|  |
| --- |
| 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.

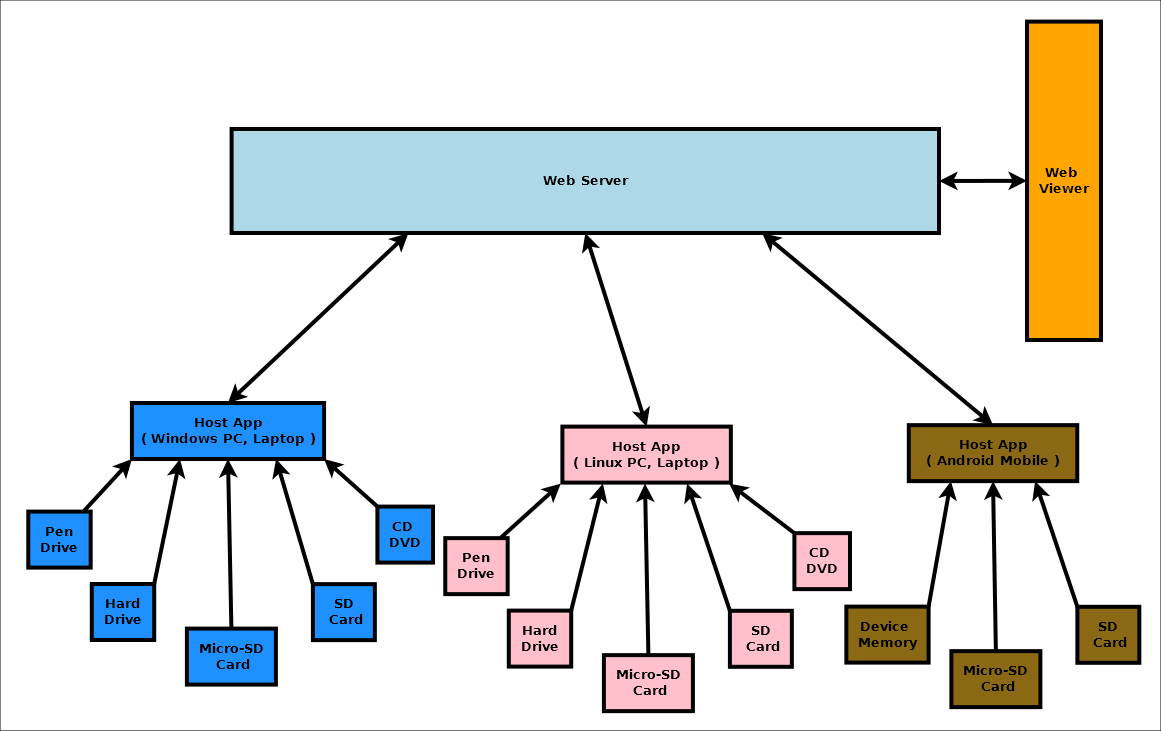
Let’s count how many storage devices anyone owns now. It will be at least more than ten. Do you know which of your device contains what content? The answer is mostly no. We need a centralized solution for storing information about the files we have. Cloud based storage is costly. Why can't we turn our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

File Management System (will be referred as FMS in this document) is standalone software for managing files in various storage devices.



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/host apps (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

Modern operating systems treat the storage devices as "Plug and Play" hardware, recognizing them as soon as they are plugged into a port and making them available for immediate use. 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#, Typescript

RDBMS: SQLite

Networking: TCP/IP

Applications: Desktop Systems , Web Application , Personal Cloud

# 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 7
* Ubuntu 12.04
* 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

We use different types of storage medium for different types of devices. Here are some Advantages of external storage:

* External storage devices provide 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.

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 access information about the data of all his storage devices from one place with a very organized way. They can store and synchronize data of their desktop, laptop and/or smart phone. The main intension is to utilize personal storage devices, avoid duplicity of files in several devices, easy to find which storage device has the desired data.

### Work To Be Done

We need to implement several clients (Host App) and server as given below:

1. **Windows Host App**: Windows OS has an API called **FileSystemWatcher.** I will use FileSystemWatcher to watch for changes in a specified directory. You can watch for changes in files and subdirectories of the specified directory. You can create a component to watch files on a local computer, a network drive, or a remote computer.
2. **Ubuntu Linux Host App**: The QFileSystemWatcher class provides an interface for monitoring files and directories for modifications. QFileSystemWatcher monitors the file system for changes to files and directories by watching a list of specified paths. Call addPath() to watch a particular file or directory. Multiple paths can be added using the addPaths() function. Existing paths can be removed by using the removePath() and removePaths() functions. FileSystemWatcher examines each path added to it. Files that have been added to the QFileSystemWatcher can be accessed using the files() function, and directories using the directories() function. The fileChanged() signal is emitted when a file has been modified, renamed or removed from disk. Similarly, the directoryChanged() signal is emitted when a directory or its contents is modified or removed. Note that QFileSystemWatcher stops monitoring files once they have been renamed or removed from disk, and directories once they have been removed from disk.
3. **Android Host App**: FileObserver (android.os.FileObserver) Monitors files (using inotify) to fire an event after files are accessed or changed by by any process on the device (including this one). FileObserver is an abstract class; subclasses must implement the event handler onEvent(int, String). Each FileObserver instance monitors a single file or directory. If a directory is monitored, events will be triggered for all files and subdirectories inside the monitored directory. An event mask is used to specify which changes or actions to report. Event type constants are used to describe the possible changes in the event mask as well as what actually happened in event callbacks.
4. **FMS Server / Controller**: This will be implemented using CodeIgniter framework and php. This will follow MVC architecture and expose REST interface so that data can be updated and synced from multiple and various kinds of operating systems.
5. **Web Viewer**: This will be implemented in HTML 5 and bootstrap which will allow users to view file information.

## 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 information about their external storage device from file management system database.

**Input**

File management will recognize device ID and display external storage devices.

**Processing**

The FMS gathers all devices’ ID and displays all external storage devices.

**Output**

The user can see all external storage devices.

#### BROWSE FILE information

**Introduction**

The user can browse their file information using file management system.

**Input**

The required file name or any index keys

**Processing**

Get the information about the file and notify user that in which Device the file is located.

**Output**

The user can find all the files and folders from memory device.

#### SEARCH FILE

**Introduction**

Users can search file.

**Input**

Insert file name to search it.

**Processing**

Searches the file by file name and give it to the user.

**Output**

Display the information about required file.

#### SEARCH DUPLICATE FILE

**Introduction**

Users can search duplicate file.

**Input**

Insert file name or check the selected file’s duplicate copy.

**Processing**

Searches for the same file name by using the file id.

**Output**

Display all duplicate file lists.

#### DELETE DUPLICATE FILE

**Introduction**

Users can delete duplicate file.

**Input**

Insert file name or check the selected file’s duplicate copy.

**Processing**

Searches for the same file name by using the file id.

**Output**

Ask to delete duplicate copy from the user. Now user can delete the file.

#### SORT FILE information

**Introduction**

Users can sort file information such as by Name, Size, Created time, last modified, Size.

**Input**

Instruction for sorting process according to user preference

**Processing**

Get the sorting instruction and sorting all the files according to the instruction.

**Output**

Get the entire sorted file list.

#### SHARE FILE

**Introduction**

Users can share any particular file in cloud storage or any website.

**Input**

File name, which is to be shared and give the link for the website or cloud storage.

**Processing**

Get the file by its name and shared it to the particular website or cloud storage.

**Output**

Get the file in cloud storage or particular website.

#### BACKUP FILE

**Introduction**

Users can back up their important file by coping in many devices.

**Input**

File name, which is to be backup and give destination of backup.

**Processing**

Get the file and copy it to the particular memory storage device.

**Output**

Get the same important file in multiple storage devices.

#### Stream FILE

**Introduction**

Stream file from any of the connected devices via web interface on demand

**Input**

File name, which is to be stream

**Processing**

Get the file and stream the file via host app.

**Output**

Get the same important file in the web interface.

### 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) , QML , XML
* **IDE:** Visual Studio Express 2012 , Eclipse ,
* **Framework:** Microsoft .NET 4.0
* **Database:** Sqlite.
* **SDK:** Android SDK 8.0.
* **Cloud Technology:** SkyDrive , Google Drive

#### 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

1. This app will not show or store the actual file. It will just display the file information. Then if the user wants to open the file it will use system default application to open the file.
2. This app won’t work in Windows Phone and Apple iPhone as these file system allow third party apps to access file information.
3. This app can copy, store and share file on explicit user request.

# Analysis

## Context Diagram

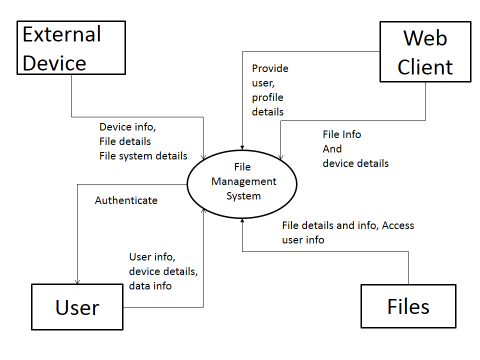


Figure 1:Context Diagram

## Data Flow Diagram

### Level 0 DFD

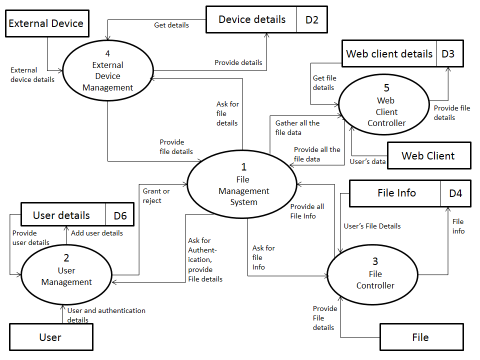


Figure 2:0 level DFD

### Level 1 DFD

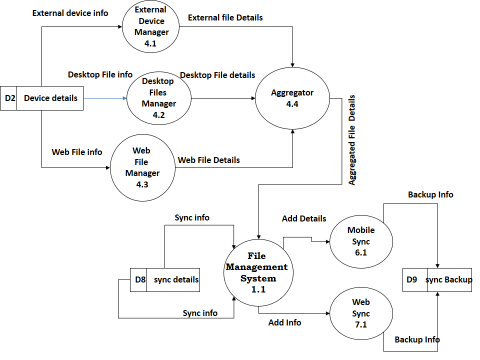


Figure 3:1 level DFD

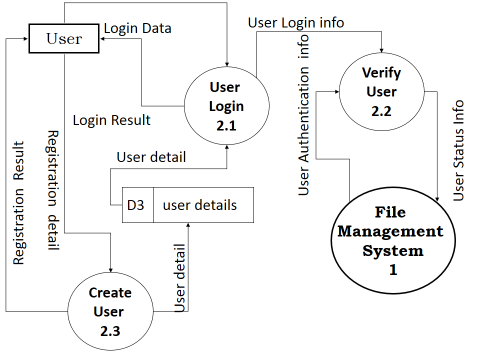


Figure 4:1 level DFD

### Level 2 DFD



Figure 5:2 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

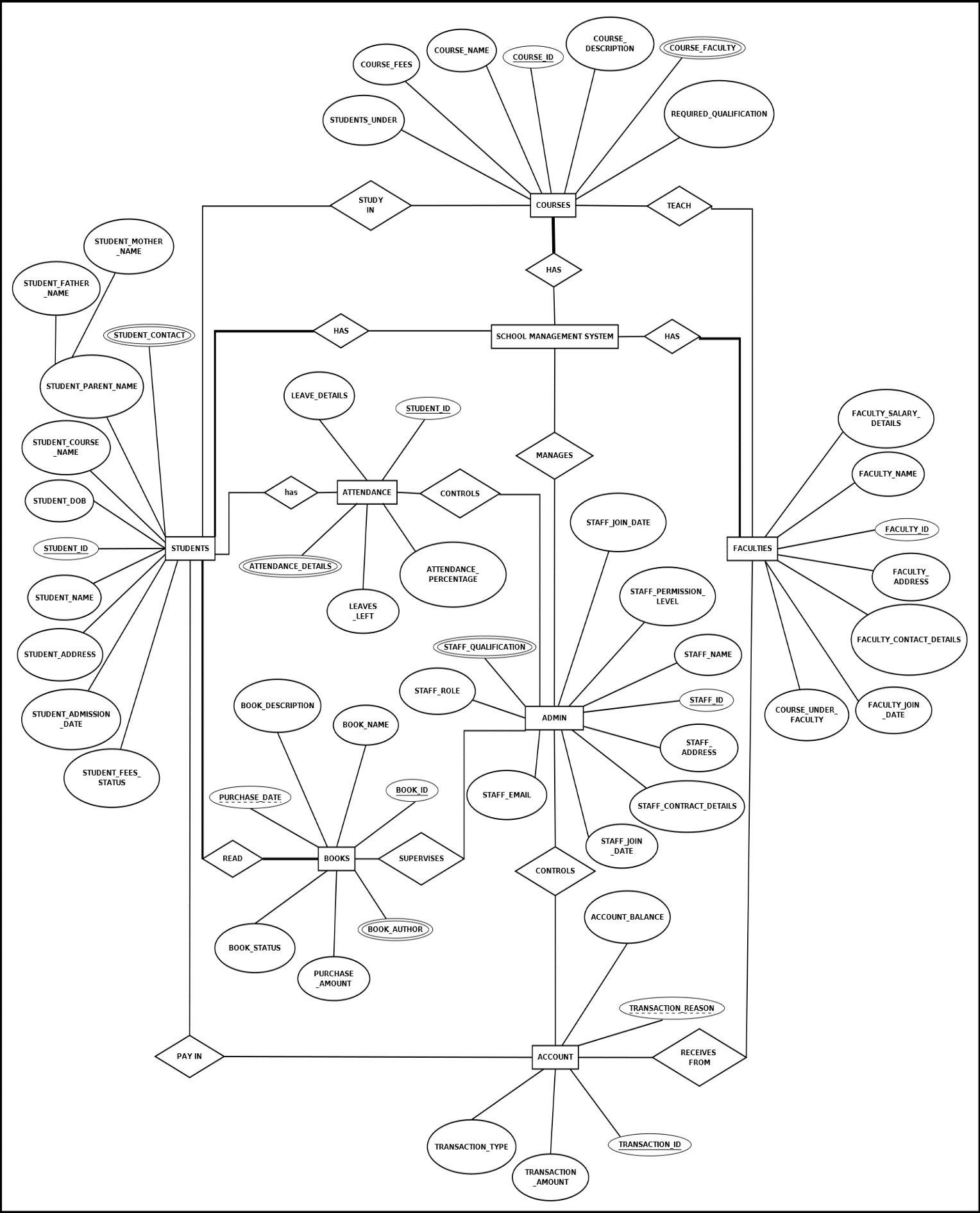
AdminChecksAttendance 1 : 1

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.