Software requirement specification document for project School Management

Ahmed Mohamed.

Supervised by: Dr. ......

October 31, 2020

1. **Introduction**

In our system we aim to find out a creative solution for school management system routine. This system will replace the usual paper-based system for school operations for a modern technology that is easy to use, and this system will make the user control his account via smart phones and laptops. it allows the client to get access on his duties, and rights.

## Purpose of this document

The purpose of this project to give solutions to the most problems in traditional schools. The functionalities of our system will facilitate the Customer’s school dealings. The intended audience are Students and Teachers.

## Scope of this document

The scope of this document will be on Students, Teachers and Admins. This system is responsible for operating an e-learning system, providing books library, taking assignments, and helping with education.

## Overview

In this system there are 5 Different types of users: Admin, Teachers, supervisor, drivers and Customer. Our system will facilitate for customers to make their schoolwork in easy way to use.

Diagram, schematic

Description automatically generated

## Business Context

Our project is tending to solve two main problems, it will solve the data redundancy and save time. it will also replace the usual paper-based system for school operations

.

# General Description

## Product Functions

. Functions of Our School Management System: • Admin Dashboard • Easy Clients registration • Student Dashboard • Teacher Dashboard • Admin Login pages are hidden from customers for security purposes. • Teacher mange the student’s assignments, exams and quizzes

## Similar System Information

## Edmodo online educational system: this is an online educational system the source code developed in PHP, HTML, JavaScript and MySQL database. This site has a beautiful and simple design but it has a limited number of users and they are the client and admin and it doesn’t have a lot of features.

## 1_vSDSgX8DC1uCLNdZClNXwAstudents_0000_Stay-On-trackedmodo-roster

## 

## maxresdefault

* 1. **User Characteristics**

People who will use our system must be proficient in using computers and smart- phones. They should know English to be able to deal with it.

## User Problem Statement

The user’s problem it was in the time wasted, data redundancy and failure Trans- actions.

## User Objectives

The user wants to have his own control panel and also wants to control his account using a software system and dispense paper dealings.

## General Constraints

This system works on computers and all types of processors are allowed to run this system.

# Functional Requirements

|  |  |
| --- | --- |
| Function code | 1.1 |
| Function Name | uploadAssignments() |
| Description | - check with an if statement if student has an account in the system and then return true  - check if the Teacher upload Assignment in the Course that student had taken  - save the string Assign on the system by inserting it in the database  - after doing the previous step check if the records are saved in the database or not by displaying a message box in the insertion to make sure. |
| Input | String Assign |
| Output | Assignment Stored in the database |
| Critically | It will make the student upload his Assignments |
| Dependencies with other requirements | Student must have an Account on the system and the system must check if the Assignment of this course is exist( ) |
| Pre-Condition | The database is connected correctly |
| Post-Condition | The Assignment is recorded on the Database system and it will be submitted |

|  |  |
| --- | --- |
| Function code | 2.1 |
| Function Name | MakeExams(Exams x) |
| Description | - check with an if statement if Teacher has an account and logged in the system  - save the exam on the system by inserting it (ExamName,ExamId,ExamDate) in the database(fn…)  - after that check if the exam is saved or not by displaying a confirmation message that is made successfully |
| Input | Exams x(ExamId,ExamName,,ExamDate) |
| Output | It will be stored in database  Exam will be formed |
| Critically | It will provide the teachers to make their Exams |
| Risks | Database may be failure |
| Dependencies with other requirements | It can’t happen if the teacher Account has not exist by (fn...) |
| Pre-Condition | The teacher has been logged in and ready for making actions |
| Post-Condition | The exam is added on the system and the numExams will be added by 1 |

|  |  |
| --- | --- |
| Function code | 2.2 |
| Function Name | MarkExams(Exams x) |
| Description | - check with an if statement if the exam exists and answered by the student in the system  - display the Exam x (ExamId,ExamName,,ExamDate)  - the Float ExamResult will be added by Teacher and it will be returned by this function |
| Input | Exams x(ExamResult) |
| Output | Exam Results |
| Critically | It will provide the teachers to mark their Exams |
| Risks | Database may be failure |
| Dependencies with other requirements | The Exam Must be exists by (fn 2.1) |
| Pre-Condition | The exam has not marked yet |
| Post-Condition | The exam results is recorded successfully |

|  |  |
| --- | --- |
| Function code | 2.3 |
| Function Name | addCourseMatrial(string CourseList) |
| Description | - check with an if statement if teacher has an account in the system and then return true  - choose the course (string CourseList) which the teacher wants to add materials in and upload the materials  - save the course material on the system by inserting it in the database  - after doing the previous step check if the records are saved in the database or not by displaying a message box in the insertion to make sure. |
| Input | The teacher choose course which want to add material on (string CourseList) |
| Output | Show stored course material in the database |
| Critically | This function will make the teacher add materials on the chosen course |
| Dependencies with other requirements | The teacher cannot add materials without having an account on the system and this teacher must be teaching this course |
| Pre-Condition | The database is connected correctly |
| Post-Condition | The Course Martial is recorded on the Database system and it will appear for the students |

|  |  |
| --- | --- |
| Function code | 3.1 |
| Function Name | setStudentsCourses(Course C, Student S) |
| Description | - check with an if statement if supervisor has an account in the system and then return true  - check with an if statement if student (S.studentID) has an account in the system and then return true  - get the CoursesList depends on Student level (S.Studentlv)  - add the student on this course  - save the student courses on the system by inserting it in the database  - after doing the previous step check if the records are saved in the database or not by displaying a message box in the insertion to make sure. |
| Input | - From object Student (int StudentID – int StudentLv)  - From object Course (string CourseList) |
| Output | Stored student courses in the database |
| Critically | This function will make the supervisor add courses for each student |
| Dependencies with other requirements | The supervisor cannot add courses without having an account on the system and studentID is exist on the database |
| Pre-Condition | The database is connected correctly |
| Post-Condition | The Courses for each student are recorded on the Database system and it will appear in the students’ schedule |

|  |  |
| --- | --- |
| Function code | 4.1 |
| Function Name | AddStudyFees(Student S) |
| Description | - check with an if statement if student (S.studentID) has an account in the system and then return true  - check if the student not pay the fees (S.feesState)  - change S.feesState to payed  - save the student fees state on the system by inserting it in the database  - after doing the previous step check if the records are saved in the database or not by displaying a message box in the insertion to make sure. |
| Input | - From object Student (int StudentID – int S.feesState) |
| Output | Stored student fees state in the database |
| Critically | This function will change the student fees state from unpaid to paid |
| Dependencies with other requirements | The accountant cannot change fees without having an account on the system and studentID is exist on the database and fees State is unpaid |
| Pre-Condition | The database is connected correctly |
| Post-Condition | The student payment state is recorded on the Database system and it will appear in the students’ schedule |

|  |  |
| --- | --- |
| Function code | 4.2 |
| Function Name | AddTeacher(teacherID) |
| Description | - check with an if statement if teacher (T.teacherID) doesn’t have an account in the system and then return true  -create new account for the new teacher(T.teacherID)  - after doing the previous step check if the records are saved in the database or not by displaying a message box in the insertion to make sure. |
| Input | From object Teacher(int teacherID-) |
| Output | The Teacher is stored in the database |
| Critically | This function will add a new teacher into the teachers list |
| Dependencies with other requirements | The admin cannot add a teacher that is already in the list |
| Pre-Condition | Database is connected correctly |
| Post-Condition | The new added teacher is recorded in the database system and will appear in the teachers list |

|  |  |
| --- | --- |
| Function code | 4.3 |
| Function Name | AddSupervisor(SupervisorID) |
| Description | - check with an if statement if Supervisor (Su.SupervisorD) doesn’t have an account in the system and then return true  -create new account for the new Supervisor (Su.teacherID)  - after doing the previous step check if the records are saved in the database or not by displaying a message box in the insertion to make sure. |
| Input | From object Supervisor(int SupervisorID-) |
| Output | The Supervisor is stored in the database |
| Critically | This function will add a new Supervisor into theSupervisors list |
| Dependencies with other requirements | The admin cannot add a Supervisor that is already in the list |
| Pre-Condition | Database is connected correctly |
| Post-Condition | The new added Supervisor is recorded in the database system and will appear in the Supervisors list |

|  |  |
| --- | --- |
| Function code | 3.2 |
| Function Name | AddStudent(StudentID) |
| Description | - check with an if statement if Student (S.StudentID) doesn’t have an account in the system and then return true  -create new account for the new Student (S.StudentID)  - after doing the previous step check if the records are saved in the database or not by displaying a message box in the insertion to make sure. |
| Input | From object Student(int StudentID-) |
| Output | The Student is stored in the database |
| Critically | This function will add a new Student into the Student list |
| Dependencies with other requirements | The Supervisor cannot add a Student that is already in the list |
| Pre-Condition | Database is connected correctly |
| Post-Condition | The new added Student is recorded in the database system and will appear in the Student list |

|  |  |
| --- | --- |
| Function code | 2.4 |
| Function Name | uploadQuiz(QuizID,DueDate) |
| Description | - check with an if statement if Teacher has an account in the system and then return true  - save the string Quiz on the system by inserting it in the database  - after doing the previous step check if the records are saved in the database or not by displaying a message box in the insertion to make sure. |
| Input | String Quiz- int QuizID- int DueDate |
| Output | Quiz Stored in the database |
| Critically | It will make the Teacher upload the Quiz |
| Dependencies with other requirements | Teacher must have an Account on the system |
| Pre-Condition | The database is connected correctly |
| Post-Condition | The Quiz is recorded on the Database system |

|  |  |
| --- | --- |
| Function code | 2.5 |
| Function Name | MarkQuiz(QuizID,QuizResult) |
| Description | - check with an if statement if the quiz exists and answered by the student in the system  - display the Quiz Q(QuizID,QuizdueDate)  - the Float QuizResult will be added by Teacher and it will be returned by this function |
| Input | Quiz Q(QuizResult) |
| Output | Quiz Results |
| Critically | It will provide the teachers to mark their Quizzes |
| Risks | Database may be failure |
| Dependencies with other requirements | The Quiz Must be existing by (fn 2.4) |
| Pre-Condition | The Quiz has not marked yet |
| Post-Condition | The Quiz results is recorded successfully |

|  |  |
| --- | --- |
| Function code | 4.4 |
| Function Name | Delete Teacher(Teacher T) |
| Description | - check with an if statement if Teacher (T.TeacherID) has an account in the system and then delete the record  - after doing the previous step check if the actions are made in the database or not by displaying a message box in the deletion to make sure. |
| Input | - From object Teacher(int TeacherID) |
| Output | Show that record has been deleted on database. |
| Critically | This function will delete the selected teacher by TeacherID |
| Dependencies with other requirements | The Admin cannot delete teacher without having an account on the system and Teacher is exist on the database and then drop the record. |
| Pre-Condition | The database is connected correctly |
| Post-Condition | The Teacher record which we select on database would not be available on database. |

|  |  |
| --- | --- |
| Function code | 4.5 |
| Function Name | DeleteSupervisor(Supervisor O) |
| Description | - check with an if statement if Teacher (O.SupervisorID) has an account in the system and then delete the record  - after doing the previous step check if the actions are made in the database or not by displaying a message box in the deletion to make sure. |
| Input | - From object Supervisor(int SupervisorID) |
| Output | Show that record has been deleted on database. |
| Critically | This function will delete the selected Supervisor by SupervisorID |
| Dependencies with other requirements | The Admin cannot delete Supervisor without having an account on the system and Supervisor is exist on the database and then drop the record. |
| Pre-Condition | The database is connected correctly |
| Post-Condition | The Supervisor record which we select on database would not be available on database. |

|  |  |
| --- | --- |
| Function code | 3.3 |
| Function Name | DeleteStudent(Student S) |
| Description | - check with an if statement if Student (S.StudenID) has an account in the system and then delete the record  - after doing the previous step check if the actions are made in the database or not by displaying a message box in the deletion to make sure. |
| Input | - From object Student(int StudenID ) |
| Output | Show that record has been deleted on database. |
| Critically | This function will delete the selected student by StudentID |
| Dependencies with other requirements | The Supervisor cannot delete student without having an account on the system and Student is exist on the database and then drop the record. |
| Pre-Condition | The database is connected correctly |
| Post-Condition | The Student record which we select on database would not be available on database. |

|  |  |
| --- | --- |
| Function code | 4.6 |
| Function Name | EditTeacher (Teacher T) |
| Description | - check with an if statement if Teacher(T.TeacherID) has an account in the system and then replace Teacher Salary with value received in database.  - after doing the previous step check if the actions are made in the database or not by displaying a message box in the update process to make sure. |
| Input | - From object Teacher(int TeacherID,int TeacherSalary) |
| Output | Show that record has been deleted on database. |
| Critically | This function will update the TeacherSalary by TeacherID |
| Dependencies with other requirements | The Teacher will edit salary if it TeacherID is exist. |
| Pre-Condition | The database is connected correctly |
| Post-Condition | The TeacherSalary record which we select on database would be updated on database. |

|  |  |
| --- | --- |
| Function code | 4.7 |
| Function Name | EditSupervisor (Supervisor O) |
| Description | - check with an if statement if Supervisor (O. SupervisorID) has an account in the system and then replace SupervisorSalary with value received in database.  - after doing the previous step check if the actions are made in the database or not by displaying a message box in the update process to make sure. |
| Input | - From object Supervisor(int SupervisorID,int SupervisorSalary) |
| Output | Show that record has been deleted on database. |
| Critically | This function will update the SupervisorSalary by Supervisor ID |
| Dependencies with other requirements | The Supervisor will edit salary if it Supervisor ID is exist. |
| Pre-Condition | The database is connected correctly |
| Post-Condition | The Supervisor record which we select on database would be updated on database. |

|  |  |
| --- | --- |
| Function code | 3.4 |
| Function Name | UpdateStudent (Student S) |
| Description | - check with an if statement if Student(S.Student) has an account in the system and then replace StdLevel with value received in database.  - after doing the previous step check if the actions are made in the database or not by displaying a message box in the update process to make sure. |
| Input | - From object Student(int StudentID,int StdLevel) |
| Output | Show that record has been deleted on database. |
| Critically | This function will update the StdLevel by StudentID |
| Dependencies with other requirements | The Supervisor will edit StdLevel if it Supervisor ID is exist. |
| Pre-Condition | The database is connected correctly |
| Post-Condition | The Student record which we select on database would be updated on database. |

|  |  |
| --- | --- |
| Function code | 2.6 |
| Function Name | DeleteAssignment(AssignmentID) |
| Description | -check if the teacher has an account and logged in in the system  - check with an if statement if Assignment (AssignmentID) Exists in the system and then delete the record  - after doing the previous step check if the actions are made in the database or not by displaying a message box in the deletion to make sure. |
| Input | - From object Assignment(int AssignmentID) |
| Output | Show that record has been deleted on database. |
| Critically | This function will delete the selected Assignment by AssignmentID |
| Dependencies with other requirements | The Teacher cannot delete Assignment without having an account on the system and Assignment is exist on the database and then drop the record. |
| Pre-Condition | The database is connected correctly |
| Post-Condition | The Assignment record which we select on database would not be available on database. |

|  |  |
| --- | --- |
| Function code | 2.7 |
| Function Name | viewSalary(Teacher T) |
| Description | -check if the teacher has an account and logged in in the system  - check with an if statement if Teacher (TeacherID) Exists in the system and then view the Salary record  - after doing the previous step check if the actions are made in the database or not by displaying a message box in the Viewing to make sure. |
| Input | - From object Teacher(int TeacherID,int TeacherSalary) |
| Output | Show the salary of teacher |
| Critically | This function will view the Salary by TeacherID |
| Dependencies with other requirements | The Teacher cannot view Salary without having an account on the system. |
| Pre-Condition | The database is connected correctly |
| Post-Condition | The Teacher will view salary. |

|  |  |
| --- | --- |
| Function code | 2.8 |
| Function Name | putExamsMark(Teacher T) |
| Description | -check if the teacher has an account and logged in in the system  - check with an if statement if Teacher (TeacherID) Exists in the system and then update the Examres record to each student  - after doing the previous step check if the actions are made in the database or not by displaying a message box in the Viewing to make sure. |
| Input | - From object Teacher(int TeacherID,int Examres) |
| Output | Put Examres by TeacherID by StudentID |
| Critically | This function will use TeacherID assign the Examres for each StudentID |
| Dependencies with other requirements | The Teacher cannot view Examres without having an account on the system. |
| Pre-Condition | The database is connected correctly |
| Post-Condition | The Teacher will put exam marks. |

|  |  |
| --- | --- |
| Function code | 1.2 |
| Function Name | ShowStudentDetails(StudentID) |
| Description | -check if the Student has an account and logged in in the system  - show student details  - after doing the previous step check if the actions are made in the database or not by displaying a message box in the deletion to make sure. |
| Input | - From object Student(int StudentID) |
| Output | Show that record has been Selected from the database. |
| Critically | This function will select Student details by StudentID |
| Dependencies with other requirements | The Student cannot show student details without having an account on the system and student details is exist on the database and then drop the record. |
| Pre-Condition | The database is connected correctly |
| Post-Condition | The student details record which we select on database would be available on database. |

|  |  |
| --- | --- |
| Function code | 1.3 |
| Function Name | ShowStudentCourses(StudentID) |
| Description | -check if the Student has an account and logged in in the system  - show student courses  - after doing the previous step check if the actions are made in the database or not by displaying a message box in the deletion to make sure. |
| Input | - From object Student(int StudentID) |
| Output | Show that record has been Selected from the database. |
| Critically | This function will select Student courses by StudentID |
| Dependencies with other requirements | The Student cannot show student courses without having an account on the system and student courses is exist on the database and then drop the record. |
| Pre-Condition | The database is connected correctly |
| Post-Condition | The student courses record which we select on database would be available on database. |

|  |  |
| --- | --- |
| Function code | 1.4 |
| Function Name | ShowStudentSchedule(StudentID) |
| Description | -check if the Student has an account and logged in in the system  - show student Schedule  - after doing the previous step check if the actions are made in the database or not by displaying a message box in the deletion to make sure. |
| Input | - From object Student(int StudentID) |
| Output | Show that record has been Selected from the database. |
| Critically | This function will select Student Schedule by StudentID |
| Dependencies with other requirements | The Student cannot show student schedule without having an account on the system and student schedule is exist on the database and then drop the record. |
| Pre-Condition | The database is connected correctly |
| Post-Condition | The student schedule record which we select on database would be available on database. |

|  |  |
| --- | --- |
| Function code | 4.8 |
| Function Name | ShowTeacherSalary(TeacherID) |
| Description | - check with an if statement if admin has an account in the system and then return true  - check with an if statement if teacher (T.teacherID) has an account in the system and then return true  - show Teacher salary  - after doing the previous step check if the actions are made in the database or not by displaying a message box in the deletion to make sure. |
| Input | - From object Teacher(int TeacherSalary) |
| Output | Show that record has been Selected from the database. |
| Critically | This function will select TeacherSalary by TeacherID |
| Dependencies with other requirements | The admin cannot show Teacher salary without having an account on the system and teacher salary is exist on the database and then drop the record. |
| Pre-Condition | The database is connected correctly |
| Post-Condition | The teacher salary record which we select on database would be available on database. |

|  |  |
| --- | --- |
| Function code | 3.5 |
| Function Name | setStudentLevel(StudentID, StudentLevel) |
| Description | - check with an if statement if supervisor has an account in the system and then return true  - check with an if statement if student (S.studentID) has an account in the system and then return true  -set StudentLevel to the value sent to the function  - save the student Level on the system by inserting it in the database  - after doing the previous step check if the records are saved in the database or not by displaying a message box in the insertion to make sure. |
| Input | - From object Student (int StudentID – int StudentLv) |
| Output | Stored student Level in the database |
| Critically | This function will make the supervisor set student level for each student |
| Dependencies with other requirements | The supervisor cannot set student level without having an account on the system and studentID is exist on the database |
| Pre-Condition | The database is connected correctly |
| Post-Condition | The student level for each student are recorded on the Database system and it will appear in the students details |

# Interface Requirements

## User Interfaces

Not Available.

### GUI

Not Available.

### CLI

We use the NetBeans to run the php and we use the SQL server as a database.

### API

Not Available.

### Diagnostics or ROM

Describes how to obtain debugging information or other diagnostic data.

## Hardware Interfaces

Describes interfaces to hardware devices.

## Communications Interfaces

Describes network interfaces.

## Software Interfaces

Describes any remaining software interfaces not included above.

# Performance Requirements

It should be fast smooth and

# Design Constraints

Specifies any constraints for the design team using this document.

## Standards Compliance

Not Available.

## Hardware Limitations

Any Device that has internet connection as our system is a web based and does not require hardware as hardware is cloud based.

## others as appropriate

1. **Other non-functional attributes**

## Security:

## in our system we care about security as we make a privilege for each user

## for example: we use PHP post method to check that the user is the right admin.

* 1. **Binary Compatibility**

Our system is compatible with all 32-bit or 64-bit operating systems as it is primarily based on the web interface as we use MySQL and php to create it.

* 1. **Reliability**
  2. **Maintainability**

Our system is can be maintained if there is any error happens because we depend on an MVC Model that it is easy to trace and maintain different error.

## Portability

Our system can easily be deployed on different platforms as it is web-based build with PHP and SQL server

## Extensibility

Our system uses SQL server so we can insert an unlimited number of users without any afraid of failures.

## Re-usability

Our system built in object-oriented methodologies so we can easily access the code and use it fast in different parts of the system and we have 6 main roles admin, supervisor, teacher, student, driver, and librarian.

## Application Affinity/Compatibility

Our system is compatible and can deal with different IDEs.

**Resource Utilization**

Our system is optimized to serve a lot of people at the same time without making any waste of resources.

* 1. **Serviceability**

Our system can be used as a service as he receives data from users and return with results depending on the action done by the user on the system.

* 1. **others as appropriate**

1. **Preliminary Object-Oriented Domain Analysis**

## Inheritance Relationships

Diagram

Description automatically generated

Figure 1: Inheritance Relations

## Class descriptions

This section presents a more detailed description of each class this information is in CRC Cards

### List of Superclasses:

1)Admin

2) Supervisor

3) Teacher

4) Student

5) Driver

6) Librarian

### List of Subclasses:

* 1. FAQ
  2. course group
  3. Exams
  4. Course

3.1) Project

3.2) Schedule

4.1) Bus

5.1) Book

5.2) library database

* + 1. **Purpose:**

1)Admin: manage everything related to teachers and supervisors

2)Supervisor: Manage students.

3)Teacher: Manage courses, assignments, quizzes and exams

4)Student: view, upload, download and submit assignments, quizzes and exams.

* + 1. **Attributes:**

2.1) int studentLevel in range from 0 to 3 for the 3 secondary years

2.2) string studentBook

2.3) Object student in range from 1 and up to 700 which is the school capacity

3.1) float Tsalary in range from 2000 and up to 4500.

3.2) object course in range from 1 and up to 4 courses

3.3) object schedule just 1 schedule.

4.1) float stdtResults in range from 0 and up to 100.

4.2) int stdLevel in range from 1 and up to 3.

4.3) int stdMobile must be 11 numbers.

4.4) object assignment

4.5) object course in range from 6 and up to 8.

4.6) object schedule just 1 schedule.

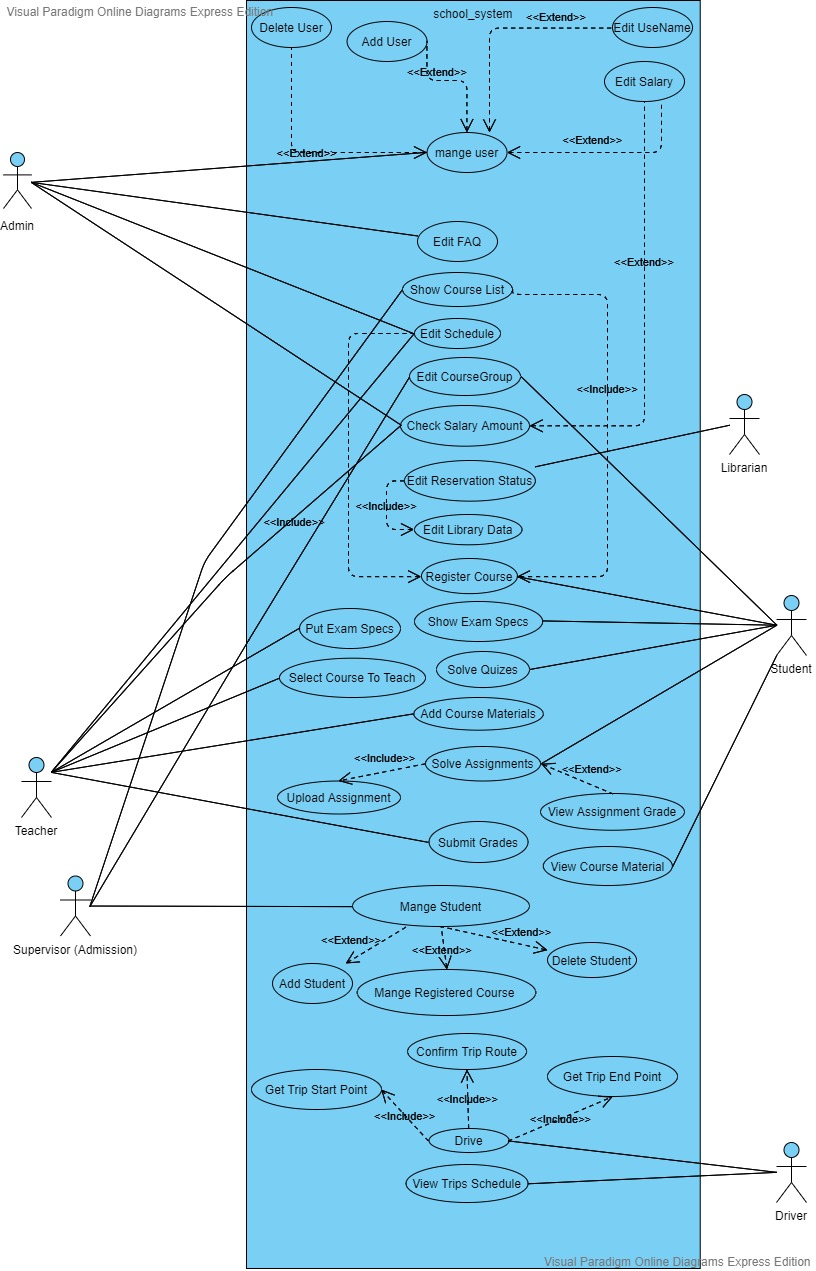
* + 1. **Collaborations:**

1. Admin will interact with teacher’s class and supervisors class to manage them by interacting with some functions for example (addTeacher , editTeacher, addSupervisor, editSupervisor,….etc).
2. Supervisor will interact with student’s class to manage them by interacting with some functions for example (add\_student, update\_student,….etc).
3. Teacher will interact with the database by uploading assignments, quizzes and exams and also marking them by using some functions for example (makeExam, markExam, MakeAssignment, MarkAssignment,…etc).
4. Student will interact with the data base to view, upload and download assignments, quizzes and exams by using some functions for example (uploadAssingment, downloadAssignments, uploadQuiz,….etc).

**8.2.6 Operations**

# Operational Scenarios

The admin can manage any user by either editing their data, adding a new user’s data and deleting user while being able to show the courses list, moreover the admin, the teacher the student and the librarian can view courses on the other hand student can solve assignments and quizzes. The librarian can edit the reservation status by going back to the library database. Finally, each driver will have a trip schedule that he will have to follow.



# Appendices

## Collected material

## C:\Users\marwa\AppData\Local\Microsoft\Windows\INetCache\Content.Word\WhatsApp Image 2020-11-15 at 22.04.27 (1).jpeg

1. **References**

**––––https://new.edmodo.com**