send to database to be set and store into itself.

1. ViewEvaluationInterface

* This interface provide user to view evaluation for student and teacher.

1. Evaluation

* Act as a database for evaluation. It will store data evaluation that has been insert by teacher and student.

1. ResultInterface

* This class provide user with 3 interface. First interface is for teacher retrieve quiz mark from the database. Second, for teacher add result for student and third for teacher edit result of student.

1. ExerciseMarkInterface

* Interface that represent for user to use the system. Represent interface about the exercise marks

1. Add Result Interface

* Interface that enable teacher to add marks to student. Represent interface and function add in the system.

1. Edit Result Interface

* Interface that enable teacher to add marks to student. Represent interface and function in the system.

1. Mark Manager

* Receive any task from other class either they can to save, search or add any data in the system.

1. View Result Interface

* Provide user with the interface to view student result.

1. Mark

* Database that store marks of students.

#### 3.1.1.2. Dynamic Organization

Figure diagram shows components and their relationships between each other in System

|  |
| --- |
|  |

Figure 3.3 : Dynamic Organization of Tuition Management System

#### 3.1.1.3. CSCs Interfaces

Not applicable

#### 3.1.1.4. External Interfaces

Not applicable

## 3.2. CSCI Design Description

Tuition Management System consists of 5 subsytem that includes Registration, Teaching Material, Financial Management, Attendance and Student Performance.

### 3.2.1 Registration

Refer to Appendix A-1

### 3.2.2 Teaching Material

Refer to Appendix A-2

### 3.2.3 Financial Management

Refer to Appendix A-3

### 3.2.4 Attendance

Refer to Appendix A-4

### 3.2.5 Student Performance

Refer to Appendix A-5

# DETAILED DESIGN

## 4.1 Registration

### 4.1.1 SystemUserInterface

The purpose of this class is to manage user information either to add, edit or delete.

**4.1.1.1 Class SystemUserInterface Design**

This subparagraph specifies the design of SystemUserInterface

1. Input/output data elements

List of input and output data elements:

Table 1: Input and Output Date Element for Class SystemUserInterface

|  |  |
| --- | --- |
| Input | Output |
| StdId, teacherId, staffId, stdName, stdAddr, stdDob, stdContNo, stdGender, stdAge, parentName, parentAddr, parentContNo, parentAge, parentJob, t\_Name, t\_Addr, t\_IcNo, t\_ContNo, t\_Dob, t\_Gender, t\_Age, t\_Nationality, t\_Email, t\_EduLvl, sName, sAddr, sIcNo, sContNo, sDob, sGender, sAge, sNationality, sEmail | Delete user, Username and password |

1. Local data elements

Table 1: Local Date Element for Class SystemUserInterface

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Name | Data type | Description |
| 1 | stdId | String | Student ID |
| 2 | teacherId | String | Teacher ID |
| 3 | staffId | String | Staff ID |
| 4 | stdName | String | Student name |
| 5 | stdAddr | String | Student address |
| 6 | stdDob | String | Student date of birth |
| 7 | stdContNo | String | Student contact number |
| 8 | stdGender | String | Student gender |
| 9 | stdAge | Integer | Student age |
| 10 | parentName | String | Student’s parent name |
| 11 | parentAddr | String | Student’s parent address |
| 12 | parentContNo | String | Student’s parent contact number |
| 13 | parentAge | Integer | Student’s parent age |
| 14 | parentJob | String | Student’s parent job |
| 15 | t\_Name | String | Teacher name |
| 16 | t\_Addr | String | Teacher Address |
| 17 | t\_IcNo | Integer | Teacher identification number |
| 18 | t\_ContNo | String | Teacher contact number |
| 19 | t\_Dob | String | Teacher date of birth |
| 20 | t\_Gender | String | Teacher gender |
| 21 | t\_Age | Integer | Teacher age |
| 22 | t\_Nationality | String | Teacher nationality |
| 23 | t\_Email | String | Teacher email |
| 24 | t\_EduLvl | String | Teacher educational level |
| 25 | sName | String | Staff name |
| 26 | sAddr | String | Staff address |
| 27 | sIcNo | Integer | Staff identification number |
| 28 | sContNo | String | Staff contact number |
| 29 | sDob | String | Staff date of birth |
| 30 | sGender | String | Staff gender |
| 31 | sAge | Integer | Staff age |
| 32 | sNationality | String | Staff nationality |
| 33 | sEmail | String | Staff email |
| 34 | subjCode | String | Subject code |
| 35 | subjName | String | Subject name |
| 36 | day | String | Day for the subject take place |
| 37 | time | Integer | Time for the learning of the subject |
| 38 | place | String | Place where subject take place |
| 39 | price | Integer | Price of the subject |

1. Algorithms

This section states the purpose and describes in detail the algorithms of this class  
Table 1: Detail for Class SystemUserInterface

|  |  |
| --- | --- |
| Class Type | Interface |
| Responsibility | Delete user, add user and edit user. |
| Attributes | stdName, stdAddr, stdDob, stdContNo, stdGender, stdAge, parentName, parentAddr, parentContNo, parentAge, parentJob, t\_Name, t\_Addr, t\_IcNo, t\_ContNo, t\_Dob, t\_Gender, t\_Age, t\_Nationality, t\_Email, t\_EduLvl, sName, sAddr, sIcNo, sContNo, sDob, sGender, sAge, sNationality, sEmail, stdId, teacherId, staffId |
| Methods | deleteUser(), addStudent(), addTeacher(), addStaff(),editStudent(), editTeacher(), editStaff() |

1. deleteUser()

Table 2: Algorithm for deleteUser

|  |  |
| --- | --- |
| Responsibility | Delete user from the system |
| Input Parameter | stdId, teacherId or staffId |
| Output Parameter | - |
| Algorithm | BEGIN  User enter either student id (stdId), teacher id (teacherId) or staff id (staffId).  User click on confirmation message.  END |

1. addStudent()

Table 3: Algorithm for addStudent

|  |  |
| --- | --- |
| Responsibility | Enter the details of the student and student’s parent. |
| Input Parameter | stdName, stdAddr, stdDob, stdContNo, stdGender, stdAge, parentName, parentAddr, parentContNo, parentAge, parentJob |
| Output Parameter | - |
| Algorithm | BEGIN  User choose add student.  User enter the information needed.  User click save.  END |

1. addTeacher()

Table 4: Algorithm for addTeacher

|  |  |
| --- | --- |
| Responsibility | Enter the details of the teacher |
| Input Parameter | t\_Name, t\_Addr, t\_IcNo, t\_ContNo, t\_Dob, t\_Gender, t\_Age, t\_Nationality, t\_Email, t\_EduLvl |
| Output Parameter | - |
| Algorithm | BEGIN  User choose add teacher.  User enter the information needed.  User click save.  END |

1. addStaff()

Table 5: Algorithm for addStaff

|  |  |
| --- | --- |
| Responsibility | Enter the details of the new staff. |
| Input Parameter | sName, sAddr, sIcNo, sContNo, sDob, sGender, sAge, sNationality, sEmail |
| Output Parameter | - |
| Algorithm | BEGIN  User choose add staff.  User enter the information needed.  User click save.  END |

1. editStudent()

Table 6: Algorithm for editStudent

|  |  |
| --- | --- |
| Responsibility | Edit the details of the student and student’s parent. |
| Input Parameter | stdName, stdAddr, stdDob, stdContNo, stdGender, stdAge, parentName, parentAddr, parentContNo, parentAge, parentJob |
| Output Parameter | - |
| Algorithm | BEGIN  User choose edit student.  User edit the information that needed to be update.  User click save.  END |

1. editTeacher()

Table 7: Algorithm for editTeacher

|  |  |
| --- | --- |
| Responsibility | Edit the details of the teacher |
| Input Parameter | t\_Name, t\_Addr, t\_IcNo, t\_ContNo, t\_Dob, t\_Gender, t\_Age, t\_Nationality, t\_Email, t\_EduLvl |
| Output Parameter | - |
| Algorithm | BEGIN  User choose edit teacher.  User edit the information that needed to be update.  User click save.  END |

1. editStaff()

Table 8: Algorithm for editStaff

|  |  |
| --- | --- |
| Responsibility | Edit the details of the new staff |
| Input Parameter | sName, sAddr, sIcNo, sContNo, sDob, sGender, sAge, sNationality, sEmail |
| Output Parameter | - |
| Algorithm | BEGIN  User choose edit staff.  User edit the information that needed to be update.  User click save.  END |

### 

### 4.1.2. StudentManager

The purpose of this class is to send the student information into the database.

4.1.2.1. Class StudentManager Design

This subparagraph specifies the design of StudentManager

1. Input/output data elements

List of input and output data elements:

Table 1: Input and Output Date Element for StudentManager Class

|  |  |
| --- | --- |
| Input | Output |
| stdName, stdAddr, stdDob, stdContNo, stdGender, stdAge, parentName, parentAddr, parentContNo, parentAge, parentJob | - |

1. Local data elements

Table 1: Local Date Element for StudentManager Class

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Name | Data type | Description |
| 1 | stdId | String | Student ID |
| 2 | teacherId | String | Teacher ID |
| 3 | staffId | String | Staff ID |
| 4 | stdName | String | Student name |
| 5 | stdAddr | String | Student address |
| 6 | stdDob | String | Student date of birth |
| 7 | stdContNo | String | Student contact number |
| 8 | stdGender | String | Student gender |
| 9 | stdAge | Integer | Student age |
| 10 | parentName | String | Student’s parent name |
| 11 | parentAddr | String | Student’s parent address |
| 12 | parentContNo | String | Student’s parent contact number |
| 13 | parentAge | Integer | Student’s parent age |
| 14 | parentJob | String | Student’s parent job |
| 15 | t\_Name | String | Teacher name |
| 16 | t\_Addr | String | Teacher Address |
| 17 | t\_IcNo | Integer | Teacher identification number |
| 18 | t\_ContNo | String | Teacher contact number |
| 19 | t\_Dob | String | Teacher date of birth |
| 20 | t\_Gender | String | Teacher gender |
| 21 | t\_Age | Integer | Teacher age |
| 22 | t\_Nationality | String | Teacher nationality |
| 23 | t\_Email | String | Teacher email |
| 24 | t\_EduLvl | String | Teacher educational level |
| 25 | sName | String | Staff name |
| 26 | sAddr | String | Staff address |
| 27 | sIcNo | Integer | Staff identification number |
| 28 | sContNo | String | Staff contact number |
| 29 | sDob | String | Staff date of birth |
| 30 | sGender | String | Staff gender |
| 31 | sAge | Integer | Staff age |
| 32 | sNationality | String | Staff nationality |
| 33 | sEmail | String | Staff email |
| 34 | subjCode | String | Subject code |
| 35 | subjName | String | Subject name |
| 36 | day | String | Day for the subject take place |
| 37 | time | Integer | Time for the learning of the subject |
| 38 | place | String | Place where subject take place |
| 39 | price | Integer | Price of the subject |

1. Algorithms

This section states the purpose and describes in detail the algorithms of this class

Table 1: Detail for StudentManager Class

|  |  |
| --- | --- |
| Class Type | Manager class |
| Responsibility | Send the student information into the database |
| Attributes | stdName, stdAddr, stdDob, stdContNo, stdGender, stdAge, parentName, parentAddr, parentContNo, parentAge, parentJob |
| Methods | enterStdInfo(stdName, stdAddr, stdDob, stdContNo, stdGender, stdAge), enterParentInfo(parentName, parentAddr, parentContNo, parentAge, parentJob), editStdInfo(stdName, stdAddr, stdDob, stdContNo, stdGender, stdAge), editParentInfo(parentName, parentAddr, parentContNo, parentAge, parentJob), enterStdId(stdId), verifyStdId(stdId) |

1. enterStdInfo(stdName, stdAddr, stdDob, stdContNo, stdGender, stdAge)

Table 2: Algorithm for enterSdInfo

|  |  |
| --- | --- |
| Responsibility | send student info to the database |
| Input Parameter | stdName, stdAddr, stdDob, stdContNo, stdGender, stdAge |
| Output Parameter | - |
| Algorithm | BEGIN  Get student information from AddSystemUserInterface class.  Check for missing information.  Send information into Student class.  END |

1. enterParentInfo(parentName, parentAddr, parentContNo, parentAge, parentJob)

Table 3: Algorithm for enterParentInfo

|  |  |
| --- | --- |
| Responsibility | send student’s parent information to the database |
| Input Parameter | parentName, parentAddr, parentContNo, parentAge, parentJob |
| Output Parameter | - |
| Algorithm | BEGIN  Get parent information from AddSystemUserInterface class.  Check for missing information.  Send the information into Student class.  END |

1. editStdInfo(stdName, stdAddr, stdDob, stdContNo, stdGender, stdAge)

Table 4: Algorithm for editStdInfo

|  |  |
| --- | --- |
| Responsibility | send edited student’s information to the database |
| Input Parameter | stdName, stdAddr, stdDob, stdContNo, stdGender, stdAge |
| Output Parameter | - |
| Algorithm | BEGIN  Get student information from EditSystemUserInterface class.  Check for missing information.  Send the information into Student class.  END |

1. editParentInfo(parentName, parentAddr, parentContNo, parentAge, parentJob)

Table 5: Algorithm for editParentInfo

|  |  |
| --- | --- |
| Responsibility | send edit student’s parent information to the database |
| Input Parameter | parentName, parentAddr, parentContNo, parentAge, parentJob |
| Output Parameter | - |
| Algorithm | BEGIN  Get edited parent information from EditSystemUserInterface class.  Check for missing information.  Send the information into Student class.  END |

1. enterStdId(stdId),

Table 6: Algorithm for enterStdId

|  |  |
| --- | --- |
| Responsibility | Enter student id |
| Input Parameter | stdId |
| Output Parameter | - |
| Algorithm | BEGIN  Staff enter student id.  Staff click okay button.  END |

1. verifyStdId(stdId)

Table7: Algorithm for verifyStdId

|  |  |
| --- | --- |
| Responsibility | Verify if student exist or not in the database based in student id |
| Input Parameter | stdId |
| Output Parameter | - |
| Algorithm | BEGIN  Get the student id.  Check student id in database.  END |

### 4.1.3. TeacherManager

The purpose of this class is to send the teacher’s information to the database

**4.1.3.1. Class TeacherManager Design**

This subparagraph specifies the design of TeacherManager

1. Input/output data elements

List of input and output data elements:

Table 1: Input and Output Date Element for Class Teacher Manager

|  |  |
| --- | --- |
| Input | Output |
| t\_Name, t\_Addr, t\_IcNo, t\_ContNo, t\_Dob, t\_Gender, t\_Age, t\_Nationality, t\_Email, t\_EduLvl | - |

1. Local data elements

Table 1: Local Date Element for Class Teacher Manager

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Name | Data type | Description |
| 1 | stdId | String | Student ID |
| 2 | teacherId | String | Teacher ID |
| 3 | staffId | String | Staff ID |
| 4 | stdName | String | Student name |
| 5 | stdAddr | String | Student address |
| 6 | stdDob | String | Student date of birth |
| 7 | stdContNo | String | Student contact number |
| 8 | stdGender | String | Student gender |
| 9 | stdAge | Integer | Student age |
| 10 | parentName | String | Student’s parent name |
| 11 | parentAddr | String | Student’s parent address |
| 12 | parentContNo | String | Student’s parent contact number |
| 13 | parentAge | Integer | Student’s parent age |
| 14 | parentJob | String | Student’s parent job |
| 15 | t\_Name | String | Teacher name |
| 16 | t\_Addr | String | Teacher Address |
| 17 | t\_IcNo | Integer | Teacher identification number |
| 18 | t\_ContNo | String | Teacher contact number |
| 19 | t\_Dob | String | Teacher date of birth |
| 20 | t\_Gender | String | Teacher gender |
| 21 | t\_Age | Integer | Teacher age |
| 22 | t\_Nationality | String | Teacher nationality |
| 23 | t\_Email | String | Teacher email |
| 24 | t\_EduLvl | String | Teacher educational level |
| 25 | sName | String | Staff name |
| 26 | sAddr | String | Staff address |
| 27 | sIcNo | Integer | Staff identification number |
| 28 | sContNo | String | Staff contact number |
| 29 | sDob | String | Staff date of birth |
| 30 | sGender | String | Staff gender |
| 31 | sAge | Integer | Staff age |
| 32 | sNationality | String | Staff nationality |
| 33 | sEmail | String | Staff email |
| 34 | subjCode | String | Subject code |
| 35 | subjName | String | Subject name |
| 36 | day | String | Day for the subject take place |
| 37 | time | Integer | Time for the learning of the subject |
| 38 | place | String | Place where subject take place |
| 39 | price | Integer | Price of the subject |

1. Algorithms

This section states the purpose and describes in detail the algorithms of this class

Table 1: Detail for Class Teacher Manager

|  |  |
| --- | --- |
| Class Type | Manager class |
| Responsibility | Send the teacher’s information to the database |
| Attributes | t\_Name, t\_Addr, t\_IcNo, t\_ContNo, t\_Dob, t\_Gender, t\_Age, t\_Nationality, t\_Email, t\_EduLvl |
| Methods | enterTeacherInfo(t\_Name, t\_Addr, t\_IcNo, t\_ContNo, t\_Dob, t\_Gender, t\_Age, t\_Nationality, t\_Email, t\_EduLvl), editTeacherInfo(t\_Name, t\_Addr, t\_IcNo, t\_ContNo, t\_Dob, t\_Gender, t\_Age, t\_Nationality, t\_Email, t\_EduLvl), enterTeacherId(teacherId), verifyTeacherId(teacherId) |

1. enterTeacherInfo(t\_Name, t\_Addr, t\_IcNo, t\_ContNo, t\_Dob, t\_Gender, t\_Age, t\_Nationality, t\_Email, t\_EduLvl)

Table 2: Algorithm for enterTeacherInfo

|  |  |
| --- | --- |
| Responsibility | Send teacher’s information to the database |
| Input Parameter | t\_Name, t\_Addr, t\_IcNo, t\_ContNo, t\_Dob, t\_Gender, t\_Age, t\_Nationality, t\_Email, t\_EduLvl |
| Output Parameter | - |
| Algorithm | BEGIN  Get teacher information from AddSystemUserInterface.  Check for missing information.  Send the teacher’s information to the Teacher class.  END |

1. editTeacherInfo(t\_Name, t\_Addr, t\_IcNo, t\_ContNo, t\_Dob, t\_Gender, t\_Age, t\_Nationality, t\_Email, t\_EduLvl),

Table 3: Algorithm for editTeacherInfo

|  |  |
| --- | --- |
| Responsibility | Send edited teacher’s information to the database |
| Input Parameter | t\_Name, t\_Addr, t\_IcNo, t\_ContNo, t\_Dob, t\_Gender, t\_Age, t\_Nationality, t\_Email, t\_EduLvl |
| Output Parameter | - |
| Algorithm | BEGIN  Get the teacher’s information from EditSystemUserInterface.  Check form missing information.  Send the teacher’s information to Teacher class.  END |

1. enterTeacherId(teacherId),

Table 4: Algorithm for enterTeacherId

|  |  |
| --- | --- |
| Responsibility | Staff enter teacher id |
| Input Parameter | teacherId |
| Output Parameter | - |
| Algorithm | BEGIN  Staff enter student id  Staff click okay button  END |

1. verifyTeacherId(teacherId)

Table 5: Algorithm for verifyTeacherId

|  |  |
| --- | --- |
| Responsibility | Verify whether teacher exist or not based on teacher id |
| Input Parameter | teacherId |
| Output Parameter | - |
| Algorithm | BEGIN  Get teacher id  Check the teacher id in the database  END |

### 

### 4.1.4 StaffManager

The purpose of this class is to send the staff information to the database

**4.1.4.1 Class StaffManager Design**

This subparagraph specifies the design of StaffManager

1. Input/output data elements

List of input and output data elements:

Table 1: Input and Output Date Element for Class StaffManager

|  |  |
| --- | --- |
| Input | Output |
| sName, sAddr, sIcNo, sContNo, sDob, sGender, sAge, sNationality, sEmail | - |

1. Local data elements

Table 1: Local Date Element for Class StaffManager

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Name | Data type | Description |
| 1 | stdId | String | Student ID |
| 2 | teacherId | String | Teacher ID |
| 3 | staffId | String | Staff ID |
| 4 | stdName | String | Student name |
| 5 | stdAddr | String | Student address |
| 6 | stdDob | String | Student date of birth |
| 7 | stdContNo | String | Student contact number |
| 8 | stdGender | String | Student gender |
| 9 | stdAge | Integer | Student age |
| 10 | parentName | String | Student’s parent name |
| 11 | parentAddr | String | Student’s parent address |
| 12 | parentContNo | String | Student’s parent contact number |
| 13 | parentAge | Integer | Student’s parent age |
| 14 | parentJob | String | Student’s parent job |
| 15 | t\_Name | String | Teacher name |
| 16 | t\_Addr | String | Teacher Address |
| 17 | t\_IcNo | Integer | Teacher identification number |
| 18 | t\_ContNo | String | Teacher contact number |
| 19 | t\_Dob | String | Teacher date of birth |
| 20 | t\_Gender | String | Teacher gender |
| 21 | t\_Age | Integer | Teacher age |
| 22 | t\_Nationality | String | Teacher nationality |
| 23 | t\_Email | String | Teacher email |
| 24 | t\_EduLvl | String | Teacher educational level |
| 25 | sName | String | Staff name |
| 26 | sAddr | String | Staff address |
| 27 | sIcNo | Integer | Staff identification number |
| 28 | sContNo | String | Staff contact number |
| 29 | sDob | String | Staff date of birth |
| 30 | sGender | String | Staff gender |
| 31 | sAge | Integer | Staff age |
| 32 | sNationality | String | Staff nationality |
| 33 | sEmail | String | Staff email |
| 34 | subjCode | String | Subject code |
| 35 | subjName | String | Subject name |
| 36 | day | String | Day for the subject take place |
| 37 | time | Integer | Time for the learning of the subject |
| 38 | place | String | Place where subject take place |
| 39 | price | Integer | Price of the subject |

1. Algorithms

This section states the purpose and describes in detail the algorithms of this class

Table 1: Detial for Class StaffManager

|  |  |
| --- | --- |
| Class Type | Manager class |
| Responsibility | Send the entered or edited information to the database |
| Attributes | sName, sAddr, sIcNo, sContNo, sDob, sGender, sAge, sNationality, sEmail |
| Methods | enterStaffInfo(sName, sAddr, sIcNo, sContNo, sDob, sGender, sAge, sNationality, sEmail), editStaffInfo(sName, sAddr, sIcNo, sContNo, sDob, sGender, sAge, sNationality, sEmail), enterStaffId(staffId) |

1. enterStaffInfo(sName, sAddr, sIcNo, sContNo, sDob, sGender, sAge, sNationality, sEmail)

Table 2: Algorithm for enterStaffInfo

|  |  |
| --- | --- |
| Responsibility | Send the entered staff’s information to the database |
| Input Parameter | sName, sAddr, sIcNo, sContNo, sDob, sGender, sAge, sNationality, sEmail |
| Output Parameter | - |
| Algorithm | BEGIN  Get staff information from AddSystemUserInterface class  Check for missing information  Send staff information to the database  END |

1. editStaffInfo(sName, sAddr, sIcNo, sContNo, sDob, sGender, sAge, sNationality, sEmail)

Table 3: Algorithm for editStaffInfo

|  |  |
| --- | --- |
| Responsibility | Send edited staff’s information to the database |
| Input Parameter | sName, sAddr, sIcNo, sContNo, sDob, sGender, sAge, sNationality, sEmail |
| Output Parameter | - |
| Algorithm | BEGIN  Get edited staff’s information from EditSystemUserInterface class  Check for any missing information  Send edited staff’s information to the database  END |

1. enterStaffId(staffId)

Table 4: Algorithm for enterStaffId

|  |  |
| --- | --- |
| Responsibility | Staff enter staff id |
| Input Parameter | staffId |
| Output Parameter | - |
| Algorithm | BEGIN  staff enter staff id  staff click okay button  END |

### 4.1.5. Student

The purpose of this class is to save the student information.

4.1.5.1. Class Student Design

This subparagraph specifies the design of Student

1. Input/output data elements

List of input and output data elements:

Table 1: Input and Output Date Element for Class Student

|  |  |
| --- | --- |
| Input | Output |
| stdName, stdAddr, stdDob, stdContNo, stdGender, stdAge, parentName, parentAddr, parentContNo, parentAge, parentJob | - |

1. Local data elements

Table 1: Local Date Element for Class Student

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Name | Data type | Description |
| 1 | stdId | String | Student ID |
| 2 | teacherId | String | Teacher ID |
| 3 | staffId | String | Staff ID |
| 4 | stdName | String | Student name |
| 5 | stdAddr | String | Student address |
| 6 | stdDob | String | Student date of birth |
| 7 | stdContNo | String | Student contact number |
| 8 | stdGender | String | Student gender |
| 9 | stdAge | Integer | Student age |
| 10 | parentName | String | Student’s parent name |
| 11 | parentAddr | String | Student’s parent address |
| 12 | parentContNo | String | Student’s parent contact number |
| 13 | parentAge | Integer | Student’s parent age |
| 14 | parentJob | String | Student’s parent job |
| 15 | t\_Name | String | Teacher name |
| 16 | t\_Addr | String | Teacher Address |
| 17 | t\_IcNo | Integer | Teacher identification number |
| 18 | t\_ContNo | String | Teacher contact number |
| 19 | t\_Dob | String | Teacher date of birth |
| 20 | t\_Gender | String | Teacher gender |
| 21 | t\_Age | Integer | Teacher age |
| 22 | t\_Nationality | String | Teacher nationality |
| 23 | t\_Email | String | Teacher email |
| 24 | t\_EduLvl | String | Teacher educational level |
| 25 | sName | String | Staff name |
| 26 | sAddr | String | Staff address |
| 27 | sIcNo | Integer | Staff identification number |
| 28 | sContNo | String | Staff contact number |
| 29 | sDob | String | Staff date of birth |
| 30 | sGender | String | Staff gender |
| 31 | sAge | Integer | Staff age |
| 32 | sNationality | String | Staff nationality |
| 33 | sEmail | String | Staff email |
| 34 | subjCode | String | Subject code |
| 35 | subjName | String | Subject name |
| 36 | day | String | Day for the subject take place |
| 37 | time | Integer | Time for the learning of the subject |
| 38 | place | String | Place where subject take place |
| 39 | price | Integer | Price of the subject |

1. Algorithms

This section states the purpose and describes in detail the algorithms of this class

Table 1: Detail for Class Student

|  |  |
| --- | --- |
| Class Type | database class |
| Responsibility | Save the student information into the database |
| Attributes | stdName, stdAddr, stdDob, stdContNo, stdGender, stdAge, parentName, parentAddr, parentContNo, parentAge, parentJob |
| Methods | setStdInfo(stdName, stdAddr, stdDob, stdContNo, stdGender, stdAge), setParentInfo(parentName, parentAddr, parentContNo, parentAge, parentJob), verifyStdId(stdId), deleteStd(stdId), checkStdId(stdId) |

1. setStdInfo(stdName, stdAddr, stdDob, stdContNo, stdGender, stdAge)

Table 2: Algorithm for setStdInfo

|  |  |
| --- | --- |
| Responsibility | Save student info |
| Input Parameter | stdName, stdAddr, stdDob, stdContNo, stdGender, stdAge |
| Output Parameter | - |
| Algorithm | BEGIN  Get student information from StudentManager class  Save the information  END |

1. setParentInfo(parentName, parentAddr, parentContNo, parentAge, parentJob)

Table 3: Algorithm for setParentInfo

|  |  |
| --- | --- |
| Responsibility | Save student’s parent information |
| Input Parameter | parentName, parentAddr, parentContNo, parentAge, parentJob |
| Output Parameter | - |
| Algorithm | BEGIN  Get parent information from StudentManager class  Save the information  END |

1. verifyStdId(stdId)

Table 4: Algorithm for verifyStdId

|  |  |
| --- | --- |
| Responsibility | Verify the student based on student id |
| Input Parameter | stdId |
| Output Parameter | - |
| Algorithm | BEGIN  Get the student id  Check the information based on student id  Verify the student id  END |

1. deleteStd(stdId)

Table 5: Algorithm for deleteStd

|  |  |
| --- | --- |
| Responsibility | Delete all the student information |
| Input Parameter | stdId |
| Output Parameter | - |
| Algorithm | BEGIN  Delete student information  Delete parent information  END |

1. checkStdId(stdId)

Table 6: Algorithm for checkStdId

|  |  |
| --- | --- |
| Responsibility | Check existing student |
| Input Parameter | stdId |
| Output Parameter | - |
| Algorithm | BEGIN  Get student id  Check if student exist or not  END |

### 4.1.6. Teacher

The purpose of this class is to save the teacher’s information

4.1.6.1. Class Teacher Design

This subparagraph specifies the design of Teacher

1. Input/output data elements

List of input and output data elements:

Table 1: Input and Output Date Element for Class Teacher

|  |  |
| --- | --- |
| Input | Output |
| t\_Name, t\_Addr, t\_IcNo, t\_ContNo, t\_Dob, t\_Gender, t\_Age, t\_Nationality, t\_Email, t\_EduLvl | - |

1. Local data elements

Table 1: Local Date Element for Class Teacher

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Name | Data type | Description |
| 1 | stdId | String | Student ID |
| 2 | teacherId | String | Teacher ID |
| 3 | staffId | String | Staff ID |
| 4 | stdName | String | Student name |
| 5 | stdAddr | String | Student address |
| 6 | stdDob | String | Student date of birth |
| 7 | stdContNo | String | Student contact number |
| 8 | stdGender | String | Student gender |
| 9 | stdAge | Integer | Student age |
| 10 | parentName | String | Student’s parent name |
| 11 | parentAddr | String | Student’s parent address |
| 12 | parentContNo | String | Student’s parent contact number |
| 13 | parentAge | Integer | Student’s parent age |
| 14 | parentJob | String | Student’s parent job |
| 15 | t\_Name | String | Teacher name |
| 16 | t\_Addr | String | Teacher Address |
| 17 | t\_IcNo | Integer | Teacher identification number |
| 18 | t\_ContNo | String | Teacher contact number |
| 19 | t\_Dob | String | Teacher date of birth |
| 20 | t\_Gender | String | Teacher gender |
| 21 | t\_Age | Integer | Teacher age |
| 22 | t\_Nationality | String | Teacher nationality |
| 23 | t\_Email | String | Teacher email |
| 24 | t\_EduLvl | String | Teacher educational level |
| 25 | sName | String | Staff name |
| 26 | sAddr | String | Staff address |
| 27 | sIcNo | Integer | Staff identification number |
| 28 | sContNo | String | Staff contact number |
| 29 | sDob | String | Staff date of birth |
| 30 | sGender | String | Staff gender |
| 31 | sAge | Integer | Staff age |
| 32 | sNationality | String | Staff nationality |
| 33 | sEmail | String | Staff email |
| 34 | subjCode | String | Subject code |
| 35 | subjName | String | Subject name |
| 36 | day | String | Day for the subject take place |
| 37 | time | Integer | Time for the learning of the subject |
| 38 | place | String | Place where subject take place |
| 39 | price | Integer | Price of the subject |

1. Algorithms

This section states the purpose and describes in detail the algorithms of this class

Table 1: Detail for Class Teacher

|  |  |
| --- | --- |
| Class Type | Database class |
| Responsibility | Send the teacher’s information to the database |
| Attributes | t\_Name, t\_Addr, t\_IcNo, t\_ContNo, t\_Dob, t\_Gender, t\_Age, t\_Nationality, t\_Email, t\_EduLvl |
| Methods | setTeacherInfo(t\_Name, t\_Addr, t\_IcNo, t\_ContNo, t\_Dob, t\_Gender, t\_Age, t\_Nationality, t\_Email, t\_EduLvl), checkTeacherId(teacherId), verifyTeacherId(teacherId), deleteTeacher(teacherId) |

1. setTeacherInfo(t\_Name, t\_Addr, t\_IcNo, t\_ContNo, t\_Dob, t\_Gender, t\_Age, t\_Nationality, t\_Email, t\_EduLvl)

Table 2: Algorithm for setTeacherInfo

|  |  |
| --- | --- |
| Responsibility | Save teacher’s information to the database |
| Input Parameter | t\_Name, t\_Addr, t\_IcNo, t\_ContNo, t\_Dob, t\_Gender, t\_Age, t\_Nationality, t\_Email, t\_EduLvl |
| Output Parameter | - |
| Algorithm | BEGIN  Get teacher information from TeacherManager class  Save the teacher’s information  END |

1. checkTeacherId(teacherId)

Table 3: Algorithm for checkTeacherId

|  |  |
| --- | --- |
| Responsibility | Check for existing teacher information |
| Input Parameter | teacherId |
| Output Parameter | - |
| Algorithm | BEGIN  Get the teacher id  Check the teacher id whether exist or not  END |

1. deleteTeacher(teacherId),

Table 4: Algorithm for deleteTecher

|  |  |
| --- | --- |
| Responsibility | Delete all the teacher information from the system |
| Input Parameter | teacherId |
| Output Parameter | - |
| Algorithm | BEGIN  Get teacher id from TeacherManager class  Check teacher id and information  Delete all the information  END |

1. verifyTeacherId(teacherId)

Table 5: Algorithm for verifyTecherId

|  |  |
| --- | --- |
| Responsibility | Verify whether teacher exist or not based on teacher id |
| Input Parameter | teacherId |
| Output Parameter | - |
| Algorithm | BEGIN  Get teacher id  Check the teacher id  END |

### 4.1.7. Staff

The purpose of this class is to save the staff information

4.1.7.1. Class Staff Design

This subparagraph specifies the design of Staff

1. Input/output data elements

List of input and output data elements:

Table 1: Input and Output Date Element for Class Staff

|  |  |
| --- | --- |
| Input | Output |
| sName, sAddr, sIcNo, sContNo, sDob, sGender, sAge, sNationality, sEmail | - |

1. Local data elements

Table 1: Local Date Element for Class Staff

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Name | Data type | Description |
| 1 | stdId | String | Student ID |
| 2 | teacherId | String | Teacher ID |
| 3 | staffId | String | Staff ID |
| 4 | stdName | String | Student name |
| 5 | stdAddr | String | Student address |
| 6 | stdDob | String | Student date of birth |
| 7 | stdContNo | String | Student contact number |
| 8 | stdGender | String | Student gender |
| 9 | stdAge | Integer | Student age |
| 10 | parentName | String | Student’s parent name |
| 11 | parentAddr | String | Student’s parent address |
| 12 | parentContNo | String | Student’s parent contact number |
| 13 | parentAge | Integer | Student’s parent age |
| 14 | parentJob | String | Student’s parent job |
| 15 | t\_Name | String | Teacher name |
| 16 | t\_Addr | String | Teacher Address |
| 17 | t\_IcNo | Integer | Teacher identification number |
| 18 | t\_ContNo | String | Teacher contact number |
| 19 | t\_Dob | String | Teacher date of birth |
| 20 | t\_Gender | String | Teacher gender |
| 21 | t\_Age | Integer | Teacher age |
| 22 | t\_Nationality | String | Teacher nationality |
| 23 | t\_Email | String | Teacher email |
| 24 | t\_EduLvl | String | Teacher educational level |
| 25 | sName | String | Staff name |
| 26 | sAddr | String | Staff address |
| 27 | sIcNo | Integer | Staff identification number |
| 28 | sContNo | String | Staff contact number |
| 29 | sDob | String | Staff date of birth |
| 30 | sGender | String | Staff gender |
| 31 | sAge | Integer | Staff age |
| 32 | sNationality | String | Staff nationality |
| 33 | sEmail | String | Staff email |
| 34 | subjCode | String | Subject code |
| 35 | subjName | String | Subject name |
| 36 | day | String | Day for the subject take place |
| 37 | time | Integer | Time for the learning of the subject |
| 38 | place | String | Place where subject take place |
| 39 | price | Integer | Price of the subject |

1. Algorithms

This section states the purpose and describes in detail the algorithms of this class

Table 1: Detail for Class Staff

|  |  |
| --- | --- |
| Class Type | Database class |
| Responsibility | Save the entered or edited information |
| Attributes | sName, sAddr, sIcNo, sContNo, sDob, sGender, sAge, sNationality, sEmail |
| Methods | setStaffInfo(sName, sAddr, sIcNo, sContNo, sDob, sGender, sAge, sNationality, sEmail), verifyStaffId(staffId), deleteStaff(staffId) |

1. setStaffInfo(sName, sAddr, sIcNo, sContNo, sDob, sGender, sAge, sNationality, sEmail)

Table 2:Algorithm for setStaffInfo

|  |  |
| --- | --- |
| Responsibility | Save the entered staff’s information |
| Input Parameter | sName, sAddr, sIcNo, sContNo, sDob, sGender, sAge, sNationality, sEmail |
| Output Parameter | - |
| Algorithm | BEGIN  Get staff information from StaffManager class  Save the staff information  END |

1. verifyStaffId(staffId)

Table 3: Algorithm for verifyStaffId

|  |  |
| --- | --- |
| Responsibility | Verify whether staff id is exist or not |
| Input Parameter | staffId |
| Output Parameter | - |
| Algorithm | BEGIN  Get staff id  Check whether staff id is exist or not  END |

1. deleteStaff(staffId)

Table 4: Algorithm for deleteStaff

|  |  |
| --- | --- |
| Responsibility | Delete all the information of the staff from the system |
| Input Parameter | staffId |
| Output Parameter | - |
| Algorithm | BEGIN  Get staff id from StaffManager class  Check the staff information  Delete all the staff information  END |

### 4.1.8. SubjectManager

The purpose of this class is to send the subject information to the database.

4.1.8.1. Class SubjectManager Design

This subparagraph specifies the design of SubjectManager

1. Input/output data elements

List of input and output data elements:

Table 1: Input and Output Date Element for Class SubjectManager

|  |  |
| --- | --- |
| Input | Output |
| subjCode, subjName, teacherId, day, time, place, price, stdId |  |

1. Local data elements

Table 1: Local Date Element for Class SubjectManager

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Name | Data type | Description |
| 1 | stdId | String | Student ID |
| 2 | teacherId | String | Teacher ID |
| 3 | staffId | String | Staff ID |
| 4 | stdName | String | Student name |
| 5 | stdAddr | String | Student address |
| 6 | stdDob | String | Student date of birth |
| 7 | stdContNo | String | Student contact number |
| 8 | stdGender | String | Student gender |
| 9 | stdAge | Integer | Student age |
| 10 | parentName | String | Student’s parent name |
| 11 | parentAddr | String | Student’s parent address |
| 12 | parentContNo | String | Student’s parent contact number |
| 13 | parentAge | Integer | Student’s parent age |
| 14 | parentJob | String | Student’s parent job |
| 15 | t\_Name | String | Teacher name |
| 16 | t\_Addr | String | Teacher Address |
| 17 | t\_IcNo | Integer | Teacher identification number |
| 18 | t\_ContNo | String | Teacher contact number |
| 19 | t\_Dob | String | Teacher date of birth |
| 20 | t\_Gender | String | Teacher gender |
| 21 | t\_Age | Integer | Teacher age |
| 22 | t\_Nationality | String | Teacher nationality |
| 23 | t\_Email | String | Teacher email |
| 24 | t\_EduLvl | String | Teacher educational level |
| 25 | sName | String | Staff name |
| 26 | sAddr | String | Staff address |
| 27 | sIcNo | Integer | Staff identification number |
| 28 | sContNo | String | Staff contact number |
| 29 | sDob | String | Staff date of birth |
| 30 | sGender | String | Staff gender |
| 31 | sAge | Integer | Staff age |
| 32 | sNationality | String | Staff nationality |
| 33 | sEmail | String | Staff email |
| 34 | subjCode | String | Subject code |
| 35 | subjName | String | Subject name |
| 36 | day | String | Day for the subject take place |
| 37 | time | Integer | Time for the learning of the subject |
| 38 | place | String | Place where subject take place |
| 39 | price | Integer | Price of the subject |

1. Algorithms

This section states the purpose and describes in detail the algorithms of this class

Table 1: Detail for Class SubjectManager

|  |  |
| --- | --- |
| Class Type | Manager |
| Responsibility | Send the related information of subject to the database |
| Attributes | subjCode, subjName, teacherId, day, time, place, price |
| Methods | chooseSubject(), enterSubjectInfo(subjCode, subjName, teacherId, day, time, place, price), editSubjectInfo(subjCode, subjName, teacherId, day, time, place, price, stdId) getSubjectDetails(subjCode, subjName, day, time, place, t\_Name), getSchedule() |

1. chooseSubject()

table 2: Algorithm for chooseSubject

|  |  |
| --- | --- |
| Responsibility | choose the subject for the student |
| Input Parameter | - |
| Output Parameter | - |
| Algorithm | BEGIN  Staff choose the subject from the drop down list  Staff click okay button  END |

1. enterSubjectInfo(subjCode, subjName, teacherId, day, time, place, price)

Table 3: Algorithm for enterSubjectinfo

|  |  |
| --- | --- |
| Responsibility | Send entered subject information to the database |
| Input Parameter | subjCode, subjName, teacherId, day, time, place, price |
| Output Parameter | - |
| Algorithm | BEGIN  Get the subject information from ManageSubjectInterface  Send the subject information to the Subject class  END |

1. editSubjectInfo(subjCode, subjName, teacherId, day, time, place, price)

Table 4: Algorithm for editSubjectInfo

|  |  |
| --- | --- |
| Responsibility | Send the edited subject information to the database |
| Input Parameter | subjCode, subjName, teacherId, day, time, place, price |
| Output Parameter | - |
| Algorithm | BEGIN  Get the edited subject information from ManageSubjectInterface  Send the edited subject information to the database  END |

1. getSubjectDetails(subjCode, subjName, day, time, place, t\_Name)

Table 5: Algorithm for getSubjectDetails

|  |  |
| --- | --- |
| Responsibility | Get the subject information from the database |
| Input Parameter | subjCode |
| Output Parameter | subjCode, subjName, day, time, place, t\_Name |
| Algorithm | BEGIN  Request for the subject details from the database  END |

1. getSchedule()

Table 6: Algorithm for getSchedule

|  |  |
| --- | --- |
| Responsibility | Generate the schedule for the student and teacher |
| Input Parameter | stdId, teacherId |
| Output Parameter | - |
| Algorithm | BEGIN  check the student’s registered subject  check the subject information  generate the schedule  END |

### 4.1.9. Subject

The purpose of this class is to save the subject information

4.1.9.1. Class Subject Design

This subparagraph specifies the design of Subject

1. Input/output data elements

List of input and output data elements:

Table 1: Input and Output Date Element for Class Subject Design

|  |  |
| --- | --- |
| Input | Output |
| subjCode, subjName, teacherId, day, time, place, price, stdId, t\_Name | - |

1. Local data elements

Table 1: Local Date Element for Class Subject Design

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Name | Data type | Description |
| 1 | stdId | String | Student ID |
| 2 | teacherId | String | Teacher ID |
| 3 | staffId | String | Staff ID |
| 4 | stdName | String | Student name |
| 5 | stdAddr | String | Student address |
| 6 | stdDob | String | Student date of birth |
| 7 | stdContNo | String | Student contact number |
| 8 | stdGender | String | Student gender |
| 9 | stdAge | Integer | Student age |
| 10 | parentName | String | Student’s parent name |
| 11 | parentAddr | String | Student’s parent address |
| 12 | parentContNo | String | Student’s parent contact number |
| 13 | parentAge | Integer | Student’s parent age |
| 14 | parentJob | String | Student’s parent job |
| 15 | t\_Name | String | Teacher name |
| 16 | t\_Addr | String | Teacher Address |
| 17 | t\_IcNo | Integer | Teacher identification number |
| 18 | t\_ContNo | String | Teacher contact number |
| 19 | t\_Dob | String | Teacher date of birth |
| 20 | t\_Gender | String | Teacher gender |
| 21 | t\_Age | Integer | Teacher age |
| 22 | t\_Nationality | String | Teacher nationality |
| 23 | t\_Email | String | Teacher email |
| 24 | t\_EduLvl | String | Teacher educational level |
| 25 | sName | String | Staff name |
| 26 | sAddr | String | Staff address |
| 27 | sIcNo | Integer | Staff identification number |
| 28 | sContNo | String | Staff contact number |
| 29 | sDob | String | Staff date of birth |
| 30 | sGender | String | Staff gender |
| 31 | sAge | Integer | Staff age |
| 32 | sNationality | String | Staff nationality |
| 33 | sEmail | String | Staff email |
| 34 | subjCode | String | Subject code |
| 35 | subjName | String | Subject name |
| 36 | day | String | Day for the subject take place |
| 37 | time | Integer | Time for the learning of the subject |
| 38 | place | String | Place where subject take place |
| 39 | price | Integer | Price of the subject |

1. Algorithms

This section states the purpose and describes in detail the algorithms of this class

Table 1: Detail for Class Subject Design

|  |  |
| --- | --- |
| Class Type | Database |
| Responsibility | Save the related information of subject |
| Attributes | subjCode, subjName, teacherId, day, time, place, price, stdId, t\_Name |
| Methods | verifySubject(subjCode), registerSubj(subjCode, stdId), setSubjInfo(subjCode, subjName, teacherId, day, time, place, price), getSubjDetails(subjCode, subjName, day, time , place, t\_Name), getSchedule(stdId), getSchedule(teacherId), checkSubject(stdId), checkSubject(teacherId) |

1. verifySubject(subjCode)

Table 2: Algorithm for verifySubject

|  |  |
| --- | --- |
| Responsibility | Verify the subject whether it exist or not based on subject code |
| Input Parameter | subjCode |
| Output Parameter | - |
| Algorithm | BEGIN  Get subjCode from SubjectManager class  Verify the subject code and subject infotmation  END |

1. registerSubj(subjCode, stdId)

Table 3: Algorithm for registerSubj

|  |  |
| --- | --- |
| Responsibility | Register subject for the student |
| Input Parameter | subjCode, stdId |
| Output Parameter | - |
| Algorithm | BEGIN  staff enter student id  staff choose the subject  staff click okay  END |

1. setSubjInfo(subjCode, subjName, teacherId, day, time, place, price)

Table 4: Algorithm for getSubjInfo

|  |  |
| --- | --- |
| Responsibility | Save the subject information |
| Input Parameter | subjCode, subjName, teacherId, day, time, place, price |
| Output Parameter | - |
| Algorithm | BEGIN  Get entered subject information from SubjectManager class  Save subject information  END |

1. getSubjDetails(subjCode, subjName, day, time , place, t\_Name)

Table 6: Algorithm for getSubjDetails

|  |  |
| --- | --- |
| Responsibility | Get the subject information |
| Input Parameter | subjCode |
| Output Parameter | subjCode, subjName, day, time, place, t\_Name |
| Algorithm | BEGIN  Get subjCode from SubjectManager class  Check the subject details  Return subject details  END |

1. getSchedule(stdId)

Table 7: Algorithm for getSchedule

|  |  |
| --- | --- |
| Responsibility | Generate the schedule for the student |
| Input Parameter | stdId |
| Output Parameter | - |
| Algorithm | BEGIN  check the student’s registered subject  check the subject information  generate the schedule  END |

1. getSchedule(teacherId)

Table 8: Algorithm for getSchedule

|  |  |
| --- | --- |
| Responsibility | Generate the schedule for the teacher |
| Input Parameter | teacherId |
| Output Parameter | - |
| Algorithm | BEGIN  check the teacher’s subject  check the subject information  generate the schedule  END |

1. checkSubject(stdId)

Table 9: Algorithm for checkSubject

|  |  |
| --- | --- |
| Responsibility | Check the subject taken for student |
| Input Parameter | stdId |
| Output Parameter | subjCode, subjName, day, time, place |
| Algorithm | BEGIN  Get student id from SubjectManager class  Check student’s subject  Return subject information  END |

1. checkSubject(teacherId)

Table 10: Algorithm for checkSubject

|  |  |
| --- | --- |
| Responsibility | Check the subject taken for teacher |
| Input Parameter | teacherId |
| Output Parameter | - |
| Algorithm | BEGIN  Get teacher id from SubjectManager class  Check teacher’s subject  Return subject information  END |

### 4.1.10. StudentSubjectRegistrationInterface

The purpose of this class is to register subject for the student

4.1.10.1. Class StudentSubjectRegistrationInterface Design

This subparagraph specifies the design of StudentSubjectRegistrationInterface

1. Input/output data elements

List of input and output data elements:

Table 1: Input and Output Date Element for StudentRegistrationInterface Class

|  |  |
| --- | --- |
| Input | Output |
| stdId, subjCode | - |

1. Local data elements

Table 1: Local Date Element for StudentRegistrationInterface Class

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Name | Data type | Description |
| 1 | stdId | String | Student ID |
| 2 | teacherId | String | Teacher ID |
| 3 | staffId | String | Staff ID |
| 4 | stdName | String | Student name |
| 5 | stdAddr | String | Student address |
| 6 | stdDob | String | Student date of birth |
| 7 | stdContNo | String | Student contact number |
| 8 | stdGender | String | Student gender |
| 9 | stdAge | Integer | Student age |
| 10 | parentName | String | Student’s parent name |
| 11 | parentAddr | String | Student’s parent address |
| 12 | parentContNo | String | Student’s parent contact number |
| 13 | parentAge | Integer | Student’s parent age |
| 14 | parentJob | String | Student’s parent job |
| 15 | t\_Name | String | Teacher name |
| 16 | t\_Addr | String | Teacher Address |
| 17 | t\_IcNo | Integer | Teacher identification number |
| 18 | t\_ContNo | String | Teacher contact number |
| 19 | t\_Dob | String | Teacher date of birth |
| 20 | t\_Gender | String | Teacher gender |
| 21 | t\_Age | Integer | Teacher age |
| 22 | t\_Nationality | String | Teacher nationality |
| 23 | t\_Email | String | Teacher email |
| 24 | t\_EduLvl | String | Teacher educational level |
| 25 | sName | String | Staff name |
| 26 | sAddr | String | Staff address |
| 27 | sIcNo | Integer | Staff identification number |
| 28 | sContNo | String | Staff contact number |
| 29 | sDob | String | Staff date of birth |
| 30 | sGender | String | Staff gender |
| 31 | sAge | Integer | Staff age |
| 32 | sNationality | String | Staff nationality |
| 33 | sEmail | String | Staff email |
| 34 | subjCode | String | Subject code |
| 35 | subjName | String | Subject name |
| 36 | day | String | Day for the subject take place |
| 37 | time | Integer | Time for the learning of the subject |
| 38 | place | String | Place where subject take place |
| 39 | price | Integer | Price of the subject |

1. Algorithms

This section states the purpose and describes in detail the algorithms of this class

Table 1:Detail for StudentRegistrationInterface Class

|  |  |
| --- | --- |
| Class Type | Model class |
| Responsibility | Register student’s subject |
| Attributes | stdId, subjCode |
| Methods | enterStdId(stdId), chooseSubject() |

1. enterStdId(stdId)

Table 2: Algorithm for enterStdId

|  |  |
| --- | --- |
| Responsibility | Enter student id to register the subject |
| Input Parameter | stdId |
| Output Parameter | - |
| Algorithm | BEGIN  Staff enter student id  Staff click okay button  END |

1. chooseSubject()

Table 3: Algorithm for chooseSubject

|  |  |
| --- | --- |
| Responsibility | Choose the subject for the student |
| Input Parameter | - |
| Output Parameter | - |
| Algorithm | BEGIN  Staff choose the subject from the drop down list  Staff click okay button  END |

### 4.1.11. ManageSubjectInterface

The purpose of this class is to allow staff to manage the subject, add subject, edit subject and view subject

4.1.11.1. Class ManageSubjectInterface Design

This subparagraph specifies the design of ManageSubjectInterface

1. Input/output data elements

List of input and output data elements:

Table 1: Input and Output Date Element for ManageSubjectInterface Class

|  |  |
| --- | --- |
| Input | Output |
| subjCode, subjName, teacherId, day, time, place, price | - |

1. Local data elements

Table 1: Local Date Element for ManageSubjectInterface Class

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Name | Data type | Description |
| 1 | stdId | String | Student ID |
| 2 | teacherId | String | Teacher ID |
| 3 | staffId | String | Staff ID |
| 4 | stdName | String | Student name |
| 5 | stdAddr | String | Student address |
| 6 | stdDob | String | Student date of birth |
| 7 | stdContNo | String | Student contact number |
| 8 | stdGender | String | Student gender |
| 9 | stdAge | Integer | Student age |
| 10 | parentName | String | Student’s parent name |
| 11 | parentAddr | String | Student’s parent address |
| 12 | parentContNo | String | Student’s parent contact number |
| 13 | parentAge | Integer | Student’s parent age |
| 14 | parentJob | String | Student’s parent job |
| 15 | t\_Name | String | Teacher name |
| 16 | t\_Addr | String | Teacher Address |
| 17 | t\_IcNo | Integer | Teacher identification number |
| 18 | t\_ContNo | String | Teacher contact number |
| 19 | t\_Dob | String | Teacher date of birth |
| 20 | t\_Gender | String | Teacher gender |
| 21 | t\_Age | Integer | Teacher age |
| 22 | t\_Nationality | String | Teacher nationality |
| 23 | t\_Email | String | Teacher email |
| 24 | t\_EduLvl | String | Teacher educational level |
| 25 | sName | String | Staff name |
| 26 | sAddr | String | Staff address |
| 27 | sIcNo | Integer | Staff identification number |
| 28 | sContNo | String | Staff contact number |
| 29 | sDob | String | Staff date of birth |
| 30 | sGender | String | Staff gender |
| 31 | sAge | Integer | Staff age |
| 32 | sNationality | String | Staff nationality |
| 33 | sEmail | String | Staff email |
| 34 | subjCode | String | Subject code |
| 35 | subjName | String | Subject name |
| 36 | day | String | Day for the subject take place |
| 37 | time | Integer | Time for the learning of the subject |
| 38 | place | String | Place where subject take place |
| 39 | price | Integer | Price of the subject |

1. Algorithms

This section states the purpose and describes in detail the algorithms of this class

Table 1: Detail for ManageSubjectInterface Class

|  |  |
| --- | --- |
| Class Type | Model class |
| Responsibility | Manage the subject as to add subject, edit subject or view subject |
| Attributes | subjCode, subjName, teacherId, day, time, place, price |
| Methods | addSubject(subjCode, subjName, teacherId, day, time, place, price), editSubject(subjCode, subjName, teacherId, day, time, place, price), viewSubject() |

1. addSubject(subjCode, subjName, teacherId, day, time, place, price)

Table 2: Algorithm for addSubject

|  |  |
| --- | --- |
| Responsibility | Add subject information |
| Input Parameter | subjCode, subjName, teacherId, day, time, place, price |
| Output Parameter | - |
| Algorithm | BEGIN  Staff choose add subject  Staff enter subject information  Staff click save button  END |

1. editSubject(subjCode, subjName, teacherId, day, time, place, price)

Table 3: Algorithm for editSubject

|  |  |
| --- | --- |
| Responsibility | Edit subject information |
| Input Parameter | subjCode, subjName, teacherId, day, time, place, price |
| Output Parameter | - |
| Algorithm | BEGIN  Staff choose edit subject  Staff edit subject information  Staff click save button  END |

1. viewSubject()

Table 4: Algorithm for ciewSubject

|  |  |
| --- | --- |
| Responsibility | View subject information |
| Input Parameter | - |
| Output Parameter | - |
| Algorithm | BEGIN  Staff choose view subject  END |

### 4.1.12. ViewScheduleInterface

The purpose of this class is to view schedule of the teacher and student

4.1.12.1. Class ViewScheduleInterface Design

This subparagraph specifies the design of ViewScheduleInterface

1. Input/output data elements

List of input and output data elements:

Table 1: Input and Output Date Element for ViewScheduleInterface Class

|  |  |
| --- | --- |
| Input | Output |
| stdId, teacherId | subjCode, subjName, t\_Name, day, time, place |

1. Local data elements

Table 1: Local Date Element for ViewScheduleInterface Class

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Name | Data type | Description |
| 1 | stdId | String | Student ID |
| 2 | teacherId | String | Teacher ID |
| 3 | staffId | String | Staff ID |
| 4 | stdName | String | Student name |
| 5 | stdAddr | String | Student address |
| 6 | stdDob | String | Student date of birth |
| 7 | stdContNo | String | Student contact number |
| 8 | stdGender | String | Student gender |
| 9 | stdAge | Integer | Student age |
| 10 | parentName | String | Student’s parent name |
| 11 | parentAddr | String | Student’s parent address |
| 12 | parentContNo | String | Student’s parent contact number |
| 13 | parentAge | Integer | Student’s parent age |
| 14 | parentJob | String | Student’s parent job |
| 15 | t\_Name | String | Teacher name |
| 16 | t\_Addr | String | Teacher Address |
| 17 | t\_IcNo | Integer | Teacher identification number |
| 18 | t\_ContNo | String | Teacher contact number |
| 19 | t\_Dob | String | Teacher date of birth |
| 20 | t\_Gender | String | Teacher gender |
| 21 | t\_Age | Integer | Teacher age |
| 22 | t\_Nationality | String | Teacher nationality |
| 23 | t\_Email | String | Teacher email |
| 24 | t\_EduLvl | String | Teacher educational level |
| 25 | sName | String | Staff name |
| 26 | sAddr | String | Staff address |
| 27 | sIcNo | Integer | Staff identification number |
| 28 | sContNo | String | Staff contact number |
| 29 | sDob | String | Staff date of birth |
| 30 | sGender | String | Staff gender |
| 31 | sAge | Integer | Staff age |
| 32 | sNationality | String | Staff nationality |
| 33 | sEmail | String | Staff email |
| 34 | subjCode | String | Subject code |
| 35 | subjName | String | Subject name |
| 36 | day | String | Day for the subject take place |
| 37 | time | Integer | Time for the learning of the subject |
| 38 | place | String | Place where subject take place |
| 39 | price | Integer | Price of the subject |

1. Algorithms

This section states the purpose and describes in detail the algorithms of this class

Table 1: Detail for ViewScheduleInterface Class

|  |  |
| --- | --- |
| Class Type | Model |
| Responsibility | View schedule for the student and teacher |
| Attributes | stdId, teacherId |
| Methods | viewSchedule(stdId), viewSchedule(teacherId) |

1. viewSchedule(stdId)

Table 2: Algorithm for viewSchedule

|  |  |
| --- | --- |
| Responsibility | View schedule for the student based on student id |
| Input Parameter | stdId |
| Output Parameter | subjCode, subjName, t\_Name, day, time, place |
| Algorithm | BEGIN  Student click view schedule  END |

1. viewSchedule(teacherId)

Table 3: Algorithm for viewSchedule

|  |  |
| --- | --- |
| Responsibility | View schedule for the teacher based on teacher id |
| Input Parameter | tecaherId |
| Output Parameter | subjCode, subjName, day, time, place |
| Algorithm | BEGIN  Teacher click view schedule  END |

## 4.2 Teaching Material Interface

Act as the main page for teaching material function. It allows the user to navigate to note, exercise, online quiz and e\_Books.

### 4.2.1 NoteManagementInterface Class Design

Ths subparagraph specifies the design of NoteManagementInterface.

a) Input/output dara elements

List of input and output data elements:

Table 1: Input and Output Date Element for NoteManagementInterace Class

|  |  |
| --- | --- |
| Input | Output |
| noteSubj, noteChap, TeachID, StdID | None |

1. Local data elements

Table 1: Local Date Element for NoteManagementInterface Class

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Name | Data type | Description |
| 1 | noteSubj | String | Hold subject ID for each note |
| 2 | noteChap | String | Hold chapter ID for each note |
| 3 | TeachID | String | Hold the teacher ID values |
| 4 | StdID | String | Hold the student ID values |

c) Algorithms

This section states the purpose and describe in detail the algorithm in this class.

Table 1: Detail for NoteManagementInterface Class

|  |  |
| --- | --- |
| Responsibility | Determine whether to go to add note page, update note page, or print note page. |
| Attributes | subjM : SubjectManager  NoteM : NoteManager |
| Methods | addnoteInterface (): void  updatenoteInterface (): void  deletenoteInterface (): void  savenoteInterface (): void  downloadnoteInterface (): void  printnoteInterface (): void |

i.addnoteInterface (): void

Table 2: Algorithm for addnoteInterface

|  |  |
| --- | --- |
| Responsibility | Enable to navigate Add Note Interface |
| Input Parameter | noteSubj, noteChap, TeachID |
| Output Parameter | void |
| Algorithm | BEGIN  Select Add Note button  Go to Add Note Page  END |

1. updatenoteInterface (): void

Table 3: Algorithm for updatenoteInterface

|  |  |
| --- | --- |
| Responsibility | Enable to navigate Update Note Interface |
| Input Parameter | noteSubj, noteChap, TeachID |
| Output Parameter | Void |
| Algorithm | BEGIN  Select Update Note button  Go to Update Note Page  END |

1. deletenoteInterface (): void

Table 4: Algorithm for deletenoteInterface

|  |  |
| --- | --- |
| Responsibility | Enable to navigate Delete Note Interface |
| Input Parameter | noteSubj, noteChap, TeachID |
| Output Parameter | Void |
| Algorithm | BEGIN  Select Delete Note button  Go to Delete Note Page  END |

1. savenoteInterface (): void

Table 6: Algorithm for savenoteInterface

|  |  |
| --- | --- |
| Responsibility | Enable to navigate Save Note Interface |
| Input Parameter | noteSubj, noteChap, TeachID, StdID |
| Output Parameter | Void |
| Algorithm | BEGIN  Select Save Note button  Go to Save Note Page  END |

1. downloadnoteInterface (): void

Table 7: Algorithm for downloadnoteInterface

|  |  |
| --- | --- |
| Responsibility | Enable to navigate Download Note Interface |
| Input Parameter | noteSubj, noteChap, StdID |
| Output Parameter | String |
| Algorithm | BEGIN  Select Download Note button  Go to Download Note Page  END |

1. printnoteInterface (): void

Table 8: Algorithm for printnoteInterface

|  |  |
| --- | --- |
| Responsibility | Enable to navigate Print Note Interface |
| Input Parameter | noteSubj, noteChap, StdID |
| Output Parameter | Void |
| Algorithm | BEGIN  Select Print Note button  Go to Print Note Page  END |

### 4.2.2 ExerciseManagementInterface Class Design

This subparagraph specifies the design of Exercise Management Interface

a) Input/output data elements.

List of input and output data elements

Table 1: Input and Output Date Element for ExerciseManagementInterface Class

|  |  |
| --- | --- |
| Input | Output |
| excSubj, excChap, TeachID, StdID | None |

1. Local data elements

Table 1: Local Date Element for ExerciseManagementInterface Class

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Name | Data type | Description |
| 1 | excSubj | String | Hold subject ID for each exercise |
| 2 | excChap | String | Hold chapter ID for each exercise |
| 3 | TeachID | String | Hold the teacher ID values |
| 4 | StdID | String | Hold the student ID values |

c) Algorithms

This section states the purpose and describe in detail the algorithm of this class.

Table 1: Detail for ExerciseManagementInterface Class

|  |  |
| --- | --- |
| Responsibility | Provide exercise interface |
| Attributes | subjM : SubjectManager  excM : ExerciseManager |
| Methods | downloadexerciseInterface(excSubj : string,excChap : string,StdID : string) : void  printexerciseInterface(excSubj : string,excChap : string, StdID : string) : void  saveexerciseInterface (excSubj : string,excChap : string,TeachID : string, StdID : string) : void  addexerciseInterface (excSubj : string,excChap : string,TeachID : string) : void  updateexerciseInterface (excSubj : string,excChap : string,TeachID : string): void  deleteexerciseInterface (excSubj : string,excChap : string,TeachID : string) : void |

i. downloaddexerciseInterface (): void

Table 2 : Algorithm for downloadexerciseInterfcae

|  |  |
| --- | --- |
| Responsibility | Enable to navigate Download Exercise Interfaece |
| Input Parameter | excSubj : string,excChap : string,StdID : string |
| Output Parameter | String |
| Algorithm | BEGIN  Select Download button  Go to Download Exercise Page  END |

1. printexerciseInterface (): void

Table 3: Algorithm for printexerciseInterfce

|  |  |
| --- | --- |
| Responsibility | Enable to navigate Print Exercise Interface |
| Input Parameter | excSubj : string,excChap : string,StdID : string |
| Output Parameter | Void |
| Algorithm | BEGIN  Select Print Exercise button  Go to Print Exercise page  END |

iii) saveexersiceInterface (): void

Table 4: Algorithm for saveexerciseInterfce

|  |  |
| --- | --- |
| Responsibility | Enable to navigate save Exercise Interface |
| Input Parameter | excSubj : string,excChap : string,TeachID : string, StdID : string |
| Output Parameter | Void |
| Algorithm | BEGIN  Select Save Exercise button  Go to Save Exercise page  END |

iv) addexerciseInterface (): void

Table 6: Algorithm for addexerciseInterface

|  |  |
| --- | --- |
| Responsibility | Enable to navigate Add Exercise Interface |
| Input Parameter | excSubj : string,excChap : string,TeachID : string |
| Output Parameter | Void |
| Algorithm | BEGIN  Select Add Exercise button  Go to Add Exercise page  END |

1. updateexerciseInterface (): void

Table 6: Algorithm for updateexerciseInterface

|  |  |
| --- | --- |
| Responsibility | Enable to navigate Update Exercise Interface |
| Input Parameter | excSubj : string,excChap : string,TeachID : string |
| Output Parameter | Void |
| Algorithm | BEGIN  Select Update Exercise button  Go to Update Exercise page  END |

1. deleteexerciseInterface ():void

Table 7: Algorithm for deleteexerciseInterfce

|  |  |
| --- | --- |
| Responsibility | Enable to navigate Delete Exercise Interface |
| Input Parameter | excSubj : string,excChap : string,TeachID : string |
| Output Parameter | Void |
| Algorithm | BEGIN  Select Delete Exercise button  Go to Delete Exercise page  END |

### 4.2.3 OnlineQuizManagementInterface Class Design

This subparagraph specifies the design of Online Quiz Management Interface

a) Input/output data elements

Table 1: Input and Output Date Element for OnlineQuizManagementInterface Class

|  |  |
| --- | --- |
| Input | Output |
| quizSubj, quizChap,TeachID,StdID | None |

1. Local Data Elements

Table 1: Local Date Element for OnlineQuizManagementInterface Class

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Name | Data type | Description |
| 1 | quizSubj | String | Hold subject ID for each quizzes |
| 2 | quizChap | String | Hold chapter ID for each quizzes |
| 3 | TeachID | String | Hold the teacher ID values |
| 4 | StdID | String | Hold the student ID values |

c) Algorithms

This section states the purpose and describes in detail the algorithms of this class.

Table 1: Detail for OnlineQuizManagementInterface Class

|  |  |
| --- | --- |
| Responsibility | Provide online quiz interface |
| Attributes | subjM : SubjectManager  excM : ExerciseManager |
| Methods | uploadquizInterface (quizSubj : string,quizChap : string,TeachID : string): void  openquizInterface (quizSubj : string,quizChap : string,TeachID : string): void  deletequizInterface (quizSubj : string,quizChap : string,TeachID : string): void  attendquizInterface (quizSubj : string,quizChap : string,StdID : string): void  submitquizInterface (quizSubj : string,quizChap : string,StdID : string): void |

i. uploadquizInterface (): void

Table 2 : Algorithm for uploadquizInterface

|  |  |
| --- | --- |
| Responsibility | Enable to navigate Upload Quiz Interface |
| Input Parameter | quizSubj, quizChap, TeachID |
| Output Parameter | Void |
| Algorithm | BEGIN  Select Upload Quiz button  Go to Upload Quiz page  END |

1. openquizInterface (): void

Table 3: Algorithm for openquizInterface

|  |  |
| --- | --- |
| Responsibility | Enable to navigate Open Quiz Interface |
| Input Parameter | quizSubj, quizChap, TeachID |
| Output Parameter | Void |
| Algorithm | BEGIN  Select Open Quiz button  Go to Open Quiz page  END |

1. deletequizInterface (): void

Table 4: Algorithm for deletequizInterface

|  |  |
| --- | --- |
| Responsibility | Enable to navigate Delete Quiz Interface |
| Input Parameter | quizSubj, quizChap, TeachID |
| Output Parameter | Void |
| Algorithm | BEGIN  Select Delete Quiz button  Go to Delete Quiz page  END |

iv.attendquizInterface (): void

Table 5: Algorithm for attendquizInterfce

|  |  |
| --- | --- |
| Responsibility | Enable to navigate Attend Quiz Interface |
| Input Parameter | quizSubj, quizChap, StdID |
| Output Parameter | Void |
| Algorithm | BEGIN  Select Attend Quiz button  Go to Attend Quiz page  END |

1. submitquizInterface (): void

Table 6 : Algorithm for submitquizInterface

|  |  |
| --- | --- |
| Responsibility | Enable to navigate Submit Quiz Interface |
| Input Parameter | quizSubj, quizChap, TeachID |
| Output Parameter | Void |
| Algorithm | BEGIN  Select Submit Quiz button  Go to Submit Quiz page  END |

### 4.2.4 E\_BooksInterface Class Design

This subparagraph specifies the design of E\_books Management Interface

a) Input/output data elements

List of input output data elements

Table 1: Input and Output Date Element for E\_BooksInterface Clas

|  |  |
| --- | --- |
| Input | Output |
| e\_bSubj, e\_bChap, TeachID, StdID | None |

1. Local Data Elements

Table 1: Local Date Element for E\_BooksInterface Clas

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Name | Data type | Description |
| 1 | e\_bSubj | String | Hold subject ID for each e\_books |
| 2 | E\_bChap | String | Hold chapter ID for each e\_books |
| 3 | TeachID | String | Hold the teacher ID values |
| 4 | StdID | String | Hold the student ID values |

c) Algorithms

This section states the purpose and describe in detail the algorithms of his class

Table 1: Detail for E\_BooksInterface Clas

|  |  |
| --- | --- |
| Responsibility | Provide e\_books interface |
| Attributes | subjM : SubjectManager  excM : ExerciseManager |
| Methods | uploade\_booksInterface (e\_bSubj : string,e\_bChap : string,TeachID : string): void  adde\_booksInterface (e\_bSubj : string,e\_bChap : string,TeachID : string): void  delete\_books (e\_bSubj : string,e\_bChap : string,TeachID : string): void  downloade\_books (e\_bSubj : string,e\_bChap : string,StdID : string): void  previewe\_books (e\_bSubj : string,e\_bChap : string,StdID : string): void |

i. uploade\_booksInterface (): void

Table 2: Algorithm for uploaded\_booksInterface

|  |  |
| --- | --- |
| Responsibility | Enable to navigate Upload E\_Books Interface |
| Input Parameter | e\_bSubj, e\_bChap, TeachID |
| Output Parameter | Void |
| Algorithm | BEGIN  Select Upload button  Go to Upload E\_Books page  END |

1. adde\_booksInterface (): void

Table 3: Algorithm for adde\_booksInterface

|  |  |
| --- | --- |
| Responsibility | Enable to navigate Add E\_Books Interface |
| Input Parameter | e\_bSubj, e\_bChap, TeachID |
| Output Parameter | Void |
| Algorithm | BEGIN  Select Add button  Go to Add E\_Books page  END |

1. delete\_booksInterface (): void

Table 4: Algorithm for delete\_booksInterface

|  |  |
| --- | --- |
| Responsibility | Enable to navigate Delete E\_Books Interface |
| Input Parameter | e\_bSubj, e\_bChap, TeachID |
| Output Parameter | Void |
| Algorithm | BEGIN  Select Delete button  Go to Delete E\_Books page  END |

1. downloade\_booksInterface (): void

Table 5: Algorithm for download\_booksInterface

|  |  |
| --- | --- |
| Responsibility | Enable to navigate download E\_Books Interface |
| Input Parameter | e\_bSubj, e\_bChap, StdID |
| Output Parameter | String |
| Algorithm | BEGIN  Select download button  Go to download E\_Books page  END |

1. previewe\_BooksInterface (): void

Table 6: Algorithm for preview\_BooksInterface

|  |  |
| --- | --- |
| Responsibility | Enable to navigate Preview E\_Books Interface |
| Input Parameter | e\_bSubj, e\_bChap, TeachID |
| Output Parameter | Void |
| Algorithm | BEGIN  1. Select Preview button  2. Go to Preview E\_Books page  END |

## 4.3 Financial Management

The followings explain the details of Financial Management.

### 4.3.1 FinancialManagementView

FinancialManagementView act as the main page for financial function. It allows the user to navigate to tuition payment, salary payment or report page.

**4.3.1 FinancialManagementView Class Design**

This subparagraph specifies the design of FinancialManagementView

1. Input/output data elements

List of input and output data elements:

Table 1: Input and Output Date Element for FinancialManagementView Class

|  |  |
| --- | --- |
| Input | Output |
| choose: int | - |

1. Local data elements

Table 1: Local Date Element for FinancialManagementView Class

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Name | Data type | Description |
| 1 | choose | int | Hold the value of choose |

1. Algorithms

This section states the purpose and describes in detail the algorithms of this class

Table 1: Detail for FinancialManagementView Class

|  |  |
| --- | --- |
| Responsibility | Act as the main page for financial function. It allows the user to navigate to tuition payment, salary payment or report page. |
| Attributes | choose: int |
| Methods | goTPayment() : void  goSPayment() : void  goMReport() : void |

1. goTPayment()

Table 2: Algorithm for goTPayment

|  |  |
| --- | --- |
| Responsibility | Enable to navigate Tuition Payment Page |
| Input Parameter | None |
| Output Parameter | None |
| Algorithm | BEGIN  Select Tuition Payment  Go to Tuition Payment page  END |

1. goSPayment()

Table 3: Algorithm for goSPayment

|  |  |
| --- | --- |
| Responsibility | Enable to navigate to Salary Payment page |
| Input Parameter | None |
| Output Parameter | None |
| Algorithm | BEGIN  Select Salary Payment  Go to Salary Payment page  END |

1. goMReport()

Table 4: Algorithm for goMReport

|  |  |
| --- | --- |
| Responsibility | Enable to navigate to Report page |
| Input Parameter | None |
| Output Parameter | None |
| Algorithm | BEGIN  Select Report  Go to Report Page  END |

### 4.3.2 TuitionPaymentView Class Design

This subparagraph specifies the design of TuitionPaymentView

1. Input/output data elements

List of input and output data elements:

Table 1: Input and Output Date Element for TuitionPaymentView Class

|  |  |
| --- | --- |
| Input | Output |
| stdId, subjCode, subjPrice, tuitionFee | None |

1. Local data elements
2. Table 1: Local Date Element for TuitionPaymentViewClass

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Name | Data type | Description |
| 1 | stdId | String | Hold student ID values |
| 2 | subjCode | String | Hold subject code values |
| 3 | subjPrice | double | Hold the price for each subject |
| 4 | tuitionFee | double | Hold tuition fee amount of each student |

1. Algorithms

This section states the purpose and describes in detail the algorithms of this class

Tabel 1: Detail for TuitionPaymentView Class

|  |  |
| --- | --- |
| Responsibility | Enable the user to either insert payment of tuition fee, view tuition fee or calculate tuition fee. |
| Attributes | stdId, subjCode, subjPrice, tuitionFee |
| Methods | getStdSubj (stdId: String): String  getSubjPrice (stdId: String): double  calcFee (stdId: String, subjCode: String, subjPrice: double): void  getFee(stdId: String): double  payFee(stdId: String,date: int): void |

1. getStdSubj(stdId : String)

Table 2: Algorithm for getStdSubj

|  |  |
| --- | --- |
| Responsibility | Get the subject taken by the student based on student ID. |
| Input Parameter | stdId : String |
| Output Parameter | subjCode : String |
| Algorithm | BEGIN  Enter student ID  Display all the subject taken by the student  END |

1. getSubjPrice(stdId : String)

Table 3: Algorithm for getSubjPrice

|  |  |
| --- | --- |
| Responsibility | Get price for each subject |
| Input Parameter | None |
| Output Parameter | stdId : String |
| Algorithm | BEGIN  Enter student ID  Return all the subject taken by the student with price for each subject.  END |

1. calcFee(stdId : String,subjCode : String,subjPrice : double)

Table 4: Algorithm for calcFee

|  |  |
| --- | --- |
| Responsibility | Calculate tuition fee of the student |
| Input Parameter | stdId : String  subjCode : String  subjPrice : double |
| Output Parameter | tuitionFee : double |
| Algorithm | BEGIN  Insert stdId, subjCode, SubjPrice  END |

1. getFee(stdId : String)

Table 5: Algorithm for getFee

|  |  |
| --- | --- |
| Responsibility | Retrieve calculated tuition fees |
| Input Parameter | stdId : String |
| Output Parameter | tuitionFee : double |
| Algorithm | BEGIN  Insert stdId  Return tuitonFee  END |

1. payFee(stdId : String,date : int) : void

Table 6: Algorithm for payFee

|  |  |
| --- | --- |
| Responsibility | Insert tuition fee payment made. |
| Input Parameter | stdId : String  date : int |
| Output Parameter | None |
| Algorithm | BEGIN  Insert stdId, date  Store payment  END |

### 4.3.3 TuitionFeeManager Class Design

This subparagraph specifies the design of TuitionFeeManager

1. Input/output data elements

List of input and output data elements:

Table 1: Input and Output Date Element for TuitionFeeManager Class

|  |  |
| --- | --- |
| Input | Output |
| stdId, subjCode, date, tuitionFee | None |

1. Local data elements

Table 1: Local Date Element for TuitionFeeManager Class

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Name | Data type | Description |
| 1 | stdId | String | Hold student ID values |
| 2 | subjCode | String | Hold subject code values |
| 3 | date | int | Hold the price for each subject |
| 4 | tuitionFee | double | Hold tuition fee amount of each student |

1. Algorithms

This section states the purpose and describes in detail the algorithms of this class

Table 1: Detail for TuitionFeeManager Class

|  |  |
| --- | --- |
| Responsibility | Do all the operations involving tuition fee calculations, inserting tuition fee payment and viewing tuition fee. |
| Attributes | tFee : TuitionFee |
| Methods | calcFee(subjID : String  subjID : String,subjPrice : double) : void  getFee(stdId : String) : double  payFee(stdId : String,date : int) : void |

1. calcFee(stdId: String, subjCode: String, subjPrice: double)

Table 2: Algorithm for calcFee

|  |  |
| --- | --- |
| Responsibility | Calculate tuition fee |
| Input Parameter | stdId : String  subjCode: String  subjPrice: double |
| Output Parameter | tuitionFee : double |
| Algorithm | BEGIN  Insert stdId, subjCode, subjPrice  Calculate tuition fee  END |

1. getFee(stdId : String)

Table 3: Algorithm for getFee

|  |  |
| --- | --- |
| Responsibility | Retrieve tuition fee amount. |
| Input Parameter | stdId : String |
| Output Parameter | tuitionFee : double |
| Algorithm | BEGIN  Insert stdID  Get tuitionfee  END |

1. payFee(stdId : String,date : int)

Table 4: Algorithm for payFee

|  |  |
| --- | --- |
| Responsibility | Process tuition fee payment |
| Input Parameter | stdId : String  date : int |
| Output Parameter | None |
| Algorithm | BEGIN  Insert stdId, date  Store payment  END |

### 4.3.4 TuitionFee Class Design

This subparagraph specifies the design of TuitionFee

1. Input/output data elements

List of input and output data elements:

Table 1: Input and Output Data Element for TuitionFess Class

|  |  |
| --- | --- |
| Input | Output |
| stdId, subjCode, date | tuitionFee |

1. Local data elements

Table 1: Local Data Element for TuitionFee Class

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Name | Data type | Description |
| 1 | stdId | String | Hold student ID values |
| 2 | subjCode | String | Hold subject code values |
| 3 | date | int | Hold the price for each subject |
| 4 | tuitionFee | double | Hold tuition fee amount of each student |

1. Algorithms

This section states the purpose and describes in detail the algorithms of this class

Table 1: Detail for TuitionFee Class

|  |  |
| --- | --- |
| Responsibility | Store tuition fee of all the students and tuition fee payment made. It also enable to retrieve tuition fee amount. |
| Attributes | rManager : ReceiptManager |
| Methods | saveFee(stdId: String, subjCode: String, subjPrice : double) : void  getFee(stdId : String) : double  savePayment(stdId : String,date : int) : void |

1. saveFee(stdId : String, subjCode : String,subjPrice : double)

Table 2: Algorithm for saveFee

|  |  |
| --- | --- |
| Responsibility | Store tuition fee of all the students |
| Input Parameter | stdId : String  subjCode : String,subjPrice : double |
| Output Parameter | None |
| Algorithm | BEGIN  Insert stdId, subjCode and date  Store payment  END |

1. getFee(stdId : String)

Table 3: Algorithm for getFee

|  |  |
| --- | --- |
| Responsibility | Retrieve tuition fee based on student ID |
| Input Parameter | stdId : String |
| Output Parameter | tuitionFee : double |
| Algorithm | BEGIN  Insert stdID  Get tuitionfee  END |

1. savePayment(stdId : String,date : int)

Table 4: Algorithm for savePayment

|  |  |
| --- | --- |
| Responsibility | Store tuition fee payment made |
| Input Parameter | stdId : String  date : int |
| Output Parameter | None |
| Algorithm | BEGIN  Insert stdId and date  Store payment  END |

### 4.3.5 SalaryPaymentView Class Design

This subparagraph specifies the design of SalaryPaymentView

1. Input/output data elements

List of input and output data elements:

Table 1: Input and Output Data for SalaryPaymentView Class

|  |  |
| --- | --- |
| Input | Output |
| teacherId, date, salaryAmount | None |

1. Local data elements

Table 1: Local data element for SalaryPaymentView Class

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Name | Data type | Description |
| 1 | teacherId | String | Hold teacher ID values |
| 2 | salaryAmount | double | Hold salary values |
| 3 | date | int | Hold the price for each subject |

1. Algorithms

This section states the purpose and describes in detail the algorithms of this class

Table 1: Detail for SalaryPaymentView Class

|  |  |
| --- | --- |
| Responsibility | Enable the user to either insert payment of salary, view salary or calculate salary. |
| Attributes | subjManager : SubjectManager |
| Methods | getSubjTeach(teacherId : String) : String  getSalary(teacherId : String) : double  paySalary(teacherId : String,date : int) : void  getSubjSalary() : double |

1. getSubjTeach(teacherId : String)hm

Table 2: Algorithm for getSubjTeach

|  |  |
| --- | --- |
| Responsibility | Use to calculate salary based on subject taught. |
| Input Parameter | teacherId: String |
| Output Parameter | subjCode: String |
| Algorithm | BEGIN  Insert teacherId  Return subject price  END |

1. getSalary(teacherId : String)

Table 3: Algorithm for getSalary

|  |  |
| --- | --- |
| Responsibility | Retrieve salary based on teacher ID |
| Input Parameter | teacherId : String |
| Output Parameter | teachSalary: double |
| Algorithm | BEGIN  Insert teacherId  Return salary  END |

1. paySalary(teacherId : String,date : int)

Table 4: Algorithm for paySalary

|  |  |
| --- | --- |
| Responsibility | Process salary payment |
| Input Parameter | teacherId : String  date : int |
| Output Parameter | None |
| Algorithm | BEGIN  Insert teacherId and date  Store payment  END |

1. getSubjSalary()

Table 5: Algorithm for getSubjSalary

|  |  |
| --- | --- |
| Responsibility | Retrieve salary for each subject |
| Input Parameter | None |
| Output Parameter | subjSalary: double |
| Algorithm | BEGIN  Get subjSalary  Return subjSalary  END |

### 4.3.6 SalaryManager Class Design

This subparagraph specifies the design of SalaryManager

1. Input/output data elements

List of input and output data elements:

Table 1: Input and Output Data for SalaryManager Class

|  |  |
| --- | --- |
| Input | Output |
| teacherId: String | - |

1. Local data elements

Table 1: Local data elements for SalaryManager Class

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Name | Data type | Description |
| 1 | teacherId | String | Teacher Id |
| 2 | date | int | Date |
| 3 | subjCode | String | Subject code |
| 4 | subjSalary | double | Subject salary |
| 5 | teachSalary: | double | teacherSalary |

1. Algorithms

This section states the purpose and describes in detail the algorithms of this class

Table 1: Detail for SalaryManager Class

|  |  |
| --- | --- |
| Class Type | Controller class |
| Responsibility |  |
| Attributes | salaries : Salary |
| Methods | getSalary(teacherId : String) : double  paySalary(teacherId : String,date : int) : void  calcSalary(teacherId : String,subjCode : String,subjPrice : double) : void |

1. getSalary(teacherId : String)

Table 2: Algorithm for getSalary

|  |  |
| --- | --- |
| Responsibility | Retrieve salary based on teacher ID |
| Input Parameter | teacherId : String |
| Output Parameter | teachSalary: double |
| Algorithm | BEGIN  END |

1. paySalary(teacherId : String,date : int)

Table 3: Algorithm for paySalary

|  |  |
| --- | --- |
| Responsibility | Process salary payment |
| Input Parameter | teacherId : String,date : int |
| Output Parameter | None |
| Algorithm | BEGIN  Insert teacherId, date  Update salary payment  END |

1. calcSalary(teacherId : String,subjCode : String,subjPrice : double)

Table 4: Algorithm for calcSalary

|  |  |
| --- | --- |
| Responsibility | Calculate salary for teacher |
| Input Parameter | teacherId : String  subjCode : String  subjPrice : double |
| Output Parameter | None |
| Algorithm | BEGIN  Insert teacherId, subjCode, subjPrice  Calculate salary  END |

### 4.3.7 Salary Class Design

This subparagraph specifies the design of Salary

1. Input/output data elements

List of input and output data elements:

Table 1: Input and Output Data for Salary Class

|  |  |
| --- | --- |
| Input | Output |
| teacherId: String, date:int, subjCode: String, subjSalary: double | - |

1. Local data elements

Table 1: Local Data Elemnts for Salary Class

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Name | Data type | Description |
| 1 | teacherId | String | Teacher Id |
| 2 | date | int | Date |
| 3 | subjCode | String | Subject code |
| 4 | subjSalary | double | Subject salary |
| 5 | teachSalary: | double | teacherSalary |

1. Algorithms

This section states the purpose and describes in detail the algorithms of this class

Table 1: Detail for Salary Class

|  |  |
| --- | --- |
| Class Type | Controller class |
| Responsibility | Save salary payment and salary |
| Attributes | pSlipManager : PaySlipManager |
| Methods | getSalary(teacherId : String) : double  saveSalary(teacherId: String,subjCode: String, subjSalary : double) : void  savePayment(teacherId: String, date: int): void |

1. getSalary(teacherId : String)

Table 2: Algorithm for getSalary

|  |  |
| --- | --- |
| Responsibility | Retrieve salary based on teacher ID |
| Input Parameter | teacherId : String |
| Output Parameter | teachSalary: double |
| Algorithm | BEGIN  Insert teacherId  Get teachSalary  END |

1. saveSalary(teacherId : String,subjCode : String,subjSalary : double)

Table 3: Algorithm for saveSalary

|  |  |
| --- | --- |
| Responsibility | Store salary of all the teachers |
| Input Parameter | teacherId : String  subjCode : String  subjSalary : double |
| Output Parameter | None |
| Algorithm | BEGIN  Insert teacherId and date  Store salary  END |

1. savePayment(teacherId : String,date : int)

Table 4: Algorithm for savePayment

|  |  |
| --- | --- |
| Responsibility | Store salary that has been paid to teachers |
| Input Parameter | teacherId : String  date : int |
| Output Parameter | None |
| Algorithm | BEGIN  Insert teacherId and date  Store salPayment  END |

### 4.3.8 ReportView Class Design

This subparagraph specifies the design of

1. Input/output data elements

List of input and output data elements:

Table 1: Input and Output Data for ReportView Class

|  |  |
| --- | --- |
| Input | Output |
| stdId: String, teacherId: String,date: int | - |

1. Local data elements

Table 1: Local Data Elements for ReportView Class

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Name | Data type | Description |
| 1 | stdId | String | Student Id |
| 2 | teacherId | String | Teacher Id |
| 3 | date | int | Date |

1. Algorithms

This section states the purpose and describes in detail the algorithms of this class

Table 1: Detail for ReportView Class

|  |  |
| --- | --- |
| Class Type | Controller class |
| Responsibility | Enable to access pay slip and receipt |
| Attributes | tFee : TuitionFee  rManager : ReceiptManager  salary : Salary  pSlipManager : PaySlipManager |
| Methods | generateReceipt(stdId : String,date : int) : void  getReceipt(stdId : String) : String  generatePaySlip(teacherId : String,date : int) : void getPaySlip(teacherId : String) : String |

1. generateReceipt(stdId : String, date: int)

Table 2: Algorithm for generateReceipt

|  |  |
| --- | --- |
| Responsibility | Generate receipt for students who has paid their tuition fees. |
| Input Parameter | stdId : String  date: int |
| Output Parameter | None |
| Algorithm | BEGIN  Insert stdId and date  Generate receipt  END |

1. getReceipt(stdId : String)

Table 3: Algorithm for getReceipt

|  |  |
| --- | --- |
| Responsibility | Retrieve receipt for student |
| Input Parameter | stdId : String |
| Output Parameter | stdReceipt: String |
| Algorithm | BEGIN  Insert studentId  Get stdReceipt  END |

1. generatePaySlip(teacherId: String, date : int)

Table 4: Algorithm for generatePaySlip

|  |  |
| --- | --- |
| Responsibility | Generate pay slip for teacher for every success salary payment |
| Input Parameter | teacherId: String, date : int |
| Output Parameter | None |
| Algorithm | BEGIN  Insert teacherId and date  Generate pay slip  END |

1. getPaySlip(teacherId : String)

Table 5: Algorithm for getPaySlip

|  |  |
| --- | --- |
| Responsibility | Retrieve pay slip for teacher |
| Input Parameter | teacherId : String |
| Output Parameter | teacherPaySlip: String |
| Algorithm | BEGIN  Insert teacherId  Get teacherPaySlip  END |

### 4.3.9 Class PaySlipManager Class Design

This subparagraph specifies the design of PaySlipManager

1. Input/output data elements

List of input and output data elements:

Table 1: Input and Output Data for PaySlipManager Class

|  |  |
| --- | --- |
| Input | Output |
| teacherId: String, date: int | - |

1. Local data elements

Table 1: Local Data Elements for PaySlipManager Class

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Name | Data type | Description |
| 3 | teacherId | String | Teacher Id |
| 4 | date | int | Date |

1. Algorithms

This section states the purpose and describes in detail the algorithms of this class

Table 1: Detail for PaySlipManager Class

|  |  |
| --- | --- |
| Class Type | Controller class |
| Responsibility | Generate and retrieve pay slip. |
| Attributes | pSlip : PaySlip |
| Methods | generatePaySlip(teacherId : String,date : int) : void  getPaySlip(teacherId : String) : String |

1. generatePaySlip(teacherId : String,date : int)

Table 2: Algorithm for generatePaySlip

|  |  |
| --- | --- |
| Responsibility | Generate pay slip for teacher for every success salary payment |
| Input Parameter | teacherId : String  date : int |
| Output Parameter | None |
| Algorithm | BEGIN  Insert teacher Id and date  Generate pay slip  END |

1. getPaySlip(teacherId : String) : String

Table 3: Algorithm for getPaySlip

|  |  |
| --- | --- |
| Responsibility | Retrieve pay slip for teacher |
| Input Parameter | teacherId : String |
| Output Parameter | teacherPaySlip: String |
| Algorithm | BEGIN  Insert teacherId  Get teacherPaySlip  END |

### 4.3.10 PaySlip Class Design

This subparagraph specifies the design of

1. Input/output data elements

List of input and output data elements:

Table 1: Input and Output for PaySlip Class

|  |  |
| --- | --- |
| Input | Output |
| teacherId: String, date:int | teacherPaySlip: String |

1. Local data elements

Table 1: Local Data Elements for PaySlip Class

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Name | Data type | Description |
| 1 | teacherId | String | Teacher Id |
| 2 | date | Integer | Date |
| 3 | teacherPaySlip | String | Teacher pay slip |

1. Algorithms

This section states the purpose and describes in detail the algorithms of this class

Table 1: Detail for PaySlip Class

|  |  |
| --- | --- |
| Class Type | Controller class |
| Responsibility | Save and get pay slip generated |
| Attributes | teacherId: String  date: int  teacherPaySlip: String |
| Methods | savePaySlip(teacherId : String,date : int) : void  getPaySlip(teacherId : String) : String |

1. savePaySlip(teacherId : String,date : int)

Table 2: Algorithm for savePaySlip

|  |  |
| --- | --- |
| Responsibility | Save generated pay slip |
| Input Parameter | teacherId : String,date : int |
| Output Parameter | None |
| Algorithm | BEGIN  Insert teacherId, date  Store payslip  END |

1. getPaySlip(teacherId : String)

Table 3: Algorithm for getPaySlip

|  |  |
| --- | --- |
| Responsibility | Retrieve pay slip for teacher |
| Input Parameter | teacherId : String |
| Output Parameter | teacherPaySlip: String |
| Algorithm | BEGIN  Insert teacherId  Get teacherPaySlip  END |

### 4.3.11 ReceiptManager Class Design

This subparagraph specifies the design of ReceiptManager

1. Input/output data elements

List of input and output data elements:

Table 1: Input and Output Data for ReceiptManager Class

|  |  |
| --- | --- |
| Input | Output |
| stdId: String , date: int | - |

1. Local data elements

Table 1: Local Data Elements for ReceiptManager Class

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Name | Data type | Description |
| 1 | stdId | String | Student Id |
| 2 | date | Date | Date |

1. Algorithms

This section states the purpose and describes in detail the algorithms of this class

Table 1: Detail for ReceiptManager Class

|  |  |
| --- | --- |
| Class Type | Controller class |
| Responsibility | Manage receipt |
| Attributes | receipt : Receipt |
| Methods | generateReceipt(stdId : String,date : int) : void  getReceipt(stdId : String) : String |

1. generateReceipt(stdId : String,date : int)

Table 2: Algorithm for generateReceipt

|  |  |
| --- | --- |
| Responsibility | Generate receipt for students |
| Input Parameter | stdId : String  date : int |
| Output Parameter | stdReceipt: String |
| Algorithm | BEGIN  Insert stdId and date  Generate receiptG  END |

1. getReceipt(stdId : String) : String

Table 3: Algorithm for getReceipt

|  |  |
| --- | --- |
| Responsibility | Retrieve generated receipt |
| Input Parameter | stdId: String |
| Output Parameter | receiptG: String |
| Algorithm | BEGIN  Insert stdId, date  Get receiptGs  END |

### 4.3.12 Class Receipt Design

This subparagraph specifies the design of Receipt

1. Input/output data elements

List of input and output data elements:

Table 1: Input And Output Data for Class Receipt

|  |  |
| --- | --- |
| Input | Output |
| teacherId: String, date: int | receiptG: String |

1. Local data elements

Table 1: Local data elements for Class Receipt

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Name | Data type | Description |
| 1 | stdId | String | Student Id |
| 2 | date | int | Date |
| 3 | receiptG | String | Receipt |

1. Algorithms

This section states the purpose and describes in detail the algorithms of this class

Table 1: Detail for Class Receipt Design

|  |  |
| --- | --- |
| Class Type | Controller class |
| Responsibility | Generate and save receipt generated |
| Attributes | stdId: String  date: int |
| Methods | saveReceipt(stdId : String,date : int) : void  getReceipt(stdId : String) : String |

1. saveReceipt(stdId : String,date : int)

Table 2: Algorithm for saveReceipt

|  |  |
| --- | --- |
| Responsibility | Save generated receipt |
| Input Parameter | stdId : String,date : int |
| Output Parameter | - |
| Algorithm | BEGIN  Insert stdId, date  Store saveReceipt  END |

1. getReceipt(stdId : String) : String

Table 3: Algorithm for getReceipt

|  |  |
| --- | --- |
| Responsibility | Retrieve generated receipt |
| Input Parameter | stdId: Strinf |
| Output Parameter | receiptG: String |
| Algorithm | BEGIN  Insert stdId, date  Get receiptGs  END |

## 4.4 Attendance

To view attendance, can select student attendance and can select teacher attendance.

### 4.4.1 Attendance Interface Class Design

This subparagraph specifies the design of AttendanceInterface

1. Input/output data elements

List of input and output data elements:

Table 1: Input and Output data for Attendance Interface Class

|  |  |
| --- | --- |
| Input | Output |
| stdId: String  staffId : String  teacherId : String | - |

1. Local data elements

Table 1: Local Data Elements for Attendance Interface Class

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Name | Data type | Description |
| 1 | stdId | String | Student Id |
| 2 | staffId | String | Staff Id |
| 3 | teacherId | String | Teacher Id |
| 4 | date | Date | Date |
| 5 | subjCode | String | Subject Code |
| 6 | time | Time | Time |
| 7 | stdAddr | String | Student Address |

1. Algorithms

This section states the purpose and describes in detail the algorithms of this class

Table 1: Detail for Attendance Interface Class

|  |  |
| --- | --- |
| Class Type | Controller Class |
| Responsibility | Provide user with another interfaces. |
| Attributes | subjM : SubjectManager  attendanceM : AttendanceManager  stdId : String  staffId : String  teachId : String  mwli : ManageWarningLetterInterface |
| Methods | viewAttendance(stdId : String, teacherId : String, staffId : String) : void  addAttendance() : void  deleteAttendance() : String  updateAttendance() : String |

1. viewAttendance(stdId : String, teacherId : String, staffId : String) : void

Table 2: Algorithm for viewAttendance

|  |  |
| --- | --- |
| Responsibility | View attendance for student, staff and teacher. |
| Input Parameter | stdId, teachId, staffId |
| Output Parameter | None |
| Algorithm | BEGIN  Enter id number  View the attendance  END |

1. addAttendance() : void

Table 3: Algorithm for addAttendance

|  |  |
| --- | --- |
| Responsibility | -add attendance for staff and teacher. For student attendance, just teacher can add. |
| Input Parameter | - |
| Output Parameter | - |
| Algorithm | BEGIN  Add attendance  END |

1. deleteAttendance() : String

Table 4: Algorithm for deleteAttendance

|  |  |
| --- | --- |
| Responsibility | Can delete attendance for staff, teacher and student. |
| Input Parameter | - |
| Output Parameter | - |
| Algorithm | BEGIN  Delete attendance  END |

### 4.4.2 Manage Warning Letter Interface Class Design

This subparagraph specifies the design of ManageWarningLetterInterface

1. Input/output data elements

List of input and output data elements:

Table 1: Input and Output for Manage Warning Letter Interface

|  |  |
| --- | --- |
| Input | Output |
| -attendance : AttendanceManager  -warningletterM : WarningLetterManager | None |

1. Local data elements

Table 1: Local Data Elements for Manage Warning Letter Interface Class

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Name | Data type | Description |
| 1 | stdId | String | Student Id |
| 2 | staffId | String | Staff Id |
| 3 | teacherId | String | Teacher Id |
| 4 | date | Date | Date |
| 5 | subjCode | String | Subject Code |
| 6 | time | Time | Time |
| 7 | stdAddr | String | Student Address |

1. Algorithms

This section states the purpose and describes in detail the algorithms of this class

Table 1: Detail for Manage Warning Letter Interface

|  |  |
| --- | --- |
| Class Type | ManageWarningLetterInterface |
| Responsibility | Provide staff with three warning interfaces. |
| Attributes | attendanceM : AttendanceManager  warningletterM : WarningLetterInterface |
| Methods | checkAbsentStudent() : String |

1. checkAbsentStudent() : String

Table 2: Algorithm for checkAbsentStudent

|  |  |
| --- | --- |
| Responsibility | To check student are absent more 3 times. |
| Input Parameter | - |
| Output Parameter | - |
| Algorithm | BEGIN  Enter subjCode and date  Check absent student  END |

### 4.4.3 Warning Letter Manager Class Design

This subparagraph specifies the design of WarningLetterManager

1. Input/output data elements

List of input and output data elements:

Table 1: Input and Output Data for Warning Letter Manager Class

|  |  |
| --- | --- |
| Input | Output |
| stdId : String  stdName : String  stdAdd : String  date : Date  subjCode : String | None |

1. Local data elements

Table 1: Local Data Elements for Warning Letter Manager Class

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Name | Data type | Description |
| 1 | stdId | String | Student Id |
| 2 | staffId | String | Staff Id |
| 3 | teacherId | String | Teacher Id |
| 4 | date | Date | Date |
| 5 | subjCode | String | Subject Code |
| 6 | time | Time | Time |
| 7 | stdAddr | String | Student Address |

1. Algorithms

This section states the purpose and describes in detail the algorithms of this class

Table 1: Detail for Warning Letter Manager Class

|  |  |
| --- | --- |
| Class Type | Controller |
| Responsibility | As a class that communicate with database. |
| Attributes | -stdAdd : String  -stdName : String  -wLetter : WarningLetter  -date : Date  -subjCode : String  -stdId : String |
| Methods | -generateWarningLetter(stdId: String, stdName : String, stdAddress : String) : String  -getStudentInfo(stdAddr) : String |

1. generateWarningLetter(stdId: String, stdName : String, stdAddress : String) : String

Table 2: Algorithm for generateWarningLetter

|  |  |
| --- | --- |
| Responsibility | To generate warning letter from class warningLetterManager. |
| Input Parameter | (stdId: String, stdName : String, stdAddress ): String |
| Output Parameter | - |
| Algorithm | BEGIN  Go to warning letter interface  Select generate warning letter  END |

1. getStudentInfo(stdAddress ) : String

Table 3: Algorithm for getStudentInfo

|  |  |
| --- | --- |
| Responsibility | Get information from student manager like stdAddr. |
| Input Parameter | stdAddress : String |
| Output Parameter | - |
| Algorithm | BEGIN  Get student info(student address)  END |

### 4.4.4 Warning Letter Class Design

This subparagraph specifies the design of WarningLetter

1. Input/output data elements

List of input and output data elements:

Table 1: Input and Output Data for Warning Letter Class

|  |  |
| --- | --- |
| Input | Output |
| stdid : String  subjCode : String | None |

1. Local data elements

Table 1: Local Data Elements for Warning Letter Class

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Name | Data type | Description |
| 1 | stdId | String | Student Id |
| 2 | staffId | String | Staff Id |
| 3 | teacherId | String | Teacher Id |
| 4 | date | Date | Date |
| 5 | subjCode | String | Subject Code |
| 6 | time | Time | Time |
| 7 | stdAddr | String | Student Address |

1. Algorithms

Table 1: Detail for Warning Letter Class

|  |  |
| --- | --- |
| Responsibility | Act as database for warning letter interfaces. |
| Attributes | managewarningletter : ManageWarningLetter  stdId : String  subjCode : String |
| Methods | setWarningLetter(stdId : string, subjCode : string) : void |

1. setWarningLetter() : String

Table 2: Algorithm for setWarningLetter

|  |  |
| --- | --- |
| Responsibility | To set warning letter from data generate warning letter. |
| Input Parameter | stdId  subjCode |
| Output Parameter | - |
| Algorithm | BEGIN  Go to warning letter interface  Go to generate warning letter  Set warning letter  END |

### 4.4.5 Attendance Manager Class Design

This subparagraph specifies the design of AttendanceManager

1. Input/output data elements

List of input and output data elements:

Table 1: Input and Output for Attendance Manager Class

|  |  |
| --- | --- |
| Input | Output |
| stdId : String  staffId : String  teachId : String  date : Date  time : Time | None |

1. Local data elements

Table 1: Local Data Elements for Attendance Manager Class

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Name | Data type | Description |
| 1 | stdId | String | Student Id |
| 2 | staffId | String | Staff Id |
| 3 | teacherId | String | Teacher Id |
| 4 | date | Date | Date |
| 5 | subjCode | String | Subject Code |
| 6 | time | Time | Time |
| 7 | stdAddr | String | Student Address |

1. Algorithms

This section states the purpose and describes in detail the algorithms of this class

Table 1: Detail for Attendance Manager Class

|  |  |
| --- | --- |
| Class Type | AttendanceManager |
| Responsibility | Data that will store in database. |
| Attributes | Att : Attendance  stdId : String  staffId : String  teachId : String  date : Date  time : Time |
| Methods | getAttendance(stdId : string,staffId : string,teachId : string) : void  setAttendance(stdId : string,staffId : string,teachId : string) : void  deleteAttendance(stdId : string,staffId : string,teachId : string) : void  updateAttendance(stdId : string,staffId : string,teachId : string) : void  enterDateTime(date : Date,time : Time) : void |

1. getAttendance(stdId : string,staffId : string,teachId : string) : void

Table 2: Algorithm for getAttendance

|  |  |
| --- | --- |
| Responsibility | Get stdId, staffId and teachId from database. |
| Input Parameter | staffId, stdId and teachId |
| Output Parameter | - |
| Algorithm | BEGIN  Go to attendance interface  Get data from student, staff and teacher interface  END |

1. setAttendance(stdId : string,staffId : string,teachId : string) : void

Table 3: Algorithm for setAttendance

|  |  |
| --- | --- |
| Responsibility | Set the attendance |
| Input Parameter | staffId  stdId  teachId |
| Output Parameter | - |
| Algorithm | BEGIN  Go to attendance interface  Set data from student, staff and teacher interface  END |

1. deleteAttendance(stdId : string,staffId : string,teachId : string) : void

Table 4: Algorithm for deleteAttendance

|  |  |
| --- | --- |
| Responsibility | To delete attendance. |
| Input Parameter | stdId  staffId  teachId |
| Output Parameter | - |
| Algorithm | BEGIN  Go to attendance interface  Delete attendance  END |

1. updateAttendance(stdId : string,staffId : string,teachId : string) : void

Table 5: Algorithm for updateAttendance

|  |  |
| --- | --- |
| Responsibility | To update attendance |
| Input Parameter | stdId, staffId and teachId |
| Output Parameter | - |
| Algorithm | BEGIN  Go to attendance interface  Delete attendance  END |

1. enterDateTime(date : Date,time : Time) : void

Table 6: Algorithm for enterDateTime

|  |  |
| --- | --- |
| Responsibility | Set the date and time for attendance. |
| Input Parameter | -Date  -Time |
| Output Parameter | - |
| Algorithm | BEGIN  Go to attendance interface  Set date and time  END |

### 4.4.6 Attendance Class Design

This subparagraph specifies the design of Attendance

1. Input/output data elements

List of input and output data elements:

Table 1: Input and Output Data for Attendance Class

|  |  |
| --- | --- |
| Input | Output |
| stdId : String  staffId : String  teachId : String | None |

1. Local data elements

Table 1: Local Data Elements for Attendance Class

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Name | Data type | Description |
| 1 | stdId | String | Student Id |
| 2 | staffId | String | Staff Id |
| 3 | teacherId | String | Teacher Id |
| 4 | date | Date | Date |
| 5 | subjCode | String | Subject Code |
| 6 | time | Time | Time |
| 7 | stdAddr | String | Student Address |

1. Algorithms

This section states the purpose and describes in detail the algorithms of this class

Table 1: Detail for Attendance Class

|  |  |
| --- | --- |
| Class Type | Database |
| Responsibility | As a database for attendance |
| Attributes | stdId : String  staffId : String  teacherId : String |
| Methods | getAttendance(stdId : string,staffId : string,teachId : string) : void  checkAttendance() : String  setAttendance(stdId : string,staffId : string,teachId : string) : vo  deleteAttendance(stdId : string,staffId : string,teachId : string) : void |

1. getAttendance(stdId : string,staffId : string,teachId : string) : void

Table 2: Algorithm for getAttendance

|  |  |
| --- | --- |
| Responsibility | Get information like stdId, staffId, and teachId |
| Input Parameter | stdId  staffId  teachId |
| Output Parameter | - |
| Algorithm | BEGIN  Go to attendance interface  Get data from student, staff and teacher interface  END |

1. checkAttandance() : String

Table 3: Algorithm for checkAttendance

|  |  |
| --- | --- |
| Responsibility | To check attendance |
| Input Parameter | - |
| Output Parameter | - |
| Algorithm | BEGIN  Attendance interface  Check attendance  END |

1. setAttendance(stdId : string,staffId : string,teachId : string) : void

Table 4: Algorithm for setAttendance

|  |  |
| --- | --- |
| Responsibility | Set attendance by use stdId, staffId and teachId |
| Input Parameter | stdId, teachId and staffId |
| Output Parameter | - |
| Algorithm | BEGIN  Set attendance  END |

1. deleteAttendance(stdId : string,staffId : string,teachId : string)

Table 5: Algorithm for deleteAttendance

|  |  |
| --- | --- |
| Responsibility | Delete attendance |
| Input Parameter | stdId  staffId  teachId |
| Output Parameter | - |
| Algorithm | BEGIN  Delete attendance  END |

## 4.5 Student Performance

### 4.5.1. Student Performance Interface Design

This subparagraph specifies the design of Student Performance Interface

1. Input/output data elements

List of input and output data elements:

Table 1: Input and Output Data for Student Performance Interface

|  |  |
| --- | --- |
| Input | Output |
| choose: int | - |

1. Local data elements

Table 1: Local Data Elements for Student Performance Interface

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Name | Data type | Description |
| 1. | stdId | String | Student Id |
| 2. | teachId | String | Teacher Id |
| 3. | quizSubj | String | Subject Quiz |
| 4. | quizChap | String | Chapter Quiz |
| 5. | attitude | String | Attitude feedback |
| 6. | comment | String | Comment feedback |
| 7. | subjCode | String | Subject Code |

1. Algorithms

This section states the purpose and describes in detail the algorithms of this class

Table 1: Detail for Student Performance Interface

|  |  |
| --- | --- |
| Class Type | Controller class |
| Responsibility | Determine whether to go evaluation interface or result interface |
| Attributes | choose: int |
| Methods | evaluationINterface() : void  resultInterface() : void |

1. evaluationINterface() : void

Table 2: Algorithm for evaluationInterface

|  |  |
| --- | --- |
| Responsibility | Enable to navigate Evaluation Interface |
| Input Parameter | None |
| Output Parameter | None |
| Algorithm | BEGIN  Select Evaluation button  Go to Evaluation Page  END |

1. resultInterface() : void

Table 3: Algorithm for resultInterface

|  |  |
| --- | --- |
| Responsibility | Enable to navigate to Result Interface |
| Input Parameter | None |
| Output Parameter | None |
| Algorithm | BEGIN  Select Result button  Go to Result Page  END |

### 4.5.2. Evaluation Interface Class Design

This subparagraph specifies the design of Evaluation Interface

1. Input/output data elements

List of input and output data elements:

Table 1: Input and Output Data for Evaluation Interface Class

|  |  |
| --- | --- |
| Input | Output |
| Add : int | - |

1. Local data elements

Table 1: Local Data Elements for Evaluation Interface Class

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Name | Data type | Description |
| 1. | stdId | String | Student Id |
| 2. | teachId | String | Teacher Id |
| 3. | quizSubj | String | Subject Quiz |
| 4. | quizChap | String | Chapter Quiz |
| 5. | attitude | String | Attitude feedback |
| 6. | comment | String | Comment feedback |
| 7. | subjCode | String | Subject Code |

1. Algorithms

This section states the purpose and describes in detail the algorithms of this class

Table 1: Detail for Evaluation Interface Class

|  |  |
| --- | --- |
| Class Type | Controller Class |
| Responsibility | Provide evaluation intercafe |
| Attributes | teacherId : String  StdId : String |
| Methods | addEvaluation(teacherId : String) : void  editEvaluation(teacherId : String) : void  addEvaluation(stdId : String) : void  editEvaluation(stdId : String) : void |

1. addEvaluation(teacherId : String) : void

Table 2: Algorithm for addEvaluation

|  |  |
| --- | --- |
| Responsibility | To add teacher evaluation |
| Input Parameter | teacherId : string |
| Output Parameter | - |
| Algorithm | BEGIN  Select Add button  Fill Evaluation  END |

1. editEvaluation(teacherId : String) : void

Table 3: Algorithm for editEvaluation

|  |  |
| --- | --- |
| Responsibility | To edit teacher evaluation |
| Input Parameter | teacherId : String |
| Output Parameter | - |
| Algorithm | BEGIN  Select Edd button  Fill Evaluation  END |

1. addEvaluation(stdId : String) : void

Table 4: Algorithm for addEvaluation

|  |  |
| --- | --- |
| Responsibility | To add evaluation to student |
| Input Parameter | stdId : String |
| Output Parameter | - |
| Algorithm | BEGIN  Select Add button  Fill Evaluation  END |

1. editEvaluation(stdId : String) : void

Table 7: Algorithm for editEvaluation

|  |  |
| --- | --- |
| Responsibility | To edit evaluation to student |
| Input Parameter | stdId : String |
| Output Parameter | - |
| Algorithm | BEGIN  Select Add button  Fill Evaluation  END |

### 4.1.3. Evaluation Form Interface Class Design

This subparagraph specifies the design of Evaluation Form Interface

1. Input/output data elements

List of input and output data elements:

Table 1: Input and Output Data for Evaluation Form Interface

|  |  |
| --- | --- |
| Input | Output |
| Fill : Char | - |

1. Local data elements

Table 1: Local Data Elements for Evaluation Form Interface Class

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Name | Data type | Description |
| 1. | stdId | String | Student Id |
| 2. | teachId | String | Teacher Id |
| 3. | quizSubj | String | Subject Quiz |
| 4. | quizChap | String | Chapter Quiz |
| 5. | attitude | String | Attitude feedback |
| 6. | comment | String | Comment feedback |
| 7. | subjCode | String | Subject Code |

1. Algorithms

This section states the purpose and describes in detail the algorithms of this class

Table 1: Detail for Evaluation Form Interface Class

|  |  |
| --- | --- |
| Class Type | Class Controller |
| Responsibility | To display interface form to user to fill the evaluation |
| Attributes | StdId : String  teacherId : String |
| Methods | fillEvaluation(stdId : String) : void  fillEvaluation(teacherId : String) : void |

1. fillEvaluation(stdId : String) : void

Table 2: Algorithm for fillEvaluation

|  |  |
| --- | --- |
| Responsibility | To fill evaluation for student |
| Input Parameter | stdId : string |
| Output Parameter | - |
| Algorithm | BEGIN  Select Add button  Fill Evaluation  END |

1. fillEvaluation(teacherId : String) : void

Table 3: Algorithm for fillEvaluation

|  |  |
| --- | --- |
| Responsibility | To fill evaluation for teacher |
| Input Parameter | teacherId : String |
| Output Parameter | - |
| Algorithm | BEGIN  Select Add button  Fill Evaluation  END |

### 4.1.4. Evaluation Manager Class Design

This subparagraph specifies the design of Evaluation Manager

1. Input/output data elements

List of input and output data elements:

Table 1: Input and Output Data for Evaluation Manager Class

|  |  |
| --- | --- |
| Input | Output |
| Manage | - |

1. Local data elements

Table 1: Local Data Elements for Evaluation Manager Class

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Name | Data type | Description |
| 1. | stdId | String | Student Id |
| 2. | teachId | String | Teacher Id |
| 3. | quizSubj | String | Subject Quiz |
| 4. | quizChap | String | Chapter Quiz |
| 5. | attitude | String | Attitude feedback |
| 6. | comment | String | Comment feedback |
| 7. | subjCode | String | Subject Code |

1. Algorithms

This section states the purpose and describes in detail the algorithms of this class

Table 1: Detail for Evaluation Manager Class

|  |  |
| --- | --- |
| Class Type | Class Controller |
| Responsibility | To manage all the add and edit evaluation in the class |
| Attributes | Std(attitude, comment) : char  Teach(attitude, comment) : char  stdId : String  teacherId : string |
| Methods | addTeach(attitude,comment : char) : void  editTeach(attitude,comment : char) : void  addStd(attitude,comment : char) : void  editStd(attitude,comment : char) : void  verify(stdId : String) : String  verify(teacherId : String) : String |

1. addTeach(attitude,comment : char) : void

Table 2: Algorithm for addTeach

|  |  |
| --- | --- |
| Responsibility | Add teacher attitude and comment evaluation |
| Input Parameter | attitude, comment : Char |
| Output Parameter | - |
| Algorithm | BEGIN  Select Add button  Fill Evaluation  END |

1. editTeach(attitude,comment : char) : void

Table 3: Algorithm for editTeach

|  |  |
| --- | --- |
| Responsibility | Edit teacher attitude and comment evaluation |
| Input Parameter | attitude, comment : Char |
| Output Parameter | - |
| Algorithm | BEGIN  Select Edd button  Fill Evaluation  END |

1. addStd(attitude,comment : char) : void

Table 4: Algorithm for addStd

|  |  |
| --- | --- |
| Responsibility | Add student attitude and comment evaluation |
| Input Parameter | attitude, comment : Char |
| Output Parameter | - |
| Algorithm | BEGIN  Select Add button  Fill Evaluation  END |

1. editStd(attitude,comment : char) : void

Table 5: Algorithm for editStd

|  |  |
| --- | --- |
| Responsibility | Edit student attitude and comment evaluation |
| Input Parameter | attitude, comment : Char |
| Output Parameter | - |
| Algorithm | BEGIN  Select Edd button  Fill Evaluation  END |

1. verify(stdId : String) : String

Table 6: Algorithm for verify

|  |  |
| --- | --- |
| Responsibility | Verify student id |
| Input Parameter | stdId : string |
| Output Parameter | - |
| Algorithm | BEGIN  none  END |

1. verify(teacherId : String) : String

Table 7: Algorithm for verify

|  |  |
| --- | --- |
| Responsibility | Verify teacher id |
| Input Parameter | teacherId : String |
| Output Parameter | - |
| Algorithm | BEGIN  None  END |

### 4.5.5. View Evaluation Interface Class Design

This subparagraph specifies the design of View Evaluation Interface

1. Input/output data elements

List of input and output data elements:

Table 1: Input and Output Data for View Evaluation Interface Class

|  |  |
| --- | --- |
| Input | Output |
| Choose : int | View evaluation |

1. Local data elements

Table 1: Local Data Elements for View Evaluation Interface Class

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Name | Data type | Description |
| 1. | stdId | String | Student Id |
| 2. | teachId | String | Teacher Id |
| 3. | quizSubj | String | Subject Quiz |
| 4. | quizChap | String | Chapter Quiz |
| 5. | attitude | String | Attitude feedback |
| 6. | comment | String | Comment feedback |
| 7. | subjCode | String | Subject Code |

1. Algorithms

This section states the purpose and describes in detail the algorithms of this class

Table 1: Detail for View Evaluation Interface

|  |  |
| --- | --- |
| Class Type | Class Controller |
| Responsibility | Display evaluation for user |
| Attributes | StdId : String  teacherId : String |
| Methods | viewEvaluation(stdId : String) : String  viewEvaluation(teacherId : String) : String |

1. viewEvaluation(stdId : String) : String

Table 2: Algorithm for viewEvaluation

|  |  |
| --- | --- |
| Responsibility | View student evaluation |
| Input Parameter | stdId : String |
| Output Parameter | Return data to user |
| Algorithm | BEGIN  Select Evaluation button  END |

1. viewEvaluation(teacherId : String) : String

Table 3: Algorithm for viewEvaluation

|  |  |
| --- | --- |
| Responsibility | View teacher evaluation |
| Input Parameter | teacherId : String |
| Output Parameter | Return data to user |
| Algorithm | BEGIN  Select evaluation button  END |

### 4.1.6. Evaluation Class Design

This subparagraph specifies the design of Evaluation

1. Input/output data elements

List of input and output data elements:

Table 1: Input and Output for Evaluation Class

|  |  |
| --- | --- |
| Input | Output |
| database | - |

1. Local data elements

Table 1: Local Data Elements for Evaluation Class

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Name | Data type | Description |
| 1. | stdId | String | Student Id |
| 2. | teachId | String | Teacher Id |
| 3. | quizSubj | String | Subject Quiz |
| 4. | quizChap | String | Chapter Quiz |
| 5. | attitude | String | Attitude feedback |
| 6. | comment | String | Comment feedback |
| 7. | subjCode | String | Subject Code |

1. Algorithms

This section states the purpose and describes in detail the algorithms of this class

Table 1: Detail for Evaluation Class

|  |  |
| --- | --- |
| Class Type | Class Controller |
| Responsibility | To save all the data inside the evaluation database |
| Attributes | Attitude,comment : char |
| Methods | setTeach(attitude,comment : char) : void  setStd(attitude,comment : char) : void |

1. setTeach(attitude,comment : char) : void

Table 2: Algorithm for setTeach

|  |  |
| --- | --- |
| Responsibility | To set and save evaluation |
| Input Parameter | Attitude,comment : char |
| Output Parameter | Display data to user |
| Algorithm | BEGIN  Fill Evaluation  Select save button  END |

1. setStd(attitude,comment : char) : void

Table 3: Algorithm for setStd

|  |  |
| --- | --- |
| Responsibility | To set and save evaluation |
| Input Parameter | Attitude,comment : char |
| Output Parameter | Display data to user |
| Algorithm | BEGIN  Fill Evaluation  Select save button  END |

### 4.1.7. Result Interface Class Design

This subparagraph specifies the design of Result Interface

1. Input/output data elements

List of input and output data elements:

Table 1: Input and Output Data for Result Interface Class

|  |  |
| --- | --- |
| Input | Output |
| Choose : int | - |

1. Local data elements

Table 1: Local Data Elements for Result Interface Class

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Name | Data type | Description |
| 1. | stdId | String | Student Id |
| 2. | teachId | String | Teacher Id |
| 3. | quizSubj | String | Subject Quiz |
| 4. | quizChap | String | Chapter Quiz |
| 5. | attitude | String | Attitude feedback |
| 6. | comment | String | Comment feedback |
| 7. | subjCode | String | Subject Code |

1. Algorithms

This section states the purpose and describes in detail the algorithms of this class

Table 1: Detail for Result Interface Class

|  |  |
| --- | --- |
| Class Type | Class Controller |
| Responsibility | To display interface for result field |
| Attributes | - |
| Methods | exerciseMark() : void  addResult() : void  editResult() : void |

1. exerciseMark() : void

Table 2: Algorithm for exerciseMark

|  |  |
| --- | --- |
| Responsibility | Help teacher to retrieve marks from teaching module |
| Input Parameter | - |
| Output Parameter | Return data to teacher |
| Algorithm | BEGIN  Select exercise button  Retrieve data  END |

1. addResult() : void

Table 3: Algorithm for addResult

|  |  |
| --- | --- |
| Responsibility | To add student result |
| Input Parameter | stdId, subjCode : String |
| Output Parameter | - |
| Algorithm | BEGIN  Select Add button  Fill result  Select save button  END |

1. editResult() : void

Table 4: Algorithm for editResult

|  |  |
| --- | --- |
| Responsibility | To edit result |
| Input Parameter | stdId, subjCode : String |
| Output Parameter | - |
| Algorithm | BEGIN  Select Add button  Fill result  Select save button  END |

### 4.1.8. Exercise Mark Interface Class Design

This subparagraph specifies the design of Evaluation Mark Interface

1. Input/output data elements

List of input and output data elements:

Table 1: Input and Output Data for Exercise Mark Interface Class

|  |  |
| --- | --- |
| Input | Output |
| Display : Char | StdId, subjCode |

1. Local data elements

Table 1: Local Data Element for Exercise Mark Interface Class

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Name | Data type | Description |
| 1. | stdId | String | Student Id |
| 2. | teachId | String | Teacher Id |
| 3. | quizSubj | String | Subject Quiz |
| 4. | quizChap | String | Chapter Quiz |
| 5. | attitude | String | Attitude feedback |
| 6. | comment | String | Comment feedback |
| 7. | subjCode | String | Subject Code |

1. Algorithms

This section states the purpose and describes in detail the algorithms of this class

Table 1: Detail for Exercise Mark Interface Class

|  |  |
| --- | --- |
| Class Type | Class controller |
| Responsibility | To retrieve student mark from teaching module |
| Attributes | sdtId : String  teacherId : String |
| Methods | getExerciseMark(stdId,subjCode : String) : void |

1. getExerciseMark(stdId,subjCode : String) : void

Table 2: Algorithm for getExerciseMark

|  |  |
| --- | --- |
| Responsibility | To get exercise mark of student |
| Input Parameter | stdId, subjCode : String |
| Output Parameter | Return student mark |
| Algorithm | BEG  Select Exercise Mark button  END |

### 4.1.9. Add Result Interface Class Design

This subparagraph specifies the design of Add Result Interface

1. Input/output data elements

List of input and output data elements:

Table 1: Input and Output Data for Add Result Interface Class

|  |  |
| --- | --- |
| Input | Output |
| Add : double | - |

1. Local data elements

Table 1: Local Data Element for Add Result Interface Class Design

|  |  |
| --- | --- |
| Name | Add |
| Description | Add mark for student result |
| Data type | double |
| Precision/resolution | - |

1. Algorithms

This section states the purpose and describes in detail the algorithms of this class

Table 1: Detail for Add Result Interface Class

|  |  |
| --- | --- |
| Class Type | Class controller |
| Responsibility | To add result for student |
| Attributes | stdId, subjCode : string |
| Methods | addMark(stdId,subjCode : String) : void |

1. addMark(stdId,subjCode : String) : void

Table 2: Algorithm for addMark

|  |  |
| --- | --- |
| Responsibility | Add mark to student |
| Input Parameter | stdId, subjCode : String |
| Output Parameter | - |
| Algorithm | BEGIN  Select Add button  Fill result  Select save button  END |

### 4.1.10. Edit Result Interface Class Design

This subparagraph specifies the design of Edit Result Interface

1. Input/output data elements

List of input and output data elements:

Table 1: Input and Output for Edit Result Interface Class

|  |  |
| --- | --- |
| Input | Output |
| Edit : double | - |

1. Local data elements

Table 1: Local Data Element for Edit Result Interface Class

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Name | Data type | Description |
| 1. | stdId | String | Student Id |
| 2. | teachId | String | Teacher Id |
| 3. | quizSubj | String | Subject Quiz |
| 4. | quizChap | String | Chapter Quiz |
| 5. | attitude | String | Attitude feedback |
| 6. | comment | String | Comment feedback |
| 7. | subjCode | String | Subject Code |

1. Algorithms

This section states the purpose and describes in detail the algorithms of this class

Table 1: Detail for Edit Result Interface

|  |  |
| --- | --- |
| Class Type | Class controller |
| Responsibility | To enable teacher to edit result |
| Attributes | stdId, subjCode :string |
| Methods | EditMarks(stdId,subjCode : String) : void |

1. EditMarks(stdId,subjCode : String) : void

Table 1: Algorithm for EditMarks

|  |  |
| --- | --- |
| Responsibility | Edit student result |
| Input Parameter | stdId, subjCode : string |
| Output Parameter | - |
| Algorithm | BEGIN  Select Edd button  Fill result  Select save button  END |

### 4.1.11. Mark Manager Class Design

This subparagraph specifies the design of Mark Manager

1. Input/output data elements

List of input and output data elements:

Table 1: Input and Output Data for Mark Manager Class

|  |  |
| --- | --- |
| Input | Output |
| Manage : char | - |

1. Local data elements

Table 1: Local Data Element for Mark Manager Class

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Name | Data type | Description |
| 1. | stdId | String | Student Id |
| 2. | teachId | String | Teacher Id |
| 3. | quizSubj | String | Subject Quiz |
| 4. | quizChap | String | Chapter Quiz |
| 5. | attitude | String | Attitude feedback |
| 6. | comment | String | Comment feedback |
| 7. | subjCode | String | Subject Code |

1. Algorithms

This section states the purpose and describes in detail the algorithms of this class

Table 1: Detail for Mark Manager Class

|  |  |
| --- | --- |
| Class Type | Class controller |
| Responsibility | Manage add and edit result |
| Attributes | stdId :string  teacherId : string  subjCode : String |
| Methods | getExerciseMark(stdId,subjCode : String) : void  fillResultMark() : void  editMark(stdId,subjCode : String) : void  getResult(stdId : String) : void |

1. getExerciseMark(stdId,subjCode : String) : void

Table 2: Algorithm for getExerciseMark

|  |  |
| --- | --- |
| Responsibility | Get exercise from teaching module |
| Input Parameter | stdId, subjCode : String |
| Output Parameter | Retrun data to user |
| Algorithm | BEGIN  Select Exercise button  Get Mark  END |

1. fillResultMark() : void

Table 3: Algorithm for fillResultMark

|  |  |
| --- | --- |
| Responsibility | Fill student result |
| Input Parameter | - |
| Output Parameter | - |
| Algorithm | BEGIN  none  END |

1. editMark(stdId,subjCode : String) : void

Table 4: Algorithm for editMark

|  |  |
| --- | --- |
| Responsibility | Edit student mark |
| Input Parameter | stdId, subjCode : String |
| Output Parameter | - |
| Algorithm | BEGIN  Select Edd button  Fill result  Select save button  END |

1. getResult(stdId : String) : void

Table 5: Algorithm for getResult

|  |  |
| --- | --- |
| Responsibility | Get student result |
| Input Parameter | stdId, subjCode : String |
| Output Parameter | Return result to user |
| Algorithm | BEGIN  Select Result button  Result display  END |

### 4.1.12. View Result Interface Class Design

This subparagraph specifies the design of View Result Interface

1. Input/output data elements

List of input and output data elements:

Table 1: Input and Output Data for View Result Interface

|  |  |
| --- | --- |
| Input | Output |
| View | View Result |

1. Local data elements

Table 1: Local Data for View

|  |  |
| --- | --- |
| Name | View |
| Description | Display result to user |
| Data type | - |
| Precision/resolution | - |

1. Algorithms

This section states the purpose and describes in detail the algorithms of this class

Table 1: Detail for View Result Interface

|  |  |
| --- | --- |
| Class Type | Class controller |
| Responsibility | Display result to user |
| Attributes | stdId : String |
| Methods | viewResult(stdId : String) : void |

1. viewResult(stdId : String) : void

Table 2: Algorithm for viewResult

|  |  |
| --- | --- |
| Responsibility | Return result to user |
| Input Parameter | stdId String |
| Output Parameter | View student result |
| Algorithm | BEGIN  Select Result button  END |

### 4.1.13. Mark Class Design

This subparagraph specifies the design of Mark

1. Input/output data elements

List of input and output data elements:

Table 1: Input and Output data for Mark Class

|  |  |
| --- | --- |
| Input | Output |
| Mark : String | -Result |

1. Local data elements

Table 1: Local Data Element for Mark Class

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Name | Data type | Description |
| 1. | stdId | String | Student Id |
| 2. | teachId | String | Teacher Id |
| 3. | quizSubj | String | Subject Quiz |
| 4. | quizChap | String | Chapter Quiz |
| 5. | attitude | String | Attitude feedback |
| 6. | comment | String | Comment feedback |
| 7. | subjCode | String | Subject Code |

1. Algorithms

This section states the purpose and describes in detail the algorithms of this class

Table 1:Detail for Mark Class

|  |  |
| --- | --- |
| Class Type | Class Controller |
| Responsibility | Set and save mark |
| Attributes | stdId : string  teacherId : string |
| Methods | setResultMark() : void  getResult(stdId : String) : void |

1. setResultMark() : void

Table 2: Algorithm for setResultMark

|  |  |
| --- | --- |
| Responsibility | Set mark |
| Input Parameter | - |
| Output Parameter | - |
| Algorithm | BEGIN  Select Add button  Fill result  Select save button  END |

1. getResult(stdId : String) : void

Table 3: Algorithm for getResult

|  |  |
| --- | --- |
| Responsibility | Get mark to user |
| Input Parameter | stdId : String |
| Output Parameter | Return data to user |
| Algorithm | BEGIN  Select Result button  END |

# 5. NOTES

Abbreviation used:

* CSCI Computer Software Configuration Items
* TMS Tuition Management System
* SDD Software Design Document
* SRS Software Requirement Specification
* ID Identification Detail

l