|  |
| --- |
| Close-up image showing the leaf-sides of two oversized books side-by-side on a bookshelf, with additional books in soft focus background |
| File Management System  Synopsis |
| |  |  |  | | --- | --- | --- | | Ajeya Paul | [Date] | [Course title] | |

Table of Contents

[Introduction & OBJECTIve 3](#_Toc344539141)

[INTRODUCTION 3](#_Toc344539142)

[Objective 3](#_Toc344539143)

[PROJECT CATEGORY 3](#_Toc344539144)

[Hardware and Software Specification 3](#_Toc344539145)

[Hardware Requirement 4](#_Toc344539146)

[Software Requirement 4](#_Toc344539147)

[REQUIREMENTS AND ANALYSIS 4](#_Toc344539148)

[Problem Definition 4](#_Toc344539149)

[Existing System 4](#_Toc344539150)

[Documents maintained 6](#_Toc344539151)

[Work To Be Done 7](#_Toc344539152)

[Requirements Specification 7](#_Toc344539153)

[Functional Requirements 7](#_Toc344539154)

[technical specification 10](#_Toc344539155)

[Planning and Scheduling 10](#_Toc344539156)

[Gantt chart 10](#_Toc344539157)

[Tracking Gantt 11](#_Toc344539158)

[Pert chart (Network Diagram) 12](#_Toc344539159)

[Scope of the Solution 12](#_Toc344539160)

[Analysis 13](#_Toc344539161)

[Context Diagram 13](#_Toc344539162)

[Data Flow Diagram 13](#_Toc344539163)

[Level 0 DFD 13](#_Toc344539164)

[Level 1 DFD 14](#_Toc344539165)

[Level 2 DFD 16](#_Toc344539166)

[E-R Diagram 16](#_Toc344539167)

[Class Diagram 20](#_Toc344539168)

[Database & Table Details 20](#_Toc344539169)

[COMPLETE DATA STRUCTURE 22](#_Toc344539170)

[Module Description 22](#_Toc344539171)

[School Management Server 22](#_Toc344539172)

[School Management Client 23](#_Toc344539173)

[School Management Database 23](#_Toc344539174)

[estimation 23](#_Toc344539175)

[Data Structure 24](#_Toc344539176)

[Implementation Methodology 27](#_Toc344539177)

[List of Reports 27](#_Toc344539178)

[sECURITY MECHANISM 27](#_Toc344539179)

[FUTURE SCOPE AND FURTHER REQUIREMENTS 28](#_Toc344539180)

[bIBLIOGRAPHY 28](#_Toc344539181)

# Introduction& OBJECTIVE

## Introduction

External storage device is an eminently useful tool for all computer users. The tiny devices fit in your pocket, but can store and transport gigabytes of your data. A USB drive is useful for moving files and folders from one computer to another, as well as serving as a backup device for your important files. Modern operating systems treat the devices as "Plug and Play" hardware, recognizing them as soon as they are plugged into a port and making them available for immediate use. We use different types of storage medium for different types of devices. Here is some Advantages of external storage:

* External storage devices provides additional storage other than that available in computer.
* Data can be transported easily from one place to another.
* It is useful to store software and data that is not needed frequently.
* External storage also works as data backup.
* This back up may prove useful at times such as fire or theft because important data is not lost.

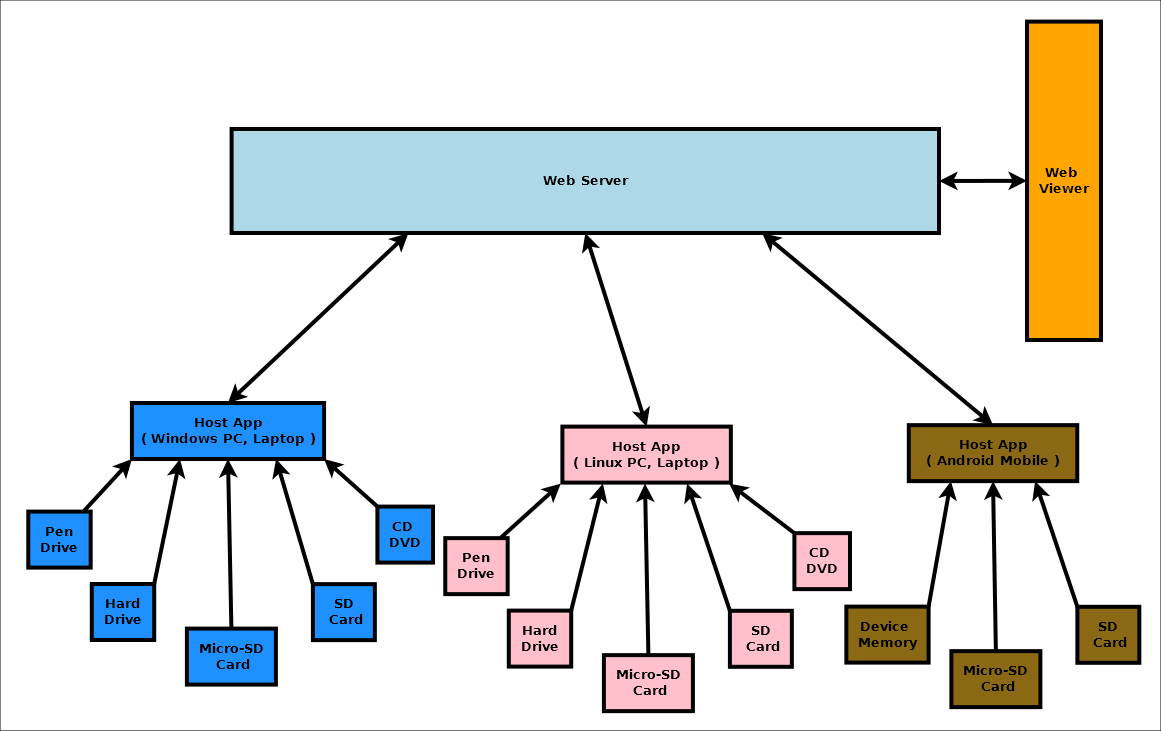
Let’s count how many storage devices anyone owns now. It will be at least more than ten. Do you know which of your device contains what content? The answer is mostly no. We need a centralized solution for storing information about the files we have. Cloud based storage is costly. Why can't we turn our own storages devices into a connected personal cloud system?  So we need to store information about the files we have and access them from the server whenever we need them.

File Management System (will be referred as FMS in this document) is versatile and complete end-to-end File management software. Unlike the traditional Windows Explorer this software is aimed to perform additional features to improve the user accessibility besides all basic features. The software can be used to access files both locally and remotely via network connection. The software will also provide various features such as Duplicate file handling, File Merging/Splitting, File permission modification, Metadata access etc.



The main components of this project are:

* In this project there will be a web based server which will store information about files stored in user's machine and storage devices.
* The clients (windows, Linux, android) will analyze the file systems and upload the information to web server.
* There will be a web based viewer for viewing the information.



## Objective

Clients will work as host app which can detect storage devices like micro SD, SD card, pen drive and hard drives. Whenever storage devices are connected to host apps, host app will scan the file system of the storage device silently and update the file information in the server along with storage device unique Id.

The main features of the file management system are:

* Keep track of files stored in various storage device owned by the user in a single place.
* Easy to search for a particular file in the fms viewer and then retrieve file from specific device.
* Detect duplicate file stored in various devices.
* Stream file from any of the connected devices via web interface on demand
* Searching and sorting file information
* Share files via social networking sites and cloud services
* Backup important files

# PROJECT CATEGORY

This software will follow Object Oriented Programming Paradigm and use below mentioned areas:

OOP Language: C#

RDBMS: SQLite

Networking: TCP/IP

Applications: Mobile Systems

# Hardware and Software Specification

## For Mobile:

### Hardware Requirement

* **Phone Memory/SD Card Memory :** 300 MB of available hard disk space
* **RAM :** 128 MB or higher
* **Processor :** 484 MHz or faster

### Software Requirement

* Windows Phone 08
* Android Honey Comb or Higher

## For Desktop

### Hardware Requirement

* Disk Space: 50 Mb of available disk space.
* RAM: 256 MB
* Processor: 1 GHz dual core

### Software Requirement

* Windows XP, Windows 07 Operating System.
* Microsoft .NET framework 3.5. / .NET framework 4
* Ubuntu
* Dia for Drawing and modeling
* Microsoft office word for documentation

# REQUIREMENTS AND ANALYSIS

## Problem Definition

### Existing System

How many storage devices anyone owns now? It will be at least more than ten. Do you know which of your device contains what content? The answer is mostly no. We need a centralized solution for storing information about the files we have. Cloud based storage is costly. Why can't we turn our own storages devices into a connected personal cloud system?  So we need to store information about the files we have and access them from the server whenever we need them. The existing system allows user to share their personal data on the web through various cloud based applications but they do not allow us to manage all our devices’ storages in one place. The existing ones, on the other hand, are not as user friendly as well. Sometimes they create confusion while logging in to the web server from other devices.

### our target

Using our application, the users can browse data of all his devices from one place with a very organized way. They can store and synchronize data of their desktop, laptop and/or smart phone. They can access those files with a predefined username and password from a web server which will show the details to the user using a web view. They can sort the data in groups and they can store the data to the external devices as well. Here are some key features we are going to implement:

* It should be able to browse phone memory, SD card storage.
* It will allow sorting and searching based on file type such as image file, video file, text file etc.
* It will allow archiving file and extracting from archive. It can be password protected as well.

### Work To Be Done

We will incorporate the above mentioned workflow of Windows Phone File Management System in a automatic computerized way.

## Requirements Specification

### Functional Requirements

#### ADD DEVICE

**Introduction**

The user can add their external storage device to file management system database.

**Input**

File management will take device ID.

**Processing**

The app will add the entire file attribute to file management system database.

**Output**

File management system database add the external storage device.

#### VIEW DEVICE

**Introduction**

The user can view their external storage device from file management system database.

**Input**

File management will recognize device ID and display all File content from different devices.

**Processing**

The Management system will display the entire external storage device’s id and show details.

**Output**

The user can browse all the files and folders of the storage.

#### bROWSE FILES

**Introduction**

The user can browse their file shared over network.

**Input**

Open the application and click on browse SD card files.

**Processing**

The app will search for the SD card attached in the phone and show them to the user just like a computer.

**Output**

The user can browse all the files and folders of the SD card.

#### Browse Phone memory

**Introduction**

Users can browse their phone memory from the application.

**Input**

Users open the application and click on browse phone memory.

**Processing**

The application searches for the phone memory structure and asks for permission from the phone to share the information to the user.

**Output**

Users can see all the files, folders of the SD card.

#### Security

**Introduction**

The application will be password protected.

**Input**

Users click on the application and provides predefined password to use the application.

**Processing**

The application takes the password input and compares it with the predefined password stored in SQLite database.

**Output**

Only after providing a valid password, the user gets access to the application.

#### read pdf files

**Introduction**

The application should be able to read PDF files from the Windows phone.

**Input**

The user browses and opens a PDF file stored in phone memory or SD card.

**Processing**

The application asks for permission from the mobile OS to use their PDF reading utility.

**Output**

The PDF gets opened and the user can read it.

#### Archived file reader

**Introduction**

The application can read and extract an archived file from inside the mobile.

**Input**

The user will browse and select an archived file to be extracted somewhere inside the SD card or phone memory.

**Processing**

The application extracts the archived file at the predefined location.

**Output**

The user gets the files extracted from archived format to a normal file.

#### Store data to cloud

**Introduction**

The user will be able to store the selected data to the skydrive account.

**Input**

The user clicks on store to cloud option and chooses a file to be kept at the cloud.

**Processing**

The application uses the mobile’s online backup utility to store the data to cloud.

**Output**

The data gets stored to the cloud so the user is able to use them from anywhere, a desktop or a laptop or from other mobile phone using an internet connection.

#### lock individual files

**Introduction**

The application should be able to lock individual files as per users’ instruction.

**Input**

The user taps on a file and chooses lock the file.

**Processing**

The application makes the file locked and asks for a predefined password if someone attempts to open that particular file.

**Output**

Only with a valid password, a user can open the file.

#### sorting of files

**Introduction**

The application should be able to sort files inside a folder.

**Input**

The user browses inside a folder and chooses to sort all files inside it.

**Processing**

The application recognizes the formats of the files and sorts them alphabetically.

**Output**

The user gets all files sorted by file types.

#### seareching of files

**Introduction**

The application should be able to search a file by its name from the phone memory and the SD card.

**Input**

The user writes name of a file inside the search text field and clicks on search.

**Processing**

The application looks for all the related data similar to the file name inside the phone and SD card memory.

**Output**

The user gets all the files similar to the name he provided inside the text field.

#### data sharing

**Introduction**

The application should be able share data through usb cable, Bluetooth or wifi.

**Input**

The user taps on a file and chooses share (using Bluetooth/wifi/usb cable)

**Processing**

The application instantly searches for availability of the device with which the file is to be shared and attempts to send the data to the device.

**Output**

The data gets shared as per users’ requirement.

### non Functional Requirements

* **Efficiency**:

It will be efficient as it reduces manual labor and searching.

* **Backup**:

Data could be stored to online storage.

* **Documentation**:

The application will have users’ manual pdf inside the help section.

* **Maintainability**:

It is designed such a way that it can be maintained with minimal effort.

* **Performance**:

The response time of file manager will be very fast. So it will be efficient enough to cater the user.

* **Privacy**:

The data will be encrypted and the user data will not be shared with third party without proper authentication.

* **Looks & Feels**: Should have very attractive looks and feels to make the user happy even by opening it.

ESCMS will use secure connection and enhanced security measures to protect data.

* **Usability**:

It will be very user friendly and usable by any person with minimal computer knowledge.

### Technical specification

#### For mobile app

* **Front End/ GUI Tools:** XAML (Extensible Application Markup Language)
* **IDE:** Visual Studio Express 2012 for Windows Phone
* **Framework:** Microsoft .NET 4.0
* **Database:** Sqlite.
* **SDK:** Windows Phone SDK 8.0.
* **Cloud Technology:** SkyDrive

#### For desktop app

* **Front End/ GUI Tools:** Windows Presentation Framework (WPF)
* **IDE:** Visual Studio 2010
* **Framework:** Microsoft .NET 4.0
* **Database:** MySQL
* **Database Tool:** MySQL workbench CE
* **Operating Systems**: Windows XP, Windows 7
* **Cloud Technology**: Google Drive, Dropbox, SkyDrive

## Planning and Scheduling

### Gantt chart



### Tracking Gantt



### Pert chart (Network Diagram)



# Scope of the Solution

## for Mobile App

Following features could be attached to the app in future to make it even more useful and a complete system manager:

1. It could be made as a system hardware tracker that shows processor/RAM status (used and free).
2. Battery remaining and temperature.
3. Synchronization with Google Drive.
4. Synchronization with Drop Box.

## for Desktop App

1. Using phone data for the computer through a USB cable.
2. Mobile OS updating and reinstallation of application/OS.
3. Installation of various apps in mobile from the desktop app through usb cable.

# Analysis

## Context Diagram

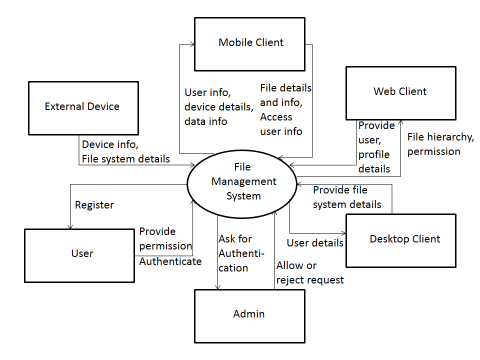


Figure :Context Diagram

## Data Flow Diagram

### Level 0 DFD

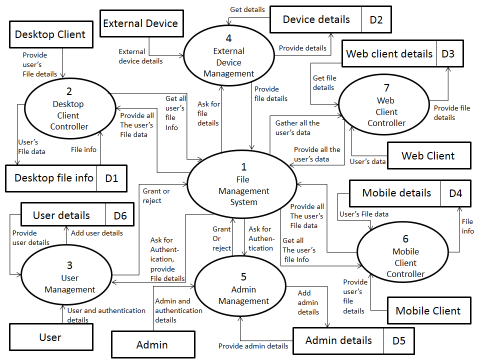


Figure :0 level DFD

### Level 1 DFD

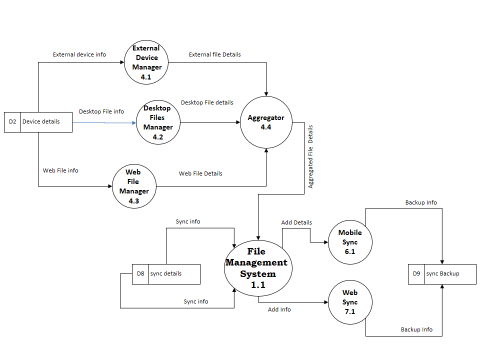


Figure :1 level DFD

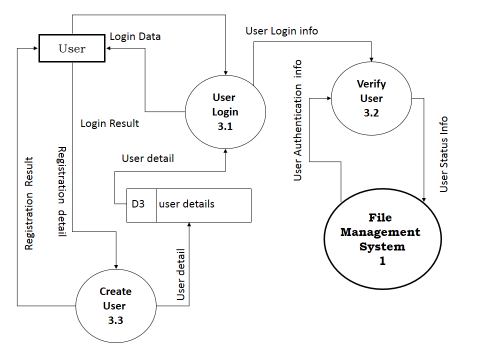


Figure :1 level DFD

### Level 2 DFD



Figure :2 level DFD

## E-R Diagram

We will design a RDBMS for School Management System. The entities and their attributes are listed below. Attributes in Bold letter is the unique key.

|  |  |
| --- | --- |
| **Entities** | **Attributes** |
| Student | **Student\_ID**, Student\_DOB, Student\_Name, Student\_Parent\_Name, Student\_Address, Student\_Admission\_Date, Student\_Course\_Name, Student\_Contact |
| Account | **Transaction\_ID**, Transaction\_Amount, Transaction\_Type, Transaction\_Reason,Account\_Balance |
| Admin | **Staff\_ID**, Staff\_Name, Staff\_Permission\_Level, Staff\_Address, Staff\_Admission\_Date, Staff\_Course\_Name, Staff\_Contract\_details, Staff\_Join\_Date, Staff\_Email, Staff\_Role |
| Books | **Book\_ID**, Book\_Name, Book\_Author, Purchase\_Date, Book\_Status, Book\_Description,Purchase\_Amount |
| Faculty | **Faculty\_ID**, Faculty\_Name, Faculty\_Address, Faculty\_Join\_Date, Faculty\_Course\_Under, Faculty\_Contact\_Details, Faculty\_Salary\_Details |
| Course | **Course\_ID**, Course\_Faculty, Course\_Name, Required\_Qualification, Course\_Fees, Course\_Admission\_Date, Students\_Under, Course\_Description |

**Relationship between Entities:**

School Management System has Courses 1 : N

School Management Systemhas Students 1 : N

School Management System has Faculties1 : N

School Management System has Admin 1 : 1

Studenthas Attendance 1 : 1

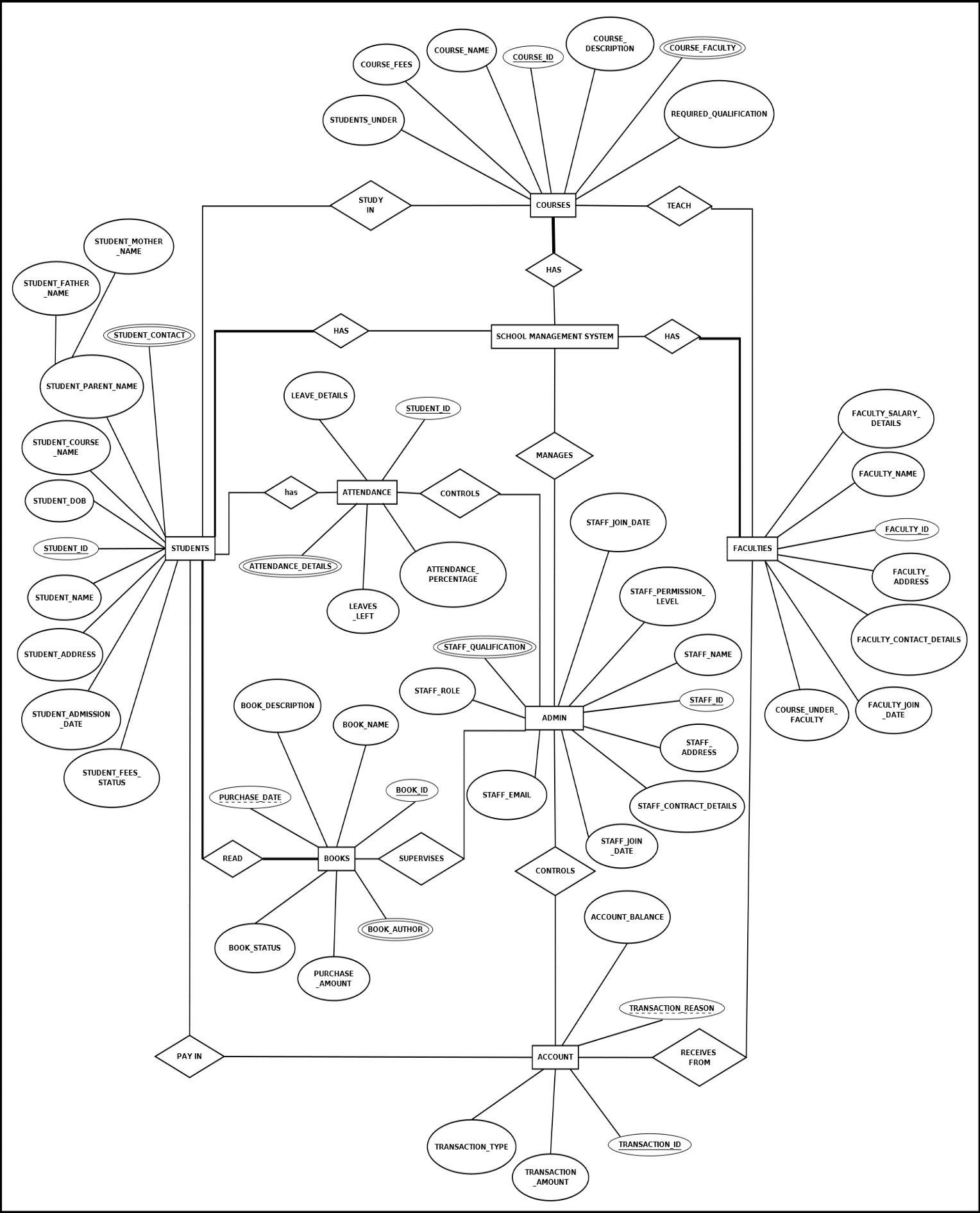
AdminChecksAttendance 1 : 1

StudentsreadsBooks M : N

StudentspaysAccount 1 : 1

AdminControlsAccount 1 : 1

Students Studies in Course N:1



## Class Diagram



# Database & Table Details

The database used for this software is called **smsdb**. A screenshot from the MySQl workbench is given below. It shows the tables and its columns. The first row is the primary key.



# COMPLETE DATA STRUCTURE

## Module Description

File Management System is divided in to following modules:

|  |
| --- |
| File Management System Web Controller |
| Android Client |
| Windows 8 Client |
| Windows 7 Client |
| Ubuntu (Linux) Client |
| Storage Device Controller |
| Server REST API |
| FMS Search Engine |
| Web Sync Handler |
| Web Viewer Client |

School Management System is divided three main modules such as:

1. School Management Server
2. School Management Client
3. School Management Database

### School Management Server

School Management server is a singleton server designed provide services for school management system. It controls various activities required for the school management system. To manage these activities it has several sub modules such as:

1. Admission Management
2. Student Management
3. Faculty Management
4. Course Management
5. Attendance & Leave Management
6. Library Management
7. Accounts Management
8. Administration Management

### School Management Client

School Management System will provide two different clients for the convenience of the user. Desktop client is for doing bulk activities and faster tasks. Web client will allow instant access from anywhere and anytime.

### School Management Database

School Management System will have a unified database for storing all the information. It can be a networked database or a database situated in the server machine.

## Estimation



## Data Structure

|  |
| --- |
| **StudentInfo** |
| publicclassStudentInfo  {  publicstring id { get; set; }  publicstring name { get; set; }  publicstring bloodGroup { get; set; }  publicstring phone { get; set; }  publicstring address { get; set; }  publicDateTime admissionDate { get; set; }  publicDateTime dob { get; set; }  } |

|  |
| --- |
| **CourseInfo** |
| publicclassCourseInfo  {  publicstring id { get; set; }  publicstring eventTitle { get; set; }  publicDateTime eventDoe { get; set; }  publicstring eventVenue { get; set; }  publicstring eventGoal { get; set; }  } |

|  |
| --- |
| **ExpenseInfo** |
| publicclassExpenseInfo  {  publicstring id { get; set; }  publicstring purpose { get; set; }  publicDateTime doe { get; set; }  publicstring expensed\_by { get; set; }  publicdouble amount { get; set; }  } |

|  |
| --- |
| **FundInfo** |
| publicclassFundInfo  {  publicstring id { get; set; }  publicstring wellwisher\_name { get; set; }  publicstring contact { get; set; }  publicDateTime dod { get; set; }  publicstring received\_by { get; set; }  publicdouble amount { get; set; }  } |

|  |
| --- |
| **FacultyInfo** |
| publicclassFacultyInfo  {  publicstring id { get; set; }  publicstring name { get; set; }  publicDateTime doj { get; set; }  publicstring address { get; set; }  publicstringphone { get; set; }  public List<string> qualifications { get; set; }  publicList<string> courses { get; set; }  } |

|  |
| --- |
| **AdmissionInfo** |
| publicclassAdmissionInfo  {  publicstring id { get; set; }  publicstring name { get; set; }  publicstringpurpose { get; set; }  publicint ageLimit { get; set; }  publicstringVenue { get; set; }  publicstringcoordinatorPhone { get; set; }  publicstring admittedAddress { get; set; }  publicDateTime examDate { get; set; }  publicDateTime resultDate { get; set; }  publicList<string> subjects { get; set; }  publicintnumberOfSeats { get; set; }  } |

|  |
| --- |
| **BookInfo** |
| publicclassBookInfo  {  publicstring id { get; set; }  publicDateTime issueDate { get; set; }  publicList<string> author { get; set; }  publicList<string> tag { get; set; }  } |

|  |
| --- |
| **TransactionInfo** |
| publicclassTransactionInfo  {  publicstring id { get; set; }  publicstring name { get; set; }  publicdouble amount { get; set; }  publicstring type { get; set; }  publicstring reason { get; set; }  publicstring receivedFrom { get; set; }  publicstring receivedBy { get; set; }  publicDateTime date { get; set; }  } |

## Implementation Methodology

* Object Oriented Programming methodology will be adopted and c sharp will be used as programming language.
* .NET framework will be used to develop desktop and mobile app.
* User interface development will be done using WPF (Windows Presentation Foundation).
* Relational DBMS MySQL will be used to implement & execute SQL query to database.
* Agile Software Development model will be used while developing this software.

# SECURITY MECHANISM

* For mobile app, password protection will used for the entire app.
* For mobile app, password protection will be used for single items, like a video or an image.
* For desktop pp, password protection will be incorporated so that the phone back up stays protected.

# FUTURE SCOPE AND FURTHER REQUIREMENTS

* The application provides only skydrive cloud storage back up. We could develop Google drive and dropbox cloud storage back up as well.
* It could be really useful if the app could provide a video and image file locker as well.

# bIBLIOGRAPHY

* <http://en.wikipedia.org>
* <http://msdn.microsoft.com/en-us/>
* <http://www.microsoft.com/en-us/default.aspx>
* <http://www.codeplex.com/>
* <http://stackoverflow.com/>
* <http://www.codeguru.com/>
* [http://www.w3schools.com](http://www.w3schools.com/)
* [www.mysql.org](http://www.mysql.org)
* IT Professionals and other Windows Phone users.