# Software Requirement Specification Document Educational Center

Alaa Mohamed, Fady Khaled, Zeyad Emad, Ammar Yasser

March 9 2018

# 1 Introduction

## 1.1 Purpose of this Document

This document will define an agreement between the customer of Educational Center and the developers Team, by formally outlining the functional requirements and non-functional requirements (performance constraints) for the System. These requirements form boundaries and constraints for the projects design process as well as providing a reference for the development of a test process.It will explain system constraints, interface and interactions with other external applications. This document is primarily intended to be proposed to a customer for its approval and a reference for developing the first version of the system for the development team

## 1.2 Scope of this Document

This software system will be a Web Application for an Educational Center. This system will be designed to maximize the Educational Center flexibility by providing capabilities for registering students in courses; documenting grading, transcripts, results of student tests and other assessment scores; building student schedules; tracking student attendance; and managing many other student related data needs in an Educational center. The software will facilitate communication between Students and staff members .The system has a user friendly and usable interface, supported by a well designed relational database.

## 1.3 Overview

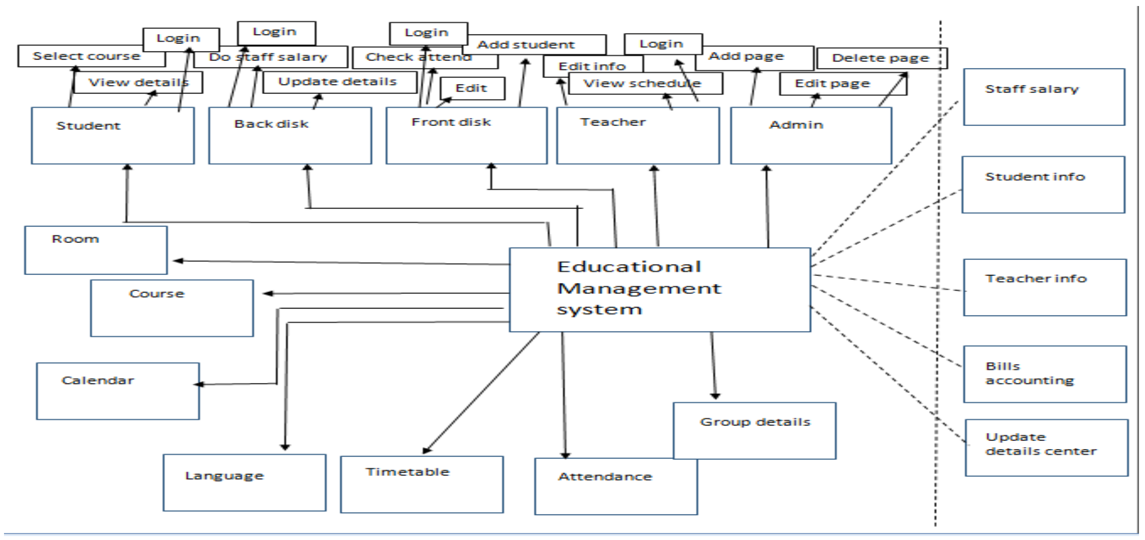
The next Section, the Overall Description section, of this document gives an overview of the functionality of the system. It describes the informal requirements and is used to establish a context for the technical requirements specification in the next chapter. The third Section, Requirements Specification section, of this document is written primarily for the developers and describes in technical terms the details of the functionality of the system. The fourth

Section provides a description of the different system interfaces. The fifth and sixth Sections provide performance requirements and design constraints. The seventh Section provides the non-functional attributes.The eighth Section deals with Preliminary Object-Oriented Domain Analysis which includes Inheritance Relationships and class descriptions.

## 1.4 Business Context

The System will save time, effort and will ensure an easy and flexible use. It will be designed to maximize the Educational Center flexibility by managing every thing in project will be computerized from setting students ,transcripts, schedules, and staff members which is going to reduce time as everything wasn't computerized before and it was a time wasting.

## 1.5 Block Diagram



# 2 General Description

## 2.1 Product Functions

### 2.1.1 Grading system

The system shall allow Back Disk to Add/update/delete grades for student and control student requirements.

### 2.1.2 Timetable

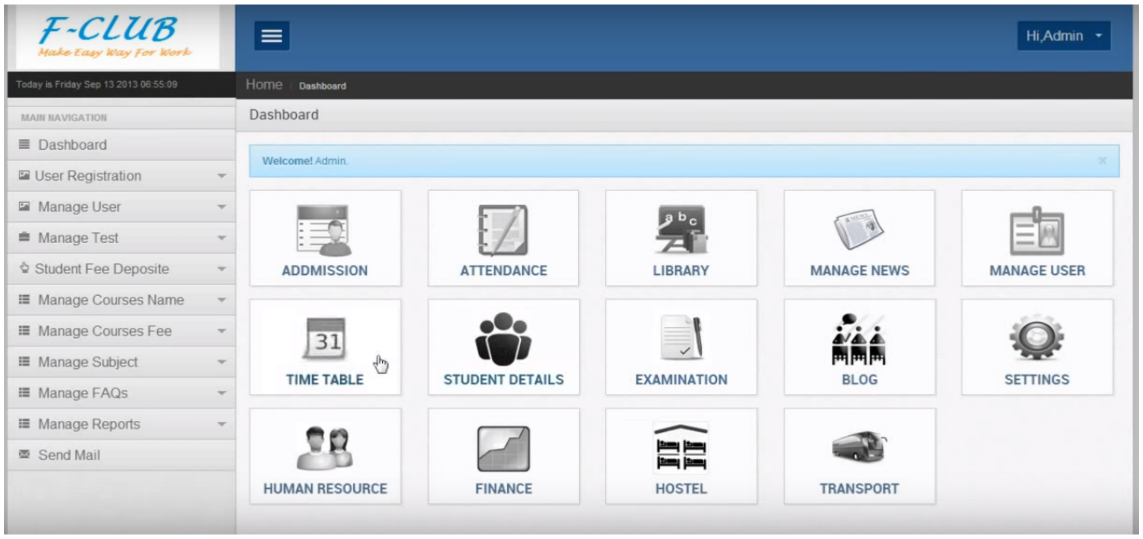
The system shall allow Back Disk to Add/update/delete courses with schedule containing time and room.

### 2.1.3 Attendance

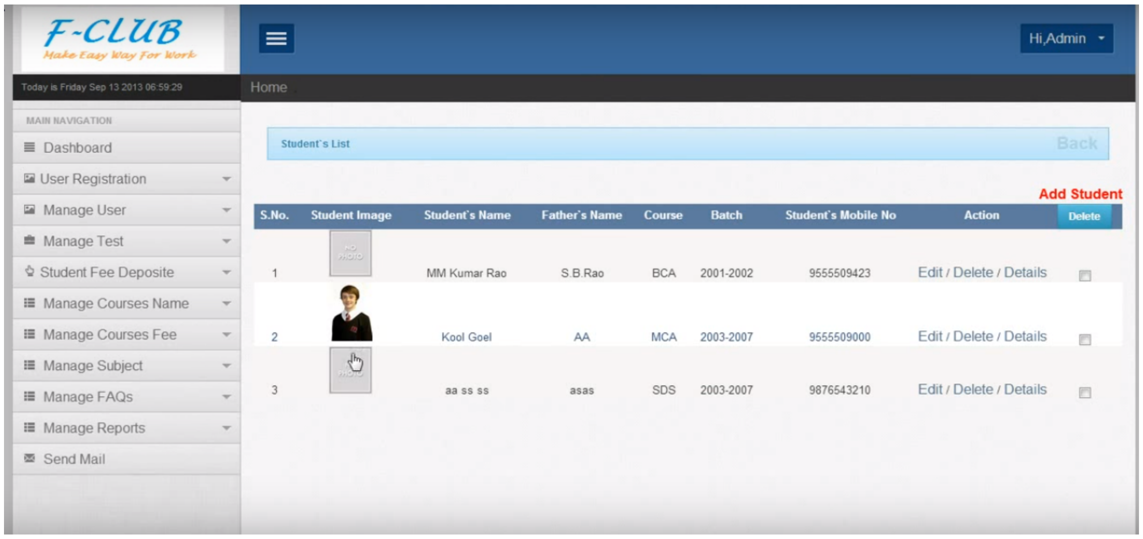
The system shall allow Front Disk to control daily basis students attendance.

## 2.2 Similar System Information

### 2.2.1 Figure 1



### 2.2.2 Figure 2



Student`s List

| S.No. | Student Image | Student`s Name | Father`s Name | Course | Batch | Student`s Mobile No | Action | Delete |
|---|---|---|---|---|---|---|---|---|
| 1 | NO PHOTO | MM Kumar Rao | S.B.Rao | BCA | 2001-2002 | 9555509423 | Edit / Delete / Details | ☐ |
| 2 | | Kool Goel | AA | MCA | 2003-2007 | 9555509000 | Edit / Delete / Details | ☐ |
| 3 | PHOTO | aa ss ss | asas | SDS | 2003-2007 | 9876543210 | Edit / Delete / Details | ☐ |

## 2.3 User Characteristics

There are five types of users that interact with the system: Back Disk, Front Disk and Administrator Each of these five types of users has different use of the system so each of them has their own requirements.
1-Admin
He is expected to manage his own system with provided schedules for student and staff members.
2- Back Disk
He is expected to be able to work with spreadsheets, databases and word processing. He is expected to be able to manage the database through forms provided in the system.
3- Front Disk
He is expected to fill in the forms provided in the system
4- Teacher
He is expected to have good communication with students and provide them with updated materials and lessons.
5- Student
He is expected to be able to use the lo-gin system provided in the system in order to Manage and view all his needs.

## 2.4 User Problem Statement

This system will save time for the Educational Center as they won't need to save their data in a lot of documents and papers to search in. because the system will allow them to manage all these requirements.Therefore, the system will help the center for making a backup which will be better from searching in a lot of documents and papers.They system also will help by providing a good security system for students grades and staff members salary which will prevent any changes in these information.

## 2.5 User Objectives

This system will provide an easy use for all users with a simple interface which helps managing everything related to Educational Center. It will help them save all data related to the student and will help them to get any reports that will help them track the student in their system with no need to search and analyze too much papers and documents to get this data, Also it will save too much time to get accurate results.

## 2.6 General Constraints

user name and password are used for the identification of users to lo gin. Only registered students and teachers will be authorized to use the system depending on permissions of each of them. Limited to HTTP. This system is working for single server.

# 3 Functional Requirements

## 3.1 Back-end Functions

## 3.2 Administrator API

### 3.2.1 Function Login

Code: F.01
Name: Login
Type: Functional
Critically: High
Input: User name,Password
Output: Redirect to specified web pages
Description: The user input his/her User name and password to check them in the database
Priority: 10/10
Expected risk: Database size, bugs,wrong User name or password
Preconditions: Open website
Post-condition: Fill session with the user data
Dependencies: none

### 3.2.2 Function Add page

Code: F.02
Nam: Add Page
Type: Functional
Critically: High
Input: user type, page code
Output: Pages are added to the database
Description: The admin adds the Page according to the member
Priority: 10/10
Expected risk: Database size, bugs
Preconditions: User must be logged in with admin account type
Post-condition: Pages are added to the new account according to account type
Dependencies: Fid.01

### 3.2.3 Function Update page

Code: F.03
Name: Update Page
Type: Functional
Critically: High
Input: user type, page code
Output: update static pages

Description: Admin edit pages from the database
Priority: 10/10
Expected risk: Database size, bugs
Preconditions: admin must be logged in
Post-condition: The password should be encrypted in the database
Dependencies: Fid.01

#### 3.2.4 Function Delete page

Code: F.04
Name: Delete Page
Type: Functional
Critically: High
Input: none
Output: Page is deleted
Description: Admin delete pages from the database
Priority: 10/10
Expected risk: Database size, bugs
Preconditions: Admin must be logged in
Post-condition: None
Dependencies: Fid.01

#### 3.2.5 Function Student add course

Code: F.05
Name: Select Course
Type: Functional
Critically: High
Input: Array of groups
Output: adding student to group
Description: Student selects his own courses and insert his group in database
Priority: 10/10
Expected risk: None
Preconditions: Student must be logged in.
Post-condition: None.
Dependencies: Fid.01,Fid.11

#### 3.2.6 Function View grades

Code: F.06
Name: View grades
Type: Functional
Critically: High

Input: none
Output: Students results
Description: Read results from database
Priority: 10/10
Expected risk: None
Preconditions: student must be logged in.
Post-condition: None.
Dependencies: Fid.01,Fid.28

### 3.2.7 Function View Timetable

Code: F.07
Name: View Timetable
Type: Functional
Critically: High
Input: none
Output: Showing schedule of students courses
Description: Student schedule
Priority: 10/10
Expected risk: Database size, bugs
Preconditions: Student must be logged in
Post-condition: None Dependencies: Fid.01,Fid.14,Fid.15,Fid.16

### 3.2.8 Function View requests

Code: F.08
Name: View requests
Type: Functional
Critically: High
Input: none
Output: view notification from teacher or admin
Description: Student check his news for any updates
Priority: 10/10
Expected risk: None
Preconditions: Student must be logged in
Post-condition: None.
Dependencies: Fid.01,Fid.28

### 3.2.9 Function Add new staff

Code: F.9
Name: Add Staff
Type: Functional
Critically: High

Input: staff details
Output: staff member is added
Description: Inserts a new staff member in database
Priority: 10/10
Expected risk: Database size, bugs
Preconditions: Back Disk must be logged in
Post-condition: None.
Dependencies: Fid.01

### 3.2.10 Function Add course

Code: F.10
Name: sub course
Type: Functional
Critically: High
Input: course details
Output: Course is added
Description: Back Disk adds course and save it to the database
Priority: 10/10
Expected risk: Database size, bugs
Preconditions: Back Disk must be logged in
Post-condition: None.
Dependencies: Fid.01

### 3.2.11 Function Add group

Code: F.11
Name: Sub group
Type: Functional
Critically: High
Input: course details
Output: group is added
Description: inserting new group in database
Priority: 10/10
Expected risk: Database size, bugs
Preconditions: Back Disk must be loggedin
Post-condition: None.
Dependencies: Fid.01,Fid.12,Fid.13,Fid.14

### 3.2.12 Function Select course

Code: F.12
Name: Select course

Type: Functional
Critically: High
Input: Array of courses
Output: Course
Description: choosing courses
Priority: 10/10
Expected risk: Database size, bugs
Preconditions: Back Disk must be logged in.
Post-condition: None.
Dependencies: Fid.01

#### 3.2.13 Function Select Teacher

Code: F.13
Name: Select teacher
Type: Functional
Critically: High
Input: Array of teachers
Output: Teacher
Description: choosing teachers
Priority: 10/10
Expected risk: Database size, bugs
Preconditions: Back Disk must be logged in
Post-condition: none
Dependencies: Fid.01

#### 3.2.14 Function Select room

Code: F.14
Name: Select room
Type: Functional
Critically: High
Input: Array of rooms
Output: Room
Description: choosing rooms
Priority: 10/10
Expected risk: None
Preconditions: Back Disk must be logged in.
Post-condition: None.
Dependencies: Fid.01

### 3.2.15 Function Select day

Code: F.15
Name: Select Day
Type: Functional
Critically: High
Input: Array of days
Output: Day
Description: choosing day
Priority: 10/10
Expected risk: None
Preconditions: Back Disk must be logged in
Post-condition: None.
Dependencies: Fid.01

### 3.2.16 Function Select time

Code: F.16
Name: Select time
Type: Functional
Critically: High
Input: Array of time
Output: time
Description: choosing time
Priority: 10/10
Expected risk: None
Preconditions: Back Disk must be logged in.
Post-condition: None.
Dependencies: Fid.01

### 3.2.17 Function View Student groups

Code: F.17
Name: View Student Groups
Type: Functional
Critically: High
Input: none
Output: view all student groups
Description: get student groups from database
Priority: 10/10
Expected risk: none
Preconditions: Back Disk must be Logged in
Post-condition: None.
Dependencies: Fid.01.Fid.11

### 3.2.18 Function Edit group

Code: F.18
Name: Edit group
Type: Functional
Critically: High
Input: group details
Output: update group details
Description: update group details in datbase
Priority: 10/10
Expected risk: Hacking
Preconditions: Back Disk must be logged in.
Post-condition: None.
Dependencies: Fid.01,Fid.11

### 3.2.19 Function View Staff

Code: F.19
Name: View Staff
Type: Functional
Critically: High
Input: none
Output: view all staff members
Description: retrieve staff members from database
Priority: 10/10
Expected risk: Database size, bugs
Preconditions: Back Disk must be logged in
Post-condition: None
Dependencies: Fid.01,Fid.09

### 3.2.20 Function Edit Staff

Code: F.20
Name: Edit Staff
Type: Functional
Critically: High
Input: Staff details
Output: Update staff details
Description: Update staff details in database
Priority: 10/10
Expected risk: Database size, bugs
Preconditions: Back Disk must be logged in
Post-condition: None
Dependencies: Fid.01,Fid.09

#### 3.2.21 Function Make Staff Salary

Code: F.21
Name: Make Staff Salary
Type: Functional
Critically: High
Input: staff attendance details per day
Output: salary for staff per day
Description: add salary per day
Priority: 10/10
Expected risk: Database size, bugs
Preconditions: Back Disk must be logged in
Post-condition: None
Dependencies: Fid.01,Fid.09

#### 3.2.22 Function View Total salary

Code: F.22
Name: View Total salary
Type: Functional
Critically: High
Input: none
Output: view total salaries
Description: Retrieve total salary from Database
Priority: 10/10
Expected risk: Database size, bugs
Preconditions: Back Disk must be logged in
Post-condition: None
Dependencies: Fid.01,Fid.09,Fid.21

#### 3.2.23 Function View Room

Code: F.23
Name: get buttons
Type: Functional
Critically: High
Input: none
Output: view all rooms
Description: retrieve all rooms from database
Priority: 10/10
Expected risk: Database size, bugs
Preconditions: Back Disk must be logged in
Post-condition: None
Dependencies: Fid.01,Fid.14

### 3.2.24 Function Add new room

Code: F.24
Name: Add New room
Type: Functional
Critically: High
Input: Room Details
Output: Room
Description: insert new room in database
Priority: 10/10
Expected risk: Database size, bugs
Preconditions: Back Disk must be logged in
Post-condition: None
Dependencies: Fid.01

### 3.2.25 Function View Room Schedule

Code: F.25
Name: View Room Schedule
Type: Functional
Critically: High
Input: none
Output: View schedule of room
Description: retrieve schedule of room from database
Priority: 10/10
Expected risk: Database size, bugs
Preconditions: Back Disk must be logged in
Post-condition: None
Dependencies: Fid.01,Fid.14

### 3.2.26 Function View Requests

Code: F.26
Name: View Requests
Type: Functional
Critically: High
Input: none
Output: view all requests from other members
Description: view requests from database
Priority: 10/10
Expected risk: Database size, bugs
Preconditions: Back Disk must be logged in
Post-condition: None
Dependencies: Fid.01,Fid.27

### 3.2.27 Function Send Request

Code: F.27
Name: Send Request
Type: Functional
Critically: High
Input: Request Details
Output: Request is sent
Description: insert new request in database
Priority: 10/10
Expected risk: Database size, bugs
Preconditions: Back Disk must be logged in
Post-condition: None
Dependencies: Fid.01

### 3.2.28 Function Add grades

Code: F.28
Name: Add grades
Type: Functional
Critically: High
Input: Grades details
Output: Grades
Description: Insert grades in database
Priority: 10/10
Expected risk: Database size, bugs
Preconditions: Back Disk must be logged in
Post-condition: None
Dependencies: Fid.01

### 3.2.29 Function View Schedule

Code: F.29
Name: View Schedule
Type: Functional
Critically: High
Input: none
Output: All courses with time
Description: Retrieve all courses with time from database
Priority: 10/10
Expected risk: Database size, bugs
Preconditions: Teacher must be logged in
Post-condition: None
Dependencies: Fid.01,Fid.12,Fid.13,Fid.15,Fid.16

#### 3.2.30 Function Edit Personal Information

Code: F.30
Name: Change details
Type: Functional
Critically: High
Input: teacher details
Output: Teacher information has been updated
Description: Teacher information has been updated in database
Priority: 10/10
Expected risk: Database size, bugs
Preconditions: Teacher must be logged in
Post-condition: None
Dependencies: Fid.01,Fid.09

#### 3.2.31 Function Submit Extra

Code: F.31
Name: Submit Extra
Type: Functional
Critically: High
Input: none
Output: Request sent to Back Disk
Description: Insert request in database
Priority: 10/10
Expected risk: Database size, bugs
Preconditions: Teacher must be logged in
Post-condition: None
Dependencies: Fid.01

#### 3.2.32 Function Contact Admin

Code: F.32
Name: Contact Admin
Type: Functional
Critically: High
Input: Request details
Output: Request sent to Admin
Description: Insert request in database
Priority: 10/10
Expected risk: Database size, bugs
Preconditions: Teacher must be logged in
Post-condition: None
Dependencies: Fid.01

### 3.2.33 Function View all Students

Code: F.33
Name: View All students
Type: Functional
Critically: High
Input: none
Output: Students
Description: Retrieve students from database
Priority: 10/10
Expected risk: Database size, bugs
Preconditions: Front Disk must be logged in
Post-condition: None
Dependencies: Fid.01,Fid.17

### 3.2.34 Function Add new Student

Code: F.34
Name: Add new Student
Type: Functional
Critically: High
Input: Student details
Output: Student is added
Description: Student is added in database
Priority: 10/10
Expected risk: Database size, bugs
Preconditions: Front Disk must be logged in
Post-condition: None
Dependencies: Fid.01

### 3.2.35 Function Select course

Code: F.35
Name: sub course
Type: Functional
Critically: High
Input: none
Output: Students
Description: Retrieve students from database
Priority: 10/10
Expected risk: Database size, bugs
Preconditions: Front Disk must be logged in
Post-condition: None
Dependencies: Fid.01,Fid.12

### 3.2.36 Function View grades

Code: F.36
Name: View grades
Type: Functional
Critically: High
Input: none
Output: Student results
Description: Retrieve student resutls from database
Priority: 10/10
Expected risk: Database size, bugs
Preconditions: Front Disk must be logged in
Post-condition: None
Dependencies: Fid.01,Fid.28

### 3.2.37 Function Edit student details

Code: F.37
Name: Edit student details
Type: Functional
Critically: High
Input: Student details
Output: Update student details
Description: Update student details in database
Priority: 10/10
Expected risk: Database size, bugs
Preconditions: Front Disk must be logged in
Post-condition: None
Dependencies: Fid.01,Fid.34

### 3.2.38 Function Delete Student

Code: F.38
Name: Delete Student
Type: Functional
Critically: High
Input: none
Output: Student is deleted
Description: Delete student from database
Priority: 10/10
Expected risk: Database size, bugs
Preconditions: Front Disk must be logged in
Post-condition: None
Dependencies: Fid.01,Fid.34

### 3.2.39 Function View amount

Code: F.39
Name: View amount
Type: Functional
Critically: High
Input: none
Output: view all amounts
Description: Retrieve student amount from database
Priority: 10/10
Expected risk: Database size, bugs
Preconditions: Front Disk must be logged in
Post-condition: None
Dependencies: Fid.01,Fid.41

### 3.2.40 Function Add Student amount

Code: F.40
Name: Add Student amount
Type: Functional
Critically: High
Input: amount details
Output: Amount
Description: insert amount in database
Priority: 10/10
Expected risk: Database size, bugs
Preconditions: Front Disk must be logged in
Post-condition: None
Dependencies: Fid.01

### 3.2.41 Function Add attendance

Code: F.41
Name: Add attendance
Type: Functional
Critically: High
Input: Attendance details
Output: Student attended or didn't attend
Description: insert student attendance in database
Priority: 10/10
Expected risk: Database size, bugs
Preconditions: Front Disk must be logged in
Post-condition: None
Dependencies: Fid.01

## 4 Interface Requirements

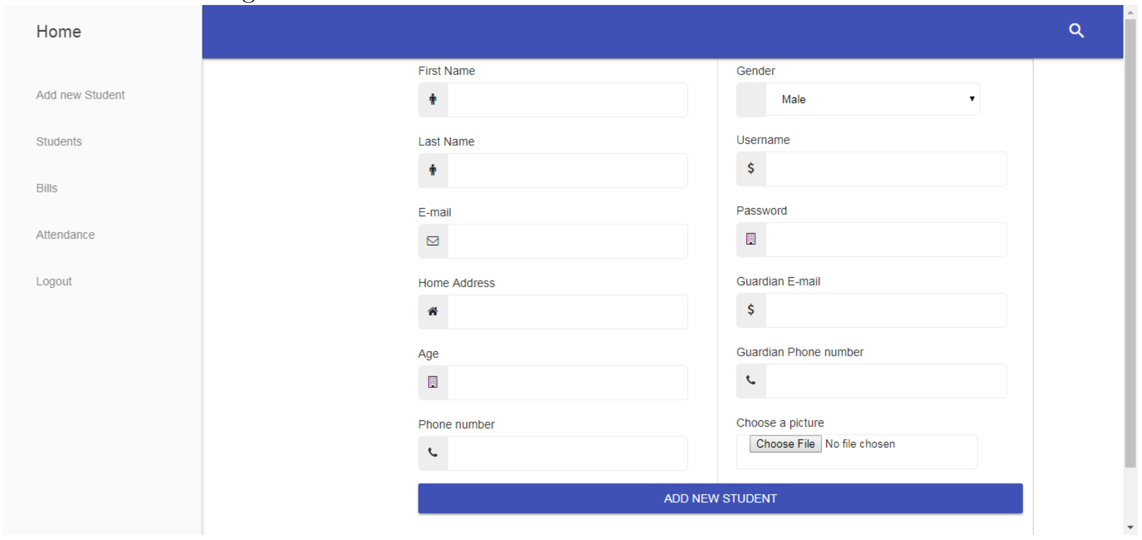
This section provides a detailed description of all inputs into and outputs from the system. It also gives a description of the hardware, software and communication interfaces and provides basic prototypes of the user interface.

### 4.1 User Interfaces

The user interface for the software shall be compatible to any browser such as Internet Explorer, Mozilla or Netscape Navigator by which user can access to the system. A first-time customer should see the register form in order to have account on the system, he can also see some information about the Educational Center.

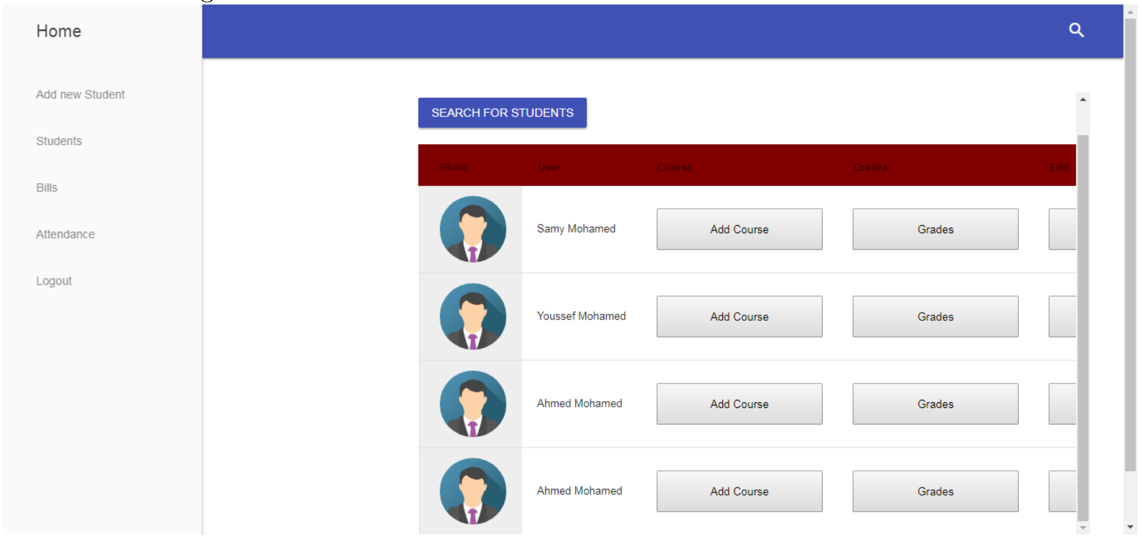
#### 4.1.1 Figure 1

Add new student Page



### 4.1.2 Figure 2

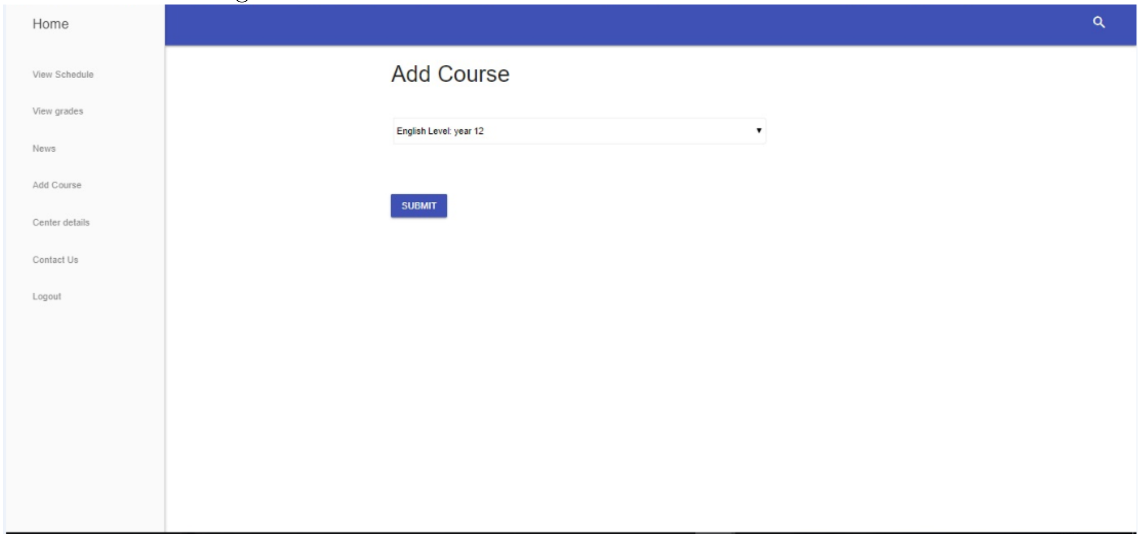
View Student Page



| Photo | User | Course | Grades | Edit |
|---|---|---|---|---|
| | Samy Mohamed | Add Course | Grades | |
| | Youssef Mohamed | Add Course | Grades | |
| | Ahmed Mohamed | Add Course | Grades | |
| | Ahmed Mohamed | Add Course | Grades | |

### 4.1.3 Figure 3

Add new Course Page

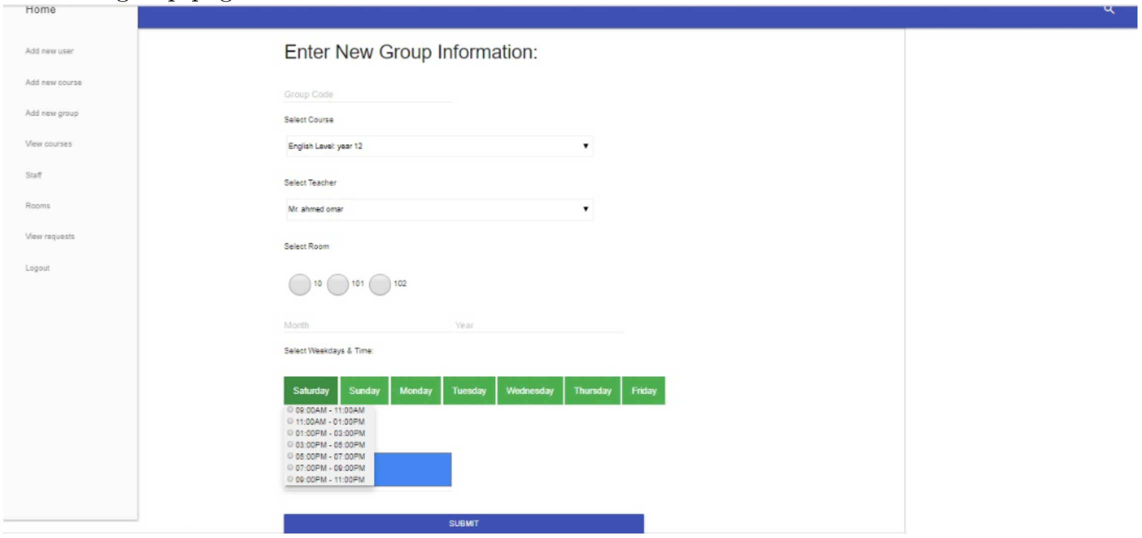


Add Course

English Level: year 12

### 4.1.4 Figure 4

Add new group page



Enter New Group Information:

Group Code

Select Course

English Level: year 12

Select Teacher

Mr. ahmed omar

Select Room

10 101 102

Month Year

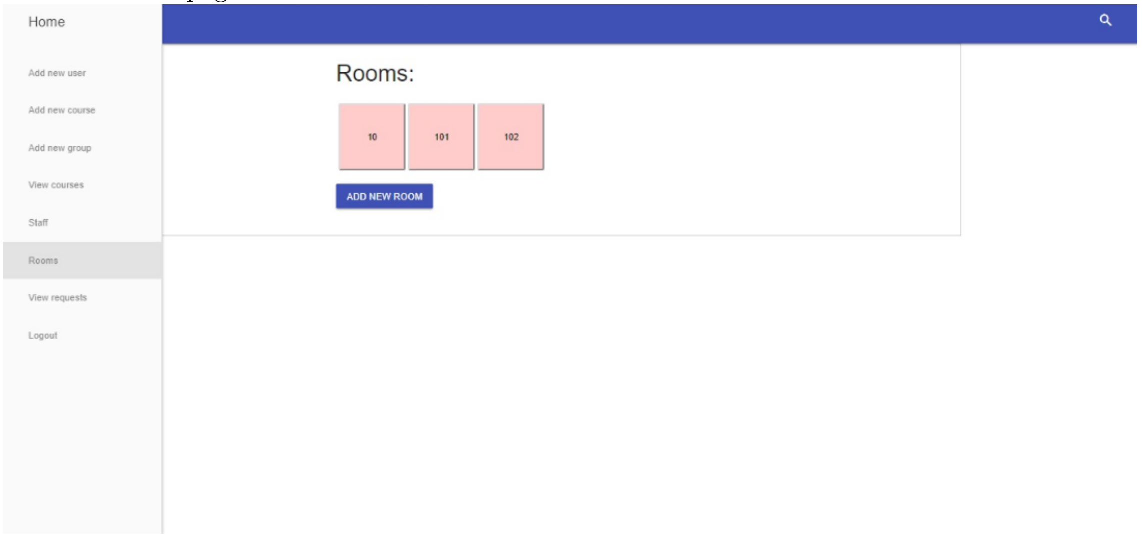
Select Weekdays & Time:

Saturday Sunday Monday Tuesday Wednesday Thursday Friday

09:00AM - 11:00AM
11:00AM - 01:00PM
01:00PM - 03:00PM
03:00PM - 05:00PM
05:00PM - 07:00PM
07:00PM - 09:00PM
09:00PM - 11:00PM

### 4.1.5 Figure 5

View all rooms page



Rooms:

10 101 102

## 4.2 Hardware Interfaces

Since the application must run over the internet, all the hardware shall require to connect internet will be hardware interface for the system. As for e.g. WAN LAN, Ethernet Cross-Cable.

### 4.3 Communication Interfaces

The communication between the different parts of the system is important since they depend on each other. However, in what way the communication is achieved is not important for the system and is therefore handled by the underlying operating systems. Client (customer) on Internet will be using HTTP protocol.Client (system user) on Internet will be using HTTP protocol.

### 4.4 Software Interfaces

Client on Internet Web Browser, Operating System (any) Web Server 000web host, Operating System (any) Data Base Server MySQL, Operating System (any) Development End HTML, JAVASCRIPT, AJAX, BOOTSTRAP, MYSQL, PHP, OS (Windows)

## 5 Performance Requirements

The requirements in this section provide a detailed specification of the user interaction with the software and measurements placed on the system performance. The system shall be based on web and has to be run from a web server. The system shall take initial load time depending on internet connection strength .The performance shall depend upon hardware components of the client/customer.

## 6 Design Constraints

This section includes the design constraints on the software caused by the hardware.

### 6.1 Hard drive space

TAG: Hard Drive Space
GIST: Hard drive space.
SCALE: The applications need of hard drive space.
METER: MB.
MUST: No more than 20 MB.
PLAN: No more than 15 MB.
WISH: No more than 10 MB.
MB: DEFINED: Megabyte

### 6.2 Application memory usage

TAG: Application Memory Usage
GIST: The amount of Operate System memory occupied by the application.
SCALE: MB.

METER: Observations done from the performance log during testing
MUST: No more than 20 MB.
PLAN: No more than 16 MB
WISH: No more than 10 MB
Operate System: DEFINED: The mobile Operate System which the application is running on.
MB: DEFINED: Megabyte

# 7 Other non-functional attributes

## 7.1 Security

TAG: Communication Security
GIST: Security of the communication between the system and server.
SCALE: The messages should be encrypted for log-in communications, so others cannot get user-name and password from those messages.
METER: Attempts to get user-name and password through obtained messages on 1000 log-in session during testing.
MUST: 100percent of the Communication Messages in the communication of a log-in session should be encrypted.
Communication Messages: Defined: Every exchanged of information between client and server.
TAG: Encryption and Decryption
GIST: The security of protecting sensitive data for users of the system.
SCALE: The back-end servers shall never display a users sensitive data such as passwords. It shall always be encrypted with special characters representing typed characters.
METER: Measurements obtained on 1000 hours of usage during testing.
MUST: 100percent of the time.

## 7.2 Reliability

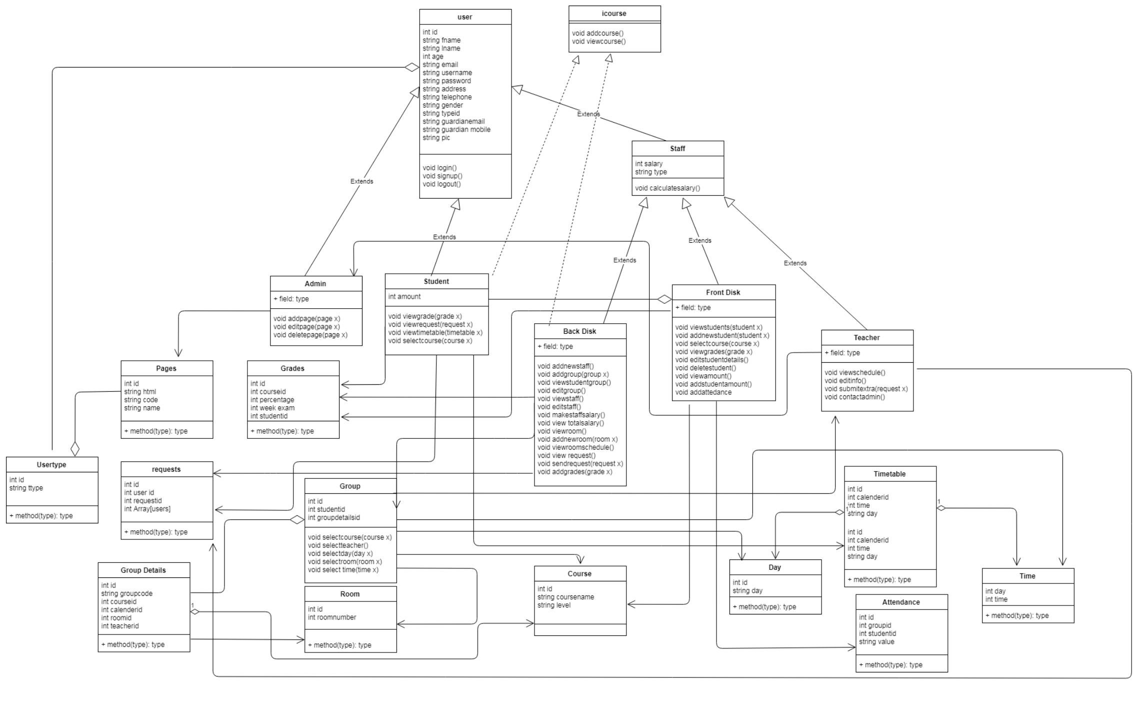
TAG: System Reliability
GIST: The reliability of the system.
SCALE: The reliability that the system gives the right result on a search.
METER: Measurements obtained from 1000 searches during testing.
MUST: More than 98 percent of the searches.
PLAN: More than 99percent of the searches.
WISH: 100percent of the searches.

## 7.3 Maintainability

TITLE: Application extendibility
DESC: The application should be easy to extend. The code should be written in a way that it favors implementation of new functions.
RAT: In order for future functions to be implemented easily to the application.
DEP: none
TITLE: Application testability
DESC: Test environments should be built for the application to allow testing of the applications different functions.
RAT: In order to test the application.
DEP: none

# 8 Preliminary Object-Oriented Domain Analysis

## 8.1 Inheritance Relationships



## 8.2 Class Descriptions

### 8.2.1 User

Abstract class
List of Super classes: None.
List of Sub classes: Administrator, Back Disk , Front Disk, Teacher and Student
Purpose: Main class has users information.
Collaborations: None.
Attributes:
user id : int
first name : string
last name : string
gender : string
Address : string
user type : int
user name : string
email : string
password : string
phone number : string
Age : string
Guardian email : string
Guardian mobile : string

Operations :
Login (user name: string, password: string);
Logout ();

### 8.2.2 Student

List of Super classes: User
List of Sub classes: None.
Purpose: The class has Student data and what they do.
Collaborations: This class Associates with project class.
Attributes: The class inherits all attributes from super class
Operations: Register(first name: string, last name: string, user name: string, email: string, password: string, phone: string, Address:string,); (select course (), View grades (),view timetable(), view requests());

### 8.2.3 Back Disk

List of Super classes: User
List of Sub classes: None
Purpose: The class has customers data and what they do.
Collaborations: The class associates with class course,teacher,room,day,group,request)
Attributes: The class inherits all attributes from super class.

Operations:( add new staff , add course(),add group,select course, select teacher(), select room(), select day(), select time(), view student group(), edit group(), view staff(), edit staff(), make staff salary(),view total salary(),view room(),add new room(), view room schedule(),view request(),send request(), add grades())

#### 8.2.4 Administrator

List of Super classes: User
List of Sub classes: None
Purpose: The class has customers data and what they do.
Collaborations: The class associates with class pages)
Attributes: The class inherits all attributes from super class.
Operations: (Add page(), Edit page(), Delete page())

#### 8.2.5 Front Disk

List of Super classes: User
List of Sub classes: None
Purpose: The class has customers data and what they do.
Collaborations: None.
Attributes: The class inherits all attributes from super class.
Operations: (view all students(), add new student(), select course(), view grades(), edit student details(), delete student(), view amount(), add student amount(), add attendance())

#### 8.2.6 Teacher

List of Super classes: User
List of Sub classes: None
Purpose: The class has customers data and what they do.
Collaborations: None.
Attributes: The class inherits all attributes from super class.
Operations:(View Schedule() , edit info(), submit request(), contact admin())

### 8.3 time table

List of Super classes: none
List of Sub classes: None
Purpose: The class has timetable data and what they contain.
Collaborations: This class Aggregate with time and day
Attributes: int id,int calender id,int time,string day.

## 8.4 calender

List of Super classes: none
List of Sub classes: None
Purpose: The class has calender data and what they contain.
Collaborations: This class Aggregate with timetable
Attributes: int id,group details id,string month,int year.

## 8.5 day

List of Super classes: none
List of Sub classes: None
Purpose: The class has day data
Collaborations: none
Attributes: int id,string day

## 8.6 time

List of Super classes: none
List of Sub classes: None
Purpose: The class has time data
Collaborations: none
Attributes: int id,int time

## 8.7 message

List of Super classes: none
List of Sub classes: None
Purpose: The class has message data
Collaborations: none
Attributes: int id,string message

## 8.8 course

List of Super classes: none
List of Sub classes: None
Purpose: The class has course data
Collaborations: none
Attributes: int id,course name,string level

## 8.9 user type pages

List of Super classes: none
List of Sub classes: None
Purpose: The class has use types data
Collaborations: none
Attributes: int id,int page id,int user type id

## 8.10 attendance

List of Super classes: none
List of Sub classes: None
Purpose: The class has attendance data
Collaborations: none
Attributes: int id,int group id ,int student id,string value

## 8.11 Room

List of Super classes: none
List of Sub classes: None
Purpose: The class has Room data
Collaborations: none
Attributes: int id,int room number

## 8.12 add extra value

List of Super classes: none
List of Sub classes: None
Purpose: The class has extra values data
Collaborations: none
Attributes: int id,int user id ,int extra details id,string value

## 8.13 user extra details

List of Super classes: none
List of Sub classes: None
Purpose: The class has user details
Collaborations: This class aggregate with add extra value
Attributes: int id,string name ,int user type id

## 8.14 user type

List of Super classes: none
List of Sub classes: None
Purpose: The class has types of users
Collaborations: none
Attributes: int id,string type

## 8.15 group

List of Super classes: none
List of Sub classes: None
Purpose: The class has group details data
Collaborations: this class aggregate with group details
Attributes: int id,int student id

## 8.16 group details

List of Super classes: none
List of Sub classes: None
Purpose: The class has group details data
Collaborations: this class aggregate with group details
Attributes: int id,string group code, int course id,int calender id, int room id, int teacher id

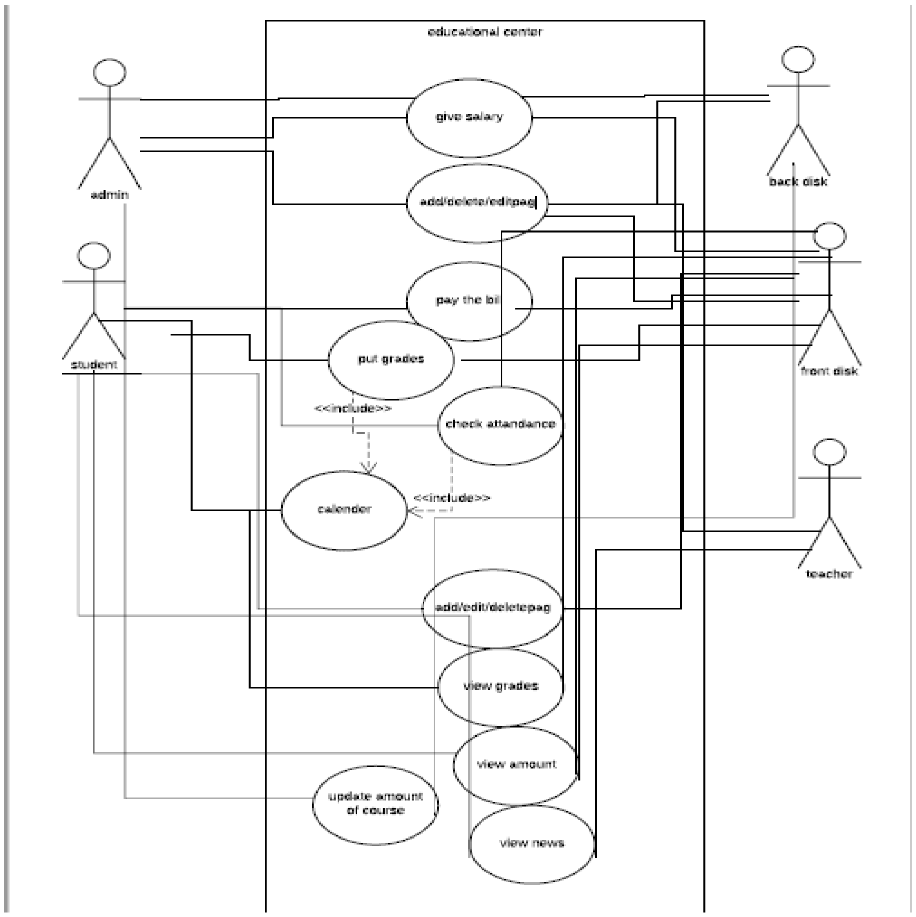
## 8.17 request

List of Super classes: none
List of Sub classes: None
Purpose: The class has request data
Collaborations: this class associates with back disk, teacher
Attributes: int id,int user id

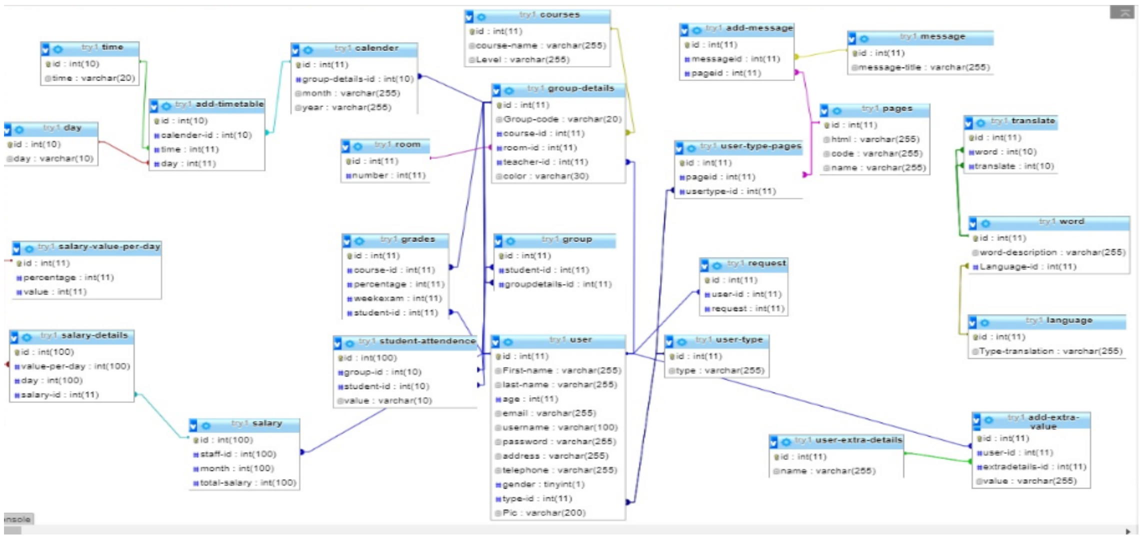
## 8.18 pages

List of Super classes: none
List of Sub classes: None
Purpose: The class has all pages data
Collaborations: this class aggregate with message
Attributes: int id,string html, string code,string name

## 9 Use Case Diagram



## 10 Data Base Tables



## 11 Operational Scenarios

### 11.1 Scenario for managing project

The student will login in using his user name and password then he will be redirected to his homepage then selects his courses with time table.
The Back Disk receives all the data entered from the student and manage them.
The Back disk provide schedule with rooms and time for teachers and student. and also manages staff members salary
The Front disk will login with his user name and password adding new students and taking their attendance daily basis.
The teacher will login with his user name and password receiving his schedule.
Finally, All the data for the students, teachers and all staff members is saved on the system no need for a lot of documents and papers.

## 12 Appendices

### 12.1 Definitions, Acronyms, Abbreviations

User: Someone who interacts with the mobile phone application.
Admin/Administrator: System administrator who is given specific permission for managing and controlling the system.
Back Disk : a person who has most of the important features in the system
Front Disk: Any person who has interaction with the system who is not a developer.

Teacher: interacts with the mobile phone application.
Customers: The owners of projects who interact with the system.
System: The web application which present special facilities for the company.
DESC: Description
RAT: Rational
DEP: Dependency TAG: A unique, persistent identifier contained in a PLanguage statement .
GIST: A short, simple description of the concept contained in a PLanguage statement .
SCALE : The scale of measure used by the requirement contained in a PLanguage statement .
METER: The process or device used to establish location on a SCALE contained in a PLanguage statement .
MUST: The minimum level required to avoid failure contained in a PLanguage statement .
PLAN: The level at which good success can be claimed contained in a PLanguage statement .
WISH: A desirable level of achievement that may not be attainable through available means contained in a PLanguage statement .
HTTP: Hypertext Transfer Protocol. Its a service protocol.
HTML: HTML (Hypertext Markup Language) is the set of markup symbols or codes inserted in a file intended for display on a World Wide Web browser page.
PHP: PHP is a server scripting language, and a powerful tool for making dynamic and interactive Web pages.
AJAX: AJAX stands for Asynchronous JavaScript and XML. In a nutshell, it is the use of the XMLHttpRequest object to communicate with server-side scripts.
MySQL: MySQL is an open source relational database management system. Information in a MySQL database is stored in the form of related tables.

## 13 References

Software Engineering 9th edition, Sommervillie, Addison Wesle